A NEURAL-SYMBOLIC APPROACH TO NATURAL LANGUAGE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning (DL) has in recent years been widely used in natural language processing (NLP) applications due to its superior performance. However, while natural languages are rich in grammatical structure, DL has not been able to explicitly represent and enforce such structures. This paper proposes a new architecture to bridge this gap by exploiting tensor product representations (TPR), a structured neural-symbolic framework developed in cognitive science over the past 20 years, with the aim of integrating DL with explicit language structures and rules. We call it the *Tensor Product Generation Network* (**TPGN**), and apply it to 1) image captioning, 2) classification of the part of speech of a word, and 3) identification of the phrase structure of a sentence. The key ideas of TPGN are: 1) unsupervised learning of *role-unbinding vectors* of words via a TPR-based deep neural network, and 2) integration of TPR with typical DL architectures including Long Short-Term Memory (LSTM) models. The novelty of our approach lies in its ability to generate a sentence and extract partial grammatical structure of the sentence by using role-unbinding vectors, which are obtained in an unsupervised manner. Experimental results demonstrate the effectiveness of the proposed approach.

1 INTRODUCTION

In this paper we attempt to address a triple challenge:

- to achieve good performance on difficult tasks — image captioning, classification of the part of speech (POS) of a word, and identification of the phrase structure of a sentence — by
- producing *grammatically interpretable* representations that are acquired through deep learning
- in a Deep Neural Network (DNN) architecture possessing a *sound rationale* based in a general theory of intelligent information processing that integrates neural and symbolic computation.

Deep learning is an important tool in many current natural language processing (NLP) applications. However, language rules or structures cannot be explicitly represented in deep learning architectures. The tensor product representation developed in Smolensky (1990); Smolensky & Legendre (2006) has the potential of integrating deep learning with explicit rules (such as logical rules, grammar rules, or rules that summarize real-world knowledge). This paper develops a TPR approach for deep-learning-based NLP applications, introducing the *Tensor Product Generation Network (**TPGN**)* architecture. To demonstrate the effectiveness of the proposed architecture, we apply it to three important NLP applications: 1) image captioning, 2) classification of POS of a word (a.k.a. POS tagging), and 3) identification of the phrase structure of a sentence.

A TPGN model generates natural language descriptions via learned representations. The representations learned in a crucial layer of the TPGN can be interpreted as encoding grammatical roles for the words being generated. This layer corresponds to the role-encoding component of a general, independently-developed architecture for neural computation of symbolic functions, including the generation of linguistic structures. The key to this architecture is the notion of *Tensor Product Representation (**TPR**)*, in which vectors embedding symbols (e.g., `lives`, `frodo`) are bound to vectors embedding structural roles (e.g., `verb`, `subject`) and combined to generate vectors embedding

symbol structures ([frodo lives]). TPRs provide the representational foundations for a general computational architecture called *Gradient Symbolic Computation (**GSC**)*, and applying GSC to the task of natural language generation yields the specialized architecture defining the model presented here. The generality of GSC means that the results reported here have implications well beyond the particular tasks we address here.

The paper is organized as follows. Section 2 discusses related work. In Section 3, we review the basics of tensor product representation. Section 4 presents the rationale for our proposed architecture. Section 5 describes our proposed model in detail. In Section 6, we present our experimental results. Finally, Section 7 concludes the paper.

2 RELATED WORK

Deep learning plays a dominant role in many NLP applications due to its exceptional performance. Hence, we focus on recent deep-learning-based literature for NLP applications, especially three important NLP applications: 1) image captioning, 2) POS tagging, and 3) identification of the phrase structure of a sentence.

Most existing DL-based image captioning methods (Mao et al. (2015); Vinyals et al. (2015); Devlin et al. (2015); Chen & Lawrence Zitnick (2015); Donahue et al. (2015); Karpathy & Fei-Fei (2015); Kiros et al. (2014a;b)) involve two phases/modules: 1) image analysis, typically by a Convolutional Neural Network (CNN), and 2) a language model for caption generation (Fang et al. (2015)). The CNN module takes an image as input and outputs an image feature vector or a list of detected words with their probabilities. The language model is used to create a sentence (caption) out of the detected words or the image feature vector produced by the CNN.

There are mainly two approaches to natural language generation in image captioning. The first approach takes the words detected by a CNN as input, and uses a probabilistic model, such as a maximum entropy (ME) language model, to arrange the detected words into a sentence. The second approach takes the penultimate activation layer of the CNN as input to a Recurrent Neural Network (RNN), which generates a sequence of words (the caption) (Vinyals et al. (2015)).

The work reported here follows the latter approach, adopting a CNN + RNN-generator architecture. Specifically, instead of using a conventional RNN, we propose a recurrent network that has substructure derived from the general GSC architecture: one recurrent subnetwork holds an encoding $\mathbf{S}$ — which is treated as an approximation of a TPR — of the words yet to be produced, while another recurrent subnetwork generates a sequence of vectors that is treated as a sequence of roles to be unbound from $\mathbf{S}$, in effect, reading out a word at a time from $\mathbf{S}$. Examining how the model deploys these roles allows us to interpret them in terms of grammatical categories; roughly speaking, a sequence of categories is generated and the words stored in $\mathbf{S}$ are retrieved and spelled out via their categories.

