The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Convolutional 2D Knowledge Graph Embeddings

Tim Dettmers*
Università della Svizzera italiana
tim.dettmers@gmail.com

Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel
University College London
{p.minervini,p.stenetorp,s.riedel}@cs.ucl.ac.uk

Abstract

Link prediction for knowledge graphs is the task of predicting missing relationships between entities. Previous work on link prediction has focused on shallow, fast models which can scale to large knowledge graphs. However, these models learn less expressive features than deep, multi-layer models – which potentially limits performance. In this work we introduce ConvE, a multi-layer convolutional network model for link prediction, and report state-of-the-art results for several established datasets. We also show that the model is highly parameter efficient, yielding the same performance as DistMult and R-GCN with 8x and 17x fewer parameters. Analysis of our model suggests that it is particularly effective at modelling nodes with high indegree – which are common in highly-connected, complex knowledge graphs such as Freebase and YAGO3. In addition, it has been noted that the WN18 and FB15k datasets suffer from test set leakage, due to inverse relations from the training set being present in the test set – however, the extent of this issue has so far not been quantified. We find this problem to be severe: a simple rule-based model can achieve state-of-the-art results on both WN18 and FB15k. To ensure that models are evaluated on datasets where simply exploiting inverse relations cannot yield competitive results, we investigate and validate several commonly used datasets – deriving robust variants where necessary. We then perform experiments on these robust datasets for our own and several previously proposed models, and find that ConvE achieves state-of-the-art Mean Reciprocal Rank across all datasets.

Introduction

Knowledge graphs are graph-structured knowledge bases, where facts are represented in the form of relationships (edges) between entities (nodes). They have important applications in search, analytics, recommendation, and data integration – however, they tend to suffer from incompleteness, that is, missing links in the graph. For example, in Freebase and DBpedia more than 66% of the person entries are missing a birthplace (Dong et al. 2014; Krompaß, Baier, and Tresp 2015). Identifying such missing links is referred to as *link prediction*. Knowledge 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 (Nickel et al. 2016). DistMult (Yang et al. 2015) 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.

The only way to increase the number of features in shallow models – and thus their expressiveness – is to increase the embedding size. However, doing so does not scale to larger knowledge graphs, since the total number of embedding parameters is proportional to the the number of entities and relations in the graph. For example, a shallow model like DistMult with an embedding size of 200, applied to Freebase, will need 33 GB of memory for its parameters. To increase the number of features independently of the embedding size requires the use of multiple layers of features. However, previous multi-layer knowledge graph embedding architectures, that feature fully connected layers, are prone to overfit (Nickel et al. 2016). 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 (Szegedy et al. 2015; Ioffe and Szegedy 2015; Srivastava et al. 2014; Szegedy et al. 2016).

In this paper we introduce ConvE, 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.

*This work was conducted during a research visit to University College London.

Specifically, our contributions are as follows:

- Introducing a simple, competitive 2D convolutional link prediction model, ConvE.
- Developing a 1-N scoring procedure that speeds up training three-fold and evaluation by 300x.
- Establishing that our model is highly parameter efficient, achieving better scores than DistMult and R-GCNs on FB15k-237 with 8x and 17x fewer parameters.
- Showing that for increasingly complex knowledge graphs, as measured by indegree and PageRank, the difference in performance between our model and a shallow model increases proportionally to the complexity of the graph.
- Systematically investigating reported inverse relations test set leakage across commonly used link prediction datasets, introducing robust versions of datasets where necessary, so that they cannot be solved using simple rule-based models.
- Evaluating ConvE and several previously proposed models on these robust datasets: our model achieves state-of-the-art Mean Reciprocal Rank across all of them.

Related Work

Several neural link prediction models have been proposed in the literature, such as the Translating Embeddings model (TransE) (Bordes et al. 2013a), the Bilinear Diagonal model (DistMult) (Yang et al. 2015) and its extension in the complex space (ComplEx) (Trouillon et al. 2016); we refer to Nickel et al. (2016) for a recent survey. The model that is most closely related to this work is most likely the Holographic Embeddings model (HolE) (Nickel, Rosasco, and Poggio 2016), 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 our model.

To the best of our knowledge, our model is the first neural link prediction model to use 2D convolutional layers. *Graph Convolutional Networks* (GCNs) (Duvenaud et al. 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016) are a related line of research, where the convolution operator is generalised to use locality information in graphs. However, the GCN framework is limited to undirected graphs, while knowledge graphs are naturally directed, and suffers from potentially prohibitive memory requirements (Kipf and Welling 2016). Relational GCNs (R-GCNs) (Schlichtkrull et al. 2017) are a generalisation of GCNs developed for dealing with highly multi-relational data such as knowledge graphs – we include them in our experimental evaluations.

