# Convolutional Hypercomplex Embeddings for Link Prediction

**Caglar Demir** CAGLAR.DEMIR@UPB.DE
*Data Science Research Group, Paderborn University*
**Diego Moussallem** DIEGO.MOUSSALLEM@UPB.DE
*Data Science Research Group, Paderborn University*
*Globo, Rio de Janeiro, Brazil*
**Stefan Heindorf** STEFAN.HEINDORF@UPB.DE
*Data Science Research Group, Paderborn University*
**Axel-Cyrille Ngonga Ngomo** AXEL.NGONGA@UPB.DE
*Data Science Research Group, Paderborn University*

**Editors:** Vineeth N Balasubramanian and Ivor Tsang

## Abstract

Knowledge graph embedding research has mainly focused on the two smallest normed division algebras, $\mathbb{R}$ and $\mathbb{C}$. Recent results suggest that trilinear products of quaternion-valued embeddings can be a more effective means to tackle link prediction. In addition, models based on convolutions on real-valued embeddings often yield state-of-the-art results for link prediction. In this paper, we investigate a composition of convolution operations with hypercomplex multiplications. We propose the four approaches QMULT, OMULT, CONVQ and CONVO to tackle the link prediction problem. QMULT and OMULT can be considered as quaternion and octonion extensions of previous state-of-the-art approaches, including DistMult and ComplEx. CONVQ and CONVO build upon QMULT and OMULT by including convolution operations in a way inspired by the residual learning framework. We evaluated our approaches on seven link prediction datasets including WN18RR, FB15K-237, and YAGO3-10. Experimental results suggest that the benefits of learning hypercomplex-valued vector representations become more apparent as the size and complexity of the knowledge graph grows. CONVO outperforms state-of-the-art approaches on FB15K-237 in MRR, Hit@1 and Hit@3, while QMULT, OMULT, CONVQ and CONVO outperform state-of-the-approaches on YAGO3-10 in all metrics. Results also suggest that link prediction performance can be further improved via prediction averaging. To foster reproducible research, we provide an open-source implementation of approaches, including training and evaluation scripts as well as pretrained models.[1]

**Keywords:** Convolution, Hypercomplex Embeddings, Link Prediction, Residual Learning

## 1. Introduction

Knowledge graphs represent structured collections of facts describing the world in the form of typed relationships between entities (Hogan et al., 2020). These collections of facts have been used in a wide range of applications, including web search, question answering, and recommender systems (Nickel et al., 2015). However, most knowledge graphs on the web

1. https://github.com/dice-group/Convolutional-Hypercomplex-Embeddings-for-Link-Prediction

are far from complete. The task of identifying missing links in knowledge graphs is referred to as *link prediction*. Knowledge Graph Embedding (KGE) models have been particularly successful at tackling the link prediction task, among many others (Nickel et al., 2015).

KGE research has mainly focused on the two smallest normed division algebras—real numbers ($\mathbb{R}$) and complex numbers ($\mathbb{C}$)—neglecting the benefits of the larger normed division algebras—quaternions ($\mathbb{H}$) and octonions ($\mathbb{O}$). While Yang et al. (2015) introduced the trilinear product of *real*-valued embeddings $\langle \mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \rangle$ of triples ($\mathtt{h}$, $\mathtt{r}$, $\mathtt{t}$) as a scoring function for link prediction, Trouillon et al. (2016) showed the usefulness of the Hermitian product of *complex*-valued embeddings $\mathrm{Re}(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle)$. In contrast to the trilinear product of *real*-valued embeddings, the Hermitian product is not symmetric and can be used to model antisymmetric relations since $\mathrm{Re}(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}_t} \rangle) \neq \mathrm{Re}(\langle \mathbf{e}_t, \mathbf{e}_r, \overline{\mathbf{e}_h} \rangle)$. To further increase the expressivity, Zhang et al. (2019) proposed learning *quaternion*-valued embeddings due to their benefits over *complex*-valued embeddings. Zhang et al. (2021) show that replacing a fully connected layer with a hypercomplex multiplication layer in a neural network leads to significant parameter efficiency without degenerating the predictive performance in many tasks.

Dettmers et al. (2018), Balažević et al. (2019a), and Demir and Ngomo (2021a) showed that convolutions are another effective means to increase the expressivity: the sparse connectivity property of the convolution operator endows models with parameter efficiency—unlike models simply increasing the embedding size which is not scalable to large knowledge graphs (Dettmers et al., 2018). Different configurations of the number of feature maps and the shape of kernels (1D or 2D) in the convolution operation are often explored to find the best ratio between expressiveness and parameter space size.

We investigate the use of convolutions on hypercomplex embeddings by proposing four models: QMult and OMult can be considered hypercomplex extensions of DistMult (Yang et al., 2015) in $\mathbb{H}$ and $\mathbb{O}$, respectively. In contrast to the state of the art (Zhang et al., 2019), we address the scaling effect of multiplication in $\mathbb{H}$ and $\mathbb{O}$ by applying the batch normalization technique. Through the batch normalization technique, QMult and OMult are allowed to control the rate of normalization and benefit from its implicit regularization effect (Ioffe and Szegedy, 2015). Importantly, Lu and Hu (2020) suggest that using solely unit quaternion-based rotations between head entity and relation limits the modeling capacity for various types of relations. ConvQ and ConvO build upon QMult and OMult by including the convolution operator that is partially inspired by the residual learning framework (He et al., 2016). ConvQ and ConvO forge QMult and OMult with a 2D convolution operation and an affine transformation via the Hadamard product, respectively. By virtue of this architecture, we show that ConvQ can degenerate QMult, ComplEx or DistMult, if such degeneration is necessary to further minimize the training loss (see Equations (6) and (10)). Experiments suggest that our models often achieve state-of-the-art performance on seven benchmark datasets (WN18, FB15K, WN18RR, FB15K-237, YAGO3-10, Kinship and UMLS). Superiority of our models against state-of-the-art models increases as the size and complexity of the knowledge graph grow. Our results also indicate that generalization performance of models can be further increased by applying ensemble learning.

