Grounded Compositional Semantics for Finding and Describing Images with Sentences

Richard Socher, Andrej Karpathy, Quoc V. Le*, Christopher D. Manning, Andrew Y. Ng
Stanford University, Computer Science Department, *Google Inc.
richard@socher.org, karpathy@cs.stanford.edu,
qvl@google.com, manning@stanford.edu, ang@cs.stanford.edu

Abstract

Previous work on Recursive Neural Networks (RNNs) shows that these models can produce compositional feature vectors for accurately representing and classifying sentences or images. However, the sentence vectors of previous models cannot accurately represent visually grounded meaning. We introduce the DT-RNN model which uses dependency trees to embed sentences into a vector space in order to retrieve images that are described by those sentences. Unlike previous RNN-based models which use constituency trees, DT-RNNs naturally focus on the action and agents in a sentence. They are better able to abstract from the details of word order and syntactic expression. DT-RNNs outperform other recursive and recurrent neural networks, kernelized CCA and a bag-of-words baseline on the tasks of finding an image that fits a sentence description and vice versa. They also give more similar representations to sentences that describe the same image.

1 Introduction

Single word vector spaces are widely used (Turney and Pantel, 2010) and successful at classifying single words and capturing their meaning (Collobert and Weston, 2008; Huang et al., 2012; Mikolov et al., 2013). Since words rarely appear in isolation, the task of learning compositional meaning representations for longer phrases has recently received a lot of attention (Mitchell and Lapata, 2010; Socher et al., 2010; Socher et al., 2012; Grefenstette et al., 2013). Similarly, classifying whole images into a fixed set of classes also achieves very high performance (Le et al., 2012; Krizhevsky et al., 2012). However, similar to words, objects in images are often seen in relationships with other objects which are not adequately described by a single label.

In this work, we introduce a model, illustrated in Fig. 1, which learns to map sentences and images into a common embedding space in order to be able to retrieve one from the other. We assume word and image representations are first learned in their respective single modalities but finally mapped into a jointly learned multimodal embedding space.

Our model for mapping sentences into this space is based on ideas from Recursive Neural Networks (RNNs) (Pollack, 1990; Costa et al., 2003; Socher et al., 2011b). However, unlike all previous RNN models which are based on constituency trees (CT-RNNs), our model computes compositional vector representations inside dependency trees. The compositional vectors computed by this new dependency tree RNN (DT-RNN) capture more of the meaning of sentences, where we define meaning in terms of similarity to a "visual representation" of the textual description. DT-RNN induced vector representations of sentences are more robust to changes in the syntactic structure or word order than related models such as CT-RNNs or Recurrent Neural Networks since they naturally focus on a sentence's action and its agents.

We evaluate and compare DT-RNN induced representations on their ability to use a sentence such as *"A man wearing a helmet jumps on his bike near a beach."* to find images that show such a scene. The goal is to learn sentence representations that capture

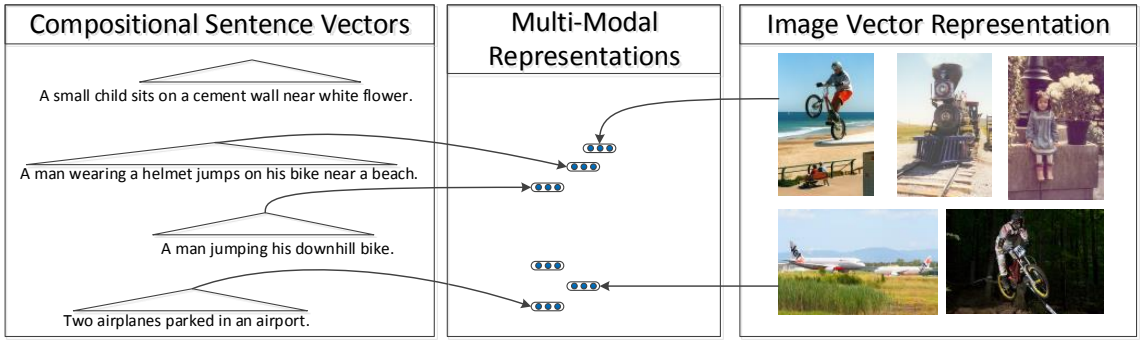

Figure 1: The DT-RNN learns vector representations for sentences based on their dependency trees. We learn to map the outputs of convolutional neural networks applied to images into the same space and can then compare both sentences and images. This allows us to query images with a sentence and give sentence descriptions to images.

the visual scene described and to find appropriate images in the learned, multi-modal sentence-image space. Conversely, when given a query image, we would like to find a description that goes beyond a single label by providing a correct sentence describing it, a task that has recently garnered a lot of attention (Farhadi et al., 2010; Ordonez et al., 2011; Kuznetsova et al., 2012). We use the dataset introduced by (Rashtchian et al., 2010) which consists of 1000 images, each with 5 descriptions. On all tasks, our model outperforms baselines and related models.

2 Related Work

The presented model is connected to several areas of NLP and vision research, each with a large amount of related work to which we can only do some justice given space constraints.

