**Behavioral-Cloning**

**Write up** 1st submit:August 17th, Kenta Kumazaki

**1. Purpose**

In this project, I used what I’ve learned about deep neural networks and convolutional neural networks to clone driving behavior. I trained, validated and tested a model using Keras. The model outputs a steering angle to an autonomous vehicle to drive autonomously around the track.

**2. Goals/Steps**

The goals / steps of this project are the following:

* Use the simulator to collect data of good driving behavior
* 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

**3. Submission**

**(1) GitHub**

<https://github.com/kkumazaki/Self-Drivig-Car_Project4_Behavioral-Cloning.git>

**(2) Directory**

* **Writeup\_of\_Lesson17.pdf**: This file
* **drive.py**: Script to drive the car (I didn’t modify the original file)
* **model.py**: Script used to create and train the model
* **model.h5**: A trained Keras model by using collected data from simulator
* **video.mp4**: A video recording of my vehicle driving autonomously around the track for at least one full lap

**4. Reflection**

**(1)Collecting training data**

To be honest, it was very difficult to drive the track smoothly in Training Mode by hand.  
I did many cycles of “collecting training data” and “making the Model”, then I finally got a good result.

I got the following training data, and finally I chose Data 1, 3, 6, 7, 9, 10.

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Purpose | Result | Final Choice |
| Data1 | Drive the center as much as possible. | There were a couple of times I reached the edge of the road, but it’s mostly good. | O |
| Data2 | Drive around the edge and try to  avoid lane departure. | I saved the unnecessary data to “try to approach to the road edge”, so I deleted it. | X |
| Data3 | Drive the curve smoothly as much as possible. | I mostly drove the track smoothly. | O |
| Data4 | Drive around the edge and try to  avoid lane departure. | I saved the unnecessary data to “try to approach to the road edge”, so I deleted it. | X |
| Data5 | Drive around the edge and try to  avoid lane departure. | I saved the unnecessary data to “try to approach to the road edge”, so I deleted it. | X |
| Data6 | Drive the center as much as possible. | There were a couple of times I reached the edge of the road, but it’s mostly good. | O |
| Data7 | Drive the center as much as possible. | There were a couple of times I reached the edge of the road, but it’s mostly good. | O |
| Data8 | Drive around the edge and try to  avoid lane departure. | I saved the unnecessary data to “try to approach to the road edge”, so I deleted it. | X |
| Data9 | Drive the curve smoothly as much as possible. | I mostly drove the track smoothly. | O |
| Data10 | Drive around the edge and try to  avoid lane departure. | I save the data only when “try to avoid lane departure”, not save data when “try to approach to the road edge”. | O |

When I collected “Recovery Laps” with Data 2, 4, 5, 8, I saved the unnecessary data to “try to approach to the road edge”, so when I trained the Model by using those data, the vehicle wandered a lot even at the straight road.

After I collected “Recovery Laps” with Data 10 as below, the vehicle behavior became very stable.

<Step1> Go to the road edge without saving data.  
<Step2> After start saving data, try to avoid lane departure.

Steer Neutral

Steer toward Left

Steer toward Left

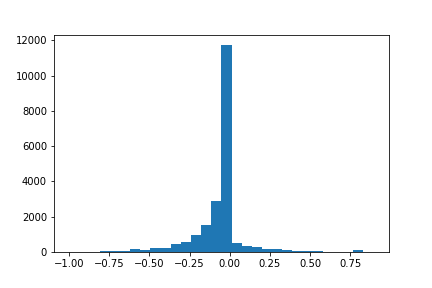
Steer Neutral

Steer toward Right

Steer toward Right

The following graph shows the histogram of Steering Angle of the Training Data (Final Choice).

I trained the data at the Track1, so the frequency of Left Side is higher than Right Side.

****

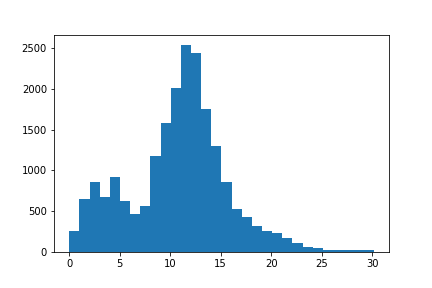
Steer toward Left

Frequency

Steering Angle

The following graph shows the histogram of Vehicle Speed of the Training Data (Final Choice).

