Two-view Graph Neural Networks for Knowledge Graph Completion

Vinh Tong
VinAI Research, Vietnam
v.vinhtv4@vinai.io

Dai Quoc Nguyen*
Oracle Labs, Australia
dai.nguyen@oracle.com

Dinh Phung
Monash University, Australia
dinh.phung@monash.edu

Dat Quoc Nguyen
VinAI Research, Vietnam
v.datnq9@vinai.io

ABSTRACT

In this paper, we introduce a novel GNN-based knowledge graph embedding model, named WGE, to capture entity-focused graph structure and relation-focused graph structure. In particular, given the knowledge graph, WGE builds a single undirected entity-focused graph that views entities as nodes. In addition, WGE also constructs another single undirected graph from relation-focused constraints, which views entities and relations as nodes. WGE then proposes a new architecture of utilizing two vanilla GNNs directly on these two single graphs to better update vector representations of entities and relations, followed by a weighted score function to return the triple scores. Experimental results show that WGE obtains state-of-the-art performances on three new and challenging benchmark datasets CoDEx for knowledge graph completion.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Neural networks**.

KEYWORDS

graph neural networks, knowledge graph completion

1 INTRODUCTION

A knowledge graph (KG) is a network of entity nodes and relationship edges, which can be represented as a collection of triples in the form of *(h, r, t)*, wherein each triple *(h, r, t)* represents a relation *r* between a head entity *h* and a tail entity *t*. KGs are often incomplete, i.e., missing a lot of valid triples [3]. Therefore, many embedding models have been proposed to predict whether a triple not in KGs is likely to be valid or not, e.g., TransE [2], DistMult [17], ComplEx [15], and QuatE [18]. These KG embedding models aim to learn vector representations for entities and relations and define a score function such that *valid triples have higher scores than invalid ones*, e.g., the score of the valid triple (New_York, city_of, USA) is higher than the score of the invalid one (New_York, city_of, Australia).

Recently, research works have adapted graph neural networks (GNNs) using an encoder-decoder architecture, e.g., CompGCN [16], SACN [14], and R-GCN [13]. In general, the encoder module aims to modify GNNs to update vector representations of entities and relations. Then, the decoder module employs an existing score function, e.g. DistMult [17] or ConvE [4], to return the triple scores. In particular, R-GCN customizes Graph Convolutional Networks (GCNs) [7] to construct a specific encoder to update only entity embeddings. CompGCN modifies GCNs to use composition operations between entities and relations in the encoder module. Note that these existing GNN-based KG embedding models mainly consider capturing the graph structure around entities; thus, they are still outperformed by conventional models on some benchmarks [9].

To this end, we propose a new KG embedding model, named WGE, to leverage GNNs to capture entity-focused graph structure and relation-focused graph structure for KG completion. In particular, WGE transforms a given KG into two views. The first view—a single undirected entity-focused graph—only includes entities as nodes to provide the entity neighborhood information. The second view—a single undirected relation-focused graph—considers both entities and relations as nodes, constructed from constraints (*subjective relation, predicate entity, objective relation*), to attain the potential dependence between two neighborhood relations. Then WGE introduces a new encoder module of adopting two vanilla GNNs directly on these two graph views to better update entity and relation embeddings, followed by the decoder module using a weighted score function.

Our contributions: (i) We present WGE—equivalent to VVGE to abbreviate Two-View Graph Embedding—that proposes a new encoder architecture of leveraging two vanilla GNNs directly on the entity-focused and relation-focused graph structures, followed by a weighted score function to compute the triple scores. (ii) We conduct extensive experiments to compare our WGE with other strong GNN-based baselines on three new and difficult benchmark datasets CoDEx-S, CoDEx-M, and CoDEx-L [12]. The experiments show that WGE outperforms the baselines and other up-to-date KG embedding models and produces state-of-the-art results on the three challenging datasets for the KG completion task.

