Parsing Natural Scenes Using Recursive Neural Network

Shubham Gupta & Vedant Mishra
Advisor : Dr Amitabh Mukherjee
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur, India
{shubhamg, vedant} @iitk.ac.in

April 16, 2013

ABSTRACT

This work explains and analyse the implementation of RNN (Recursive Neural Networks) proposed in [1] to identify different classes in an image. (For e.g. sky, building, water, etc). In our project the image is first over segmented into super-pixels using ***Mean Shift Algorithm,*** *then the features are extracted for each of these segments and mapped into the semantic space. And finally using* ***Recursive Neural Network*** *architecture and features as input, we merge the different neighbouring segments based on the similarity in their features so as to output the parsed image.*

1. INTRODUCTION

Object detection and classification in an image has been one of the most fascinating applicative problems in AI. Many methods have been proposed in this respect. Humans are highly efficient in classifying different objects in an image. Although much of the work has been done in this field but human like efficiency and accuracy is yet not achieved. Humans are highly efficient when it comes to recognizing and classifying objects in an image.

Recursive structure is commonly found in images. For e.g. a building can be recursively broken down into windows, doors, rooftops and rooftop must always be on the top of the building. Building must be on the top of the streets. Thus every scene or image is a function of smaller regions.

Our project focuses on identifying and categorizing different images into various discrete units like sky, buildings, tree to form a whole scene and then try to understand the way these units interact using a Recursive Neural Network. RNN recovers such recursive structure in images and parse them.

2. PREVIOUS WORK

Scene understanding is one of the most fascinating task in AI. Many methods have been proposed. Main work on which our project would be based is Socher et al. ICML 2011 [1].

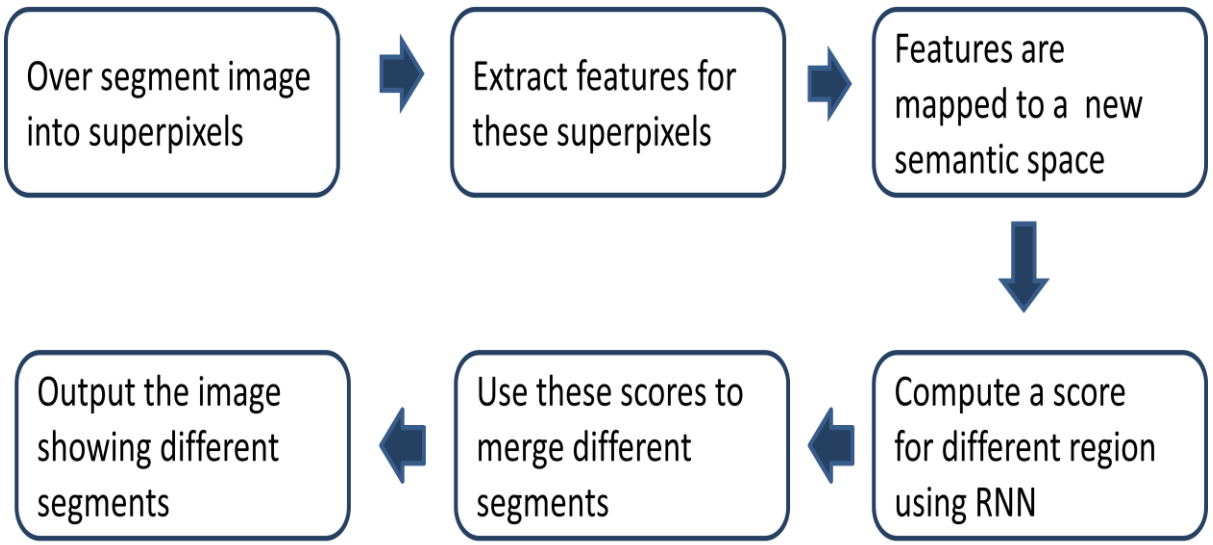
Over segmentation of an image: This uses Comaniciu et al. 2002 [3] which oversegment an image into different superpixels.

