Learning Distributed Representations of Symbolic Structure Using Binding and Unbinding Operations

[α]Shuai Tang [β,γ]Paul Smolensky [α]Virginia R. de Sa
[α] Department of Cognitive Science, UC San Diego
[β] Microsoft Research AI, Redmond
[γ] Department of Cognitive Science, Johns Hopkins University
shuaitang93@ucsd.edu, psmo@microsoft.com, desa@ucsd.edu

Abstract

Widely used recurrent units, including Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU), perform well on natural language tasks, but their ability to learn structured representations is still questionable. Exploiting Tensor Product Representations (TPRs) — distributed representations of symbolic structure in which vector-embedded symbols are bound to vector-embedded structural positions — we propose the TPRU, a recurrent unit that, at each time step, explicitly executes structural-role binding and unbinding operations to incorporate structural information into learning. Experiments are conducted on both the Logical Entailment task and the Multi-genre Natural Language Inference (MNLI) task, and our TPR-derived recurrent unit provides strong performance with significantly fewer parameters than LSTM and GRU baselines. Furthermore, our learnt TPRU trained on MNLI demonstrates solid generalisation ability on downstream tasks.

1 Introduction

Recent advances in deep learning benefit largely from neural networks' ability to learn distributed representations of inputs from various domains [8]; even samples from different modalities can be easily compared in a common representation space [16, 42]. In contrast to localist (1-hot) representations that are not able to directly represent the possible componential structure within data [20], distributed representations are potentially capable of inducing implicit structure in the data, or explicit structure that is not presented along with the data. When it comes to statistical inference, distributed representations show considerable power, attesting to strong ability to encode world knowledge and to efficiently use representation space. However, the interpretability of learnt distributed representations is limited; it is typically quite unclear what specifically has been encoded in the representations.

Symbolic computing can take advantage of the presented structure of data and denote each substructure as a symbol; throughout computation, the representations derived by symbolic computing maintain the structure of data explicitly, and each substructure can be retrieved by simple straightforward computation [9]. In terms of inducing implicit structure from data, symbolic computing aims to decompose each sample to an ensemble of unique symbols that carry potential substructure of the data. With enough symbols, the underlying structure of data can be encoded thoroughly. The explicit usage of symbols in symbolic computing systems improves the capability of induced representations, but also, it brings in issues including inefficient memory usage and computational expense.

Tensor product representation (TPR) [38] is an instantiation of general neural-symbolic computing in which symbol structures are given a *filler-role decomposition*: the structure is captured by a set of N **roles** r_i (e.g., left-child-of-root), each of which is bound to a **filler** f_i (e.g., a symbol). A TPR embedding of a symbol structure $\boldsymbol{S}$ derives from vector embeddings of the roles $\{\boldsymbol{r}_i\}$ and their fillers

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

$\{\boldsymbol{f}_i\}$ via the outer or tensor product: $\boldsymbol{S} = \sum_{i=1}^{N} \boldsymbol{r}_i \otimes \boldsymbol{f}_i = \sum_{i=1}^{N} \boldsymbol{r}_i \boldsymbol{f}_i^\top = \boldsymbol{R}\boldsymbol{F}^\top$, where $\boldsymbol{R}$ and $\boldsymbol{F}$ respectively denote matrices having the role or filler vectors as columns. Each $\boldsymbol{r}_i \otimes \boldsymbol{f}_i$ is the embedding of a role-filler binding; $\otimes$ is the **binding** operation. The **unbinding** operation returns the filler of a particular role in $\boldsymbol{S}$; it is performed by the inner product: $\boldsymbol{f}_i = \boldsymbol{u}_i^\top \boldsymbol{S}$, where $\boldsymbol{u}_i$ is the dual of $\boldsymbol{r}_i$, satisfying $\boldsymbol{u}_i^\top \boldsymbol{r}_j = \delta_{ij}$. Letting $\boldsymbol{U}$ be the matrix with columns $\{\boldsymbol{u}_i\}$, we have $\boldsymbol{U}^\top \boldsymbol{R} = \mathbb{I}$.[1]

Let $\{\boldsymbol{r}_i\} \subset \mathbb{R}^{d\times 1}$. For present purposes it turns out that it suffices to consider filler vectors $\{\boldsymbol{f}_i\} \subset \mathbb{R}^{1\times 1}$, in which case $\boldsymbol{S} \in \mathbb{R}^{d\times 1}$; we henceforth denote $\boldsymbol{S}$ as $\boldsymbol{b}$, a **binding complex**. Let $\boldsymbol{f} \in \mathbb{R}^{N\times 1} = \boldsymbol{F}^\top$ be the column vector comprised of the N $\{\boldsymbol{f}_i\} \subset \mathbb{R}^{1\times 1}$. Now the binding and unbinding operations, simultaneously over all roles, become

$$\boldsymbol{b} = \boldsymbol{R}\boldsymbol{f} \qquad \text{and} \qquad \boldsymbol{f} = \boldsymbol{U}^\top \boldsymbol{b}. \tag{1}$$

With binding and unbinding operations and sufficient role vectors, the binding complex $\boldsymbol{b}$ is able to represent data from a wide range of domains, with the structure of the data preserved.

