ParConvE: Parrallel Convolutional 2D Knowledge Graph Embeddings for Link Prediction at Scale

Nilesh Agrawal [*1] **Nihar Ranjan Sahoo** [*1]

Abstract

The aim of knowledge graphs is to gather knowledge about the world and provide a structured representation of this knowledge. Current knowledge graphs are far from complete. To address the incompleteness of the knowledge graphs, link prediction approaches have been developed which make probabilistic predictions about new links in a knowledge graph given the existing links. Link prediction is a key research direction within this area. In this work, we will focus on link prediction at large scale using ParConvE and thus introduces ParConvE, a multi-layer convolutional network model for link prediction.

1. Introduction

Knowledge graphs are structured graphs with various entities as nodes and relations as edges. They are usually in form of RDF-style triples (se, r, oe), where se represents a subject entity, oe an object entity, and r the relation between them. In the past decades, a quantity of large scale knowledge graphs have sprung up, e.g., *Freebase, WordNet, YAGO, OpenKN* and have played a pivotal role in supporting many applications, such as link prediction, question answering, etc. Although these knowledge graphs are very large, i.e., usually containing thousands of relation types, millions of entities and billions of triples, they are still far from complete. As a result, **knowledge graph completion (KGC)** has been payed much attention to, which mainly aims to predict missing relations between entities under the supervision of existing triples.

1.1. Motivation

Recent years have witnessed great advances of translating embedding methods to tackle KGC problem. The methods represent entities and relations as the embedding vectors by regarding relations as translations from subject entities to object entities, such as TransE (Bordes et al., 2013),TransH(Wang et al., 2014), TransR(Lin et al., 2015), etc. However, the training procedure is time consuming, since they all employ **stochastic gradient descent** (SGD) to optimize a translation-based loss function, which may require days to converge for large knowledge graphs. Konwledge graphs can contain millions of facts; as a consequence, link predictors should scale in a manageable way with respect to both the number of parameters and computational costs to be applicable in real-world scenarios.

For solving such scaling problems, link prediction models are often composed of simple operations, like inner products and matrix multiplications over an embedding space, and use a limited number of parameters. DistMult(Yang et al., 2014) is such a model, characterised by three-way interactions between embedding parameters, which produce one feature per parameter. Using such simple, fast, shallow models allows one to scale to large knowledge graphs, at the cost of learning less expressive features. One way to solve the scaling problem of shallow architectures, and the overfitting problem of fully connected deep architectures, is to use parameter efficient, fast operators which can be composed into deep networks.

The convolution operator, commonly used in computer vision, has exactly these properties: it is parameter efficient and fast to compute, due to highly optimised GPU implementations. Furthermore, due to its ubiquitous use, robust methodologies have been established to control overfitting when training multi-layer convolutional networks. ConvE(Dettmers et al., 2017), a model that uses 2D convolutions over embeddings to predict missing links in knowledge graphs. ConvE is the simplest multi-layer convolutional architecture for link prediction: it is defined by a single convolution layer, a projection layer to the embedding dimension, and an inner product layer.

1.2. Problem Statement

This work aims to learn low-dimensional vector-space representations for entities/relations in knowledge Graphs, and at the same time attempts to handle large knowledge graphs

[*]Equal contribution [1]Department of Computer science and Automation, Indian Institute of Science, Bangalore. Correspondence to: Nilesh Agrawal <anilesh@iisc.ac.in>, Nihar Ranjan Sahoo <niharsahoo@iisc.ac.in>.

by distributing the graph across the nodes in a cluster, which results in learning graph embeddings at scale, with this huge knowledge graphs.

In recent years many approaches have been proposed to predict the missing facts in KGs but all these approaches try to predict the missing facts while learning representations in a sequential way and by considering only limited amount of data. This work takes three of the approaches proposed earlier - ConvE (Dettmers et al., 2017), DistMult (Yang et al., 2014) and LiteralE (Kristiadi et al., 2018) and tries to scale these approaches so they can learn latent vector representations across nodes by using different Parameter Server Strategies (Synchronous, Asynchronous and Stale-Synchronous) and All-reduce ring architectures for gradient updates while distributing each batch of data across the cluster.

