Parsing Natural Scenes and Natural Language with Recursive Neural Networks

Richard Socher RICHARD@SOCHER.ORG
Cliff Chiung-Yu Lin CHIUNGYU@STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Christopher D. Manning MANNING@STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

Recursive structure is commonly found in the inputs of different modalities such as natural scene images or natural language sentences. Discovering this recursive structure helps us to not only identify the units that an image or sentence contains but also how they interact to form a whole. We introduce a max-margin structure prediction architecture based on recursive neural networks that can successfully recover such structure both in complex scene images as well as sentences. The same algorithm can be used both to provide a competitive syntactic parser for natural language sentences from the Penn Treebank and to outperform alternative approaches for semantic scene segmentation, annotation and classification. For segmentation and annotation our algorithm obtains a new level of state-of-the-art performance on the Stanford background dataset (78.1%). The features from the image parse tree outperform Gist descriptors for scene classification by 4%.

1. Introduction

Recursive structure is commonly found in different modalities, as shown in Fig. 1. The syntactic rules of natural language are known to be recursive, with noun phrases containing relative clauses that themselves contain noun phrases, e.g., ... *the church which has nice windows* .... Similarly, one finds nested hierarchical structuring in scene images that capture both part-of and proximity relationships. For instance, cars are often on top of street regions. A large car region

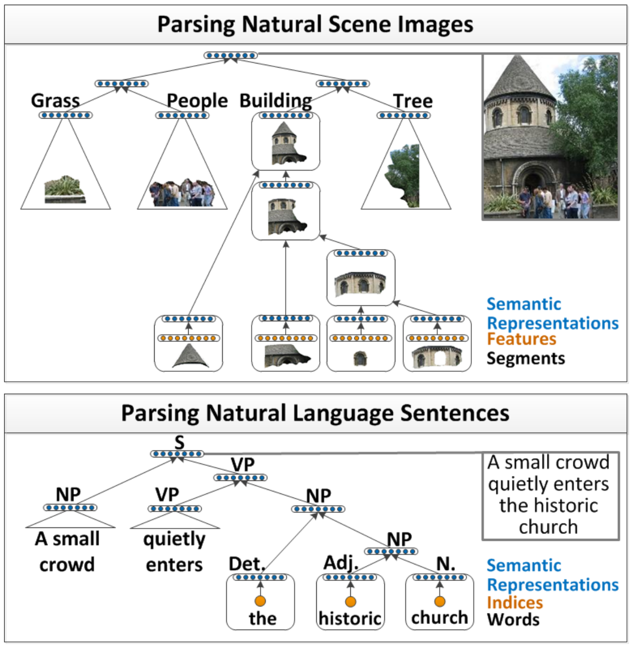


Figure 1. Illustration of our recursive neural network architecture which parses images and natural language sentences. Segment features and word indices (orange) are first mapped into semantic feature space (blue) and then recursively merged by the same neural network until they represent the entire image or sentence. Both mappings and mergings are learned.

can be recursively split into smaller car regions depicting parts such as tires and windows and these parts can occur in other contexts such as beneath airplanes or in houses. We show that recovering this structure helps in understanding and classifying scene images. In this paper, we introduce recursive neural networks (RNNs) for predicting recursive structure in multiple modalities. We primarily focus on scene understanding, a central task in computer vision often subdivided into segmentation, annotation and classification of scene images. We show that our algorithm is a general tool

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

for predicting tree structures by also using it to parse natural language sentences.

Fig. 1 outlines our approach for both modalities. Images are oversegmented into small regions which often represent parts of objects or background. From these regions we extract vision features and then map these features into a "semantic" space using a neural network. Using these semantic region representations as input, our RNN computes (i) a score that is higher when neighboring regions should be merged into a larger region, (ii) a new semantic feature representation for this larger region, and (iii) its class label. Class labels in images are visual object categories such as *building* or *street*. The model is trained so that the score is high when neighboring regions have the same class label. After regions with the same object label are merged, neighboring objects are merged to form the full scene image. These merging decisions implicitly define a tree structure in which each node has associated with it the RNN outputs (i)-(iii), and higher nodes represent increasingly larger elements of the image.

