QuatRE: Relation-Aware Quaternions for Knowledge Graph Embeddings

Abstract

We propose an effective embedding model, named QuatRE, to learn quaternion embeddings for entities and relations in knowledge graphs. QuatRE aims to enhance correlations between head and tail entities given a relation within the Quaternion space with Hamilton product. QuatRE achieves this goal by further associating each relation with two relation-aware quaternion vectors which are used to rotate the head and tail entities' quaternion embeddings, respectively. To obtain the triple score, QuatRE rotates the rotated embedding of the head entity using the normalized quaternion embedding of the relation, followed by a quaternion-inner product with the rotated embedding of the tail entity. Experimental results demonstrate that our QuatRE produces state-of-the-art performances on well-known benchmark datasets for knowledge graph completion.

1 Introduction

Knowledge graphs (KGs) are constructed to represent relationships between entities in the form of triples *(head, relation, tail)* denoted as *(h, r, t)*. A typical problem in KGs is the lack of many valid triples [35]; therefore, research approaches have been proposed to predict whether a new triple missed in KGs is likely valid [3, 2, 26]. These approaches often utilize embedding models to compute a score for each triple, such that valid triples have higher scores than invalid ones. For example, the score of the valid triple (Melbourne, city_Of, Australia) is higher than the score of the invalid one (Melbourne, city_Of, Germany).

Most of the aforementioned existing models focus on embedding entities and relations within the real-valued vector space [2, 34, 13, 37, 4, 17, 18]. Recently the use of hyper-complex vector space has considered on the Quaternion space $\mathbb{H}$ consisting of a real and three separate imaginary axes. It provides highly expressive computations through the Hamilton product compared to the real-valued and complex vector spaces. QuatE [38] is proposed to embed entities and relations within the Quaternion space via a Hamilton product-based rotation between the head and relation embeddings followed by a quaternion-inner product with the tail embedding. QuatE is considered as one of recent state-of-the-art models as it outperforms up-to-date strong baselines for knowledge graph completion [38]. QuatE, however, has a limitation in capturing the correlations between the head and tail entities. Addressing the problem, we propose an effective embedding model, named QuatRE, to learn the quaternion embeddings for entities and relations. QuatRE further associates each relation with two relation-aware quaternion vectors to rotate the head and tail embeddings through the Hamilton product, respectively. As a result, QuatRE strengthens the correlations between the head and tail entities. To summarize, our main contributions are as follows:

• We present an effective embedding model QuatRE to embed entities and relations within the Quaternion space with the Hamilton product. QuatRE further utilizes two relation-aware quaternion vectors for each relation to increase the correlations between the head and tail entities.

• Experimental results show that our QuatRE obtains state-of-the-art performances on well-known benchmark datasets for the knowledge graph completion task; thus, it can act as a new strong baseline for future works.

NeurIPS 2020 workshop on Differential Geometry meets Deep Learning.

2 The approach

2.1 Quaternion background

We provide key notations and operations related to quaternion space required for our model. Additional details can further be found in the appendix.

A quaternion $q \in \mathbb{H}$ is a hyper-complex number consisting of a real and three separate imaginary components [9] defined as: $q = q_\mathsf{r} + q_\mathsf{i}\mathbf{i} + q_\mathsf{j}\mathbf{j} + q_\mathsf{k}\mathbf{k}$, where $q_\mathsf{r}, q_\mathsf{i}, q_\mathsf{j}, q_\mathsf{k} \in \mathbb{R}$, and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units that $\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, leads to noncommutative multiplication rules as $\mathbf{ij} = \mathbf{k}, \mathbf{ji} = -\mathbf{k}, \mathbf{jk} = \mathbf{i}, \mathbf{kj} = -\mathbf{i}, \mathbf{ki} = \mathbf{j}$, and $\mathbf{ik} = -\mathbf{j}$. Correspondingly, a n-dimensional quaternion vector $\boldsymbol{q} \in \mathbb{H}^n$ is defined as: $\boldsymbol{q} = \boldsymbol{q}_\mathsf{r} + \boldsymbol{q}_\mathsf{i}\mathbf{i} + \boldsymbol{q}_\mathsf{j}\mathbf{j} + \boldsymbol{q}_\mathsf{k}\mathbf{k}$, where $\boldsymbol{q}_\mathsf{r}, \boldsymbol{q}_\mathsf{i}, \boldsymbol{q}_\mathsf{j}, \boldsymbol{q}_\mathsf{k} \in \mathbb{R}^n$.

Norm. The normalized quaternion vector $\boldsymbol{q}^\triangleleft$ of $\boldsymbol{q} \in \mathbb{H}^n$ is computed as: $\boldsymbol{q}^\triangleleft = \frac{\boldsymbol{q}_\mathsf{r} + \boldsymbol{q}_\mathsf{i}\mathbf{i} + \boldsymbol{q}_\mathsf{j}\mathbf{j} + \boldsymbol{q}_\mathsf{k}\mathbf{k}}{\sqrt{\boldsymbol{q}_\mathsf{r}^2 + \boldsymbol{q}_\mathsf{i}^2 + \boldsymbol{q}_\mathsf{j}^2 + \boldsymbol{q}_\mathsf{k}^2}}$

