Neural Tensor Networks with Diagonal Slice Matrices

Takahiro Ishihara[1] **Katsuhiko Hayashi**[2] **Hitoshi Manabe**[1]
Masashi Shimbo[1] **Masaaki Nagata**[3]
[1] Nara Institute of Science and Technology [2] Osaka University
[3] NTT Communication Science Laboratories
[1] {ishihara.takahiro.in0, manabe.hitoshi.me0, shimbo}@is.naist.jp
[2] khayashi0201@gmail.com [3] nagata.masaaki@lab.ntt.co.jp

Abstract

Although neural tensor networks (NTNs) have been successful in many natural language processing tasks, they require a large number of parameters to be estimated, which often results in overfitting and long training times. We address these issues by applying eigendecomposition to each slice matrix of a tensor to reduce the number of parameters. We evaluate our proposed NTN models in two tasks. First, the proposed models are evaluated in a knowledge graph completion task. Second, a recursive NTN (RNTN) extension of the proposed models is evaluated on a logical reasoning task. The experimental results show that our proposed models learn better and faster than the original (R)NTNs.

1 Introduction

Alongside the nonlinear activation functions, linear mapping by matrix multiplication is an essential component of neural network (NN) models, as it determines the feature interaction and thus the expressiveness of models. In addition to the matrix-based mapping, neural tensor networks (NTNs) (Socher et al., 2013a) employ a 3-dimensional tensor to capture direct interactions among input features. Due to the large expressive capacity of 3D tensors, NTNs have been successful in an array of natural language processing (NLP) and machine learning tasks, including knowledge graph completion (KGC) (Socher et al., 2013a), sentiment analysis (Socher et al., 2013b), and reasoning with logical semantics (Bowman et al., 2015). However, since a 3D tensor has a large number of parameters, NTNs need longer time to train than other NN models. Moreover, the millions of parameters often make the model suffer from overfitting (Yang et al., 2015).

To solve these problems, we propose two new parameter reduction techniques for NTNs. These techniques drastically decrease the number of parameters in an NTN without diminishing its expressiveness. We use the matrix decomposition techniques that are utilized for KGC in Yang et al. (2015) and Trouillon et al. (2016). Yang et al. (2015) imposed a constraint that a matrix in the bilinear term in their model had to be diagonal. As mentioned in a subsequent section, this is essentially equal to assuming that the matrix be symmetric and performing eigendecomposition. Trouillon et al. (2016) also applied eigendecomposition to a matrix by regarding it as the real part of a normal matrix. Following these studies, we perform simultaneous diagonalization on all slice matrices of a NTN tensor. As a result, mapping by a 3D ($n \times n \times k$) tensor is replaced with an array of k "triple inner products" of two input vectors and a weight vector. Thus, we obtain two new NTN models where the number of parameters is reduced from $O(n^2k)$ to $O(nk)$.

On a KGC task, these parameter-reduced NTNs (NTN-Diag and NTN-Comp) alleviate overfitting and outperform the original NTN. Moreover, our proposed NTNs can learn faster than the original NTN. We also show that our proposed models perform better and learn faster in a recursive setting by examining a logical reasoning task.

2 Background

We consider mapping in a neural network (NN) layer that takes two vectors as input, such as recursive neural networks. Recurrent neural networks also has this structure, with one input vector being the hidden state from the previous time step. As a mapping before activation in the NN layer, linear mapping (matrix multiplication) is commonly used:

$$\boldsymbol{W}_1\boldsymbol{x}_1 + \boldsymbol{W}_2\boldsymbol{x}_2 = [\boldsymbol{W}_1, \boldsymbol{W}_2]\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} = \boldsymbol{W}\boldsymbol{x}.$$

Here, since $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n, \boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{k \times n}$, this linear mapping is a transformation from $\mathbb{R}^{2n}$ to $\mathbb{R}^k$. Linear mapping, which is a standard component of NNs, has been applied successfully in many tasks. However, it cannot consider the interaction between different components of two input vectors, which renders it not ideal for modeling complex compositional structures such as trees and graphs.

To alleviate this problem, some models such as NTNs (Socher et al., 2013a) have explored 3D tensors to yield more expressive mapping:

$$\begin{aligned}
\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[1:k]} \boldsymbol{x}_2 &= \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[1]} \boldsymbol{x}_2 \\ \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[2]} \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[k]} \boldsymbol{x}_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathrm{sum}\left(\boldsymbol{W}^{[1]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \\ \mathrm{sum}\left(\boldsymbol{W}^{[2]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \\ \vdots \\ \mathrm{sum}\left(\boldsymbol{W}^{[k]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)\right) \end{bmatrix}
\end{aligned}$$

where $\boldsymbol{W}^{[1:k]} \in \mathbb{R}^{n \times n \times k}$. The output of this mapping is an array of k bilinear products in the form of $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$. Thus, this is also a transformation from $\mathbb{R}^{2n}$ to $\mathbb{R}^k$. Each element of the output of this mapping equals the sum of $\boldsymbol{W}^{[i]} \odot (\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)$, where $\odot$ and $\otimes$ represent, respectively, the Hadamard and the outer products. Hence this mapping captures the direct interaction between different components (or "features") in two input vectors. Thanks to this expressiveness, NTNs are effective in tasks such as knowledge graph completion (Socher et al., 2013a), sentiment analysis (Socher et al., 2013b), and logical reasoning (Bowman et al., 2015).

Although mapping by a 3D tensor provides expressiveness, it has a large number ($O(n^2k)$) of parameters. Due to this, NTNs often suffer from overfitting and long training times.