The second task we consider is POS tagging. Methods for automatic POS tagging include unigram tagging, bigram tagging, tagging using Hidden Markov Models (which are generative sequence models), maximum entropy Markov models (which are discriminative sequence models), rule-based tagging, and tagging using bidirectional maximum entropy Markov models (Jurafsky & Martin (2017)). The celebrated Stanford POS tagger of Manning (2017) uses a bidirectional version of the maximum entropy Markov model called a cyclic dependency network in Toutanova et al. (2003), which achieves 97.24% accuracy on the Penn Treebank WSJ data set (Toutanova et al. (2003)).

Methods for automatic phrase detection/classification, our third task, include supervised learning for a classifier with a set of features extracted from a context window that surrounds the word to be classified (Jurafsky & Martin (2017)). The input of the classifier is features extracted from the context window including the words themselves, their parts-of-speech, and the phrase types of the preceding inputs in the window. The output of the classifier will be the type of the phrase containing the word.

3 REVIEW OF TENSOR PRODUCT REPRESENTATION

Tensor product representation (TPR) is a general framework for embedding a space of symbol structures $\mathfrak{S}$ into a vector space. This embedding enables neural network operations to perform

symbolic computation, including computations that provide considerable power to symbolic NLP systems (Smolensky & Legendre (2006); Smolensky (2012)). Motivated by these successful examples, we are inspired to extend the TPR to the challenging task of learning image captioning. And as a by-product, the symbolic character of TPRs makes them amenable to conceptual interpretation in a way that standard learned neural network representations are not.

A particular TPR embedding is based in a *filler/role decomposition* of $\mathfrak{S}$. A relevant example is when $\mathfrak{S}$ is the set of strings over an alphabet $\{\mathtt{a}, \mathtt{b}, \ldots\}$. One filler/role decomposition deploys the *positional roles* $\{r_k\}, k \in \mathbb{N}$, where the *filler/role binding* $\mathtt{a}/r_k$ assigns the 'filler' (symbol) $\mathtt{a}$ to the kth position in the string. A string such as $\mathtt{abc}$ is uniquely determined by its filler/role bindings, which comprise the (unordered) set $\mathfrak{B}(\mathtt{abc}) = \{\mathtt{b}/r_2, \mathtt{a}/r_1, \mathtt{c}/r_3\}$. Reifying the notion *role* in this way is key to TPR's ability to encode complex symbol *structures*.

Given a selected filler/role decomposition of the symbol space, a particular TPR is determined by an embedding that assigns to each filler a vector in a vector space $V_F \cong \mathbb{R}^{d_F}$, and a second embedding that assigns to each role a vector in a space $V_R \cong \mathbb{R}^{d_R}$. The vector embedding a symbol $\mathtt{a}$ is denoted by $\mathbf{f}_\mathtt{a}$ and is called a *filler vector*; the vector embedding a role r_k is $\mathbf{r_k}$ and called a *role vector*. The TPR for $\mathtt{abc}$ is then the following 2-index tensor in $V_F \otimes V_R \cong \mathbb{R}^{d_F \times d_R}$:

$$\mathbf{S}_\mathtt{abc} = \mathbf{f}_\mathtt{b} \otimes \mathbf{r_2} + \mathbf{f}_\mathtt{a} \otimes \mathbf{r_1} + \mathbf{f}_\mathtt{c} \otimes \mathbf{r_3}, \tag{1}$$

where $\otimes$ denotes the tensor product. The tensor product is a generalization of the vector outer product that is recursive; recursion is exploited in TPRs for, e.g., the distributed representation of trees, the neural encoding of formal grammars in connection weights, and the theory of neural computation of recursive symbolic functions. Here, however, it suffices to use the outer product; using matrix notation we can write (1) as:

$$\mathbf{S}_\mathtt{abc} = \mathbf{f}_\mathtt{b}\mathbf{r_2}^\top + \mathbf{f}_\mathtt{a}\mathbf{r_1}^\top + \mathbf{f}_\mathtt{c}\mathbf{r_3}^\top. \tag{2}$$

Generally, the embedding of any symbol structure $\mathtt{S} \in \mathfrak{S}$ is $\sum\{\mathbf{f}_i \otimes \mathbf{r}_i \mid \mathtt{f}_i/r_i \in \mathfrak{B}(\mathtt{S})\}$; here: $\sum\{\mathbf{f}_i\mathbf{r}_i^\top \mid \mathtt{f}_i/r_i \in \mathfrak{B}(\mathtt{S})\}$ (Smolensky (1990); Smolensky & Legendre (2006)).

A key operation on TPRs, central to the work presented here, is *unbinding*, which undoes binding. Given the TPR in (2), for example, we can unbind $\mathbf{r_2}$ to get $\mathbf{f}_\mathtt{b}$; this is achieved simply by $\mathbf{f}_\mathtt{b} = \mathbf{S}_\mathtt{abc}\mathbf{u_2}$. Here $\mathbf{u_2}$ is *the unbinding vector dual to* the binding vector $\mathbf{r_2}$. To make such exact unbinding possible, the role vectors should be chosen to be linearly independent. (In that case the unbinding vectors are the rows of the inverse of the matrix containing the binding vectors as columns, so that $\mathbf{r_2} \cdot \mathbf{u_2} = 1$ while $\mathbf{r_k} \cdot \mathbf{u_2} = 0$ for all other role vectors $\mathbf{r_k} \neq \mathbf{r_2}$; this entails that $\mathbf{S}_\mathtt{abc}\mathbf{u_2} = \mathbf{b}$, the filler vector bound to $\mathbf{r_2}$. Replacing the matrix inverse with the pseudo-inverse allows approximate unbinding when the role vectors are not linearly independent).

