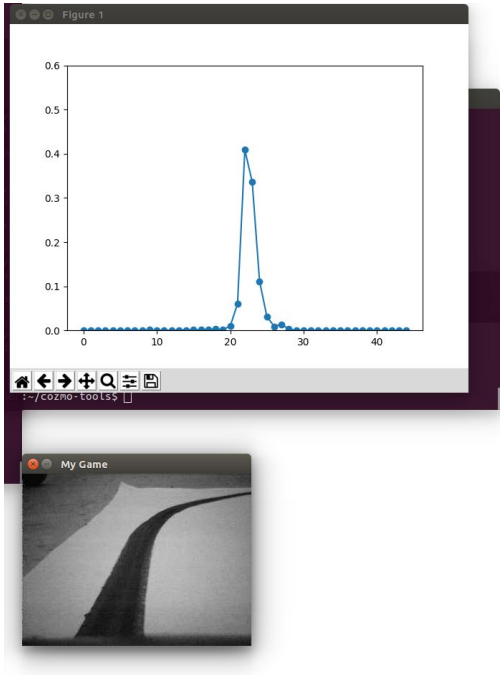
Self-Driving Cozmo

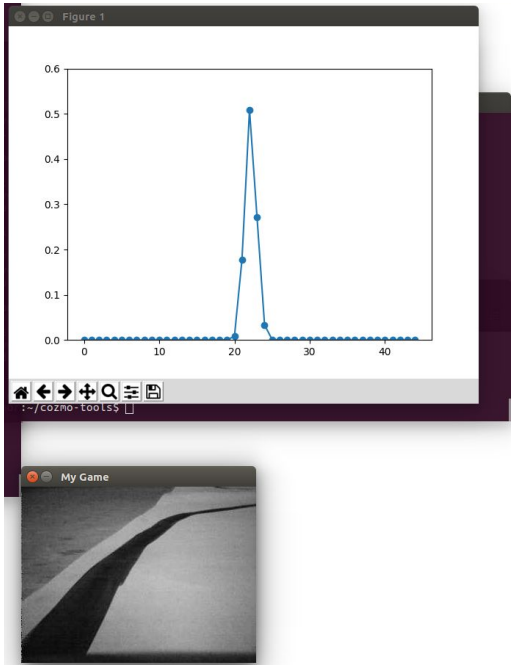
Jason Ma & Kristin Yin

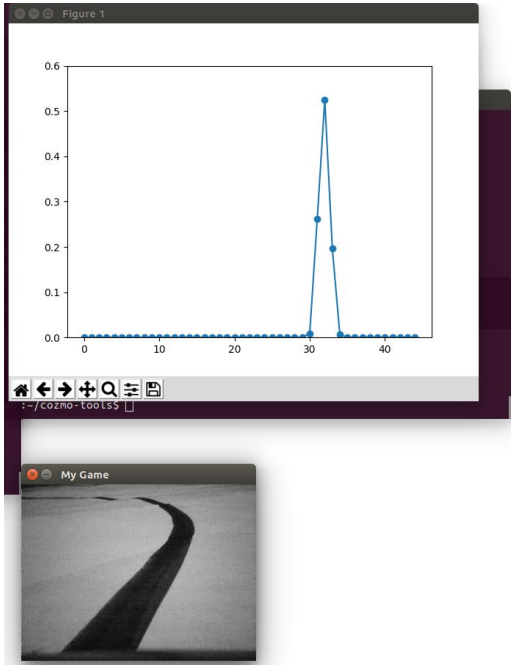
Overview

- Trained Cozmo to follow a line by training camera images through a Convolutional Neural Net
- Training images are labeled with steering angles by manually driving the robot on a course
- Tried a couple of different ways to train the model, including augmenting training images and trying different combinations of numbers of convolutional and fully connected layers
- The model with the best results has 5 convolutional layers and 4 fully connected layers with ~5000 images