Here we propose our new architecture **ParConvE**, **ParDistMult**, **ParLiteralE** that exploits the advantages of the approach in this paper.

2. Related Work

In recent years, translating embedding methods have played a pivotal role in Knowledge Graph Completion, which usually employ stochastic gradient descent algorithm to optimize a translation based loss function, i.e.,

$$L = \sum_{(se,r,oe)} \sum_{(se',r,oe')} \max[0, f_r(se,oe) + M - f_r(se',oe')]$$

where (se, r, oe) represents the positive triple that exists in the knowledge graph, while (se', r', oe') stands for the negative triple that is not in the knowledge graph. $\max[0, .]$ is the hinge loss, and M is the margin between positive and negative triples.$f_r(se, oe)$ is the score function to determine whether the triple (se, r, oe) should exist in the knowledge graph, which varies from different translating embedding methods.

A significant work is TransE(Bordes et al., 2013) which heralds the start of translating embedding methods. It looks upon a triple (se, r, oe) as a translation from the subject entity se to the object entity oe, i.e., "$se + r \approx oe$", and the score function is $f_r(se, oe) = ||se + r - oe||$, where $||.||$ represents L1-similarity or L2-similarity.

Other neural link prediction models like Bilinear Diagonal model DistMult(Yang et al., 2014)and its extension in the complex space ComplEx(Trouillon et al., 2016). Scoring function for DistMult and ComplEx is $f_r(se, oe) = \langle se, r, oe \rangle$, where $\langle se, r, oe \rangle = \sum_i x_i y_i z_i$ denotes the trilinear dot product.

Other model like the Holographic Embeddings model (HolE)(Nickel et al., 2015), which uses cross-correlation the inverse of circular convolution for matching entity embeddings; it is inspired by holographic models of associative memory. However, HolE does not learn multiple layers of non-linear features, and it is thus theoretically less expressive than ConvE(Dettmers et al., 2017) model.

ConvE(Dettmers et al., 2017), is a model that uses 2D convolutions over embeddings to predict missing links in knowledge graphs. ConvE is the simplest multi-layer convolutional architecture for link prediction: it is defined by a single convolution layer, a projection layer to the embedding dimension, and an inner product layer. Scoring function for ConvE is given by

$$f_r(se, oe) = f(vec(f([\bar{se}; \bar{r}] * w))\mathbf{W})oe$$

where $*$ denotes the convolution operator; f denotes a nonlinear function and $\bar{se}$ and $\bar{r}$ denotes a 2D reshaping se, r respectively.

LiteralE(Kristiadi et al., 2018), is a model that studies the effect of incorporating literal information into existing link prediction methods. Let $L \in R^{N_e * N_l}$ be a matrix, where each entry L_{ik} contains the k^{th} literal value of the i^{th} entity if a triple with the i^{th} entity and the k^{th} data relation exists in the knowledge graph, and zero otherwise. At the core of LiteralE is a function $g : R^{N'_e} * R^{N_l} R^{N'_e}$ that takes entity embeddings N'_e and their literal vectors N_l as inputs and maps them onto another N'_e - dimensional vector. And thus, the original embedding vectors in the scoring function of any latent feature model can be replaced with these literal-enriched vectors.

ParTransX(Zhang et al., 2017) employs parallel embedding method for TransE and TransH. This reduces training time drastically as in this approach they make use of parallelization setup across cluster to train the model. Our model also follows from their path to make ConvE massively parallel and thus ParConvE.

3. Method

We have implemented three approaches for generating knowledge graph at scale by using distribured deep learning. The three methods are **ParConvE**, **ParDistMult** and **ParLiteralE** which is ParConvE + literals incorporation. To train these models we have used two data parallel distributed training aproaches which are:

- Parameter Server Strategy
- Ring All-Reduce Strategy

In **data parallel training**, we expect to reduce overall training time by dividing each training batch among the available workers, and have them process data in parallel. Data parallel training is, however, a strong scaling problem, in that communication is required between the parameter-servers and workers in PS-Strategy and between workers in Ring All-reduce strategy.