Several convolutional models have been proposed in natural language processing (NLP) for solving a variety of tasks, including semantic parsing (Yih et al. 2011), sentence classification (Kim 2014), search query retrieval (Shen et al. 2014), sentence modelling (Kalchbrenner, Grefenstette, and Blunsom 2014), as well as other NLP tasks (Collobert et al. 2011). However, most work in NLP uses 1D-convolutions, that is convolutions which operate over a temporal sequence of embeddings, for example a sequence of words in embedding space. In this work, we use 2D-convolutions which operate spatially over embeddings.

Number of Interactions for 1D vs 2D Convolutions

Using 2D rather than 1D convolutions increases the expressiveness of our model through additional points of interaction between embeddings. For example, consider the case where we concatenate two rows of 1D embeddings, a and b with dimension $n = 3$:

$$([a \quad a \quad a]\,;[b \quad b \quad b]) = [a \quad a \quad a \quad b \quad b \quad b]\,.$$

A padded 1D convolution with filter size $k = 3$ will be able to model the interactions between these two embeddings around the concatenation point (with a number of interactions proportional to k).

If we concatenate (i.e. stack) two rows of 2D embeddings with dimension $m \times n$, where $m = 2$ and $n = 3$, we obtain the following:

$$\left(\begin{bmatrix} a & a & a \\ a & a & a \end{bmatrix}; \begin{bmatrix} b & b & b \\ b & b & b \end{bmatrix}\right) = \begin{bmatrix} a & a & a \\ a & a & a \\ b & b & b \\ b & b & b \end{bmatrix}.$$

A padded 2D convolution with filter size 3×3 will be able to model the interactions around the entire concatenation line (with a number of interactions proportional to n and k).

We can extend this principle to an alternating pattern, such as the following:

$$\begin{bmatrix} a & a & a \\ b & b & b \\ a & a & a \\ b & b & b \end{bmatrix}.$$

In this case, a 2D convolution operation is able to model even more interactions between a and b (with a number of interactions proportional to m, n, and k). Thus, 2D convolution is able to extract more feature interactions between two embeddings compared to 1D convolution. The same principle can be extending to higher dimensional convolutions, but we leave this as future work.

Background

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

The *link prediction* problem can be formalised as a pointwise learning to rank problem, where the objective is learning a scoring function $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto \mathbb{R}$. Given an input triple $x = (s, r, o)$, its score $\psi(x) \in \mathbb{R}$ is proportional to the likelihood that the fact encoded by x is true.

Neural Link Predictors

Neural link prediction models (Nickel et al. 2016) can be seen as multi-layer neural networks consisting of an *encoding component* and a *scoring component*. Given an input triple (s, r, o), the encoding component maps entities $s, o \in \mathcal{E}$ to their distributed embedding representations $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$. In

Table 1: Scoring functions $\psi_r(\mathbf{e}_s, \mathbf{e}_o)$ from neural link predictors in the literature, their relation-dependent parameters and space complexity; n_e and n_r respectively denote the number of entities and relation types, i.e. $n_e = |\mathcal{E}|$ and $n_r = |\mathcal{R}|$.

Model	Scoring Function $\psi_r(\mathbf{e}_s, \mathbf{e}_o)$	Relation Parameters	Space Complexity
SE (Bordes et al. 2014)	$\left\|\mathbf{W}_r^L \mathbf{e}_s - \mathbf{W}_r^R \mathbf{e}_o\right\|_p$	$\mathbf{W}_r^L, \mathbf{W}_r^R \in \mathbb{R}^{k \times k}$	$\mathcal{O}(n_e k + n_r k^2)$
TransE (Bordes et al. 2013a)	$\left\|\mathbf{e}_s + \mathbf{r}_r - \mathbf{e}_o\right\|_p$	$\mathbf{r}_r \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
DistMult (Yang et al. 2015)	$\langle \mathbf{e}_s, \mathbf{r}_r, \mathbf{e}_o \rangle$	$\mathbf{r}_r \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
ComplEx (Trouillon et al. 2016)	$\langle \mathbf{e}_s, \mathbf{r}_r, \mathbf{e}_o \rangle$	$\mathbf{r}_r \in \mathbb{C}^k$	$\mathcal{O}(n_e k + n_r k)$
ConvE	$f(\mathrm{vec}(f([\overline{\mathbf{e}_s}; \overline{\mathbf{r}_r}] * \omega))\mathbf{W})\mathbf{e}_o$	$\mathbf{r}_r \in \mathbb{R}^{k'}$	$\mathcal{O}(n_e k + n_r k')$

the scoring component, the two entity embeddings $\mathbf{e}_s$ and $\mathbf{e}_o$ are scored by a function ψ_r. The score of a triple (s, r, o) is defined as $\psi(s, r, o) = \psi_r(\mathbf{e}_s, \mathbf{e}_o) \in \mathbb{R}$.