3 Matrix Decomposition

3.1 Simple Matrix Decomposition (SMD)

To reduce the number of parameters of a slice matrix $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n \times n}$ in a tensor, simple matrix decomposition (SMD) is commonly used (Bai et al., 2009). SMD factorizes $\boldsymbol{W}^{[i]}$ into a product of two low rank matrices $\boldsymbol{S}^{[i]} \in \mathbb{R}^{n \times m}$ and $\boldsymbol{T}^{[i]} \in \mathbb{R}^{m \times n}$ $(m \ll n)$:

$$\boldsymbol{W}^{[i]} \simeq \boldsymbol{S}^{[i]} \boldsymbol{T}^{[i]}. \tag{1}$$

By plugging (1) into bilinear term $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$, we obtain the approximation $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{S}^{[i]} \boldsymbol{T}^{[i]} \boldsymbol{x}_2$. SMD reduces the number of parameters of $\boldsymbol{W}^{[i]}$ from n^2 to $2nm$. However, the dimension m for $\boldsymbol{S}$ and $\boldsymbol{T}$ is a hyperparameter and must be determined prior to training.

3.2 Simultaneous Diagonalization

This section introduces two techniques that can simultaneously diagonalize all slice matrices $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]} \in \mathbb{R}^{n \times n}$. As described in (Liu et al., 2017), we make use of the fact that if matrices $\boldsymbol{V}^{[1:k]}$ form a commuting family: i.e., $\boldsymbol{V}^{[i]} \boldsymbol{V}^{[j]} = \boldsymbol{V}^{[j]} \boldsymbol{V}^{[i]}, \forall i, j \in \{1, 2, \ldots, k\}$, they can be diagonalized by a shared orthogonal or unitary matrix. Both of the two techniques reduce the number of parameters of $\boldsymbol{W}^{[i]}$ to $O(n)$ from $O(n^2)$.

3.2.1 Orthogonal Diagonalization

Many NLP datasets contain symmetric patterns. For example, if binary relation *(Bob, is_relative_of, Alice)* holds in a knowledge graph, then *(Alice, is_relative_of, Bob)* should also hold in it. English phrases "dog and cat" and "cat and dog" have identical meaning. For symmetric structures, we can reasonably suppose that each slice matrix $\boldsymbol{W}^{[i]}$ of a 3D tensor is symmetric because $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$ must equal $\boldsymbol{x}_2^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_1$.

When $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n \times n}$ is symmetric, it can be diagonalized as:

$$\boldsymbol{W}^{[i]} = \boldsymbol{O}^{[i]} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{[i]\mathrm{T}}$$

where $\boldsymbol{O}^{[i]} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\boldsymbol{W}^{[i]\prime} \in \mathbb{R}^{n \times n}$ is a diagonal matrix. Note that an orthogonal matrix $\boldsymbol{O}^{[i]}$ may not be equal to $\boldsymbol{O}^{j}$ if $i \neq j$. However, if all of the slice matrices $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]} \in \mathbb{R}^{n \times n}$ are commuting, we can diagonalize every slice matrix with the same orthogonal matrix $\boldsymbol{O}$. By substituting $\boldsymbol{W}^{[i]}$ with $\boldsymbol{O} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{\mathrm{T}}$ into bilinear term $\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2$, we can rewrite it as follows:

$$\begin{aligned}
\boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{W}^{[i]} \boldsymbol{x}_2 &= \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{O} \boldsymbol{W}^{[i]\prime} \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_2 \\
&= \boldsymbol{y}_1^{\mathrm{T}} \boldsymbol{W}^{[i]\prime} \boldsymbol{y}_2 \\
&= \langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \boldsymbol{y}_2 \rangle
\end{aligned} \tag{2}$$

where $\boldsymbol{y}_1 = \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_1$, $\boldsymbol{y}_2 = \boldsymbol{O}^{\mathrm{T}} \boldsymbol{x}_2$, $\boldsymbol{w}^{[i]} = \mathrm{diag}(\boldsymbol{W}^{[i]\prime}) \in \mathbb{R}^n$ and $\langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \rangle$ denotes a "triple inner product" defined by $\langle \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \rangle = \sum_{l=1}^{n} a_l b_l c_l$. This reduces the number of parameters in a single slice matrix from n^2 to n.

3.2.2 Unitary Diagonalization

Since most of the structures in the NLP data are not symmetric, the symmetric matrix assumption is usually violated. To obtain more expressive diagonal matrix, we regard each slice matrix $\boldsymbol{W}^{[i]}$ as the real part of a complex matrix and consider its eigendecomposition.

For any real matrix $\boldsymbol{W}^{[i]}$, there exists a complex normal matrix $\boldsymbol{Z}^{[i]}$ whose real part is equal to it: $\boldsymbol{W}^{[i]} = \Re\left(\boldsymbol{Z}^{[i]}\right)$. $\Re(\cdot)$ represents an operation that takes the real part of a complex number, vector or matrix. Further, any complex normal matrix can be diagonalized by a unitary matrix. With these two properties, any real matrix $\boldsymbol{W}^{[i]}$ can be diagonalized as follows (Trouillon et al., 2016):

$$\boldsymbol{W}^{[i]} = \Re\left(\boldsymbol{Z}^{[i]}\right) = \Re\left(\boldsymbol{U}^{[i]}\boldsymbol{Z}^{[i]\prime}\boldsymbol{U}^{[i]*}\right).$$

Here, $\boldsymbol{U}^{[i]} \in \mathbb{C}^{n\times n}$ is a unitary matrix, $\boldsymbol{Z}^{[i]\prime} \in \mathbb{C}^{n\times n}$ is a diagonal matrix, and $\boldsymbol{U}^{[i]*}$ is the conjugate transpose of $\boldsymbol{U}^{[i]}$. To guarantee that every slice matrix can be diagonalized with the same unitary matrix $\boldsymbol{U}$ instead of $\boldsymbol{U}^{[i]}$, we assume all of the normal matrices $\boldsymbol{Z}^{[1]}, \ldots, \boldsymbol{Z}^{[i]}, \ldots, \boldsymbol{Z}^{[k]} \in \mathbb{C}^{n\times n}$ are commuting as in Section 3.2.1.