Extracting Features: This involves feature extraction of different segments of an image. Some methods for this task such as Gould et al., 2009 [2] uses energy function E for extracting features. At the same time, [4] proposes a method based on texton map for extracting contextual features or information from the surrounding image.

Parsing Natural Scenes: Parsing involves merging of different segments into superpixels based on semantic transformation of their features. Socher et al. ICML 2011 [1] proposes a recursive neural network architecture which for parsing and classifying images.

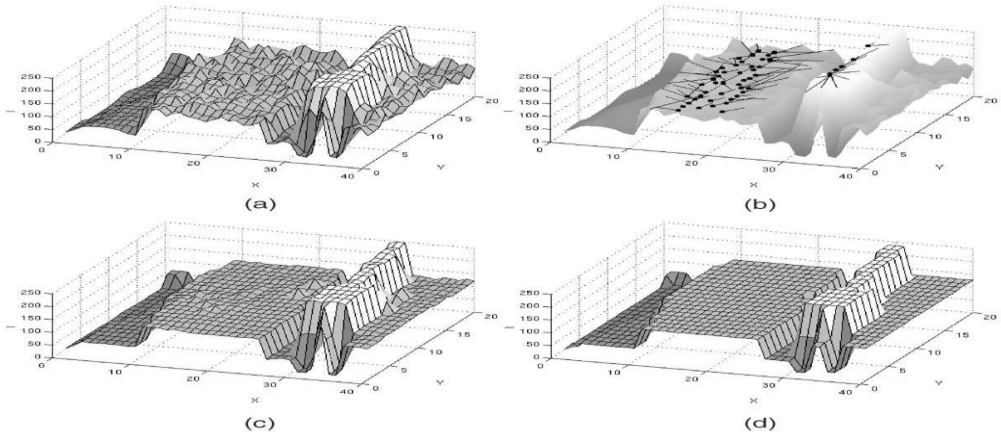
Other work in the recent years on scene classification include Socher & Fei-Fei, 2010 [6] which is based on segmentation and classification. Lee et al. (2009) [5] propose an algorithm based on deep learning for the classification of images with more realistic sizes.

3. OVERVIEW OF THE PROJECT



4. IMAGE SEGMENTATION AND FEATURE EXTRACTION

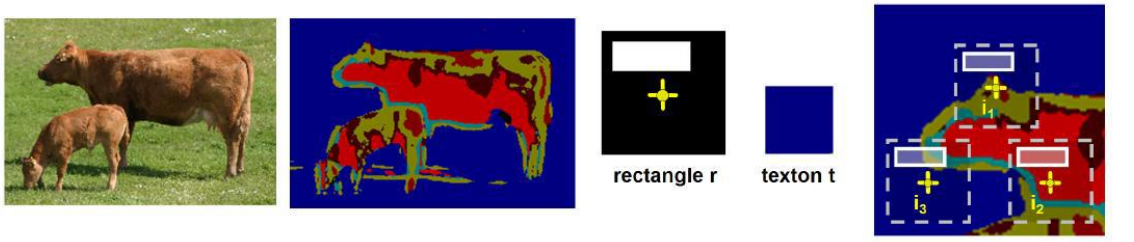
Image segmentation is done using Mean Shift Algorithm. This algorithm segments an image into different regions based on the similarity of their appearances.



(taken from [3])

Basic algorithm for image segmentation is depicted in the above figure. In (a) an image is represented with its spatial representation on x-y plane and feature representation (RGB or L*u*v) on the vertical axis. Now for each pixel a mean-shift vector is calculated (as described in Comaniciu & Meer, 2002 [3]). Based on these vectors a mean shift path is calculated for each of the pixel as shown in (b). This path basically points in the direction of mean shift vectors. The feature space is redefined and smoothened based on the new positions of each pixel on vertical axis (in (c) and (d)).

We then compute features for each of these segments. Using colour histogram we extract its colour features, area, shape. Texton map is used to get the contextual information. We calculate feature response at different position in the texton map. For each feature, we calculate its count and hence exploits the contextual information.



