Robot Soccer

Nate Sampo, Benjamin Ziemann, Max Wei, MJ McMillen

November 8, 2018

The Little Robot that Could

All code referenced in this document and more can be found in the project's git repository and Google Colaboratory Notebook
https://github.com/zneb97/robot_learning
and
https://colab.research.google.com/drive/1sQ8eI3aEmbkcNVClrn354dc$_O r7kCWpl$

1 Overview

Throughout the process we refined our goal as we learned more about the scoping of machine learning project. Initially we wanted to move a soccer ball into a goal, before moving to a more realistically scoped project of recognizing a soccer ball and driving to bump it.

The system looked like this:
Setup

1. Capture base images
2. Generate training data based on generated images
3. Train the model (see below) to output angle relative to the ball

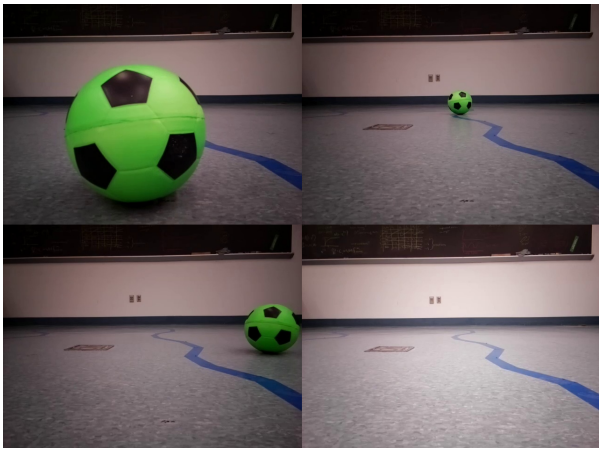
Running on the robot

1. Capture current image
2. Use image with model to get the desired angle to turn to the ball
3. Turn to face the ball
4. Drive forward to bump the ball

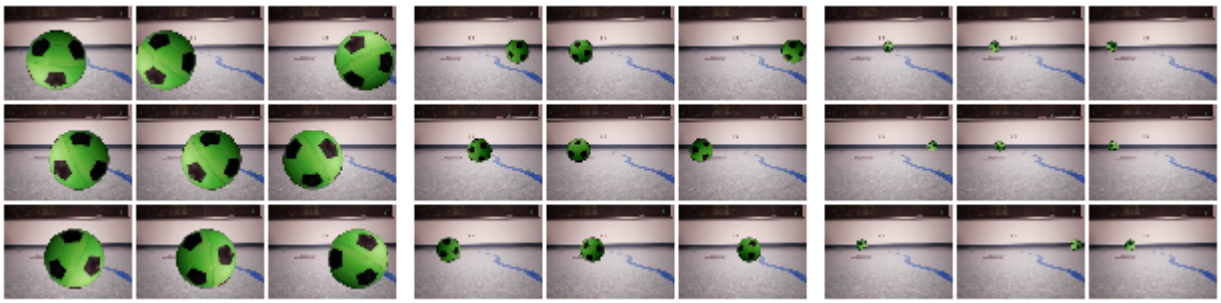
2 Dataset

Understanding that labelled training data for this project would be difficult to come across and may not map well to the images the Neato takes, we knew that we should generate our own dataset of training and validation images, and label the images as we went. To this end, we used Adobe Photoshop to snip a ball out of a Raspberry Pi camera image and created a series of test images where we randomly placed, resized, and rotated the ball over a background image of the AC109 classroom with no balls in front of the camera.

After creating some test images and comparing them to real images, we realized the ball size and appearance in frame changes drastically depending on the ball's distance relative to the camera. This meant that while we could freely translate the ball along the X axis, we could not do the same for the Y axis, since moving the ball farther away significantly altered its appearance. To counteract this, we constructed a small image set of three images from the Raspberry Pi camera attached to the Neato, with an image of a close, mid-range, and far soccer ball. We again snipped out the ball in each image and wrote a script to translate the ball along only the X-axis of the empty classroom image in random orientations (in an attempt to prevent over-fitting on the ball's pattern). Finally the size of these balls and their locations were written into a .csv file to be read as the labels for our training and validation data.