Substituting $\Re\left(\boldsymbol{U}\boldsymbol{Z}^{[i]\prime}\boldsymbol{U}^{*}\right)$ whose $\boldsymbol{U}$ is the same unitary matrix in all slice matrices, we can rewrite every bilinear term $\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2$ as follows:

$$\begin{aligned}\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2 &= \Re\left(\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle\right) \\ &= \langle \Re(\boldsymbol{y}_1), \Re(\boldsymbol{w}^{[i]}), \Re(\boldsymbol{y}_2)\rangle \\ &\quad + \langle \Re(\boldsymbol{y}_1), \Im(\boldsymbol{w}^{[i]}), \Im(\boldsymbol{y}_2)\rangle \\ &\quad + \langle \Im(\boldsymbol{y}_1), \Re(\boldsymbol{w}^{[i]}), \Im(\boldsymbol{y}_2)\rangle \\ &\quad - \langle \Im(\boldsymbol{y}_1), \Im(\boldsymbol{w}^{[i]}), \Re(\boldsymbol{y}_2)\rangle, \quad (3)\end{aligned}$$

where $\boldsymbol{y}_1 = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}_1$, $\overline{\boldsymbol{y}_2} = \boldsymbol{U}^{*}\boldsymbol{x}_2$, $\boldsymbol{w}^{[i]} = \mathrm{diag}(\boldsymbol{Z}^{[i]\prime}) \in \mathbb{C}^n$, and $\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle$ is the triple Hermitian inner product of $\boldsymbol{y}_1$, $\boldsymbol{w}^{[i]}$ and $\boldsymbol{y}_2$ defined by $\langle \boldsymbol{a}, \boldsymbol{b}, \overline{\boldsymbol{c}}\rangle = \sum_{l=1}^{n} a_l b_l \overline{c_l}$. This technique reduces the number of parameters of the matrices from n^2 to $2n$. As shown in the right-hand side of Eq. (3), $\Re\left(\langle \boldsymbol{y}_1, \boldsymbol{w}^{[i]}, \overline{\boldsymbol{y}_2}\rangle\right)$ can be replaced with three additions and a subtraction of the triple inner product of real vectors.

4 Neural Network Models

This section introduces the baseline and our proposed models. After describing them, we explain how to extend them for handling compositional structures like binary trees.

Model	# of Parameters
NN	$(2n+1)k$
NTN	$(n^2+2n+1)k$
NTN-SMD	$(2mn+2n+1)k$
NTN-Diag	$(3n+1)k$
NTN-Comp	$(6n+1)k$

Table 1: Comparison of the number of parameters among the models

4.1 Baseline Models

Neural Network (NN)

First, we describe a standard single layer neural network (NN) model for two vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$. The model uses linear mapping $\boldsymbol{V} \in \mathbb{R}^{k\times 2n}$ to combine two input vectors:

$$f(\boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b})$$

where $\boldsymbol{b} \in \mathbb{R}^k$ is a bias term and f is a non-linear activation function. The NN model has only $(2n+1)k$ parameters, and does not consider the direct interactions between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

Neural Tensor Network (NTN)

Socher et al. (2013a) proposed a neural tensor network (NTN) model that uses a 3D tensor $\boldsymbol{W}^{[1:k]} \in \mathbb{R}^{n\times n\times k}$ to combine two input vectors:

$$f(\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[1:k]}\boldsymbol{x}_2 + \boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b}).$$

Unlike the standard NN model, NTN can directly relate two input vectors using a tensor. However, it has too many parameters; $(n^2+2n+1)k$.

NTN-SMD

Although the NTN model has tremendous expressive power, it is extremely time-consuming to compute, since a naive 3D tensor product incur $O(n^2k)$ computation time. To overcome this weakness, Zhao et al. (2015) and Liu et al. (2015) independently introduced simple matrix decomposition (SMD) to the NTN model by replacing each slice matrix $\boldsymbol{W}^{[i]}$ with its factorized approximation given by Eq. (1):

$$f(\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{S}^{[1:k]}\boldsymbol{T}^{[1:k]}\boldsymbol{x}_2 + \boldsymbol{V}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} + \boldsymbol{b})$$

where $\boldsymbol{S}^{[1:k]} \in \mathbb{R}^{n\times m\times k}, \boldsymbol{T}^{[1:k]} \in \mathbb{R}^{m\times n\times k}$. When $m \ll n$, the NTN-SMD model drastically reduces the number of parameters compared to the original NTN model; i.e., from $(n^2+2n+1)k$ to $(2mn+2n+1)k$.

4.2 NTNs with Diagonal Slice Matrices

In this paper, we introduce two new NTN models: NTN-Diag and NTN-Comp, both of which reduce the number of parameters in a 3D tensor more than NTN-SMD with little loss in the model's generalization performance. Table 1 summarizes the number of parameters in each model.

NTN-Diag

We replace all slice matrices $\boldsymbol{W}^{[i]}$ of $\boldsymbol{W}^{[1:k]}$ with the triple inner product formulation of Eq. (2) by assuming that they are symmetric and commuting. As a result, we derive the following new NTN formulation:

$$f(\begin{bmatrix} \langle \boldsymbol{x}_1, \boldsymbol{w}^{[1]}, \boldsymbol{x}_2 \rangle \\ \vdots \\ \langle \boldsymbol{x}_1, \boldsymbol{w}^{[k]}, \boldsymbol{x}_2 \rangle \end{bmatrix} + \boldsymbol{V} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} + \boldsymbol{b})$$

where $\boldsymbol{w}^{[i]} \in \mathbb{R}^n$, $\forall i \in \{1, 2, \ldots, k\}$. Thus, under the symmetric and commuting matrix constraints, we regard mapping by a 3D tensor as an array of k triple inner products. The total number of parameters is just $(3n + 1)k$.

