MULTI-MODAL OBJECT RECOGNITION USING RGB-D AND DEEP LEARNING

Khoi-Nguyen Mac

University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering
knmac@illinois.edu

ABSTRACT

Object recognition is an import topics in different areas, such as: signal processing, computer vision, and deep learning. One of the most common approaches is to train a model using only color images. However, RGB information has multiple drawbacks, such as lighting condition, noisy backgrounds, or inconsistent saturation. Those problems can be solved by introducing depth information to the systems. In this project, I propose to replicate a deep learning multimodal object recognition system from Andreas Eitel etal's paper that can make use of both color and depth information. By experiment, the system achieves the accuracy of 63.66% by fusing the two channels.

Index Terms— Object recognition, deep learning, multimodal, RGB-D, fusion

1. INTRODUCTION AND BACKGROUND

Object recognition is a problem of computer vision, where the objective is to determine objects in an image or video sequence. This task has multiple applications in real life, such as in autonomous vehicles, where pedestrians, stop signs, traffic lights, etc. are automatically recognized, helping the cars react accordingly. It can be used in security systems, such as video surveillance or face recognition.

There are different researches that try to solve this problem. The classical approach is by using hand crafted features. Such features are extracted from images and clustered together such that each cluster represent a specific aspect of the objects. The process's objective is to create a visual dictionary to represent training objects. New objects are classified based on their features, which are extracted in the same manner as training samples, using nearest neighbors approach [1]. With the advance of technologies and algorithms, solutions for object recognition are gradually shifting to deep learning methods, where extracted features are machined-based instead of hand-crafted-based, i.e. features are understandable from computer's perspective instead of human. There are several network architectures, started by AlexNet that won the ImageNet challenge and set the new face for object recognition. Some networks recently such as VGG Net [2], GoogleNet [3], and ResNet [4] are boosting the architecture complexity and achieving state-of-the-art performance.

Color images suffer from numerous problems, such as inconsistent lighting condition, noisy background data, etc. that reduce system's robustness. Such problems can be solved by using depth information, since depth images contain more information about the shape of objects and ignore patterns on the surface. Therefore, depth images have higher consistency among objects from the same category. In addition, depth cameras are getting more popular therefore such images are easily obtainable.

In this project, I propose to reproduce Eitel et al. paper "Multimodal deep learning for robust RGB-D object recognition" [5] to build an object recognition system that combines color and depth information to improve accuracy, implemented using Google's TensorFlow [6]. Depth information in converted into RGB images and fed in a branch of the network along side with normal color information. They are then fused and recomputed to produced the final results.

2. DATASET

In this project, I use "RGB-D object dataset" from University of Washington [7]. The dataset contains 200,000 samples (each is a collection of color and depth images, and a mask for background removal) of 300 household objects, divided into 51 categories (Figure 1). Samples from the original dataset are recorded, synchronized, and aligned at the framerate of 30Hz and resolution of 640×480. However, I use a modified version of the dataset (also provided with the original one), where the objects are localized and cropped out. Each object is recorded on a turntable with three different camera viewpoints: low, middle, and high (Figure 2).

University of Washington provides another dataset for evaluation. This dataset is constructed as the subset of the original one, keeping one sample out of every five. Samples in this set also have their backgrounds removed. The evaluation set comes with a list of 10 trial, each of which is a split of the dataset for evaluation. For each trial, one object per category is selected for validation, leaving the other objects of the same class for training. Since there are 300 objects and 51 categories, there are 51 objects for evaluation and 249 for

Fig. 1: University of Washington's "RGB-D object dataset"

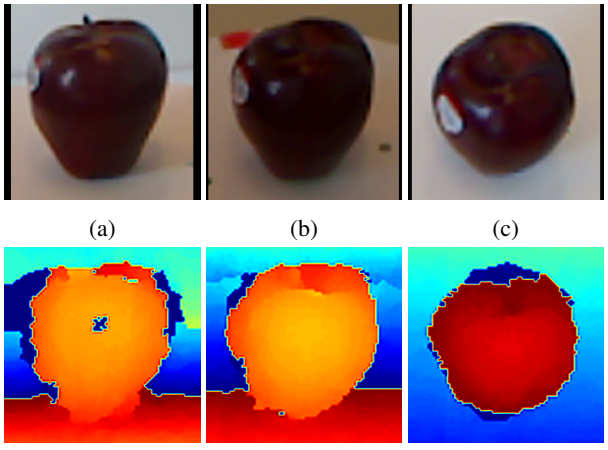

Fig. 2: Color and (visualized) depth images of an apple from 3 different camera viewpoints.

training. For the sake of simplicity, I use only the first trial throughout this project's experiments.