2 OUR PROPOSED WGE

A knowledge graph $G = \{V, \mathcal{R}\}$ can be represented as a collection of factual valid triples *(head, relation, tail)* denoted as (h, r, t) with

*This work was done before Dai Quoc Nguyen joined Oracle Labs, Australia.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

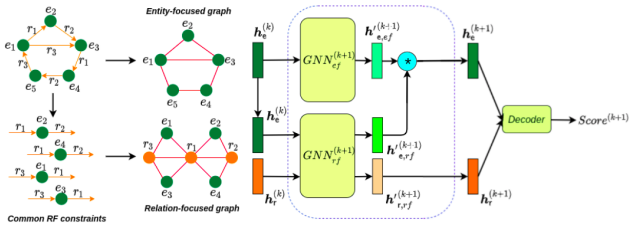

Figure 1: An illustration of our proposed WGE. Here, $\boldsymbol{h}_{\mathrm{e}}^{(k)}$ and $\boldsymbol{h}_{r}^{(k)}$ are the vector representations of the entity e and the relation r at k-th layer of the encoder module.

$h, t \in V$ and $r \in \mathcal{R}$, wherein V is the set of entities, and $\mathcal{R}$ is the set of relations. Recently, the existing GNN-based KG embedding models, such as R-GCN, SACN, and CompGCN, mainly focus on capturing the graph structure around entities. Hence, arguably this could lower their performance as they are still outperformed by other conventional models on some benchmarks [9].

To enhance capturing the graph structure, as illustrated in Figure 1, we introduce WGE as follows: (i) WGE transforms a given knowledge graph into two views: a single undirected entity-focused graph and a single undirected relation-focused graph. (ii) WGE introduces a new encoder architecture of leveraging two vanilla GNNs directly on these two single graphs to update vector representations for entities and relations. (iii) WGE then utilizes a weighted score function as the decoder module to return the triple scores.

2.1 Two-view construction

Entity-focused view. WGE aims to obtain the entity neighborhood information. Thus, given a knowledge graph G, WGE constructs a single undirected graph $\mathcal{G}_{ef}$ viewing entities as individual nodes, as illustrated in Figure 1. Here, $\mathcal{G}_{ef} = \{\mathcal{V}_{ef}, \mathcal{E}_{ef}\}$, wherein $\mathcal{V}_{ef}$ is the set of nodes and $\mathcal{E}_{ef}$ is the set of edges. The number of nodes in $\mathcal{G}_{ef}$ is equal to the number of entities in G, i.e., $|\mathcal{V}_{ef}| = |V|$. In particular, for each triple (h, r, t) in G, entities h and t become individual nodes in $\mathcal{G}_{ef}$ with an edge between them. $\mathcal{G}_{ef}$ is associated with an adjacency matrix $\boldsymbol{A}_{ef}$:

$$\boldsymbol{A}_{ef}(\mathrm{v}, \mathrm{u}) = \begin{cases} 1 & \text{if there is an edge between entity nodes v and u} \\ 0 & \text{otherwise} \end{cases}$$

Relation-focused view. WGE also aims to attain the potential dependence between two neighborhood relations (e.g. "child_of" and "spouse") to enhance learning representations. To do that, from G, our WGE extracts relation-focused (RF) constraints in the form of (*subjective relation, predicate entity, objective relation*), denoted as (r_s, e_p, r_o), wherein e_p is the tail entity for the relation r_s and also the head entity for the relation r_o, e.g. (born_in, New_York, city_of). Here, WGE keeps a certain fraction β of common RF constraints based on ranking how often two relations r_s and r_o co-appear in all extracted RF ones. Then, WGE transforms those obtained common RF constraints into a single undirected relation-focused graph $\mathcal{G}_{rf} = \{\mathcal{V}_{rf}, \mathcal{E}_{rf}\}$ viewing both entities and relations as individual nodes, wherein $\mathcal{V}_{rf}$ is the set of entity and relation nodes, $\mathcal{E}_{rf}$ is the set of edges. For example, as shown in Figure 1, given an RF constraint (r_1, e_2, r_2), WGE considers r_1, e_2, and r_2 as individual nodes in $\mathcal{G}_{rf}$ with edges among them. $\mathcal{G}_{rf}$ is associated with an adjacency matrix $\boldsymbol{A}_{rf}$:

$$\boldsymbol{A}_{rf}(\mathrm{v}, \mathrm{u}) = \begin{cases} 1 & \text{if there is an edge between nodes v and u} \\ 0 & \text{otherwise} \end{cases}$$