NTN-Comp

By assuming that $\boldsymbol{W}^{[1]}, \ldots, \boldsymbol{W}^{[i]}, \ldots, \boldsymbol{W}^{[k]}$ are real parts of normal matrices forming a commuting family, we can replace each slice matrix of a tensor term in NTN with the triple Hermitian inner product shown in Eq. (3):

$$f(\begin{bmatrix} \Re\left(\langle \boldsymbol{x}_1, \boldsymbol{w}^{[1]}, \overline{\boldsymbol{x}_2} \rangle\right) \\ \vdots \\ \Re\left(\langle \boldsymbol{x}_1, \boldsymbol{w}^{[k]}, \overline{\boldsymbol{x}_2} \rangle\right) \end{bmatrix} + \Re\left(\boldsymbol{V} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}\right) + \boldsymbol{b})$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{C}^n$, $\boldsymbol{V} \in \mathbb{C}^{n \times n}$ and $\boldsymbol{w}^{[i]} \in \mathbb{C}^n$, $\forall i \in \{1, 2, \ldots, k\}$. Similar to NTN-Diag, we regard mapping by a 3D tensor as an array of k triple Hermitian inner products. The total number of parameters is just $(6n + 1)k$. As is clear of its form, NTN-Diag is a special case of NTN-Comp whose vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{w}^{[i]}$ are constrained to be real.

4.3 Recursive Neural Tensor Networks

We extend the above NTN models to handle compositional structures. As a representative of compositional structures, we consider a binary tree where each NTN layer computes a vector representation for a node by combining two vectors from its child nodes in the lower layer. Except for NTN-Comp, the models implement mappings $\mathbb{R}^n \rightarrow \mathbb{R}^k$ so that each of their layers can receive its lower layer's output directly, if k equals to n. Thus, the models do not have to be modified for them. However, NTN-Comp cannot receive its lower layer's output as it is because NTN-Comp is a mapping from $\mathbb{C}^n$ to $\mathbb{R}^k$. To solve this problem, we set k to $2n$ and treat the output $\boldsymbol{y}' \in \mathbb{R}^{2n}$ as the concatenation of vectors representing the real and imaginary parts of $\boldsymbol{y} \in \mathbb{C}^n$:

$$\Re(\boldsymbol{y}) = (y'_1, \cdots, y'_n), \ \Im(\boldsymbol{y}) = (y'_{n+1}, \cdots, y'_{2n}).$$

Note that this approach is valid since Eq. (3) can actually be defined in real vector space by transforming the complex vectors in $\mathbb{C}^n$ into real vectors in $\mathbb{R}^{2n}$.

5 Related Work

Knowledge Graph Completion

In KGC, researchers usually design scoring function Φ for the given triplet (s, r, o) to judge whether it is a fact or not. Here (s, r, o) denotes that entity s is linked to entity o by relation r. RESCAL (Nickel et al., 2011) uses $\boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{W}_r \boldsymbol{e}_o$ as Φ, where $\boldsymbol{e}_s, \boldsymbol{e}_o$ are entity embedding vectors and $\boldsymbol{W}_r$ is an embedding matrix of relation r. This bilinear operation is effective for the task, but its computational cost is high and it suffers from overfitting. To overcome these problems, DistMult (Yang et al., 2015) adopts the triple inner product $\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \boldsymbol{e}_o \rangle$ as Φ, where $\boldsymbol{w}_r$ is an embedding vector of relation r. This solves those problems, but it degrades the model's ability to capture directionality of relations, because the scoring function of DistMult is symmetric with respect to s and o; i.e., $\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \boldsymbol{e}_o \rangle = \langle \boldsymbol{e}_o, \boldsymbol{w}_r, \boldsymbol{e}_s \rangle$. To reconcile the complexity and expressiveness of a model, ComplEx (Trouillon et al., 2016) uses complex vectors for entity and relation embeddings. As scoring function Φ, they adopted the triple Hermitian inner product $\Re\left(\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \overline{\boldsymbol{e}_o} \rangle\right)$, where $\overline{\boldsymbol{e}_o}$ denotes the complex conjugate of $\boldsymbol{e}_o$. Since $\Re\left(\langle \boldsymbol{e}_s, \boldsymbol{w}_r, \overline{\boldsymbol{e}_o} \rangle\right) \neq \Re\left(\langle \boldsymbol{e}_o, \boldsymbol{w}_r, \overline{\boldsymbol{e}_s} \rangle\right)$, ComplEx solves the expressiveness problem of DistMult without full matrices as relation embeddings. We can regard DistMult as a special case of RESCAL with a symmetric matrix constraint on $\boldsymbol{W}_r$. ComplEx is also a RESCAL variant with $\boldsymbol{W}_r$ as the real part of a normal matrix. Our research is based on these works, but to the best of our knowledge, no previous work applied this approach to reduce the number of parameters in a tensor.

NN Architectures

To give additional expressiveness power to standard (R)NNs, many architectures have been proposed, such as LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), and CNN (LeCun et al., 1998). NTN (Socher et al., 2013a) and RNTN (Socher et al., 2013b) are other such architectures. However, (R)NTNs differ in that they only add 3D tensor mapping to standard neural networks. Thus, they can also be regarded as a powerful basic component of NNs because 3D tensor mapping can be applied to more complicated architectures such as those examples.

