Introduction:

The main idea of this project is to implement a Deep learning-based control method for self-driving car. Rather than building a neural network from scratch, we adopted a technique called transfer learning – which means, taking a network that was successful in solving a similar problem and building on top of it as needed. For this project, we adopted well-tested NVIDIA architecture published on arXiv.

The architecture:

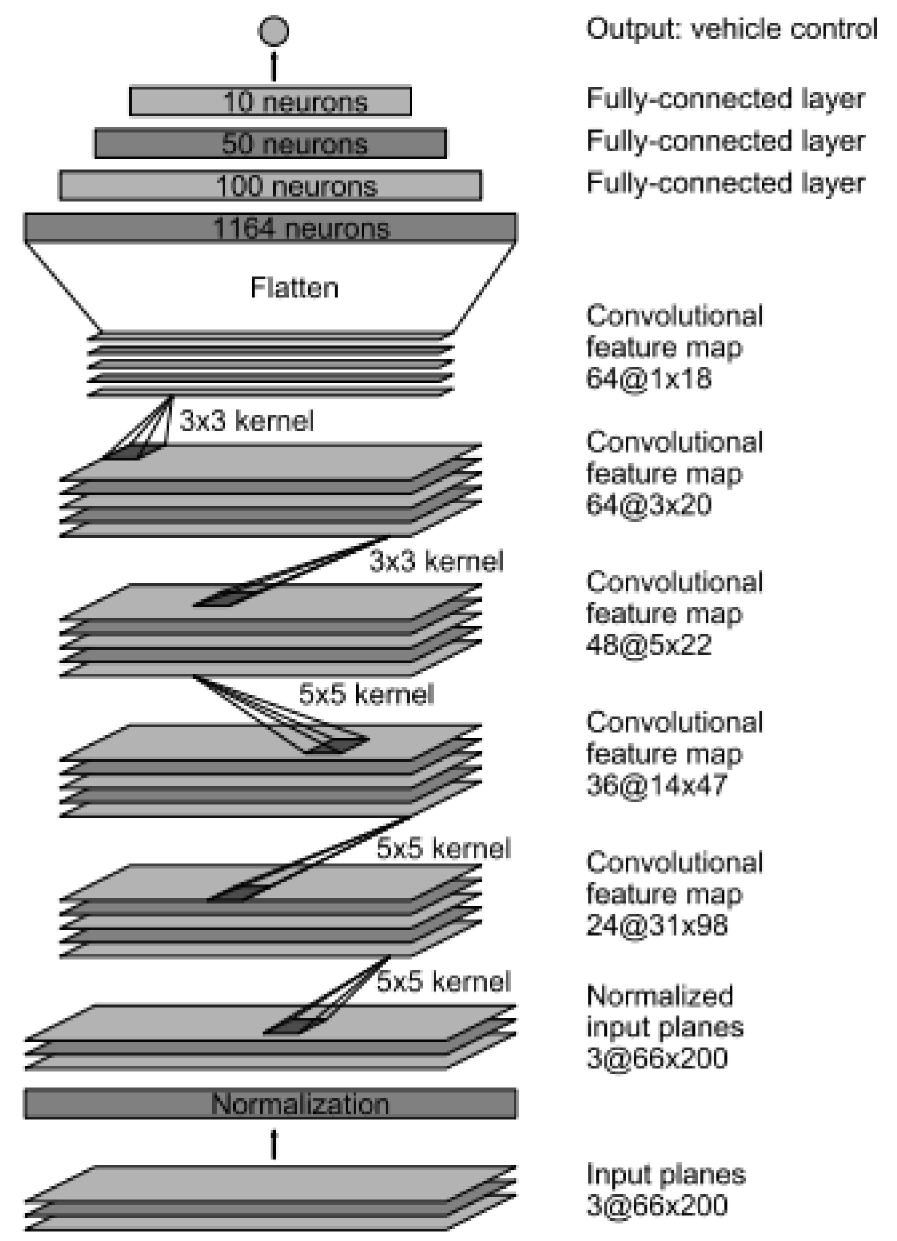
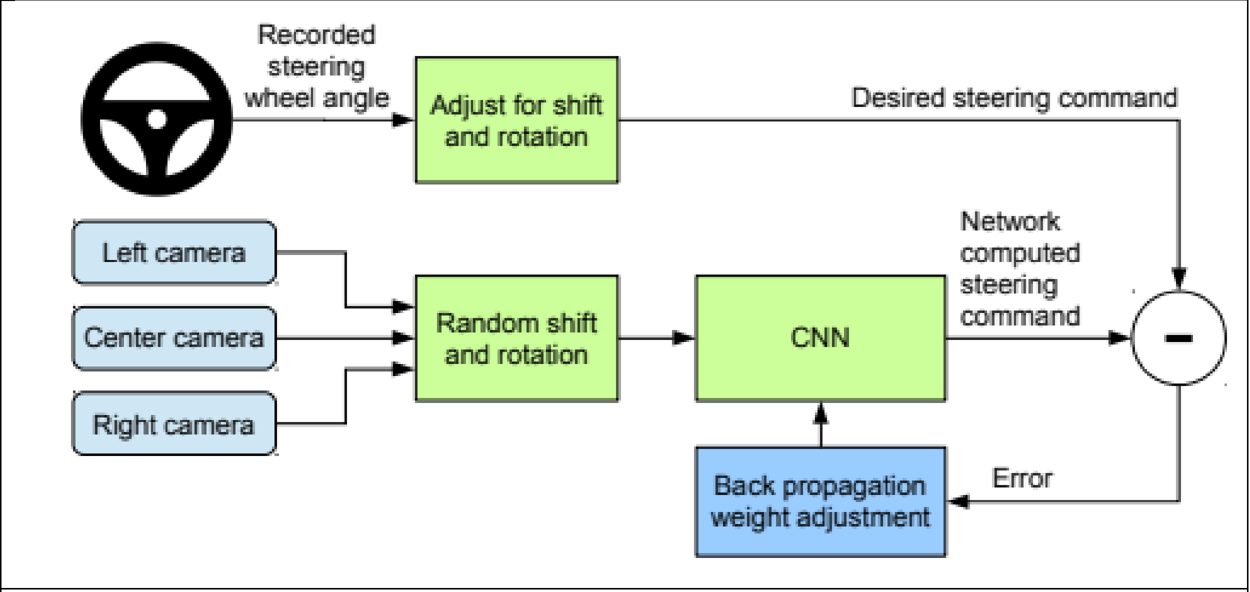