Scaling Batch Size & Learning Rate: The batch size limits the amount of parallelism possible in data-parallel training, and therefore we have increased the batch size as more workers are added and also following the linear scaling rule, we have increased the learning rate linearly with the batch size. When the amount of time required to communicate updates in weights between workers/parameter-servers & workers grows linearly, network I/O can quickly become a bottleneck preventing training from scaling further.

Data PipeLine: In data-parallel training, for each training iteration, each worker receives a non-overlapping partition of samples from the batch, for which we have created a highly efficient data pipeline method, by sharding the dataset across all workers, so that dataset is splitted across workers and each worker sees the same set of data in every iterations while shuffling the data at every iteration on worker. Shuffling the splitted-dataset at every iteration on each worker helps to achieve high accuracy.

3.1. Parameter Server Strategy

The first generation of data-parallel distributed training was dominated by the parameter-server architecture. In the parameter-server architecture, one or more parameter servers holds the current model and synchronizes it between a set of worker-nodes for each iteration. The problem with this type of training is that the network links between the parameter server and the workers become a bottleneck. The workers cannot utilize their full bandwidth, while the bandwidth of the parameter server becomes a bottleneck, slowing down training. This problem motivated the Ring-AllReduce technique for training.

Further parameter server strategy can be divided into 3 methods based on the weight update rule - synchronous, asynchronous and stale synchronous training. **We have used all the 3 PS-strategy while training models for ParConvE, ParDistMult and ParLiteralE** and compared them for different combination of number of PS-Workers and batch size and learning rate. We have presented our hypothesis in results section.

3.2. Ring All-Reduce Strategy

In ring-allreduce, each node corresponds to a worker node, see illustration below. During training, the servers work in lockstep processing a large minibatch of training data. Each server computes gradients on its local shard (partition) of the minibatch and each server then both sends and receives gradients to/from their successor neighbor and predecessor neighbor on the ring, in a bandwidth-optimal manner, utilizing each nodes upload and download capacity. All gradients travel in the same direction on the ring, and when all servers have received all the gradients computed for the minibatch, they update the weights for their local copy of the model using an optimization algorithm such as stochastic gradient descent. For a ring with N workers, all workers will have received the gradients necessary to calculate the updated model after N-1 gradient messages are sent and received by each worker.Note that all servers will have the same copy of the model after this update step. In effect, the model is replicated at all servers in the system. Ring-allreduce is bandwidth optimal, as it ensures that the available upload and download network bandwidth at each host is fully utilized (in contrast to the parameter server model). Ring-allreduce can also overlap the computation of gradients at lower layers in a deep neural network with the transmission of gradients at higher layers, further reducing training time.

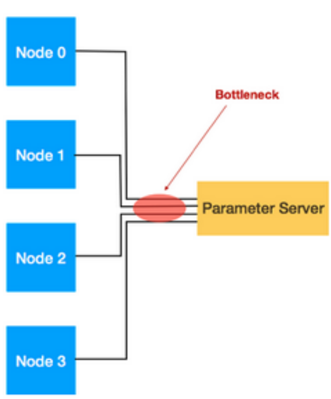

Figure 1. Parameter Server Approach: Parameter Server becomes the bottleneck while aggregating and updating weights from each worker.

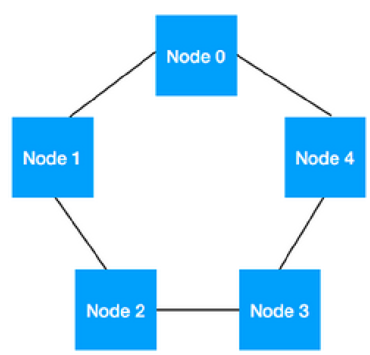

Figure 2. Ring All-Reduce Approach: Ring with N nodes requires N-1 gradient messages to be sent and received.