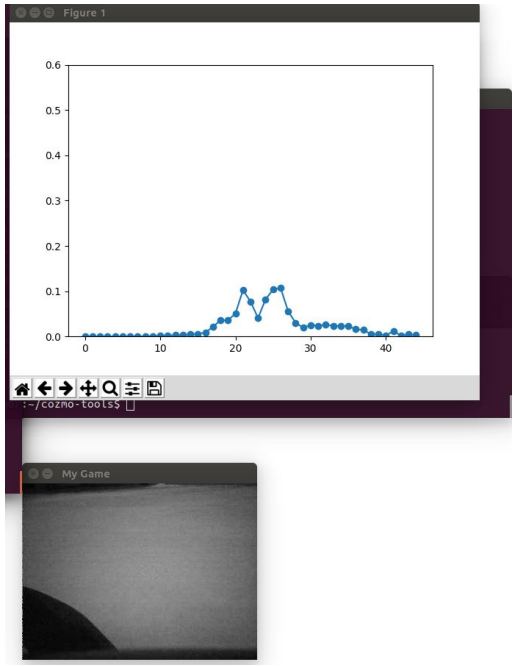
Probability Distribution: Confident

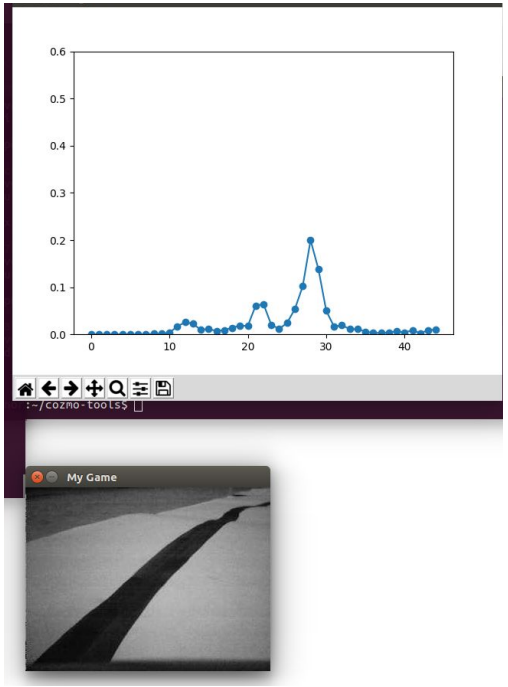


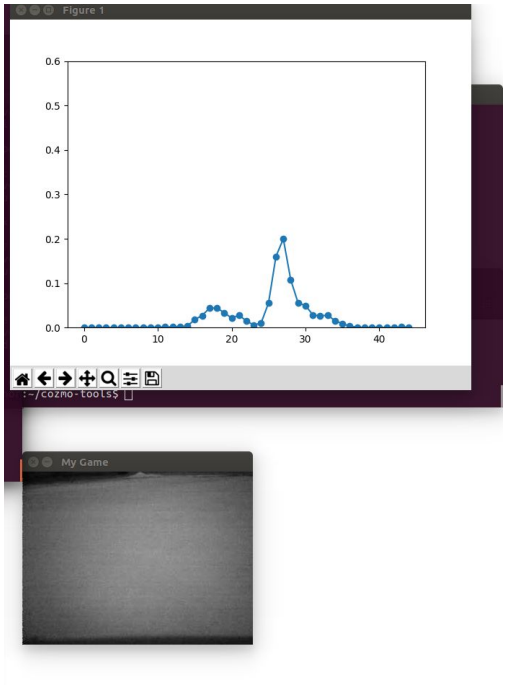