Soccer Ball test images from which others were generated

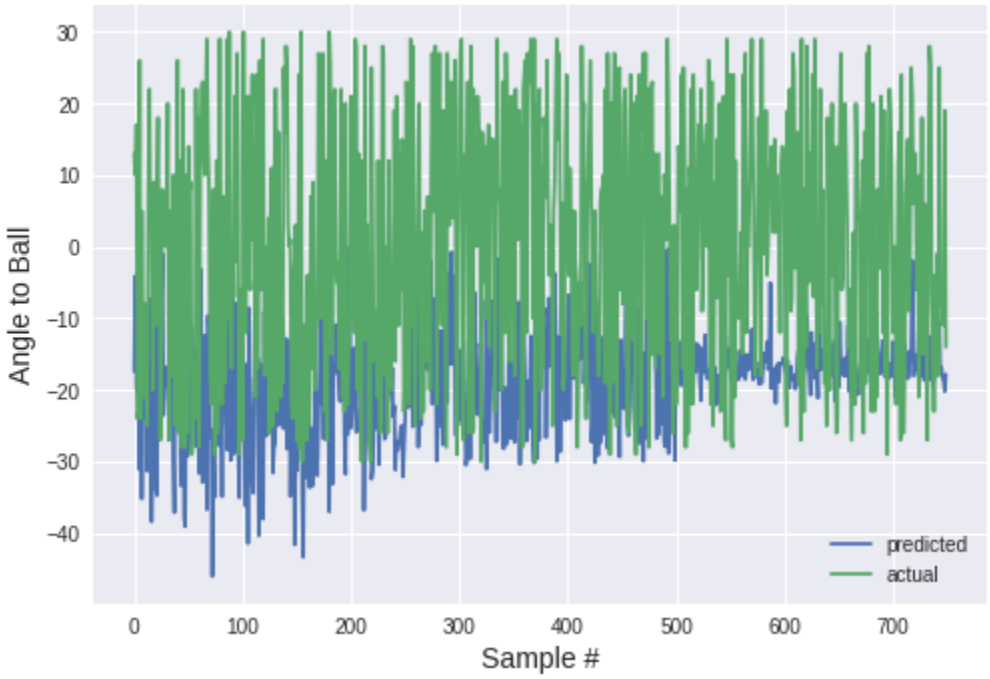


Several generated training and validation images

3 Algorithm

We tried two different machine learning algorithms. We first attempted to train a logistic regression model using Scikit learn to determine only the angle between the ball and the robot. We then built a convolutional neural network using Keras to determine the bounding box of the ball.

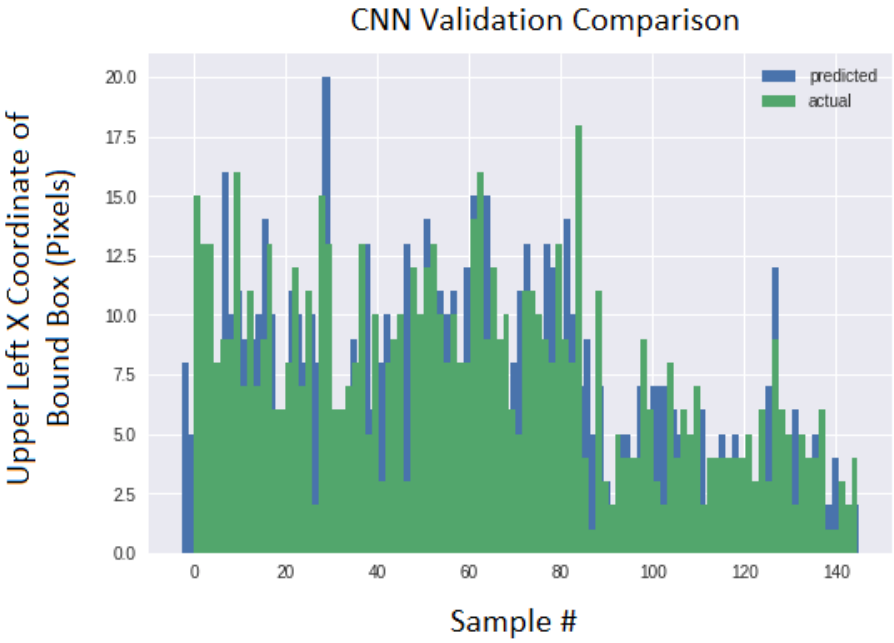
The first pass was an approach much closer to that of what we did in class with a logistic regression model taking in images and outputting a theta value representative of the ball's angle relative to the robot.



Scikit logistic regression model validation comparison

This was largely unsuccessful, and the data predicted by the model had almost no correlation to the real data we took. This lead us to our second pass.

The second pass model is a convolution model through Keras which instead outputs the upper left and lower right x and y coordinates (in pixels) of the bounding of box of the ball in a given an image. The model is made up of alternating 2D convolutions and pooling. By putting the image through multiple layers, as well as using convolutions to more intuitively link parts of the image together, meaning possibly taking the ball as a whole as opposed to just individual pixels, we hoped to achieve a more accurate result. This improvement was seen in our the graph of predicted vs actual values for the top-left x coordinate seen in the figure below.