## 2. Related Work

In the last decade, a plethora of KGE approaches have been successfully applied to tackle various tasks (Nickel et al., 2015). In this section, we give a brief chronological overview of selected KGE approaches. RESCAL computes a three-way factorization of a third-order adjacency tensor representing the input knowledge graph to compute scores for triples (Nickel et al., 2011). RESCAL captures various types of relations in the input Knowledge Graph (KG) but is limited in its scalability as it has quadratic complexity in the factorization rank (Trouillon et al., 2017). DistMult can be regarded as an efficient extension of RESCAL with a diagonal matrix per relation to reduce the complexity of RESCAL (Yang et al., 2015). DistMult performs poorly on antisymmetric relations, whereas it performs well on symmetric relations (Trouillon et al., 2016). ComplEx extends DistMult by learning representations in a complex vector space (Trouillon et al., 2016). ComplEx is able to infer both symmetric and antisymmetric relations via a Hermitian inner product of embeddings that involves the conjugate-transpose of one of the two input vectors. Lacroix et al. (2018) designed two novel regularizers along with a data augmentation technique and propose ComplEx-N3 that can be seen as ComplEx with the N3 regularization. ConvE applies a 2D convolution operation to model the interactions between entities and relations (Dettmers et al., 2018). ConvKB extends ConvE by omitting the reshaping operation in the encoding of representations in the convolution operation (Nguyen et al., 2017). Similarly, HypER extends ConvE by applying relation-specific 1D convolutions as opposed to applying filters from concatenated subject and relation vectors (Balažević et al., 2019a). TuckER employs the Tucker decomposition on the binary tensor representing the input knowledge graph triples (Balažević et al., 2019b). RotatE employs a rotational model taking predicates as rotations from subjects to objects in complex space via the element-wise Hadamard product (Sun et al., 2019). By these means, RotatE performs well on composition relations where other approaches perform poorly. QuatE applies the quaternion multiplication followed by an inner product to compute scores of triples (Zhang et al., 2019).

Nickel and Kiela (2017), Balazevic et al. (2019), and Demir and Ngomo (2019) propose learning embeddings of input KG such that distances between embeddings of entities and relations reflect their semantic similarity.

## 3. Link Prediction & Hypercomplex Numbers

**Link Prediction.** Let $\mathcal{E}$ and $\mathcal{R}$ represent the sets of entities and relations. Then, a KG can be formalised as a set of triples $\mathcal{G} = \{(\mathtt{h}, \mathtt{r}, \mathtt{t})\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where each triple contains two entities $\mathtt{h}, \mathtt{t} \in \mathcal{E}$ and a relation $\mathtt{r} \in \mathcal{R}$. The link prediction problem is formalised by learning a scoring function $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ ideally characterized by $\phi(\mathtt{h}, \mathtt{r}, \mathtt{t}) > \phi(\mathtt{x}, \mathtt{y}, \mathtt{z})$ if $(\mathtt{h}, \mathtt{r}, \mathtt{t})$ is true and $(\mathtt{x}, \mathtt{y}, \mathtt{z})$ is not Dettmers et al. (2018); Demir et al. (2021).

**Hypercomplex Numbers.** The quaternions are a 4-dimensional normed division algebra (Hamilton, 1844). A quaternion number $Q \in \mathbb{H}$ is defined as $Q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ where $a, b, c, d$ are real numbers and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units satisfying Hamilton's rule: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. The inner product of two quaternions is defined as

$$Q_1 \cdot Q_2 = \langle a_1 a_2 \rangle + \langle b_1 b_2 \rangle + \langle c_1 c_2 \rangle + \langle d_1 d_2 \rangle.$$

The quaternion multiplication of $Q_1$ and $Q_2$ is defined as

$$\begin{aligned} Q_1 \otimes Q_2 = &(a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2) \\ &+ (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2)\ \mathbf{i} \\ &+ (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2)\ \mathbf{j} \\ &+ (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2)\ \mathbf{k}. \end{aligned}$$

The quaternion multiplication is also known as the Hamilton product (Zhang et al., 2021). For a $d$-dimensional quaternion vector a + b $\mathbf{i}$ + c $\mathbf{j}$ + d $\mathbf{k}$ with $a, b, c, d \in \mathbb{R}^d$, the inner product and multiplication is defined accordingly. The Octonions are an 8-dimensional algebra where an octonion number $O_1 \in \mathbb{O}$ is defined as $O_1 = x_0 + x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + \ldots + x_7\mathbf{e}_7$, where $\mathbf{e}_1, \mathbf{e}_2 \ldots \mathbf{e}_7$ are imaginary units (Baez, 2002). Their product ($\bigstar$), inner product ($\cdot$) and vector operations are defined analogously to quaternions.

The quaternion multiplication subsumes real-valued multiplication and enjoys a parameter saving with 1/4 as compared to the real-valued matrix multiplication (Parcollet et al., 2018; Zhang et al., 2021). Leveraging such properties of quaternions in neural networks showed promising results in numerous tasks (Zhang et al., 2021, 2019; Chen et al., 2020). However, the octonion multiplication in neural networks and learning octonion-valued knowledge graph embeddings had not yet been fully explored.

## 4. Convolutional Hypercomplex Embeddings

**Motivation.** Dettmers et al. (2018) suggest that indegree and PageRank can be used to quantify the difficulty of predicting missing links in KGs. Results indicate that the superiority of ConvE becomes more apparent against DistMult and ComplEx as the complexity of the knowledge graph increases, i.e., indegree and PageRank of a KG increase (see Table 6 in Dettmers et al. (2018)). In turn, Zhang et al. (2019) show that learning quaternion-valued embeddings via multiplicative interactions can be a more effective means of predicting missing links than learning real and complex-valued embeddings. Although learning quaternion-valued embeddings through multiplicative interactions yields promising results, the only way to further increase the expressiveness of such models is to increase the number of dimensions of embeddings. This does not scale to larger knowledge graphs (Dettmers et al., 2018). Increasing parameter efficiency while retaining effectiveness is a desired property in many applications (Zhang et al., 2021; Trouillon et al., 2016, 2017).