The same algorithm is used to parse natural language sentences. Again, words are first mapped into a semantic space and then they are merged into phrases in a syntactically and semantically meaningful order. The RNN computes the same three outputs and attaches them to each node in the parse tree. The class labels are phrase types such as noun phrase (NP) or verb phrase (VP).

Contributions. This is the first deep learning method to achieve state-of-the-art results on segmentation and annotation of complex scenes. Our recursive neural network architecture predicts hierarchical tree structures for scene images and outperforms other methods that are based on conditional random fields or combinations of other methods. For scene classification, our learned features outperform state of the art methods such as Gist descriptors. Furthermore, our algorithm is general in nature and can also parse natural language sentences obtaining competitive performance on maximum length 15 sentences of the Wall Street Journal dataset. Code for the RNN model is available at `www.socher.org`.

2. Related Work

Five key research areas influence and motivate our method. We briefly outline connections and differences between them. Due to space constraints, we cannot do justice to the complete literature.

Scene Understanding has become a central task in computer vision. The goal is to understand what objects are in a scene (annotation), where the objects are located (segmentation) and what general scene type the image shows (classification). Some methods for this task such as (Aude & Torralba, 2001; Schmid, 2006) rely on a global descriptor which can do very well for classifying scenes into broad categories. However, these approaches fail to gain a deeper understanding of the objects in the scene. At the same time, there is a myriad of different approaches for image annotation and semantic segmentation of objects into regions (Rabinovich et al., 2007; Gupta & Davis, 2008). Recently, these ideas have been combined to provide more detailed scene understanding (Hoiem et al., 2006; Li et al., 2009; Gould et al., 2009; Socher & Fei-Fei, 2010).

Our algorithm *parses* an image; that is, it recursively merges pairs of segments into *super segments* in a semantically and structurally coherent way. Many other scene understanding approaches only consider a flat set of regions. Some approaches such as (Gould et al., 2009) also consider merging operations. For merged super segments, they compute new features. In contrast, our RNN-based method *learns* a representation for super segments. This learned representation together with simple logistic regression outperforms the original vision features and complex conditional random field models. Furthermore, we show that the image parse trees are useful for scene classification and outperform global scene features such as Gist descriptors (Aude & Torralba, 2001).

Syntactic parsing of natural language sentences is a central task in natural language processing (NLP) because of its importance in mediating between linguistic expression and meaning. Our RNN architecture jointly learns how to parse and how to represent phrases in a continuous vector space of features. This allows us to embed both single lexical units and unseen, variable-sized phrases in a syntactically coherent order. The learned feature representations capture syntactic and compositional-semantic information. We show that they can help inform accurate parsing decisions and capture interesting similarities between phrases and sentences.

Using NLP techniques in computer vision. The connection between NLP ideas such as parsing or grammars and computer vision has been explored before (Zhu & Mumford, 2006; Tighe & Lazebnik, 2010; Zhu et al., 2010; Siskind et al., 2007), among many others. Our approach is similar on a high level, however, more general in nature. We show that the same neural network based architecture can be used for both natural language and image parsing.

Deep Learning in vision applications can find lower dimensional representations for fixed size input images which are useful for classification (Hinton & Salakhutdinov, 2006). Recently, Lee et al. (2009) were able to scale up deep networks to more realistic image sizes. Using images of single objects which were all in roughly the same scale, they were able to learn parts and classify the images into object categories. Our approach differs in several fundamental ways to any previous deep learning algorithm. (i) Instead of learning features from raw, or whitened pixels, we use off-the-shelf vision features of segments obtained from oversegmented full scene images. (ii) Instead of building a hierarchy using a combination of convolutional and max-pooling layers, we recursively apply the same network to merged segments and give each of these a semantic category label. (iii) This is the first deep learning work which learns full scene segmentation, annotation and classification. The objects and scenes vary in scale, viewpoint, lighting etc.

Using deep learning for NLP applications has been investigated by several people (*inter alia* Bengio et al., 2003; Henderson, 2003; Collobert & Weston, 2008). In most cases, the inputs to the neural networks are modified to be of equal size either via convolutional and max-pooling layers or looking only at a fixed size window around a specific word. Our approach is different in that it handles variable sized sentences in a natural way and captures the recursive nature of natural language. Furthermore, it jointly learns parsing decisions, categories for each phrase and phrase feature embeddings which capture the semantics of their constituents. In (Socher et al., 2010) we developed an NLP specific parsing algorithm based on RNNs. That algorithm is a special case of the one developed in this paper.