We aim to incorporate symbolic computing into learning distributed representations of data when explicit structure is not presented to neural networks. Specifically, we propose a recurrent unit that executes binding and unbinding operations according to Eq. 1. The proposed recurrent unit leverages both the advantages of distributed representations and the ability to explore and maintain learnt structure from symbolic computing. Our contribution is threefold:

• We propose a recurrent unit, named TPRU, which integrates symbolic computing with learning distributed representations, and has significantly fewer parameters than widely used Long-short Term Memory (LSTM) [21] and Gated Recurrent Unit (GRU) [12].

• We present experimental results with the TPRU on both the Logical Entailment task [18] and the Multi-genre Natural Language Inference (MNLI) dataset [44], both of which arguably require high-quality structured representations for making good predictions. The proposed unit provides strong performance on both tasks.

• The TPRU trained on MNLI with plain (attentionless) architecture demonstrates solid generalisation ability on downstream natural language tasks.

2 Related Work

Recent efforts on learning structured distributed representations can be roughly categorised into two types: enforcing a strong global geometrical constraint on the representation space, such as hyperbolic embedding [27, 46], and introducing inductive biases into the architecture of networks, including the Relational Memory Core [36] and neural-symbolic computing methods [9]. The latter category divides into models that insert neural networks into discrete structures [7, 34, 39] and those that insert discrete structures into neural network representations [19]. Our work falls in this last category: learning structured representations by incorporating the inductive biases inherent in TPRs.

Some prior work has incorporated TPRs into RNN representations. Question-answering on SQuAD [35] was addressed [28] with a GRU in which the hidden state was a single TPR binding; the present work deploys complexes containing multiple bindings. TPR-style unbinding was applied in caption generation [23], but the representations were deep-learnt and not explicitly designed to be TPRs as in the present work. A contracted version of TPRs, Holographic Reduced Representations, was utilised to decompose input space and output space with filler-role decomposition [6]. However, our work differs from prior work in that TPR filler-role binding and unbinding operations are explicitly carried out in our proposed recurrent unit, and also, the two operations directly determine the calculation of the update on the hidden states.

The logical entailment task was introduced recently, and a model [18] was proposed that used given parse trees of the input propositions and was designed specifically for the task. Our model does not receive parsed input and must learn simultaneously to identify, encode, and use the structure necessary to compute logical entailment. NLI (or Recognising Textual Entailment) has assumed a central role in NLP [15]. Neural models have persistently made errors explicable as failure to encode propositional structure [10] and our work targets that particular capability. Our proposed recurrent unit contains essential operations in symbolic computing systems, which encourages the

[1] $\delta_{ij} = 1$ if $i = j$; otherwise $\delta_{ij} = 0$. $\mathbb{I}$ is the identity matrix.

learnt distributed representations to encode more structure-related information. We can thus expect that our TPRU will give strong performance on both tasks even with significantly reduced parameters.

3 Proposed Recurrent Unit: The TPRU

As shown in both LSTM [21] and GRU [12] design, a gating mechanism helps the hidden state at the current time step to directly copy information from the previous time step, and alleviate vanishing and exploding gradient issues. Our proposed recurrent unit keeps the gating mechanism and adopts the design of the input gate in GRU.

At each time step, the TPRU receives two input vectors, one of which is the binding complex from the previous time step $\boldsymbol{b}_{t-1} \in \mathbb{R}^{d\times 1}$ and the other is the vector representation of the external input to the network at the current time step $\boldsymbol{x}_t \in \mathbb{R}^{d'\times 1}$. The TPRU produces a binding complex $\boldsymbol{b}_t \in \mathbb{R}^{d\times 1}$. An input gate $\boldsymbol{g}_t$ is computed to calculate a weighted sum of the information produced at current time step $\widetilde{\boldsymbol{b}}_t$ and the binding complex from the previous time step $\boldsymbol{b}_{t-1}$,

$$\boldsymbol{b}_t = \boldsymbol{g}_t \circ \tanh(\widetilde{\boldsymbol{b}}_t) + (1 - \boldsymbol{g}_t) \circ \boldsymbol{b}_{t-1} \tag{2}$$

$$\boldsymbol{g}_t = \sigma(\boldsymbol{W}_b \boldsymbol{b}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t) \tag{3}$$

where $\sigma(\cdot)$ is the logistic sigmoid function, $\tanh(\cdot)$ is the hyperbolic tangent function, $\circ$ is the Hadamard (element-wise) product, and $\boldsymbol{W}_b \in \mathbb{R}^{d\times d}$ and $\boldsymbol{W}_x \in \mathbb{R}^{d\times d'}$ are matrices of learnable parameters. As we now explain, the calculation of $\widetilde{\boldsymbol{b}}_t$ is carried out by the unbinding and binding operations of TPR (Eq. 1).

