Boobot: The shy yet insidious Neato

Nick Steelman

November 1, 2018

Figure 1: Photo of the sneaky vacuum

All code referenced in this document and more can be found in the project's git repo:
https://github.com/CleanestMink126/robot_learning

Also links to the Boobot in action can be found here:
https://www.youtube.com/watch?v=Y1jJRKBl6GU
https://www.youtube.com/watch?v=CC9o__TzhSg

1 Overview

The goal of this project was to develop experience in integration of machine learning algorithms into a robotic system, specifically to solve some sort of robotic problem. The direction I chose to take this was in the form of a slight twist on person following, where the robot (called a Neato) will only move when the subject's back is turned. For the uninitiated, this is the primary characteristic of the "Boo" from Mario which will follow Mario whenever his back is turned. This method is implemented using only data from a camera affixed to the front of the Neato.

2 Implementation

This challenge breaks down into two subproblems, person tracking and gaze detection.

2.1 Person Tracking

Person Tracking was the first problem I implemented as it is used to effectively train the gaze detection. The main idea is to get a good segmentation of a single person inside the scope of the attached camera. Then, use these bounding boxes along with the assumption that the center of the camera pointing towards the direction of the robot to create a proportional control loop dictating how fast the Neato should drive and turn.

Since the problem of human bounding box segmentation is a well studied and implemented problem, I decided to use a pre-trained model and focus my attention on the gaze aspect. I used a Tensorflow Zoo Mobilenet model (formally SSD mobilenet V1 COCO 2017/11/17) for human pose detection to get bounding boxes. This model worked perfectly for the purposes of the project, effectively segmenting the human pose in any reasonable frame.

There was an amount of fine tuning done in the proportional control such that the Neato would move and turn the right speed as to not lose the subject.

2.2 Gaze Detection

With the person tracking done, I moved onto stop detection. I initially considered with the idea of using facial detection to determine when to stop, but this was abandoned for reasons discussed in the next section. I decided instead to train my own since the task is fairly specific and to aid my own learning.

The first step to developing an effective machine learning solution is to gather appropriate data. For this I took pictures with the Neato's camera of my bounding box in different poses, clothing, lighting, and places. This problem provides a convenient method of data collection, as I just had the Neato take pictures as it followed me around the room. Using this method I was able to get 3,500 moderately diverse training photos (shown in Figure 2) in a couple hours. I also recorded 900 testing photos on a different day for effective evaluation.

Figure 2: Example of the training data with facing the Neato examples (top) and facing away examples (bottom).

For inference, I implemented a Convolutional Neural Network (CNN) in Tensorflow. A visualization of its convolutional output can be found in Figure 3.

Architecture: For the architecture I used two layers of 32 and 64 filters maps respectively. These were created with filters of size 3x3 which had 2x2 strides in order to reduce dimensionality. The output of these convolutions is then flattened and fed through a fully connected layer of size 100 with a training dropout drop rate of .5 to act as regularization. After applying a Relu activation function on this layer, it is fully connected to an output layer of size 2 representing facing towards versus facing away. This output is then fed through the softmax function to determine a final probability of each.

Training: From the output the cost function is calculated through standard cross entropy. This is then minimized through Adaptive Momentum (Adam) Optimization with a rate of 0.5 and learning rate of 1e-3. The network was trained for 2,000-4,000 iterations with a batch size of 12. By the end of training it had converged to a accuracy of about **92%**.

Considerations: Considering the complexity of the data provided, it is surprising that the size of the network is so small. I experimented with larger and deeper convolutions as well as a larger hidden layer size but found that all resulted in overfitting and at least a 5% reduction in accuracy. Additionally, using lower values of the drop probability in Dropout also resulted in significant overfitting.