2.2 Encoder module

Given a single graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we can adopt vanilla GNNs directly on $\mathcal{G}$ and its adjacency matrix $\boldsymbol{A}$. Recently, QGNN—Quaternion Graph Neural Networks [10]—has been proposed to learn quaternion node embeddings as:

$$\boldsymbol{h}_{\mathrm{v}}^{(k+1),Q} = \mathrm{g}\left(\sum_{\mathrm{u} \in \mathcal{N}_{\mathrm{v}} \cup \{\mathrm{v}\}} \mathrm{a}_{\mathrm{v,u}} \boldsymbol{W}^{(k),Q} \otimes \boldsymbol{h}_{\mathrm{u}}^{(k),Q} \right)$$

where the superscript ^{Q} denotes the Quaternion space; k is the layer index; $\mathcal{N}_{\mathrm{v}}$ is the set of neighbors of node v; $\boldsymbol{W}^{(k),Q}$ is a quaternion weight matrix; $\otimes$ denotes the Hamilton product; and g can be a nonlinear activation function such as tanh; $\boldsymbol{h}_{\mathrm{u}}^{(0),Q} \in \mathbb{H}^n$ is an input embedding vector for node u, which is initialized and learned during training; and $\mathrm{a}_{\mathrm{v,u}}$ is an edge constant between nodes v and u in the Laplacian re-normalized adjacency matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ with $\tilde{\mathbf{A}} = \boldsymbol{A} + \mathbf{I}$, where $\boldsymbol{A}$ is the adjacency matrix, $\mathbf{I}$ is the identity matrix, and $\tilde{\mathbf{D}}$ is the diagonal node degree matrix of $\tilde{\mathbf{A}}$. QGNN performs better than GCNs on some downstream tasks such as graph classification and node classification.

Our WGE thus proposes a new QGNN-based encoder architecture of leveraging two different QGNNs on $\mathcal{G}_{ef}$ and $\mathcal{G}_{rf}$ respectively. This new encoder aims to capture entity-focused graph structure and relation-focused graph structure to better update vector representations for entities and relations as follows:

$$\boldsymbol{h'}_{\mathrm{v},ef}^{(k+1),Q} = \mathrm{g}\left(\sum_{\mathrm{u} \in \mathcal{N}_{\mathrm{v}} \cup \{\mathrm{v}\}} \mathrm{a}_{\mathrm{v,u},ef} \boldsymbol{W}_{ef}^{(k),Q} \otimes \boldsymbol{h}_{\mathrm{u},ef}^{(k),Q} \right) \tag{1}$$

where the subscript $_{ef}$ denotes for QGNN on the entity-focused graph $\mathcal{G}_{ef}$, and we define $\boldsymbol{h}_{\mathrm{u},ef}^{(k),Q}$ as:

$$\boldsymbol{h}_{\mathrm{u},ef}^{(k),Q} = \boldsymbol{h'}_{\mathrm{u},ef}^{(k),Q} * \boldsymbol{h'}_{\mathrm{u},rf}^{(k),Q} \tag{2}$$

where $*$ denotes a quaternion element-wise product, and:

$$\boldsymbol{h'}_{\mathrm{v},rf}^{(k+1),Q} = \mathrm{g}\left(\sum_{\mathrm{u} \in \mathcal{N}_{\mathrm{v}} \cup \{\mathrm{v}\}} \mathrm{a}_{\mathrm{v,u},rf} \boldsymbol{W}_{rf}^{(k),Q} \otimes \boldsymbol{h}_{\mathrm{u},rf}^{(k),Q} \right) \tag{3}$$

where the subscript $_{rf}$ denotes for QGNN on the relation-focused graph $\mathcal{G}_{rf}$. Here, we define $\boldsymbol{h}_{\mathrm{u},rf}^{(k),Q}$ as:

$$\boldsymbol{h}_{\mathrm{u},rf}^{(k),Q} = \begin{cases} \boldsymbol{h}_{\mathrm{u},ef}^{(k),Q} & \text{if u is an entity node, as in Eqn. 2} \\ \boldsymbol{h'}_{\mathrm{u},rf}^{(k),Q} & \text{if u is a relation node, following Eqn. 3} \end{cases} \tag{4}$$