3.1 Unbinding Operation

Consider a set of hypothesised unbinding vectors $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_N] \in \mathbb{R}^{d\times N}$. At time step t, these can be used to unbind fillers $\boldsymbol{f}_{b,t}$ from the previous binding complex $\boldsymbol{b}_{t-1}$ using Eq. 1. We posit a matrix $\boldsymbol{W} \in \mathbb{R}^{d\times d'}$ that transforms the current input $\boldsymbol{x}_t \in \mathbb{R}^{d'\times 1}$ into the binding space $\mathbb{R}^{d\times 1}$ where it too can be unbound, yielding fillers $\boldsymbol{f}_{x,t}$:

$$\boldsymbol{f}_{b,t} = \boldsymbol{U}^\top \boldsymbol{b}_{t-1} \in \mathbb{R}^{N\times 1}, \qquad \boldsymbol{f}_{x,t} = \boldsymbol{U}^\top \boldsymbol{W} \boldsymbol{x}_t \in \mathbb{R}^{N\times 1}. \tag{4}$$

A strong sparsity constraint is enforced by applying a rectified linear unit (ReLU) to both $\boldsymbol{f}_{b,t}$ and $\boldsymbol{f}_{x,t}$ [45], and their interaction is calculated by taking the square of the sum of the two sparse vectors. The resulting vector $\widetilde{\boldsymbol{f}}_t$ is then normalised to form a distribution $\boldsymbol{f}_t$.

$$(\widetilde{\boldsymbol{f}}_{b,t})_n = \text{ReLU}\left((\boldsymbol{f}_{b,t})_n + b_b\right), \qquad (\widetilde{\boldsymbol{f}}_{x,t})_n = \text{ReLU}\left((\boldsymbol{f}_{x,t})_n + b_x\right), \tag{5}$$

$$(\boldsymbol{f}_t)_n = \frac{\left((\widetilde{\boldsymbol{f}}_{b,t})_n + (\widetilde{\boldsymbol{f}}_{x,t})_n\right)^2}{\sum_{m=1}^{N}\left((\widetilde{\boldsymbol{f}}_{b,t})_m + (\widetilde{\boldsymbol{f}}_{x,t})_m\right)^2}, \qquad \forall n \in \{1, 2, ..., N\}. \tag{6}$$

where b_b and b_x are two scalar parameters for stable learning.

3.2 Binding Operation

Given a hypothesised set of binding role vectors $\boldsymbol{R} = [\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_N] \in \mathbb{R}^{d\times N}$, we apply the binding operation in Eq. 1 to the fillers $\boldsymbol{f}_t$ at time t to get the candidate update for the binding complex $\widetilde{\boldsymbol{b}}_t$,

$$\widetilde{\boldsymbol{b}}_t = \boldsymbol{R}\boldsymbol{f}_t. \tag{7}$$

The gating mechanism controls the weighted sum of the candidate vector $\widetilde{\boldsymbol{b}}_t$ and the previous binding complex $\boldsymbol{b}_{t-1}$ to produce a binding complex $\boldsymbol{b}_t$ at current time step, as given by Eqs. 2 and 3.

3.3 Unbinding and Binding Role Vectors

In the TPRU, there is a matrix of role vectors $\boldsymbol{R}$ used for the binding operation and a matrix of unbinding vectors $\boldsymbol{U}$ used for the unbinding operation. To control the number of parameters in

our proposed unit, instead of directly learning the role and unbinding vectors, a fixed set of vectors $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N] \in \mathbb{R}^{d \times N}$ is sampled from a standard normal distribution, and two linear transformations $\boldsymbol{W}_u, \boldsymbol{W}_r \in \mathbb{R}^{d \times d}$ are learnt to transform $\boldsymbol{V}$ to $\boldsymbol{U} = \boldsymbol{W}_u \boldsymbol{V}$ and $\boldsymbol{R} = \boldsymbol{W}_r \boldsymbol{V}$.

Therefore, in total, our proposed TPRU has five learnable matrices, including $\boldsymbol{W}_u$, $\boldsymbol{W}_r$, $\boldsymbol{W}_b \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_x$, $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$. Compared to the six parameter matrices of the GRU and the eight of the LSTM, the total number of parameters in the TPRU is significantly fewer.