4 A TPR-CAPABLE GENERATION ARCHITECTURE

In this work we propose an approach to network architecture design we call the *TPR-capable method*. The architecture we use (see Fig. 1) is designed so that TPRs could, in theory, be used within the architecture to perform the target task — here, generating a caption one word at a time. Unlike previous work where TPRs are hand-crafted, in our work, end-to-end deep learning will induce representations which the architecture can use to generate captions effectively.

In this section, we consider the problem of image captioning. As shown in Fig. 1, our proposed system is denoted by $\mathcal{N}$. The input of $\mathcal{N}$ is an image feature vector $\mathbf{v}$ and the output of $\mathcal{N}$ is a caption. The image feature vector $\mathbf{v}$ is extracted from a given image by a pre-trained CNN. The first part of our system $\mathcal{N}$ is a *sentence-encoding subnetwork* $\mathcal{S}$ which maps $\mathbf{v}$ to a representation S which will drive the entire caption-generation process; S contains all the image-specific information for producing the caption. (We will call a caption a "sentence" even though it may in fact be just a noun phrase.)

If S were a TPR of the caption itself, it would be a matrix (or 2-index tensor) $\mathbf{S}$ which is a sum of matrices, each of which encodes the binding of one word to its role in the sentence constituting the caption. To serially read out the words encoded in $\mathbf{S}$, in iteration 1 we would *unbind* the first word from $\mathbf{S}$, then in iteration 2 the second, and so on. As each word is generated, $\mathbf{S}$ could update itself, for example, by subtracting out the contribution made to it by the word just generated; $\mathbf{S}_t$ denotes

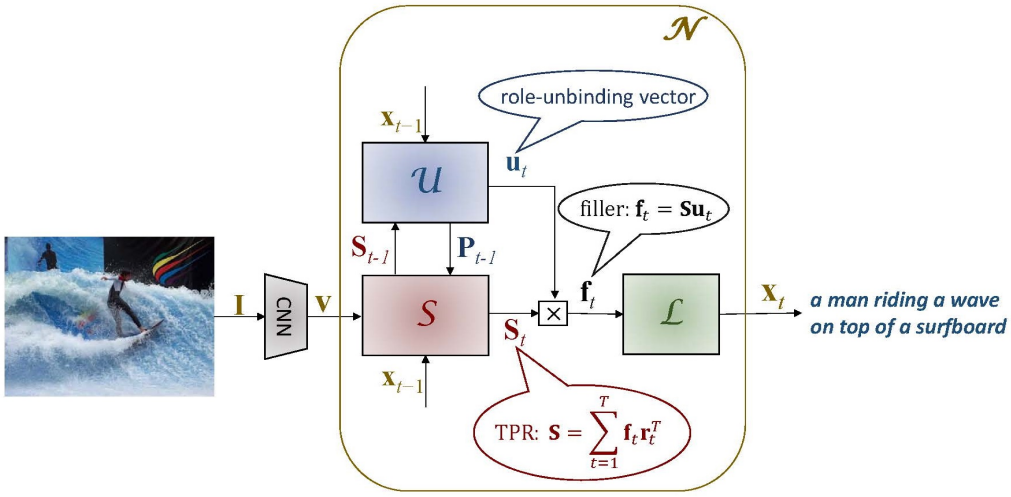

Figure 1: Architecture of TPGN, a TPR-capable generation network. "⊠" denotes the matrix-vector product.

the value of $\mathbf{S}$ when word w_t is generated. At time step t we would unbind the role r_t occupied by word w_t of the caption. So the second part of our system $\mathcal{N}$ — the *unbinding subnetwork* $\mathcal{U}$ — would generate, at iteration t, the *unbinding vector* $\mathbf{u}_t$. Once $\mathcal{U}$ produces the unbinding vector $\mathbf{u}_t$, this vector would then be applied to $\mathbf{S}$ to extract the symbol $\mathbf{f_t}$ that occupies word t's role; the symbol represented by $\mathbf{f_t}$ would then be decoded into word w_t by the third part of $\mathcal{N}$, i.e., the *lexical decoding subnetwork* $\mathcal{L}$, which outputs $\mathbf{x}_t$, the 1-hot-vector encoding of w_t.

Recalling that unbinding in TPR is achieved by the matrix-vector product, the key operation in generating w_t is thus the unbinding of r_t within $\mathbf{S}$, which amounts to simply:

$$\mathbf{S}_t \mathbf{u_t} = \mathbf{f_t}. \tag{3}$$

This matrix-vector product is denoted "⊠" in Fig. 1.

Thus the system $\mathcal{N}$ of Fig. 1is TPR-capable. This is what we propose as the Tensor-Product Generation Network (TPGN) architecture. The learned representation $\mathbf{S}$ will not be proven to literally be a TPR, but by analyzing the unbinding vectors $\mathbf{u}_t$ the network learns, we will gain insight into the process by which the learned matrix $\mathbf{S}$ gives rise to the generated caption.