Semantic Vector Spaces and Their Compositionality. The dominant approach in semantic vector spaces uses distributional similarities of single words. Often, co-occurrence statistics of a word and its context are used to describe each word (Turney and Pantel, 2010; Baroni and Lenci, 2010), such as tf-idf. Most of the compositionality algorithms and related datasets capture two-word compositions. For instance, (Mitchell and Lapata, 2010) use two-word phrases and analyze similarities computed by vector addition, multiplication and others. Compositionality is an active field of research with many different models and representations being explored (Grefenstette et al., 2013), among many others. We compare to supervised compositional models that can learn task-specific vector representations such as constituency tree recursive neural networks (Socher et al., 2011b; Socher et al., 2011a), chain structured recurrent neural networks and other baselines. Another alternative would be to use CCG trees as a backbone for vector composition (K.M. Hermann, 2013).

Multimodal Embeddings. Multimodal embedding methods project data from multiple sources such as sound and video (Ngiam et al., 2011) or images and text. Socher et al. (Socher and Fei-Fei, 2010) project words and image regions into a common space using kernelized canonical correlation analysis to obtain state of the art performance in annotation and segmentation. Similar to our work, they use unsupervised large text corpora to learn semantic word representations. Among other recent work is that by Srivastava and Salakhutdinov (2012) who developed multimodal Deep Boltzmann Machines. Similar to their work, we use techniques from the broad field of deep learning to represent images and words.

Recently, single word vector embeddings have been used for zero shot learning (Socher et al., 2013c). Mapping images to word vectors enabled their system to classify images as depicting objects such as ”cat” without seeing any examples of this class. Related work has also been presented at NIPS (Socher et al., 2013b; Frome et al., 2013). This work moves zero-shot learning beyond single categories per image and extends it to unseen phrases and full length sentences, making use of similar ideas of semantic spaces grounded in visual knowledge.

Detailed Image Annotation. Interactions between images and texts is a growing research field. Early work in this area includes generating single words or fixed phrases from images (Duygulu et al., 2002; Barnard et al., 2003) or using contextual information to improve recognition (Gupta and Davis, 2008; Torralba et al., 2010).

Apart from a large body of work on single object image classification (Le et al., 2012), there is also work on attribute classification and other mid-level elements (Kumar et al., 2009), some of which we hope to capture with our approach as well.

Our work is close in spirit with recent work in describing images with more detailed, longer textual descriptions. In particular, Yao et al. (2010) describe images using hierarchical knowledge and humans in the loop. In contrast, our work does not require human interactions. Farhadi et al. (2010) and Kulkarni et al. (2011), on the other hand, use a more automatic method to parse images. For instance, the former approach uses a single triple of objects estimated for an image to retrieve sentences from a collection written to describe similar images. It forms representations to describe 1 object, 1 action, and 1 scene. Kulkarni et al. (2011) extends their method to describe an image with multiple objects. None of these approaches have used a compositional sentence vector representation and they require specific language generation techniques and sophisticated inference methods. Since our model is based on neural networks inference is fast and simple. Kuznetsova et al. (2012) use a very large parallel corpus to connect images and sentences. Feng and Lapata (2013) use a large dataset of captioned images and experiments with both extractive (search) and abstractive (generation) models.

Most related is the very recent work of Hodosh et al. (2013). They too evaluate using a ranking measure. In our experiments, we compare to kernelized Canonical Correlation Analysis which is the main technique in their experiments.

3 Dependency-Tree Recursive Neural Networks

In this section we first focus on the DT-RNN model that computes compositional vector representations for phrases and sentences of variable length and syntactic type. In section 5 the resulting vectors will then become multimodal features by mapping images that show what the sentence describes to the same space and learning both the image and sentence mapping jointly.

The most common way of building representations for longer phrases from single word vectors is to simply linearly average the word vectors. While this bag-of-words approach can yield reasonable performance in some tasks, it gives all the words the same weight and cannot distinguish important differences in simple visual descriptions such as *The bike crashed into the standing car.* vs. *The car crashed into the standing bike.*.

RNN models (Pollack, 1990; Goller and Küchler, 1996; Socher et al., 2011b; Socher et al., 2011a) provided a novel way of combining word vectors for longer phrases that moved beyond simple averaging. They combine vectors with an RNN in binary constituency trees which have potentially many hidden layers. While the induced vector representations work very well on many tasks, they also inevitably capture a lot of syntactic structure of the sentence. However, the task of finding images from sentence descriptions requires us to be more invariant to syntactic differences. One such example are active-passive constructions which can collapse words such as "by" in some formalisms (de Marneffe et al., 2006), relying instead on the semantic relationship of "agent". For instance, *The mother hugged her child.* and *The child was hugged by its mother.* should map to roughly the same visual space. Current Recursive and Recurrent Neural Networks do not exhibit this behavior and even bag of words representations would be influenced by the words *was* and *by*. The model we describe below focuses more on recognizing actions and agents and has the potential to learn representations that are invariant to active-passive differences.

3.1 DT-RNN Inputs: Word Vectors and Dependency Trees

In order for the DT-RNN to compute a vector representation for an ordered list of m words (a phrase or sentence), we map the single words to a vector space and then parse the sentence.