4 Tasks

Two entailment tasks, including an abstract logical entailment task and a relatively realistic natural language entailment task, along with other downstream natural language tasks, are considered to demonstrate that the TPRU is capable of inducing structured representation through learning. Each of the two entailment tasks provides pairs of samples, and for each pair, the model needs to tell whether the first (the premise) entails the second (the hypothesis).

As our goal is to learn structured vector representations, the proposed TPRU serves as an encoding function, which learns to process a proposition or sentence one token at a time, and then produces a vector representation. During learning, two vector representations are produced by the same recurrent unit given a pair of samples, then a simple feature engineering method (e.g. concatenation of the two representations) is applied to form an input vector to a subsequent classifier which makes a final prediction. In general, with a simple classifier, e.g. a linear classifier or a multi-layer perceptron with a single hidden layer, the learning process forces the encoding function to produce high quality representations of samples, either propositions or sentences, and better vector representations lead to stronger performance.

4.1 Logical Entailment

In propositional logic, for a pair of propositions, A and B, the value of $A \vDash B$ is independent of the identities of the shared variables between A and B, and is dependent only on the structure of the expression and the connectives in each subexpression, because of α-equivalence. For example, $p \wedge q \vDash q$ holds no matter how we replace variable p or q with any other variables or propositions. Thus, logical entailment naturally is a good testbed for evaluating a model's ability to carry out abstract, highly structure-sensitive reasoning [18].

Theoretically, it is possible to construct a truth table that contains as rows/worlds all possible combinations of values of variables in both propositions A and B; the value of $A \vDash B$ can be checked by going through every entry in each row. An example is given in the supplementary material. As the logical entailment task emphasises reasoning on connectives, it requires the learnt distributed vector representations to encode the structure of any given proposition to excel at the task.

The dataset used in our experiments has balanced positive and negative classes, and the task difficulty of the training set is comparable to that of the validation set.[2] Five test sets are generated to evaluate the generalisation ability at different difficulty levels: some test sets have significantly more variables and operators than both the training and validation sets (see Table 1).

4.2 Multi-genre Natural Language Inference

Natural language inference (NLI) tasks require inferring word meaning in context as well as the hierarchical relations among constituents in a given sentence (either premise or hypothesis), and then reasoning whether the premise sentence entails the hypothesis sentence or not. Compared to logical entailment, the inference and reasoning in NLI also rely on the identities of words in sentences in addition to their structure. More importantly, the ambiguity and polysemicity of language lead to the impossibility of creating a truth table that lists all cases. Therefore, NLI is an intrinsically hard task.

The Multi-genre Natural Language Inference (MNLI) dataset [44] collected sentence pairs in ten genres; only five genres are available in the training set, while all ten genres are presented in the development set. There are three classes, Entailment, Neutral and Contradiction. The performance of a model on the mismatched genres, which exist in the development but not the training set, tells us

[2] https://github.com/deepmind/logical-entailment-dataset

how well the structure encoded in distributed vector representations of sentences learnt from seen genres in training generalises to sentence pairs in unseen genres. As the nature of NLI tasks requires inferring both word meaning and structure of constituents in a given sentence, supervised training signals from labelled datasets enforce an encoding function to analyse meaning and structure at the same time during learning, which eventually forces distributed vector representations of sentences produced from the learnt encoding function to be structured. Thus, a suitable inductive bias that enhances the ability to learn structures of sentences will enhance success on the MNLI task.

4.3 Downstream Natural Language Tasks

Vector representations of sentences learnt from labelled NLI tasks demonstrate strong transferability and generalisation ability, which indicates that the learnt encoding function can be applied as a general-purpose sentence encoder to other downstream natural language tasks [14]. As our proposed TPRU is also able to map any given sentence into a distributed vector representation, it is reasonable to evaluate the learnt vector representations on other natural language tasks, and the performance of our proposed recurrent unit will tell us the generalisation ability of the learnt representations.

SentEval [13] presents a collection of natural language tasks in various domains, including sentiment analysis (MR [30], SST [40], CR [22], SUBJ [29], MPQA [43]), paraphrase detection (MRPC [17]), semantic relatedness (SICK [26]), question-type classification (TREC [25]) and semantic textual similarity (STS [1, 2, 3, 4, 5]). Except for STS tasks, in which the cosine similarity of a pair of sentence representations is compared with a human-annotated similarity score, each of the tasks requires learning a linear classifier on top of produced sentence representations to make predictions.