What type of roles might the unbinding vectors be unbinding? A TPR for a caption could in principle be built upon *positional roles*, *syntactic/semantic roles*, or some combination of the two. In the caption ***a man standing in a room with a suitcase***, the initial *a* and *man* might respectively occupy the positional roles of POS(ITION)$_1$ and POS$_2$; *standing* might occupy the syntactic role of VERB; *in* the role of SPATIAL-P(REPOSITION); while *a room with a suitcase* might fill a 5-role schema DET(ERMINER)$_1$ N(OUN)$_1$ P DET$_2$ N$_2$. In fact we will see evidence below that our network learns just this kind of hybrid role decomposition.

What form of information does the sentence-encoding subnetwork $\mathcal{S}$ need to encode in $\mathbf{S}$? Continuing with the example of the previous paragraph, $\mathbf{S}$ needs to be some approximation to the TPR summing several filler/role binding matrices. In one of these bindings, a filler vector $\mathbf{f}_a$ — which the lexical subnetwork $\mathcal{L}$ will map to the article *a* — is bound (via the outer product) to a role vector $\mathbf{r}_{\text{POS}_1}$ which is the dual of the first unbinding vector produced by the unbinding subnetwork $\mathcal{U}$: $\mathbf{u}_{\text{POS}_1}$. In the first iteration of generation the model computes $\mathbf{S}_1 \mathbf{u}_{\text{POS}_1} = \mathbf{f}_a$, which $\mathcal{L}$ then maps to *a*. Analogously, another binding approximately contained in $\mathbf{S}_2$ is $\mathbf{f}_{man} \mathbf{r}_{\text{POS}_2}^\top$. There are corresponding bindings for the remaining words of the caption; these employ syntactic/semantic roles. One example is $\mathbf{f}_{standing} \mathbf{r}_V^\top$. At iteration 3, $\mathcal{U}$ decides the next word should be a verb, so it generates the unbinding vector $\mathbf{u}_V$ which when multiplied by the current output of $\mathcal{S}$, the matrix $\mathbf{S}_3$, yields a filler vector $\mathbf{f}_{standing}$ which $\mathcal{L}$ maps to the output *standing*. $\mathcal{S}$ decided the caption should deploy *standing* as a verb and included

in $\mathbf{S}$ the binding $\mathbf{f}_{standing}\mathbf{r}_V^\top$. It similarly decided the caption should deploy *in* as a spatial preposition, including in $\mathbf{S}$ the binding $\mathbf{f}_{in}\mathbf{r}_{\text{SPATIAL-P}}^\top$; and so on for the other words in their respective roles in the caption.

5 SYSTEM DESCRIPTION

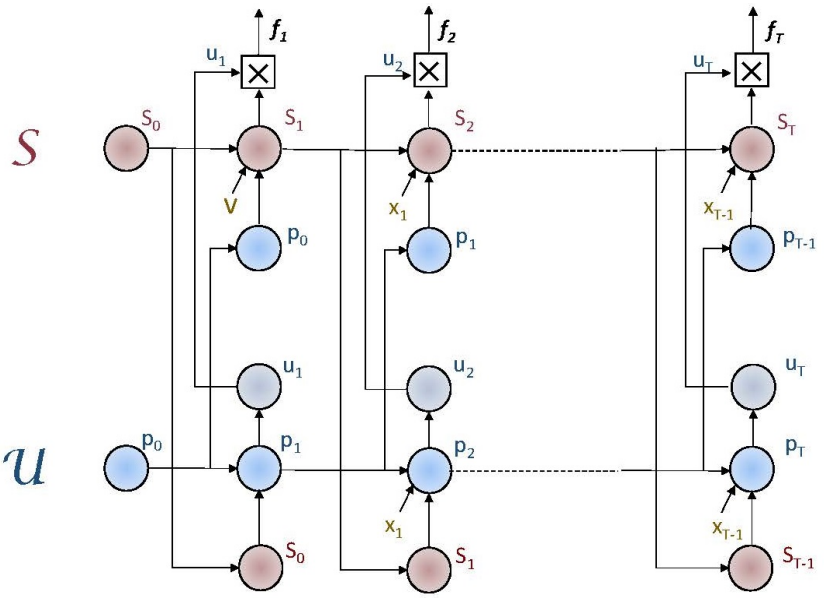

Figure 2: The sentence-encoding subnet $\mathcal{S}$ and the unbinding subnet $\mathcal{U}$ are inter-connected LSTMs; $\mathbf{v}$ encodes the visual input while the $\mathbf{x}_t$ encode the words of the output caption.

The unbinding subnetwork $\mathcal{U}$ and the sentence-encoding network $\mathcal{S}$ of Fig. 1 are each implemented as (1-layer, 1-directional) LSTMs (see Fig. 2); the lexical subnetwork $\mathcal{L}$ is implemented as a linear transformation followed by a softmax operation. In the equations below, the LSTM variables internal to the $\mathcal{S}$ subnet are indexed by 1 (e.g., the forget-, input-, and output-gates are respectively $\hat{\mathbf{f}}_1, \hat{\mathbf{i}}_1, \hat{\mathbf{o}}_1$) while those of the unbinding subnet $\mathcal{U}$ are indexed by 2.