Parameter Reduction in NN

Several researchers reduced the number of parameters of NNs by using specific parameter sharing mechanisms. Cheng et al. (2015) used circulant matrix mapping instead of conventional linear mapping and improved the time complexity of the matrix-vector product by using Fast Fourier Transformation (FFT). Circulant matrix

$$C(\boldsymbol{w}) = \begin{bmatrix} w_1 & w_n & \dots & w_3 & w_2 \\ w_2 & w_1 & \dots & w_4 & w_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n-1} & w_{n-2} & \dots & w_1 & w_n \\ w_n & w_{n-1} & \dots & w_2 & w_1 \end{bmatrix}$$

for $\boldsymbol{w}^{\mathrm{T}} = (w_1, \dots, w_n)$ can be factorized into $\mathfrak{F}^{-1}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w})\,\mathfrak{F}$ with the Fourier matrix $\mathfrak{F}$. By assuming each slice matrix $\boldsymbol{W}^{[i]}$ of $\boldsymbol{W}^{[1:k]}$ is circulant, we get the same scoring function as that in Eq. (3); $\boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{W}^{[i]}\boldsymbol{x}_2 = \boldsymbol{x}_1^{\mathrm{T}}\,\mathfrak{F}^{-1}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w}^{[i]})\,\mathfrak{F}\,\boldsymbol{x}_2 = \Re(\langle \boldsymbol{x}_1', \boldsymbol{w}^{[i]\prime}, \overline{\boldsymbol{x}_2'}\rangle)$ where $\boldsymbol{x}_1' = \overline{\mathfrak{F}\,\boldsymbol{x}_1}$, $\boldsymbol{x}_2' = \overline{\mathfrak{F}\,\boldsymbol{x}_2}$, and $\boldsymbol{w}^{[i]\prime} = \frac{1}{n}\,\mathrm{diag}(\mathfrak{F}\,\boldsymbol{w}^{[i]})$ are complex vectors in $\mathbb{C}^n$. In this sense, NTN-Comp is equivalent to NTN where slice matrices of the 3D tensor are restricted to be circulant. Hayashi and Shimbo (2017) established a more detailed proof of the equivalence. Lu et al. (2016) employed a Toeplitz-like structured matrix, reducing parameters of LSTM. Chen et al. (2015) used a feature hashing technique to reduce parameters in RNN. Although these techniques can also be extended to reduce the number of tensor-related parameters in NTN, the former needs FFT operations; i.e., $O(n \log n)$ computation time, and the latter's contribution is only a reduction in memory consumption.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test
FB15k	14,951	1,345	483,142	50,000	59,701
WN18	40,943	18	141,442	5,000	5,000

Table 2: Dataset statistics.

6 Experiment

6.1 Knowledge Graph Completion

To evaluate their performance for link prediction on knowledge graphs, we compared our proposed methods (NTN-Diag and NTN-Comp) to baseline methods (NTN (Socher et al., 2013a) and NTN-SMD).

Task

Let $\mathcal{E}$ and $\mathcal{R}$ denote entities and relations, respectively. A *relational triplet*, or simply a *triplet*, (s, r, o) is a triple with $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. It represents a proposition that relation r holds between subject entity s and object entity o. A triplet is called a fact if the proposition it denote is true. A *knowledge graph* is a collection of knowledge triplets, with the understanding that all its member triplets are facts. It is called a graph because each triplet can be regarded as an edge in a directed graph; the vertices in this graph represent entities in $\mathcal{E}$, and each edge is labeled by a relation in $\mathcal{R}$. Let $\mathcal{G}$ be a knowledge graph, viewed as a collection of facts. *Knowledge graph completion* (KGC) is the task of predicting whether unknown triplet $(s', r', o') \notin \mathcal{G}$ such that $s', o' \in \mathcal{E}, r' \in \mathcal{R}$ is a fact or not.

Models and Loss Function

The standard approach to KGC is to design a score function $\Phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ that assigns a large value when a triplet seems to be a fact. Socher et al. (2013a) defined it as follows.

$$\boldsymbol{u}_r^{\mathrm{T}} f\left(\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{W}_r^{[1:k]}\boldsymbol{e}_o + \boldsymbol{V}_r\begin{bmatrix}\boldsymbol{e}_s\\ \boldsymbol{e}_o\end{bmatrix} + \boldsymbol{b}_r\right)$$

Here, $\boldsymbol{e}_s, \boldsymbol{e}_o \in \mathbb{R}^n$ are entity embeddings and $\boldsymbol{W}_r, \boldsymbol{V}_r, \boldsymbol{b}_r, \boldsymbol{u}_r$ are parameters for each relation r. $\boldsymbol{u}_r$ is a k-dimensional vector to map f's output $\mathbb{R}^k$ to $\mathbb{R}$ which indicates a score. f is the hyperbolic tangent. To compare the performances of the baselines and proposed models, we change the mapping before an activation. For NTN-SMD, we change term $\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{W}_r^{[1:k]}\boldsymbol{e}_o$ to $\boldsymbol{e}_s^{\mathrm{T}}\boldsymbol{S}_r^{[1:k]}\boldsymbol{T}_r^{[1:k]}\boldsymbol{e}_o$. To apply NTN-Diag and NTN-Comp in this model,

	WN18					FB15K				
	MRR		Hits@			MRR		Hits@		
model	Filter	Raw	1	3	10	Filter	Raw	1	3	10
NN	0.111	0.106	7.0	11.7	18.3	0.259	0.165	17.9	28.1	41.7
NTN ($k = 1$)	0.740	0.512	67.6	78.4	85.2	0.347	0.188	24.1	39.3	55.2
NTN ($k = 4$)	0.754	0.530	69.3	79.5	86.3	0.380	0.198	27.1	43.0	59.2
NTN-SMD ($m = 1$)	0.243	0.216	15.9	26.1	40.9	0.278	0.172	19.3	30.1	44.7
NTN-SMD ($m = 2$)	0.224	0.199	15.1	23.8	37.2	0.298	0.177	20.7	32.7	47.8
NTN-SMD ($m = 3$)	0.299	0.255	20.4	32.4	49.2	0.312	0.183	21.7	34.5	49.9
NTN-SMD ($m = 10$)	0.533	0.413	42.2	59.4	74.5	0.333	0.188	22.8	37.5	53.8
NTN-SMD ($m = 25$)	0.618	0.463	52.1	67.8	80.0	0.341	0.187	23.2	38.6	55.5
NTN-Diag	0.824	0.590	74.8	89.6	92.7	0.443	0.238	31.5	51.2	68.5
NTN-Comp	**0.857**	**0.610**	**80.1**	**90.9**	**93.1**	**0.490**	**0.246**	**36.3**	**56.7**	**71.9**
DistMult*	0.822	0.532	72.8	91.4	93.6	0.654	0.242	54.6	73.3	82.4
ComplEx*	0.941	0.587	93.6	94.5	94.7	0.692	0.242	59.9	75.9	84.0