First, we map each word to a d-dimensional vector. We initialize these word vectors with the un-

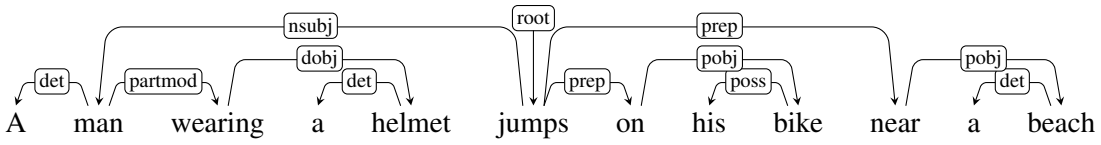

Figure 2: Example of a full dependency tree for a longer sentence. The DT-RNN will compute vector representations at every word that represents that word and an arbitrary number of child nodes. The final representation is computed at the *root* node, here at the verb *jumps*. Note that more important activity and object words are higher up in this tree structure.

supervised model of Huang et al. (2012) which can learn single word vector representations from both local and global contexts. The idea is to construct a neural network that outputs high scores for windows and documents that occur in a large unlabeled corpus and low scores for window-document pairs where one word is replaced by a random word. When such a network is optimized via gradient descent the derivatives backpropagate into a word embedding matrix A which stores word vectors as columns. In order to predict correct scores the vectors in the matrix capture co-occurrence statistics. We use $d = 50$ in all our experiments. The embedding matrix X is then used by finding the column index i of each word: $[w] = i$ and retrieving the corresponding column x_w from X. Henceforth, we represent an input sentence s as an ordered list of (word,vector) pairs: $s = ((w_1, x_{w_1}), \ldots, (w_m, x_{w_m}))$.

Next, the sequence of words $(w_1, \ldots, w_m)$ is parsed by the dependency parser of de Marneffe et al. (2006). Fig. 2 shows an example. We can represent a dependency tree d of a sentence s as an ordered list of (child,parent) indices: $d(s) = \{(i, j)\}$, where every child word in the sequence $i = 1, \ldots, m$ is present and has any word $j \in \{1, \ldots, m\} \cup \{0\}$ as its parent. The root word has as its parent 0 and we notice that the same word can be a parent between zero and m number of times. Without loss of generality, we assume that these indices form a tree structure. To summarize, the input to the DT-RNN for each sentence is the pair (s, d): the words and their vectors and the dependency tree.

3.2 Forward Propagation in DT-RNNs

Given these two inputs, we now illustrate how the DT-RNN computes parent vectors. We will use the following sentence as a running example: *Students*$_1$ *ride*$_2$ *bikes*$_3$ *at*$_4$ *night*$_5$. Fig. 3 shows its tree and computed vector representations. The dependency tree for this sentence can be summarized by the following set of (child, parent) edges: $d = \{(1, 2), (2, 0), (3, 2), (4, 2), (5, 4)\}$.

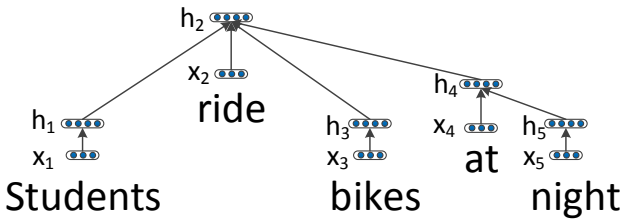

Figure 3: Example of a DT-RNN tree structure for computing a sentence representation in a bottom up fashion.

The DT-RNN model will compute parent vectors at each word that include all the dependent (children) nodes in a bottom up fashion using a compositionality function g_θ which is parameterized by all the model parameters θ. To this end, the algorithm searches for nodes in a tree that have either (i) no children or (ii) whose children have already been computed and then computes the corresponding vector.

In our example, the words x_1, x_3, x_5 are leaf nodes and hence, we can compute their corresponding hidden nodes via:

$$h_c = g_\theta(x_c) = f(W_v x_c) \quad \text{for } c = 1, 3, 5, \quad (1)$$

where we compute the hidden vector at position c via our general composition function g_θ. In the case of leaf nodes, this composition function becomes simply a linear layer, parameterized by $W_v \in \mathbb{R}^{n \times d}$, followed by a nonlinearity. We cross-validate over using no nonlinearity ($f = \mathrm{id}$), $\tanh$, $\mathrm{sigmoid}$ or rectified linear units ($f = \max(0, x)$), but generally find $\tanh$ to perform best.

The final sentence representation we want to compute is at h_2, however, since we still do not have h_4,

we compute that one next:

$$h_4 = g_\theta(x_4, h_5) = f(W_v x_4 + W_{r1} h_5), \quad (2)$$

where we use the same W_v as before to map the word vector into hidden space but we now also have a linear layer that takes as input h_5, the only child of the fourth node. The matrix $W_{r1} \in \mathbb{R}^{n \times n}$ is used because node 5 is the first child node on the right side of node 4. Generally, we have multiple matrices for composing with hidden child vectors from the right and left sides: $W_{r\cdot} = (W_{r1}, \ldots, W_{rk_r})$ and $W_{l\cdot} = (W_{l1}, \ldots, W_{lk_l})$. The number of needed matrices is determined by the data by simply finding the maximum numbers of left k_l and right k_r children any node has. If at test time a child appeared at an even large distance (this does not happen in our test set), the corresponding matrix would be the identity matrix.