3.3. ParConvE , ParDistMult & ParLiteralE

These are the three methods which we have used for learning latent vector representations for entities and relations in knowledge graph. Par**X** is based on **X** method with distributed deep learning. **ConvE**(Dettmers et al., 2017) discusses about generating KG embedding using convolutional neural network, while **DistMult**(Yang et al., 2014) uses tri-linear product and **LiteralE**(Kristiadi et al., 2018) incorporates literal information into ConvE.

Let H be the dimension of the latent variables. Further, let $\mathbf{E} \in R^{N_e * H}$ be the entity embedding matrix and $\mathbf{R} \in R^{N_r * H}$ be the relation embedding matrix for $\mathcal{E}$ and $\mathcal{R}$, respectively. Given these embedding matrices, we define a score function $f : R^H * R^H * R^H \to R$ that maps a triples embedding (e_i, r_k, e_j) , where e_i , e_j are the i-th and j-th rows of $\mathbf{E}$ respectively, and r_k is the k-th row of $\mathbf{R}$, to a score $f(e_i, r_k, e_j)$ that correlates with the truth value of the triple. In latent feature methods, the score of any triple $(e_i, r_k, e_j) \in \mathcal{E} \times \mathcal{E} \times \mathcal{R}$ is then defined as $\psi(e_i, r_k, e_j) = f(e_i, r_k, e_j)$.

These methods follow a score-based approach as described above but different kind of scoring functions f. For each method, we scaled the algorithm using data parallel distributed training parameter server and ring all-reduce approaches, while using lazy SGD optimizer, which lazily updates weights by communicating only the non-sparse gradient part for weight updates.

ParConvE: ConvE uses convolution operations to extract features from entity and relation embeddings. Let f be a nonlinear function, ω be convolution filters, and $\mathbf{W}$ be a weight matrix. The ConvE score function is then defined as follows

$$f_{ConvE}(e_i, r_k, e_j) = f(vec(f([e_i; r_k] * \omega))\mathbf{W})e_j$$

where e_i and e_j denote the i-th and j-th row of $\mathbf{E}$ respectively, r_k denotes the k-th row of $\mathbf{R}$.

ParDistMult: The DistMult scoring function is defined as diagonal bilinear interaction between the two entities and the relation embeddings corresponding to a given triple, as follows

$$f_{DistMult}(e_i, r_k, e_j) = \langle e_i, r_k, e_j \rangle = \sum_{h=1}^{H} e_i \odot r_k \odot e_j$$

where e_i and e_j denote the i-th and j-th row of $\mathbf{E}$ respectively, r_k denotes the k-th row of $\mathbf{R}$, and $\odot$ denotes the element-wise multiplication.

ParLiteralE: Let $L \in R^{N_e * N_l}$ be a matrix, where each entry L_{ik} contains the k^{th} literal value of the i^{th} entity if a triple with the i^{th} entity and the k^{th} data relation exists in the knowledge graph, and zero otherwise. At the core of LiteralE is a function $g : R^H * R^{N_l} R^H$ that takes entity embeddings H and their literal vectors N_l as inputs and maps them onto another H - dimensional vector. And thus, the original embedding vectors in the scoring function of any latent feature model can be replaced with these literal-enriched vectors. Thus we replace every entity embedding e_i with $e_i^{lit} = g(e_i, l_i)$ in the scoring functions. Here we'll only consider convE.

$$f_{LiteralE}(e_i, r_k, e_j) = f(vec(f([e_i^{lit}; r_k] * \omega))\mathbf{W})e_j^{lit}$$

Model	Scoring Function	# Parameters
DistMult	$\langle e_i, r_k, e_j \rangle$	$N_e H + N_r H$
ConvE	$f(vec(f([e_i; r_k] * \omega))\mathbf{W})e_j$	$N_e H + N_r H + C$
LiteralE	$f(vec(f([e_i^{lit}; r_k] * \omega))\mathbf{W})e_j^{lit}$	$\Lambda + (H + N_l)H$

Table 1. Model complexity in terms of number of parameters. N_e and N_r respectively denote the number of entities and relation types, H is the dimension of the embeddings, C is the number of parameters of convolution networks of ConvE, Λ denotes the number of overall parameters in ConvE, N_l is the number of literals.

