

Convolutional Neural Networks' Performance on NU Robotics Competition Identification Task

Garrett Thomas

garrettthomas2018@u.northwestern.edu

EECS_349 Machine Learning – Northwestern University

Dataset Collection:

A total of 9 minutes of raw video data from 5 separate environments was collected by recording ROS bag file info using a USB camera plugged into a laptop. These bag files were then replayed and image frames were captured from the playback twice a second, resulting in an initial raw image set of ~1080 images. The different recording environments included the workshop at the Ford Design Center, 2 different areas of a dorm lounge, a dorm room, and a stairwell - all during the afternoon on a sunny day. In order to further diversify the initial dataset, the target objects were moved randomly within the recording area every 30 seconds. Example images from each of the environments are shown in Figure1.

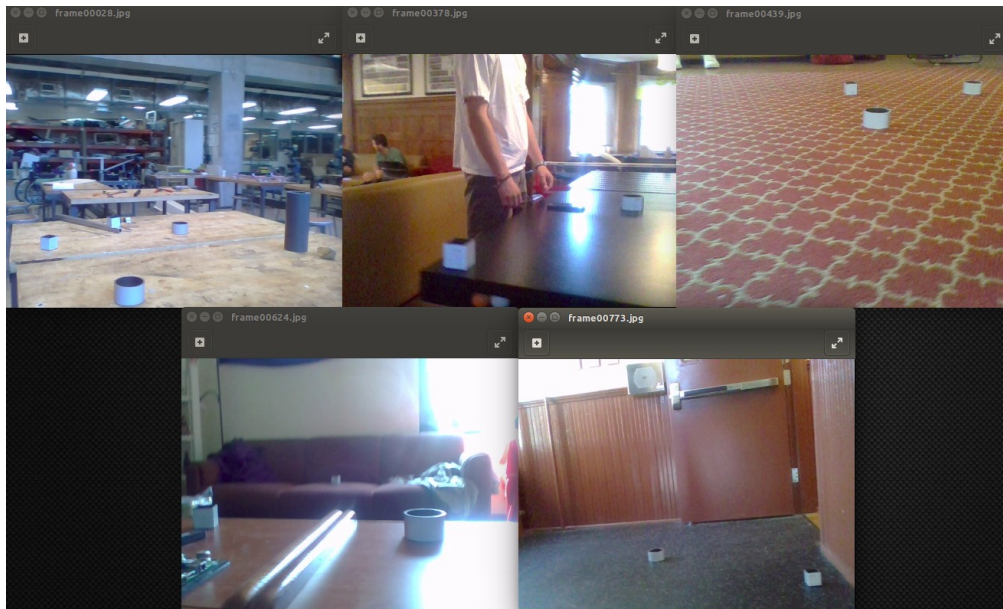


Figure1

Data Preparation:

These 1000+ original images were then annotated using a custom utility script, `image_annotation_tool.py`, which recorded the class (CUBE or CYLINDER) and x, y coordinates and boundaries for user-selected ROIs (Regions Of Interest) into PASCAL VOC style .xml files. Images which were deemed too blurry were left unannotated. Figure2 shows an annotated image and its corresponding .xml file.