3. METHOD

3.1. Network architecture

Since the dataset we use is lightweight, containing only 200,000 samples, it is essential to reuse pretrained weights from other network because training from scratch with small dataset would likely result in network overfitting and poor results on test data. As proposed in the paper, AlexNet is inherited in this project. However, this network only accepts inputs with three channels (e.g. RGB images), therefore it can only works with color images instead of depth maps. To bypass this issue, depth maps are colorized in the preprocessing step to convert the data from 1D to 3D. More information about this is discussed in Section 3.2.

Figure 3 illustrates the network's architecture, which can be broken into three networks: one for color-based classification, a network for depth-based classification, and one for fusion that combines the two single-modality streams into a final classification result. Architectures of single stream networks are inherited from AlexNet. However, the final fully connected layer (fc8) from the original network is only used for training single stream networks separately. To fuse them, that classification layer is replaced with two different ones: one for concatenating output from the two streams and one fully connected layer for classifying.

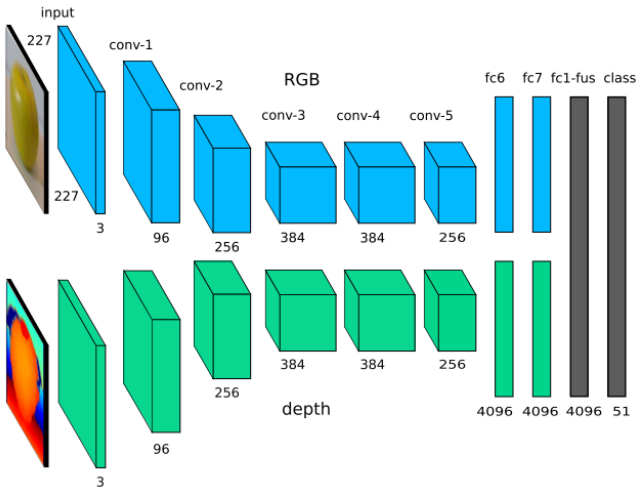

Fig. 3: Fusion network architecture.

3.2. Preprocessing

There are three primary steps in preprocessing: background removal, resizing, and depth colorization. The first step is straightforward as the mask for each sample is provided with the dataset. However, to test the effect of background removal, I also create another version of prepocessed data with background intact.

For resizing, the images are rescaled to the shape of 256×256 although the network receives inputs of dimension 227×227. This is because we want to remove the mean image of original AlexNet, whose size is 256×256. After that, the images are randomly cropped to the size of 227×227 to produce jittering during training. To ensure the shape are not distorted, the images are scaled along the their longer side, the other side is padded with zero (color) and duplicated with boundary (depth), as showed in Figure 2. However, this does not affect if we remove background from the images.

The third step is to colorize depth. Depth sensors like the Xbox Kinect only give a single-channel intensity image proportional to the distance from the sensor. By converting depth images into three-channel data, we can treat them as images and feed into AlexNet (or other pretrained image recognition networks). Although there are different ways to colorize depth maps, jet colorization is proven superior in term of boosting systems performance [5].

3.3. Network training

The training process in divided into two different phases: (1) training the stream networks and (2) training the fusion network. Suppose that we have the dataset

$$\mathcal{D} = \left\{ \left(\mathbf{x}^1, \mathbf{d}^1, \mathbf{y}^1 \right), ..., \left(\mathbf{x}^N, \mathbf{d}^N, \mathbf{y}^N \right) \right\} \tag{1}$$

where $\mathbf{x}^i$ is a RGB image, $\mathbf{d}^i$ is a depth image, and $\mathbf{y}^i$ is a label in the form of

$$\mathbf{y}^i = (y_1^i, y_2^i, ..., y_k^i, ..., y_M^i)^\top \tag{2}$$

$$\text{where} \quad y_k^i = \begin{cases} 1, & \mathbf{x}^i \text{ and } \mathbf{d}^i \text{ belong to class } k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where M is the number of classes. The data are fed into the first training phase and then the second one. In this project, I use Adam optimizer [6][8], included in TensorFlow, to minimize the loss. The learning rates are set as 10^{-3} for color and depth network and 10^{-5} for fusion; default decay rates of the optimizer is used.