4. Dataset

For evaluating our proposed model, we use a selection of link prediction datasets from the literature. Here we use three benchmark datasets: FB15k, FB15k-237, DB100k.

FB15k(Bollacker et al., 2008) is a subset of Freebase which contains about 14,951 entities with 1,345 different relations. A large fraction of content in this knowledge graph describes facts about movies, actors, awards, sports, and sport teams.

As discussed by Dettmers et al.(Dettmers et al., 2017), FB15k has a large number of test triples which can simply be obtained by inverting training triples. This results in a biased test set, for which a simple model which is symmetric with respect to object and subject entity is capable of achieving excellent results. To address this problem, Toutanova and Chen created **FB15k-237** by removing inverse relations from FB15k. FB15k-237 does not suffer from invertible relation problem, for which KGC problem is more realistic.

DB100k(Ding et al., 2018) is a subset of DBpedia which contains 99,604 entities with 470 different relations.

Triples on each datasets are further divided into training, validation, and test sets, used for model training, hyperparameter tuning, and evaluation respectively. We follow the original split for FB15k, FB15k-237 and DB100k.

Data set	#Relation	#Entity	#Train	#Valid	#Test
FB15k	1,345	14,951	483,142	50,000	59,071
FB15k-237	237	14,541	272,115	17,535	20,466
DB100k	470	99,604	597,572	50,000	50,000

Table 2. Statistics of Datasets used, where the columns respectively indicate the number of relations, entities, train / validation / test triplets.

5. Evaluation

5.1. Automated Metrics

A knowledge graph $G = \{(se, r, oe)\} \subseteq E * R * E$ can be formalised as a set of triples (facts), each consisting of a relationship $r \in R$ and two entities $se, oe \in E$, referred to as the subject and object of the triple. Each triple (se, r, oe) denotes a relationship of type r between the entities se and oe.

The link prediction problem can be formalised as a pointwise learning to rank problem, where the objective is learning a scoring function $f_r(se, oe) : E * R * E \Rightarrow R$. Given an input triple $x = (se, r, oe)$, its score $f_r(se, oe) \in R$ is proportional to the likelihood that the fact encoded by x is true.

To tackle the KGC problem, experiments are conducted on the link prediction task which aims to predict the missing entities se or oe for a triple (se, r, oe). Namely, it predicts oe given (se, r) or predict se given (r, oe).

To evaluate the performance of link prediction, we adopt $MeanRank$ and $Hits@10$. $MeanRank$ is the average rank of the correct entities, and $Hits@10$ is proportion of correct entities ranked in top-10. It is clear that a good predictor has low mean rank and high $Hits@10$.

We now describe the evaluation metrics used for assessing the quality of the models. Let $T = \{x_1, x_2, ..., x_{|T|}\}$ denote the test set. Now, for the i-th test triple x_i in T , we generate all its possible corruptions $C^{se}(x_i)\,(resp. C^{oe}(x_i))-$ obtained by replacing its subject (resp. object) with any other entity in the Knowledge Graph $-$ to check whether the model assigns an higher score to x_i and a lower score to its corruptions. Note that the set of corruptions can also contain several true triples, and it is not a mistake to rank them with an higher score than x_i. The *left* and *right* rank of the i-th test triple each associated to corrupting either the subject or the object according to a model with scoring function $f_r(.)$, are defined as follows:

$$rank_i^{se} = 1 + \sum_{\tilde{x_i} \in C^{se}(x_i) \setminus G} I\left[f_r(x_i) < f_r(\tilde{x_i})\right]$$

$$rank_i^{oe} = 1 + \sum_{\tilde{x_i} \in C^{oe}(x_i) \setminus G} I\left[f_r(x_i) < f_r(\tilde{x_i})\right]$$

where $I[P]$ is 1 iff the condition P is true, and 0 otherwise. For measuring the quality of the ranking, we use the $Hits@k$ metrics, which is defined as follows:

$$Hits@\,k\,(\%) : \frac{100}{2|T|} \sum_{x_i \in T} I\left[rank_i^{se} \leq k\right] + \left[rank_i^{oe} \leq k\right]$$

$Hits@k$ is the percentage of ranks lower than or equal to k: the higher, the better.