To avoid unnecessary lane deviation, I drove slowly about 10-15 mph.



Frequency

Vehicle Speed

The total number of Training Data (image file and steering angle) is “20,928”.

**(2)Description of my pipeline**

My pipeline consisted of 3 steps as following.

Step 1: Preparation (import, read data, etc)

Step 2: Augment Images (Flip the image and steering right/left)

Step 3: Make Model

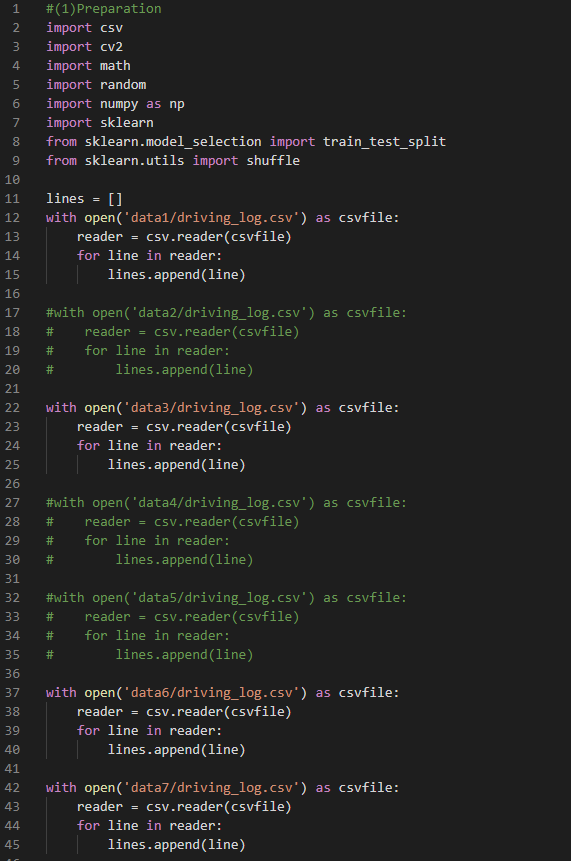
**Step 1: Preparation**

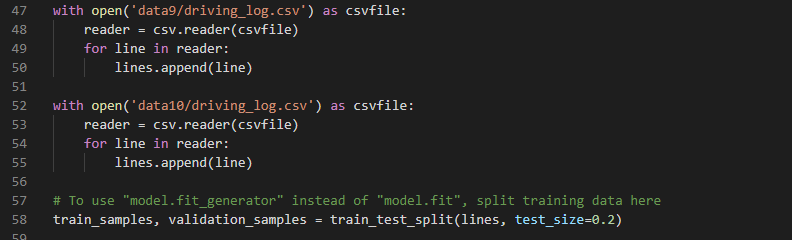
Load the Trained Datasets at the workspace.

As I explained before, I took multiple datasets but finally comment out unnecessary data.

At the end of this step, I generated “train\_samples” and “validation\_samples” by splitting the dataset.

As recommended in the Lecture, I set 80%: Training Data, and 20%: Validation Data (A).



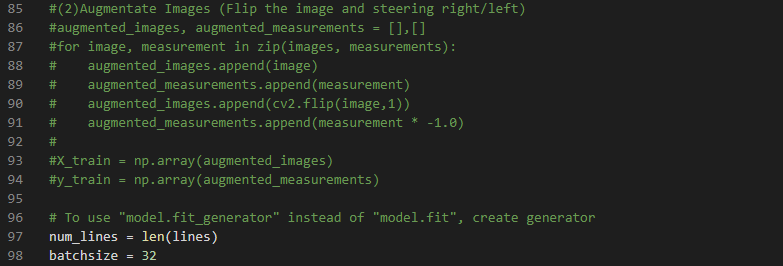


A

**Step 2: Augment Images**

At first, I was trying to make the Model without generator, so the codes for augmentation was quite simple.

