Behavioral Cloning

1. Data preprocessing

Because my driving skill in game is so poor, I decided to use the training data from Udacity. There are 8037 lines of driving record and 240111 images in the data set. In order to get enough data for training, I think images from left and right cameras should not be wasted.

To make things simpler later, first I use pandas to read the csv log file and extract only the useful information (center,left and right images, steering angel)and save them in a new file.

I spilt 10% of the whole data set as validation data and 90% as training set. For validation data, I use only image from center cameras because that's what our computer driver sees when he drives car. However for training data, I need to use everything I have to get a better result. So images from left and right cameras are used and an angel compensation for these images is applied.

2. Data augmentation

After the first try, I find I need more data to make my model work better. So I followed some strategy from a blog post: “An augmentation based deep neural network approach to learn human driving behavior”. Following methods are used: Brightness Augmentation, Horizontal and Vertical Shifts and Flipping.

Because angel of most driving log is zero, it gives my model a strong bias to output 0. To avoid this problem, I used a variable keep_pro to control drop probability of the data if it’s angel is less than 0.1. Keep_pro is 1 at the beginning of the training , but it will decrease as epoch goes up so my model is able to reach that part of data later.

3. Model architectures.

I followed most guides in udacity course. First, images are cropped so only useful part of the image is feed into the model. Then images are resized to 64 * 64 and normalized to -1~1 so the training will be faster. I build a model based on NVIDIA Architecture which contains 4 convolutional layers followed by

a flatten layer and then 4 dense layers. Dropout layer is added between dense layers to avoid overfitting. And elu(Exponential Linear Unit) is used as activation fuction except for the last layer to avoid dead neuro problem during the training.

4. Training and result.

I used adam as my optimizer and 0.0001 as learning rate. As the training loss and validation loss is not dropping much after 2 epoch, I stopped training and saved the model.

As you can see in my video, this model is working great on first track. And to my surprise, even though I never trained it on the second track, it’s running perfect too.