Motivated by findings of aforementioned works, we investigate the composition of convolution operations with hypercomplex multiplications. The rationale behind this composition is to increase the expressiveness without increasing the number of parameters. This nontrivial endeavor is the keystone of embedding models (Trouillon et al., 2016). The sparse connectivity property of the convolution operation endows models with parameter efficiency, which helps to scale to larger knowledge graphs. Additionally, different configurations of the number of kernels and their shapes can be explored to find the best ratio between expressiveness and the number of parameters. Although increasing the number of feature maps results in increasing the number of parameters, we are able to benefit from the parameter sharing property of convolutions (Goodfellow et al., 2016).

**Approaches.** Inspired by the early works DistMult and ConvE, we dub our approaches QMult, OMult, ConvQ, and ConvO, where "Q" represents the quaternion variant and "O" the octonion variant. Given a triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, QMult : $\mathbb{H} \rightarrow \mathbb{R}$ computes a triple score through the quaternion multiplication of head entity embeddings $\mathbf{e}_h$ and relation embeddings $\mathbf{e}_r$ followed by the inner product with tail entity embeddings $\mathbf{e}_t$ as

$$\text{QMult}(h, r, t) = \mathbf{e}_h \otimes \mathbf{e}_r \cdot \mathbf{e}_t, \tag{1}$$

where $\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \in \mathbb{H}^d$. Similarly, OMult : $\mathbb{O} \rightarrow \mathbb{R}$ performs the octonion multiplication followed by the inner product as

$$\text{OMult}(h, r, t) = \mathbf{e}_h \bigstar \mathbf{e}_r \cdot \mathbf{e}_t, \tag{2}$$

where $\mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \in \mathbb{O}^d$. Computing scores of triples in this setting can be illustrated in two consecutive steps: (1) rotating $\mathbf{e}_h$ through $\mathbf{e}_r$ by applying quaternion/octonion multiplication, and (2) squishing $(\mathbf{e}_h \otimes \mathbf{e}_r)$ and $\mathbf{e}_t$ into a real number line by taking the inner product. During training, the degree between $(\mathbf{e}_h \otimes \mathbf{e}_r)$ and $\mathbf{e}_t$ is minimized provided $(\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}$.

Motivated by the response of John T. Graves to W. R. Hamilton,[2] we combine 2D convolutions with QMult and OMult as defined

$$\text{ConvQ}(h, r, t) = \text{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \otimes \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{3}$$
$$\text{ConvO}(h, r, t) = \text{conv}(\mathbf{e}_h, \mathbf{e}_r) \circ (\mathbf{e}_h \bigstar \mathbf{e}_r) \cdot \mathbf{e}_t, \tag{4}$$

where $\text{conv}(\cdot, \cdot) : \mathbb{H}^{2d} \rightarrow \mathbb{H}^d$ (respectively : $\mathbb{O}^{2d} \rightarrow \mathbb{O}^d$) is defined as

$$\text{conv}(\mathbf{e}_h, \mathbf{e}_r) = f\big(\text{vec}(f([\mathbf{e}_h, \mathbf{e}_r] * \omega)) \cdot \mathbf{W} + \mathbf{b}\big). \tag{5}$$

$f(\cdot), \text{vec}(\cdot), *, \omega$, $[\cdot, \cdot]$, and $(\mathbf{W}, \mathbf{b})$ denote the rectified linear unit function, a flattening operation, convolution operation, kernel in the convolution, stacking operation, and an affine transformation, respectively.

**Connection to ComplEx and DistMult.** During training, $\text{conv}(\cdot, \cdot)$ can reduce its range into $\gamma \in 1$ if such reduction is necessary to further decrease the training loss. In the following Equations (6) to (10), we elucidate the reduction of ConvQ into QMult and ComplEx:

$$\text{ConvQ}(\mathtt{h}, \mathtt{r}, \mathtt{t}) = \gamma \circ (\mathbf{e}_h \otimes \mathbf{e}_r) \cdot \mathbf{e}_t. \tag{6}$$

Equation (6) corresponds to QMult provided that $\text{conv}(\mathbf{e}_h, \mathbf{e}_r) = \gamma = 1$. ConvQ can be further reduced into ComplEx by setting the imaginary parts $\mathbf{j}$ and $\mathbf{k}$ of $\mathbf{e}_h$, $\mathbf{e}_r$ and $\mathbf{e}_t$ to zero:

$$\gamma \circ \big((a_h + b_h\mathbf{i}) \otimes (a_r + b_r\mathbf{i})\big) \cdot (a_t + b_t\mathbf{i}). \tag{7}$$

Computing the quaternion multiplication of two quaternion-valued vectors corresponds to Equation (8):

$$\gamma \circ [(a_h \circ a_r - b_h \circ b_r) + (a_h \circ b_r + b_h \circ a_r)\mathbf{i}] \cdot (a_t + b_t\mathbf{i}). \tag{8}$$

2. "If with your alchemy you can make three pounds of gold, why should you stop there?" Baez (2002).

The resulting quaternion-valued vector is scaled with $\gamma = [\gamma_1, \gamma_2]$:

$$[(\gamma_1 \circ a_h \circ a_r - \gamma_1 \circ b_h \circ b_r) + (\gamma_2 \circ a_h \circ b_r + \gamma_2 \circ b_h \circ a_r)\mathbf{i}] \cdot (a_t + b_t\mathbf{i}). \tag{9}$$

Through taking the inner product of the former vector with $(a_t + b_t\mathbf{i})$, we obtain

$$\begin{aligned} \textsc{ConvQ}(\mathtt{h}, \mathtt{r}, \mathtt{t}) = &\langle \gamma_1, a_h, a_r, a_t \rangle \\ &+\langle \gamma_2, b_h, a_r, b_t \rangle \\ &+\langle \gamma_2, a_h, b_r, b_t \rangle \\ &-\langle \gamma_1, b_h, b_r, a_t \rangle, \end{aligned} \tag{10}$$

where $\langle a, b, c, d \rangle = \sum_k a_k b_k c_k d_k$ corresponds to the multi-linear inner product. Equation (10) corresponds to ComplEx provided that $\gamma = 1$. In the same way, ConvQ can be reduced into DistMult by setting all imaginary parts $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ to zero for $\mathbf{e}_h$, $\mathbf{e}_r$, and $\mathbf{e}_t$ yielding

$$\textsc{ConvQ}(\mathtt{h}, \mathtt{r}, \mathtt{t}) = \langle \gamma_1, a_h, a_r, a_t \rangle. \tag{11}$$

