Indoor Object Recognition of 3D Kinect Dataset with RNNs

Thiraphat Charoensripongsa, Yue Chen, Brian Cheng

1. Introduction

Recent work at Stanford in the area of scene understanding has involved using recursive neural networks (RNNs). The basic idea behind RNN is that images can be thought of as being composed by small units, and the relationship between the units can be modeled as a recursive structure. According to Socher et al. [1], this concept has proven to be general in nature as it can also parse natural language sentences achieving competitive performance. This algorithm obtains a state-of-the-art performance on the Stanford background dataset.

Recently a 3D dataset of indoor scenes obtained using the Microsoft Kinect was released. Silberman and Fergus [2] show that depth information gives a significant performance improvement for indoor scene understanding. With the addition of depth information, we believe that the performance of the RNN algorithm can be improved. In this study, we modified the existing RNN algorithm to run on the new dataset. Then we incorporate features extracted from the depth information and investigate how it improves performance.

2. The Kinect 3D Dataset

The dataset we are using is provided by NYU and consists of a variety of indoor scenes captured by the Microsoft Kinect [2]. The Kinect has a RGB camera and an infrared camera that provides depth information. The dataset contains 2347 frames across 64 indoor locations. Labeling was done manually through Amazon Mechanical Turk. Unfortunately, there was no standard rule for grouping up similar objects which resulted in the problem of label inconsistency. For instance, a book shelf is sometimes labeled as ‘books cabinet’, ‘bookstand’, or even ‘books_’. Furthermore, rare objects such as ‘roti maker’ or ‘sewing machine’ account for additional labels. The end result is that the dataset has more than 1400 classes, which is intractable for any model to learn. Therefore, we resolve this issue by considering only the 12 most common labels and relabeling the rest, including those previously unlabeled objects, as the background class.

Due to the relatively large size of the Kinect 3D dataset, limited computing power, and time constraints for our experiments, we made a smaller dataset by randomly selecting 745 out of 2347 frames. We split the data into 515 frames and 230 frames for a training set and a test set, respectively, while ensuring that frames from the same scene are not split over both the training and the test set. Table 1 and Table 2 show the statistics of our dataset. From table 1, note that we are given only one scene for class ‘Cafe’ so we include this in the training set.

3. Recursive Neural Networks

Images often contain recursive structure. For example, a house can be recursively divided into regions such as walls, windows, and roofs, which can be further divided into smaller regions. The structure behind how all the regions are related is the basis behind classifying images.

Scene class	Training		Test	
	Scenes	Frames	Scenes	Frames
Bathroom	4	40	2	20
Bedroom	10	118	7	65
Bookstore	2	72	1	10
Café	1	36	0	0
Kitchen	5	74	4	40
Living Room	7	81	5	45
Office	8	94	6	50
Total	37	515	25	230

Table 1. Scene statistics in our 745 frame dataset.

Object class	Train (% Pixels)	Test (% Pixels)
1. background	50.93	40.84
2. bed	1.85	2.20
3. blind	0.80	2.66
4. bookshelf	2.46	2.19
5. cabinet	3.75	5.49
6. ceiling	2.35	2.13
7. floor	2.20	1.06
8. picture	1.70	1.01
9. sofa	1.18	2.62
10. table	4.02	2.58
11. television	0.57	0.73
12. wall	27.27	32.67
13. window	0.91	3.82

Table 2. Objects statistics in our 745 frame dataset.

The RNN algorithm first splits an image into many segments. Image segmentation is performed by the Edge Detection and Image Segmentation System (EDISON) [3]. Next, the algorithm recursively combines pairs of segments into super segments to form a tree structure, where nodes closer to the root represent larger regions of the image.

Training data is used to teach the system how to predict trees accurately. Each tree has a score associated with it. A higher score means higher confidence that the tree structure is correct. The learning algorithm uses training examples to obtain parameters that minimize a risk function [1]. The result is that a correct tree's score is maximized while the highest scoring incorrect tree's score is minimized (to a certain extent).

4. Feature Extraction

Similar to Socher et al. [1], we oversegment an image into approximately 200 segments and compute 2D features for the segments. We investigate over a set of interesting features including RGB and Lab color, texture, and position features. Furthermore, we focus on developing features from depth information to incorporate with 2D features.

4.1 Normalized Depth Feature

This feature simply normalizes the depth map to the range [0, 1] and extracts a histogram, mean value, and standard deviation of the normalized depth in each segment. It's easy to implement and provides a good indicator for certain classes (e.g. 'wall'). However, it is sensitive to outliers and objects far away in the scene. If there is one pixel in the image that has a large depth value, the normalized depth of the entire scene will be quite different than the same feature calculated without that point, which will affect the prediction accuracy.