Now that all children of h_2 have their hidden vectors, we can compute the final sentence representation via:

$$h_2 = g_\theta(x_2, h_1, h_3, h_4) = \quad (3)$$
$$f(W_v x_2 + W_{l1} h_1 + W_{r1} h_3 + W_{r2} h_4).$$

Notice that the children are multiplied by matrices that depend on their location relative to the current node.

Another modification that improves the mean rank by approximately 6 in image search on the dev set is to weight nodes by the number of words underneath them and normalize by the sum of words under all children. This encourages the intuitive desideratum that nodes describing longer phrases are more important. Let $\ell(i)$ be the number of leaf nodes (words) under node i and $C(i, y)$ be the set of child nodes of node i in dependency tree y. The final composition function for a node vector h_i becomes:

$$h_i = f\left(\frac{1}{\ell(i)}\left(W_v x_i + \sum_{j \in C(i)} \ell(j) W_{\text{pos}(i,j)} h_j\right)\right), \quad (4)$$

where by definition $\ell(i) = 1 + \sum_{j \in C(i)} \ell(j)$ and $\text{pos}(i, j)$ is the relative position of child j with respect to node i, e.g. $l1$ or $r2$ in Eq. 3.

3.3 Semantic Dependency Tree RNNs

An alternative is to condition the weight matrices on the semantic relations given by the dependency parser. We use the collapsed tree formalism of the Stanford dependency parser (de Marneffe et al., 2006). With such a semantic untying of the weights, the DT-RNN makes better use of the dependency formalism and could give active-passive reversals similar semantic vector representation. The equation for this semantic DT-RNN (**SDT-RNN**) is the same as the one above except that the matrices $W_{\text{pos}(i,j)}$ are replaced with matrices based on the dependency relationship. There are a total of 141 unique such relationships in the dataset. However, most are very rare. For examples of semantic relationships, see Fig. 2 and the model analysis section 6.7.

This forward propagation can be used for computing compositional vectors and in Sec. 5 we will explain the objective function in which these are trained.

3.4 Comparison to Previous RNN Models

The DT-RNN has several important differences to previous RNN models of Socher et al. (2011a) and (Socher et al., 2011b; Socher et al., 2011c). These constituency tree RNNs (CT-RNNs) use the following composition function to compute a hidden parent vector h from exactly two child vectors (c_1, c_2) in a binary tree: $h = f\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}\right)$, where $W \in \mathbb{R}^{d \times 2d}$ is the main parameter to learn. This can be rewritten to show the similarity to the DT-RNN as $h = f(W_{l1} c_1 + W_{r1} c_2)$. However, there are several important differences.

Note first that in previous RNN models the parent vectors were of the same dimensionality to be recursively compatible and be used as input to the next composition. In contrast, our new model first maps single words into a hidden space and then parent nodes are composed from these hidden vectors. This allows a higher capacity representation which is especially helpful for nodes that have many children.

Secondly, the DT-RNN allows for n-ary nodes in the tree. This is an improvement that is possible even for constituency tree CT-RNNs but it has not been explored in previous models.

Third, due to computing parent nodes in constituency trees, previous models had the problem that words that are merged last in the tree have a larger weight or importance in the final sentence rep-

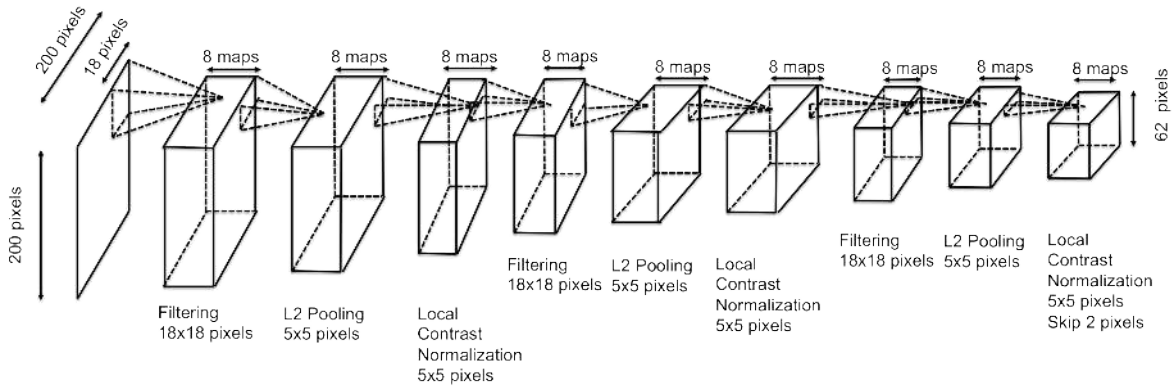

Figure 4: The architecture of the visual model. This model has 3 sequences of filtering, pooling and local contrast normalization layers. The learnable parameters are the filtering layer. The filters are not shared, i.e., the network is nonconvolutional.

resentation. This can be problematic since these are often simple non-content words, such as a leading 'But,'. While such single words can be important for tasks such as sentiment analysis, we argue that for describing visual scenes the DT-RNN captures the more important effects: The dependency tree structures push the central content words such as the main action or verb and its subject and object to be merged last and hence, by construction, the final sentence representation is more robust to less important adjectival modifiers, word order changes, etc.