In Table 1 we summarise the scoring function of several link prediction models from the literature. The vectors $\mathbf{e}_s$ and $\mathbf{e}_o$ denote the subject and object embedding, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{C}^k$ in ComplEx and $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$ in all other models, and $\langle x, y, z \rangle = \sum_i x_i y_i z_i$ denotes the tri-linear dot product; $*$ denotes the convolution operator; f denotes a non-linear function.

Convolutional 2D Knowledge Graphs Embeddings

In this work we propose a neural link prediction model where the interactions between input entities and relationships are modelled by convolutional and fully-connected layers. The main characteristic of our model is that the score is defined by a convolution over 2D shaped embeddings. The architecture is summarised in Figure 1; formally, the scoring function is defined as follows:

$$\psi_r(\mathbf{e}_s, \mathbf{e}_o) = f(\mathrm{vec}(f([\overline{\mathbf{e}_s}; \overline{\mathbf{r}_r}] * \omega))\mathbf{W})\mathbf{e}_o, \quad (1)$$

where $\mathbf{r}_r \in \mathbb{R}^k$ is a relation parameter depending on r, $\overline{\mathbf{e}_s}$ and $\overline{\mathbf{r}_r}$ denote a 2D reshaping of $\mathbf{e}_s$ and $\mathbf{r}_r$, respectively: if $\mathbf{e}_s, \mathbf{r}_r \in \mathbb{R}^k$, then $\overline{\mathbf{e}_s}, \overline{\mathbf{r}_r} \in \mathbb{R}^{k_w \times k_h}$, where $k = k_w k_h$.

In the feed-forward pass, the model performs a row-vector look-up operation on two embedding matrices, one for entities, denoted $\mathbf{E}^{|\mathcal{E}| \times k}$ and one for relations, denoted $\mathbf{R}^{|\mathcal{R}| \times k'}$, where k and k' are the entity and relation embedding dimensions, and $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the number of entities and relations. The model then concatenates $\overline{\mathbf{e}_s}$ and $\overline{\mathbf{r}_r}$, and uses it as an input for a 2D convolutional layer with filters ω. Such a layer returns a feature map tensor $\mathcal{T} \in \mathbb{R}^{c \times m \times n}$, where c is the number of 2D feature maps with dimensions m and n. The tensor $\mathcal{T}$ is then reshaped into a vector $\mathrm{vec}(\mathcal{T}) \in \mathbb{R}^{cmn}$, which is then projected into a k-dimensional space using a linear transformation parametrised by the matrix $\mathbf{W} \in \mathbb{R}^{cmn \times k}$ and matched with the object embedding $\mathbf{e}_o$ via an inner product. The parameters of the convolutional filters and the matrix $\mathbf{W}$ are independent of the parameters for the entities s and o and the relationship r.

For training the model parameters, we apply the logistic sigmoid function $\sigma(\cdot)$ to the scores, that is $p = \sigma(\psi_r(\mathbf{e}_s, \mathbf{e}_o))$, and minimise the following binary cross-entropy loss:

$$\mathcal{L}(p, t) = -\frac{1}{N} \sum_i (t_i \cdot \log(p_i) + (1 - t_i) \cdot \log(1 - p_i)), \quad (2)$$

where t is the label vector with dimension $\mathbb{R}^{1x1}$ for 1-1 scoring or $\mathbb{R}^{1xN}$ for 1-N scoring (see the next section for 1-N scoring); the elements of vector t are ones for relationships that exists and zero otherwise.

