CS230

Visuomotor Learning: Object Classification

Dylan Moore, Payton Broaddus, Justin Kang
Department of Computer Science
Stanford University
dmoore2@stanford.edu, broaddus@stanford.edu, jhkang00@stanford.edu

Abstract

Machine learning in object classification is critical to automation of certain industries such as warehouse stocking. We created model for an object classification task in a simulated warehouse environment using a multi-input convolutional neural network, which can take as input any number of images from different perspectives on the same scene. Our project compares three variations on our model: the baseline (a single input image), a model based on Sun et al.[1] and a deeper network of our own design. We observed that the performance of the model grew worse, not better, as the number of angles increased. The deeper models performed better than the simpler ones.

1 Introduction

Improvements in object classification in machine learning has broad ramification for automation of numerous industries and jobs, such as warehouse stocking. It is an active area of research on whether training a model on multiple images at different angles of a scene can improve the accuracy, and if so, by how much and whether it justifies the costs of adding an additional camera.

We tried to answer these questions in our project proposal, which 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. Our hope is that this work will be useful to future researchers and engineers who, for instance, might want to make an informed decision about how many cameras to attach to a robot that will be doing something similar to the pick and place task.

The input of the model is a labeled dataset X with the dimensions (299x299x3)x4x10000, which represents 10000 images of 299 pixels of length and width, multiplied by 3(for RGB colors) and 4(for number of camera angles). We then use a four-input CNN to output a k-hot vector representation of length 55 of all the present objects in the scene.

2 Related work

This project builds upon a growing body of research on the topic of using machine learning for visuomotor control tasks, such as the Amazon Picking Challenge. We drew specific inspiration from several papers that discuss the pros and cons of approaches to training such a model. For instance, Bousmalis et al.'s work, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping" was illuminative for us in understanding the shortcomings of past approaches to creating generated data for a real world problem.

Zhang et al.'s work, "Sim-to-real Transfer of Visuo-motor Policies for Reaching in Clutter: Domain Randomization and Adaptation with Modular Networks," outlines how to integrate real data with generated data for a more effective model. On a similar note, James et al.'s work, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task" discusses several best practices for doing transfer learning in this context. Additionally, we looked to a number of different works for ideas on how to handle multiple images as input.

Ultimately, we decided that the model devised by Sun et al., in "Multi-Input Convolutional Neural Network for Flower Grading," held the most promise, due to the similarity of his task to ours and the recency of the paper (2018).

3 Dataset and Features

We received our dataset from Simon Kalouche, who provided us with 10000 simulated scenes. Each scene had four images representing a photo taken at a different angle, and each image had the dimensions 299x299 pixels. We split the dataset into 8000 scenes in the training set and 2000 scenes in the testing set.

Our raw input features are the pixels of each image (299x299) for each of the RGB colors, multiplied by the four camera angles. Each image (299x299x3) is one of the four inputs for the multi-input CNN.