**Connection to residual learning.** The residual learning framework facilitates the training of deep neural networks. A simple residual learning block consists of two weight layers denoted by $\mathcal{F}(x)$ and an identity mapping of the input $x$ (see Figure 2 in He et al. (2016)). Increasing the depth of a neural model via stacking residual learning blocks led to significant improvements in many domains. In our setting, $\mathcal{F}(\cdot)$ and $x$ correspond to $\text{conv}(\cdot, \cdot)$ and $[\mathbf{e}_h, \mathbf{e}_r]$, respectively. We replaced the identity mapping of the input with the hypercomplex multiplication. To scale the output, we replaced the elementwise vector addition with the Hadamard product. By virtue of such inclusion, ConvQ and ConvO are endowed with the ability of controlling the impact of $\text{conv}(\cdot, \cdot)$ on predicted scores as shown in Equation (10). Ergo, the gradients of loss (see Equation (12)) w.r.t. head entity and relation embeddings can be propagated in two ways, namely, via $\text{conv}(\mathbf{e}_h, \mathbf{e}_r)$ or hypercomplex multiplication. Moreover, the number of feature maps and the shape of kernels can be used to find the best ratio between expressiveness and the number of parameters. Hence, the expressiveness of models can be adjusted without necessarily increasing the embedding size. Although increasing the number of feature maps results in increasing the number of parameters in the model, we are able to benefit from the parameter sharing property of convolutions.

## 5. Experimental Setup

### 5.1. Datasets

We used seven datasets: WN18RR, FB15K-237, YAGO3-10, FB15K, WN18, UMLS and Kinship. An overview of the datasets is provided in Table 1. The latter four datasets are included for the sake of the completeness of our evaluation. Dettmers et al. (2018) suggest that indegree and PageRank can be used to indicate difficulty of performing link prediction on an input KG. In our experiments, we are particularly interested in link prediction results on complex KGs. As commonly done, we augment the datasets by adding reciprocal triples $(\mathtt{t}, \mathtt{r}^{-1}, \mathtt{h})$ (Dettmers et al., 2018; Balažević et al., 2019a,b). For link prediction based on only tail entity ranking experiments (see Table 5), we omit the data augmentation on the test set, as in (Bansal et al., 2019).

Table 1: Overview of datasets in terms of entities, relations, average node degree plus/minus standard deviation.

| **Dataset** | $\|\mathcal{E}\|$ | $\|\mathcal{R}\|$ | Degree | $\|\mathcal{G}^{\text{Train}}\|$ | $\|\mathcal{G}^{\text{Val.}}\|$ | $\|\mathcal{G}^{\text{Test}}\|$ |
|---|---|---|---|---|---|---|
| YAGO3-10 | 123,182 | 37 | 9.6±8.7 | 1,079,040 | 5,000 | 5,000 |
| FB15K | 14,951 | 1,345 | 32.5±69.5 | 483,142 | 50,000 | 59,071 |
| WN18 | 40,943 | 18 | 3.5±7.7 | 141,442 | 5,000 | 5,000 |
| FB15k-237 | 14,541 | 237 | 19.7±30 | 272,115 | 17,535 | 20,466 |
| WN18RR | 40,943 | 11 | 2.2±3.6 | 86,835 | 3,034 | 3,134 |
| KINSHIP | 104 | 25 | 82.2±3.5 | 8,544 | 1,068 | 1,074 |
| UMLS | 135 | 46 | 38.6±32.5 | 5,216 | 652 | 661 |

### 5.2. Training and optimization

We apply the standard training strategy as Dettmers et al. (2018); Balažević et al. (2019a,b): Following the KvsAll training procedure,[3] for a given pair $(\mathtt{h}, \mathtt{r})$, we compute scores for all $x \in \mathcal{E}$ with $\phi(h, r, x)$ and apply the logistic sigmoid function $\sigma(\phi(h, r, x))$. Models are trained to minimize the binary cross entropy loss function, where $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{E}|}$ and $\mathbf{y} \in [0, 1]^{|\mathcal{E}|}$ denote the predicted scores and binary label vector, respectively.

$$L = -\frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} (\mathbf{y}^{(i)} \log(\hat{\mathbf{y}}^{(i)}) + (1 - \mathbf{y}^{(i)}) \log(1 - \hat{\mathbf{y}}^{(i)})). \tag{12}$$

We employ the Adam optimizer (Kingma and Ba, 2014), dropout (Srivastava et al., 2014), label smoothing and batch normalization (Ioffe and Szegedy, 2015), as in the literature (Balažević et al., 2019a,b; Dettmers et al., 2018; Demir et al., 2021). Moreover, we selected hyperparameters of our approaches by random search based on validation set performances (Balažević et al., 2019b). Notably, we did not search a good random seed for the random number generator, and we fixed the seed to 1 throughout our experiments.

### 5.3. Evaluation

We employ the standard metrics *filtered* Mean Reciprocal Rank (MRR) and hits at N (H@N) for link prediction (Dettmers et al., 2018; Balažević et al., 2019a). For each test triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, we construct its reciprocal $(\mathtt{t}, \mathtt{r}^{-1}, \mathtt{h})$ and add it into $\mathcal{G}^{\text{test}}$, which is a common technique to decrease the computational cost during testing (Dettmers et al., 2018). Then, for each test triple $(\mathtt{h}, \mathtt{r}, \mathtt{t})$, we compute the score of $(\mathtt{h}, \mathtt{r}, \mathtt{x})$ triples for all $x \in \mathcal{E}$ and calculate the filtered ranking $rank_t$ of the triple having $t$. Then we compute the MRR: $\frac{1}{|\mathcal{G}^{\text{test}}|} \sum_{(\mathtt{h},\mathtt{r},\mathtt{t}) \in \mathcal{G}^{\text{test}}} \frac{1}{rank_t}$. Consequently, given a $(\mathtt{h}, \mathtt{r}, \mathtt{t}) \in \mathcal{G}^{\text{test}}$, we compute ranks of missing entities based on *the rank of head and tail entities* analogously to (Dettmers et al., 2018; Balažević et al., 2019b,a). For the sake of completeness, we also report link prediction performances based on *only tail rankings*, i.e., without including triples with reciprocal relations into test data, as in Bansal et al. (2019).