Table 1: **Accuracy of each model on the propositional Logical Entailment task.** Each value besides 'Ours' indicates the number of role vectors in our proposed TPRU, and numbers in parentheses refer to results of models with BiDAF. On each set, 'Mean $2^{\#\text{Vars}}$' refers to mean number of worlds of propositions, and the **bold** number indicates the best result among models with the same architecture. '†' indicates that the LSTM with plain architecture overfitted the training set thrice, and '‡' indicates that the GRU with BiDAF architecture failed twice. Our TPRU didn't fail or overfit during learning and provided comparable performance with to the LSTM and GRU.

model		valid	test					# params
			easy	hard	big	massive	exam	
Mean $2^{\#\text{Vars}}$		75.7	81.0	184.4	3310.8	848,570.0	5.8	
Plain (BiDAF) Architecture - dim 64								
LSTM		71.7 (88.5)	71.8 (88.7)	64.1 (**74.5**)	64.2 (**73.8**)	53.7 (**66.8**)	68.3 (**80.0**)	65.5k (230.0k)
GRU		75.1 (87.9)	**77.1** (88.3)	63.7 (72.5)	63.8 (71.3)	54.4 (66.1)	73.7 (78.0)	49.1k (172.4k)
Ours	8	66.8 (86.2)	67.2 (87.1)	59.3 (69.1)	60.9 (68.2)	51.9 (62.5)	67.0 (74.3)	40.1k (131.3k)
	32	73.7 (88.4)	73.7 (88.4)	62.7 (71.1)	62.8 (70.1)	53.0 (64.9)	**76.7** (77.0)	
	128	75.9 (88.5)	76.0 (88.6)	**64.9** (71.5)	64.0 (69.8)	53.8 (64.1)	75.7 (**80.0**)	
	512	**76.8** (**88.6**)	76.8 (**89.2**)	64.4 (72.6)	**64.6** (71.2)	**54.6** (64.4)	75.3 (**80.0**)	
Plain (BiDAF) Architecture - dim 128								
LSTM †		64.5 (**88.6**)	64.2 (**89.3**)	59.7 (**74.7**)	62.1 (73.5)	50.9 (**67.4**)	65.0 (78.3)	196.6k (917.5k)
GRU ‡		**80.8** (86.2)	**80.3** (85.7)	65.9 (69.1)	**66.0** (69.1)	55.0 (63.1)	77.3 (72.7)	147.5k (688.1k)
Ours	8	63.7 (87.1)	63.4 (87.3)	57.5 (69.4)	59.6 (68.1)	51.3 (62.7)	65.0 (76.0)	131.1k (524.3k)
	32	71.5 (88.2)	71.7 (88.5)	62.6 (71.6)	62.4 (70.3)	52.0 (64.4)	78.3 (78.3)	
	128	72.8 (88.4)	73.1 (89.0)	63.8 (72.4)	62.8 (**71.5**)	52.6 (66.3)	71.3 (**80.0**)	
	512	79.6 (**88.6**)	79.6 (89.2)	**66.1** (72.7)	65.9 (70.8)	**55.2** (64.9)	**80.3** (79.7)	

5 Training Details

Experiments are conducted in PyTorch [32] with the Adam optimiser [24] and gradient clipping [31]. Reported results are averaged from the results of three different random initialisations.

5.1 Plain Architecture

For the Logical Entailment task, we train our proposed TPRU as well as LSTM [21] and GRU [12] RNNs for 90 epochs. Only the output at the last time step is regarded as the representation of a given proposition, and two proposition representations are concatenated, as done in previous work [18], and fed into a multi-layer perceptron which has only one hidden layer with ReLU activation function. The initial learning rate is 0.001 and divided by 10 every 30 epochs. The best model is picked based on the performance on the validation set, and then evaluated on all five test sets with different difficulty levels. Symbolic Vocabulary Permutation [18] is applied as data augmentation during learning which systematically replaces variables with randomly sampled variables according to α-equivalence as only connectives matter on this task. Detailed results are presented Table 1.

Table 2: **Results on MNLI.** The number of role vectors in each of our models is indicated by the value besides 'Ours', and results of models with BiDAF architecture are presented in parentheses. In each combination of dimension and architecture, **bold** number refers to the best result among models.

model		MNLI		# params
		dev matched	dev mismatched	
Plain (BiDAF) Architecture - dim 512				
LSTM		72.0 (76.0)	73.2 (75.5)	10.5m (29.4m)
GRU		72.1 (74.2)	72.8 (74.8)	7.9m (22.0m)
Ours	16	72.4 (73.9)	73.5 (75.0)	5.8m (15.7m)
	64	73.0 (74.8)	73.5 (75.5)	
	256	73.1 (75.9)	**73.9** (**76.8**)	
	1024	**73.2** (**76.2**)	73.8 (76.6)	
Plain (BiDAF) Architecture - dim 1024				
LSTM		72.5 (75.5)	73.9 (76.6)	25.2m (83.9m)
GRU		72.6 (74.8)	73.6 (75.9)	18.9m (62.9m)
Ours	16	72.9 (73.9)	73.7 (74.8)	14.7m (46.1m)
	64	73.4 (75.2)	74.4 (76.0)	
	256	73.7 (75.5)	74.6 (76.7)	
	1024	**74.2** (**76.7**)	**74.7** (**77.3**)	