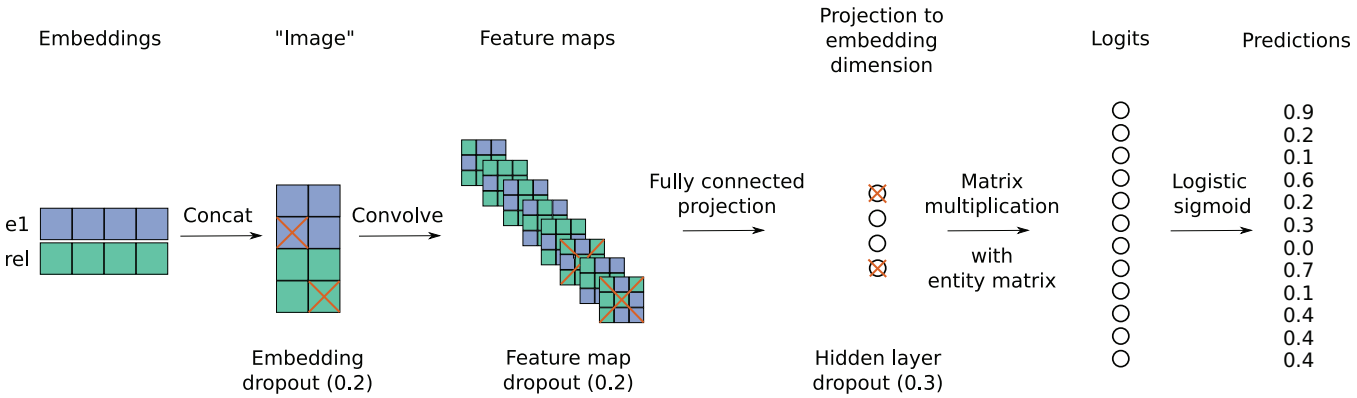
We use rectified linear units as the non-linearity f for faster training (Krizhevsky, Sutskever, and Hinton 2012), and batch normalisation after each layer to stabilise, regularise and increase rate of convergence (Ioffe and Szegedy 2015). We regularise our model by using dropout (Srivastava et al. 2014) in several stages. In particular, we use dropout on the embeddings, on the feature maps after the convolution operation, and on the hidden units after the fully connected layer. We use Adam as optimiser (Kingma and Ba 2014), and label smoothing to lessen overfitting due to saturation of output non-linearities at the labels (Szegedy et al. 2016).

Fast Evaluation for Link Prediction Tasks

In our architecture convolution consumes about 75-90% of the total computation time, thus it is important to minimise the number of convolution operations to speed up computation as much as possible. For link prediction models, the batch size is usually increased to speed up evaluation (Bordes et al. 2013b). However, this is not feasible for convolutional models since the memory requirements quickly outgrow the GPU memory capacity when increasing the batch size.

Unlike other link prediction models which take an entity pair and a relation as a triple (s, r, o), and score it (1-1 scoring), we take one (s, r) pair and score it against all entities $o \in \mathcal{E}$ simultaneously (1-N scoring). If we benchmark 1-1 scoring on a high-end GPU with batch size and embedding size 128, then a training pass and an evaluation with a convolution model on FB15k – one of the dataset used in the experiments – takes 2.4 minutes and 3.34 hours. Using 1-N scoring, the respective numbers are 45 and 35 seconds – a considerable improvement of over 300x in terms of evaluation time. Additionally, this approach is scalable to large knowledge graphs and increases convergence speed. For a single forward-backward pass with batch size of 128, going from $N = 100,000$ to $N = 1,000,000$ entities only increases the computational time from 64ms to 80ms – in other words, a ten-fold increase in the number of entities only increases the computation time by 25% – which attests the scalability of the approach.

Figure 1: In the ConvE model, the entity and relation embeddings are first reshaped and concatenated (steps 1, 2); the resulting matrix is then used as input to a convolutional layer (step 3); the resulting feature map tensor is vectorised and projected into a k-dimensional space (step 4) and matched with all candidate object embeddings (step 5).



If instead of 1-N scoring, we use 1-(0.1N) scoring – that is, scoring against 10% of the entities – we can compute a forward-backward pass 25% faster. However, we converge roughly 230% slower on the training set. Thus 1-N scoring has an additional effect which is akin to batch normalisation (Ioffe and Szegedy 2015) – we trade some computational performance for greatly increased convergence speed and also achieve better performance as shown in Section 7. Do note that the technique in general could by applied to any 1-1 scoring model. This practical trick in speeding up training and evaluation can be applied to any 1-1 scoring model, such as the great majority of link prediction models.

Experiments

Knowledge Graph Datasets

For evaluating our proposed model, we use a selection of link prediction datasets from the literature.

WN18 (Bordes et al. 2013a) is a subset of WordNet which consists of 18 relations and 40,943 entities. Most of the 151,442 triples consist of hyponym and hypernym relations and, for such a reason, WN18 tends to follow a strictly hierarchical structure.

FB15k (Bordes et al. 2013a) 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.

YAGO3-10 (Mahdisoltani, Biega, and Suchanek 2015) is a subset of YAGO3 which consists of entities which have a minimum of 10 relations each. It has 123,182 entities and 37 relations. Most of the triples deal with descriptive attributes of people, such as citizenship, gender, and profession.