3. Mapping Segments and Words into Syntactico-Semantic Space

This section contains an explanation of the inputs used to describe scene images and natural language sentences and how they are mapped into the space in which the RNN operates.

3.1. Input Representation of Scene Images

We closely follow the procedure described in (Gould et al., 2009) to compute image features. First, we oversegment an image x into superpixels (also called segments) using the algorithm from (Comaniciu & Meer, 2002). Instead of computing multiple oversegmentations, we only choose one set of parameters. In our dataset, this results in an average of 78 segments per image. We compute 119 features for the segments as described in Sec. 3.1 of (Gould et al., 2009). These features include color and texture features (Shotton et al., 2006), boosted pixel classifier scores (trained on the labeled training data), as well as appearance and shape features.

Next, we use a simple neural network layer to map these features into the "semantic" n-dimensional space in which the RNN operates. Let F_i be the features described above for each segment $i = 1, \dots, N_{segs}$ in an image. We then compute the representation:

$$a_i = f(W^{sem} F_i + b^{sem}), \tag{1}$$

where $W^{sem} \in \mathbb{R}^{n \times 119}$ is the matrix of parameters we want to learn, b^{sem} is the bias and f is applied element-wise and can be any sigmoid-like function. In our vision experiments, we use the original sigmoid function $f(x) = 1/(1 + e^{-x})$.

3.2. Input Representation for Natural Language Sentences

In order to efficiently use neural networks in NLP, neural language models (Bengio et al., 2003; Collobert & Weston, 2008) map words to a vector representation. These representations are stored in a word embedding matrix $L \in \mathbb{R}^{n \times |V|}$, where $|V|$ is the size of the vocabulary and n is the dimensionality of the semantic space. This matrix usually captures co-occurrence statistics and its values are learned. Assume we are given an ordered list of N_{words} words from a sentence x. Each word $i = 1, \dots, N_{words}$ has an associated vocabulary index k into the columns of the embedding matrix. The operation to retrieve the i-th word's semantic representation can be seen as a simple projection layer where we use a binary vector e_k which is zero in all positions except at the k-th index,

$$a_i = Le_k \in \mathbb{R}^n. \tag{2}$$

As with the image segments, the inputs are now mapped to the semantic space of the RNN.

4. Recursive Neural Networks for Structure Prediction

In our discriminative parsing architecture, the goal is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible binary parse trees. An input x consists of two parts: (i) A set of activation vectors $\{a_1, \dots, a_{N_{segs}}\}$, which represent input elements such as image segments or words of a sentence. (ii) A symmetric adjacency matrix A, where $A(i, j) = 1$, if segment i neighbors j. This matrix defines which elements can be merged. For sentences, this matrix has a special form with 1's only on the first diagonal below and above the main diagonal.

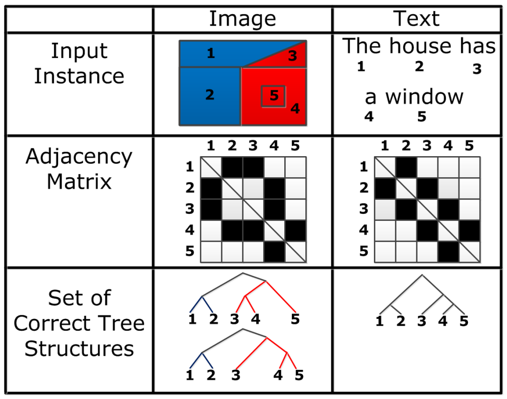


Figure 2. Illustration of the RNN training inputs: An adjacency matrix of image segments or words. A training image (red and blue are differently labeled regions) defines a set of correct trees which is oblivious to the order in which segments with the same label are merged. See text for details.

We denote the set of all possible trees that can be constructed from an input x as $\mathcal{T}(x)$. When training the visual parser, we have labels l for all segments. Using these labels, we can define an equivalence set of correct trees $Y(x,l)$. A visual tree is correct if all adjacent segments that belong to the same class are merged into one super segment before merges occur with super segments of different classes. This equivalence class over trees is oblivious to how object parts are internally merged or how complete, neighboring objects are merged into the full scene image. For training the language parser, the set of correct trees only has one element, the annotated ground truth tree: $Y(x) = \{y\}$. Fig. 2 illustrates this.