Thus the state updating equations for $\mathcal{S}$ are, for $t = 1, \cdots, T$ = caption length:

$$\hat{\mathbf{f}}_{1,t} = \sigma_g(\mathbf{W}_{1,f}\mathbf{p}_{t-1} - \mathbf{D}_{1,f}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,f}\hat{\mathbf{S}}_{t-1}) \quad (4)$$

$$\hat{\mathbf{i}}_{1,t} = \sigma_g(\mathbf{W}_{1,i}\mathbf{p}_{t-1} - \mathbf{D}_{1,i}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,i}\hat{\mathbf{S}}_{t-1}) \quad (5)$$

$$\hat{\mathbf{o}}_{1,t} = \sigma_g(\mathbf{W}_{1,o}\mathbf{p}_{t-1} - \mathbf{D}_{1,o}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,o}\hat{\mathbf{S}}_{t-1}) \quad (6)$$

$$\mathbf{g}_{1,t} = \sigma_h(\mathbf{W}_{1,c}\mathbf{p}_{t-1} - \mathbf{D}_{1,c}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{1,c}\hat{\mathbf{S}}_{t-1}) \quad (7)$$

$$\mathbf{c}_{1,t} = \hat{\mathbf{f}}_{1,t} \odot \mathbf{c}_{1,t-1} + \hat{\mathbf{i}}_{1,t} \odot \mathbf{g}_{1,t} \quad (8)$$

$$\hat{\mathbf{S}}_t = \hat{\mathbf{o}}_{1,t} \odot \sigma_h(\mathbf{c}_{1,t}) \quad (9)$$

where $\hat{\mathbf{f}}_{1,t}$, $\hat{\mathbf{i}}_{1,t}$, $\hat{\mathbf{o}}_{1,t}$, $\mathbf{g}_{1,t}$, $\mathbf{c}_{1,t}$, $\hat{\mathbf{S}}_t \in \mathbb{R}^{d\times d}$, $\mathbf{p}_t \in \mathbb{R}^d$, $\sigma_g(\cdot)$ is the (element-wise) logistic sigmoid function; $\sigma_h(\cdot)$ is the hyperbolic tangent function; the operator $\odot$ denotes the Hadamard (element-wise) product; $\mathbf{W}_{1,f}, \mathbf{W}_{1,i}, \mathbf{W}_{1,o}, \mathbf{W}_{1,c} \in \mathbb{R}^{d\times d\times d}$, $\mathbf{D}_{1,f}, \mathbf{D}_{1,i}, \mathbf{D}_{1,o}, \mathbf{D}_{1,c} \in \mathbb{R}^{d\times d\times d}$, $\mathbf{U}_{1,f}$, $\mathbf{U}_{1,i}$, $\mathbf{U}_{1,o}$, $\mathbf{U}_{1,c} \in \mathbb{R}^{d\times d\times d\times d}$. For clarity, biases — included throughout the model — are omitted from all equations in this paper. The initial state $\hat{\mathbf{S}}_0$ is initialized by:

$$\hat{\mathbf{S}}_0 = \mathbf{C}_s(\mathbf{v} - \bar{\mathbf{v}}) \quad (10)$$

where $\mathbf{v} \in \mathbb{R}^{2048}$ is the vector of visual features extracted from the current image by ResNet (Gan et al. (2017)) and $\bar{\mathbf{v}}$ is the mean of all such vectors; $\mathbf{C}_s \in \mathbb{R}^{d\times d\times 2048}$. On the output side, $\mathbf{x}_t \in \mathbb{R}^V$ is a 1-hot vector with dimension equal to the size of the caption vocabulary, V, and $\mathbf{W}_e \in \mathbb{R}^{d\times V}$ is a word embedding matrix, the i-th column of which is the embedding vector of the i-th word in

the vocabulary; it is obtained by the Stanford GLoVe algorithm with zero mean (Pennington et al. (2017)). $\mathbf{x}_0$ is initialized as the one-hot vector corresponding to a "start-of-sentence" symbol.

For $\mathcal{U}$ in Fig. 1, the state updating equations are:

$$\hat{\mathbf{f}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,f} - \mathbf{D}_{2,f}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,f}\mathbf{p}_{t-1}) \tag{11}$$

$$\hat{\mathbf{i}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,i} - \mathbf{D}_{2,i}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,i}\mathbf{p}_{t-1}) \tag{12}$$

$$\hat{\mathbf{o}}_{2,t} = \sigma_g(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,o} - \mathbf{D}_{2,o}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,o}\mathbf{p}_{t-1}) \tag{13}$$

$$\mathbf{g}_{2,t} = \sigma_h(\hat{\mathbf{S}}_{t-1}\mathbf{w}_{2,c} - \mathbf{D}_{2,c}\mathbf{W}_e\mathbf{x}_{t-1} + \mathbf{U}_{2,c}\mathbf{p}_{t-1}) \tag{14}$$

$$\mathbf{c}_{2,t} = \hat{\mathbf{f}}_{2,t} \odot \mathbf{c}_{2,t-1} + \hat{\mathbf{i}}_{2,t} \odot \mathbf{g}_{2,t} \tag{15}$$

$$\mathbf{p}_t = \hat{\mathbf{o}}_{2,t} \odot \sigma_h(\mathbf{c}_{2,t}) \tag{16}$$

where $\mathbf{w}_{2,f}, \mathbf{w}_{2,i}, \mathbf{w}_{2,o}, \mathbf{w}_{2,c} \in \mathbb{R}^d$, $\mathbf{D}_{2,f}, \mathbf{D}_{2,i}, \mathbf{D}_{2,o}, \mathbf{D}_{2,c} \in \mathbb{R}^{d\times d}$, and $\mathbf{U}_{2,f}, \mathbf{U}_{2,i}, \mathbf{U}_{2,o}, \mathbf{U}_{2,c} \in \mathbb{R}^{d\times d}$. The initial state $\mathbf{p}_0$ is the zero vector.