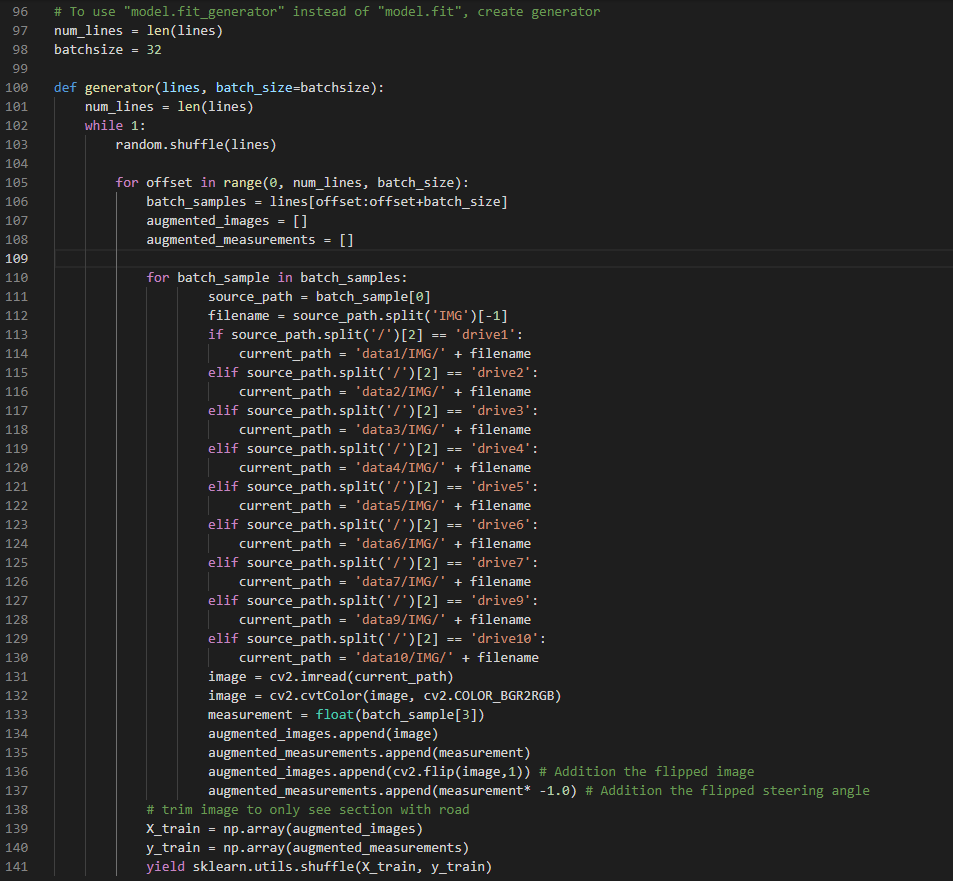
But as I write later I created generator, so I commented out it here.



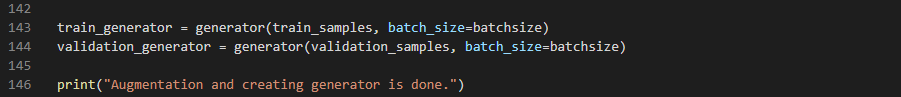
I created the generator as below. I followed the Lesson and I set the same batch\_size = 32.

Here, I augmented the image data by flipping left/right (B) to generalize the training data.

By shuffling the training data and validation data, I can avoid overfitting (C).

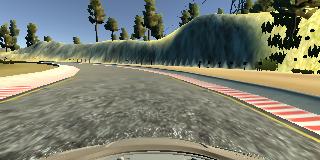


B



C

Show in the Lesson, the original image and flipped image will be as following. It helps generalization.

Original Image Flipped Image

**Step 3: Make Model**

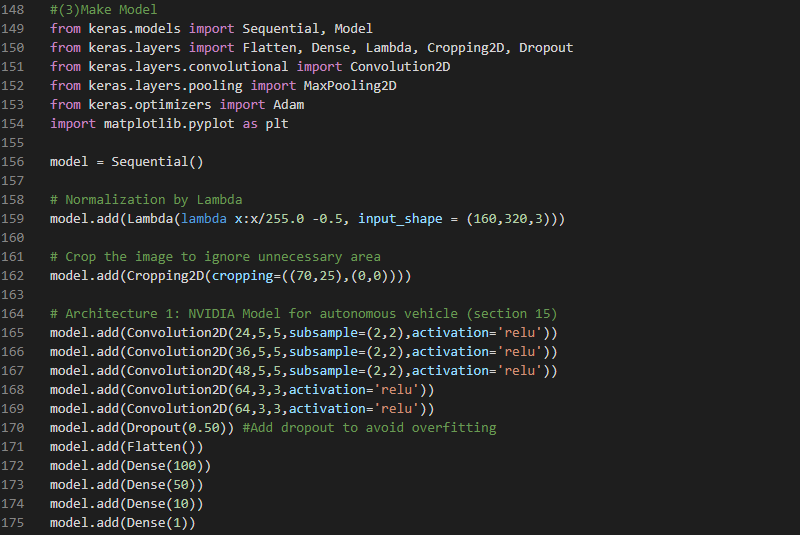
I added the Lambda Layer at first (C) to normalize the image data.

