RGB-D Object Recognition using Deep Learning

Dario Aranguiz
University of Illinois at Urbana-Champaign
arangui2@illinois.edu

Cu-Khoi-Nguyen Mac
University of Illinois at Urbana-Champaign
knmac@illinois.edu

Abstract

TODO

1 Introduction

Define and motivate the problem, discuss background material or related work, and briefly summarize your approach.

Photography has remained generally unchanged since the first photograph was taken in 1816 [1]. The proceeding years since have seen a plethora of technical advances bettering image quality, speed of image capture, as well as other aspects, but imaging has remained a two-dimensional art. Developments in recent years, however, have begun to fundamentally alter how we consider imaging, with the release of the first Xbox Kinect in November 2010 [2]. With this launch, consumers and researchers alike were able to view the world in three dimensions, opening up a second modality to image processing.

2 Details of approach

Include any formulas, pseudocode, diagrams – anything that is necessary to clearly explain your system and what you have done. If possible, illustrate the intermediate stages of your approach with result images.

For our approach, we implemented a variation on the CVPR 2015 paper from Eitel et al titled *Multimodal Deep Learning for Robust RGB-D Object Recognition* [3]. The main contribution of this paper was to show how pretrained networks such as AlexNet or VGGNet can be used to process single-channel depth information, enabling networks to run on depth information despite a lack of large-scale depth datasets such as ImageNet for color images [4]. In this section, we will discuss our network architecture, our preprocessing phase, and our training methodology.

2.1 Washington RGB-D Object Dataset

2.2 Network Architecture

The network structure proposed by Eitel et al [3] is pictured in Figure 2. Its structure can be broken into three parts: a network for color-based classification, a network for depth-based classification, and a final fusion network combining the two single-modality streams into a final classification result. Fusing two single-stream networks is beneficial for two main reasons. Firstly, unlike large companies like Microsoft and Google who have recently become involved in deep learning competitions, we do not have a GPU farm at our disposal to train networks ad-infinitum. Additionally, the first consumer depth camera was only released within the last 5 years [2]. Unlike ImageNet, we do not have millions of labeled depth captures to use for network training. A smaller dataset could be used to train a depth network from scratch, but using such a comparatively small dataset would likely result in network overfitting and poor results on test data.

To this end, we use the pretrained network AlexNet for our single stream network. AlexNet is a top-tier network without as many layers as a deep residual learning network, making it a better choice for our limited compute capacity and timeframe. Our implementation of AlexNet is shown in further detail in Figure 1.

Finally, we strip the `fc8` classification layers from the our AlexNet streams and concatenate the two models, feeding the now 8092-node layer into `fc1-fus`, a 4096-node fully-connected layer. This then goes into a final classification layer with softmax activation. The individual elements of the network are described below.

2.2.1 Convolutional Layer

TODO, what is a convolutional layer

2.2.2 ReLU

TODO, what is ReLU

2.2.3 Batch Normalization

TODO, what is Batch Normalization

2.2.4 Max Pooling

TODO, what is max pooling

2.3 Preprocessing

Preprocessing for this network has three primary stages: segmentation, rescaling, and depth colorization. First, we segment the color and depth images using a mask provided with the dataset, setting pixel values to black in the color image and zero in the depth image. Our dataset provides segmentation masks for every object capture in the dataset, although Eitel et al [3] showed in their paper that doing classification without the segmentation mask was also possible.

The image pairs are then cropped and rescaled to 227×227, the size of the input for the single-stream networks. We do this by scaling the image by a factor:

$$\kappa = \frac{\max(W, H)}{227}.$$

where W and H are object's width and height, respectively. The smaller dimension is then zero-padded to fit the square 227×227. By this way the object is kept as the image's center, making it efficient in case the boundary is cropped during training.

More interesting, however, is the process of depth colorization. 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 system's performance [3].

2.4 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

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

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

$$\mathbf{y}^i = (y_1^i, y_2^i, ..., y_k^i, ..., y_M^i)^\top \qquad \text{s.t.} \qquad y_k^i = \begin{cases} 1, & \mathbf{x}^i \text{ and } \mathbf{d}^i \text{ belong to class } k \\ 0, & \text{otherwise} \end{cases}$$

where M is the number of classes. The data are fed into the first training phase and then the second one.