4.1. Max-Margin Estimation

Similar to (Taskar et al., 2004), we define a structured margin loss $\Delta(x,l,\hat{y})$ for proposing a parse $\hat{y}$ for input x with labels l. The loss increases when a segment merges with another one of a different label before merging with all its neighbors of the same label. We can formulate this by checking whether the subtree $subTree(d)$ underneath a nonterminal node d in $\hat{y}$ appears in any of the ground truth trees of $Y(x,l)$:

$$\Delta(x,l,\hat{y}) = \kappa \sum_{d \in N(\hat{y})} \mathbf{1}\{subTree(d) \notin Y(x,l)\}, \qquad (3)$$

where $N(\hat{y})$ is the set of non-terminal nodes and κ is a parameter. The loss of the language parser is the sum over incorrect spans in the tree, see (Manning & Schütze, 1999).

Given the training set, we search for a function f with small expected loss on unseen inputs. We consider the following functions:

$$f_\theta(x) = \arg\max_{\hat{y} \in \mathcal{T}(x)} \mathrm{s}(\mathrm{RNN}(\theta, x, \hat{y})), \qquad (4)$$

where θ are all the parameters needed to compute a score s with an RNN. The score of a tree y is high if the algorithm is confident that the structure of the tree is correct. Tree scoring with RNNs will be explained in detail below. In the max-margin estimation framework (Taskar et al., 2004; Ratliff et al., 2007), we want to ensure that the highest scoring tree is in the set of correct trees: $f_\theta(x_i) \in Y(x_i, l_i)$ for all training instances (x_i, l_i), $i = 1, \ldots, n$. Furthermore, we want the score of the highest scoring correct tree y_i to be larger up to a margin defined by the loss Δ. $\forall i, \hat{y} \in \mathcal{T}(x_i)$:

$$\mathrm{s}(\mathrm{RNN}(\theta, x_i, y_i)) \geq \mathrm{s}(\mathrm{RNN}(\theta, x_i, \hat{y})) + \Delta(x_i, l_i, \hat{y}).$$

These desiderata lead us to the following regularized risk function:

$$\begin{aligned} J(\theta) &= \frac{1}{N}\sum_{i=1}^{N} r_i(\theta) + \frac{\lambda}{2}||\theta||^2, \quad \text{where} \qquad (5)\\ r_i(\theta) &= \max_{\hat{y} \in \mathcal{T}(x_i)} \big(s(\mathrm{RNN}(\theta, x_i, \hat{y})) + \Delta(x_i, l_i, \hat{y})\big) \\ &\quad - \max_{y_i \in Y(x_i, l_i)} \big(s(\mathrm{RNN}(\theta, x_i, y_i))\big) \end{aligned}$$

Minimizing this objective maximizes the correct tree's score and minimizes (up to a margin) the score of the highest scoring but incorrect tree.

Now that we defined the general learning framework, we will explain in detail how we predict parse trees and compute their scores with RNNs.

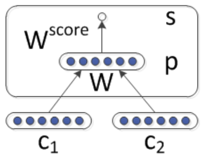
4.2. Greedy Structure Predicting RNNs

We can now describe the RNN model that uses the activation vectors $\{a_1, \ldots, a_{N_{segs}}\}$ and adjacency matrix A (as defined above) as inputs. There are more than exponentially many possible parse trees and no efficient dynamic programming algorithms for our RNN setting. Therefore, we find a greedy approximation. We start by explaining the feed-forward process on a test input.

Using the adjacency matrix A, the algorithm finds the pairs of neighboring segments and adds their activations to a set of potential child node pairs:

$$C = \{[a_i, a_j] : \mathrm{A(i,j)}{=}1\}. \qquad (6)$$

In the small toy image of Fig. 2, we would have the following pairs: $\{[a_1,a_2], [a_1,a_3], [a_2,a_1], [a_2,a_4], [a_3,a_1], [a_3,a_4], [a_4,a_2], [a_4,a_3], [a_4,a_5], [a_5,a_4]\}$. Each pair of activations is concatenated and given as input to a



$$s = W^{score} p \quad (9)$$
$$p = f(W[c_1; c_2] + b)$$

Figure 3. One recursive neural network which is replicated for each pair of possible input vectors. This network is different to the original RNN formulation in that it predicts a score for being a correct merging decision.

neural network. The network computes the potential parent representation for these possible child nodes:

$$p_{(i,j)} = f(W[c_i; c_j] + b). \quad (7)$$

With this representation we can compute a local score using a simple inner product with a row vector $W^{score} \in \mathbb{R}^{1\times n}$:

$$s_{(i,j)} = W^{score} p_{(i,j)}. \quad (8)$$

The network performing these functions is illustrated in Fig. 3. Training will aim to increase scores of good segment pairs (with the same label) and decrease scores of pairs with different labels, unless no more good pairs are left.