Table 3: Mean Reciprocal Rank (MRR) and Hits@n for the models tested on WN18 and FB15k. MRR is reported in the raw and filtered settings. Hits@n metrics are percentages of test examples that lie in the top n ranked results. We report Hits@n in the filtered setting. *Results are those in (Trouillon et al., 2016)

we assume all slice matrices of tensors among relations form a commuting family. The loss function used to train the models is shown below:

$$\sum_{i=1}^{N}\sum_{c=1}^{C} \max\left(0, 1 - \Phi\left(T^{(i)}\right) + \Phi\left(T_c^{(i)}\right)\right) + \lambda\|\mathbf{\Omega}\|_2^2,$$

where $\lambda\|\mathbf{\Omega}\|_2^2$ is an L2 regularization term, $T^{(i)}$ denotes the i-th example of training data of size N, and $T_c^{(i)}$ is one of C randomly sampled negative examples for the i-th training example. We generated negative samples of a triplet (s, r, o) by corrupting its subject or object entity.

Experimental Setup

We used the Wordnet (WN18) and Freebase (FB15k) datasets to verify the benefits of our proposed methods. The dataset statistics are given in Table 2. We selected hyper-parameters based on Socher et al. (2013a) and Yang et al. (2015): For all of the models, the size of mini-batches was set to 1000, the dimensionality of the entity vector to $d = 100$, and the regularization parameter to 0.0001; the tensor slice size was set to $k = 4$ for all models, except NTN for which we also tested with $k = 1$ to see the influence of the slice size on the performance. We performed 300 epochs of training for Wordnet and 100 on Freebase using Adagrad (Duchi et al., 2011) with the initial learning rate set to 0.1.

For evaluation, we removed the subject or object entity of each test example and then replaced it with all the entities in $\mathcal{E}$. We computed the scores of these corrupted triplets and ranked them in descending order of scores. We here report the results collected in filtered and raw settings. In the filtered setting, given test example (s, r, o), we remove from the ranking all the other positive triplets that appear in either training, validation, or test dataset, whereas the raw metrics do not remove these triplets.

Result

Experimental results are shown in Table 3. We observe the following:

- The performance of NN and NTNs differs considerably; Apparently, NN is inadequate for this task.
- By comparing the results of NTNs with different slice sizes, we see that $k = 4$ performs better than $k = 1$.
- NTN-SMDs perform better than NN, but are all inferior to NTNs, although their results improved as m (the rank of decomposed matrices) is increased.
- NTN-Diag achieved better results than NTN, although it has far fewer parameters than NTN and the datasets contain many unsymmetrical triplets. This demonstrates that NTN-Diag solves the overfitting problem of NTN without sacrificing the expressiveness power. NTN-Diag also has fewer parameters than the smallest ($m = 1$) NTN-SMD. Thus,

Conjunctive normal form	$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} A_{ij}$
Disjunctive normal form	$\bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} A_{ij}$

Table 4: Conjunctive and disjunctive normal forms in propositional logic. A_{ij} is a *literal*, which is a propositional variable or its negation. For example, p_1 and $\neg p_2$ are literal, but not $\neg\neg p_3$.

Name	Symbol	Set-theoretic definition
Entailment	$A \sqsubset B$	$A \subset B$
Reverse entailment	$A \sqsupset B$	$A \supset B$
Equivalence	$A \equiv B$	$A = B$
Alternation	$A \mid B$	$A \cap B = \emptyset \wedge A \cup B \neq \mathcal{D}$
Negation	$A \,^\wedge\, B$	$A \cap B = \emptyset \wedge A \cup B = \mathcal{D}$
Cover	$A \smile B$	$A \cap B \neq \emptyset \wedge A \cup B = \mathcal{D}$
Independence	$A \# B$	else

Table 5: Natural logic relations over formula pairs. A and B denote a formula in propositional logic.

$\textit{not } p_3$	$^\wedge$	p_3
p_3	$\sqsubset$	$(p_3 \textit{ or } p_2)$
$(p_1 \textit{ or}(p_2 \textit{ or } p_4)))$	$\sqsupset$	$(p_2 \textit{ and not } p_4)$

Table 6: Short examples of type of formulas and their relations in datasets.

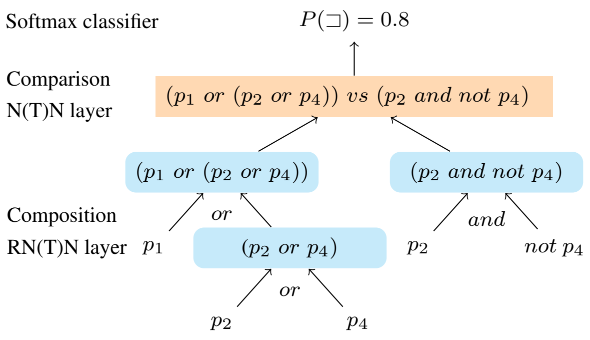

Softmax classifier
$P(\sqsupset) = 0.8$
Comparison N(T)N layer
$(p_1 \textit{ or } (p_2 \textit{ or } p_4)) \textit{ vs } (p_2 \textit{ and not } p_4)$
$(p_1 \textit{ or } (p_2 \textit{ or } p_4))$
$(p_2 \textit{ and not } p_4)$
Composition RN(T)N layer
p_1
or
$(p_2 \textit{ or } p_4)$
p_2
and
$\textit{not } p_4$
or
p_2
p_4

Figure 1: Comparison and composition layers. *not* p_4 is treated as an embedding.

we conclude that NTN-Diag is a better alternative of NTN than NTN-SMD is, in terms of both accuracy and computational cost.