Countries (Bouchard, Singh, and Trouillon 2015) is a benchmark dataset that is useful to evaluate a model's ability to learn long-range dependencies between entities and relations. It consists of three sub-tasks which increase in difficulty in a step-wise fashion, where the minimum path-length to find a solution increases from 2 to 4.

It was first noted by Toutanova and Chen (2015) that WN18 and FB15k suffer from test leakage through inverse relations: a large number of test triples can be obtained simply by inverting triples in the training set. For example, the test set frequently contains triples such as $(s, \text{hyponym}, o)$ while the training set contains its inverse $(o, \text{hypernym}, s)$. To create a dataset without this property, Toutanova and Chen (2015) introduced FB15k-237 – a subset of FB15k where inverse relations are removed. However, they did not explicitly investigate the severity of this problem, which might explain why research continues using these datasets for evaluation without addressing this issue (e.g. Trouillon et al. (2016), Nickel, Rosasco, and Poggio (2016), Nguyen et al. (2016), Liu et al. (2016)).

In the following section, we introduce a simple rule-based model which demonstrates the severity of this bias by achieving state-of-the-art results on both WN18 and FB15k. In order to ensure that we evaluate on datasets that do not have inverse relation test leakage, we apply our simple rule-based model to each dataset. Apart from FB15k, which was corrected by FB15k-237, we also find flaws with WN18. We thus create WN18RR to reclaim WN18 as a dataset, which cannot easily be completed using a single rule – but requires modelling of the complete knowledge graph. WN18RR[1] contains 93,003 triples with 40,943 entities and 11 relations. For future research, we recommend against using FB15k and WN18 and instead recommend FB15k-237, WN18RR, and YAGO3-10.

Experimental Setup

We selected the hyperparameters of our ConvE model via grid search according to the mean reciprocal rank (MRR) on the validation set. Hyperparameter ranges for the grid search were as follows – embedding dropout $\{0.0, 0.1, 0.2\}$, feature map dropout $\{0.0, 0.1, 0.2, 0.3\}$, projection layer dropout $\{0.0, 0.1, 0.3, 0.5\}$, embedding size $\{100, 200\}$, batch size $\{64, 128, 256\}$, learning rate $\{0.001, 0.003\}$, and label smoothing $\{0.0, 0.1, 0.2, 0.3\}$.

Besides the grid search, we investigated modifications of the 2D convolution layer for our models. In particular, we

[1] https://github.com/TimDettmers/ConvE

Table 2: Parameter scaling of DistMult vs ConvE.

Model	Param. count	Emb. size	MRR	Hits @10	Hits @3	Hits @1
DistMult	1.89M	128	.23	.41	.25	.15
DistMult	0.95M	64	.22	.39	.25	.14
DistMult	0.23M	16	.16	.31	.17	.09
ConvE	5.05M	200	.32	.49	.35	.23
ConvE	1.89M	96	.32	.49	.35	.23
ConvE	0.95M	54	.30	.46	.33	.22
ConvE	0.46M	28	.28	.43	.30	.20
ConvE	0.23M	14	.26	.40	.28	.19

experimented with replacing it with fully connected layers and 1D convolution; however, these modifications consistently reduced the predictive accuracy of the model. We also experimented with different filter sizes, and found that we only receive good results if the first convolutional layer uses small (i.e. 3x3) filters.

We found that the following combination of parameters works well on WN18, YAGO3-10 and FB15k: embedding dropout 0.2, feature map dropout 0.2, projection layer dropout 0.3, embedding size 200, batch size 128, learning rate 0.001, and label smoothing 0.1. For the Countries dataset, we increase embedding dropout to 0.3, hidden dropout to 0.5, and set label smoothing to 0. We use early stopping according to the mean reciprocal rank (WN18, FB15k, YAGO3-10) and AUC-PR (Countries) statistics on the validation set, which we evaluate every three epochs. Unlike the other datasets, for Countries the results have a high variance, as such we average 10 runs and produce 95% confidence intervals. For our DistMult and ComplEx results with 1-1 training, we use an embedding size of 100, AdaGrad (Duchi, Hazan, and Singer 2011) for optimisation, and we regularise our model by forcing the entity embeddings to have a L2 norm of 1 after each parameter update. As in Bordes et al. (2013a), we use a pairwise margin-based ranking loss.

The code for our model and experiments is made publicly available,[2] as well as the code for replicating the DistMult results.[3]

Inverse Model

It has been noted by Toutanova and Chen (2015), that the training datasets of WN18 and FB15k have 94% and 81% test leakage as inverse relations, that is, 94% and 81% of the triples in these datasets have inverse relations which are linked to the test set. For instance, a test triple *(feline, hyponym, cat)* can easily be mapped to a training triple *(cat, hypernym, feline)* if it is known that hyponym is the inverse of hypernym. This is highly problematic, because link predictors that do well on these datasets may simply learn which relations that are the inverse of others, rather than to model the actual knowledge graph.