Fourth, we allow some untying of weights depending on either how far away a constituent is from the current word or what its semantic relationship is.

Now that we can compute compositional vector representations for sentences, the next section describes how we represent images.

4 Learning Image Representations with Neural Networks

The image features that we use in our experiments are extracted from a deep neural network, replicated from the one described in (Le et al., 2012). The network was trained using both unlabeled data (random web images) and labeled data to classify 22,000 categories in ImageNet (Deng et al., 2009). We then used the features at the last layer, before the classifier, as the feature representation in our experiments. The dimension of the feature vector of the last layer is 4,096. The details of the model and its training procedures are as follows.

The architecture of the network can be seen in Figure 4. The network takes 200x200 pixel images as inputs and has 9 layers. The layers consist of three sequences of filtering, pooling and local contrast normalization (Jarrett et al., 2009). The pooling function is L2 pooling of the previous layer (taking the square of the filtering units, summing them up in a small area in the image, and taking the square-root). The local contrast normalization takes inputs in a small area of the lower layer, subtracts the mean and divides by the standard deviation.

The network was first trained using an unsupervised objective: trying to reconstruct the input while keeping the neurons sparse. In this phase, the network was trained on 20 million images randomly sampled from the web. We resized a given image so that its short dimension has 200 pixels. We then cropped a fixed size 200x200 pixel image right at the center of the resized image. This means we may discard a fraction of the long dimension of the image.

After unsupervised training, we used ImageNet (Deng et al., 2009) to adjust the features in the entire network. The ImageNet dataset has 22,000 categories and 14 million images. The number of images in each category is equal across categories. The 22,000 categories are extracted from WordNet.

To speed up the supervised training of this network, we made a simple modification to the algorithm described in Le et al. (2012): adding a "bottleneck" layer in between the last layer and the classifier. to reduce the number of connections. We added one "bottleneck" layer which has 4,096 units in between the last layer of the network and the softmax layer. This newly-added layer is fully connected to the previous layer and has a linear activation function. The total number of connections of this network is approximately 1.36 billion.

The network was trained again using the supervised objective of classifying the 22,000 classes in ImageNet. Most features in the networks are local, which allows model parallelism. Data parallelism by asynchronous SGD was also employed as in Le et al. (2012). The entire training, both unsupervised and supervised, took 8 days on a large cluster of machines. This network achieves 18.3% precision@1 on the full ImageNet dataset (Release Fall 2011).

We will use the features at the bottleneck layer as the feature vector z of an image. Each scaled and cropped image is presented to our network. The network then performs a feedforward computation to compute the values of the bottleneck layer. This means that every image is represented by a fixed length vector of 4,096 dimensions. Note that during training, no aligned sentence-image data was used and the ImageNet classes do not fully intersect with the words used in our dataset.

5 Multimodal Mappings

The previous two sections described how we can map sentences into a $d = 50$-dimensional space and how to extract high quality image feature vectors of 4096 dimensions. We now define our final multimodal objective function for learning joint image-sentence representations with these models. Our training set consists of N images and their feature vectors z_i and each image has 5 sentence descriptions $s_{i1}, \ldots, s_{i5}$ for which we use the DT-RNN to compute vector representations. See Fig. 5 for examples from the dataset. For training, we use a max-margin objective function which intuitively trains pairs of correct image and sentence vectors to have high inner products and incorrect pairs to have low inner products. Let $v_i = W_I z_i$ be the mapped image vector and $y_{ij} = DTRNN_\theta(s_{ij})$ the composed sentence vector. We define $\mathcal{S}$ to be the set of all sentence indices and $\mathcal{S}(i)$ the set of sentence indices corresponding to image i. Similarly, $\mathcal{I}$ is the set of all image indices and $\mathcal{I}(j)$ is the image index of sentence j. The set $\mathcal{P}$ is the set of all correct image-sentence training pairs (i, j). The ranking cost function to minimize is then: $J(W_I, \theta) =$

$$\sum_{(i,j)\in\mathcal{P}} \sum_{c\in\mathcal{S}\setminus\mathcal{S}(i)} \max(0, \Delta - v_i^T y_j + v_i^T y_c) + \sum_{(i,j)\in\mathcal{P}} \sum_{c\in\mathcal{I}\setminus\mathcal{I}(j)} \max(0, \Delta - v_i^T y_j + v_c^T y_j), \quad (5)$$

where θ are the language composition matrices, and both second sums are over other sentences coming from different images and vice versa. The hyperparameter Δ is the margin. The margin is found via cross validation on the dev set and usually around 1.

