Self Driving Car Engineer Nanodegree
Udacity

Manikandan R
vasumaniram@gmail.com

Behavioral Cloning - Project 4

Objective:

The main objective of the project is to train an end-to-end CNN model to predict the steering angles based on the given input image.

- Use the simulator to collect the data of good driving behavior
- Build a Convolution Neural Network that predicts steering angles from images.
- Train and Validate the model with the training and validation sets
- Test that the model drives in autonomous mode successfully in 'track one' without leaving the road.
- Summarizing the results with the written report.

Required Files:

The submission includes the following files,

File	Description
drive.py	Script to drive the car in autonomous mode given the model file.
model.py	Script to define the CNN architecture, preprocessing and to train the model.
model.h5	Trained CNN Model with Udacity provided dataset.
generator_model.h5	Trained CNN Model with the collected dataset.
Writeup.pdf	Report summarizing the results.

Quality Of Code:

The model.h5 and genrator_model.h5 models can be used with the *drive.py* and *Udacity Simulator* to drive the car in autonomous mode. The following commands will drive the car in autonomous mode and the simulator should be running in autonomous mode.

python drive.py model.h5

python drive.py generator_model.h5

Code comments have been added in model.py to explain the functionality of each and every function and the pipeline.

1.Model Architecture and Training Strategy:

1.1 Model Architecture:

The CNN model architecture used has the following layers,

1. Normalization Layer
2. Preprocessing Layer
3. Convolutional layers
4. Relu Layer
5. Fully Connected Layers
6. Dropout layers

1.1.1 Normalization Layer: It helps to scale down the image input data from 0..255 to 0..1. Deep Learning heavily dependent on Matrix Calculations and normalization also helps faster calculations. There are multiple ways to normalize and the applied normalization formula is,

$$Normalized\ Image\ =\ (Image\ /\ 255.0\ )\ -\ 0.5)$$

Note: Subtracting 0.5, helps to get the zero mean centered data.

1.1.2 Preprocessing Layer: There are sceneries other than the tracks from the captured images. As these information does not help the model and also constitutes for unnecessary computations. Hence those sceneries have been cropped out of the images. Keras Cropping2D layer has been used as its much effective in cropping the images while leveraging the GPU's parallel processing capabilities.

1.1.3 Convolutional Layer: The convolutional layers with the filters helps understanding from the low level features like lines, curves, shapes to the high level features like while objects.

1.1.4 Relu Layer: Relu layer is used to include the non-linearity

1.1.5 Fully Connected Layer: The convolutional layers are finally flatten to connect into the set of fully connected layers.

1.1.6 Dropout Layer: Dropout layer helps the model not to overfit by randomly switching off the active neurons.

1.2 Overfitting:

The following strategies have been used to avoid overfitting,

Shuffling: It makes sure that the order of the dataset does not influence the model.

Mixed Data: 1 lap of anti-clockwise driving data in addition to 2 laps of clockwise which will help the model to generalize better by not memorizing the particular track.

Dropout Layer: Dropout layers have been added between the fully connected layers for the model to generalize and not depending on the same number of active neurons to predict.

1.3 Hyper Parameters:

The following table summarizes the model Hyper Parameters used to train the model.

Hyper Param	**Value**
EPOCH	3
BATCH SIZE	32
LEARNING RATE	'Adam' optimizer has been used.

1.4 Training Data:

Training data has been collected to keep the car on track in the straight/curve lanes and under different road/lane conditions as well. Training data included the center lap driving for both clockwise and anti clockwise laps. Also the one lap of recovery has been captured to teach the model how to get into the center when its going off to the boundaries. The details of these training data have been explained in the next section.