4.2 Normal Vector Feature

We try to explore the 3D information provided by the depth map by incorporating the normal vector of the corresponding object surface of each pixel. Given the depth map, the normal

vector of each pixel is computed by doing the cross product of partial derivatives along each axis (tangent). Figure 1 shows the partial derivatives used to calculate the normal vector. This feature contains rich information about the 3D geometry, and is helpful in distinguishing between objects such as 'table', 'ceiling', and 'wall'.

5. Experiments

We performed experiments with our modified Kinect dataset. Due to a highly unbalanced number of objects, the pixel level accuracy is not a good measurement as one simple model can obtain up to 40% accuracy by just predicting every object as a background. Therefore, we instead compute the accuracy by averaging the classification accuracy of all 13 classes and use this measurement throughout this report.

5.1 3D Features

We start off by considering a simple feature, Lab color, and compute the accuracy as a baseline. Then we incorporate depth features, which are the normalized depth feature and normal vector feature described in the previous section. From the result shown in Table 3, we have the highest accuracy over the 2D features, Lab + RGB + texture + position, combined with our 3D features. This gives us a performance gain of over 15% in accuracy compared to the best result using 2D features that we have experimented with.

Feature	Accuracy (%)
Lab	17.38
Lab + Depth	23.29
all2D	29.70
all2D + Depth	32.78
all2D + Depth + Normal Vector	45.60

Table 3. A comparison of different set of features. We use the term *all2D* for Lab + RGB + texture + position.

5.2 **Unbalanced Dataset**

Our indoor dataset is very challenging to deal with in that the total number of pixels for each object class is highly unbalanced. As shown in Table 2, in the training set, we have roughly 50% labeled as background and 20% labeled as wall, while the rest of the classes can be as low as 1%. Concretely, we have far fewer training examples for the classes 'bed', 'blind', and 'television' than for the classes 'background' and 'wall,' which results in fairly low accuracy for those minority classes. Furthermore, the model has a tendency to predict many segments as 'wall' and 'background'. To overcome this problem, we reduce the population of majority classes by randomly excluding some segments from training examples. We found that when some background segments are excluded, the model will tend to classify more objects as 'wall', which is the second largest class, resulting in a drop of accuracy by 0.5%. However, we achieve better accuracy after some wall segments are also excluded. The confusion matrices are shown in Figure 2. Each entry (i, j) of the matrix is for predicting class j with true class i, and the values are normalized such that the sum for each row equals one. The number at the top of the matrix is the classification accuracy.

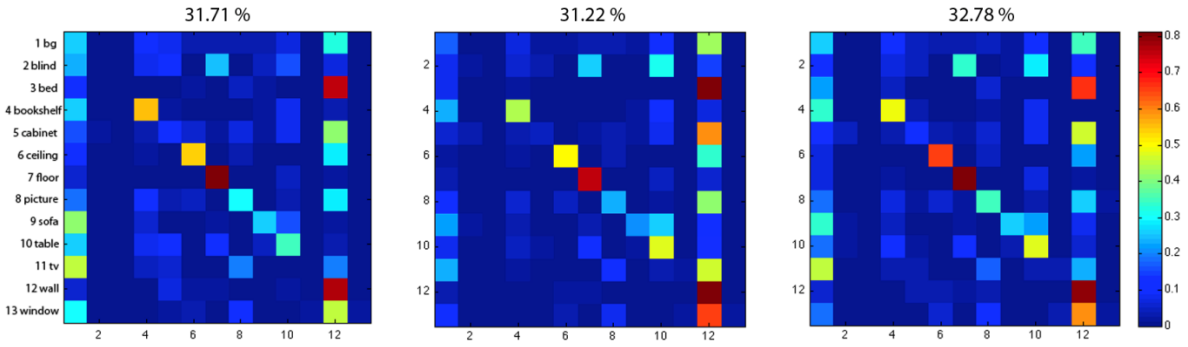

Figure 2. A comparison of normalized confusion matrices. A result from the full training set (Left), excluding some background segments (center), and excluding some background and wall segments (right). The accuracy is shown at the top of the matrix.

5.3 Error Analysis

We are interested in the per-class accuracy of the algorithm using different features. In general, the model is relatively good at predicting the objects ‘wall’, ‘floor’, and ‘ceiling’, but it is difficult to predict the objects ‘bed’, ‘blind’, ‘television’, and ‘window’ as shown in Figure 3. This poses a very challenging problem of how can we improve the performance over those difficult classes with very few training examples. Additionally, it is interesting to see that we have a performance gain over almost all the classes with the depth features.