(taken from [4])

These features are then mapped onto a semantic space so as to give them as an input to the neural network using the following equation:

$$a_j = f(W^{sem} F_i + b^{sem})$$ (from Socher et al. ICML 2011 [1])

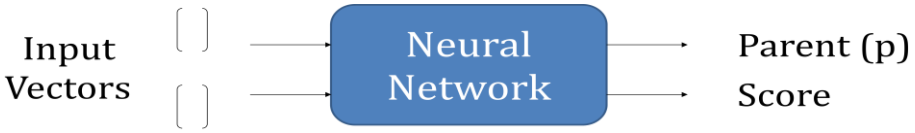
Where a_i is the map in semantic space, F_i is the feature vector and W is the parameter matrix (which is trained along with the other parameters of the network)

5. ALGORITHM FOR PARSING THE IMAGES

RNN Algorithm has two inputs.

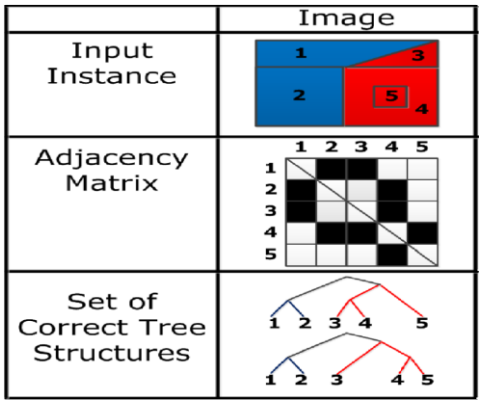
1. Set of input vectors corresponding to each input segment
2. Adjacency matrix A(i,j) which is 1 if segment i is a neighbour of j.

Our Algorithm outputs a new vector corresponding to the merged segment along with its score. Score represents how plausible the new node formed is. Thus if the score is lower, then the two segments won't be merged.



- ARCHITECTURE

Let us assume that we have 5 segments marked as 1,2,3,4,5 in the fig with 1 and 2 having the similar features and label and 3,4,5 having similar features and label.



(Image from Socher et al. ICML 2011 [1])

An image is divided into different segments. We also have the feature vector corresponding to each of the segments (say c1, c2, c3, c4 and c5) calculated earlier. Also we have computed the adjacency matrix describing the neighbourhood relation among different segments.

Neural network calculate all the possible pairs which can be merged and computes the parent of all possible child node pairs using the formula :

$$p = \text{sigmoid}\left(W\begin{Bmatrix}c1\\c2\end{Bmatrix} + b\right), \qquad \text{Using [1]}$$

where W is a parameter and b is bias.

Once we have this parent vector p, we multiply it with parameter matrix so as to get the score.

$$\text{Score} = (W_T)\text{score } p \qquad \text{Using [1]}$$

Out of the all possible neighbouring pairs merging, the one with the highest score is implemented.

Merged child nodes are then replaced by their parent node and again the whole process goes on till we are left with a single merged image.
This whole process can be visualized as having a tree structure. Nodes of the trees representing segments of image at different stages of algorithm. For each possible tree, we sum the score for each node and the one having the highest cumulative score would be the correct visualization of the parsed image. For the above considered case possible correct trees are shown in the figure.

One of the advantages of this network is that it each node in the tree has associated with it a feature vector which can be used to predict the class labels using a simple softmax layer:

$$label_p = softmax(W^{label} p)$$ Using [1]

The parameter W_{label} could be trained with the other parameters.