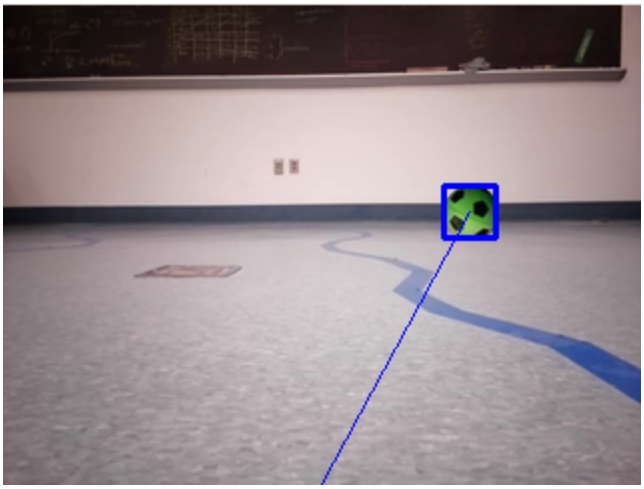
Keras model validation comparison

Both of these models were created in Google Colaboratory notebooks and then saved and downloaded in separate files for local usage. The prepping of the dataset and training of both of these models can be seen in this Google Colaboratory notebook:
https://colab.research.google.com/drive/1sQ8eI3aEmbkcNVClrn354dc$_Or7kCWpl$

Bounding box formed by the output of our model

4 Non Machine Learning Approach

Our non-machine learning approach was based on the method we used to generate one of our datasets. Instead of reading the edited images, we take the input from a pi camera and run a Gaussian blur, erode, color mask filter, and then blob detection to identify the soccer ball. Once the soccer ball was identified in a picture, we could obtain the coordinates of the ball center and radius. Using this information in conjunction with knowledge on optical physics and the Pi Camera's field of view, we can then calculate the approximate angle and distance of the ball from the Pi Camera.



Bounding box and distance formed by our non machine learning algorithm approach

Additional factors, like color (we used very brightly colored soccer balls) or the pentagon patterning, could be added to the workflow to increase the accuracy of ball tracking.

5 Design Decisions

Throughout our project, we had to constantly refine our goal based on our progress level and knowledge. The original goal was to move a soccer ball into a goal seemed well scoped, but to achieve this we needed to break down the project into multiple components. These components are: identifying/labelling a soccer ball, determining the position of the soccer ball, and then figuring out the strike position.

Of the different ways to identify a soccer ball, we decided to run a base train logistic regression on the generated dataset using the position of the ball as the feature. Since there were three different ball distances in the generated dataset, three quarters of each distance set was used to train the model, while the last quarter was used for validation.

Because there is only one kind of soccer ball used, we decided object classification was unnecessary. Since we wanted the bounding box of the ball as our output, object localization was the best course of action. We limited our scope to one bright green soccer ball and took a pictures of it at various distances and angles. We then took the soccer ball and moved it around the photo using Adobe Photoshop. We rotated the ball and moved it laterally so the size of the ball relative to the distance was maintained. Through various rotations and transformations we ended up with 3,000 photos. We then split the photos up into two groups. Seventy-five percent of the data was used for training the model. The remaining twenty-five percent was reserved for validation.

We built and tested two models. The first model uses logistic regression and Scikit Learn to predict the angle between the robot and the center of the ball. The Keras model uses a convolutional neural network to determine the bounding box around the ball. We decided to test both methods because we did not know which one would produce the better model and wanted experience with setting up different types of models for future use. Fortunately while shaping data was a time sink and a problem, we only needed to solve it once and having a second model did not take away from the effort we put forth in the other.

6 Reflections

6.1 Challenges

One of the largest challenges we faced was working with Google Colaboratory and Numpy to process our data efficiently. After spending a lot of time trying to get Google to recognize correct file paths, there was a caching problem that prevented our updated files from being read. Additionally once we had the data in, formatting it in the correct shape for the models took a lot more work through learning Numpy and trying to conceptualize what this data actually looked like index by index. While it definitely was part of the learning curve to using Numpy and we did in the end get a better intuition and understanding for it, it ultimately ended up eating hours of time we would rather have spent exploring the intricacies of the models and machine learning.

Having a larger team, making sure everyone was on the same page proved difficult at the start and cost us some time. Once we got going though we agreed on communication methods and meeting times, overall communication improved and productivity increased. Meeting more often contributed to more progress, as well as helping to make sure everyone was aware on the current progress of the project.

6.2 Improvements

Improvements we could make to this project would be one of our original plans to have the robot push the ball into a goal. This probably would have taken the form of the robot learning to align the goal and ball in frame then driving forward.

We also could have created a machine learning model (mainly the dataset to train it) to explicitly put out a command velocity as opposed to just finding the angle and using odometry to turn. The dataset would probably have been generated by a script such that the robot would turn a set number of degrees with PI control behind it controlling the angular command velocities, recording the pi camera image and command velocity at the same time.

6.3 Lessons Learned

Using the Google Colaboratory was nice for allowing us all to write code at once rather than git pull/pushing. However because the training of the model depended on pulling data from Google Drive, us having different paths to the relevant data made more than one person running the code difficult without restructuring our individual drives.

Having a larger team, we learned how important it was that everyone was on the same page. There was a few times in the beginning we all thought we knew what the plan was but upon articulating it we found that we had different ideas of what that plan was. Good communication was key and by the end we had gotten much better at it.