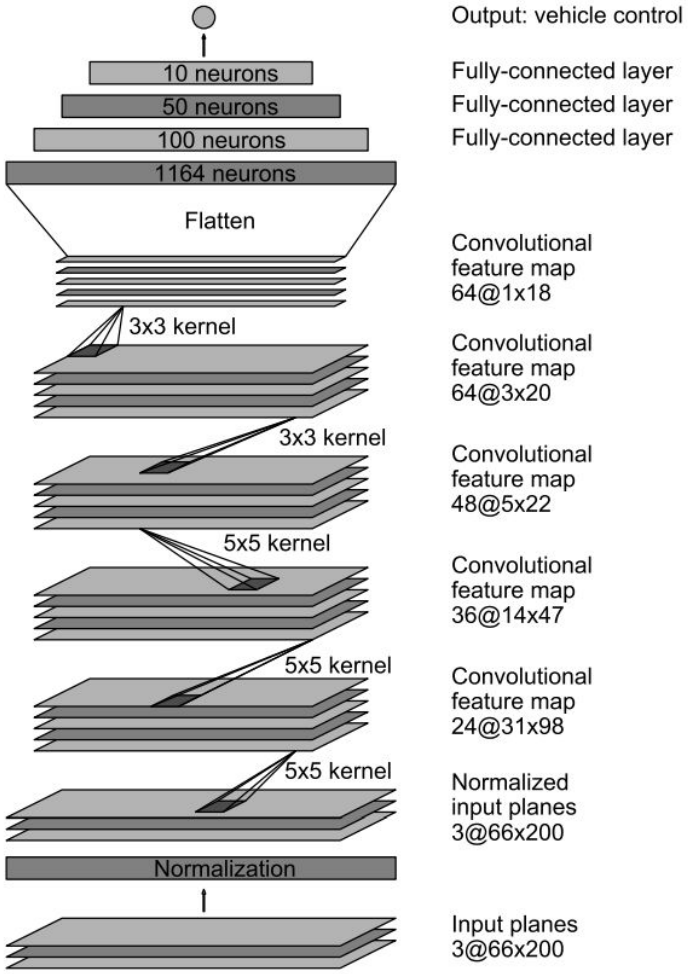
2. Architecture and Training Documentation:

2.1 Solution Design: The objective is to find the steering angle from the given input image. The following steps describe the solutioning part,

1. The training data has been collected with images and the steering angles when the image was captured.
2. Collected data has been shuffled and split into training and validation dataset. The ratio of training vs validation is 80:20.
3. The images are further normalized and preprocessed for a better model efficiency.
4. Mixed data,shuffling and dropout layer were used to avoid overfitting.
5. “adam” optimizer and “mse” loss function have been used to derive the error term and do backpropagation.
6. Model built has been run using the simulator in autonomous mode for testing to make sure the car does not leave the drivable portation of the track.

2.2 Model Architecture:

The model architecture is inspired from NVIDIA - End2End CNN Model For SDC as this model performed well in predicting the steering angles from the given images. The original NVIDIA model architecture is below,



Picture 2.2.1 - NVIDIA - End2End CNN Model For SDC

The following table describes the CNN Model Architecture, layers and parameters used to train the model.

Layer	**Description**
Input	160 X 320 X 3 RGB Image
Normalization	(Image / 255.0) - 0.5
Preprocessing	Crops only the ROI of the image
Convolution 1 - 5 * 5	'VALID' Padding | 2 X 2 Stride | Filters - 24
RELU	

Convolution 2 - 5 * 5	'VALID' Padding | 2 X 2 Stride | Filters - 36
RELU	
Convolution 3 - 5 * 5	'VALID' Padding | 2 X 2 Stride | Filters - 48
RELU	
Convolution 4 - 3 * 3	'VALID' Padding | 1 X 1 Stride | Filters - 64
RELU	
Convolution 5 - 3 * 3	'VALID' Padding | 1 X 1 Stride | Filters - 64
RELU	
Fully Connected 1	Flatten to Fully Connected - 100 Outputs
Fully Connected 2	100 Inputs | 50 Outputs
Drop Out	0.25% Neurons will be dropped
Fully Connected 3	50 Inputs | 10 Outputs
Dropout	0.5% Neurons will be dropped
Output	10 Inputs | 1 Output class.

Table 2.2.1 - Model Architecture

2.3 Training Data: The training data has been collected with the following approach,

1. ***Two laps of center driving*** - These two laps helps the model to train how to drive at the center.

Picture 2.3.1 - Center Lap Driving

2. ***One lap of Anti-Clockwise reverse driving*** - This lap gives a whole new lap experience which helps the model to generalize better.

Picture 2.3.2 - Reverse Lap Driving

3. ***One lap of recovery driving*** - It helps to teach the model what to do when it is off the track. The samples were collected only when its back to center from the sides.

Picture 2.3.3 - Recovery Lap Driving

One lap of *'track two'* could have been helpful to generalize the model better. As the data size had already reached 500 MB, it is not included to train the model.

The collected training data is available here.

The following table summarizes the size of training and validation dataset and the dimension of the image.

Number of samples	11195
Number of samples after Data Augmentation	22390
Number of training samples	17912
Number of validation samples	4478
Image data shape	(160, 320, 3)

Table - 2.3.1 - Training and Validation Dataset

From the above summarization, we could infer the following,

1. The size of the total dataset is 22390 after data augmentation
2. The *train: validation* split percentage is *80:20*
3. The images are color images with 3 color channels with the size of (160,320)

3. Simulation:

The model has successfully run on autonomous mode without leaving the drivable portion of the track and the collected data had been used to create the output video recording. The output video named *video.mp4* is saved under the project root folder.

Youtube Link: Autonomous Test Video

Github Repo: Behavioral Cloning