WGE considers $\boldsymbol{h}_{\mathrm{v},rf}^{(k+1),Q}$ as computed following Equation 4 as the vector representation for node v at the $(k+1)$-th hidden layer of the new encoder module:

$$\boldsymbol{h}_{\mathrm{v},rf}^{(k+1),Q} = \begin{cases} \boldsymbol{h}_{\mathrm{v},ef}^{(k+1),Q} & \text{if v is an entity node, following Eqn. 2} \\ \boldsymbol{h'}_{\mathrm{v},rf}^{(k+1),Q} & \text{if v is a relation node, as in Eqn. 3} \end{cases} \tag{5}$$

2.3 Decoder module

WGE employs QuatE [18] across all hidden layers of the encoder module to return a weighted sum of scores as a final score for each triple (h, r, t) as follows:

$$\begin{aligned} f_k(h,r,t) &= \left(\boldsymbol{h}_h^{(k),Q} \otimes \boldsymbol{h}_r^{\triangleleft,(k),Q}\right) \bullet \boldsymbol{h}_t^{(k),Q} \\ f(h,r,t) &= \sum_k \alpha_k f_k(h,r,t) \end{aligned} \tag{6}$$

where $\alpha_k \in [0,1]$ is a fixed important weight of the k-th layer with $\sum_k \alpha_k = 1$; $\boldsymbol{h}_h^{(k),Q}$, $\boldsymbol{h}_r^{(k),Q}$, and $\boldsymbol{h}_t^{(k),Q}$ are quaternion vectors taken from the k-th layer of the encoder; $\otimes$ denotes the Hamilton product; $^{\triangleleft}$ denotes the normalized quaternion; and $\bullet$ denotes the quaternion-inner product.

2.4 Objective function

We train WGE by using Adam [6] to optimize a weighted loss function as:

$$\begin{aligned} \mathcal{L} = -\sum_{(h,r,t)\in\{G\cup G'\}} \sum_k \alpha_k \Big(& l_{(h,r,t)} \log(p_{k,(h,r,t)}) + \\ & (1 - l_{(h,r,t)}) \log(1 - p_{k,(h,r,t)}) \Big) \end{aligned} \tag{7}$$

where $p_{k,(h,r,t)} = \text{sigmoid}\,(f_k(h,r,t))$; $l_{(h,r,t)} = 1$ for $(h,r,t) \in G$; $l_{(h,r,t)} = 0$ for $(h,r,t) \in G'$. Here, G and G' are collections of valid and invalid triples, respectively.

3 EXPERIMENTAL SETUP RESULTS

We evaluate our proposed WGE for the knowledge graph completion task, i.e., link prediction [2], which aims to predict a missing entity given a relation with another entity, e.g., predicting a head entity h given $(?, r, t)$ or predicting a tail entity t given $(h, r, ?)$. The results are calculated by ranking the scores produced by the score function f on triples in the test set.

3.1 Setup

3.1.1 Datasets. Safavi and Koutra [12] show that there are several issues with existing KG completion datasets. Hence, they introduce three new KG completion benchmarks, namely CoDEx-S, CoDEx-M, and CoDEx-L.[1] Those datasets are considered to be more difficult than existing datasets. In addition, compared to the existing ones, CoDEx covers more diverse and interpretable content.

3.1.2 Evaluation protocol. To generate corrupted triples for each test triple (h, r, t), we replace either h or t by each of all other entities following [2]. We also apply the "Filtered" setting protocol [2]. We then rank the valid test triple as well as the corrupted triples in descending order of their scores. Base on this, we report mean reciprocal rank (MRR) and Hits@10 [2]. The final scores on the test set are reported for the model that obtains the highest MRR on the validation set.

