Write Up for Project 3: Behavioral Cloning

The steps taken to complete this project were as follows:

- Load the training data (*provided by Udacity*) consisting of 320x160 images of the road ahead as seen by on board cameras of the car in the simulator
- Pre-process the training images
- Build a convolutional neural network in Keras to accept pre-processed images and predict the steering wheel angle (*regression*).
- Train and validate the model by splitting the original training data into a training set and a validation set.
- Deploy the trained model on the simulator to successfully drive the virtual vehicle around Track 1 in autonomous mode.

The specific points on the evaluation rubric were addressed as follows:

- The submission includes a model.py file, drive.py, model.h5, a writeup report and video.mp4.
    - All 5 files mentioned were submitted as part of the zip folder. However, model.py was submitted in the form of an iPython notebook and is therefore named model.ipynb.
- The model provided can be used to successfully operate the simulation.
    - The model provided can be successfully run as python drive.py model.h5 in a terminal. Given that the simulator is running in autonomous mode, predicted steering wheel angles will continuously stream on the terminal and the vehicle on the simulator will move forward.
- The code in model.py uses a Python generator, if needed, to generate data for training rather than storing the training data in memory. The model.py code is clearly organized and comments are included where needed.
    - The code in model.py does not use a Python generator because the data does not take too long to load.
    - The code in model.ipynb is clearly organized into appropriate cells with relevant comments.
- The neural network uses convolution layers with appropriate filter sizes. Layers exist to introduce nonlinearity into the model. The data is normalized in the model.
    - The convolutional neural network in this assignment was inspired by the extremely small network discussed by Mengxi Wu (link).
    - First, the training data from all 3 cameras was loaded. An offset of 0.3 was applied on the steering wheel angle for images from the left camera and an offset of -0.3 was applied on the steering wheel angle for images from the right camera.
    - The network relied on a pre-processing step implemented before each image was passed to the network.
        - The pre-processing step consisted of converting each RGB image into the HSV color space, extracting the S channel, cropping the upper 50 pixels of the

image to remove irrelevant parts and then resizing the image to a tenth of its original dimensions to 32x11.
        - This pre-processing step was done outside the Keras layer. It was therefore also implemented in the drive.py file so as to pre-process images being streamed from the simulator in the same fashion.
    - The pre-processing step was followed by data augmentation. The data was augmented as follows:
        - Every image was flipped and appended to the training data.
        - The steering wheel angle corresponding to each flipped image was taken to be the negative value of the original steering wheel angle.
        - This process doubled the amount of training images used.
    - The network itself consisted of the following layers in order in Keras:
        - Normalization layer that normalized all pixel values between -1 and 1
        - Convolutional layer with two 1x1 kernels with 'same' border mode and 'ReLU' activation
        - Max Pooling layer with a size of (4, 4) and a stride of (4, 4)
        - Drop-out layer with 30% drop rate
        - Fully connected layer
        - One steering wheel angle value
    - The number of epochs used was 25. Because of the small size of the network, the training was performed on a standard CPU as opposed to a GPU and took less than a minute.
- Train/validation/test splits have been used, and the model uses dropout layers or other methods to reduce overfitting.
    - The training data was split into a training set and a validation set with in a 9:1 ratio inside the model.fit command in Keras.
    - A drop-out layer with a 30% drop rate was inserted after the only Max pooling layer in the network to guard against over-fitting.
- Learning rate parameters are chosen with explanation, or an Adam optimizer is used.
    - An Adap optimizer was used in the network. The learning rate was not modified manually.
- Training data has been chosen to induce the desired behavior in the simulation (i.e. keeping the car on the track).
    - The training data chosen was the data set provided by Udacity for this specific project.
    - Data collected by manually driving around the simulator was then appended to the original data set to train the model on especially difficult sections of the track and to teach it to recover from extreme steering wheel angles.
- No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle).

- As observed in the video.mp4 file, the vehicle in the simulator receives its steering wheel angle and throttle values from the drive.py file and never leaves the drivable surface of the track at any point.

- Solution Design Approach
  - The convolutional network was first trained with only the training data provided by Udacity (center, left and right images).
  - While the vehicle moved reasonably well, it invariably went off track at the location depicted in the screenshot below. The short segment to the right where there is a gap in lane marking confused the vehicle.

  - This issue was rectified by manually collecting training data at specifically that location.
  - This process was further improved by also deliberately driving the vehicle toward the brown sand section off the road and then rapidly re-centering it back on the road as it almost went off track. This was done to teach the car to recover.
  - The total number of images used in the final training was 26,466. This included images from all 3 cameras and included images from the original dataset as well as images saved during manual training.