3. Here, we follow the terminology of Ruffinelli et al. (2019).

### 5.4. Implementation Details and Reproducibility

We implemented and evaluated our approach in the framework provided by Balažević et al. (2019b). To alleviate the hardware requirements for the reproducibility, we provide hyperparameter optimization, training and evaluation scripts along with pretrained models at the project page. Experiments were conducted on a single NVIDIA GeForce RTX 3090.

## 6. Results

Table 2 reports link prediction results on the WN18RR, FB15K-237, and YAGO3-10 datasets. Overall, the superior performance of our approaches becomes more and more apparent as the size and complexity of the knowledge graphs grow. On the smallest benchmark dataset (WN18RR), QMult, OMult, ConvQ and ConvO outperform many approaches including DistMult, ConvE and ComplEx in all metrics. However, QuatE, TuckER, and RotatE yield the best performance. On the second-largest benchmark dataset (FB15K-237 is **3.1**× larger than WN18RR), ConvO outperforms all state-of-the-art approaches in 3 out of 4 metrics. Additionally, QMult and ConvQ outperform all state-of-the-art approaches except for TucKER in terms of MRR, H@1 and H@3. On the largest benchmark dataset (YAGO3-10 is **12.4**× larger than WN18RR), QMult, ConvO, ConvQ outperform all approaches in all metrics. Surprisingly, QMult and OMult reach the best and second-best performance in all metrics, whereas ConvO does not perform particularly well compared to our other approaches. ConvO outperforms QMult, OMult, and ConvQ in 8 out of 12 metrics, whereas QMult yields better performance on YAGO3-10. Overall, these results suggest that superiority of learning hypercomplex embeddings becomes more apparent as the size and complexity of the input knowledge graph increases, as measured by indegree (see Table 1) and PageRank (see Table 6 in Dettmers et al. (2018)). In Table 3, we compare some of the best performing approaches on WN18RR, FB15K-237 and YAGO3-10 in terms of the number of trainable parameters. Results indicate that our approaches yield competitive (if not better) performance on all benchmark datasets.

### 6.1. Ensemble Learning

Table 4 reports link prediction results of ensembled models on benchmark datasets. Averaging the predicted scores of models improved the performance by circa 1–2% in MRR. These results suggest that performance may be further improved through optimizing the impact of each model in the ensemble.

### 6.2. Impact of Tail Entity Rankings

During our experiments, we observed that models often perform more accurately in predicting missing tail entities compared to predicting missing head entities, which was also observed in Bansal et al. (2019). Table 5 indicates that MRR performance based on *only tail entity rankings* are on average absolute 10% higher than MRR results based on *head and tail entity rankings* on FB15K-237 while such difference was not observed on WN18RR.

Table 2: Link prediction results on WN18RR, F15K-237 and YAGO3-10. Results are obtained from corresponding papers. Bold and underlined entries denote best and second-best results. The dash (-) denotes values missing in the papers.

| | **WN18RR** | | | | **FB15K-237** | | | | **YAGO3-10** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 |
| TransE (Ruffinelli et al., 2019) | .228 | .053 | .368 | .520 | .313 | .221 | .347 | .497 | - | - | - | - |
| ConvE (Ruffinelli et al., 2019) | .442 | .411 | .451 | .504 | .339 | .248 | .359 | .521 | - | - | - | - |
| TuckER (Balažević et al., 2019b) | .470 | **.443** | .482 | .526 | .358 | .266 | .394 | **.544** | - | - | - | - |
| A2N (Bansal et al., 2019) | .450 | .420 | .460 | .510 | .317 | .232 | .348 | .486 | - | - | - | - |
| QuatE (Zhang et al., 2019) | **.482** | .436 | **.499** | **.572** | .311 | .221 | .342 | .495 | - | - | - | - |
| HypER (Balažević et al., 2019a) | .465 | .436 | .477 | .522 | .341 | .252 | .376 | .520 | .533 | .455 | .580 | .678 |
| DistMult (Dettmers et al., 2018) | .430 | .390 | .440 | .490 | .240 | .160 | .260 | .420 | .340 | .240 | .380 | .540 |
| ConvE (Dettmers et al., 2018) | .430 | .400 | .440 | .520 | .335 | .237 | .356 | .501 | .440 | .350 | .490 | .620 |
| ComplEx (Dettmers et al., 2018) | .440 | .410 | .460 | .510 | .247 | .158 | .275 | .428 | .360 | .260 | .400 | .550 |
| REFE (Chami et al., 2020) | .455 | .419 | .470 | .521 | .302 | .216 | .330 | .474 | .370 | .289 | .403 | .527 |
| ROTE (Chami et al., 2020) | .463 | .426 | .477 | .529 | .307 | .220 | .337 | .482 | .381 | .295 | .417 | .548 |
| ATTE (Chami et al., 2020) | .456 | .419 | .471 | .526 | .311 | .223 | .339 | .488 | .374 | .290 | .410 | .538 |
| ComplEx-N3 (Chami et al., 2020) | .420 | .390 | .420 | .460 | .294 | .211 | .322 | .463 | .336 | .259 | .367 | .484 |
| MuRE (Chami et al., 2020) | .458 | .421 | .471 | .525 | .313 | .226 | .340 | .489 | .283 | .187 | .317 | .478 |
| RotatE (Sun et al., 2019) | .476 | .428 | .492 | .571 | .338 | .241 | .375 | .533 | .495 | .402 | .550 | .670 |
| QMult | .438 | .393 | .449 | .537 | .346 | .252 | .383 | .535 | **.555** | **.475** | **.602** | **.698** |
| OMult | .449 | .406 | .467 | .539 | .347 | .253 | .383 | .534 | .543 | .461 | .592 | .692 |
| ConvQ | .457 | .424 | .470 | .525 | .343 | .251 | .376 | .528 | .539 | .459 | .587 | .687 |
| ConvO | .458 | .427 | .473 | .521 | **.366** | **.271** | **.403** | .543 | .489 | .395 | .546 | .664 |