The final objective also includes the regularization term $\lambda / left(\|\theta\|_2^2 + \|W_I\|_F)$. Both the visual model and the word vector learning require a very large amount of training data and both have a huge number of parameters. Hence, to prevent overfitting, we assume their weights are fixed and only train the DT-RNN parameters W_I. If larger training corpora become available in the future, training both jointly becomes feasible and would present a very promising direction. We use a modified version of AdaGrad (Duchi et al., 2011) for optimization of both W_I and the DT-RNN as well as the other baselines (except kCCA). Adagrad has achieved good performance previously in neural networks models (Dean et al., 2012; Socher et al., 2013a). We modify it by resetting all squared gradient sums to 1 every 5 epochs. With both images and sentences in the same multimodal space, we can easily query the model for similar images or sentences by finding the nearest neighbors in terms of negative inner products.

An alternative objective function is based on the squared loss $J(W_I, \theta) = \sum_{(i,j)\in\mathcal{P}} \|v_i - y_j\|_2^2$. This requires an alternating minimization scheme that first trains only W_I, then fixes W_I and trains the DT-RNN weights θ and then repeats this several times. We find that the performance with this objective function (paired with finding similar images using Euclidean distances) is worse for all models than the margin loss of Eq. 5. In addition kCCA also performs much better using inner products in the multimodal space.

6 Experiments

We use the dataset of Rashtchian et al. (2010) which consists of 1000 images, each with 5 sentences. See Fig. 5 for examples.

We evaluate and compare the DT-RNN in three different experiments. First, we analyze how well the sentence vectors capture similarity in visual meaning. Then we analyze *Image Search with Query Sentences*: to query each model with a sentence in order to find an image showing that sen-

1. A woman and her dog watch the cameraman in their living with wooden floors.
2. A woman sitting on the couch while a black faced dog runs across the floor.
3. A woman wearing a backpack sits on a couch while a small dog runs on the hardwood floor next to her.
4. A women sitting on a sofa while a small Jack Russell walks towards the camera.
5. White and black small dog walks toward the camera while woman sits on couch, desk and computer seen in the background as well as a pillow, teddy bear and moggie toy on the wood floor.

1. A man in a cowboy hat check approaches a small red sports car.
2. The back and left side of a red Ferrari and two men admiring it.
3. The sporty car is admired by passer by.
4. Two men next to a red sports car in a parking lot.
5. Two men stand beside a red sports car.

Figure 5: Examples from the dataset of images and their sentence descriptions (Rashtchian et al., 2010). Sentence length varies greatly and different objects can be mentioned first. Hence, models have to be invariant to word ordering.

tence's visual 'meaning.' The last experiment *Describing Images by Finding Suitable Sentences* does the reverse search where we query the model with an image and try to find the closest textual description in the embedding space.

In our comparison to other methods we focus on those models that can also compute fixed, continuous vectors for sentences. In particular, we compare to the RNN model on constituency trees of Socher et al. (2011a), a standard recurrent neural network; a simple bag-of-words baseline which averages the words. All models use the word vectors provided by Huang et al. (2012) and do not update them as discussed above. Models are trained with their corresponding gradients and backpropagation techniques. A standard recurrent model is used where the hidden vector at word index t is computed from the hidden vector at the previous time step and the current word vector: $h_t = f(W_h h_{t-1} + W_x x_t)$. During training, we take the last hidden vector of the sentence chain and propagate the error into that. It is also this vector that is used to represent the sentence.

Other possible comparisons are to the very different models mentioned in the related work section. These models use a lot more task-specific engineering, such as running object detectors with bounding boxes, attribute classifiers, scene classifiers, CRFs for composing the sentences, etc. Another line of work uses large sentence-image aligned resources (Kuznetsova et al., 2012), whereas we focus on easily obtainable training data of each modality separately and a rather small multimodal corpus.

In our experiments we split the data into 800 training, 100 development and 100 test images. Since there are 5 sentences describing each image, we have 4000 training sentences and 500 testing sentences. The dataset has 3020 unique words, half of which only appear once. Hence, the unsupervised, pre-trained semantic word vector representations are crucial. Word vectors are not fine tuned during training. Hence, the main parameters are the DT-RNN's $W_{l\cdot}, W_{r\cdot}$ or the semantic matrices of which there are 141 and the image mapping W_I. For both DT-RNNs the weight matrices are initialized to block identity matrices plus Gaussian noise. Word vectors and hidden vectors are set o length 50. Using the development split, we found $\lambda = 0.08$ and the learning rate of AdaGrad to 0.0001. The best model uses a margin of $\Delta = 3$.

Inspired by Socher and Fei-Fei (2010) and Hodosh et al. (2013) we also compare to kernelized Canonical Correlation Analysis (kCCA). We use the average of word vectors for describing sentences and the same powerful image vectors as before. We use the code of Socher and Fei-Fei (2010). Technically, one could combine the recently introduced deep CCA Andrew et al. (2013) and train the recursive neural network architectures with the CCA objective. We leave this to future work. With linear kernels, kCCA does well for image search but is worse for sentence self similarity and describing images with sentences close-by in embedding space. All other models are trained by replacing the DT-RNN function in Eq. 5.