- TRAINING

Max-Margin estimation is the basic framework implemented for training. Our objective is to maximize the highest scoring correct tree and minimize the score of the highest scoring but incorrect tree. Let s is the score for a given tree structure and loss function, **Δ** represents its deviation from correct parse tree. Theta, **Θ** be the parameters to be trained. **Ŷ** represents the incorrect tree and y represents the correct tree. Then the desired conditions could be satisfied by minimizing the following risk function:

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

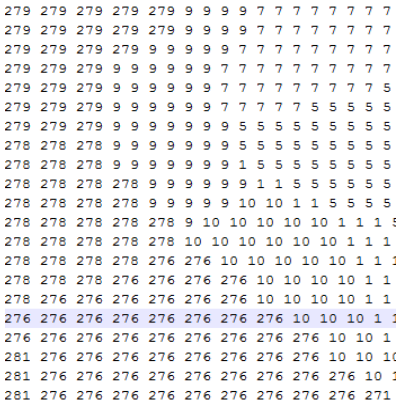
A modified form of gradient descent is used for the given risk function. The gradient is described as:

$$\frac{\partial J}{\partial \theta} = \frac{1}{n}\sum_{i} \frac{\partial s(\hat{y}_i)}{\partial \theta} - \frac{\partial s(y_i)}{\partial \theta} + \lambda\theta$$ from [1]

This equation is calculated using back propagation through structure. This involves the cumulative error of the node and its child nodes. L-BFGS algorithm is implemented over the whole training set to minimize our objective.

6. RESULTS

- Oversegmentation



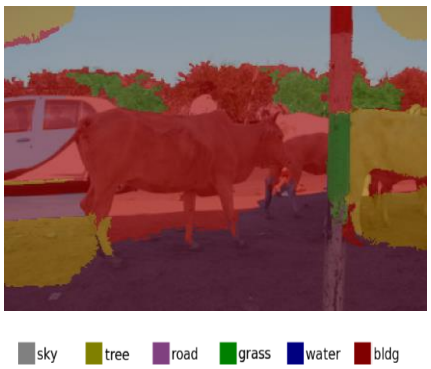
Test Image

Image after Segmentation

Screenshot of Matrix denoting different segments

The matrix here shows the labels of different segments of an image. Each number corresponds to a different segment.

- Parsing



In the first image green portion denotes the tree. Grey portion in the upper area denotes the sky.

In the second image the algorithm has considered the car as a foreground object and colour it with saddle brown. And the portion road has been marked with purple colour.

Similar observations could be carried out for 3rd image also. But in all the 3 results there is some portion which is incorrectly marked. This could be due to limited training dataset or some glitches due to feature extraction process.

7. CONCLUSION AND FUTURE WORK

We have made an attempt to implement the Recursive Neural Network for parsing and merging different segments of the images. For this we have used the training set of the data as it is not possible for us to preprocess our own training data due to time constraint. Also the software used for labelling the images (Amazon Mechnical Turk) was not freeware. We have then tested it over our own test data. We took an image and oversegment it into superpixels. We then extracted features of these segments. It can be seen that the results we got were able to parse an image into distinct units but not completely. It showed some errors in categorizing the scene.

Possible ways to improve the performance could be training the network over a larger and more exhaustive dataset. Also we can try to incorporate context into local decisions by looking at the neighboring superpixels.

A possible extension can be that no. of classes in which a scene image is parsed could be increased. Right now the code parses into 8 classes namely grass, building, water, road, mountain, tree, sky and foreground objects. We can further distinguish between different foreground objects.

8. ACKNOWLEDGMENT

We thank Prof. Amitabha Mukerjee for his valuable support throughout the project, guiding us from time to time and helping us get through various roadblock. Also we are thankful to Richard Socher for sharing his code with us and guiding us in every possible way.

9. REFERENCES

[1] Richard Socher and Cliff C. Lin and Andrew Y. Ng and Christopher D. Manning, Parsing Natural Scenes and Natural Language with Recursive Neural Networks, ICML 2011

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

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

[4] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Texton-Boost: Joint appearance, shape and context modeling for multi-class obj. rec. and seg. In ECCV, 2006.

Dataset and source code:

[5]nlp.stanford.edu/~socherr/cppFeatures.tar.bz2

[6]http://www.socher.org/index.php/Main/ParsingNaturalScenesAndNaturalLanguageWithRecursiveNeuralNetworks