2.4.1 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

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

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

where softmax function is

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

and the loss is

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

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

2.4.2 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*)

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

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

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

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 Results

Clearly describe your experimental protocols. If you are using training and test data, report the numbers of training and test images. Be sure to include example output figures. Quantitative evaluation is always a big plus (if applicable). If you are working with videos, put example output on YouTube or some other external repository and include links in your report.

3.1 Implementation

4 Discussion and conclusions

Summarize the main insights drawn from your analysis and experiments. You can get a good project grade with mostly negative results, as long as you show evidence of extensive exploration, thoughtfully analyze the causes of your negative results, and discuss potential solutions.

- `conv-1`:
  - $96 \times 11 \times 11$ Convolutional Filter
  - ReLU Activation
  - Batch Normalization
  - 2×2 Max Pooling
- `conv-2`:
  - 2-width Zero Padding
  - $256 \times 5 \times 5$ Convolutional Filter
  - ReLU Activation
  - Batch Normalization
  - 2×2 Max Pooling
- `conv-3`:
  - 1-width Zero Padding
  - $384 \times 3 \times 3$ Convolutional Filter
  - ReLU Activation
- `conv-4`:
  - 1-width Zero Padding
  - $384 \times 3 \times 3$ Convolutional Filter
  - ReLU Activation
- `conv-5`:
  - 1-width Zero Padding
  - $256 \times 3 \times 3$ Convolutional Filter
  - ReLU Activation
  - 2×2 Max Pooling
- `fc6`:
  - 4096-node Fully Connected Layer
  - ReLU Activation
  - 50% Dropout
- `fc7`:
  - 4096-node Fully Connected Layer
  - ReLU Activation
  - 50% Dropout
- `fc8`:
  - 51-node Fully Connected Layer
  - Softmax Activation

Figure 1: Single-stream network layout for depth and color channels respectively. Dropout layers are removed during test time.

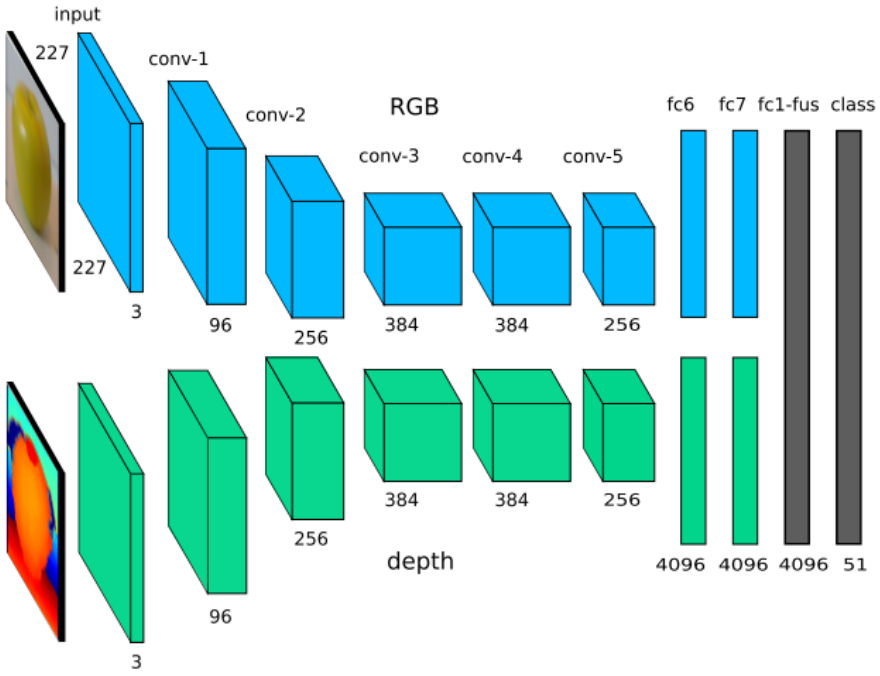


Figure 2: Network architecture using separate color and depth inputs. Inputs for color and depth are $227 \times 227 \times 3$, assuming depth colorization during preprocessing stage. Each modality has its own individual network, with the two resultant fusion layers providing final classification.

(a) Color image (b) Depth image

Figure 3: Original color and depth photos of object captured on a turntable.