6.1 Similarity of Sentences Describing the Same Image

In this experiment, we first map all 500 sentences from the test set into the multi-modal space. Then for each sentence, we find the nearest neighbor sen-

Sentences Similarity for Image		*Image Search*		*Describing Images*	
Model	Mean Rank	Model	Mean Rank	Model	Mean Rank
Random	101.1	Random	52.1	Random	92.1
BoW	11.8	BoW	14.6	BoW	21.1
CT-RNN	15.8	CT-RNN	16.1	CT-RNN	23.9
Recurrent NN	18.5	Recurrent NN	19.2	Recurrent NN	27.1
kCCA	10.7	kCCA	15.9	kCCA	18.0
DT-RNN	11.1	DT-RNN	13.6	DT-RNN	19.2
SDT-RNN	**10.5**	SDT-RNN	**12.5**	SDT-RNN	**16.9**

Table 1: **Left:** Comparison of methods for sentence similarity judgments. Lower numbers are better since they indicate that sentences describing the same image rank more highly (are closer). The ranks are out of the 500 sentences in the test set. **Center:** Comparison of methods for image search with query sentences. Shown is the average rank of the single correct image that is being described. **Right:** Average rank of a correct sentence description for a query image.

tences in terms of inner products. We then sort these neighbors and record the rank or position of the nearest sentence *that describes the same image*. If all the images were very unique and the visual descriptions close-paraphrases and consistent, we would expect a very low rank. However, usually a handful of images are quite similar (for instance, there are various images of airplanes flying, parking, taxiing or waiting on the runway) and sentence descriptions can vary greatly in detail and specificity for the same image.

Table 1 (left) shows the results. We can see that averaging the high quality word vectors already captures a lot of similarity. The chain structure of a standard recurrent neural net performs worst since its representation is dominated by the last words in the sequence which may not be as important as earlier words.

6.2 Image Search with Query Sentences

This experiment evaluates how well we can find images that display the visual meaning of a given sentence. We first map a query sentence into the vector space and then find images in the same space using simple inner products. As shown in Table 1 (center), the new DT-RNN outperforms all other models.

6.3 Describing Images by Finding Suitable Sentences

Lastly, we repeat the above experiments but with roles reversed. For an image, we search for suitable textual descriptions again simply by finding close-by sentence vectors in the multi-modal embedding space. Table 1 (right) shows that the DT-RNN again outperforms related models. Fig. 2assigned to images. The average ranking of 25.3 for a correct sentence description is out of 500 possible sentences. A random assignment would give an average ranking of 100.

Image Search		*Describing Images*	
Model	mRank	Model	mRank
BoW	24.7	BoW	30.7
CT-RNN	22.2	CT-RNN	29.4
Recurrent NN	28.4	Recurrent NN	31.4
kCCA	13.7	kCCA	38.0
DT-RNN	13.3	DT-RNN	26.8
SDT-RNN	15.8	SDT-RNN	37.5

Table 2: Results of multimodal ranking when models are trained with a squared error loss and using Euclidean distance in the multimodal space. Better performance is reached for all models when trained in a max-margin loss and using inner products as in the previous table.

6.4 Analysis: Squared Error Loss vs. Margin Loss

We analyze the influence of the multimodal loss function on the performance. In addition, we compare using Euclidean distances instead of inner products. Table 2 shows that performance is worse for all models in this setting.

6.5 Analysis: Recall at n vs Mean Rank

Hodosh et al. (2013) and other related work use recall at n as an evaluation measure. Recall at n captures how often one of the top n closest vectors were a correct image or sentence and gives a good intuition of how a model would perform in a ranking task that presents n such results to a user. Below, we compare three commonly used and high performing models: bag of words, kCCA and our SDT-RNN on

A gray convertible sports car is parked in front of the trees.
A close-up view of the headlights of a blue old-fashioned car.
Black shiny sports car parked on concrete driveway.
Five cows grazing on a patch of grass between two roadways.

A green steam train running down the tracks.
Steamy locomotive speeding thou the forest.
A steam engine comes down a train track near trees.
A double decker bus is driving by Big Ben in London.

A jockey rides a brown and white horse in a dirt corral.
A young woman is riding a Bay hose in a dirt riding-ring.
A white bird pushes a miniature teal shopping cart.
A person rides a brown horse.

People in an outrigger canoe sail on emerald green water
Two people sailing a small white sail boat.
behind a cliff, a boat sails away
Tourist move in on Big Ben on a typical overcast London day.

A motocross bike with rider flying through the air.
White propeller plane parked in middle of grassy field.
The white jet with its landing gear down flies in the blue sky.
An elderly woman catches a ride on the back of the bicycle.

A group of people sitting around a table on a porch.
A group of four people walking past a giant mushroom.
A man and women smiling for the camera in a kitchen.
A group of men sitting around a table drinking while a man behind stands pointing.

Figure 6: Images and their sentence descriptions assigned by the DT-RNN.

