Behavioral Cloning Project

Overview

This project for a model that predicts steering wheel angle from a front-facing vehicle camera.

- The input data for the model is recorded drive sessions from a driving simulator
- The simulator captures images from left-front, center and right-front facing cameras, mounted on the simulation vehicle. In addition, other parameters such as steering wheel angle and throttle position are also captured.
- Simulator data was used to train and validate a Keras-Tensorflow convolutional network model
- The model was tested with the special 'autonomous' mode of the simulator where the model 'drives' the vehicle by providing steering input in response to center camera images.

The rest of the document describes the files submitted; the model architecture along with the training strategy used; and the final model architecture and performance results.

Files Submitted & Code Quality

The project includes the following files:

- model.py the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 – the trained convolutional network model
- run1.mp4 – recording of the final model driving Track 1
- run2.mp4 – recording of the final model driving Track 2
- A version of this report in PDF format

Code Quality

The model.py contains fully functional code in terms of creating a trained model from input data. The only change required is to set the folder where the training data resides. The code is modular and well-commented. The entire file can be submitted - after setting the training data folder - to get a trained model as output.

The code does not use a generator for data batching as the training environment on which the model was trained had sufficient resources to cope with the 30K-40K images used for training.

Model Architecture and Training Strategy

The general steps followed to create a fully trained and functional model were as follows;

- Run the simulator and record initial batch of training data for visualization and understanding
- Create a simple, flat and dense model to test end-to-end execution of:
    - Recording the training data
    - Training a model from the data
    - Executing the model in autonomous mode in the simulator using the provided drive.py file.
- Verify that end-to-end flow works and all python packages are available and correctly loaded
- Use the Nvidia architecture as the base- model and build upon that
- The Adam optimizer was used so the learning rate was not tuned manually
- Drive two laps for both tracks – one in each direction to record base line data
- Drive slowly on curves to capture more training data for difficult conditions
- Drive from the curb towards the center to record training data for steering correction, if the vehicle goes off-course
- Train a base-line model that at-least responds in terms of emitting non-constant steering wheel angle values base on input
    - Initially only the center camera image was used for training. This caused the model to emit constant steering wheel angle
    - The Nvidia trick of using all 3 cameras with corrected angles of +/- 0.20 for the left and right cameras, was then employed
    - This resulted in a model that responded with varying angles in response to input
- Test the model in autonomous mode and note the problem areas
- Record more test data for areas where the model did not perform well. It was expected that by giving the model more training samples for problem areas would improve model performance. The car was driven slowly in such cases to generate more samples.
- Tweak the model parameter and layers to get better model
    - Experimented by adding Keras dropout after different layers to reduce overfitting and getting a more robust model
    - Also experimented with Convolution layer regularization to improve performance
- Visualize the training and validation losses to pick the optimal number of epochs for balancing good accuracy with training time

Final Model Architecture

The final model is based on the Nvidia architecture adjust for input size and with dropout and regularization added. See Figure 1 for the visualization of the model.