3.1.3 Our model's training protocol. We implement our model using Pytorch [11]. We apply the standard Glorot initialization [5] for parameter initialization. We employ tanh for the nonlinear activation function g. We use the Adam optimizer [6] to train our WGE model up to 2000 epochs on all datasets. We use a grid search to choose the number K of hidden layers $\in \{1, 2, 3\}$, the Adam initial learning rate $\in \{1e^{-4}, 5e^{-4}, 1e^{-3}, 5e^{-3}, 1e^{-2}\}$, the batch size $\in \{1024, 2048, 4096\}$, and the input dimension and hidden sizes of the QGNN hidden layers $\in \{32, 64, 128\}$. For the embedding layer, we perform a grid search to select its mixture weight value $\alpha_0 \in \{0.3, 0.6, 0.9\}$. The mixture weight values for the K layers are fixed at $\alpha_k = \frac{1-\alpha_0}{K}$. For the percentage β of kept RF constraints, we use grid search in $\{0.1, 0.2, ..., 1\}$ for CoDEx-S dataset and the best value is 0.2, we use this best value of β for both CoDEx-M and CoDEx-L. To select the best model checkpoints, we evaluate the MRR scores after each training epoch on the validation set.

3.1.4 Baselines' training protocol. For other baseline models, we apply the same evaluation protocol. The training protocol is the same w.r.t. the optimizer, the hidden layers, the initial learning rate values, and the number of training epochs.

In addition, we use the following model-specific configuration for each baseline: **QuatE** [18]: We use grid search to choose batch size $\in \{1024, 2048, 4096\}$ and vary the embedding dimension in {64, 128, 256, 512}. Regarding the GNN-based baseline models – **R-GCN** [13], **CompGCN** [16], **SACN** [14], and **our five WGE variants** – we also set the same dimension value for both the embedding size and the hidden size, wherein we vary the dimension value in {64, 128, 256, 512}. **CompGCN**: We consider a CompGCN variant that set ConvE [4] as its decoder module, circular-correlation as its composition operator, the kernel size to 7, and the number of output channels to 200, producing the best results as reported in the original implementation. **SACN**: For its decoder Conv-TransE, we set the kernel size to 5 and the number of output channels 200 as used in the original implementation.

3.2 Main results

Table 1 shows our experimental results obtained for WGE and other strong baselines on the experimental datasets CoDEx. We find that R-GCN produces lower results than all other models. CompGCN is outperformed by ConvE that is employed as the decoder module in CompGCN, on CoDEx-S and CoDEx-M. Besides, QuatE is not as good as ComplEx, ConvE, and TuckER.

In general, WGE obtains state-of-the-art MRR and Hits@10 scores (except the second-best MRR on CoDEx-S). In particular, WGE gains a substantial gap compared to QuatE and CompGCN, e.g., WGE achieves absolute Hits@10 improvements of 3.8%, 4.7%, and 9.1% over QuatE, and 4.2%, 2.8%, and 1.5% over CompGCN on CoDEx-S, CoDEx-M, and CoDEx-L, respectively.

Ablation analysis. Table 2 presents our ablation results on the validation sets for five variants of our WGE, including: (i) **WGE variant with GCN**: This is a variant of our proposed method that

[1] https://github.com/tsafavi/codex

Table 1: Experimental results on the CoDEx *test* sets. Hits@10 (H@10) is reported in %. The best scores are in bold, while the second best scores are in underline. The results of TransE [2], ComplEx [15], ConvE [4], and TuckER [1] are taken from [12]. ⋆ denotes our own results for the baseline models. We get an out-of-memory for SACN on the large dataset CoDEx-L.

Method	CoDEx-S		CoDEx-M		CoDEx-L	
	MRR	H@10	MRR	H@10	MRR	H@10
TransE	0.354	63.4	0.303	45.4	0.187	31.7
ComplEx	**0.465**	64.6	0.337	47.6	0.294	40.0
ConvE	0.444	63.5	0.318	46.4	0.303	42.0
TuckER	0.444	63.8	0.328	45.8	0.309	43.0
QuatE⋆	0.449	64.4	0.323	48.0	0.312	44.3
R-GCN⋆	0.275	53.3	0.124	24.1	0.073	14.2
SACN⋆	0.374	59.4	0.294	44.3	–	–
CompGCN⋆	0.395	62.1	0.312	45.7	0.304	42.8
WGE	0.450	**66.3**	**0.338**	**48.5**	**0.320**	**44.5**

Table 2: Experimental results on the *validation* sets for five variants of our proposed WGE.

