Behavior Cloning

Udacity self-driving car Nanodegree project

1.Collecting Training data

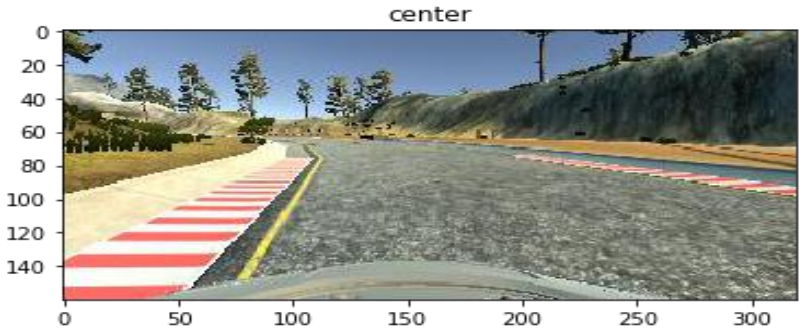
Obtaining good enough training data is of great importance to this project. In order to teach the model to steering the car and avoid overfiting. I steered the car in the center of the road for quite a bit of time and collected more than 40000 training images in total. Recovery was performed by stearing towards the edges and before the car could leave the track it was steared back towards the center of the road.

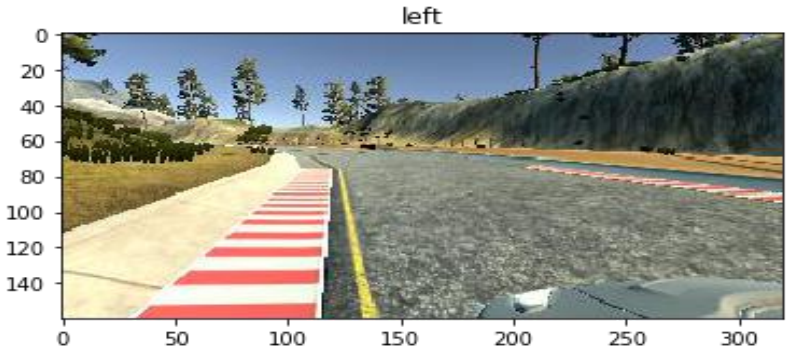
I flipped center images and toke the opposite sign of steering angles to augment my training data.Finally, I got 13939 frames(55876 images) for training.

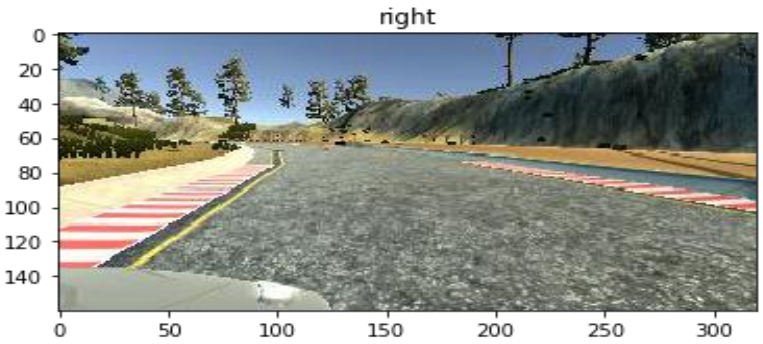
2.Camera Angles

Each frame consisted of three images(taken from left, right and center camera)

Here are some examples of camera images







All these three angles images are used in my model, where an offset angle has been added to left and right images. After a few experiments,I ended up using +0.20 for left images and -0.20 for right images.

3.Data preprocessing

In order to make the model capture useful features faster and reduce the size of my model.I firstly cropped the image using layers built in keras.I tried different cropping size and ended up by cropping the first 50 row pixels and the last 20 row pixels from each image.

After cropping the image size,I centered the pixels values to zero by dividing 255.0 and subtracting 0.5 for each pixels.

4.**Model Architecture**

In my core model architecture,I firstly I employed convolution layers for feature extraction and fully connected layers for regression.I used batch normalization layers to accelerate training and depress overfiting.Dropout layers have also been applied to each hidden layers to avoid overfitting.

Layers	Output shape
Two 3×3 ×16 convolution layer	42×157×16
BatchNormalization layer	Same as above
2×2 strides Maxpooling layer(dropout)	21×79×16
Two 3×3×32 convolution layer	17×75×32
BatchNormalization layer	Same as above
2×2 strides Maxpooling layer(dropout)	9×38×32
Two 3×3×32 convolution layer	5×34×32
BatchNormalization layer	Same as above
2×2 strides Maxpooling layer(dropout)	3×17×32
Flatten layer	1632
Fully connected 1024 (dropout)	1024
Fully connected 512(dropout)	512
Fully connected 100(dropout)	100
Fully connected 1	1

5.Training process

As this is a regression task and the model output only one real number, the mean squared error are selected as the only metric of the model. I split 20% of the samples for validation set and train the model with a generator in order to save memory space.I set both the train generator and valid generator batch size to 32. I employed adam optimization algorithm to train the model for 8 epochs. The validation and training mean squared error are both no more than 0.03 finally.