CarND-P3-Report

Data Generation

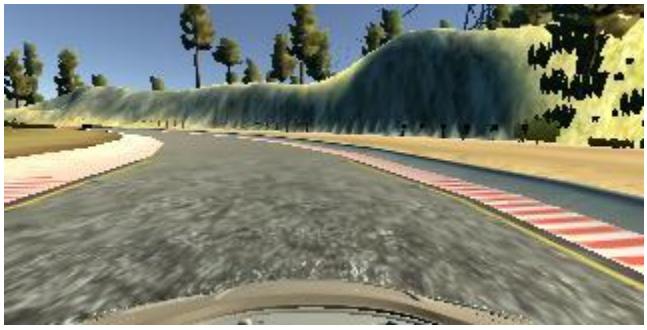
The vehicle was drove through Track 1 twice, one time clockwise and the second time counter-clockwise. During driving, I tried to weave back and forth between the middle of the road and the shoulder to mimic the real-world scenario.

To generate more data, as suggested by the lecture, correction (0.2) was applied to the center steering angle to utilize the images from left and right cameras.

Finally I got 6,972 images, 80% are randomly chosen as training dataset and the rest 20% are used as validation dataset.

The following are the left, center and right example images at the same time step:

(a) image from left camera



(b) image from the center camera

© image from the right camera

Data Preprocessing

The images were normalized to (-1, 1) first. After that the top 50 and the bottom 20 rows of pixels were cropped off because of the useless objects.

Model Architecture and Training

Python generator is used to generate the training and validation batches instead of importing the whole data in memory.

The basic architecture of the model was from the NVIDIA autonomous driving paper. It includes five convolution layers and three fully connected layers. The architecture was realized using Keras.

Multiple experiments were conducted to decide the size of each layer. It is observed that it is mainly the data quality not the model architecture that impacts the model results. Finally the size of each layer is as follows:

Layer	Filter size	Layer size
Input layer		(90, 320, 3)
Convolution layer 1	(24, 5, 5)	(32, 156, 24)
Convolution layer 2	(36, 5, 5)	(14, 76, 36)
Convolution layer 3	(48, 5, 5)	(5, 36, 48)
Convolution layer 4	(64, 3, 3)	(3, 34, 64)
Convolution layer 5	(64, 3, 3)	(1, 32, 64)
Flatten layer		2048

Fully connected layer 1		100
Fully connected layer 2		50
Fully connected layer 3		10
Output		1

The loss objective function is “mse”, and the optimizer is “adam”. The number of epochs is set as “3”.

Dropout

A dropout layers was added between the input layer and the first Convolution layer. Different values were tested and the results are as follows:

Dropout probability	**0**	**0.2**	**0.5**	**0.8**
Validation loss	**0.0074**	**0.0092**	**0.0079**	**0.01603**

It shows dropout didn’t prevent overfitting.

Results

The program was run on my local machine. It took about more than 3 minutes for each epoch (In total around 10 minutes). Finally the best validation MSE loss is 0.0074.

The performance of the model under autonomous model is shown in the run video.

Problems Met

When the model was applied for the autonomous mode directly after training, the program crashed. It's because there was no enough memory.

Failed Explorations

I tried to consider the images as a time series to generate the training dataset. Simply speaking, the images at time step $t-1$ and t are concatenated together as a new image, which then corresponds to the steering angle at t. The other processes were kept the same.

I guess it will make sense because if there was a curve on the road, the steering angle should be impacted for a few time steps. However, the local computer cannot afford this heavy computation (It crashed during the training). The validation loss was not obviously improved either.

The following paper may be useful in the future.

Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4489-4497. 2015.