Hamilton product. The Hamilton product of two vectors $\boldsymbol{q}$ and $\boldsymbol{p} \in \mathbb{H}^n$ is computed as:

$$\begin{aligned} \boldsymbol{q} \otimes \boldsymbol{p} &= (\boldsymbol{q}_\mathsf{r} \circ \boldsymbol{p}_\mathsf{r} - \boldsymbol{q}_\mathsf{i} \circ \boldsymbol{p}_\mathsf{i} - \boldsymbol{q}_\mathsf{j} \circ \boldsymbol{p}_\mathsf{j} - \boldsymbol{q}_\mathsf{k} \circ \boldsymbol{p}_\mathsf{k}) + (\boldsymbol{q}_\mathsf{i} \circ \boldsymbol{p}_\mathsf{r} + \boldsymbol{q}_\mathsf{r} \circ \boldsymbol{p}_\mathsf{i} - \boldsymbol{q}_\mathsf{k} \circ \boldsymbol{p}_\mathsf{j} + \boldsymbol{q}_\mathsf{j} \circ \boldsymbol{p}_\mathsf{k})\mathbf{i} \\ &+ (\boldsymbol{q}_\mathsf{j} \circ \boldsymbol{p}_\mathsf{r} + \boldsymbol{q}_\mathsf{k} \circ \boldsymbol{p}_\mathsf{i} + \boldsymbol{q}_\mathsf{r} \circ \boldsymbol{p}_\mathsf{j} - \boldsymbol{q}_\mathsf{i} \circ \boldsymbol{p}_\mathsf{k})\mathbf{j} + (\boldsymbol{q}_\mathsf{k} \circ \boldsymbol{p}_\mathsf{r} - \boldsymbol{q}_\mathsf{j} \circ \boldsymbol{p}_\mathsf{i} + \boldsymbol{q}_\mathsf{i} \circ \boldsymbol{p}_\mathsf{j} + \boldsymbol{q}_\mathsf{r} \circ \boldsymbol{p}_\mathsf{k})\mathbf{k} \end{aligned} \quad (1)$$

where $\circ$ denotes the element-wise product. We note that the Hamilton product is not commutative, i.e., $q \otimes p \neq p \otimes q$.

Quaternion-inner product. The quaternion-inner product $\bullet$ of two quaternion vectors $\boldsymbol{q}$ and $\boldsymbol{p} \in \mathbb{H}^n$ returns a scalar, which is computed as: $\boldsymbol{q} \bullet \boldsymbol{p} = \boldsymbol{q}_\mathsf{r}^\mathsf{T}\boldsymbol{p}_\mathsf{r} + \boldsymbol{q}_\mathsf{i}^\mathsf{T}\boldsymbol{p}_\mathsf{i} + \boldsymbol{q}_\mathsf{j}^\mathsf{T}\boldsymbol{p}_\mathsf{j} + \boldsymbol{q}_\mathsf{k}^\mathsf{T}\boldsymbol{p}_\mathsf{k}$.

QuatE: QuatE [38] computes the score of the triple *(h, r, t)* as: $(\boldsymbol{v}_h \otimes \boldsymbol{v}_r^\triangleleft) \bullet \boldsymbol{v}_t$. It is noted that directly using the quaternion embeddings $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, $\boldsymbol{v}_t$ to obtain the triple score might lead to the problem of struggling to strengthen the relation-aware correlations between the head and tail entities. Thus, arguably this could lower the performance of QuatE. Our key contribution is to overcome this limitation by integrating relation-aware quaternions to increase the correlations between the entities.

2.2 The proposed QuatRE

A knowledge graph (KG) $\mathcal{G}$ is a collection of valid factual triples in the form of *(head, relation, tail)* denoted as (h, r, t) such that $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$ where $\mathcal{E}$ is a set of entities and $\mathcal{R}$ is a set of relations. KG embedding models aim to embed entities and relations to a low-dimensional vector space to define a score function f. This function is to give an implausibility score for each triple (h, r, t), such that the valid triples obtain higher scores than the invalid triples.

Given a triple (h, r, t), QuatRE also represents the embeddings of entities and relations within the Quaternion space as $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{H}^n$. QuatRE further associates each relation r with two quaternion vectors $\mathbf{v}_{r,1}$ and $\mathbf{v}_{r,2} \in \mathbb{H}^n$. QuatRE then uses the Hamilton product to rotate the quaternion embeddings $\boldsymbol{v}_h$ and $\boldsymbol{v}_t$ by the normalized vectors $\mathbf{v}_{r,1}^\triangleleft$ and $\mathbf{v}_{r,2}^\triangleleft$ respectively as:

$$\boldsymbol{v}_{h,r,1} = \boldsymbol{v}_h \otimes \mathbf{v}_{r,1}^\triangleleft \quad (2)$$

$$\boldsymbol{v}_{t,r,2} = \boldsymbol{v}_t \otimes \mathbf{v}_{r,2}^\triangleleft \quad (3)$$

After that, QuatRE rotates $\boldsymbol{v}_{h,r,1}$ by the normalized quaternion embedding $\boldsymbol{v}_r^\triangleleft$ before computing the quaternion-inner product with $\boldsymbol{v}_{t,r,2}$. The quaternion components of input vectors are shared during computing the Hamilton product, as shown in Equation 14. Therefore, QuatRE uses two rotations in Equations 2 and 3 for $\boldsymbol{v}_h$ and $\boldsymbol{v}_t$ to increase the correlations between the head h and tail t entities given the relation r, as illustrated in Figure 3.