The dimensionality of the crucial vectors shown in Fig. 1, $\mathbf{u}_t$ and $\mathbf{f}_t$, is increased from $d \times 1$ to $d^2 \times 1$ as follows. A block-diagonal $d^2 \times d^2$ matrix $\mathbf{S}_t$ is created by placing d copies of the $d \times d$ matrix $\hat{\mathbf{S}}_t$ as blocks along the principal diagonal. This matrix is the output of the sentence-encoding subnetwork $\mathcal{S}$. Now, following Eq. (3), the 'filler vector' $\mathbf{f}_t \in \mathbb{R}^{d^2}$ — 'unbound' from the sentence representation $\mathbf{S}_t$ with the 'unbinding vector' $\mathbf{u}_t$ — is obtained by Eq. (17).

$$\mathbf{f}_t = \mathbf{S}_t\mathbf{u}_t \tag{17}$$

Here $\mathbf{u}_t \in \mathbb{R}^{d^2}$, the output of the unbinding subnetwork $\mathcal{U}$, is computed as in Eq. (18), where $\mathbf{W}_u \in \mathbb{R}^{d^2\times d}$ is $\mathcal{U}$'s output weight matrix.

$$\mathbf{u}_t = \sigma_h(\mathbf{W}_u\mathbf{p}_t) \tag{18}$$

Finally, the lexical subnetwork $\mathcal{L}$ produces a decoded word $\mathbf{x}_t \in \mathbb{R}^V$ by

$$\mathbf{x}_t = \sigma_s(\mathbf{W}_x\mathbf{f}_t) \tag{19}$$

where $\sigma_s(\cdot)$ is the softmax function and $\mathbf{W}_x \in \mathbb{R}^{V\times d^2}$ is the overall output weight matrix. Since $\mathbf{W}_x$ plays the role of a word de-embedding matrix, we can set

$$\mathbf{W}_x = (\mathbf{W}_e)^\top \tag{20}$$

where $\mathbf{W}_e$ is the word-embedding matrix. Since $\mathbf{W}_e$ is pre-defined, we directly set $\mathbf{W}_x$ by Eq. (20) without training $\mathcal{L}$ through Eq. (19). Note that $\mathcal{S}$ and $\mathcal{U}$ are learned jointly through the end-to-end training.

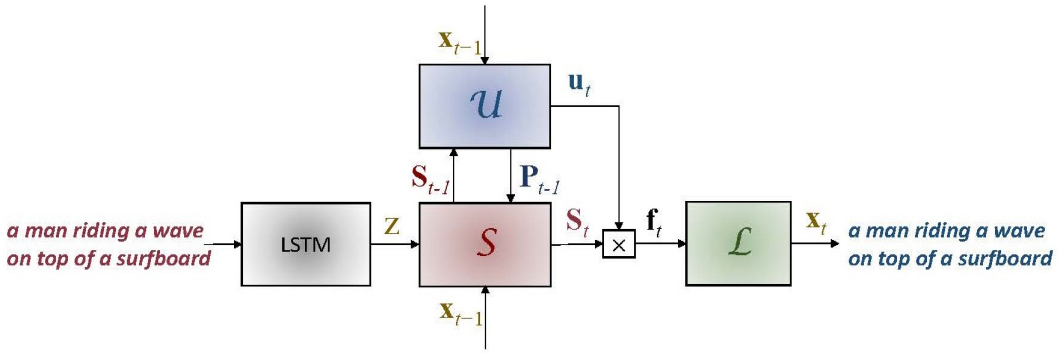

Figure 3: Pre-training of TPGN.

Fig. 3 shows a pre-training method for initializing TPGN. During the pre-training phase, there is no image input, i.e., image feature vector $\mathbf{v} = 0$. In Fig. 3, at time $t = -T + 1$, the LSTM module takes a sentence of length T as input and outputs a vector $\mathbf{z}$ ($\mathbf{z} \in \mathbb{R}^{d^2}$) at time $t = 0$. That is, the LSTM

converts a sentence into $\mathbf{z}$, which is the input of TPGN. We use end-to-end training to train the whole system shown in Fig. 3. After finishing pre-training, we let $\mathbf{z} = 0$ and use images as input to train the TPGN in Fig. 1, initialized by the pretrained parameter values.

The architecture in Fig. 3 allows us to design a POS tagger and a phrase detector/classifier.