For the MNLI task, our proposed TPRU as well as LSTM and GRU units are trained for 10 epochs. A global max-pooling over time is applied on top of binding complexes produced by each recurrent unit at all time steps to generate the vector representation for a given sentence. Given a pair of generated sentence representations $\boldsymbol{u}$ and $\boldsymbol{v}$, a vector is constructed to represent the difference between two vectors $[\boldsymbol{u}; \boldsymbol{v}; |\boldsymbol{u} - \boldsymbol{v}|; \boldsymbol{u} \circ \boldsymbol{v}]$, where $\boldsymbol{u} \circ \boldsymbol{v}$ is the Hadamard (element-wise) product and $|\boldsymbol{u} - \boldsymbol{v}|$ is the absolute difference, and the vector is fed into a multi-layer perceptron with the same settings as given above. The feature engineering and the choice of classifier are suggested by prior work [37, 41]. The Stanford Natural Language Inference (SNLI) dataset [10] is added as additional training data as recommended [41], and ELMo [33] is applied for producing vector representations of words. The initial learning rate is 0.0001, and kept constant during learning. The best model is chosen according to the averaged classification accuracy on matched (five genres that exist in both training and dev set) and mismatched (five genres in dev set only) set. LSTM and GRU models use the same settings. The performance of each model is presented in Table 2.

For downstream natural language tasks, the parameters in the learnt recurrent unit — our proposed TPRU, LSTM or GRU trained on the MNLI task — are fixed and used to extract vector representations of sentences for each task. Linear logistic regression or softmax regression is applied when additional learning is required to make predictions. Details of hyperparameter settings of the classifiers can be found in the SentEval package.[3] Table 3 presents macro-averaged results, which are Binary (MR, CR, SUBJ, MPQA, and SST), STS (Su., including SICK-R and SICK-Benchmark), STS (Un., including STS12-16), TREC, SICK-E, and MRPC.

[3] https://github.com/facebookresearch/SentEval

Table 3: **Results on downstream tasks in SentEval.** Each value besides 'Ours' indicates the number of role vectors in our proposed TPRU.

Model		Downstream Tasks in SentEval						
		Binary	SST-5	TREC	SICK-E	STS (Su.)	STS (Un.)	MRPC
Measure		Accuracy				Pearson's $\rho \times 100$		Acc./F1
Plain Architecture - dim 512								
LSTM		87.0	47.5	89.7	84.4	81.8	62.5	**77.8** / 83.8
GRU		87.0	47.5	**91.1**	84.8	80.3	62.5	76.9 / 83.4
Ours	16	86.8	47.0	89.5	84.8	80.0	60.7	76.3 / 82.8
	64	87.1	46.9	89.9	85.1	80.8	62.1	76.8 / 83.3
	256	87.2	47.2	90.1	85.2	81.3	62.6	77.4 / **84.1**
	1024	**87.4**	**48.1**	90.5	**85.4**	**82.4**	**62.8**	77.1 / 83.9
Plain Architecture - dim 1024								
LSTM		87.6	47.3	**92.7**	85.0	**81.7**	63.3	77.0 / 83.6
GRU		87.5	**48.9**	92.6	85.8	81.2	62.8	**77.6** / 84.0
Ours	16	87.4	47.5	91.3	85.6	79.6	60.9	76.2 / 83.2
	64	87.8	47.8	92.0	85.6	80.7	62.3	77.5 / 83.8
	256	87.8	47.9	92.5	**86.0**	80.6	63.3	**77.6** / 83.9
	1024	**87.9**	48.5	91.9	85.9	81.5	**63.9**	77.5 / **84.4**

5.2 BiDAF Architecture

Bi-directional Attention Flow (**BiDAF**) [37] has been adopted in various natural language tasks, including machine comprehension [37] and question answering [11], and provides strong performance on NLI tasks [41]. The BiDAF architecture can generally be applied to any task that requires modelling relations between pairs of sequences. As both the Logical Entailment and MNLI tasks require classification of whether sequence A entails sequence B, BiDAF is well-suited here.

The BiDAF architecture contains a layer for encoding two input sequences, and another for encoding the concatenation of the output from the first layer and the context vectors determined by the bi-directional attention mechanism. In our experiments, the dimensions of both layers are set to be the same, and same type of recurrent unit is applied across both layers. The same settings are used for experiments on LSTM, GRU and our TPRU models. Specifically, for TPRU, the recurrent units in both layers have the same number of role vectors. Other learning details are as in the plain architecture. Tables 1 and 2 respectively present results on the Logical Entailment and the MNLI task, with BiDAF results in parentheses.