Formally, we define the QuatRE score function f for the triple (h, r, t) as:

$$f(h, r, t) = (\boldsymbol{v}_{h,r,1} \otimes \boldsymbol{v}_r^\triangleleft) \bullet \boldsymbol{v}_{t,r,2} = \left(\left(\boldsymbol{v}_h \otimes \mathbf{v}_{r,1}^\triangleleft\right) \otimes \boldsymbol{v}_r^\triangleleft\right) \bullet \left(\boldsymbol{v}_t \otimes \mathbf{v}_{r,2}^\triangleleft\right)$$

Learning process. We employ the Adagrad optimizer [5] to train our proposed QuatRE by minimizing the following loss function [31] with the regularization on model parameters $\boldsymbol{\theta}$ as:

$$\mathcal{L} = \sum_{(h,r,t)\in\{\mathcal{G}\cup\mathcal{G}'\}} \log \left(1 + \exp\left(-l_{(h,r,t)} \cdot f(h,r,t)\right)\right) + \lambda\|\boldsymbol{\theta}\|_2^2 \tag{4}$$

$$\text{in which, } l_{(h,r,t)} = \begin{cases} 1 & \text{for } (h,r,t) \in \mathcal{G} \\ -1 & \text{for } (h,r,t) \in \mathcal{G}' \end{cases}$$

where we use l_2-norm with the regularization rate λ; and $\mathcal{G}$ and $\mathcal{G}'$ are collections of valid and invalid triples, respectively. $\mathcal{G}'$ is generated by corrupting valid triples in $\mathcal{G}$.

Discussion. If we fix the real components of both $\mathbf{v}_{r,1}$ and $\mathbf{v}_{r,2}$ to $\mathbf{1}$, and fix the imaginary components of both $\mathbf{v}_{r,1}$ and $\mathbf{v}_{r,2}$ to $\mathbf{0}$, our QuatRE is simplified to QuatE. Hence the QuatRE's derived formula might look simple as an extension of QuatE. However, to come with the extension, our original intuition is not straightforward, and this intuition has a deeper insight. We also note that given the same embedding dimension, QuatE and our QuatRE have comparable numbers of parameters.

3 Experimental results

Setup. We present the statistics of the datasets, the evaluation protocol, the training protocol, and the optimal hyper-parameters on the validation set for each dataset in appendix.

Table 1: Experimental results on the WN18RR and FB15k-237 test sets. Hits@k (H@k) is reported in %. The best scores are in bold, while the second best scores are in underline. The results of TransE are taken from [17]. The results of DistMult and ComplEx are taken from [4]. The results of ConvKB are taken using the Pytorch implementation released by [17]. We note that GC-OTE and RotatE$_{Adv}$ apply a self-adversarial negative sampling, which is different from the common sampling strategy used in the previous baselines, QuatE and our QuatRE. QuatE$_{N3Rec}$ uses the N3 regularization and reciprocal learning [12], which requires a large embedding dimension. GC-OTE, ReInceptionE, and R-GCN+ integrate information about relation paths. Thus, for a fair comparison, we do not compare our QuatRE with these models.

Method	**WN18RR**					**FB15k-237**				
	MR	MRR	H@10	H@3	H@1	MR	MRR	H@10	H@3	H@1
TransE [2]	3384	0.226	50.1	–	–	357	0.294	46.5	–	–
DistMult [37]	5110	0.430	49.0	44.0	39.0	254	0.241	41.9	26.3	15.5
ComplEx [31]	5261	0.440	51.0	46.0	41.0	339	0.247	42.8	27.5	15.8
ConvE [4]	5277	0.460	48.0	43.0	39.0	246	0.316	49.1	35.0	23.9
ConvKB [17]	2741	0.220	50.8	–	–	196	0.302	48.3	–	–
NKGE [33]	4170	0.450	52.6	46.5	42.1	237	0.330	51.0	36.5	24.1
RotatE [27]	3277	0.470	56.5	48.8	42.2	185	0.297	48.0	32.8	20.5
InteractE [32]	5202	0.463	52.8	–	43.0	172	<u>0.354</u>	53.5	–	<u>26.3</u>
QuatE [38]	<u>2314</u>	<u>0.488</u>	<u>58.2</u>	<u>50.8</u>	<u>43.8</u>	**87**	0.348	<u>55.0</u>	<u>38.2</u>	24.8
QuatRE	**1986**	**0.493**	**59.2**	**51.9**	**43.9**	<u>88</u>	**0.367**	**56.3**	**40.4**	**26.9**
GC-OTE [28]	–	0.491	58.3	51.1	44.2	–	0.361	55.0	39.6	26.7
ReInceptionE [36]	1894	0.483	58.2	–	–	173	0.349	52.8	–	–
RotatE$_{Adv}$ [27]	3340	0.476	57.1	49.2	42.8	177	0.338	53.3	37.5	24.1
QuatE$_{N3Rec}$ [38]	–	0.482	57.2	49.9	43.6	–	0.366	55.6	40.1	27.1
R-GCN+ [25]	–	–	–	–	–	–	0.249	41.7	26.4	15.1

Main results. We report the experimental results on the benchmark datasets in Table 1. In general, QuatRE outperforms up-to-date baselines for all metrics except the second-best MR on FB15k-237. Especially when comparing with QuatE, on WN18RR, QuatRE gains significant improvements of $2314 - 1986 = 328$ in MR (which is about 14% relative improvement), and 1.0% and 1.1% absolute improvements in Hits@10 and Hits@3 respectively. Besides, on FB15k-237, QuatRE achieves improvements of $0.367 - 0.348 = 0.019$ (which is 5.5% relative improvement) and obtains absolute gains of 1.3%, 2.2%, and 2.1% in Hits@10, Hits@3, and Hits@1 respectively.