A POS tagger is designed in the following way. First, apply a given sentence $\mathbf{x}_1, \cdots, \mathbf{x}_T$ to the input of a trained system shown in Fig. 3. The resulting sequence of vectors $\mathbf{u}_1, \cdots, \mathbf{u}_T$ are considered as the feature for sentence $\mathbf{x}_1, \cdots, \mathbf{x}_T$. A POS tagger takes the vectors $\mathbf{u}_1, \cdots, \mathbf{u}_T$ as input and produces the POS tag for each word in the sentence $\mathbf{x}_1, \cdots, \mathbf{x}_T$. A POS tagger can be realized by a support vector machine or bidirectional LSTM. To train the POS tagger in a supervised manner, we need to have input features $\mathbf{u}_1, \cdots, \mathbf{u}_T$ and output the POS tag of each word $\mathbf{x}_t$ ($t = 1, \cdots, T$). The POS tag of each word can be obtained by the Stanford tagger (Manning (2017)).

A phrase classifier is designed in an analogous way. We change only the outputs of the classifier, replacing the POS with the phrase type of each word, which can be obtained by the Stanford parser (Manning (2017)). A phrase classifier can be realized by a bidirectional LSTM.

6 EXPERIMENTAL RESULTS

6.1 DATASET

To evaluate the performance of our proposed architecture, we use the COCO dataset (COCO (2017)). The COCO dataset contains 123,287 images, each of which is annotated with at least 5 captions. We use the same pre-defined splits as Karpathy & Fei-Fei (2015); Gan et al. (2017): 113,287 images for training, 5,000 images for validation, and 5,000 images for testing. We use the same vocabulary as that employed in Gan et al. (2017), which consists of 8,791 words.

6.2 EVALUATION OF IMAGE CAPTIONING SYSTEM

For the CNN of Fig. 1, we used ResNet-152 (He et al. (2016)), pretrained on the ImageNet dataset. The feature vector $\mathbf{v}$ has 2048 dimensions. Word embedding vectors in $\mathbf{W}_e$ are downloaded from the web (Pennington et al. (2017)). The model is implemented in TensorFlow (Abadi et al. (2015)) with the default settings for random initialization and optimization by backpropagation.

In our experiments, we choose $d = 25$ (where d is the dimension of vector $\mathbf{p}_t$). The dimension of $\mathbf{S}_t$ is 625×625 (while $\hat{\mathbf{S}}_t$ is 25×25); the vocabulary size $V = 8,791$; the dimension of $\mathbf{u}_t$ and $\mathbf{f}_t$ is $d^2 = 625$.

Table 1: Performance of the proposed TPGN model on the COCO dataset.

Methods	METEOR	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDEr
NIC Vinyals et al. (2015)	–	0.666	0.451	0.304	0.203	–
CNN-LSTM	0.238	0.698	0.525	0.390	0.292	0.889
TPGN	**0.243**	**0.709**	**0.539**	**0.406**	**0.305**	**0.909**

The main evaluation results on the MS COCO dataset are reported in Table 1. The widely-used BLEU (Papineni et al. (2002)), METEOR (Banerjee & Lavie (2005)), and CIDEr (Vedantam et al. (2015)) metrics are reported in our quantitative evaluation of the performance of the proposed schemes. In evaluation, our baseline is the widely used CNN-LSTM captioning method originally proposed in Vinyals et al. (2015). For comparison, we include results in that paper in the first line of Table 1. We also re-implemented the model using the latest ResNet feature and report the results in the second line of Table 1. Our re-implementation of the CNN-LSTM method matches the performance reported in Gan et al. (2017), showing that the baseline is a state-of-the-art implementation. As shown in Table 1, compared to the CNN-LSTM baseline, the proposed TPGN significantly outperforms the benchmark schemes in all metrics across the board. The improvement in BLEU-n is greater for greater n; TPGN particularly improves generation of longer subsequences. The results clearly attest to the effectiveness of the TPGN architecture.

6.3 EVALUATION OF THE POS TAGGER

We run the system shown in Fig. 3 with 5,000 sentences from the COCO test set as input, and obtain an unbinding vector $\mathbf{u}_t$ of each word $\mathbf{x}_t$ in the sentence produced by the TPGN system.

We design two POS taggers for classifying the POS of each word $\mathbf{x}_t$. The first POS tagger is realized by a kernel support vector machine with stochastic gradient descent, where a radial basis function kernel is used. The input of the classifier is N_w unbinding vectors corresponding to a window of N_w words, whose center is the word to be classified. For example, if $\mathbf{x}_t$ is the word to be classified, unbinding vectors corresponding to a window of words $\mathbf{x}_{t-(N_w-1)/2}, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_{t+(N_w-1)/2}$ are supplied as input to the classifier. Note that, to classify word $\mathbf{x}_1$, we need to add $\mathbf{x}_{1-(N_w-1)/2}, \cdots, \mathbf{x}_0$ to make a window of N_w words. Since words $\mathbf{x}_t$ ($t < 1$ or $t > T$) do not exist, we assign a 625-dimensional unbinding vector $\mathbf{u}_t$ (each dimension of which equals 0.5) to each of $\mathbf{x}_t$ ($t < 1$ or $t > T$). The output of the classifier is the POS of the word in the center of the window. We use the unbinding vectors and POS tags of 4,000 sentences for training, and the unbinding vectors of 1,000 sentences for testing. We use the Stanford parser in (Manning (2017)) to identify the POS of each word in the 5,000 sentences.

Table 2: Performance of POS tagger.

