Visuomotor Learning: Object Classification for Generated Data

Dylan Moore (dmoore2) | Payton Broaddus (broaddus) | Justin Kang (jhkang00)

Stanford University | CS230 | March 20, 2018

Introduction

Our project proposal builds on one of the suggested CS230 project ideas, "Sim2Real Visuomotor Learning for Robotic Manipulation," which was to train a convolutional neural network to accomplish end-to-end visuomotor control for a robotic pick and place task using simulated data with domain randomization. We expanded on this goal by delving into how four different images taken at different angles of the simulated data can improve the convolutional neural network's ability to classify and locate the objects. As a result, we managed to achieve a high but descending accuracy after training on 10,000 images from four angles each, identifying typically 8-12 objects out of a total of 55 possible classifications as seen below.

Problem Statement

To label the objects in a simulated environment by building a multi-input convolutional neural network that takes images from different angles to classify the objects for that scene.

Datasest

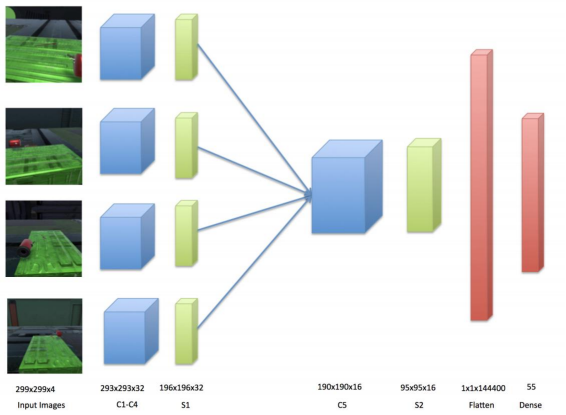
The data set consists of images for each of the 10,000 virtual scenes generated. Each scene contains some of the 55 classifiable objects in total, so each scene can contain anywhere from 0 to 55 classifiable objects in it. Each scene contains 4 perspectives: each perspective has an 299x299RGB and 299x299D image. Thus, our labeled dataset X has dimensions (299x299x4)x4x10000, and each image in each scene has an associated Y, a vector of the image name (string), an array of object classification numbers (numbered 0-54), and actual 6-D data for the objects present in the image.

Feature

Our raw input features are the pixels of each image (299x299), for each of the RGBD (RGB colors and the depth image), multiplied by the four camera angles. We do not have any derived input features. The pixels and the colors are standard inputs for a convolutional neural network, and the number of camera angles enhances classification.

Model

We are using a four-input CNN, which is a modified version of the three-input CNN used in the paper [1]. The first convolutional layers filter four $299 \times 299 \times 4$ input images with 16 kernels of size $7 \times 7 \times 4$ with stride of 1 pixel, and put through pooling layer with the stride of 2 pixels. Then, the four convolutional layers are merged into one and put through the second convolutional layer with 32 kernels of size 3 x 3 x 4 with stride of 1 pixel, and a pooling layer of stride of 4. We then flatten and create a fully connected layer with 55 neurons and use a linear activation for the output layer.



Architecture of the Four-input CNN.

Analysis

In order to verify the performance of the model, we altered the positive weight to a low and high value (0.0001 and 1000) and achieved the expected result of pushing the model to learn to output that no objects are present and all objects are present, respectively. Then we assigned a positive weight of 5, which is the ratio of 0s to 1s, and balanced the two to achieve relative equity between the accuracy and the recall. We tested three models — varying the number of images for each scene to test our hypothesis that we would see improvements in performance as the number of camera angles increased. However, we found the additional images impeded the performance of the model, and the four-input model yielded the least accurate predictions.

Comparison

Model	Metric					
	Train			Eval / Test		
	Acc	Recall	Loss	Acc	Recall	Loss
Validation (0) (Epoch = 1)	0.531	0.469	0.584	0.854	0	0.001
Validation (1) (Epoch = 1)	0.499	0.494	99.95	0.255	1	12.632
One Camera Angle (Epoch = 10)	0.87	0.833	0.525	0.735	0.219	2.41
Two Camera Angles (Epoch = 10)	0.778	0.57	0.7605	0.718	0.2895	1.832
Four Camera Angles (Epoch = 10)	0.686	0.307	0.996	0.701	0.36	1.254

Comparison of the different models on train and eval/test sets.

Conclusion

When we copied the architecture of Sun et al, we expected the performance of the model to improve with additional images. However, our results indicate that for this new task, a deeper neural net architecture would perform better, since additional layers would be better able to represent relationships in the more complex inputs that we have here.

Future implementation would include expanding the output to include the location of each object, which would be more difficult and would fully utilize the four camera angles.

Citation

SUN, Y., ZHU, L., WANG, G. AND ZHAO, F. Multi-Input Convolutional Neural Network for Flower Grading In-text: (Sun et al., 2018) Your Bibliography: Sun, Y., Zhu, L., Wang, G. and Zhao, F. (2018). Multi-Input Convolutional Neural Network for Flower Grading.