After computing the scores for all pairs of neighboring segments, the algorithm selects the pair which received the highest score. Let the score s_{ij} be the highest score; we then (i) Remove $[a_i, a_j]$ from C, as well as all other pairs with either a_i or a_j in them. (ii) Update the adjacency matrix with a new row and column that reflects that the new segment has the neighbors of both child segments. (iii) Add potential new child pairs to C:

$$C = C - \{[a_i, a_j]\} - \{[a_j, a_i]\} \quad (10)$$
$$C = C \cup \{[p_{(i,j)}, a_k] : a_k \text{ has boundary with } i \text{ or } j\}$$

In the case of the image in Fig. 2, if we merge $[a_4, a_5]$, then $C = \{[a_1, a_2], [a_1, a_3], [a_2, a_1], [a_2, p_{(4,5)}], [a_3, a_1], [a_3, p_{(4,5)}], [p_{(4,5)}, a_2], [p_{(4,5)}, a_3]\}$.

The new potential parents and corresponding scores of new child pairs are computed *with the same neural network* of Eq. 7. For instance, we compute, $p_{(2,(4,5))} = f(W[a_2, p_{(4,5)}] + b)$, $p_{(3,(4,5))} = f(W[a_3, p_{(4,5)}] + b)$, etc.

The process repeats (treating the new $p_{i,j}$ just like any other segment) until all pairs are merged and only one parent activation is left in the set C. This activation then represents the entire image. Hence, the *same* network (with parameters W, b, W^{score}) is *recursively* applied until all vector pairs are collapsed. The tree is then recovered by unfolding the collapsed decisions down to the original segments which are the leaf nodes of the tree. The final score that we need for structure prediction is simply the sum of all the local decisions:

$$s(\text{RNN}(\theta, x_i, \hat{y})) = \sum_{d \in N(\hat{y})} s_d. \quad (11)$$

To finish the example, assume the next highest score was $s_{((4,5),3)}$, so we merge the $(4,5)$ super segment with segment 3, so $C = \{[a_1, a_2], [a_1, p_{((4,5),3)}], [a_2, a_1], [a_2, p_{((4,5),3)}], [p_{((4,5),3)}, a_1], [p_{((4,5),3)}, a_2]\}$. If we then merge segments $(1, 2)$, we get $C = \{[p_{(1,2)}, p_{((4,5),3)}], [p_{((4,5),3)}, p_{(1,2)}]\}$, leaving us with only the last choice of merging the differently labeled super segments. This results in the bottom tree in Fig. 2.

4.3. Category Classifiers in the Tree

One of the main advantages of our approach is that each node of the tree built by the RNN has associated with it a distributed feature representation (the parent vector p). We can leverage this representation by adding to each RNN parent node (after removing the scoring layer) a simple softmax layer to predict class labels, such as visual or syntactic categories:

$$label_p = softmax(W^{label} p). \quad (12)$$

When minimizing the cross-entropy error of this softmax layer, the error will backpropagate and influence both the RNN parameters and the word representations.

4.4. Improvements for Language Parsing

Since in a sentence each word only has 2 neighbors, less-greedy search algorithms such as a bottom-up beam search can be used. In our case, beam search fills in elements of the chart in a similar fashion as the CKY algorithm. However, unlike standard CNF grammars, in our grammar each constituent is represented by a continuous feature vector and not just a discrete category. Hence we cannot prune based on category equality. We could keep the k-best subtrees in each cell but initial tests showed no improvement over just keeping the single best constituent in each cell.

Since there is only a single correct tree the second maximization in the objective of Eq. 5 can be dropped. For further details see (Socher et al., 2010).