3.3.1. *Training the stream networks*

Let $g^I(\mathbf{x}^i; \theta^I)$ be the representation for the color image $\mathbf{x}^i$ of the last fully connected layer from stream networks (*fc7*) of AlexNet, where θ^I is the parameter. Similarly, we have $g^D(\mathbf{d}^i; \theta^D)$ for the depth image $\mathbf{d}^i$. Since we initialize the stream networks with pretrained weights, θ^I and θ^D are known. The weights $\mathbf{W}^I$ and $\mathbf{W}^D$ for RGB and depth streams are trained by solving

$$\min_{\mathbf{W}^I, \theta^I} = \sum_{i=1}^{N} \mathcal{L} \left(\text{softmax} \left(\mathbf{W}^I g^I \left(\mathbf{x}^i; \theta^I \right), \mathbf{y}^i \right) \right), \tag{4}$$

$$\min_{\mathbf{W}^D, \theta^D} = \sum_{i=1}^{N} \mathcal{L} \left(\text{softmax} \left(\mathbf{W}^D g^D \left(\mathbf{d}^i; \theta^D \right), \mathbf{y}^i \right) \right), \tag{5}$$

where softmax function is

$$\text{softmax}(z) = \frac{e^z}{\|z\|_1} \tag{6}$$

and the loss is

$$\mathcal{L}(x, y) = -\sum_k y_k \log s_k. \tag{7}$$

This loss function is actually the "categorical crossentropy" function, which is commonly used in neural networks.

3.3.2. *Training the fusion network*

After acquiring $\mathbf{W}^I$ and $\mathbf{W}^D$, we discard the softmax layers (*fc8*) and concatenate the last responses of the two streams: $g^I(\mathbf{x}^i; \theta^I)$ and $g^D(\mathbf{d}^i; \theta^D)$ and feed them through the additional fusion layer (*fc1_fus*)

$$\mathcal{F} = f \left(\left[g^I(\mathbf{x}^i; \theta^I); g^D(\mathbf{d}^i; \theta^D) \right]; \theta^F \right) \tag{8}$$

where θ^F is the parameters of this layer. We can train this layer with a similar manner as in the previous training phase:

$$\min_{\mathbf{W}^F, \theta^I, \theta^D, \theta^F} = \sum_{i=1}^{N} \mathcal{L} \left(\text{softmax} \left(\mathbf{W}^F \mathcal{F}, \mathbf{y}^i \right) \right) \tag{9}$$

Table 1: Accuracy of the networks

	Color	Depth	Fusion
Training (top 1)	98.48%	96.20%	100%
Evaluation (top 1)	54.80%	71.51%	63.66%
Evaluation (top 5)	76.91%	89.79%	85.63%

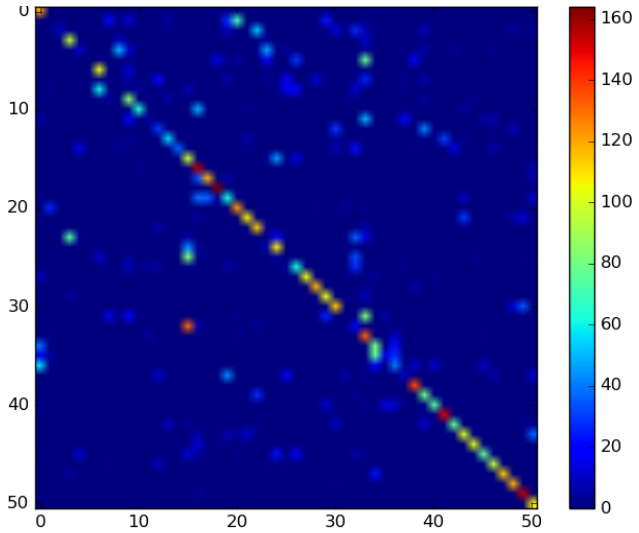

Fig. 4: Confusion matrix of RGB network.

Note that in this phase, the weights trained from the previous one are kept unchanged as they play the role of feature extractor. Only the weights of the fusion network are optimized.