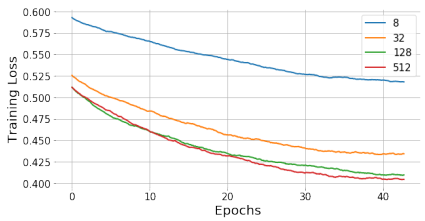
6 Discussion

As presented in Tables 1 and 2, our proposed TPRU provides solid performance. On the Logical Entailment task, the TPRU provides similar performance with the LSTM and GRU under both the plain and BiDAF architectures, but the TPRU has significantly fewer parameters. When it comes to larger dimensionality, our TPRU appears to be more stable than LSTM and GRU during learning as we observed that the LSTM with the plain architecture overfitted the training set terribly in all three trials and the GRU with the BiDAF architecture failed in two out of three trials.

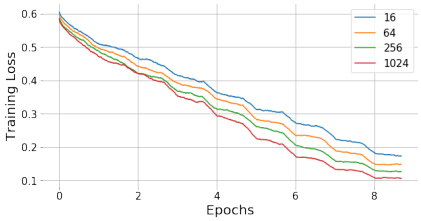
On the MNLI task, our proposed TPRU consistently outperforms both the LSTM and GRU under all four combinations of different dimensions and architectures. Unexpectedly, all models, including LSTM, GRU and our TPRU, provide better results on dev mismatched set than on the matched one, and this is possibly because the dev mismatched set is slightly easier. In Table 3, the TPRU under the plain architecture generalises as well as the LSTM and GRU on 16 downstream tasks in SentEval.

Effect of Increasing the Number of Role Vectors : In TPR [38], the number of role vectors indicates the number of unique symbols that will be used in the final representations of the data.

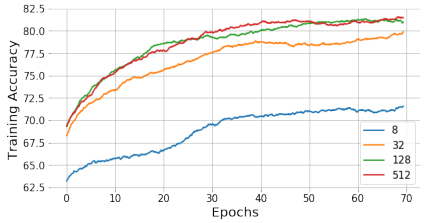
Figure 1: **Training Plots** of our TPRU with difference number of role vectors. In general, our proposed recurrent unit converges faster and leads to better results on both tasks when more role vectors are available, but the total number of parameters remains the same. (Better view in colour.)



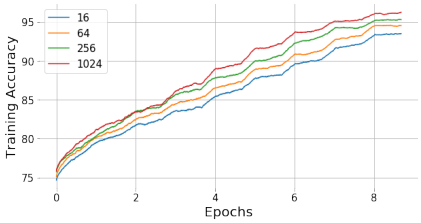
(a) Training loss on Logical Entailment



(b) Training loss on MNLI dataset



(c) Training accuracy on Logical Entailment



(d) Training accuracy on MNLI dataset

Since each symbol is capable of representing a specific substructure of the input data, increasing the number of role vectors eventually leads to more highly structured representations if there is no limit on the dimensionality of role vectors.

Experiments are conducted to show the effect of increasing the number of role vectors on the performance on both tasks. As shown in Tables 1 and 2, adding more role vectors into our proposed TPRU gradually improves the performance on the two entailment tasks. Interestingly, on the MNLI task, our proposed TPRU with only 16 role vectors achieves similar performance to that of the LSTM and GRU, which implies that the distributed representations learnt in both the LSTM and GRU are highly redundant and can be reduced to 16 or even fewer dimensions, which also shows that the LSTM and GRU are not able to extensively exploit the representation space. Meanwhile, the introduced symbolic computing executed by binding and unbinding operations in our proposed unit encourages the model to take advantage of distinct role vectors to learn useful structured representations.

Figure 1 presents the learning curves, including training loss and accuracy, of our proposed TPRU with different number of role vectors on the two entailment tasks. As shown in the graphs, incorporating more role vectors leads to not only better performance, but also faster convergence during training. The observation is consistent on both the Logical Entailment and MNLI tasks.

7 Conclusion

We proposed a recurrent unit (TPRU) that executes binding and unbinding operations in Tensor Product Representations. The explicit execution in our proposed recurrent unit helps it leverage advantages of both distributed representations and neural-symbolic computing, which essentially allows it to learn structured representations. Compared to widely used recurrent units, including LSTM and GRU, our proposed TPRU has many fewer parameters.

The Logical Entailment and Multi-genre Natural Language Inference tasks are selected for experiments as both tasks require highly structured representations to make good predictions. Plain and BiDAF architectures are applied on both tasks. Our proposed TPRU outperforms its comparison partners, the LSTM and GRU, on MNLI tasks with different dimensions and architectures, and it performs similarly to others on the Logical Entailment task. Analysis shows that adding more role

vectors tends to provide stronger results and faster convergence during learning, which parallels the utility of symbols in symbolic computing systems.