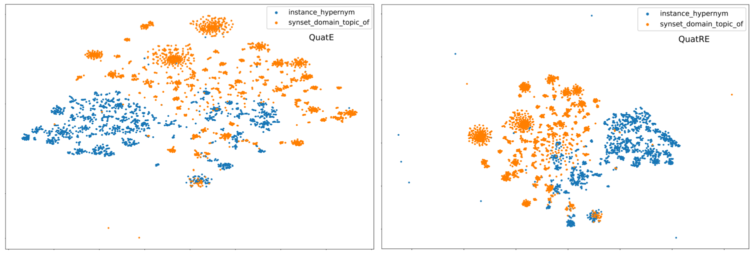


Figure 1: Visualization of the learned entity embeddings on WN18RR.

Correlation analysis. To qualitatively demonstrate the correlations between the entities, we use t-SNE [14] to visualize the learned quaternion embeddings of the entities on WN18RR for QuatE and QuatRE. We select all entities associated with two relations consisting of "instance_hypernym" and "synset_domain_topic_of". We then vectorize each quaternion embedding using a vector concatenation across the four components; hence, we obtain a real-valued vector representation for applying t-SNE. The visualization in Figure 1 shows that the entity distribution in our QuatRE is denser than that in QuatE; hence this implies that QuatRE strengthens the correlations between the entities.

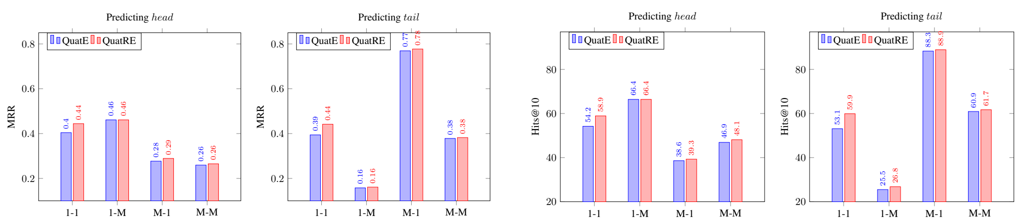


Figure 2: MRR and Hits@10 on the FB15k-237 test set for QuatE and our QuatRE with respect to each relation category.

Relation analysis. Following [2], for each relation r, we calculate the averaged number η_h of head entities per tail entity and the averaged number η_t of tail entities per head entity. If η_h <1.5 and η_t <1.5, r is categorized one-to-one (1-1). If η_h <1.5 and η_t $\geq$1.5, r is categorized one-to-many (1-M). If η_h $\geq$1.5 and η_t <1.5, r is categorized many-to-one (M-1). If η_h $\geq$1.5 and η_t $\geq$1.5, r is categorized many-to-many (M-M). Figure 2 shows the MRR and H@10 scores for predicting the head entities and then the tail entities with respect to each relation category on FB15k-237, wherein our QuatRE outperforms QuatE on these relation categories. Furthermore, we report the MRR scores for each relation on WN18RR in Table 2, which shows the effectiveness of QuatRE in modeling different types of relations.

Table 2: MRR score on the WN18RR test set with respect to each relation.

Relation	**QuatE**	**QuatRE**
hypernym	0.173	**0.190**
derivationally_related_form	**0.953**	0.943
instance_hypernym	0.364	**0.380**
also_see	0.629	**0.633**
member_meronym	0.232	**0.237**
synset_domain_topic_of	0.468	**0.495**
has_part	**0.233**	0.226
member_of_domain_usage	0.441	**0.470**
member_of_domain_region	0.193	**0.364**
verb_group	**0.924**	0.867
similar_to	**1.000**	**1.000**

4 Conclusion

In this paper, we propose QuatRE – an advantageous knowledge graph embedding model – to learn the embeddings of entities and relations within the Quaternion space with the Hamilton product. QuatRE further utilizes two relation-aware quaternion vectors for each relation to strengthen the correlations between the head and tail entities. Experimental results show that QuatRE outperforms up-to-date embedding models and produces state-of-the-art performances on well-known benchmark datasets for the knowledge graph completion task.

Acknowledgements

This research was partially supported by the ARC Discovery Projects DP150100031 and DP160103934.

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795, 2013.

[3] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306, 2011.

[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1811–1818, 2018.

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[6] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1819–1826, 2018.

[7] Chase J Gaudet and Anthony S Maida. Deep quaternion networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[9] William Rowan Hamilton. Ii. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(163):10–13, 1844.

[10] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, 2015.

[11] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295, 2018.

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

[13] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Learning*, pages 2181–2187, 2015.

[14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[15] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[16] Dai Quoc Nguyen, Dat Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Convolutional Neural Network-based Model for Knowledge Base Completion and Its Application to Search Personalization. *Semantic Web*, 10(5):947–960, 2019.

[17] 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. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333, 2018.

[18] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. A Relational Memory-based Embedding Model for Triple Classification and Search Personalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3429—-3435, 2020.

[19] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2180–2189, 2019.

[20] Dat Quoc Nguyen. An Overview of Embedding Models of Entities and Relationships for Knowledge Base Completion. *arXiv preprint*, arXiv:1703.08098, 2017.

[21] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, 2016.

[22] Titouan Parcollet, Mirco Ravanelli, Mohamed Morchid, Georges Linarès, Chiheb Trabelsi, Renato De Mori, and Yoshua Bengio. Quaternion recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[23] 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. In *The 19th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 22–26, 2018.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[25] Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607, 2018.

[26] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934, 2013.

[27] 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.

[28] Yun Tang, Jing Huang, Guangtao Wang, Xiaodong He, and Bowen Zhou. Orthogonal relation transforms with graph context modeling for knowledge graph embedding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2713–2722. Association for Computational Linguistics, 2020.

[29] Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. Lightweight and efficient neural natural language processing with quaternion networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1494–1503, 2019.