Table 3: Number of parameter comparisons on the WN18RR, FB15K-237 and YAGO3-10 datasets. The dash (-) denotes values missing in the papers.

| | **WN18RR** | **FB15K-237** | **YAGO3-10** |
|---|---|---|---|
| QuatE (Zhang et al., 2019) | 16.38M | 5.82M | - |
| RotatE (Sun et al., 2019) | 40.95M | 29.32M | 123.22M |
| QMult | 16.38M | 6.01M | 49.30M |
| OMult | 16.38M | 6.01M | 49.30M |
| ConvQ | 21.51M | 11.13M | 54.42M |
| ConvO | 21.51M | 11.13M | 54.42M |

Table 4: Link prediction results via ensembling models.

| | **WN18RR** | | | | **FB15K-237** | | | | **YAGO3-10** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 |
| Q-OMult | .444 | .399 | .458 | .544 | .356 | .260 | .393 | .545 | .557 | .478 | .601 | .700 |
| QMult-ConvQ | .446 | .406 | .455 | .538 | .357 | .263 | .392 | .546 | **.561** | **.483** | **.606** | **.703** |
| QMult-ConvO | .449 | .410 | .459 | .536 | **.372** | .275 | **.411** | .564 | .543 | .460 | .594 | .693 |
| OMult-ConvQ | .444 | .403 | .453 | .537 | .357 | .262 | .391 | .547 | .558 | .478 | .602 | .700 |
| OMult-ConvO | .462 | **.425** | **.475** | **.539** | **.372** | **.277** | **.411** | .564 | .535 | .450 | .588 | .692 |
| ConvQ-O-OMult | **.463** | **.425** | **.475** | **.539** | **.372** | .275 | **.411** | **.567** | .552 | .470 | .599 | .702 |

Table 5: Link prediction results based on only tail entity rankings.

| | **WN18RR** | | | | **FB15K-237** | | | | **YAGO3-10** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 | MRR | @1 | @3 | @10 |
| DistMult | .430 | .410 | .440 | .480 | .370 | .275 | .417 | .568 | - | - | - | - |
| ComplEx | .420 | .380 | .430 | .480 | .394 | .303 | .434 | .572 | - | - | - | - |
| ConvE | .440 | .400 | .450 | .520 | .410 | .313 | .457 | .600 | - | - | - | - |
| MINERVA | .450 | .410 | .460 | .510 | .293 | .217 | .329 | .456 | - | - | - | - |
| A2N | **.490** | **.450** | **.500** | .550 | .422 | .328 | .464 | .608 | - | - | - | - |
| QMult | .451 | .403 | .472 | .553 | .439 | .341 | .485 | .636 | .692 | .626 | .736 | .799 |
| OMult | .461 | .414 | .482 | .559 | .440 | .340 | .486 | .636 | .689 | .623 | .733 | .801 |
| ConvQ | .470 | .437 | .482 | .538 | .441 | .344 | .482 | .632 | .674 | .612 | .715 | .783 |
| ConvO | .473 | .442 | .491 | .535 | .465 | .367 | .512 | .654 | .622 | .535 | .682 | 774 |
| Q-OMult | .457 | .401 | .48 | .559 | .448 | .347 | .495 | .644 | .696 | .632 | .734 | **.804** |
| QMult-ConvQ | .466 | .424 | .480 | .557 | .451 | .353 | .496 | .646 | **.697** | **.635** | **.737** | .803 |
| QMult-ConvO | .474 | .435 | .487 | .558 | .467 | .367 | .516 | **.662** | .680 | .610 | .727 | .800 |
| OMult-ConvQ | .471 | .430 | .487 | **.560** | .452 | .354 | .495 | .647 | .696 | .632 | .735 | .803 |
| OMult-ConvO | .476 | .436 | .488 | .559 | .466 | .366 | .515 | **.662** | .676 | .602 | .724 | .803 |
| ConvQ-O | .477 | .442 | .494 | .548 | **.468** | **.370** | **.515** | .661 | .675 | .603 | .724 | .795 |

Table 6: MRR link prediction results per relations on WN18RR. Ensemble refers to averaging predictions of ConvQ-ConvO-OMult.

| **Relation Name** | **Rel. Type** | **RotatE** | **QMult** | **ConvQ** | **OMult** | **ConvO** | **Ensemble** |
|---|---|---|---|---|---|---|---|
| hypernym | S | **.15** | .10 | .14 | .11 | .13 | .13 |
| instance_hypernym | S | .32 | .35 | .37 | .36 | .37 | **.39** |
| member_meronym | C | **.23** | .22 | .20 | **.23** | .20 | **.23** |
| synset_domain_topic_of | C | **.34** | .31 | .31 | .32 | .33 | **.34** |
| has_part | C | .18 | **.19** | .17 | .18 | .18 | **.19** |
| member_of_domain_usage | C | .32 | .29 | .28 | .27 | **.33** | .29 |
| member_of_domain_region | C | .20 | .25 | **.38** | .30 | .37 | **.38** |
| derivationally_related_form | R | .95 | **.98** | **.98** | **.98** | **.98** | **.98** |
| also_see | R | .59 | **.67** | .65 | .66 | .66 | .66 |
| verb_group | R | .94 | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |
| similar_to | R | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |

Table 7: Link prediction results depending on the direction of prediction (head vs. tail prediction) on WN18RR. Ensemble refers to averaging predictions of ConvQ-ConvO-OMult.

| | QMult | ConvQ | OMult | ConvO | Ensemble |
|---|---|---|---|---|---|
| (h, r, x) | | | | | |
| hypernym | .12 | **.18** | .13 | **.18** | .17 |
| instance_hypernym | .53 | .56 | .53 | .57 | **.58** |
| member_meronym | **.17** | .09 | .16 | .08 | .14 |
| synset_domain_topic_of | .49 | .47 | .51 | **.52** | **.52** |
| has_part | **.15** | .12 | .15 | .14 | .14 |
| member_of_domain_usage | .04 | .02 | .07 | **.08** | .05 |
| member_of_domain_region | .05 | **.06** | .05 | **.06** | .05 |
| derivationally_related_form | .98 | .98 | .98 | .98 | .98 |
| also_see | **.67** | .63 | .65 | .63 | .63 |
| verb_group | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |
| similar_to | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |
| (x, r, t) | | | | | |
| hypernym | .07 | .09 | .09 | .08 | **.10** |
| instance_hypernym | .17 | .18 | **.19** | .17 | **.19** |
| member_meronym | .27 | .31 | .29 | **.33** | .32 |
| synset_domain_topic_of | .13 | .14 | .13 | **.15** | **.15** |
| has_part | .22 | .22 | .22 | .22 | **.24** |
| member_of_domain_usage | .53 | .54 | .47 | **.59** | .54 |
| member_of_domain_region | .45 | .69 | .55 | .67 | **.70** |
| derivationally_related_form | .98 | .98 | .98 | **.99** | .98 |
| also_see | .68 | .67 | .66 | **.69** | .68 |
| verb_group | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |
| similar_to | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** |

