Behavior Planning
Project 4

Sarbjeet Singh
Feb 28, 2019

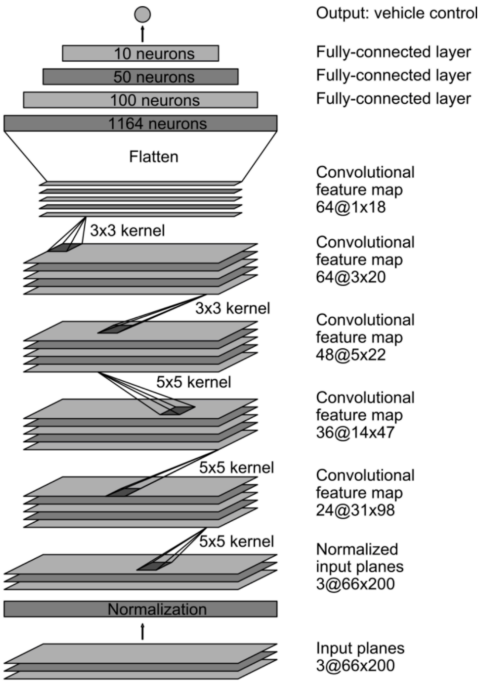
Model Architecture

The model architecture utilized for this assignment is one developed by researchers at Nvidia and detailed in a paper written by the team [Muller, Urs, et al. “End-to-End Deep Learning for Self-Driving Cars.” *NVIDIA Developer Blog*, Nvidia , 17 Aug. 2016, devblogs.nvidia.com/deep-learning-self-driving-cars/.]. This model takes an input image that is 66x200x3 and normalizes it [line 66] before performing five convolutional layers on it [line 69-75]. The normalization is performed with a Keras Lambda layer and after each convolution layer is a RELU activation layer to add non-linearity to the model. The feature map generated by the convolutions are flattened and processed through three fully connected dense layers [line 76-79].

To avoid over fitting the data, a validation amount is removed from the dataset and adding flipped images to the set augmented the data.

Below is the final architecture utilized in the project and can be compared to the nVidia layer shown below it:

Layer	Description
Input	66x200x3
Normalization	Image / 255
Convolution + Activation	5x5 filter, 24 layers
Convolution + Activation	5x5 filter, 36 layers
Convolution + Activation	5x5 filter, 48 layers
Convolution + Activation	3x3 filter, 64 layers
Convolution + Activation	3x3 filter, 64 layers
Flatten	
Dense	Output 100
Dense	Output 50
Dense	Output 25
Dense	Output 1



Model Architecture and Training Strategy

The development of this structure was a step-by-step process that started with implementing the very basics. Initially the dataset utilized only the center camera image, no image augmentation and the LeNet architecture. I utilized 3 epochs and the Adam optimizer.

The training data collected consisted of three laps around the track of proper driving, and than additional examples of how to recover from being two close to the side. Extra training samples for performing hard turns on corners were also collected to help the network learn how to get around them. This dataset consisted of images totaling 600MB in size.

Initially training on LeNet with no data augmentation and only images from the center camera, I found the car autonomously performed well in straight sections but struggled on corners. My first attempt to fix this was to add more data on how to recover around turns to the set, but this did not help.

I experimented by adding Dropout to LeNet but did not see any improvement. This led to augmenting the images by adding flipped images and steering angles for each image to the dataset. This slightly improved some of the over fitting that was occuring, but didn't yield the needed results.

Next I decided that the model needed to be more robust, and so followed the example giving in the lectures and implemented the architecture from Nvidia

Research. I kept the RGB images and input them as 160x320x3 into the new CovNet. The new model handled turns far better but still struggled with the sharper turns that appear after the bridge. At this point I read through the Nvidia paper in detail and decided to implement some of the additional image augmentation mentioned in the paper. This involved cropping the images as done in the lecture, resizing the image to match the 66x200x3 size utilized by Nvidia, and converting from RGB to the YUV color space.

The new model generated with these changes drove far more robustly, but still struggled on the sharper corners. The lectures detailed utilizing all three front facing cameras. I added the side camera to the image set, as well as a steering factor associated with them [lines 14-48]. The steering factor is added to the left side and subtracted from the right side. This enables the CovNet to learn how to center itself better. An example of the collected images is shown below in the order, left side, center and right side:

As you will see in the video.mp4, the model trained using data from all three cameras performed very well on corners and avoided going off the track. I did notice that it oscillates more on straight sections, most likely due to the steering factor correction coming from the side cameras training the model to wobble.

For future improvements, adding more dataset augmentation, such as darkening some images in the set, shifting and blurring images will help train the set better Additionally more data can be collected for training and a more complex steering correction can be implemented.