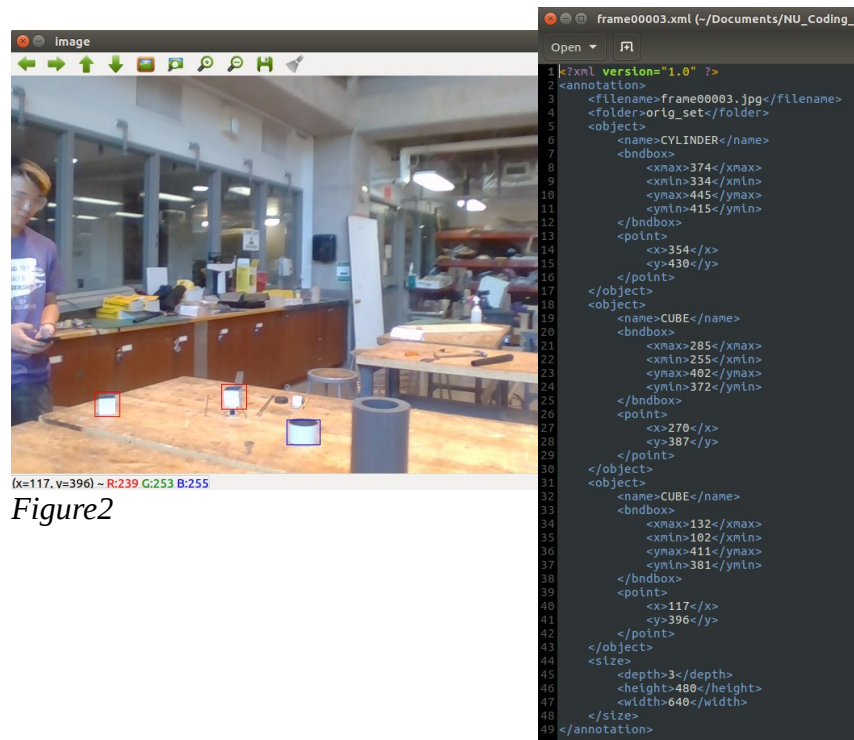


Figure2

Since 1000 images is a very small dataset for training an object detection network, a custom image augmentation script, `image_augmentation_tool.py`, was run on the set of the original images. This python script reads in a file of augmentation methods lists. Each of the methods lists contains instructions to append an augmented version of the original image set to a new dataset which will hold all the augmented data. In the final implementation, the augmentation script was passed a file with 15 methods lists, and our final augmented set contained ~15000 images (1000 original + 14(1000 augmented))¹. Figure3 shows an image from the original dataset next to an augmented version.



Figure3 - Augmented image on right (horiz. flip, perspective transform, HSV jitter)

After the augmentations were complete, the new dataset was cleaned using `clean_dataset.py`. This deleted any image and corresponding annotation files which had no actual annotations associated with them (too blurry or simply didn't contain target objects), and renamed the remaining image and annotation files to be in consecutive order again. After this operation, the dataset contained 13008 annotated images. Using `partition_dataset.py`, the dataset was then shuffled and divided into three parts

¹ Augmentation methods implemented include: vertical and horizontal flipping, random scaling, random translation, random perspective transformations, and HSV color space jittering. These augmentations were combined to produce the 15 methods lists.

– 75% (9756) for training, 15% (1951) for validation, and 10% (1301) for testing. Given that image detection networks are typically trained with even larger datasets, a bit more weight was given to training to initially train a better model.

The models in the TensorFlow Object Detection API read from .record files, and the package came with a create_pascal_tf_record.py script to format PASCAL-style .xml files to .record files. With a little bit of tweaking and removal of unneeded parameters, the script accepted the format of the cube and cylinder dataset, and .record files were created for each of the three data partitions.

Model Training:

Two separate TensorFlow SSD²-Inception_v2³ CNNs – one utilizing dropout, and one without – were trained ~11 hours overnight for a total of 6024 steps with a batch size of 8. After 1000 steps, the dropout network attained a classification loss⁴ of 3.894 and a localization loss⁵ of 1.008, and the non-dropout network had a classification loss of 3.157 and a localization loss of 0.9388. After the full 6024 steps, the dropout network attained a classification loss of 2.750 and a localization loss of 0.5824, and the non-dropout network had a classification loss of 2.621 and a localization loss of 0.7602. Figure4 illustrates below.

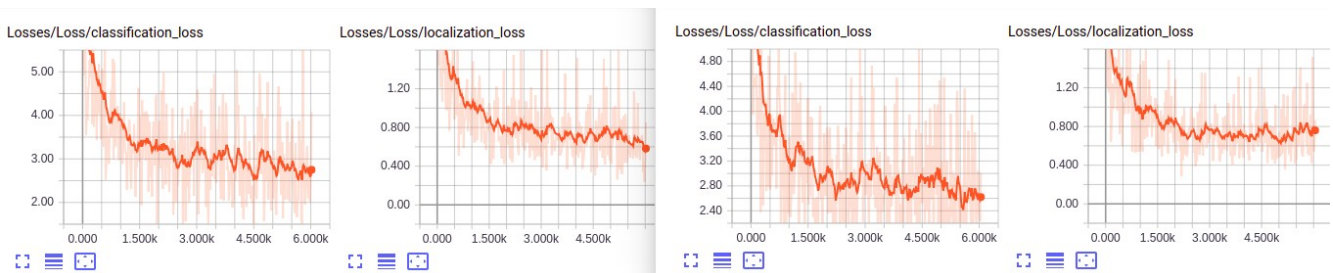


Figure4 - Non-dropout network on left, w/ dropout on right

Model Validation:

When run on 10-fold cross validation, the non-dropout network attained a performance of 78.82% for detecting cubes and 86.85% for cylinders, and the dropout network correctly predicted 74.98% of cubes and 88.17% of cylinders. The avg val. accuracy for the non-dropout network was then 82.84%, and the avg accuracy for the dropout network was 81.58%. Figure5 illustrates below.

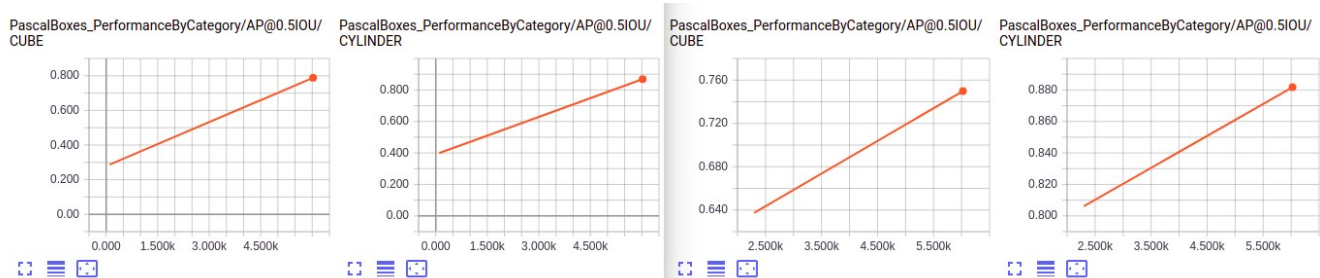


Figure5 - Non-dropout network on left, w/ dropout on right

2 SSD (Single Shot MultiBox Detector) – Discretizes output space of proposal ROIs into a set of default boxes which, at prediction time, are given a score for the presence of target objects and are removed or adjusted to better match the target.