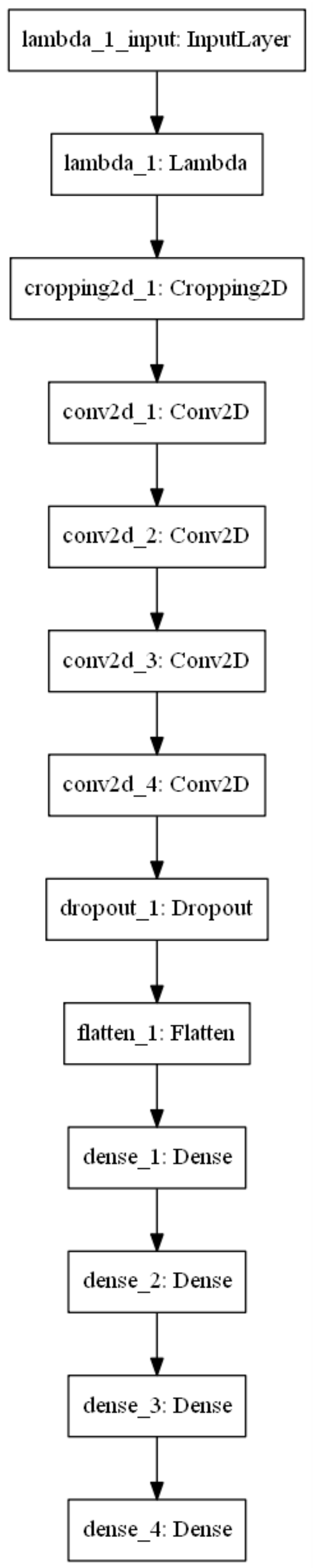
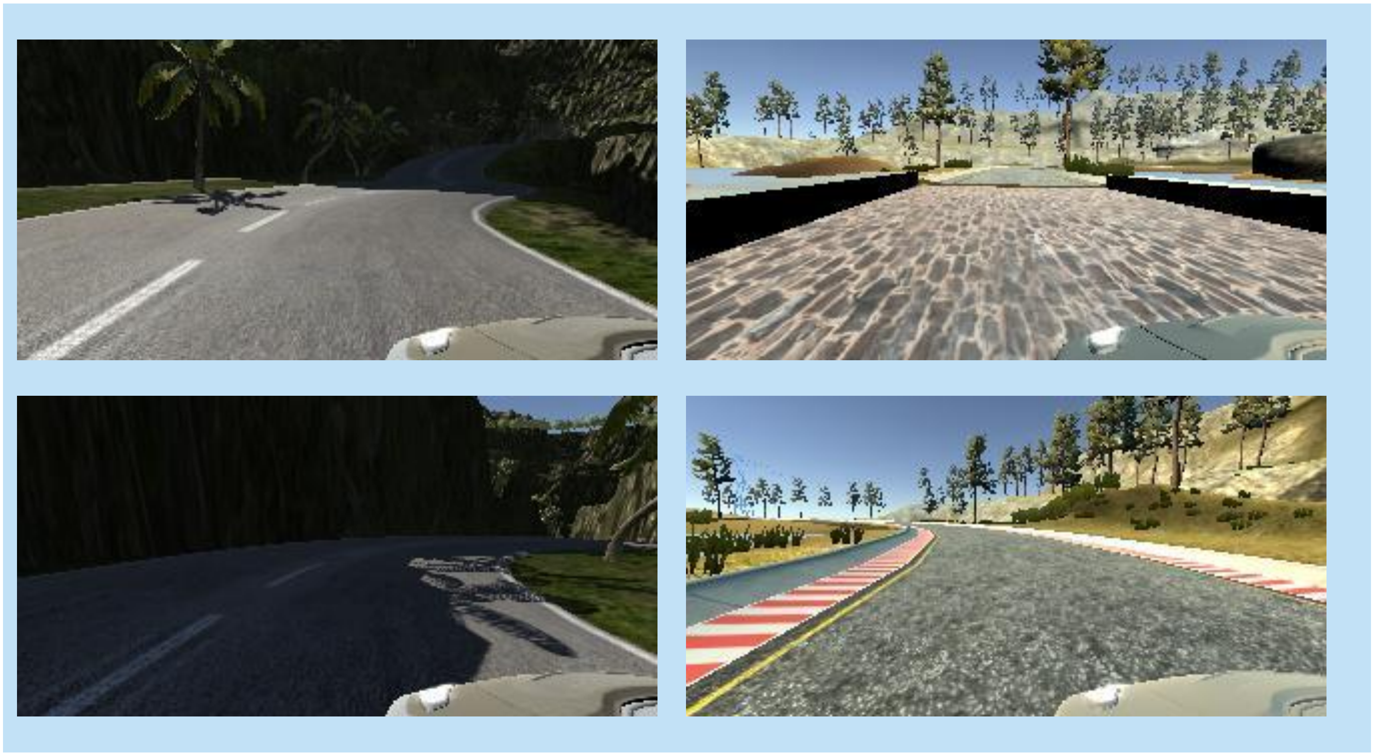


Figure 1: Model Structure

Here are some example images from the training from multiple cameras and both tracks:



After all training runs, there were a total of 37K images available for training. The train/validation split was 80/20. The training and validation losses were 0.2% and 12% respectively.

The final model works well on Track 1 as indicated by the accompanying video. The car successfully negotiates all sections without driving up on the curb or getting stuck off road.

The model also generally performs well on Track 2 (the twisty, mountainous track). The model can drive the vehicle completely around the track. The simulator was run continuously for more than 40 minutes without the car getting stuck. A YouTube video of one lap is available here: https://youtu.be/pibQJUNxFKs.