Use Deep Learning to Clone Driving Behavior

Data Collection

Three kinds of runs were performed using the simulator to collect diverse data:

1. Forward or Clockwise motion – The car was driven along the road within the lane. This drive consisted mostly of straight or right-curving roads. A few example images from this run are stored in the example directory.
   The texture of the bridge was different. To collect sufficient samples with that particular texture, the car was driven multiple times back and forth over the bridge. Only the forward motion on the bridge was recorded.
2. Reverse or anticlockwise motion – The car first made a 180 degree turn which was not recorded. Subsequently, the car completed the loop. This drive consisted mostly of straight or left-curving roads.
3. Fixes to get back – In order to train the car to get back to the center of the road, this run mainly recorded the car moving from left/right corner of the road back to the middle.

Data Augmentation

The data was augmented using two techniques:

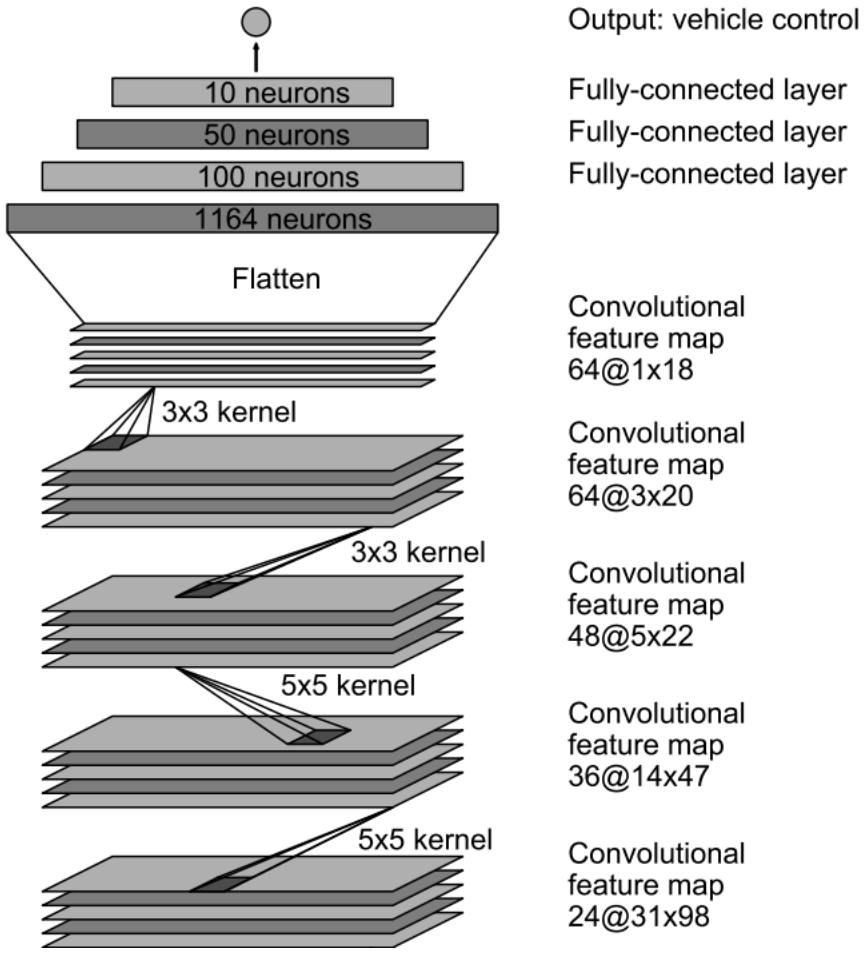
1. The images captured in the center camera was flipped and represented as a separate entry of data.
2. The left and right camera images were used with a 0.15 offset to the angle

Using these two methods, each entry in the csv file resulted in four entries of data –

a. Central camera image
b. Central camera image flipped
c. Left Camera image with offset angle
d. Right Camera image with offset angle

The DNN Model

The DNN in the Nvidia Autonomous Driving Vehicle paper that was discussed in the class was used as a starting point. However, during the training, it was observed that the training set loss went down substantially but the validation set loss did not. This indicated "overfitting". To fix this, dropout layers were introduced in between the fully connected layers in the model. The image was cropped and data was normalized around zero mean before it was passed to the DNN.



Training and Python Generator
The training data could not fit into the GPU RAM. So, a python generator was used to read in the image files one batch at a time. There were under fitting issues with a batch size of 32. A batch size of 8 could fit in the memory and also did not exhibit under-fitting issues. The validation loss tapered off around the 4th or 5th epoch. So it was an epoch of 5 was chosen for the training.

Future Work
Perhaps there is a effective way to pre process the image using RGB/Gray/HSR color manipulation. This approach was not explored and could potentially improve the results.