[2]https://github.com/TimDettmers/ConvE

[3]https://github.com/uclmr/inferbeddings

To gauge the severity of this problem, we construct a simple, rule-based model that solely models inverse relations. We call this model the *inverse model*. The model extracts inverse relationships automatically from the training set: given two relation pairs $r_1, r_2 \in \mathcal{R}$, we check whether (s, r_1, o) implies (o, r_2, s), or vice-versa.

We assume that inverse relations are randomly distributed among the training, validation and test sets and, as such, we expect the number of inverse relations to be proportional to the size of the training set compared to the total dataset size. Thus, we detect inverse relations if the presence of (s, r_1, o) co-occurs with the presence of (o, r_2, s) with a frequency of at least $0.99 - (f_v + f_t)$, where f_v and f_t is the fraction of the validation and test set compared to the total size of the dataset. Relations matching this criterion are assumed to be the inverse of each other.

At test time, we check if the test triple has inverse matches outside the test set: if k matches are found, we sample a permutation of the top k ranks for these matches; if no match is found, we select a random rank for the test triple.

Results

Similarly to previous work (Yang et al. 2015; Trouillon et al. 2016; Niepert 2016), we report results using a filtered setting, i.e. we rank test triples against all other candidate triples not appearing in the training, validation, or test set (Bordes et al. 2013a). Candidates are obtained by permuting either the subject or the object of a test triple with all entities in the knowledge graph. Our results on the standard benchmarks FB15k and WN18 are shown in Table 3; results on the datasets with inverse relations removed are shown in Table 4; results on YAGO3-10 and Countries are shown in Table 5.

Strikingly, the inverse model achieves state-of-the-art on many different metrics for both FB15k and WN18. However, it fails to pick up on inverse relations for YAGO3-10 and FB15k-237. The procedure used by Toutanova and Chen (2015) to derive FB15k-237 does not remove certain symmetric relationships, for example "similar to". The presence of these relationships explains the good score of our inverse model on WN18RR, which was derived using the same procedure.

Our proposed model, ConvE, achieves state-of-the-art performance for all metrics on YAGO3-10, for some metrics on FB15k, and it does well on WN18. On Countries, it solves the S1 and S2 tasks, and does well on S3, scoring better than other models like DistMult and ComplEx.

For FB15k-237, we could not replicate the basic model results from Toutanova et al. (2015), where the models in general have better performance than what we can achieve. Compared to Schlichtkrull et al. (2017), our results for standard models are a slightly better then theirs, and on-a-par with their R-GCN model.

Parameter efficiency of ConvE

From Table 2 we can see that ConvE for FB15k-237 with 0.23M parameters performs better than DistMult with 1.89M parameters for 3 metrics out of 5.

ConvE with 0.46M parameters still achieves state-of-the-art results on FB15k-237 with 0.425 Hits@10. Comparing to

Table 3: Link prediction results for WN18 and FB15k

	WN18					FB15k				
			Hits					Hits		
	MR	MRR	@10	@3	@1	MR	MRR	@10	@3	@1
DistMult (Yang et al. 2015)	902	.822	.936	.914	.728	97	.654	.824	.733	.546
ComplEx (Trouillon et al. 2016)	–	.941	.947	.936	.936	–	.692	.840	.759	.599
Gaifman (Niepert 2016)	**352**	–	.939	–	.761	75	–	.842	–	.692
ANALOGY (Liu, Wu, and Yang 2017)	–	**.942**	.947	.944	**.939**	–	.725	.854	.785	.646
R-GCN (Schlichtkrull et al. 2017)	–	.814	.964	.929	.697	–	.696	.842	.760	.601
ConvE	504	**.942**	.955	.947	.935	**64**	**.745**	**.873**	**.801**	.670
Inverse Model	567	.861	**.969**	**.968**	.764	1897	.706	.737	.718	**.689**

Table 4: Link prediction results for WN18RR and FB15k-237

	WN18RR					FB15k-237				
			Hits					Hits		
	MR	MRR	@10	@3	@1	MR	MRR	@10	@3	@1
DistMult (Yang et al. 2015)	**5110**	.43	.49	.44	.39	254	.241	.419	.263	.155
ComplEx (Trouillon et al. 2016)	5261	.44	**.51**	**.46**	**.41**	339	.247	.428	.275	.158
R-GCN (Schlichtkrull et al. 2017)	–	–	–	–	–	–	.248	.417	.258	.153
ConvE	5277	**.46**	.48	.43	.39	**246**	**.316**	**.491**	**.350**	**.239**
Inverse Model	13219	.36	.36	.36	.36	7148	.009	.012	.010	.006

the previous best model, R-GCN (Schlichtkrull et al. 2017), which achieves 0.417 Hits@10 with more than 8M parameters.