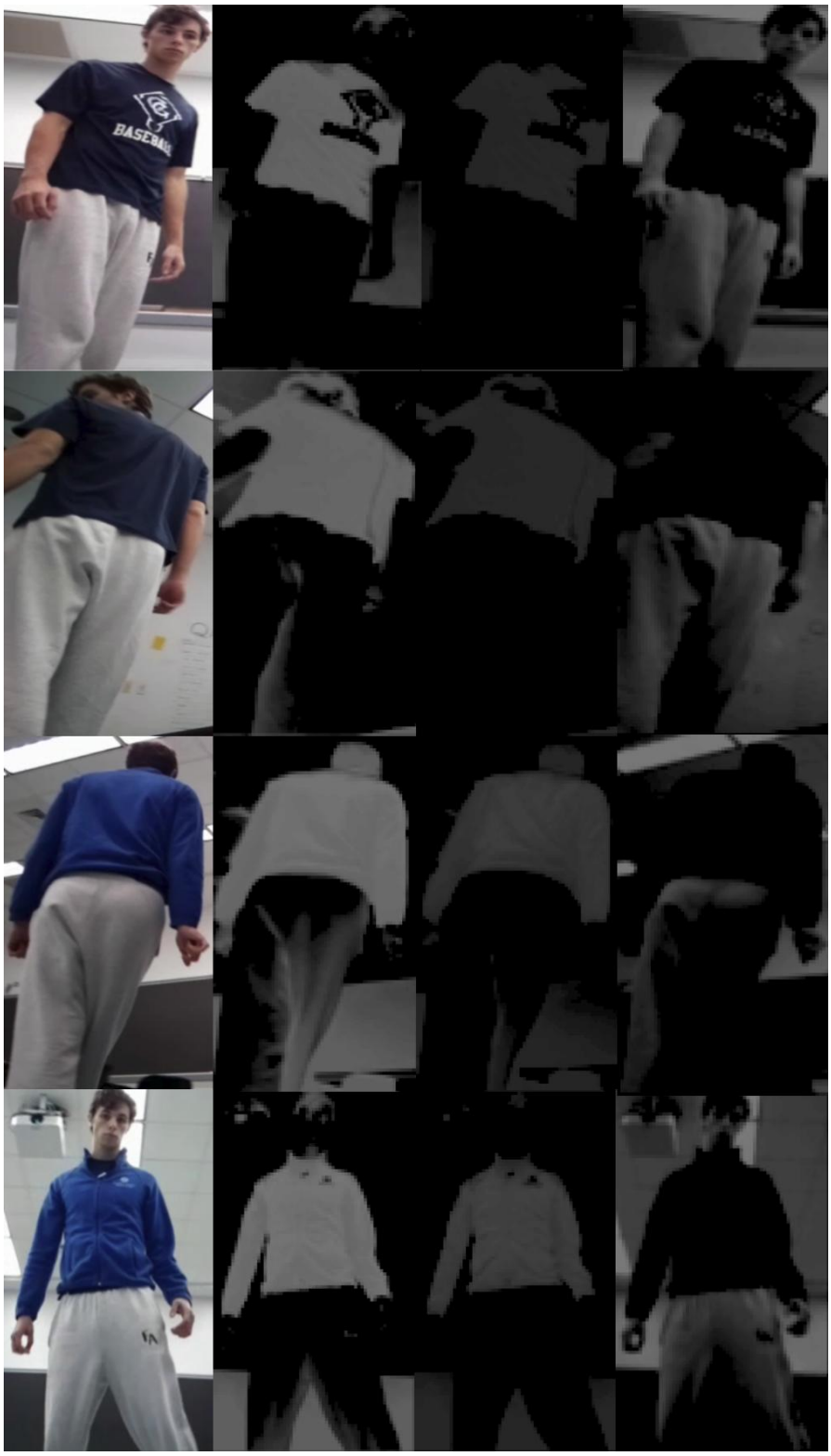

Figure 3: The convolutional output of some of the more prominant feature maps (tanh activated and brightness adjusted for visualization). Notice some learned aspects that could be more easily interpreted for inference, such as the dots for eyes in the first map and the presence of hair at the top of the second map.

3 Face Detection Comparison

As mentioned, originally I had planned on using facial detection to determine whether or not a person is facing the Neato. Although I went the other route, I decided to still implement a classical facial detection algorithm to compare the results. For this example, I used the Haarcascade Frontal Face Default Model included in the Opencv library. Haar methods include the same idea as convolutions but instead to choose to not learn the convolutions but use some helpful preset filters in inference.

For the fairest comparison, I tested both methods with the parameters I would use in a run time scenario. Since we care more about the Neato stopping when the subject is looking then it going when the subject is not looking, we want the model to be particularly sensitive to facing features. This means in the Haar cascade setting the facial detection to a sensitive setting (i.e. only requiring two boxes to detect a face) , and in my model it means stopping the Neato for facing confidence values above 0.3 (which does decrease the total accuracy in exchange for a better true negative rate).

Method	True Positive	True Negative	Total Accuracy
Haar Cascade	**97.2**%	70.6%	80.1%
Lightweight CNN (Mine)	84.9%	**95.3%**	**88.9%**

Figure 4: Comparison of Classical vs Deep Learning Methods

As shown in Figure 4, the results are not unanimous. The Haar classifier does have a substantially better True Positive rate, meaning that the Neato will rarely stop when the subject's back is turned. This is compared to the CNN method which will sometimes predict facing and cause the Neato to suddenly stop while the subject's back is turned.

However, this is a minor consideration for this specific task. The more important aspect is that the robot does not move when the subject is moving, in which case my model performs much better than more classical techniques. The 70.6% accuracy on negative cases means the Haar classifier will often not stop when the subject turns around, which is not surprising considering the blurriness, face occlusion and poor quality of some of the facial data as shown in Figure 5. In this case, my model's 95.3% negative accuracy provides a reliable model for inference on live data, as shown qualitatively in the videos provided.

Figure 5: Example of the Haar Classifier working (left)and two example failure cases (middle and right).

4 Reflections

4.1 Challenges

Surprisingly, the progression of this project was relatively easy and natural to implement. The most major challenges I faced were deciding how to implement gaze detection after getting a working person following. At the time it seemed like a larger challenge since pre-trained models would not fufill my learning goals for the project and the data sets around gaze detection were poorly suited for this problem since they focus on segmentation and rarely involve null examples. I was intially worried about the challenge of collecting data for this problem, but having a already working person follower made the collection process significantly easier.

4.2 Improvements

With more time I would both try and improve my model and use a more advanced model for comparison. For improving my model I would need to collect more, diverse training data to help keep the model from so easily over fitting. This would allow me to build a larger, more advanced neural net that could achieve a higher accuracy. In retrospect, the Haar classifier is somewhat outdated and better suited for cleaner data. For a more fair comparison, I would expand on this by comparing my method to that of a modern facial detection algorithm that was trained to deal with messy and occluded data.

4.3 Lessons Learned

One **major** aspect that I believe is often overlooked and incredibly important is spending time on problem definition and scoping. For this project I spent at least a couple days just ideating before

writing any code. Making myself spend extra time deciding if a project was appropriately scoped and fully fleshing out the entire pipeline was imperative and led to the entire process being quick, logical, and easy to execute.

Another thing I learned is data collection is not as scary as I thought. I figured (like ImageNet) I would need millions of images to build a reasonable model but found that I could build a workable model with just 3500 and a couple hours work.

Overall, I believe that this project was a great learning experience that allowed me to view and implement my knowledge about machine learning through the lens of a robotics in a successful way.