5.2. Baseline

Our baseline is primarily to learn knowledge graph embedding using ConvE, DistMult and LiteralE model. We'll compare our model ParConvE, ParDistMult and ParLiteralE for speedup and evaluation metrics against serial model.

Model	Metrics	FB15k	FB15k-237	DB100k
ConvE	MR	51	244	-
	MRR	0.657	0.325	-
	Hits@10	0.831	0.501	-
	Hits@3	0.723	0.356	-
	Hits@1	0.558	0.237	-
DistMult	MR	97	254	-
	MRR	0.654	0.241	0.233
	Hits@10	0.824	0.419	0.448
	Hits@3	0.733	0.263	0.301
	Hits@1	0.546	0.155	0.115

Table 3. Baseline results on test set of FB15k, FB15k-237, DB100k. Results are taken from the original papers. Missing scores in the literature are represented by '-'.

6. Results

Model (Dataset)	# GPU	Asynchronous MRR	Hits@10	Time	SpeedUp
ConvE (FB15k-237)	2	0.379	0.562	1.21	4
	4	0.382	0.562	0.53	4
ConvE (DB100k)	2	0.424	0.602	2.12	4
	4	0.115	0.207	0.88	4
DistMult (FB15k-237)	4	0.255	0.385	0.09	4
LiteralE (FB15k-237)	4	0.379	0.558	0.29	4

Table 7. Parameter Server Approach - MultiGPU

References

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: 10.1145/1376616.1376746. URL http://doi.acm.org/10.1145/1376616.1376746.

Model	Dataset	CPU					GPU				
		MR	MRR	Hits@10	Hits@3	Time	MR	MRR	Hits@10	Hits@3	Time
ConvE	FB15k-237	229	0.382	0.568	0.418	13.09	573	0.385	0.566	0.421	0.85
	FB15k	-	-	-	-	-	318	0.476	0.643	0.523	1.54
	DB100k	1	2	3	4	5	3038	0.477	0.653	0.521	3.77
DistMult	FB15k-237	2892	0.27	0.42	0.29	1.67	2857	0.275	0.42	0.3	0.20
	DB100k	-	-	-	-	-	21831	0.291	0.447	0.334	2.92
LiteralE	FB15k-237	258	0.38	0.558	0.413	39.04	249	0.383	0.562	0.42	1.13
ConvE(2 layers)	FB15k-237	-	-	-	-	-	1091	0.375	0.562	0.411	1.56
ConvE(3 layers)	FB15k-237	-	-	-	-	-	819	0.381	0.562	0.42	1.08

Table 4. Serial Approach

Model (Dataset)	# PS / # Worker	Synchronous				Asynchronous				SSP			
		MRR	Hits @10	Time	Speed Up	MRR	Hits @10	Time	Speed Up	MRR	Hits @10	Time	Speed Up
ConvE (FB15k-237)	1-PS & 4-W	0.357	0.515	3.51	4	0.377	0.56	3.16	4	0.382	0.564	4.5	4
	2-PS & 4-W	0.356	0.509	4.14	4	0.011	0.03	3.85	4	0.378	0.559	3.96	4
	1-PS & 6-W	-	-	-	-	0.375	0.558	1.94	4	0.06	0.07	1.79	4
	2-PS & 6-W	-	-	-	-	0.377	0.558	2.47	4	0.375	0.558	1.92	4
	3-PS & 6-W	-	-	-	-	0.373	0.552	1.45	4	0.372	0.551	2.26	4
ConvE (DB100k)	1-PS & 4-W	0.05	0.08	7.71	4	0.309	0.485	11.07	4	0.003	0.006	7.39	4
DistMult (FB15k-237)	1-PS & 4-W	0.193	0.314	1.88	4	0.335	0.496	1.01	4	0.001	0.002	1.20	4
LiteralE (FB15k-237)	1-PS & 4-W	0.338	0.49	3.35	4	0.381	0.56	3.49	4	0.035	0.063	4.36	4