5. Learning

Our objective J of Eq. 5 is not differentiable due to the hinge loss. Therefore, we will generalize gradient descent via the subgradient method (Ratliff et al., 2007) which computes a gradient-like direction called the subgradient. Let $\theta = (W^{sem}, W, W^{score}, W^{label})$ be the set of our model parameters,[1] then the gradi-

[1]In the case of natural language parsing, $W^{sem}$is replaced by the look-up table L.

ent becomes:

$$\frac{\partial J}{\partial \theta} = \frac{1}{n}\sum_i \frac{\partial s(\hat{y}_i)}{\partial \theta} - \frac{\partial s(y_i)}{\partial \theta} + \lambda\theta, \qquad (13)$$

where $s(\hat{y}_i) = s(\text{RNN}(\theta, x_i, \hat{y}_{max(\mathcal{T}(x_i))}))$ and $s(y_i) = s(\text{RNN}(\theta, x_i, y_{max(Y(x_i,l_i))}))$. In order to compute Eq. 13 we calculate the derivative by using backpropagation through structure (Goller & Küchler, 1996), a simple modification to general backpropagation where error messages are split at each node and then propagated to the children.

We use L-BFGS over the complete training data to minimize the objective. Generally, this could cause problems due to the non-differentiable objective function. However, we did not observe problems in practice.

6. Experiments

We evaluate our RNN architecture on both vision and NLP tasks. The only parameters to tune are n, the size of the hidden layer; κ, the penalization term for incorrect parsing decisions and λ, the regularization parameter. We found that our method is robust to these parameters, varying in performance by only a few percent for some parameter combinations. With proper regularization, training accuracy was highly correlated with test performance. We chose $n = 100$, $\kappa = 0.05$ and $\lambda = 0.001$.

6.1. Scene Understanding: Segmentation and Annotation

The vision experiments are performed on the Stanford background dataset[2]. We first provide accuracy of multiclass segmentation where each pixel is labeled with a semantic class. Like (Gould et al., 2009), we run five-fold cross validation and report pixel level accuracy in Table 1. After training the full RNN model which influences the leaf embeddings through backpropagation, we can simply label the superpixels by their most likely class based on the multinomial distribution from the softmax layer at the leaf nodes. As shown in Table 1, we outperform previous methods that report results on this data, such as the recent methods of (Tighe & Lazebnik, 2010). We report accuracy of an additional logistic regression baseline to show the improvement using the neural network layer instead of the raw vision features. We also tried using just a single neural network layer followed by a softmax layer. This corresponds to the leaf nodes of the RNN and performed about 2% worse than the full RNN model.

[2]The dataset is available at
`http://dags.stanford.edu/projects/scenedataset.html`

Table 1. Pixel level multi-class segmentation accuracy of other methods and our proposed RNN architecture on the Stanford background dataset. TL(2010) methods are reported in (Tighe & Lazebnik, 2010).

Method and Semantic Pixel Accuracy in	%
Pixel CRF, Gould et al.(2009)	74.3
Log. Regr. on Superpixel Features	75.9
Region-based energy, Gould et al.(2009)	76.4
Local Labeling,TL(2010)	76.9
Superpixel MRF,TL(2010)	77.5
Simultaneous MRF,TL(2010)	77.5
RNN (our method)	**78.1**