I added Cropping Layer (D) to eliminate the unnecessary area of image, such as image of sky, trees, etc.

I modified the Architecture of NVIDIA (E), which was recommended in the Lesson. I will explain it later.

I added Dropout Layer between Convolution Layers and Flatten Layer (F) to avoid overfitting.

I set the dropout parameter = 0.5, which looks common parameter in many Models.



F

D

C

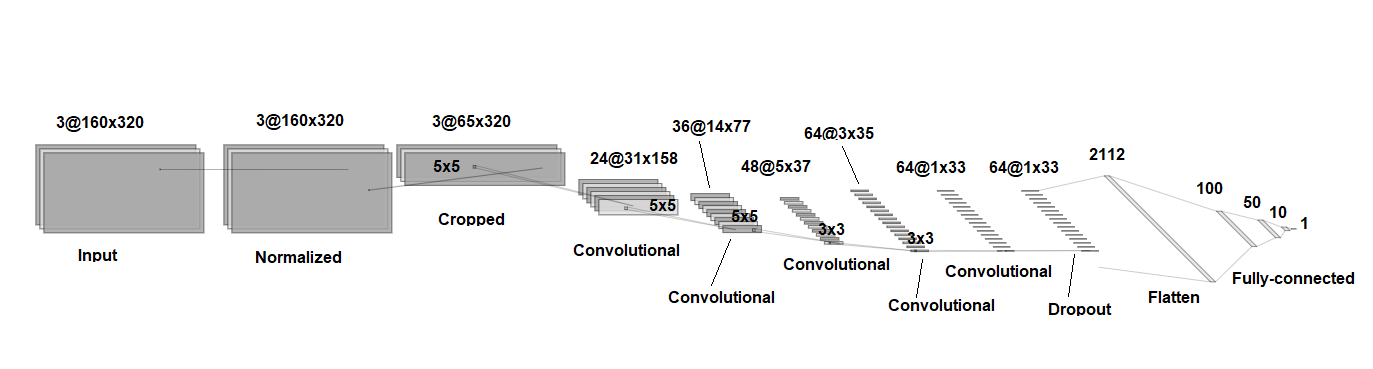
E

The architecture of my Model is show below.

My model consists of a convolution neural network based on NVIDIA model.

There are 5 Convolution Layers with 5x5 or 3x3 filter sizes and depth between 24 and 64, and there are 4 Fully-connected Layers with neurons size between 1 and 100.

In the Convolution Layer, I chose the activation function = RELU to introduce nonlinearity.



I chose Adam Optimizer and set the Learning Rate = 0.0001 (G).   
I searched Internet and people set the parameter for Adam Optimizer from 0.001 to 0.0001 range, so I chose the small rate to get better results even though it takes longer time. The training data number is “20,928” as previously written, so I thought it’s not so big data if I can use GPU Mode.

I tried learning many times by changing data sets and the number of epoch, and I finally found that the epoch number = 10 is good for my Dataset and Model (H).

