RGB-D Object Recognition using Deep Learning

Dario Aranguiz Cu-Khoi-Nguyen MAC

***Abstract*—**

I. 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 [todo ref]. 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 [todo ref]. With this launch, consumers and researchers alike were able to view the world in three dimensions, opening up a second modality to image processing.

II. 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* [todo ref]. 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 [todo ref]. In this section, we will discuss our network architecture, our preprocessing phase, and our training methodology.

A. Network Architecture

The network structure proposed by Eitel et al [todo ref] 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 [todo ref]. 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. Lastly, this goes into a final classification layer with softmax activation.

B. Preprocessing

C. Training

III. 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.

A. Implementation

IV. 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

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

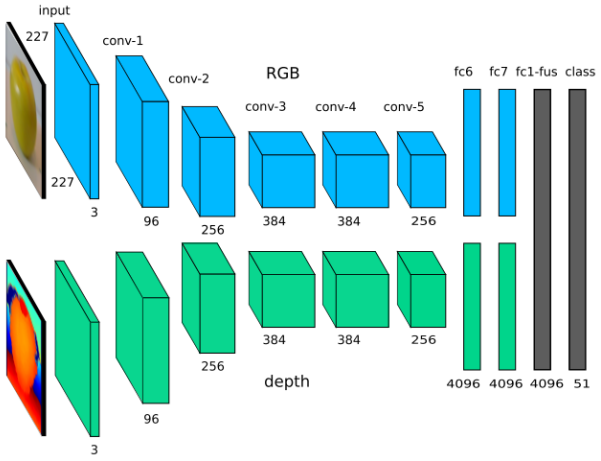

Fig. 2: Network architecture using separate color and depth inputs. Inputs for color and depth are $227 \times 227 \times 3$, assuming depth colorization during preprocessing stage. Each modality has its own individual network, with the two resultant fusion layers providing final classification.

V. Individual Contributions

Required if there is more than one group member. including URLs for any external code or data used.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.