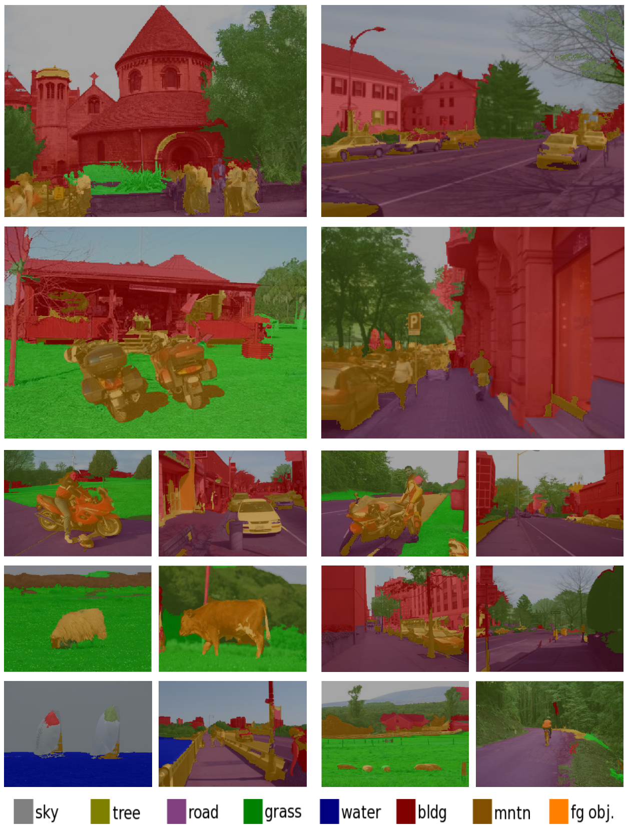


Figure 4. Results of multi-class image segmentation and pixel-wise labeling with recursive neural networks. Best viewed in color.

On a 2.6GHz laptop our Matlab implementation needs 16 seconds to parse 143 test images. We show segmented and labeled scenes in Fig. 4.

6.2. Scene Classification

The Stanford background dataset can be roughly categorized into three scene types: city, countryside and sea-side. We label the images with these three labels and train a linear SVM using the average over all nodes' activations in the tree as features. Hence, we use the entire parse tree and the learned feature repre-

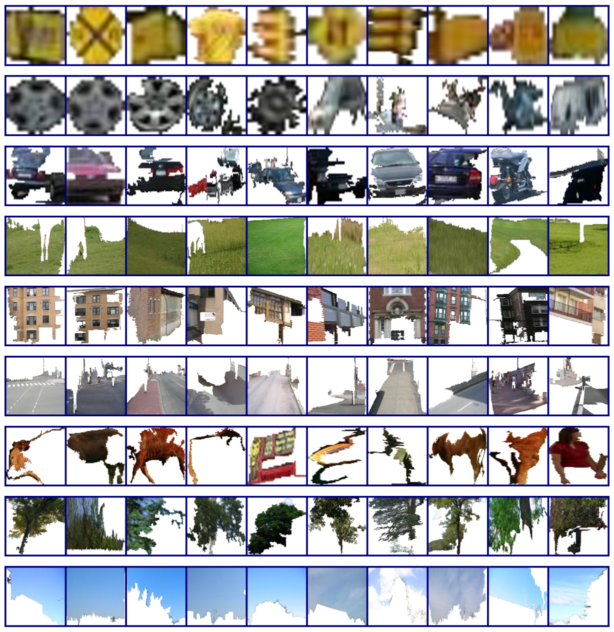


Figure 5. Nearest neighbor image region trees (of the first region in each row): The learned feature representations of higher level nodes capture interesting visual and semantic properties of the merged segments below them.

sentations of the RNN. With an accuracy of 88.1%, we outperform the state-of-the art features for scene categorization, Gist descriptors (Aude & Torralba, 2001), which obtain only 84.0%. We also compute a baseline using our RNN. In the baseline we use as features only the very top node of the scene parse tree. We note that while this captures enough information to perform well above a random baseline (71.0% vs. 33.3%), it does lose some information that is captured by averaging all tree nodes.

6.3. Nearest Neighbor Scene Subtrees

In order to show that the learned feature representations capture important appearance and label information even for higher nodes in the tree, we visualize nearest neighbor super segments. We parse all test images with the trained RNN. We then find subtrees whose nodes have all been assigned the same class label by our algorithm and save the top nodes' vector representation of that subtree. This also includes initial superpixels. Using this representation, we compute nearest neighbors across all images and all such subtrees (ignoring their labels). Fig. 5 shows the results. The first image is a random subtree's top node and the remaining regions are the closest subtrees in the dataset in terms of Euclidean distance between the vector representations.

Center Phrase and Nearest Neighbors	
All the figures are adjusted for seasonal variations	
1. All the numbers are adjusted for seasonal fluctuations	
2. All the figures are adjusted to remove usual seasonal patterns	
3. All Nasdaq industry indexes finished lower , with financial issues hit the hardest	
Knight-Ridder would n't comment on the offer	
1. Harsco declined to say what country placed the order	
2. Coastal would n't disclose the terms	
3. Censorship is n't a Marxist invention	
Sales grew almost 7% to $UNK m. from $UNK m.	
1. Sales rose more than 7% to $94.9 m. from $88.3 m.	
2. Sales surged 40% to UNK b. yen from UNK b.	
3. Revenues declined 1% to $4.17 b. from$ 4.19 b.	
Fujisawa gained 50 to UNK	The dollar dropped
1. Mead gained 1 to 37 UNK	1. The dollar retreated
2. Ogden gained 1 UNK to 32	2. The dollar gained
3. Kellogg surged 4 UNK to 7	3. Bond prices rallied