Overall, ConvE is more than 17x parameter efficient than R-GCNs, and 8x more parameter efficient than DistMult. For the entirety of Freebase, the size of these models would be more than 82GB for R-GCNs, 21GB for DistMult, compared to 5.2GB for ConvE.

Analysis

Ablation Study

Table 7 shows the results from our ablation study where we evaluate different parameter initialisation ($n = 2$) to calculate confidence intervals. We see that hidden dropout is by far the most important component, which is unsurprising since it is our main regularisation technique. 1-N scoring improves performance, as does input dropout, feature map dropout has a minor effect, while label smoothing seems to be unimportant – as good results can be achieved without it.

Analysis of Indegree and PageRank

Our main hypothesis for the good performance of our model on datasets like YAGO3-10 and FB15k-237 compared to WN18RR, is that these datasets contain nodes with very high relation-specific indegree. For example the node "United States" with edges "was born in" has an indegree of over 10,000. Many of these 10,000 nodes will be very different from each other (actors, writers, academics, politicians, business people) and successful modelling of such a high indegree nodes requires capturing all these differences. Our hypothesis is that deeper models, that is, models that learn multiple layers of features, like ConvE, have an advantage over shallow models, like DistMult, to capture all these constraints.

However, deeper models are more difficult to optimise, so we hypothesise that for datasets with low average relation-specific indegree (like WN18RR and WN18), a shallow model like DistMult might suffice for accurately representing the structure of the network.

To test our two hypotheses, we take two datasets with low (low-WN18) and high (high-FB15k) relation-specific indegree and reverse them into high (high-WN18) and low (low-FB15k) relation-specific indegree datasets by deleting low and high indegree nodes. We hypothesise that, compared to DistMult, ConvE will always do better on the dataset with high relation-specific indegree, and vice-versa.

Indeed, we find that both hypotheses hold: for low-FB15k we have ConvE 0.586 Hits@10 vs DistMult 0.728 Hits@10; for high-WN18 we have ConvE 0.952 Hits@10 vs DistMult 0.938 Hits@10. This supports our hypothesis that deeper models such as ConvE have an advantage to model more complex graphs (e.g. FB15k and FB15k-237), but that shallow models such as DistMult have an advantage to model less complex graphs (e.g. WN18 WN18RR).

To investigate this further, we look at PageRank, a measure of centrality of a node. PageRank can also be seen as a measure of the recursive indegree of a node: the PageRank value of a node is proportional to the indegree of this node, its

Table 5: Link prediction results for YAGO3-10 and Countries

	YAGO3-10					Countries		
			Hits			AUC-PR		
	MR	MRR	@10	@3	@1	S1	S2	S3
DistMult (Yang et al. 2015)	5926	.34	.54	.38	.24	**1.00±0.00**	0.72±0.12	0.52±0.07
ComplEx (Trouillon et al. 2016)	6351	.36	.55	.40	.26	0.97±0.02	0.57±0.10	0.43±0.07
ConvE	**2792**	**.52**	**.66**	**.56**	**.45**	**1.00±0.00**	**0.99±0.01**	**0.86 ±0.05**
Inverse Model	60251	.02	.02	.02	.01	–	–	–

Table 6: Mean PageRank $\times 10^{-3}$ of nodes in the test set vs reduction in error in terms of AUC-PR or Hits@10 of ConvE wrt. DistMult.

Dataset	PageRank	Error Reduction
WN18RR	0.104	0.91
WN18	0.125	1.28
FB15k	0.599	1.23
FB15-237	0.733	1.17
YAGO3-10	0.988	1.91
Countries S3	1.415	3.36
Countries S1	1.711	0.00
Countries S2	1.796	18.6

Table 7: Ablation study for FB15k-237.

Ablation	Hits@10
Full ConvE	0.491
Hidden dropout	-0.044 ± 0.003
Input dropout	-0.022 ± 0.000
1-N scoring	-0.019
Feature map dropout	-0.013 ± 0.001
Label smoothing	-0.008 ± 0.000

neighbours indegrees, its neighbours-neighbours indegrees and so forth scaled relative to all other nodes in the network. By this line of reasoning, we also expect ConvE to be better than DistMult on datasets with high average PageRank (high connectivity graphs), and vice-versa.