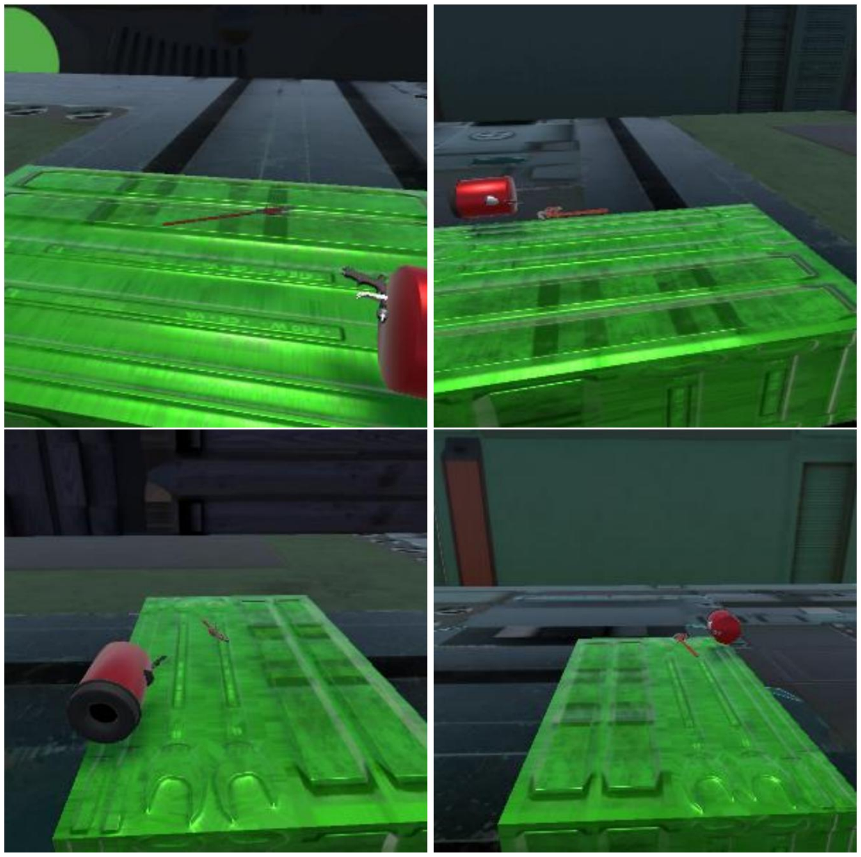


Figure 1: Example of four images of one scene

4 Methods

We are using a four-input CNN, which is a modified version of the three-input CNN used in Sun et al[1]. We initially tried a model as prescribed in the paper: the first convolutional layers filter four 299 x 299 x 3 input images with 16 kernels of size 7 x 7 x 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. However, our NN would only predict "no images" for each scene regardless of the number of the items.

We used a weighted cross entropy loss. The cost function is shown below.

$$Loss = targets * -log(S(logits)) * pos_{weight} + (1 - targets) * -log(1 - S(logits))$$

$$S(x) = \frac{1}{1 + e^{-x}}$$

The cost is similar to sigmoid cross entropy loss, except for the positive weight scalar, which is set as a hyperparameter. If the positive weight is set to 1, it runs and trains exactly as if one had a sigmoid cross entropy loss. For positive weight greater than 1, the higher penalty for assigning 1's as 0's improves recall, at the expense of false 1's. We used high positive weight to persuade our NN to not output only zeros. This method is also used in many 1-hot machine learning problems [1].

Once we had successfully trained the shallow architecture, we implemented a second, deeper model (see picture below for architecture). We duplicated the convolution and pooling layers and added an additional fully connected layer with 128 neurons before the 55-neuron layer. This slightly improved recall, while only slightly decreasing total accuracy.