- NTN-Comp outperformed NTN-Diag, showing that its flexible constraint on matrices yielded additional expressiveness. However, NTN-Diag and NTN-Comp do not exceed DistMult and ComplEx, respectively, in almost all measures.

Although not shown in the table, in this experiment, NTN-Diag and NTN-Comp was, respectively, 3 and 1.7 times as fast as NTN to train.

6.2 Logical Reasoning

To validate the performance of our proposed models in a recursive neural network setting, we experimentally tested them by having them solve a semantic compositionality problem in logic.

Task

This task definition basically follows Bowman et al. (2015): Given a pair of artificially generated propositional logic formulas, classify the relation between the formulas into one of the seven *basic semantic relations* of natural logic (MacCartney and Manning, 2009). Table 5 shows these seven relation types. The formulas consist of propositional variables, negation, and conjunction and disjunction connectives. Although Bowman et al. (2015) generated formulas with no constraint on its form, we restricted them to disjunctive normal form (DNF) or conjunctive normal form (CNF) (Table 4). Recall that any propositional formula can be transformed into these forms.

Models and Loss Function

Following Bowman et al. (2015), we constructed a model that infers the relations between formula pairs, as described in Table 6.

The model consists of two layers: composition and comparison layers (Figure 1). The composition layer outputs the embeddings of both left and right formulas by recursive neural networks. Subsequently, the comparison layer compares the two embeddings using a single layer neural network, and then a softmax classifier receives its output. In the composition layer, we set different parameters for *and* and *or* operations. As a loss function, we used cross entropy with L2 regularization and apply the NTNs in Section 4 to the comparison layer and uses RNTNs for as the composition layer.

Experimental Setup

In this experiment, an example is a pair of propositional formulas, and its class label is the seven relation types between the pair. We generated examples following the protocol described in Bowman et al. (2015), with the exception that the formulas are restricted to CNF or DNF, as mentioned above. We obtained 62,589 training examples, 13,413 validation examples, and 55,150 test examples. Each formula in the training and validation examples contains up to four logical operators, whereas those in the test examples have

Model	1	2	3	4	5	6	7	8	9	10	11	12	Avg.
Majority class	56.0	53.0	53.4	53.2	55.9	56.5	56.5	57.8	56.5	57.7	56.8	59.9	56.1
RNN	98.0	97.5	95.5	93.3	89.9	86.1	82.8	79.9	74.8	73.2	71.8	71.7	84.5
RNTN	**99.9**	**99.5**	98.2	95.7	92.7	88.5	84.7	81.2	78.1	77.5	74.4	74.4	87.0
RNTN-SMD ($m = 1$)	93.7	92.5	90.9	89.1	86.9	84.1	81.7	79.8	76.1	75.7	75.3	75.1	83.4
RNTN-SMD ($m = 2$)	93.0	93.4	91.7	90.3	88.2	85.5	82.7	81.4	77.6	77.0	75.4	**75.8**	84.3
RNTN-SMD ($m = 4$)	90.2	90.3	89.4	87.6	86.0	83.6	81.2	79.6	76.5	75.2	74.6	75.7	82.4
RNTN-SMD ($m = 8$)	86.8	84.9	83.5	82.5	81.1	79.1	76.6	75.6	72.4	71.3	70.9	71.2	77.9
RNTN-SMD ($m = 16$)	86.6	83.9	82.4	81.4	80.2	78.6	76.5	75.5	73.1	72.7	72.2	73.3	78.0
RNTN-Diag	**99.9**	98.9	**98.5**	**97.4**	**94.9**	**91.5**	**87.6**	**85.0**	**80.3**	**78.5**	**77.1**	75.2	**88.7**
RNTN-Comp	99.3	98.1	98.0	96.9	94.3	90.6	86.1	83.5	79.2	76.6	74.5	74.6	87.6

Table 7: Result of logical inference for Tests 1–12. Example in Test n has n logical operators in either or both left and right formulas. Each score is the average accuracy of five trials of the λ that achieved best performance on validation set. "Majority class" denotes the ratio of the majority class (relation "#", i.e., Independence; see Table 5).

Model	Accuracy	(Std. Dev.)
RNN	95.0	(0.8)
RNTN	97.2	(0.4)
RNTN-SMD ($m = 1$)	90.1	(3.4)
RNTN-SMD ($m = 2$)	91.4	(4.6)
RNTN-SMD ($m = 4$)	88.6	(7.1)
RNTN-SMD ($m = 8$)	82.7	(10.2)
RNTN-SMD ($m = 16$)	81.8	(11.7)
RNTN-Diag	98.1	(0.1)
RNTN-Comp	97.5	(0.1)

Table 8: Average accuracy and standard deviation on the validation dataset. The reported values are average over the best-performing model λ in each method.

up to 12 logical operators. Every formula consists of up to four variables taken from six propositional variables that are shared among all the examples. Hyperparameters and optimization are based on Bowman et al. (2015): Embedding size $d = 25$ (for RNN, $d = 45$) and the output size of comparison layer is $k = 75$, and we used AdaDelta (Zeiler, 2012) for an optimizer. We searched for the best coefficient λ of L2 regularization in $\lambda \in \{0.0001, 0.0003, 0.0005, 0.0007, 0.0009, 0.001\}$, whereas Bowman et al. (2015) set λ to 0.001 for RNN and 0.0003 for RNTN.

Result

The results are shown in Table 7. From the table, we observe the following:

- As with KGC, the large difference in performance between RNN and RNTN suggests that this logical reasoning task requires feature interactions to be captured[1].