Window size N_w	1	3	5	7	9	11	13
Precision	0.757	0.944	0.937	**0.946**	0.934	0.929	0.919
Recall	0.763	0.929	0.936	**0.942**	0.928	0.927	0.922
F-measure	0.760	0.937	0.936	**0.944**	0.931	0.928	0.921

Table 2 shows the results for classifying the POS of the words in a sentence. It can be seen that using the unbinding vector of a word can classify the POS of the word with an accuracy of 76.3%, which means that a single unbinding vector contains important, but partial, grammatical information about the corresponding word. If the unbinding vectors of neighboring words are used, the accuracy of POS classification can be significantly increased to over 92%. The highest accuracy is achieved when the window size is 7; the F-score is **94.4**%.

The second POS tagger is realized by a bidirectional LSTM (B-LSTM) with a hidden dimension of 625. The input of the B-LSTM is a sequence of unbinding vectors of a sentence; the output of the B-LSTM is a sequence of POS tags, each of which corresponds to one word in the sentence. We use 4,000 sentences for training, and 1,000 sentences for testing. The accuracy of classification is **97.7**%, comparable to that of the state-of-the-art POS taggers (Toutanova et al. (2003)).

6.4 EVALUATION OF THE PHRASE CLASSIFIER

As for POS tagging, we run the system shown in Fig. 3 with 5,000 sentences (from the COCO test set) as input, and obtain unbinding vector $\mathbf{u}_t$ of each word $\mathbf{x}_t$ in the sentence produced by the TPGN system.

We design a phrase classifier by a B-LSTM with a hidden dimension of 625. The input of the B-LSTM is a sequence of unbinding vectors of a sentence; the output of the B-LSTM is a sequence of phrase types (e.g., noun phrase, verb phrase), each of which corresponds to one word in the sentence. We use 4,000 sentences for training, and 1,000 sentences for testing. The accuracy of classification is **84**%. Phrase classifiers are also evaluated by precision, recall, and the F-measure (Jurafsky & Martin (2017)). Precision measures the percentage of system-provided phrases that were correct. Correct here means that both the boundaries of the phrase and the phrase's type are correct. The precision, recall, and F-measure of our phrase classifier are **82.5**%, **79.4**%, **80.9**%, respectively, which are comparable to the state-of-the-art phrase classifier in Zhu et al. (2013).

Combining the results in Sections 6.3 and 6.4, we are able to create an incomplete four-level parse tree shown in Fig. 4. In our future work, we will design a system to create a complete parse tree of a sentence, given the unbinding vectors of the sentence.

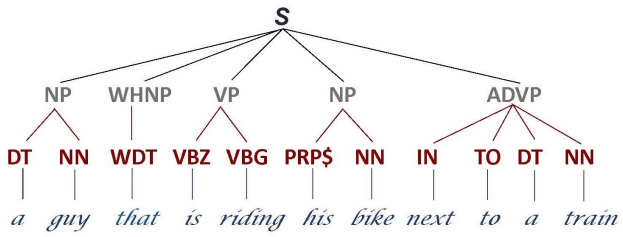

Figure 4: A generated parse tree for a sentence where NP, WHNP, VP, ADVP, DT, NN, WDT, VBZ, VBG, PRP$, IN, and TO denote noun phrase, Wh-noun phrase, verb phrase, adverb phrase, determiner, noun, Wh-determiner, Verb (3rd person singular present), verb (gerund or present participle), possessive pronoun, preposition, and "to", respectively.

7 Conclusion

In this paper, we proposed a new Tensor Product Generation Network (TPGN) for natural language generation and related tasks. The model has a novel architecture based on a rationale derived from the use of Tensor Product Representations for encoding and processing symbolic structure through neural network computation. In evaluation, we tested the proposed model on captioning with the MS COCO dataset, a large-scale image captioning benchmark. Compared to widely adopted LSTM-based models, the proposed TPGN gives significant improvements on all major metrics including METEOR, BLEU, and CIDEr. Moreover, we observe that the unbinding vectors contain important grammatical information, which allows us to design effective POS tagger and phrase detector/classifier with unbinding vectors as input. Our findings in this paper show great promise of TPRs. In the future, we will explore extending TPR to a variety of other NLP tasks.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72. Association for Computational Linguistics, 2005.

Xinlei Chen and C Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2422–2431, 2015.

COCO. Coco dataset for image captioning. `http://mscoco.org/dataset/#download`, 2017.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809*, 2015.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1473–1482, 2015.

Zhe Gan, Chuang Gan, Xiaodong He, Yunchen Pu, Kenneth Tran, Jianfeng Gao, Lawrence Carin, and Li Deng. Semantic compositional networks for visual captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Daniel Jurafsky and James H Martin. *Speech and Language Processing*. 3rd ed. draft edition edition, 2017.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137, 2015.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 595–603, 2014a.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014b.

Christopher Manning. Stanford parser. `https://nlp.stanford.edu/software/lex-parser.shtml`, 2017.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). In *Proceedings of International Conference on Learning Representations*, 2015.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Stanford glove: Global vectors for word representation. `https://nlp.stanford.edu/projects/glove/`, 2017.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

Paul Smolensky. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society — A: Mathematical, Physical and Engineering Sciences*, 370:3543 – 3569, 2012.

Paul Smolensky and Géraldine Legendre. *The harmonic mind: From neural computation to optimality-theoretic grammar. Volume 1: Cognitive architecture*. MIT Press, 2006.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 173–180. Association for Computational Linguistics, 2003.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4566–4575, 2015.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2015.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 434–443, 2013.