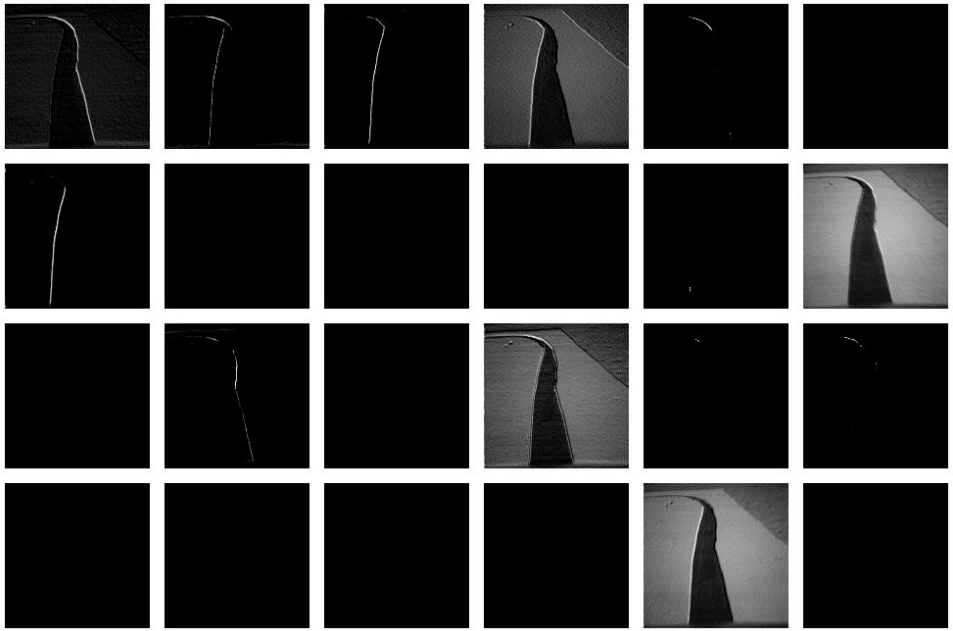
Probability Distribution: Not Confident







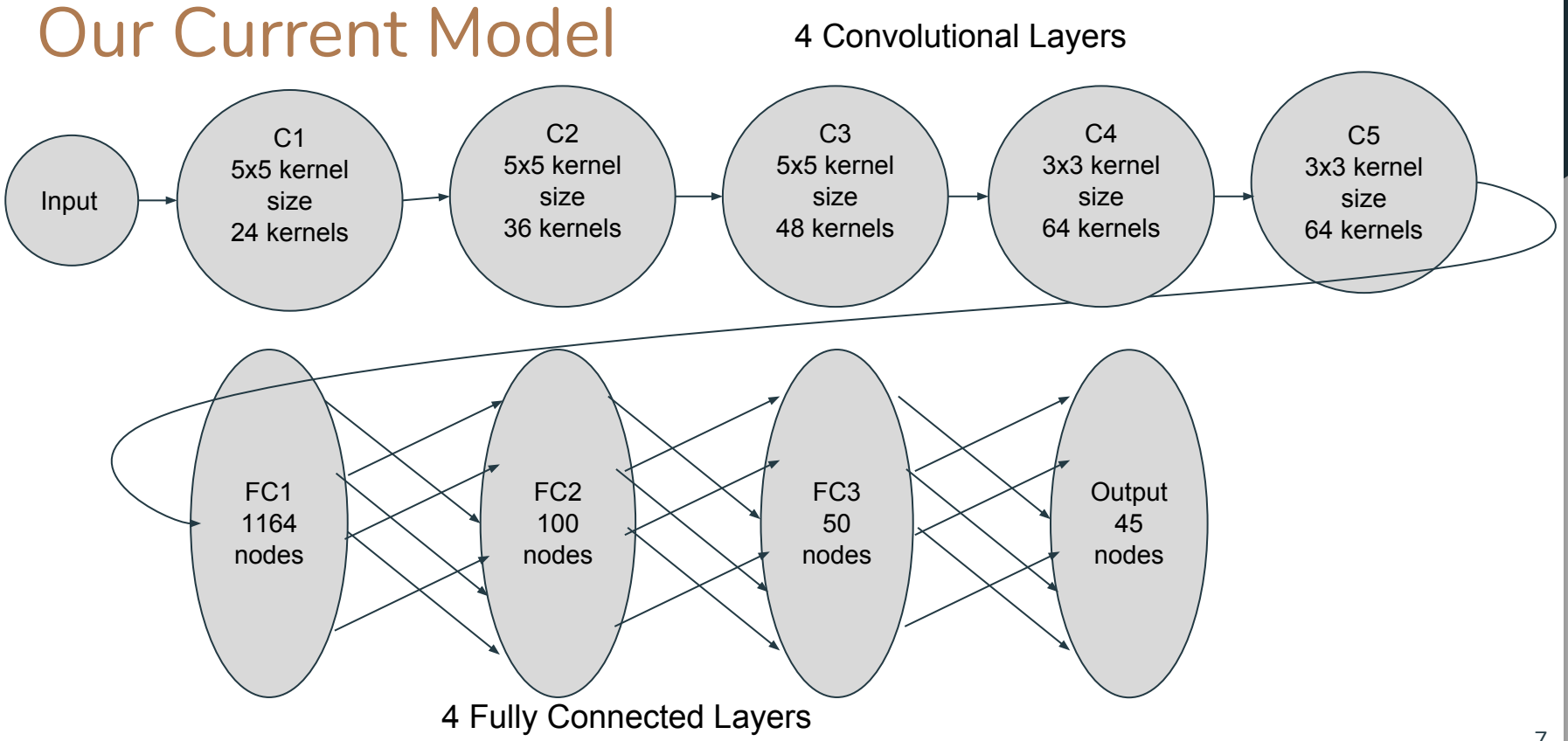
Weights learned in 1st Convolutional layer



