Project 3: Use Deep Learning to Clone Driving Behavior

The goals of this project

- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

My project includes the following files

- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network (please note there are two model2.h5 is a convnet trained by transfer learning based on vgg16.)
- video script to show the test results
- writeup_report.pdf summarizing the results

Data Exploration (see Data_Exploration.ipynb)

- The simulator recods three images (left, right, center) at a given time, cf. Fig.1
- The steering is mainly determined by the edges of the road
- The given sample data shows the data unbalance of dataset, most steering angles are negative where apprears many left turns in first track, cf. Fig.2
- We can cut off useless information, such as sky, car bonnet

Fig. 1 Sample image captured by the left, center and right camera.

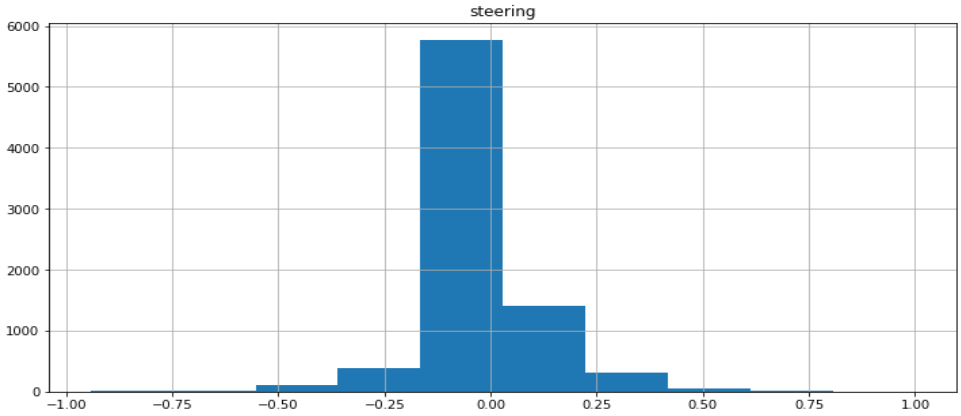

Fig.2 Unbalance in sample dataset

Data augmentation and Preprocessing (see Data_Exploration.ipynb)

- Cut off useless pixels of image
- Augment jitter data by change brightness, rotation, shearing, translation
- The first track contains lots of left turn, we can generate right turn artificially
- I use the data simulated by myself to record some recovery scenes. And I also inspired by some post in Internet to simulate recovery events.
- All generation tests are illustrated as following:

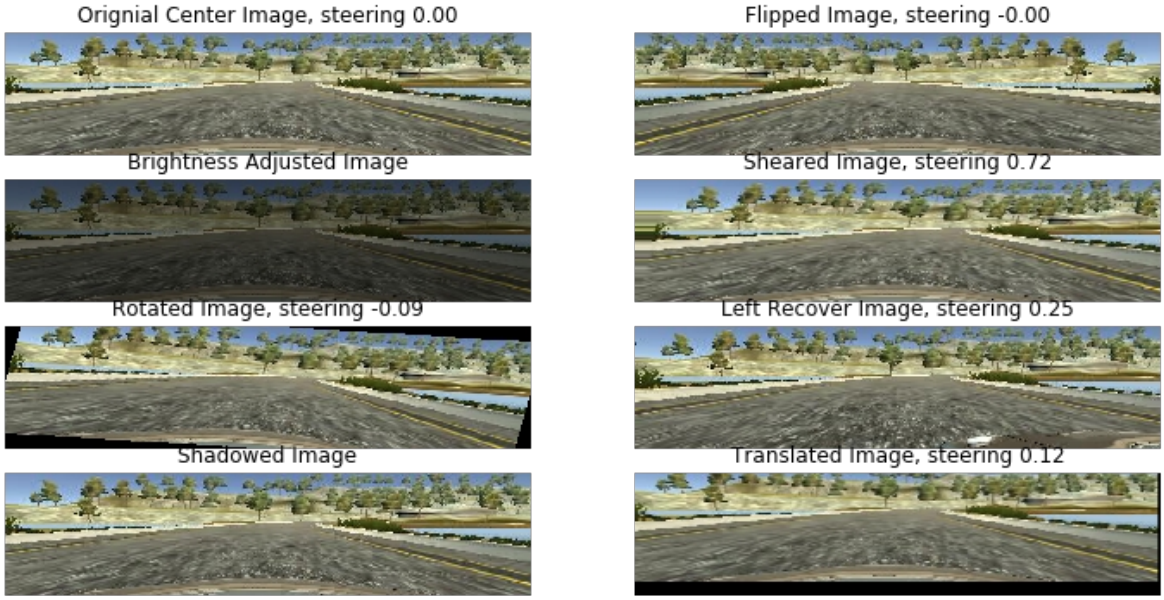

Fig. 3: Data preprocessing and augmentation

Model Architecture and Training Strategy

As you can see from model.py, I tried to build the ConvNet model which is inspired by the proposed article “End to End Learing for Self-Driving Cars” from NVIDIA;