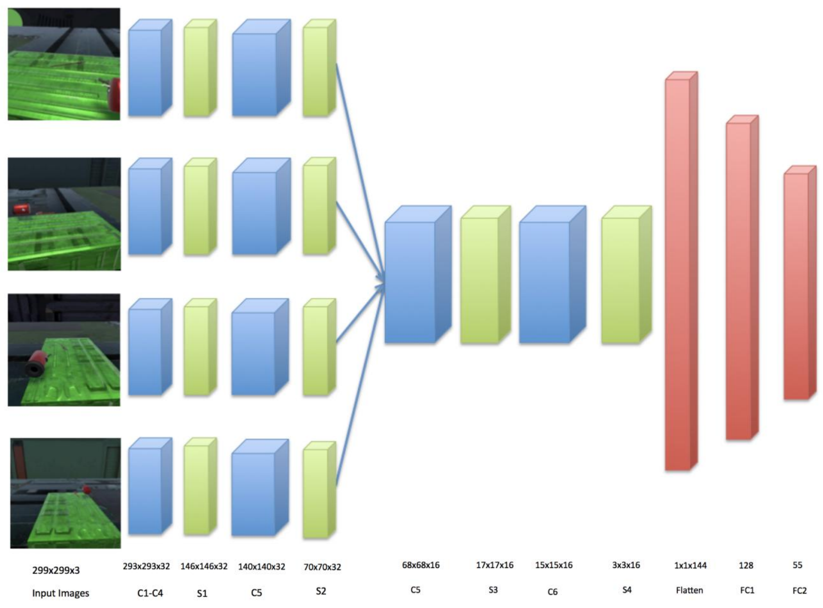


Figure 2: The Architecture of the deeper four-input CNN model

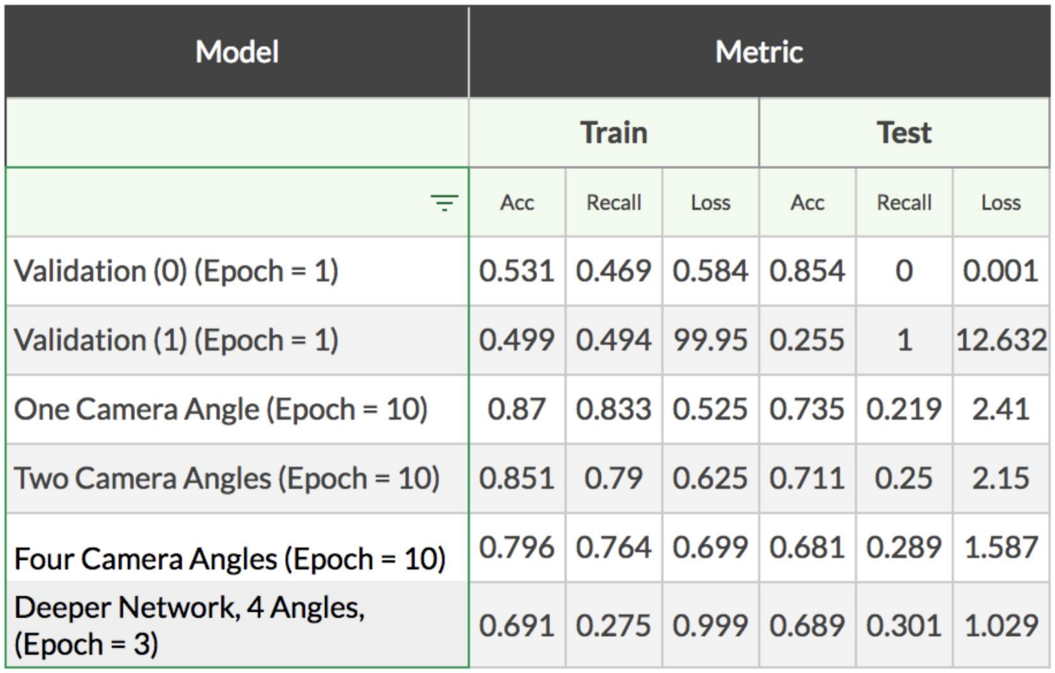
5 Experiments

As explained in the methods section, setting the positive weight (pos weight) scalar to numbers other than 1 would either increase 1's accuracy or increase 0's accuracy. So, 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. We choose to keep the hyperparameters numbers used in the example cs230 vision project given to us. Shown below are the hyperparameters and their values.

- learning rate: 1e-3
- batch size: 32
- num epochs: 10,
- use batch norm: true,
- bn momentum: 0.9,
- image size: 299,
- use random flip: true,
- num labels: 55,
- pos weight: 5

We mainly used recall and accuracy in evaluating the model. Loss was not a good metric between runs, since changing the pos weight parameter could change the loss number significantly even if it didn't change the accuracy or recall. Accuracy is the percent of correct, recall is the percent correct of 1's.

6 Results



Model	Metric					
	Train			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)	0.851	0.79	0.625	0.711	0.25	2.15
Four Camera Angles (Epoch = 10)	0.796	0.764	0.699	0.681	0.289	1.587
Deeper Network, 4 Angles, (Epoch = 3)	0.691	0.275	0.999	0.689	0.301	1.029

Figure 3: Comparison of the different models on train and eval/test sets.

7 Conclusion/Future Work

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, increasing the number of camera angles did not necessarily help. Instead, a deeper neural net architecture performed 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. For future work, a more robust hyper parameter search with a potentially different model could improve performance. In fact, we only changed pos weight in our hyper parameter search, so exploring over other hyper parameters would almost certainly improve results.

8 Code and Data

Github Link: `github.com/dmoore2/cs230_project_repo`

Image Data: Email for data. Simon Kalouche: kalouche@stanford.edu

References

[1] Sun, Y., Zhu, L., Wang, G. and Zhao, F. (2018). Multi-Input Convolutional Neural Network for Flower Grading.

[2] UW Robotics and State Estimation Lab. (2018). PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. [online] Available at: https://rse-lab.cs.washington.edu/projects/posecnn/.

[3] Vision.princeton.edu. (2018). MIT-Princeton at the Amazon Picking Challenge 2016. [online] Available at: http://vision.princeton.edu/projects/2016/apc/

[4] Arxiv.org. (2018). Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task. [online] Available at: https://arxiv.org/pdf/1707.02267.pdf

[5] Amazon Picking Challenge | RoboCup -, www.robocup2016.org/en/events/amazon-picking-challenge/.

[6] Konstantinos Bousmalis et al. "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping." Https://Arxiv.org/Pdf/1709.07857.Pdf.

[7]Zhang, et al. "Sim-to-Real Transfer of Visuo-Motor Policies for Reaching in Clutter: Domain Randomization and Adaptation with Modular Networks." [1709.05746] Sim-to-Real Transfer of Visuo-Motor Policies for Reaching in Clutter: Domain Randomization and Adaptation with Modular Networks, 18 Sept. 2017, arxiv.org/abs/1709.05746.

[8] James, Stephen, et al. "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task." PMLR, 18 Oct. 2017, proceedings.mlr.press/v78/james17a.html.