Model	mRank △	R@1 ▽	R@5 ▽	R@10 ▽
Image Search				
BoW	14.6	15.8	42.2	60.0
kCCA	15.9	**16.4**	41.4	58.0
SDT-RNN	**12.5**	**16.4**	**46.6**	**65.6**
Describing Images				
BoW	21.1	19.0	38.0	57.0
kCCA	18.0	21.0	**47.0**	61.0
SDT-RNN	**16.9**	**23.0**	45.0	**63.0**

Table 3: Evaluation comparison between mean rank of the closest correct image or sentence (lower is better △) with recall at different thresholds (higher is better, ▽). With one exception (R@5, bottom table), the SDT-RNN outperforms the other two models and all other models we did not include here.

this different metric. Table 3 shows that the measures do correlate well and the SDT-RNN also performs best on the multimodal ranking tasks when evaluated with this measure.

6.6 Error Analysis

In order to understand the main problems with the composed sentence vectors, we analyze the sentences that have the worst nearest neighbor rank between each other. We find that the main failure mode of the SDT-RNN occurs when a sentence that should describe the same image does not use a verb but the other sentences of that image do include a verb. For example, the following sentence pair has vectors that are very far apart from each other even though they are supposed to describe the same image:

1. A blue and yellow airplane flying straight down while emitting white smoke

2. Airplane in dive position

Generally, as long as both sentences either have a verb or do not, the SDT-RNN is more robust to different sentence lengths than bag of words representations.

6.7 Model Analysis: Semantic Composition Matrices

The best model uses composition matrices based on semantic relationships from the dependency parser. We give some insights into what the model learns by listing the composition matrices with the largest Frobenius norms. Intuitively, these matrices have learned larger weights that are being multiplied with the child vector in the tree and hence that child will have more weight in the final composed parent vector. In decreasing order of Frobenius norm, the relationship matrices are: nominal subject, possession modifier (e.g. their), passive auxiliary, preposition at, preposition in front of, passive auxiliary, passive nominal subject, object of preposition, preposition in and preposition on.

The model learns that nouns are very important as well as their spatial prepositions and adjectives.

7 Conclusion

We introduced a new recursive neural network model that is based on dependency trees. For evaluation, we use the challenging task of mapping sentences and images into a common space for finding one from the other. Our new model outperforms baselines and other commonly used models that can compute continuous vector representations for sentences. In comparison to related models, the DT-RNN is more invariant and robust to surface changes such as word order.

References

G. Andrew, R. Arora, K. Livescu, and J. Bilmes. 2013. Deep canonical correlation analysis. In *ICML*, Atlanta, Georgia.

K. Barnard, P. Duygulu, N. de Freitas, D. Forsyth, D. Blei, and M. Jordan. 2003. Matching words and pictures. *JMLR*.

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

F. Costa, P. Frasconi, V. Lombardo, and G. Soda. 2003. Towards incremental parsing of natural language using recursive neural networks. *Applied Intelligence*.

M. de Marneffe, B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A.Y. Ng. 2012. Large scale distributed deep networks. In *NIPS*.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12, July.

P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. 2002. Object recognition as machine translation. In *ECCV*.

A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *ECCV*.

Y. Feng and M. Lapata. 2013. Automatic caption generation for news images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35.

A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*.

C. Goller and A. Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks*.

E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *IWCS*.

A. Gupta and L. S. Davis. 2008. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *ECCV*.

M. Hodosh, P. Young, and J. Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Intell. Res. (JAIR)*, 47:853–899.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.

K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. 2009. What is the best multi-stage architecture for object recognition? In *ICCV*.

P. Blunsom. K.M. Hermann. 2013. The role of syntax in vector space models of compositional semantics. In *ACL*.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. 2011. Baby talk: Understanding and generating image descriptions. In *CVPR*.

N. Kumar, A. C. Berg, P. N. Belhumeur, , and S. K. Nayar. 2009. Attribute and simile classifiers for face verification. In *ICCV*.

P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *ACL*.

Q. V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A. Y. Ng. 2012. Building high-level features using large scale unsupervised learning. In *ICML*.

T. Mikolov, W. Yih, and G. Zweig. 2013. Linguistic regularities in continuous spaceword representations. In *HLT-NAACL*.

J. Mitchell and M. Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A.Y. Ng. 2011. Multimodal deep learning. In *ICML*.

V. Ordonez, G. Kulkarni, and T. L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*.

J. B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46, November.

C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. 2010. Collecting image annotations using Amazon's Mechanical Turk. In *Workshop on Creating Speech and Language Data with Amazon's MTurk*.

R. Socher and L. Fei-Fei. 2010. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*.

R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *NIPS*.

R. Socher, C. Lin, A. Y. Ng, and C.D. Manning. 2011b. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011c. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *EMNLP*.

R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. 2013a. Parsing With Compositional Vector Grammars. In *ACL*.

R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. 2013b. Zero-Shot Learning Through Cross-Modal Transfer. In *NIPS*.

R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, and A. Y. Ng. C. D. Manning and. 2013c. Zero-shot learning through cross-modal transfer. In *Proceedings of the International Conference on Learning Representations (ICLR, Workshop Track)*.

N. Srivastava and R. Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *NIPS*.

A. Torralba, K. P. Murphy, and W. T. Freeman. 2010. Using the forest to see the trees: exploiting context for visual object detection and localization. *Communications of the ACM*.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

B. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. 2010. I2t:image parsing to text description. *IEEE Xplore*.