Project 3 Behavior cloning project

November 5, 2017

1 FILES SUBMITTED AND CODE QUALITY

My project includes the following files:

1.1 SUBMITTED FILES

- model.py: this file containing the script to create and train the model.
- drive.py: this file for driving the car in autonomous mode.
- model-final.h5: containing a trained convolution neural network.
- writeup.pdf summarizing the results.

1.2 DRIVE CAR USING AUTONOMOUS MODE

Drive the car in autonomous mode by using the following:
python drive.py model-final.h5

1.3 SUBMISSION CODE IS USABLE AND READABLE

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

2 Model Architecture and Training Strategy

2.1 An appropriate model architecture has been employed

NVIDIA model [1] architecture has been employed. The employed model consists of a convolution neural network with 9 total layers. For Conv layers we have 2 sizes of filters, 3×3 and 5×5 filter sizes (see model.py lines Lei). The model (Fig 2.1) includes RELU layers to introduce nonlinearity (code line 104, 108, 112 and so on), and the data is normalized in the model using a Keras lambda layer (code line 101). The original data is split into YUV planes, however, I only used RGB channels in this project.

2.2 Attempts to reduce overfitting in the model

The model contains one dropout layer in order to reduce overfitting (model.py lines Lei). The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 130). The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

2.3 Model parameter tuning

The model used an Adam optimizer, so the learning rate was not tuned manually (model.py line 146). I set the learning rate as $1e-4$ as an empirical value.

2.4 Appropriate training data

The training data downloaded from Udacity was chosen to keep the vehicle driving on the road. In the begining, I used only center lane driving and ignore the left and right sides of the road. To keep the same strategy with the Nvidia's paper, I resize the image from original size 160×320 to 66×200. Before the resize, I remove the useless information such as sky from the original images. In addition, I added changing the brightness of image randomly. With a number of experiments, I observed the number of center images only is relatively small, therefore, I added a step to randomly choose left, right or center image in every batch. The left and right steering angles are corrected by a constant value 0.15. At the same time, I triple the number of steps in each epoch to increase the training data.

3 Model Architecture and Training Strategy

3.1 Solution Design Approach

The overall strategy for deriving a model architecture was to predict a steering angle based on the recored image frame. Basically, my step was to use a convolution neural network model similar to the NVIDIA [1]. I thought this model might be appropriate because it has been tested and proved to be surprisingly powerful to end-to-end steering angle prediction project. And the CNN architecture can be show as Fig 2.1.

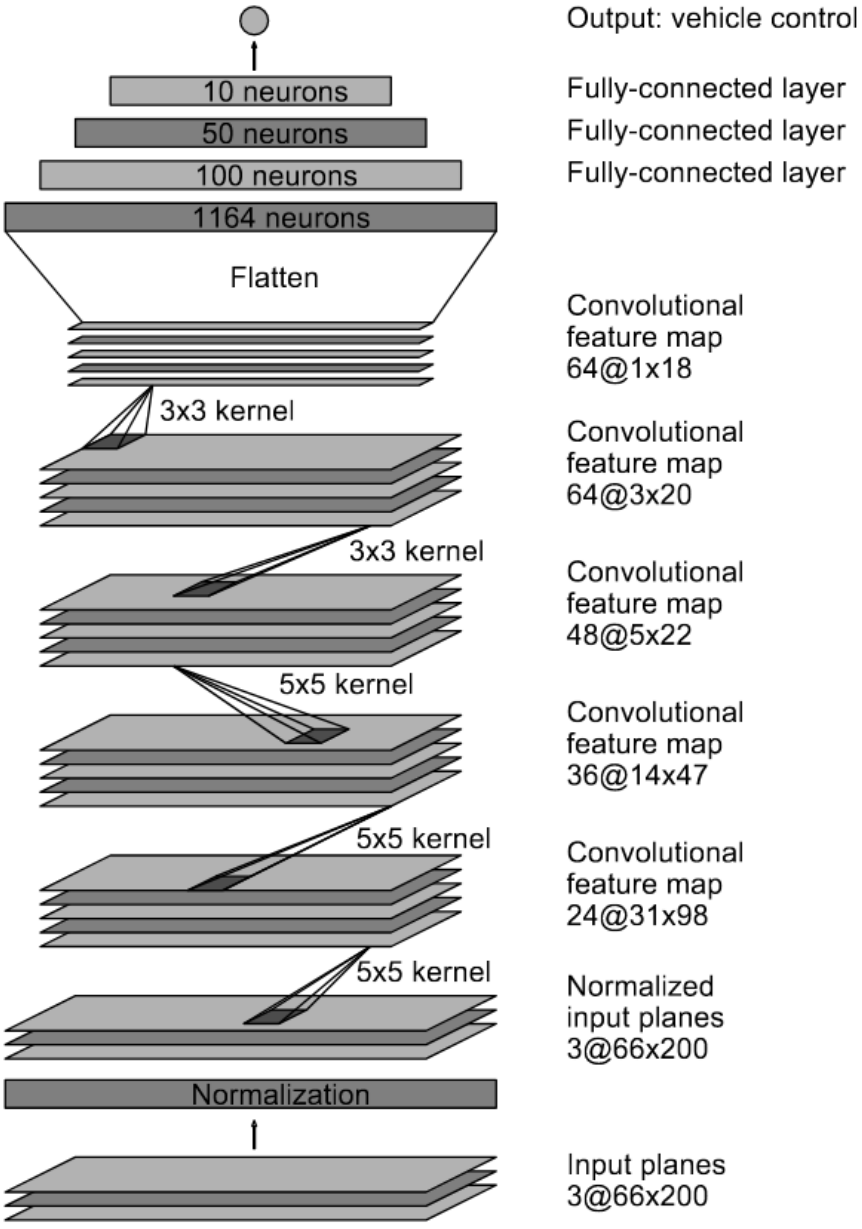
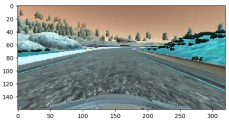
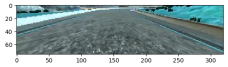


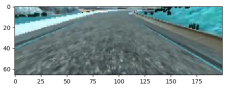
Figure 2.1: CNN architecture from NVIDIA paper [1].



(a) original center image



(b) cropped center image



(c) resized center image

Figure 2.2: preprocess center image

(a) Failed with overfitting

(b) Pass the failed spot

Figure 3.1: Failed and success case by augment the training data

In order to gauge how well the model was working, I split my image and steering angle data into a training (80%) and validation set (20%). To avoid the overfitting, I preprocessed images by randomly flip and change the brightness of images.

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track. For example, the car failed to turning left when I only trained with center camera and with flip the image. I tried to add random brightness and incorporate the left and right cameras. In this way, the new trained parameter make the car pass the same spot successfully. At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

3.2 Final Model Architecture

The final model architecture (model.py lines 103-141) consisted of a convolution neural network with the 5 convolutional layers and 4 dense layers and 1 dropout layer.

References

[1] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." arXiv preprint arXiv:1604.07316 (2016).