4. EXPERIMENTAL RESULTS

To save time, only the first, out of ten, trial is used to split out training data. For each category, one object is cut out for evaluation while the rest is used for training. Since the number of objects varies for every category, the number of training objects also differs.

By experiment, the accuracy for RGB channel is 54.80%, for depth channel is 71.51%, and for fusion is 63.66%, as seen if Table 1. The results also show that the system is overfitting training data, as the accuracy on training set is approaching 100%.

A more detailed view of the results are illustrated as confusion matrices in Figure 4, 5, and 6, with respect to the recognition results on color, depth, and fusion networks. For a confusion matrices C, the value at cell $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j. Therefore, the more values focusing on the diagonal the better it is since they are the correct recognition results ($i = j$). From confusion matrix figures, most of the results stays on the diagonal but there are several false recognition results on the two halves.

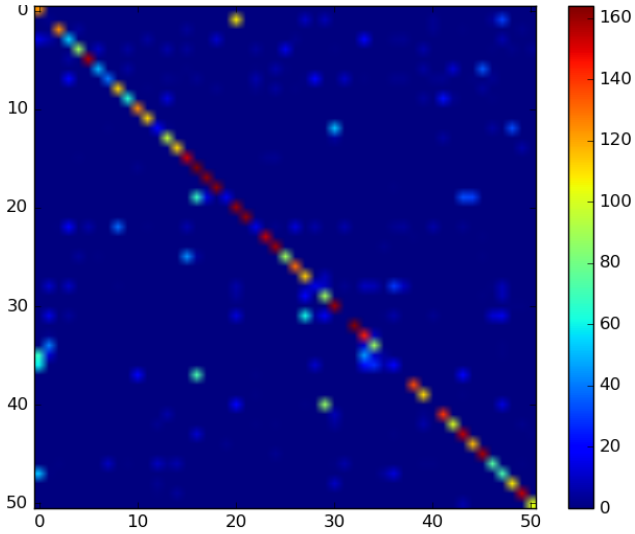

Fig. 5: Confusion matrix of depth network.

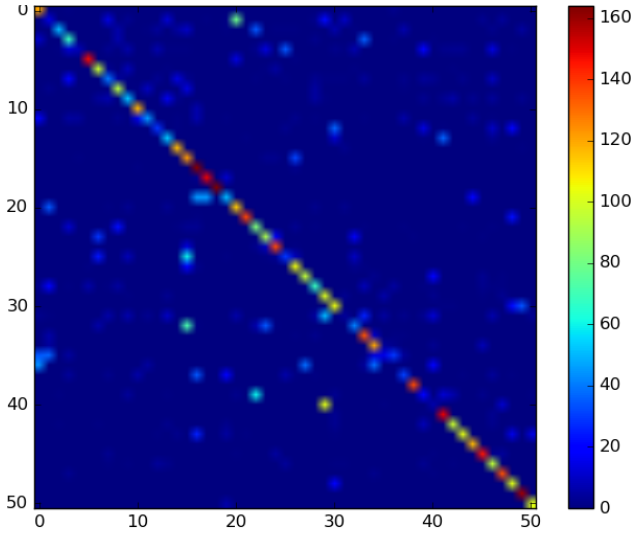

Fig. 6: Confusion matrix of fusion network.

5. CONCLUSIONS

In this project, I build a system that can recognize objects using both color and depth information at the same time using TensorFlow. A surprising result is depth information always produce higher accuracy than color only by a simple preprocessing technique: colormapping. It is suspected that depth images strip off patterns on object's surface and reduce the variance in the system, thus the accuracy is higher. Although the system is overfitting, the results are a starting point for further improvement.

For future work, more experiments with processed depth data are preferred, rather than using raw information. Some possibilities are using ICP algorithmto construct point clouds from depth maps, applying point cloud voxelization, or synthesizing training data from foreign viewpoints. Since depth maps are proved to carry useful information about objects' shape, it is likely that a system cognition can be built using only the mask for background removal. Last but not least, it is promising to try a reversed process from depth colorization, i.e. building a network using only one dimensional depth data.

6. REFERENCES

[1] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, vol. 2, pp. 2169–2178.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[5] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin A. Riedmiller, and Wolfram Burgard, "Multimodal deep learning for robust RGB-D object recognition," *CoRR*, vol. abs/1507.06821, 2015.

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

[7] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 1817–1824.

[8] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.