To test this hypothesis, we calculate the PageRank for each dataset as a measure of centrality. We find that the most central nodes in WN18 have a PageRank value more than one order of magnitude smaller than the most central nodes in YAGO3-10 and Countries, and about 4 times smaller than the most central nodes in FB15k. When we look at the mean PageRank of nodes contained in the test sets, we find that the difference of performance in terms of Hits@10 between DistMult and ConvE is roughly proportional to the mean test set PageRank, that is, the higher the mean PageRank of the test set nodes the better ConvE does compared to DistMult, and vice-versa. See Table 6 for these statistics. The correlation between mean test set PageRank and relative error reduction of ConvE compared to DistMult is strong with $r = 0.83$. This gives additional evidence that models that are deeper have an advantage when modelling nodes with high (recursive) indegree.

From this evidence we conclude, that the increased performance of our model compared to a standard link predictor, DistMult, can be partially explained due to our it's ability to model nodes with high indegree with greater precision – which is possibly related to its depth.

Conclusion and Future Work

We introduced ConvE, a link prediction model that uses 2D convolution over embeddings and multiple layers of non-linear features to model knowledge graphs. ConvE uses fewer parameters; it is fast through 1-N scoring; it is expressive through multiple layers of non-linear features; it is robust to overfitting due to batch normalisation and dropout; and achieves state-of-the-art results on several datasets, while still scaling to large knowledge graphs. In our analysis, we show that the performance of ConvE compared to a common link predictor, DistMult, can partially be explained by its ability to model nodes with high (recursive) indegree.

Test leakage through inverse relations of WN18 and FB15k was first reported by Toutanova and Chen (2015): we investigate the severity of this problem for commonly used datasets by introducing a simple rule-based model, and find that it can achieve state-of-the-art results on WN18 and FB15k. To ensure robust versions of all investigated datasets exists, we derive WN18RR.

Our model is still shallow compared to convolutional architecture found in computer vision, and future work might deal with convolutional models of increasing depth. Further work might also look at the interpretation of 2D convolution, or how to enforce large-scale structure in embedding space so to increase the number of interactions between embeddings.

Acknowledgments

We would like to thank Johannes Welbl and Peter Hayes for their feedback and helpful discussions related to this work. This work was supported by a Marie Curie Career Integration Award, an Allen Distinguished Investigator Award, a Google Europe Scholarship for Students with Disabilities, and the H2020 project SUMMA.

References

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013a. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of NIPS*, 2787–2795.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013b. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of NIPS*, 2787–2795.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. *Machine Learning* 94(2):233–259.

Bouchard, G.; Singh, S.; and Trouillon, T. 2015. On approximate reasoning capabilities of low-rank vector spaces. *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* 12:2493–2537.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proceedings of NIPS*, 3837–3845.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of KDD 2014*, 601–610.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Duvenaud, D. K.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Proceedings of NIPS 2015*, 2224–2232.

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of ACL 2014, Volume 1: Long Papers*, 655–665.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP 2014*, 1746–1751.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR 2016*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of NIPS 2012*, 1097–1105.

Krompaß, D.; Baier, S.; and Tresp, V. 2015. Type-Constrained Representation Learning in Knowledge Graphs. In *Proceedings of ISWC 2015*, 640–655.

Liu, Q.; Jiang, L.; Han, M.; Liu, Y.; and Qin, Z. 2016. Hierarchical random walk inference in knowledge graphs. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 445–454. ACM.

Liu, H.; Wu, Y.; and Yang, Y. 2017. Analogical Inference for Multi-Relational Embeddings. *ArXiv e-prints*.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. M. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Proceedings of CIDR 2015*.

Nguyen, D. Q.; Sirts, K.; Qu, L.; and Johnson, M. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE* 104(1):11–33.

Nickel, M.; Rosasco, L.; and Poggio, T. A. 2016. Holographic Embeddings of Knowledge Graphs. In *Proceedings of AAAI*, 1955–1961.

Niepert, M. 2016. Discriminative Gaifman Models. In *Proceedings of NIPS 2016*, 3405–3413.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2017. Modeling Relational Data with Graph Convolutional Networks. *arXiv preprint arXiv:1703.06103*.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of WWW 2014*, 373–374.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of IEEE CVPR*, 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of IEEE CVPR*, 2818–2826.

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

Toutanova, K.; Chen, D.; Pantel, P.; Poon, H.; Choudhury, P.; and Gamon, M. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of EMNLP 2015*, volume 15, 1499–1509.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of ICML 2016*, 2071–2080.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of ICLR 2015*.

Yih, W.; Toutanova, K.; Platt, J. C.; and Meek, C. 2011. Learning Discriminative Projections for Text Similarity Measures. In *Proceedings of CoNLL 2011*, 247–256.