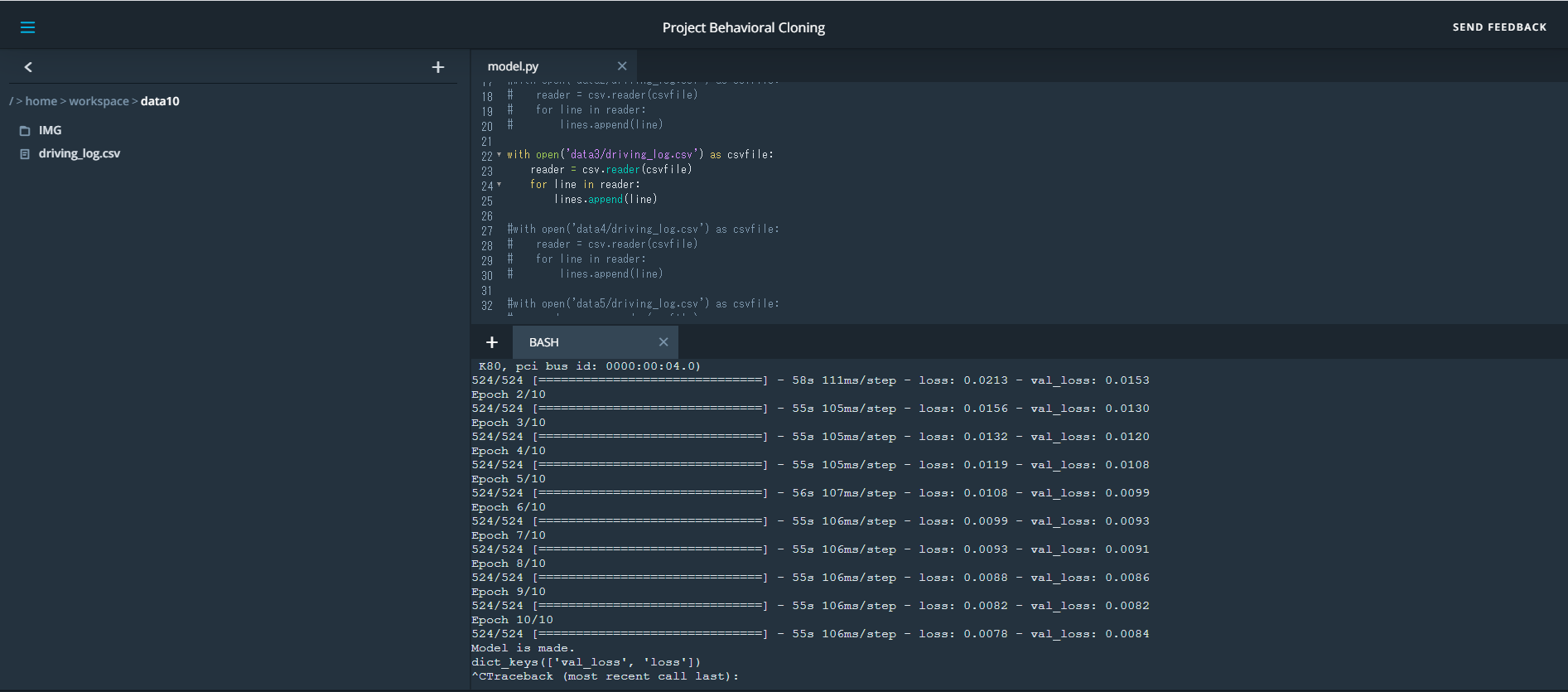
****

H

G

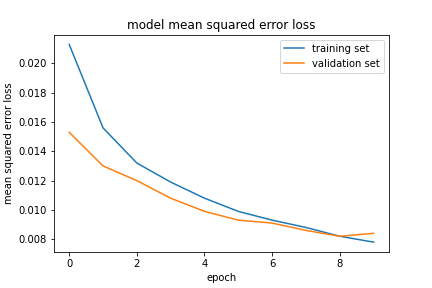
**(3)Result**

The following is the console log of the Udacity Workspace when I ran the final set of parameters and Datasets.



The resulting graph is shown below. The vertical axis means Error Loss.

Loss of Training Dataset keeps decreasing, and Loss of Validation Dataset became saturate at the 9th and 10th epoch. It looks good timing to finish the learning (I).



I

In the console of Workspace, I ran “python drive.py model.h5 <folder name>” and I ran the Simulation with Autonomous Mode. After that, I ran “python video.py <folder name>” and created **video.mp4** file.

(\* Before running video.py, I had to install “ffmpeg” in the Workspace.)

The vehicle drove well in the track 1. There was one time it drove very near from the right line, but it recovered soon thanks to the “Recovery Lap” training data. I think it was able to recover faster if I add more “Recovery Lap” training data.