[30] Kristina Toutanova and Danqi Chen. Observed Versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.

[31] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080, 2016.

[32] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *International Conference on Learning Representations*, 2020.

[33] Kai Wang, Yu Liu, Xiujuan Xu, and Dan Lin. Knowledge graph embedding with entity neighbors and deep memory network. *arXiv preprint arXiv:1808.03752*, 2018.

[34] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[35] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge Base Completion via Search-based Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526, 2014.

[36] Zhiwen Xie, Guangyou Zhou, Jin Liu, and Jimmy Xiangji Huang. ReInceptionE: Relation-aware inception network with joint local-global structural information for knowledge graph embedding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5929–5939, 2020.

[37] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*, 2015.

[38] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2731–2741, 2019.

[39] Xuanyu Zhu, Yi Xu, Hongteng Xu, and Changjian Chen. Quaternion convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–647, 2018.

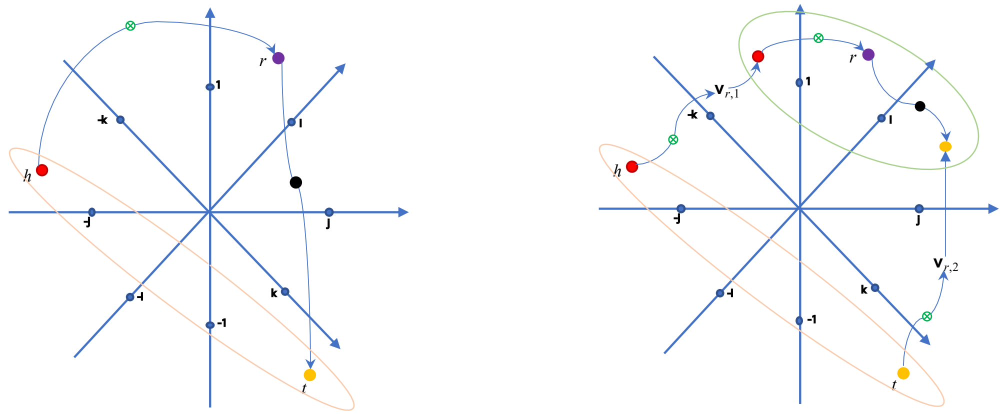
Table 3: The score functions in previous models. The table is adapted from [20].

Model	The score function $f(h,r,t)$
TransE	$-\|\boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t\|_p$ where $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{R}^n$; $\|\boldsymbol{v}\|_p$ denotes the p-norm of vector $\boldsymbol{v}$
ConvE	$\boldsymbol{v}_t^{\mathsf{T}} g\left(\mathbf{W}\mathsf{vec}\left(g\left(\mathsf{concat}\left(\widehat{\boldsymbol{v}}_h, \widehat{\boldsymbol{v}}_r\right) * \boldsymbol{\Omega}\right)\right)\right)$ where $*$ denotes a convolution operator $\boldsymbol{\Omega}$ denotes a set of filters; concat denotes a concatenation operator g denotes a non-linear function; $\widehat{\boldsymbol{v}}$ denotes a 2D reshaping of $\boldsymbol{v}$
ConvKB	$\mathbf{w}^{\mathsf{T}}\mathsf{concat}\left(g\left([\boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t] * \boldsymbol{\Omega}\right)\right)$
DistMult	$\langle \boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t \rangle = \sum_i^n \boldsymbol{v}_{h_i} \boldsymbol{v}_{r_i} \boldsymbol{v}_{t_i}$ where $\langle\rangle$ denotes a multiple-linear dot product
ComplEx	$\mathsf{Re}\left(\langle \boldsymbol{v}_h, \boldsymbol{v}_r, \boldsymbol{v}_t^* \rangle\right)$ where $\mathsf{Re}(c)$ denotes the real part of the complex c $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{C}^n$; $\boldsymbol{v}^*$ denotes the conjugate of the complex vector $\boldsymbol{v}$
RotatE	$-\|\boldsymbol{v}_h \circ \boldsymbol{v}_r - \boldsymbol{v}_t\|_p$ where $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{C}^n$; and $\circ$ denotes the element-wise product
QuatE	$(\boldsymbol{v}_h \otimes \boldsymbol{v}_r^{\triangleleft}) \bullet \boldsymbol{v}_t$ where $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{H}^n$; $\bullet$ denotes a quaternion-inner product $\otimes$ denotes the Hamilton product; the superscript $^{\triangleleft}$ denotes the normalized embedding
Our **QuatRE**	$\left(\left(\boldsymbol{v}_h \otimes \mathbf{v}_{r,1}^{\triangleleft}\right) \otimes \boldsymbol{v}_r^{\triangleleft}\right) \bullet \left(\boldsymbol{v}_t \otimes \mathbf{v}_{r,2}^{\triangleleft}\right)$ where $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, $\boldsymbol{v}_t$, $\mathbf{v}_{r,1}$, and $\mathbf{v}_{r,2} \in \mathbb{H}^n$

A Related work

Existing embedding models [2, 34] have been proposed to learn the vector representations of entities and relations for the knowledge graph completion task, where the goal is to score valid triples higher than invalid triples. As an example, Table 3 illustrates the score functions $f(h,r,t)$ in previous state-of-the-art models as well as our proposed model.