Figure 6. Nearest neighbors phrase trees. The learned feature representations of higher level nodes capture interesting syntactic and semantic similarities between the phrases. (b.=billion, m.=million)

6.4. Supervised Parsing

In all experiments our word and phrase representations are 100-dimensional. We train all models on the Wall Street Journal section of the Penn Treebank using the standard training (2–21), development (22) and test (23) splits.

The final unlabeled bracketing F-measure (see (Manning & Schütze, 1999) for details) of our language parser is 90.29%, compared to 91.63% for the widely used Berkeley parser (Petrov et al., 2006) (development F1 is virtually identical with 92.06% for the RNN and 92.08% for the Berkeley parser). Unlike most previous systems, our parser does not provide a parent with information about the syntactic categories of its children. This shows that our learned, continuous representations capture enough syntactic information to make good parsing decisions.

While our parser does not yet perform as well as the current version of the Berkeley parser, it performs respectably (1.3% difference in unlabeled F1). On a 2.6GHz laptop our Matlab implementation needs 72 seconds to parse 421 sentences of length less than 15.

6.5. Nearest Neighbor Phrases

In the same way we collected nearest neighbors for nodes in the scene tree, we can compute nearest neighbor embeddings of multi-word phrases. We embed complete sentences from the WSJ dataset into the syntactico-semantic feature space. In Fig. 6 we show several example sentences which had similar sentences in the dataset. Our examples show that the learned features capture several interesting semantic and syntactic similarities between sentences or phrases.

7. Conclusion

We have introduced a recursive neural network architecture which can successfully merge image segments or natural language words based on deep learned semantic transformations of their original features. Our method outperforms state-of-the-art approaches in segmentation, annotation and scene classification.

Acknowledgements

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181 and the DARPA Deep Learning program under contract number FA8650-10-C-7020. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government. We would like to thank Tianshi Gao for helping us with the feature computation, as well as Jiquan Ngiam and Quoc Le for many helpful comments.

References

Aude, O. and Torralba, A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*, 42, 2001.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *JMLR*, 3, 2003.

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

Comaniciu, D. and Meer, P. Mean shift: a robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, May 2002.

Goller, C. and Küchler, A. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 1996.

Gould, S., Fulton, R., and Koller, D. Decomposing a Scene into Geometric and Semantically Consistent Regions. In *ICCV*, 2009.

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

Henderson, J. Neural network probability estimation for broad coverage parsing. In *EACL*, 2003.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313, 2006.

Hoiem, D., Efros, A.A., and Hebert, M. Putting Objects in Perspective. *CVPR*, 2006.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

Li, L-J., Socher, R., and Fei-Fei, L. Towards total scene understanding:classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009.

Manning, C. D. and Schütze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. Learning accurate, compact, and interpretable tree annotation. In *ACL*, 2006.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. Objects in context. In *ICCV*, 2007.

Ratliff, N., Bagnell, J. A., and Zinkevich, M. (Online) subgradient methods for structured prediction. In *AIStats*, 2007.

Schmid, Cordelia. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

Siskind, J. M., J. Sherman, Jr, Pollak, I., Harper, M. P., and Bouman, C. A. Spatial Random Tree Grammars for Modeling Hierarchal Structure in Images with Regions of Arbitrary Shape. *IEEE PAMI*, 29, 2007.

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

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

Taskar, B., Klein, D., Collins, M., Koller, D., and Manning, C. Max-margin parsing. In *EMNLP*, 2004.

Tighe, Joseph and Lazebnik, Svetlana. Superparsing: scalable nonparametric image parsing with superpixels. In *ECCV*, 2010.

Zhu, Long, Chen, Yuanhao, Torralba, Antonio, Freeman, William T., and Yuille, Alan L. Part and appearance sharing: Recursive Compositional Models for multiview. In *CVPR*, 2010.

Zhu, Song C. and Mumford, David. A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, 2(4): 259–362, 2006.