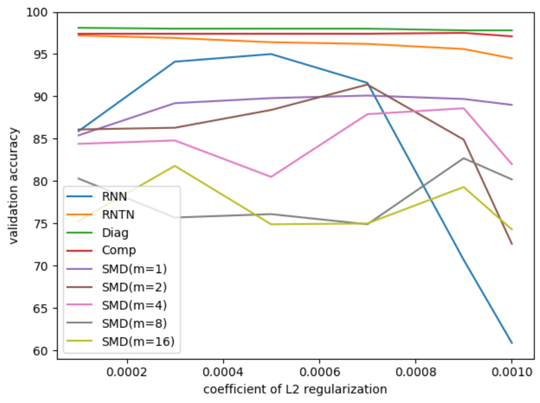
- RNTN-Diag achieved the best accuracy except for Tests 2 and 12 and outperformed RNTN except for Test 2. This is not surprising because both *and* and *or* are symmetric: p_1 *and* p_2 equals p_2 *and* p_1. This matches the tensor term in RNTN-Diag which is symmetric with respect to $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

- RNTN-Comp was the second best except for Tests 1–3 and 10–12. For all tests, its accuracy was comparable with or superior to that of RNTN.

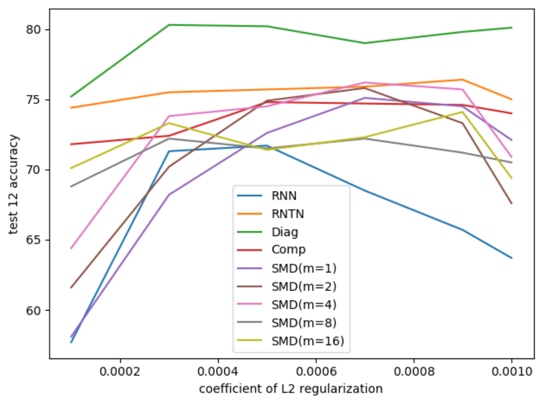
- RNTN-SMD ($m = 1$) was inferior to RNTN for most test sets, although some good results were observed with $m = 1, 2, 3$ on Tests 11 and 12. Indeed, except for Tests 9–12, RNTN-SMD ($m = 1$) was inferior even to RNN despite the larger number of parameters in RNTN-SMD. RNTN-SMD ($m = 2$) obtained better results than $m = 1$, but it is still worse than RNTN except for Tests 10-12. Further increase in m ($m = 4, 8, 16$) worsened the accuracy despite an increase of the number of parameters.

We also evaluated the stability of the model over different trials and hyperparameters. Table 8 shows the best average accuracy for each compared model (among all the tested λ) on the validation set. The parenthesized figures (on the rightmost column) show the standard deviation over five independent trials used for computing the average, i.e., all five trials used the same λ value that achieved the best average accuracy. We see that RNTN-SMDs have larger standard deviations than

[1]Bowman (2016) also evaluated TreeLSTM, but its advantage over RNN was unclear in their experiment. For that reason, we did not test TreeLSTM in this paper.



(a) Validation set.



(b) Test 12.

Figure 2: Sensitivity of accuracy to λ.

RNTN, RNTN-Diag and RNTN-Comp. This indicates that RNTN-SMD is a less reliable model.

RNTN-SMDs are also unstable, not only within the same λ, but also between different λs. Figure 2 describes how accuracies are impacted by λs. The top graph shows validation accuracies between different λ values. RNTN, RNTN-Diag and RNTN-Comp are stable, whereas RNN and RNTN-SMDs have steep drops. The bottom one describes the accuracies for Test 12. This also shows that RNTN-SMDs are unstable and that RNTN-Diag achieves distinctive performances.

Finally, Figure 3 shows that training times increase quadratically with dimension for RNTN that has $O(n^2k)$ parameters, but not for our methods, which have only $O(nk)$ parameters.

7 Conclusion

We proposed two new parameter reduction methods for tensors in NTNs. The first method constrains the slice matrices to be symmetric, and the second assumes them to be normal matrices. In both methods, the number of a 3D tensor parameters is reduced from $O(n^2k)$ to $O(nk)$ after the constrained matrices are eigendecomposed. By removing the tensor's surplus parameters, our methods learn better and faster as was shown in experiments.[2] Future work will test the versatility of our proposals, RNTN-Diag and RNTN-Comp, in other tasks that deal with data sets exhibiting carious structures.

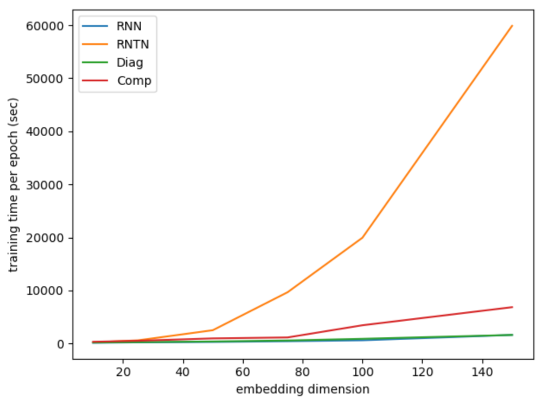


Figure 3: Training times of the models.

References

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Corinna Cortes, and Mehryar Mohri. 2009. Polynomial semantic indexing. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pages 64–72.

Samuel R. Bowman. 2016. *Modeling natural language semantics in learned representations*. Ph.D. thesis, Stanford University.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015. Recursive neural networks can learn logical semantics. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality* .

Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 2285–2294.

Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2857–2865.

[2]Code of the two experiments will be available at https://github.com/tkhrshhr

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation pages 1724–1734.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* pages 554–559.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. pages 2168–2178.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 1284–1290.

Zhiyun Lu, Vikas Sindhwani, and Tara N Sainath. 2016. Learning compact recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 5960–5964.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*. Association for Computational Linguistics, pages 140–156.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning*. pages 809–816.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*. pages 926–934.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 1631–1642.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*. pages 2071–2080.

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

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Phrase type sensitive tensor indexing model for semantic composition. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. pages 2195–2202.