3 Inception_v2 – 2nd version of Google’s ‘Inception Module,’ which dramatically reduces the number of parameters required in the network.

- Both modifications serve to make using the network computationally more feasible for platforms without GPUs.

4 Inverse relation to how well the network is classifying the target objects in the training set.

5 Inverse relation to how well-placed the locations of the predictions on the training set are relative to the correct spot.

Model Testing/Results:

Since the difference between the models' avg accuracy is not statistically significant, both were run on the test data. The non-dropout network attained a performance of 79.32% for detecting cubes and 86.90% for cylinders, and the dropout network correctly predicted 77.63% of cubes and 88.52% of cylinders. The avg test accuracy for the non-dropout network was then 83.11%, and the avg accuracy for the dropout network was 83.08%. Figure6 illustrates below.

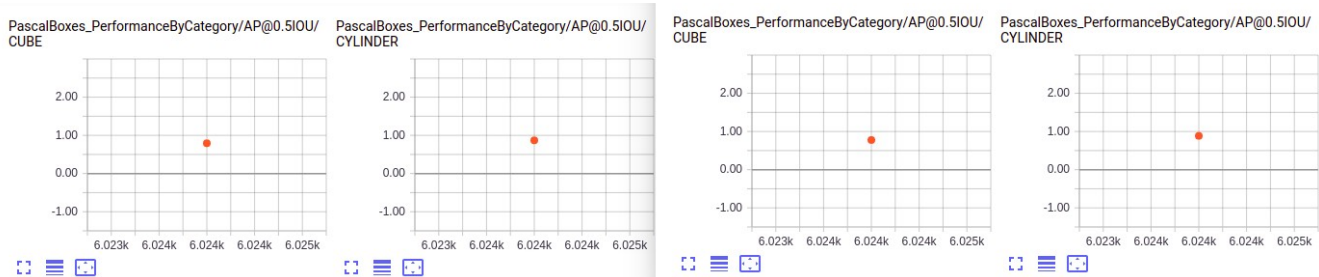


Figure6 – Non-dropout network on left, w/ dropout on right

Analysis:

A typical method of reasoning about a machine learner's performance is by comparing it to ZeroR's performance on the same task. However, this comparison wouldn't mean anything to us, since our task involves localization, as well as classification. Therefore, we are reserved to say that detecting 83% of all target objects in our various environments is 'pretty good,' especially considering the computational constraint of just using a modern laptop for these procedures⁶. Unfortunately, given the similarity in the test results, it's impossible to choose one model over the other.

Dropout in a NN works by omitting a percentage of hidden nodes at each layer to prevent overfitting and to encourage neighboring nodes to more-or-less 'carry their own weight'⁷. Usually utilizing dropout allows the network's performance to converge more quickly, though this was not the case in this study. A possible explanation for this is that the augmentations used were weak enough to allow the non-dropout network to converge on the training data⁸ as quickly as the benefits from dropout allowed the dropout network to converge.

Considering the original question: "Could one of these networks have been used for the NU Robotics Competition this year?" Given the irregularity in the camera perspective of the gathered data compared to the uniform perspective from a robot in the competition, the networks would most certainly perform better than the traditional vision application I implemented, which relied on a floodlight and constant parameter tuning based on outside lighting conditions. With more training data and perhaps tweaked augmentation methods, I'm confident the dropout network would have been able to perform near flawlessly in the competitions environment. The only worry would be computational power on board the robot. There are embedded GPU systems made for this kind of situation, but no tests in this study have been performed to consider the plausibility of using one.

Future Work:

Continuing work on this project would entail testing the network on truly novel data. Although the networks performed well on images they had never seen before from the same environments, we

⁶ Google recommends training their TensorFlow learners for 200,000 steps w/ a batch size of 24.

⁷ No pun intended.

⁸ And also the validation and test data, since the weak augmentations would have allowed a lot of similar images in all three data partitions

don't have an idea whether they would generalize well in a new setting. Regardless if they generalized well or not, the networks would certainly benefit from a more expansive and diverse dataset.

As for my work with CNNs in general, it is exciting to have built this data pipeline, knowing that modifying it for other tasks would be relatively trivial. I actually have an Nvidia Jetson embedded GPU system, and I may attempt to use it to deploy a model in the future.