### 6.3. Link Prediction Per Relation and Direction

We reevaluate link prediction performance of some of the best-performing models from Table 2 in Tables 6 and 7. Allen et al. (2021) distinguish three types of relations: Type S relations are specialization relations such as `hypernym`, type C denote so-called generalized context-shifts and include `has_part` relations, and type R relations include so-called highly-related relations such as `similar_to`. Our results show that our approaches accurately rank missing tail and head entities for type R relations. For instance, our approaches perfectly rank (1.0 MRR) missing entities of symmetric relations (`verb_group` and `similar_to`). However, the direction of entity prediction has a significant impact on the results for non-symmetric type C relations. For instance, MRR performance of QMult, ConvQ, OMult and ConvO vary by up to absolute 0.63 for the relation `member_of_domain_region`. The low performance of `hypernym` (type S) may stem from the fact that there are 184 triples in the test split of WN18RR where `hypernym` occurs with entities of which at least one did not occur in the training split (Demir and Ngomo, 2021b). Models often perform poorly on type C relations but considerably better on type R relations corroborating findings by Allen et al. (2021).

### 6.4. Batch vs. Unit Normalization

We investigate the effect of using batch-normalization, instead of unit normalization as previously proposed by Zhang et al. (2019). Table 8 indicates that the scaling effect of hypercomplex multiplications can be effectively alleviated by using the batch normalization

Table 8: Batch normalization vs. unit normalization for link prediction.

| | Kinship | | | | UMLS | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | H@1 | @3 | @10 | MRR | @1 | @3 | @10 |
| GNTP-Standard Minervini et al. (2020) | .72 | .59 | .82 | .96 | .80 | .70 | .88 | .95 |
| GNTP-Attention Minervini et al. (2020) | .76 | .64 | .85 | .96 | .86 | .76 | .95 | .98 |
| NTP Minervini et al. (2020) | .35 | .24 | .37 | .57 | .80 | .70 | .88 | .95 |
| NeuralLP Minervini et al. (2020) | .62 | .48 | .71 | .91 | .78 | .64 | .87 | .96 |
| MINERVA Minervini et al. (2020) | .72 | .60 | .81 | .92 | .83 | .73 | .90 | .97 |
| ConvE Dettmers et al. (2018) | .83 | .74 | .92 | .98 | .94 | .92 | .96 | .99 |
| QMult (batch) | **.88** | **.81** | **.94** | **.99** | **.96** | **.93** | **.98** | **1.0** |
| QMult (unit) | .69 | .58 | .78 | .90 | .77 | .69 | .82 | .93 |
| OMult (batch) | .87 | .80 | **.94** | **.99** | .95 | .91 | **.98** | **1.0** |
| OMult (unit) | .69 | .57 | .77 | .89 | .76 | .66 | .82 | .93 |
| ConvQ (batch) | .86 | .77 | .93 | .98 | .92 | .86 | **.98** | **1.0** |
| ConvQ (unit) | .61 | .49 | .68 | .85 | .55 | .45 | .59 | .75 |
| ConvO (batch) | .86 | .77 | .93 | .98 | .90 | .82 | **.98** | **1.0** |
| ConvO (unit) | .65 | .53 | .72 | .86 | .56 | .46 | .61 | .78 |

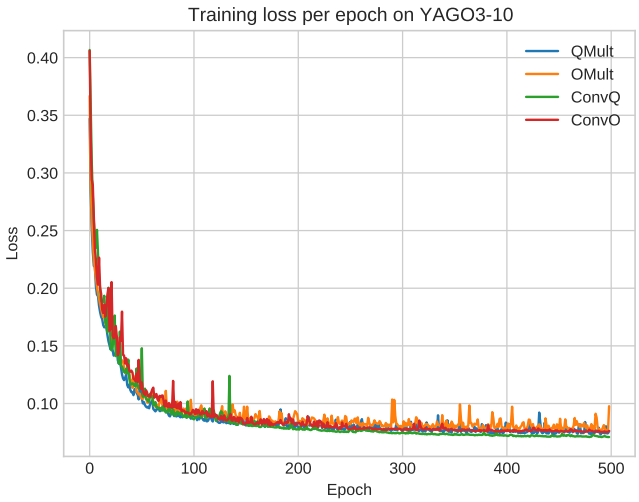


Figure 1: Convergence on the training set.

technique. Replacing unit normalization with the batch normalization technique allows benefiting (1) from its regularization effect and (2) from its numerical stability. Through batch normalization, our models are able to control the rate of normalization and benefit from its implicit regularization effect (Ioffe and Szegedy, 2015).

### 6.5. Convergence on YAGO3-10

Figure 1 indicates that incurred binary cross entropy losses significantly decrease within the first 100 epochs. After the 300th iteration, ConvQ and ConvO appear to converge as losses do not fluctuate, whereas training losses of QMult and OMult continue fluctuating.

