MULTI-MODAL OBJECT RECOGNITION USING RGB-D

Khoi-Nguyen Mac

University of Illinois at Urbana-Champaign

ABSTRACT

The abstract should appear at the top of the left-hand column of text, about 0.5 inch (12 mm) below the title area and no more than 3.125 inches (80 mm) in length. Leave a 0.5 inch (12 mm) space between the end of the abstract and the beginning of the main text. The abstract should contain about 100 to 150 words, and should be identical to the abstract text submitted electronically along with the paper cover sheet. All manuscripts must be in English, printed in black ink.

Index Terms— One, two, three, four, five

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 [ref???]. 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 [ref???], GoogleNet [ref???], and ResNet [ref???] 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" [ref??] to build an object recognition system that combines color and depth information to improve accuracy, implemented using Google's TensorFlow [ref???]. 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 [ref???]. 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 training. For the sake of simplicity, I use only the first trial throughout this project's experiments.

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

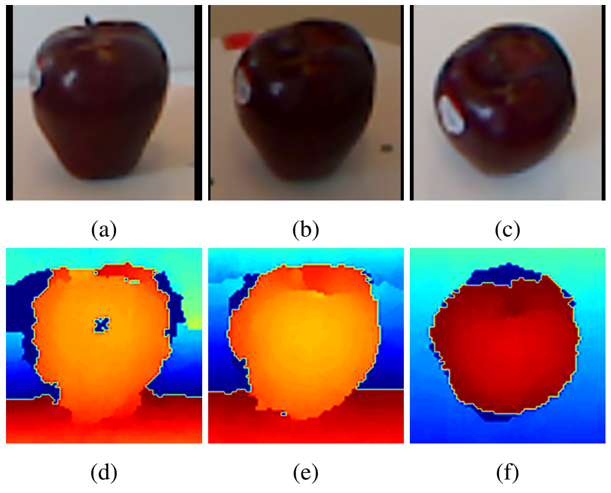

Fig. 2: Color and (visualized) depth images of an apple from 3 different camera viewpoints.

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.

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 re-

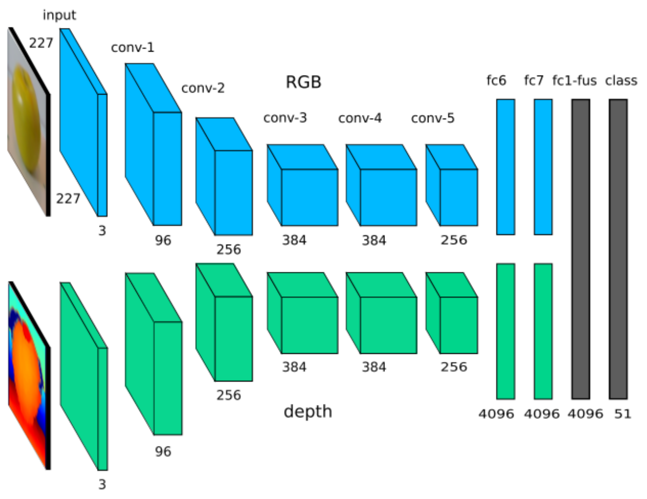

Fig. 3: Fusion network architecture.

moval, 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 [ref???].

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.

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:

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

Note that in this phase, the weights trained from the previous one are kept unchanged. Only the weights of the fusion network are optimized.

3.3.3. Minimizer

4. EXPERIMENTAL RESULTS

5. CONCLUSIONS

6. REFERENCES

List and number all bibliographical references at the end of the paper. The references can be numbered in alphabetic order or in order of appearance in the document. When referring to them in the text, type the corresponding reference number in square brackets as shown at the end of this sentence [**?**]. An additional final page (the fifth page, in most cases) is allowed, but must contain only references to the prior literature.