Behavioral Cloning Project

Introduction

The goal of this project is to develop a software pipeline that predicts and drives car steering angles through deep learning.

Environment

- Linux 18.04, Python3.6, OpneCV 3.1, Tensorflow, keras

File

- main.py : main code
- utils.py : sub code (image data preprocessing)
- model.py : sub code (model architecture)
- model.h5 : result of training model

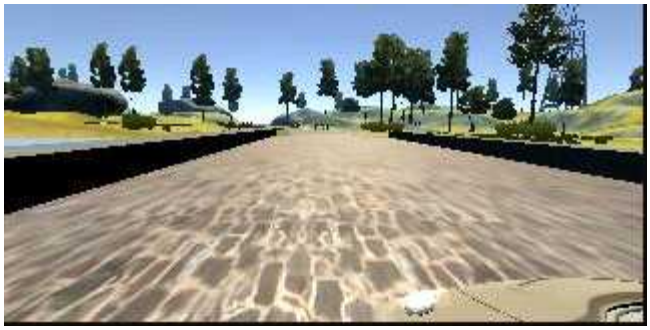
Step of this project are the following

- *step 1 : Use the simulator to collect data of good driving behavior*
- *step 2 : Build, a convolution neural network in Keras that predicts steering angles from images*
- *step 3 : Train and validate the model with a training and validation set*
- *step 4 : Test that the model successfully drives around track one without leaving the road*

step 1 : Data Collection

- In the training of the simulator, three front camera cameras fitted to the vehicle are used to record images and collect data. From the center, left, right camera. It also outputs the relevant steering angles used as labels for the three images.

[Image 1. Center]



[Image 2. Left]

[Image 3. Right]

step 2 : Image Preprocessing

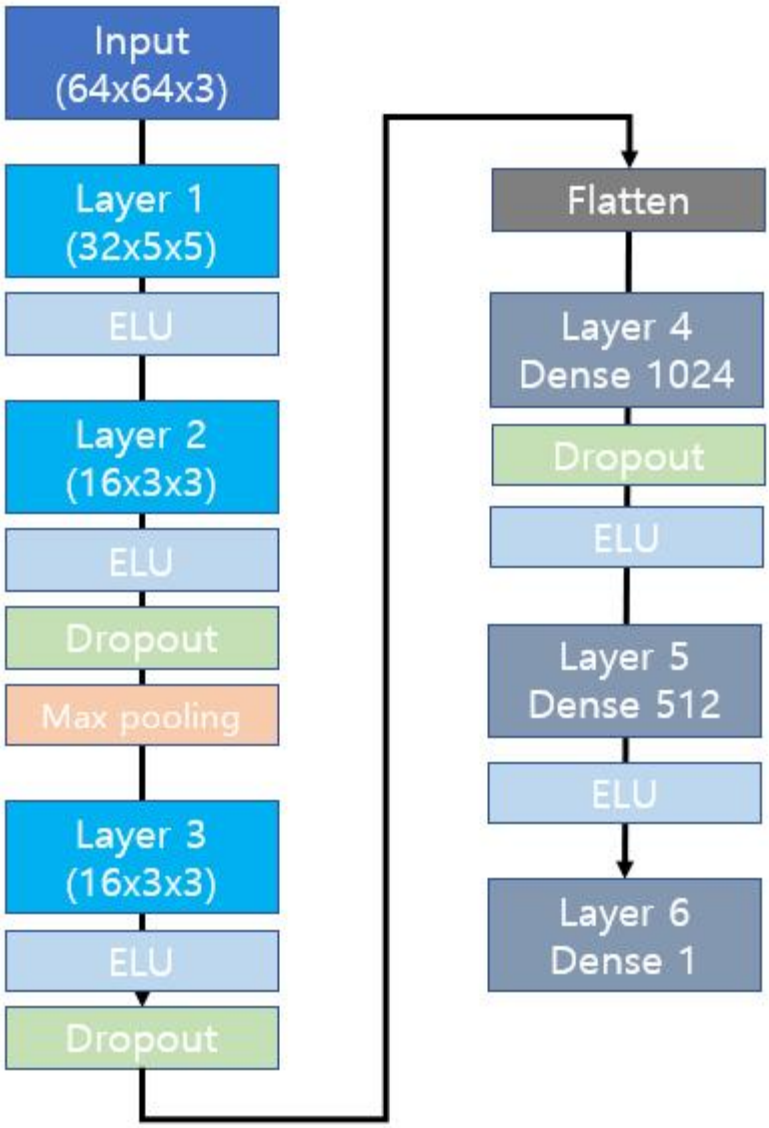
- The simulator's image is 160 x 320 pixels on a three-color channel (RGB).The first step in preprocessing was to reduce the size of the image and speed up the training. I resized the image to 32 x 64, and finally the neural network of each image has an input shape of 20x64 with a three-color channel.

- The following processing created a mirrored image dataset containing a copy of each vertically inverted image and a mirrored label with a reversed orientation for each value. I'm trying to prevent left-turn bias that seems to affect the car even when I'm driving straight on the road. The final data set (including more than 60,000 images) was created.

- The last pre-processing step was to create a validation set consisting of some random selection of image labels from the entire dataset. The verification set was used to monitor the performance of neural networks in training.

step 3 : Design Neural Network

- Input size : 64x64x3
- Layer 1 : Convolution2D(32x5x5), subsample(2x2) + ELU
- Layer 2 : Convolution2D(16x3x3), subsample(1x1) + ELU + Dropout(0.4) + MaxPooling2d(2x2)
- Layer 3 : Convolution2D(16x3x3), subsample(1x1) + ELU + Dropout(0.4)
- Layer 4 : Fully connected Dense(1024) + Dropout(0.3) + ELU
- Layer 5 : Fully connected Dense(512) + ELU



[Image 4. Model Architecture]

step 4 : Test model in Simulator

- The main model used the fit_generator function of "Keras", I used "Adam" as an optimizer, and the epoch was set to 3. Images generated also operates 20,000 images per epoch. In the middle, Validation was divided into 1/10 of the train image.
- In this function, there is only write code for learning.
- "drive.py" takes a constant image frame, converts the size to match the input shape of the model, and passes the transformed image to the model to predict the appropriate steering angle. The following steering angles are passed to the vehicle's controller and the vehicle will steer itself accordingly. Continuously receiving steering angles, the received steering angles have been well learned to ensure that the vehicle is safely driven in the middle of the lane.

[Image 5. Test video]