We believe that our work pushes the existing research topic on interpreting RNNs into another direction by incorporating symbolic computing. Future work should focus on the interpretabilty of our proposed TPRU as the symbolic computing is explicitly conducted by binding and unbinding operations.

Acknowledgements

Many thanks to Microsoft Research AI, Redmond for supporting the research, and to Elizabeth Clark and YooJung Choi for helpful clarification of concepts. Thanks to Zeyu Chen for the technical support.

References

[1] E. Agirre, C. Banea, C. Cardie, D. M. Cer, M. T. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, and J. Wiebe. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@NAACL-HLT*, 2015.

[2] E. Agirre, C. Banea, C. Cardie, D. M. Cer, M. T. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@COLING*, 2014.

[3] E. Agirre, C. Banea, D. M. Cer, M. T. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*, 2016.

[4] E. Agirre, D. M. Cer, M. T. Diab, and A. Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *SemEval@NAACL-HLT*, 2012.

[5] E. Agirre, D. M. Cer, M. T. Diab, A. Gonzalez-Agirre, and W. Guo. *sem 2013 shared task: Semantic textual similarity. In **SEM@NAACL-HLT*, 2013.

[6] Anonymous. Towards decomposed linguistic representations with holographic reduced representation. In *Under review for ICLR2019*, 2019.

[7] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[8] Y. Bengio, A. C. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828, 2013.

[9] T. R. Besold, A. S. d'Avila Garcez, S. Bader, H. Bowman, P. M. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017.

[10] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*, 2015.

[11] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[13] A. Conneau and D. Kiela. Senteval: An evaluation toolkit for universal sentence representations. In *LREC*, 2018.

[14] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, 2017.

[15] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In *MLCW*, 2005.

[16] V. R. de Sa. Learning classification with unlabeled data. In *NIPS*, pages 112–119, 1993.

[17] W. B. Dolan, C. Quirk, and C. Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*, 2004.

[18] R. Evans, D. Saxton, D. Amos, P. Kohli, and E. Grefenstette. Can neural networks understand logical entailment? In *ICLR*, 2018.

[19] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec. Embedding logical queries on knowledge graphs. *arXiv preprint arXiv:1806.01445*, 2018.

[20] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. 1984.

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[22] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.
[23] Q. Huang, P. Smolensky, X. He, L. Deng, and D. O. Wu. Tensor product generation networks for deep NLP modeling. In *NAACL-HLT*, 2018.
[24] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[25] X. Li and D. Roth. Learning question classifiers. In *COLING*, 2002.
[26] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, 2014.
[27] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 2017.
[28] H. Palangi, P. Smolensky, X. He, and L. Deng. Deep learning of grammatically-interpretable representations through question-answering. *CoRR*, abs/1705.08432, 2017.
[29] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, 2004.
[30] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.
[31] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.
[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.
[33] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. S. Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018.
[34] J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1–2):77–105, 1990.
[35] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.
[36] A. Santoro, R. Faulkner, D. Raposo, J. W. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. P. Lillicrap. Relational recurrent neural networks. *CoRR*, abs/1806.01822, 2018.
[37] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. 2017.
[38] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46:159–216, 1990.
[39] R. Socher, C. D. Manning, and A. Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.
[40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.
[41] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018.
[42] W. Wang, R. Arora, K. Livescu, and J. A. Bilmes. On deep multi-view representation learning. In *ICML*, 2015.
[43] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210, 2005.
[44] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.
[45] Z. Yang, J. J. Zhao, B. Dhingra, K. He, W. W. Cohen, R. Salakhutdinov, and Y. LeCun. Glomo: Unsupervisedly learned relational graphs as transferable representations. *CoRR*, abs/1806.05662, 2018.
[46] Çaglar Gülçehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. M. Hermann, P. W. Battaglia, V. Bapst, D. Raposo, A. Santoro, and N. de Freitas. Hyperbolic attention networks. *CoRR*, abs/1805.09786, 2018.

A A Look-up Table Approach to Logical Entailment

Given proposition $A = p \wedge q$ and proposition $B = q$, a truth table presents all possible combinations of values of p and q, and values of A and B for each combination of p and q. $A \vDash B$ holds iff, as here, in every row/world, the value of A is less than or equal to that of B.

Table 4: A truth table for $A = p \wedge q$ and $B = q$.

p	q	A	B	
T	T	T (1)	T (1)	$(1 = 1)$
T	F	F (0)	F (0)	$(0 = 0)$
F	T	F (0)	T (1)	$(0 < 1)$
F	F	F (0)	F (0)	$(0 = 0)$