Figure shows the implemented NVIDIA architecture for behavior cloning.

The architecture above takes in a 3channel color images, normalizes them before feeding them to the network. The network has 5 convolution layers (3 5x5 and 2 3x3 kernels) as shown. This is followed by funneling the output down to one neuron (just the steering angle) using three dense layers. Parameters of the neural network is self-explanatory in the code.

Training:
The architecture discussed in the previous section was implemented using Keras framework. Trained over 5 epochs, loss function is mean square error and used ADAM optimizer. Steps were taken to split data to work with and dropout layer was used so as to make sure that the model doesn't memorize the data (Training and validation ratio of 1:5).


Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
−
Back propagation weight adjustment
Error
Figure shows the training procedure.

The training procedure uses three images (from left, center and right cameras) and the corresponding steering angle that was applied to keep the car on track (When using left and right images, the steering angle was corrected accordingly). I also augmented the center image by flipping it so that the network is robust.

Control:

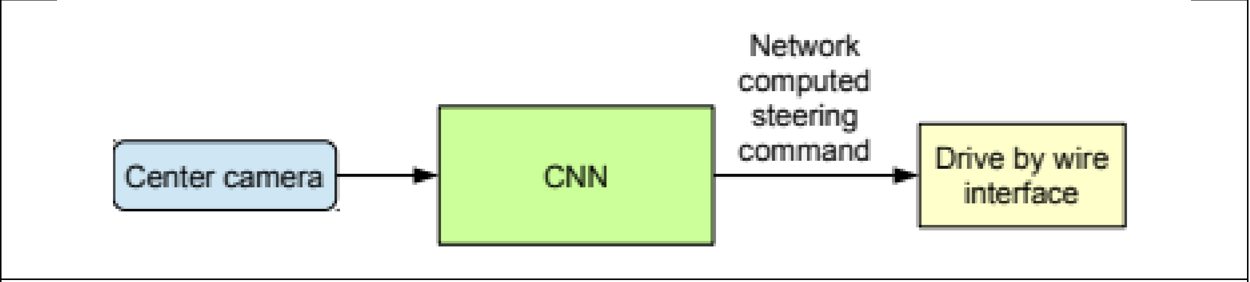


Figure shows the control block diagram.

In order to control the car in simulator (using the model), we take the scene from the simulated car’s center camera, crop it according and feed it to the model. The model then analyzes the scene and generates the steering angle. Thereby keeping the car on the track.

Note: You don’t have to crop the images coming from the simulator if you use keras crop functions inside the network. I didn’t use the keras crop methods so, I had to crop them using OPENCV before feeding them to the model. I left the model untouched as training after the cropping changes will take a lot of time. I instead edited the drive.py as needed.