Method	CoDEx-S		CoDEx-M		CoDEx-L	
	MRR	H@10	MRR	H@10	MRR	H@10
R-GCN⋆	0.287	54.7	0.122	23.8	0.073	14.1
SACN⋆	0.377	62.3	0.294	44.0	–	–
CompGCN⋆	0.400	62.9	0.305	45.3	0.303	42.6
WGE	**0.467**	**67.7**	**0.339**	**48.4**	**0.320**	**44.1**
(i) w/ GCN	0.441	66.5	0.322	47.0	0.306	43.0
(ii) w/ only enity-focused	0.452	66.9	0.329	46.5	0.314	43.0
(iii) w/ only relation-focused	0.455	66.9	0.323	46.7	0.305	42.9
(iv) w/ only Levi graph	0.447	63.5	0.320	45.7	0.288	41.1
(v) wo/ predicate entities	0.448	67.1	0.328	47.1	0.312	43.1

utilizes GCN in the encoder module. (ii) **WGE variant with entity-focused view**: This is a variant of our proposed WGE using only entity-focused view. (iii) **WGE variant with relation-focused view**: This is a variant of our proposed WGE using only relation-focused view. (iv) **WGE variant with Levi graph**: This is a variant of WGE where a single Levi graph is used as the input of the encoder module. (v) **WGE variant with only relation nodes**: This is a variant of our proposed WGE which only keeps relation nodes in the relation-focused view.

We find that: (i) The scores decrease when we utilize GCNs in the encoder module rather than QGNN. (ii) & (iii) The scores also degrade when either using only the entity-focused view or using only the relation-focused view. It is worth noting that these WGE variants (i), (ii), and (iii) also outperform three GNN-based baselines R-GCN, SACN, and CompGCN.

From the given KG, we also investigate another strategy of constructing a single undirected graph, which can be considered a direct extension of our entity-focused graph view with addition relation nodes, following the Levi graph transformation [8]. (iv) The scores degrade when using only the Levi graph transformation-based view. In particular, this WGE variant (iv) obtains lower scores than the WGE variant (ii), showing that the Levi graph transformation is not as effective as the entity-focused graph transformation.

In addition, (v) we conduct another experiment using only relations when constructing the relation-focused view, i.e., we do not consider the predicate entities as nodes from the extracted RF constraints. This WGE variant (v) is outperformed by WGE, showing that the predicate entities can help to better infer the potential dependence between two neighborhood relations.

4 CONCLUSION

In this paper, we introduce WGE—a novel GNN-based knowledge graph embedding model—to enhance the entity neighborhood information with the potential dependence between two neighborhood relations. In particular, WGE constructs two views from the given knowledge graph: a single undirected entity-focused graph and a single undirected relation-focused graph. Then WGE proposes a new encoder architecture of utilizing two GNNs on these two graph views to update vector representation for entities and relations. After that, WGE employs a weighted score function to compute the triple scores. WGE outperforms other strong GNN-based baselines and other up-to-date KG embedding models and obtains state-of-the-art results on three new and challenging benchmark datasets CoDEx-S, CoDEx-M, and CoDEx-L for knowledge graph completion. Upon acceptance, our WGE implementation is available at: https://anonymous-url.

REFERENCES

[1] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *EMNLP*. 5185–5194.
[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.
[3] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *AAAI*. 301–306.
[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI*. 1811–1818.
[5] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[6] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).
[7] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[8] Friedrich Wilhelm Levi. 1942. *Finite Geometrical Systems: Six Public Lectues Delivered in February, 1940, at the University of Calcutta*. University of Calcutta.
[9] Dat Quoc Nguyen. 2020. A survey of embedding models of entities and relationships for knowledge graph completion. In *TextGraphs*. 1–14.
[10] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2021. Quaternion Graph Neural Networks. In *Asian Conference on Machine Learning*.
[11] Adam Paszke, Sam Gross, Francisco Massa, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
[12] Tara Safavi and Danai Koutra. 2020. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *EMNLP*. 8328–8350.
[13] Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*. 593–607.
[14] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*, Vol. 33. 3060–3067.
[15] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. 2071–2080.
[16] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.
[17] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*.
[18] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion Knowledge Graph Embeddings. In *NeurIPS*. 2731–2741.