- **Modified NVIDA Model (code line 162 - 221)**

In this model, I set a model that is similar to the one in NVIDIA paper. The network depth and width is described in detail below. I chose 256*80 training data, 6680 validation data and batch size as 256. I add a 0.5 dropout layer to prevent overfitting. Adam optimizer is used and the learning rate is set to 0.0001. We trained the model in 15 epochs until the train/valid loss are less than 0.04 according to the model loss line in Fig.4.

- **Input layer:** RGB images with normalized size 66x200x3
- **Layer 1: ConvNet** 66x200x3 => 33x100x24
- **Activation:** ReLu
- **Max pooling:** 33x100x24 => 32x99x24
- **Layer 2: ConvNet** 32x99x24 => 16x50x36
- **Activation:** ReLu
- **Max pooling:**16x50x36 => 15x49x36

- **Layer 3: ConvNet** 15x49x36 => 8x25x48
- **Activation:** ReLu
- **Max pooling:**8x25x48 => 7x24x48
- **Layer 4: ConvNet** 7x24x48 => 7x24x64
- **Activation:** ReLu
- **Max pooling:**7x24x64 => 6x23x64
- **Layer 5: ConvNet** 6x23x64 => 6x23x64
- **Activation:** ReLu
- **Max pooling:** 6x23x64 => 5x22x64
- **Dropout:** 0.5 keep
- **Flanten:** 5x22x64 => 7040
- **Layer 6: Fully Connected** 7040 => 1164
- **Activation:** ReLu
- **Layer 7: Fully Connected** 1164=> 100
- **Activation:** ReLu
- **Layer 8: Fully Connected** 100 => 50
- **Activation:** ReLu
- **Layer 9: Fully Connected** 50 => 10
- **Activation:** ReLu
- **Logits: Fully Connected** 10 => 1

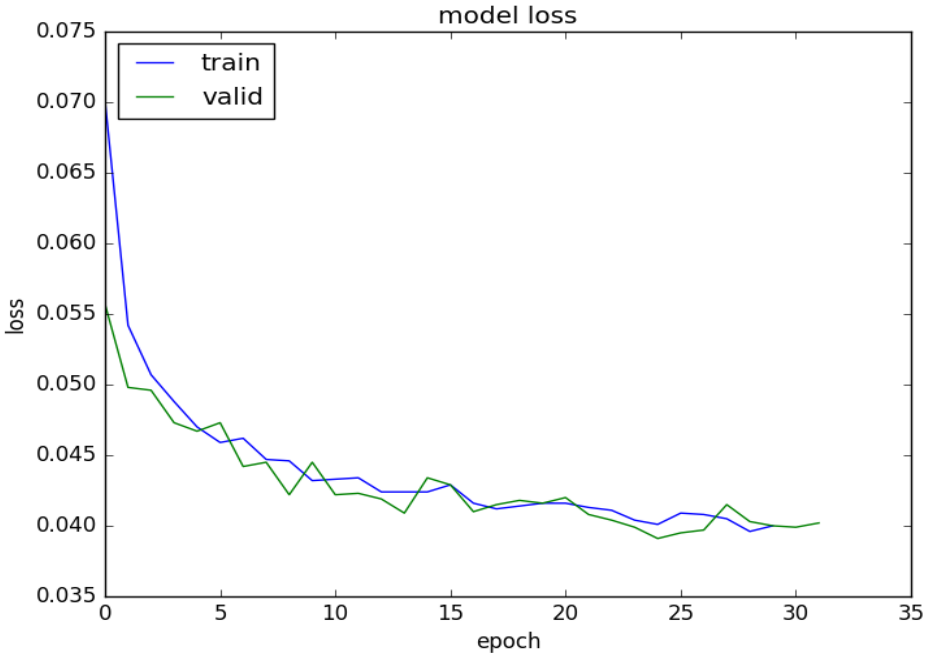

Fig.4 Model loss in training process

Model Results

As tested in two tracks, the trained model works well in both track as a autonomous driving within the lane. Because I introduced some recovery dataset, the car will be driven with lateral swings sometimes.

Some notes

In the test, the car is not driven smoothly and I think it is caused by the training dataset recorded. I use the keyboard to change the sheer and sometimes it works bad. I also give a second model of VGG16 transfer learning model and the training time is too long without GPU but I think it will be a good method to train our model.