Early translation-based approaches exploit a translational characteristic so that the embedding of tail entity t should be close to the embedding of head entity h plus the embedding of relation t. For example, TransE [2] defines a score function: $f(h,r,t) = -\|\boldsymbol{v}_h + \boldsymbol{v}_r - \boldsymbol{v}_t\|_p$, where $\boldsymbol{v}_h$, $\boldsymbol{v}_r$, and $\boldsymbol{v}_t \in \mathbb{R}^n$ are vector embeddings of h, r and t respectively; and $\|\boldsymbol{v}\|_p$ denotes the p-norm of vector $\boldsymbol{v}$. As a result, TransE is suitable for 1-to-1 relationships, but not well-adapted for Many-to-1, 1-to-Many, and Many-to-Many relationships. To this end, some translation-based methods



QuatE QuatRE

Figure 3: An illustration of QuatE versus our proposed QuatRE.

have been proposed to deal with this issue such as TransH [34], TransR [13], TransD [10], and STransE [21]. Notably, DistMult [37] employs a multiple-linear dot product to score the triples as: $f(h, r, t) = \sum_i^n \boldsymbol{v}_{h_i} \boldsymbol{v}_{r_i} \boldsymbol{v}_{t_i}$.

One of the recent trends is to apply deep neural networks to measure the triples [4, 25, 32, 16, 19]. For example, ConvE [4] uses a convolution layer on a 2D input matrix of reshaping the embeddings of both the head entity and relation to produce feature maps that are then vectorized and computed with the embedding of the tail entity to return the score. While most of the existing models have worked in the real-valued vector space, several works have moved beyond the real-valued vector space to the complex vector space such as ComplEx [31] and RotatE [27]. ComplEx extends DistMult to use the multiple-linear dot product on the complex vector embeddings of entities and relations. Besides, RotatE considers a rotation-based translation within the complex vector space.

Recently the use of hyper-complex vector space has considered on the Quaternion space consisting of a real and three separate imaginary axes. It provides highly expressive computations through the Hamilton product compared to the real-valued and complex vector spaces. [39] and [7] embed the greyscale and each of RGB channels of the image to the real and three separate imaginary axes of the Quaternion space and achieve better accuracies compared real-valued convolutional neural networks with same structures for image classification tasks. The Quaternion space has also been successfully applied to speech recognition [23, 22], and natural language processing [29]. Regarding knowledge graph embeddings, [38] has recently proposed QuatE, which aims to learn entity and relation embeddings within the Quaternion space with the Hamilton product. QuatE, however, has a limitation in capturing the correlations between the head and tail entities. Our key contribution is to overcome this limitation by integrating relation-aware quaternion vectors to increase the correlations between the entities as illustrated in Figure 3.

B Quaternion background

For completeness, we briefly provide a background in quaternion, which has also similarly described in recent works [39, 22, 38, 29]. A quaternion $q \in \mathbb{H}$ is a hyper-complex number consisting of a real and three separate imaginary components [9] defined as:

$$q = q_\mathsf{r} + q_\mathsf{i}\mathbf{i} + q_\mathsf{j}\mathbf{j} + q_\mathsf{k}\mathbf{k} \tag{5}$$

where $q_\mathsf{r}, q_\mathsf{i}, q_\mathsf{j}, q_\mathsf{k} \in \mathbb{R}$, and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are imaginary units that $\mathbf{ijk} = \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, leads to noncommutative multiplication rules as $\mathbf{ij} = \mathbf{k}, \mathbf{ji} = -\mathbf{k}, \mathbf{jk} = \mathbf{i}, \mathbf{kj} = -\mathbf{i}, \mathbf{ki} = \mathbf{j}$, and $\mathbf{ik} = -\mathbf{j}$. Correspondingly, a n-dimensional quaternion vector $\boldsymbol{q} \in \mathbb{H}^n$ is defined as:

$$\boldsymbol{q} = \boldsymbol{q}_\mathsf{r} + \boldsymbol{q}_\mathsf{i}\mathbf{i} + \boldsymbol{q}_\mathsf{j}\mathbf{j} + \boldsymbol{q}_\mathsf{k}\mathbf{k} \tag{6}$$

where $\boldsymbol{q}_\mathsf{r}, \boldsymbol{q}_\mathsf{i}, \boldsymbol{q}_\mathsf{j}, \boldsymbol{q}_\mathsf{k} \in \mathbb{R}^n$. The operations for the Quaternion algebra are defined as follows:

Conjugate. The conjugate q^* of a quaternion q is defined as:

$$q^* = q_{\mathsf{r}} - q_{\mathsf{i}}\mathbf{i} - q_{\mathsf{j}}\mathbf{j} - q_{\mathsf{k}}\mathbf{k} \tag{7}$$

Addition. The addition of two quaternions q and p is defined as:

$$q + p = (q_{\mathsf{r}} + p_{\mathsf{r}}) + (q_{\mathsf{i}} + p_{\mathsf{i}})\mathbf{i} + (q_{\mathsf{j}} + p_{\mathsf{j}})\mathbf{j} + (q_{\mathsf{k}} + p_{\mathsf{k}})\mathbf{k} \tag{8}$$

Scalar multiplication. The multiplication of a scalar λ and a quaternion q is defined as:

$$\lambda q = \lambda q_{\mathsf{r}} + \lambda q_{\mathsf{i}}\mathbf{i} + \lambda q_{\mathsf{j}}\mathbf{j} + \lambda q_{\mathsf{k}}\mathbf{k} \tag{9}$$

Norm. The norm $\|q\|$ of a quaternion q is defined as:

$$\|q\| = \sqrt{q_{\mathsf{r}}^2 + q_{\mathsf{i}}^2 + q_{\mathsf{j}}^2 + q_{\mathsf{k}}^2} \tag{10}$$