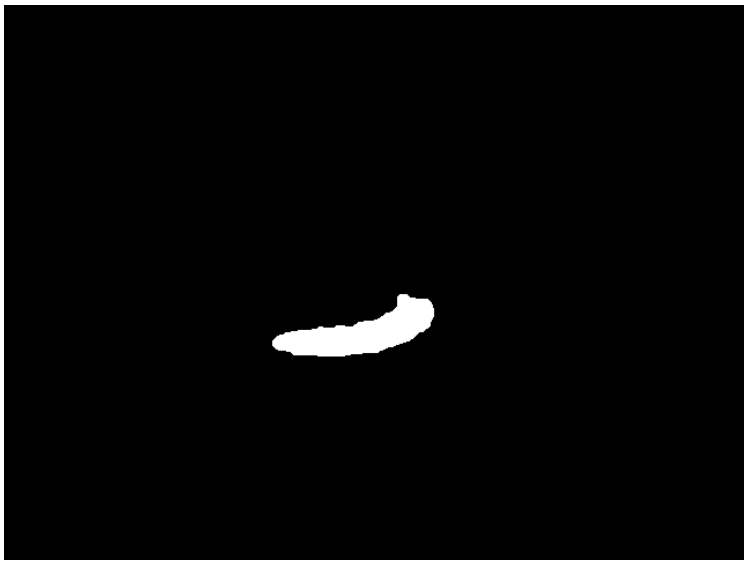
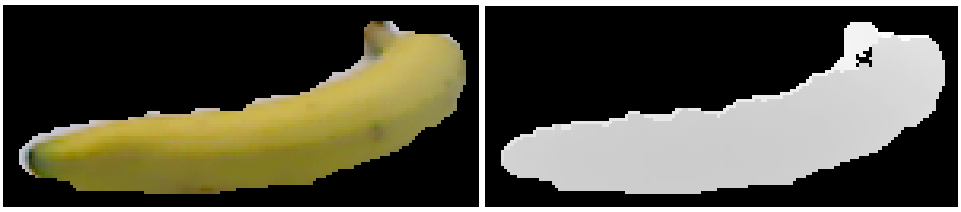
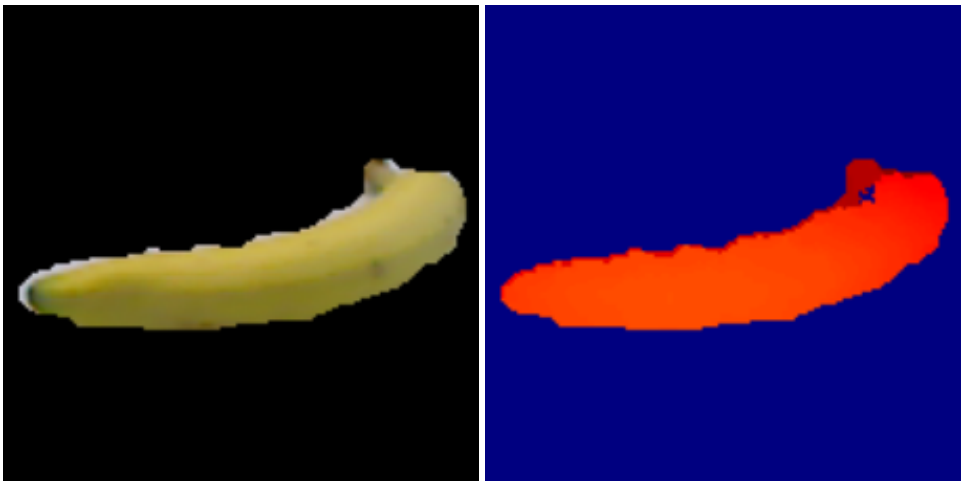


Figure 4: Ground-truth segmentation mask for banana object.



(a) Color image (b) Depth image

Figure 5: Cropped banana photo using given segmentation mask.



(a) Color image (b) Depth image

Figure 6: Rescaled cropped color photo and rescaled cropped depth photo colorized using *Jet* heatmap.

5 Individual contributions

Required if there is more than one group member.

References

[1] B. Newhall, *The History of Photography*. New York: The Museum of Modern Art, 1982.

[2] A. Pham, "E3: Microsoft shows off gesture control technology for xbox 360." `http://latimesblogs.latimes.com/technology/2009/06/microsofte3.html`, June 2009. (visited on 2016-May-10).

[3] A. Eitel, J. T. Springenberg, L. Spinello, M. A. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," *CoRR*, vol. abs/1507.06821, 2015.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.