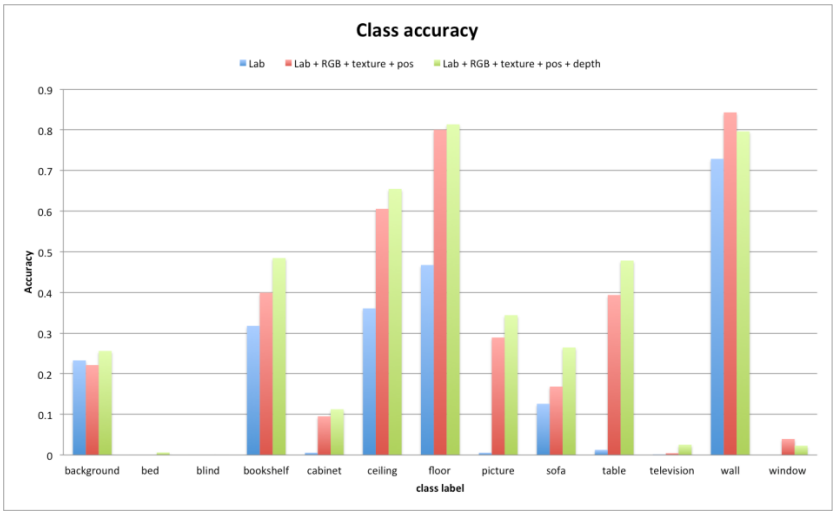

Figure 3. Prediction accuracy for each class for different sets of features.

We also evaluate a frame-level accuracy and discover that label inconsistency in the test set can make the result inaccurate. Figure 4 shows a specific scene where a vast majority of objects are labeled inconsistently resulting in most of the image being relabeled to background. It is obvious that all of the predicted objects of class ‘table’ are all wrong because it does not even exist in the ground truth labels.

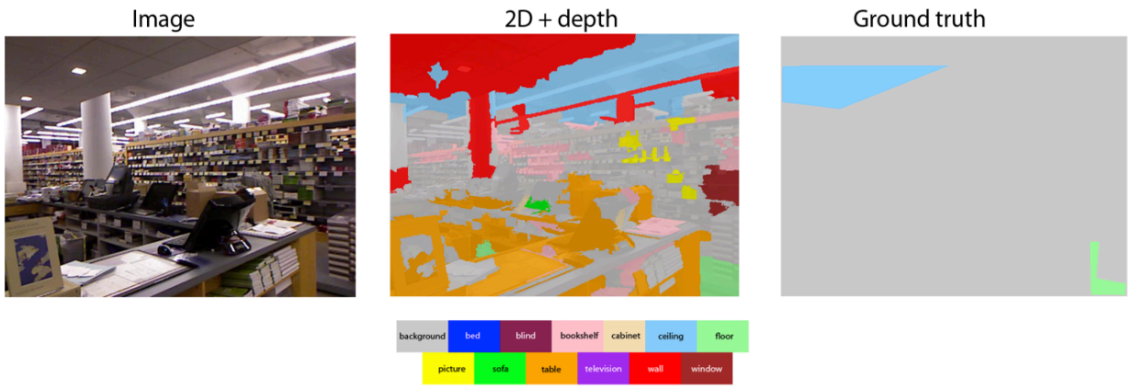

Figure 4. An example of inconsistent labeling in the dataset affecting our class accuracy.

6. Conclusion and Future Work

With the presence of depth information provided by the Kinect dataset, we have introduced 3D features and incorporated them with 2D features for use with the recently proposed RNN-based algorithm to classify objects in indoor environments. The results show that 3D can improve the accuracy by 15.9%. However, the accuracy is largely affected by the problem of unbalanced data and label inconsistency. In the future work, we should study how to effectively resolve these problems. Furthermore, if we are given more powerful computing resources, we would like to experiment with a larger dataset and report the average accuracy over multiple folds.

Acknowledgements

We would like to thank Richard Socher for providing useful advice for this project.

References

1. Socher, R., Lin, C.C., Ng, A.Y., and Manning, C.D. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. Proceedings of the 28th International Conference on Machine Learning, 2011

2. Christoudias, C. M., Georgescu, B., and Meer, P. Synergism in low level vision.
16th International Conference on Pattern Recognition., Quebec City, Canada, August 2002, vol. IV, 150-155.

3. Silberman, N. and Fergus, R. Indoor Scene Segmentation using a Structured Light Sensor. 3DRR Workshop, ICCV 2011.