Table 5. Parameter Server Approach - CPU

Model (Dataset)	# PS / # Worker	Synchronous				Asynchronous				SSP			
		MRR	Hits @10	Time	Speed Up	MRR	Hits @10	Time	Speed Up	MRR	Hits @10	Time	Speed Up
ConvE (FB15k-237)	1-PS & 4-W	0.309	0.438	0.19	4	0.376	0.558	0.39	4	0.376	0.555	0.38	4
ConvE (DB100k)	1-PS & 4-W	0.05	0.09	1.09	4	0.156	0.278	1.69	4	0.05	0.20	1.67	4
DistMult (FB15k-237)	1-PS & 4-W	0.194	0.304	0.24	4	0.332	0.487	0.28	4	0.334	0.487	0.29	4
LiteralE (FB15k-237)	1-PS & 4-W	0.318	0.459	0.27	4	0.376	0.550	0.54	4	0.376	0.552	0.53	4

Table 6. Parameter Server Approach - GPU

Model (Dataset)	# Worker	Parameter Server*				All Reduce			
		MRR	Hits @10	Time	Speed Up	MRR	Hits @10	Time	Speed Up
ConvE (FB15k-237)	4	0.309	0.438	0.19	4	0.366	0.542	0.94	4
ConvE (DB100k)	4	0.05	0.09	1.09	4	0.110	0.2	2.13	4
ConvE (FB15k-237)	6	0.309	0.438	0.19	4	0.329	0.472	0.56	4
DistMult (FB15k-237)	4	0.194	0.304	0.24	4	0.181	0.275	0.25	4
LiteralE (FB15k-237)	4	0.318	0.459	0.27	4	0.312	0.449	0.96	4

Table 8. PS vs. Collective AllReduce - CPU

Model (Dataset)	# Worker	Parameter Server* MRR	Hits @10	Time	Speed Up	All Reduce MRR	Hits @10	Time	Speed Up
ConvE (FB15k-237)	4	0.309	0.438	0.19	4	0.353	0.516	0.14	4
ConvE (DB100k)	4	0.05	0.09	1.09	4	0.115	0.209	0.91	4
DistMult (FB15k-237)	4	0.194	0.304	0.24	4	0.179	0.271	0.09	4
LiteralE (FB15k-237)	4	0.318	0.459	0.27	4	0.349	0.507	0.14	4

Table 9. PS vs. Collective AllReduce - GPU

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2787–2795. Curran Associates, Inc., 2013.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. *CoRR*, abs/1707.01476, 2017. URL http://arxiv.org/abs/1707.01476.

Ding, B., Wang, Q., Wang, B., and Guo, L. Improving knowledge graph embedding using simple constraints. *CoRR*, abs/1805.02408, 2018. URL http://arxiv.org/abs/1805.02408.

Kristiadi, A., Khan, M. A., Lukovnikov, D., Lehmann, J., and Fischer, A. Incorporating literals into knowledge graph embeddings. *CoRR*, abs/1802.00934, 2018. URL http://arxiv.org/abs/1802.00934.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2181–2187. AAAI Press, 2015. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=2886521.2886624.

Nickel, M., Rosasco, L., and Poggio, T. A. Holographic embeddings of knowledge graphs. *CoRR*, abs/1510.04935, 2015. URL http://arxiv.org/abs/1510.04935.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357, 2016. URL http://arxiv.org/abs/1606.06357.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pp. 1112–1119. AAAI Press, 2014. URL http://dl.acm.org/citation.cfm?id=2893873.2894046.

Yang, B., Yih, W., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014. URL http://arxiv.org/abs/1412.6575.

Zhang, D., Li, M., Jia, Y., and Wang, Y. Efficient parallel translating embedding for knowledge graphs. *CoRR*, abs/1703.10316, 2017. URL http://arxiv.org/abs/1703.10316.