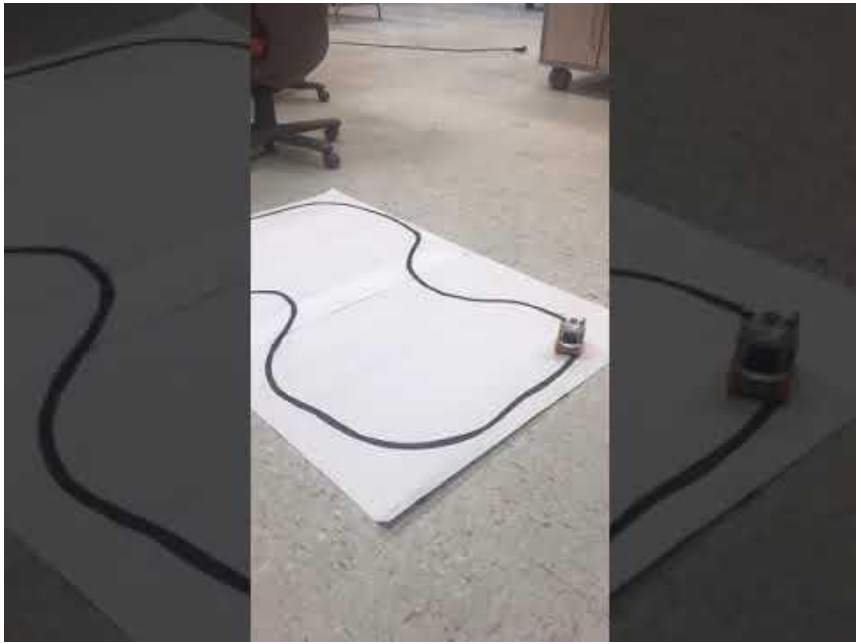
Training & Processing

- We were able to train up the model using 120x160 images, instead of our original 300x300. However, we were running into weird cozmo(?) bugs where the camera would somehow get really delayed, even though the only code we change was reshaping the matrix to fit the new model.
- It took around 30 minutes to train 50 epochs
- Training size was around 5000 images
- Processing time took 0.33 sec.

Our Current Model



Self-Driving Cozmo Demo (Time-Lapse)



https://youtu.be/MXsPVDGYFY8

Challenges: Running the Robot

- We performed several optimisations with regards to how we actually run our neural net.
- Our RunCNN.fsm script has two parallel threads running at the same time.
- One thread runs the cnn and updates a globally shared weight variable w.
- The second thread reads in the most current weight variable, and it performs a steering command based on w.
- By doing so, cozmo is able to effectively run without any noticeable jitters.
- The cnn on average updates around 3 times per second.

Challenges: Correcting Itself

- Since we only train cozmo “perfectly” on the course, it never learns how to correct itself if it were to drive off the course
  - Our naive implementation was to check for a confidence level as it drives and if it is lower a threshold, it would go into “recovery mode”.
  - In this mode, the cozmo will backup until its confidence level is above the threshold again.

Challenges: Training the CNN

- We tried many different ways to try improving the performance
  - Augmenting the images: offsetting and flipping each training image to create more training data (didn’t improve performance)
  - Using a gaussian as the target output pattern instead of one hot (improved performance)
  - Trying different combinations of numbers of convolutional layers and fully connected layers (didn’t improve performance)

Interesting Findings

- Our naive implementation of “recovery mode” surprisingly worked really well, and manages to get the cozmo back on track almost every time.
- We thought that since our training set was not very large, reducing the number of convolutional or fully connected layers would improve the performance, but it turns out that the performance was still the best with more layers.
- Performed decently well even when given an anonymous new track, a track that the cozmo has never been trained on that included more variety of turns.

Future Extensions

- Improving the performance
  - we can analyze the weights of each layer of the neural net to see what cozmo learned
- Let cozmo identify and drive on forks and intersections
- Let cozmo drive faster
- Identify obstacles or obstructions