The normalized or unit quaternion $q^{\triangleleft}$ is defined as:

$$q^{\triangleleft} = \frac{q}{\|q\|} \tag{11}$$

And the normalized quaternion vector $\boldsymbol{q}^{\triangleleft}$ of $\boldsymbol{q} \in \mathbb{H}^n$ is computed as:

$$\boldsymbol{q}^{\triangleleft} = \frac{\boldsymbol{q}_{\mathsf{r}} + \boldsymbol{q}_{\mathsf{i}}\mathbf{i} + \boldsymbol{q}_{\mathsf{j}}\mathbf{j} + \boldsymbol{q}_{\mathsf{k}}\mathbf{k}}{\sqrt{\boldsymbol{q}_{\mathsf{r}}^2 + \boldsymbol{q}_{\mathsf{i}}^2 + \boldsymbol{q}_{\mathsf{j}}^2 + \boldsymbol{q}_{\mathsf{k}}^2}} \tag{12}$$

Hamilton product. The Hamilton product $\otimes$ (i.e., the quaternion multiplication) of two quaternions q and p is defined as:

$$\begin{aligned} q \otimes p &= (q_{\mathsf{r}}p_{\mathsf{r}} - q_{\mathsf{i}}p_{\mathsf{i}} - q_{\mathsf{j}}p_{\mathsf{j}} - q_{\mathsf{k}}p_{\mathsf{k}}) \\ &+ (q_{\mathsf{i}}p_{\mathsf{r}} + q_{\mathsf{r}}p_{\mathsf{i}} - q_{\mathsf{k}}p_{\mathsf{j}} + q_{\mathsf{j}}p_{\mathsf{k}})\mathbf{i} \\ &+ (q_{\mathsf{j}}p_{\mathsf{r}} + q_{\mathsf{k}}p_{\mathsf{i}} + q_{\mathsf{r}}p_{\mathsf{j}} - q_{\mathsf{i}}p_{\mathsf{k}})\mathbf{j} \\ &+ (q_{\mathsf{k}}p_{\mathsf{r}} - q_{\mathsf{j}}p_{\mathsf{i}} + q_{\mathsf{i}}p_{\mathsf{j}} + q_{\mathsf{r}}p_{\mathsf{k}})\mathbf{k} \end{aligned} \tag{13}$$

The Hamilton product of two quaternion vectors $\boldsymbol{q}$ and $\boldsymbol{p} \in \mathbb{H}^n$ is computed as:

$$\begin{aligned} \boldsymbol{q} \otimes \boldsymbol{p} &= (\boldsymbol{q}_{\mathsf{r}} \circ \boldsymbol{p}_{\mathsf{r}} - \boldsymbol{q}_{\mathsf{i}} \circ \boldsymbol{p}_{\mathsf{i}} - \boldsymbol{q}_{\mathsf{j}} \circ \boldsymbol{p}_{\mathsf{j}} - \boldsymbol{q}_{\mathsf{k}} \circ \boldsymbol{p}_{\mathsf{k}}) \\ &+ (\boldsymbol{q}_{\mathsf{i}} \circ \boldsymbol{p}_{\mathsf{r}} + \boldsymbol{q}_{\mathsf{r}} \circ \boldsymbol{p}_{\mathsf{i}} - \boldsymbol{q}_{\mathsf{k}} \circ \boldsymbol{p}_{\mathsf{j}} + \boldsymbol{q}_{\mathsf{j}} \circ \boldsymbol{p}_{\mathsf{k}})\mathbf{i} \\ &+ (\boldsymbol{q}_{\mathsf{j}} \circ \boldsymbol{p}_{\mathsf{r}} + \boldsymbol{q}_{\mathsf{k}} \circ \boldsymbol{p}_{\mathsf{i}} + \boldsymbol{q}_{\mathsf{r}} \circ \boldsymbol{p}_{\mathsf{j}} - \boldsymbol{q}_{\mathsf{i}} \circ \boldsymbol{p}_{\mathsf{k}})\mathbf{j} \\ &+ (\boldsymbol{q}_{\mathsf{k}} \circ \boldsymbol{p}_{\mathsf{r}} - \boldsymbol{q}_{\mathsf{j}} \circ \boldsymbol{p}_{\mathsf{i}} + \boldsymbol{q}_{\mathsf{i}} \circ \boldsymbol{p}_{\mathsf{j}} + \boldsymbol{q}_{\mathsf{r}} \circ \boldsymbol{p}_{\mathsf{k}})\mathbf{k} \end{aligned} \tag{14}$$

where $\circ$ denotes the element-wise product. We note that the Hamilton product is not commutative, i.e., $q \otimes p \neq p \otimes q$.

Quaternion-inner product. The quaternion-inner product $\bullet$ of two quaternion vectors $\boldsymbol{q}$ and $\boldsymbol{p} \in \mathbb{H}^n$ returns a scalar, which is computed as:

$$\boldsymbol{q} \bullet \boldsymbol{p} = \boldsymbol{q}_{\mathsf{r}}^{\mathsf{T}}\boldsymbol{p}_{\mathsf{r}} + \boldsymbol{q}_{\mathsf{i}}^{\mathsf{T}}\boldsymbol{p}_{\mathsf{i}} + \boldsymbol{q}_{\mathsf{j}}^{\mathsf{T}}\boldsymbol{p}_{\mathsf{j}} + \boldsymbol{q}_{\mathsf{k}}^{\mathsf{T}}\boldsymbol{p}_{\mathsf{k}} \tag{15}$$

C Experimental setup

In the knowledge graph completion task [2], the goal is to predict a missing entity given a relation with another entity, for example, inferring a head entity h given (r, t) or inferring 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.