Table 9: Link prediction results on WN18 and FB15K.

| | WN18 | | | | | FB15K | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | Param. | MRR | Hit@10 | Hit@3 | Hit@1 | Param. | MRR | Hit@10 | Hit@3 | Hit@1 |
| TransE | - | .495 | .943 | .888 | .113 | - | .463 | .749 | .578 | .297 |
| TransR | - | .605 | .940 | .876 | .335 | - | .346 | .582 | .404 | .218 |
| ER-MLP | - | .712 | .863 | .775 | .626 | - | .288 | .501 | .317 | .173 |
| RESCAL | - | .890 | .928 | .904 | .842 | - | .354 | .587 | .409 | .235 |
| HolE | - | .938 | .949 | .945 | .930 | - | .524 | .739 | .613 | .402 |
| SimplE | - | .942 | .947 | .944 | .939 | - | .727 | .838 | .773 | .660 |
| TorusE | - | .947 | .954 | .950 | .943 | - | .733 | .832 | .771 | .674 |
| RotatE | - | .947 | .961 | .953 | .938 | - | .699 | .872 | .788 | .585 |
| QuatE | - | .949 | .960 | .954 | .941 | - | .770 | .878 | .821 | .700 |
| QuatE$^2$ | - | .950 | .962 | .954 | .944 | - | **.833** | .900 | .859 | **.800** |
| QMult | 16.39M | .975 | .980 | .976 | .972 | 7.05M | .755 | .896 | .819 | .668 |
| OMult | 16.39M | .975 | **.981** | .976 | .972 | 7.05M | .748 | .889 | .813 | .660 |
| ConvQ | 21.51M | **.976** | .980 | **.977** | **.973** | 12.17M | .813 | **.923** | **.868** | .743 |
| ConvO | 21.51M | **.976** | .980 | **.977** | **.973** | 12.17M | .810 | **.923** | .865 | .739 |

### 6.6. Link Prediction Results on Previous Benchmark Datasets

Table 9 reports results on WN18 and FB15K showing that our approaches ConvQ and ConvQ outperform state-of-the-art approaches in 6 out of 8 metrics on the datasets.

## 7. Discussion

Our approaches often outperform many state-of-the-art approaches on all benchmark datasets. QMult and OMult outperform many state-of-the-art approaches including DistMult and ComplEx. These results indicate that scoring functions based on hypercomplex multiplications are more effective than scoring functions based on real and complex multiplications. This observation corroborates findings of Zhang et al. (2019). ConvO often perform slightly better than ConvQ on all datasets. Additionally, QMult and QMult perform particularly well on YAGO3-10. Figure 1 indicates that ConvO reaches a lower training error than QMult on YAGO3-10. This suggests that ConvO might have overfitted despite its higher expressiveness due to non-linearities and convolutions. In future work, we will design a regularization technique tailored to convolutions. Overall, superior performance of our models stems from (1) hypercomplex embeddings and (2) the inclusion of convolution operations. Our models are allowed to degrade into ComplEx or DistMult if necessary (see Section 4). Inclusion of the convolution operation followed by an affine transformation permits finding a good ratio between expressiveness and the number of parameters.

## 8. Conclusion

In this study, we presented effective compositions of convolution operations with hypercomplex multiplications in the quaternion and octonion algebras to address the link prediction problem. Experimental results showed that QMult and OMult performing hypercomplex multiplication on hypercomplex-valued embeddings of entities and relations are effective

methods to tackle the link prediction problem. ConvQ and ConvO forge QMult and OMult with convolution operations followed by an affine transformation. By virtue of this novel composition, ConvQ and ConvO facilitate finding a good ratio between expressiveness and the number of parameters. Experiments suggest that (1) generalizing real- and complex-valued models such as DistMult and ComplEx to the hypercomplex space is beneficial, particularly for larger knowledge graphs, (2) the scaling effect of hypercomplex multiplication can be more effectively tackled with batch normalization than unit normalization, and (3) the application of ensembling can be used to further increase generalization performance.

In future work, we plan to investigate the generalization of our approaches to temporal knowledge graphs and translation based models on hypercomplex vector spaces.

## Acknowledgments

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the project DAIKIRI under the grant no 01IS19085B and by the the German Federal Ministry for Economic Affairs and Energy (BMWi) within the project RAKI under the grant no 01MD19012B. We are grateful to Pamela Heidi Douglas for editing the manuscript.

## References

Carl Allen, Ivana Balažević, and Timothy Hospedales. Interpreting knowledge graph relation representation from word embeddings. In *International Conference on Learning Representations*, 2021.

John Baez. The octonions. *Bulletin of the American Mathematical Society*, 2002.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. *Advances in Neural Information Processing Systems*, 32:4463–4473, 2019.

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

Ivana Balažević, Carl Allen, and Timothy M Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019b.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. A2n: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392, 2019.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, 2020.

Heng Chen, Weimei Wang, Guanyu Li, and Yimin Shi. A quaternion-embedded capsule network model for knowledge graph completion. *IEEE Access*, 2020.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. A physical embedding model for knowledge graphs. In *Joint International Semantic Technology Conference*, pages 192–209. Springer, 2019.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. Convolutional complex knowledge graph embeddings. In *European Semantic Web Conference*, pages 409–424. Springer, 2021a.

Caglar Demir and Axel-Cyrille Ngonga Ngomo. Out-of-vocabulary entities in link prediction. *arXiv preprint arXiv:2105.12524*, 2021b.

Caglar Demir, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. A shallow neural model for relation prediction. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 179–182. IEEE, 2021.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

William Rowan Hamilton. Lxxviii. on quaternions; or on a new system of imaginaries in algebra: To the editors of the philosophical magazine and journal. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1844.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, pages 2863–2872. PMLR, 2018.

Haonan Lu and Hailin Hu. Dense: An enhanced non-abelian group representation for knowledge graph embedding. *arXiv preprint arXiv:2008.04548*, 2020.

Pasquale Minervini, Matko Bošnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. Differentiable reasoning on large knowledge bases and natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pages 5182–5190, 2020.

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

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International Conference on International Conference on Machine Learning*, 2011.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30:6338–6347, 2017.

Titouan Parcollet, Ying Zhang, Mohamed Morchid, Chiheb Trabelsi, Georges Linarès, Renato De Mori, and Yoshua Bengio. Quaternion convolutional neural networks for end-to-end automatic speech recognition. *arXiv preprint arXiv:1806.07789*, 2018.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2019.

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

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on International Conference on Machine Learning*, 2016.

Théo Trouillon, Christopher R Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*, 2017.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

Aston Zhang, Yi Tay, Shuai Zhang, Alvin Chan, Anh Tuan Luu, Siu Cheung Hui, and Jie Fu. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations*, 2021.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 2019.