C.1 Datasets

We evaluate our proposed QuatRE on four benchmark datasets: WN18, FB15k [2], WN18RR [4], and FB15k-237 [30]. WN18 and FB15k are derived from the lexical KG WordNet [15] and the real-world KG Freebase [1] respectively. As mentioned in [30], WN18 and FB15k contains many reversible relations, which makes the prediction task become trivial and irrealistic. As shown in [4], recent state-of-the-art results on WN18 are still obtained by using a simple reversal. Therefore, their subsets WN18RR and FB15k-237 are derived to eliminate the reversible relation problem to create more realistic and challenging prediction tasks.

C.2 Evaluation protocol

Following [2], for each valid test triple (h, r, t), we replace either h or t by each of other entities to create a set of corrupted triples. We use the "Filtered" setting protocol [2], i.e., not including any corrupted triples that appear in the KG. We rank the valid test triple and corrupted triples in descending order of their scores. We employ evaluation metrics: mean rank (MR), mean reciprocal rank (MRR), and Hits@k (the proportion of the valid triples ranking in top k predictions). The final scores on the test set are reported for the model which obtains the highest Hits@10 on the validation set. Lower MR, higher MRR, and higher Hits@k indicate better performance.

C.3 Training protocol

Parameter initialization. For the fairness, similar to previous works, we apply the standard Glorot initialization [8] for parameter initialization in our QuatRE instead of utilizing a specialized initialization scheme used in QuatE [38].

Negative sampling. We use the Bernoulli negative sampling [34, 13] when sampling invalid triples in $\mathcal{G}'$. More formally, for each relation r, η_h denotes the averaged number of head entities per tail entity whilst η_t denotes the averaged number of tail entities per head entity. Given a valid triple (h, r, t) of relation r, we then generate a new head entity h' with probability $\frac{\eta_t}{\eta_h+\eta_t}$ to form an invalid triple (h', r, t) and a new tail entity t' with probability $\frac{\eta_h}{\eta_h+\eta_t}$ to form an invalid triple (h, r, t'). The Bernoulli negative sampling is very commonly used in the translation-based models and later embedding models, and also implemented in both QuatE and our QuatRE for a fair comparison.

Hyper-parameters. We implement our QuatRE based on Pytorch [24] and test on a single GPU. We set 100 batches for all four datasets. We then vary the learning rate α in {0.02, 0.05, 0.1}, the number s of negative triples sampled per training triple in {1, 5, 10}, the embedding dimension n in {128, 256, 384}, and the regularization rate λ in $\{0.05, 0.1, 0.2, 0.5\}$. We train our QuatRE up to 8,000 epochs on WN18 and WN18RR, and 2,000 epochs on FB15k and FB15k-237. We monitor the Hits@10 score after each 400 epochs on on WN18 and WN18RR, and each 200 epochs on FB15k and FB15k-237. We select the hyper-parameters using grid search and early stopping on the validation set with Hits@10. We present the statistics of the datasets in Table 4 and the optimal hyper-parameters on the validation set for each dataset in Table 5.

Table 4: Statistics of the experimental datasets.

Dataset	$\mid \mathcal{E} \mid$	$\mid \mathcal{R} \mid$	#Triples in train/valid/test		
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 5: The optimal hyper-parameters on the validation sets.

Dataset	α	n	λ	s
WN18RR	0.1	256	0.5	5
FB15k-237	0.1	384	0.5	10

Table 6: Experimental results on the WN18 and FB15k test sets. Hits@k (H@k) is reported in %. The best scores are in bold, while the second best scores are in underline. RotatE$_{Adv}$ uses a self-adversarial negative sampling. QuatE$_{N3Rec}$ applies N3 regularization and reciprocal learning. R-GCN+ exploits information about relation paths.

Method	**WN18**					**FB15k**				
	MR	MRR	H@10	H@3	H@1	MR	MRR	H@10	H@3	H@1
TransE [2]	–	0.495	94.3	88.8	11.3	–	0.463	74.9	57.8	29.7
DistMult [37]	655	0.797	94.6	–	–	42	<u>0.798</u>	89.3	–	–
ComplEx [31]	–	0.941	94.7	94.5	93.6	–	0.692	84.0	75.9	59.9
ConvE [4]	374	0.943	95.6	94.6	93.5	51	0.657	83.1	72.3	55.8
SimplE [11]	–	0.942	94.7	94.4	93.9	–	0.727	83.8	77.3	66.0
NKGE [33]	336	<u>0.947</u>	95.7	94.9	94.2	56	0.730	87.1	79.0	65.0
TorusE [6]	–	<u>0.947</u>	95.4	95.0	<u>94.3</u>	–	0.733	83.2	77.1	67.4
RotatE [27]	184	<u>0.947</u>	<u>96.1</u>	<u>95.3</u>	93.8	32	0.699	87.2	78.8	58.5
QuatE [38]	<u>162</u>	**0.950**	95.9	**95.4**	**94.5**	**17**	0.782	**90.0**	<u>83.5</u>	<u>71.1</u>
QuatRE	**116**	0.939	**96.3**	<u>95.3</u>	92.3	<u>23</u>	**0.808**	<u>89.6</u>	**85.1**	**75.1**
RotatE$_{Adv}$ [27]	309	0.949	95.9	95.2	94.4	40	0.797	88.4	83.0	74.6
QuatE$_{N3Rec}$ [38]	–	0.950	96.2	95.4	94.4	–	0.833	90.0	85.9	80.0
R-GCN+ [25]	–	0.819	96.4	92.9	69.7	–	0.696	84.2	76.0	60.1