Self-driving Car Nanodegree Program



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project 4: Behavioral Cloning

Due: 02/01/2019

# **Objectives**

The goals of this project are:

* Use the simulator to collect data of good driving habit
* Build a convolutional neural network in Keras to predict steering angles from images
* Train and validate the model with a training and validation set
* Test the model so that the car drives around track #1 without leaving the road autonomously

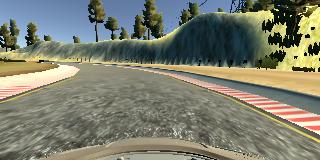
# **Procedures**

## Data Collection from Simulation

For this project, we will teach the car to drive autonomously. To do so, we need to train the convolutional neural network so that the car can decide steering angles at different scenes. We used the simulation tool provided by Udacity to collect series of images capturing by our driving behavior. Figure 1 shows an example of the image collected from the simulation.



*Figure 1a: Left camera*



*Figure 1b: Center camera*



*Figure 1c: Right camera*

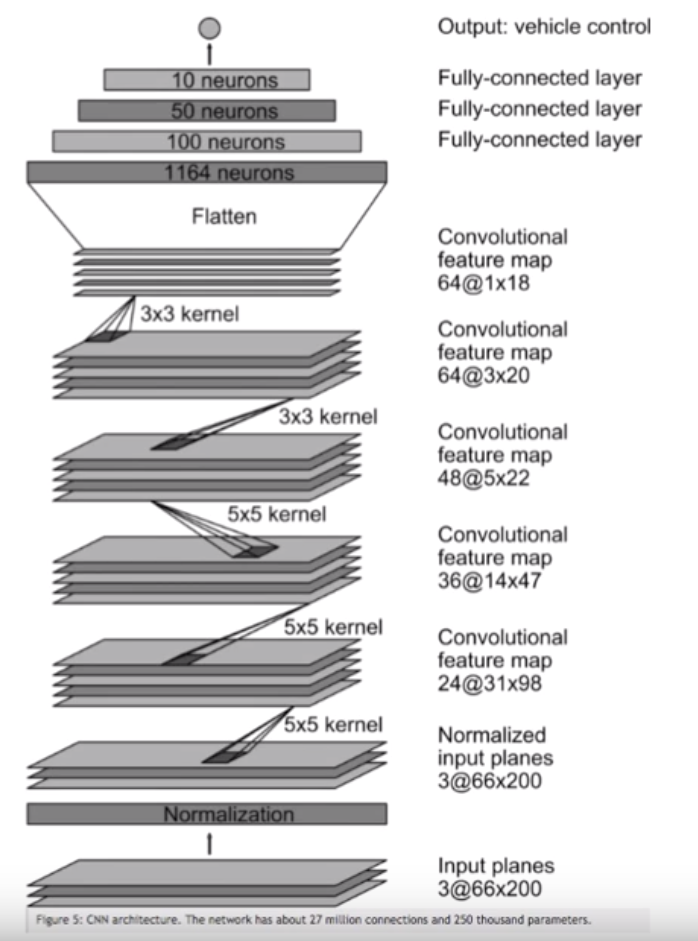
To teach the car how to drive safely, I collected multiple datasets, each of which teaches the car to drive properly in different situations. In particular:

* Data 1: 3 laps of track 1, driving at center
* Data 2: 1 lap recovery from sides in case the car is sliding to sideways
* Data 3: 1 lap of smooth driving on curves
* Data 4: 2 laps on track 2

To generalize the model, we can also try to drive clockwise (currently it drives counter clockwise according to this map so the car tends to learn more from turning left) and can also flip collected images to create new dataset.

## Design Model Architecture

The model architecture for this project is based on NVDIA architecture for autonomous driving. The model can be seen in Figure 2.



*Figure 2: NVDIA Model for Autonomous Driving*

The code snippet to implement this model is shown below.

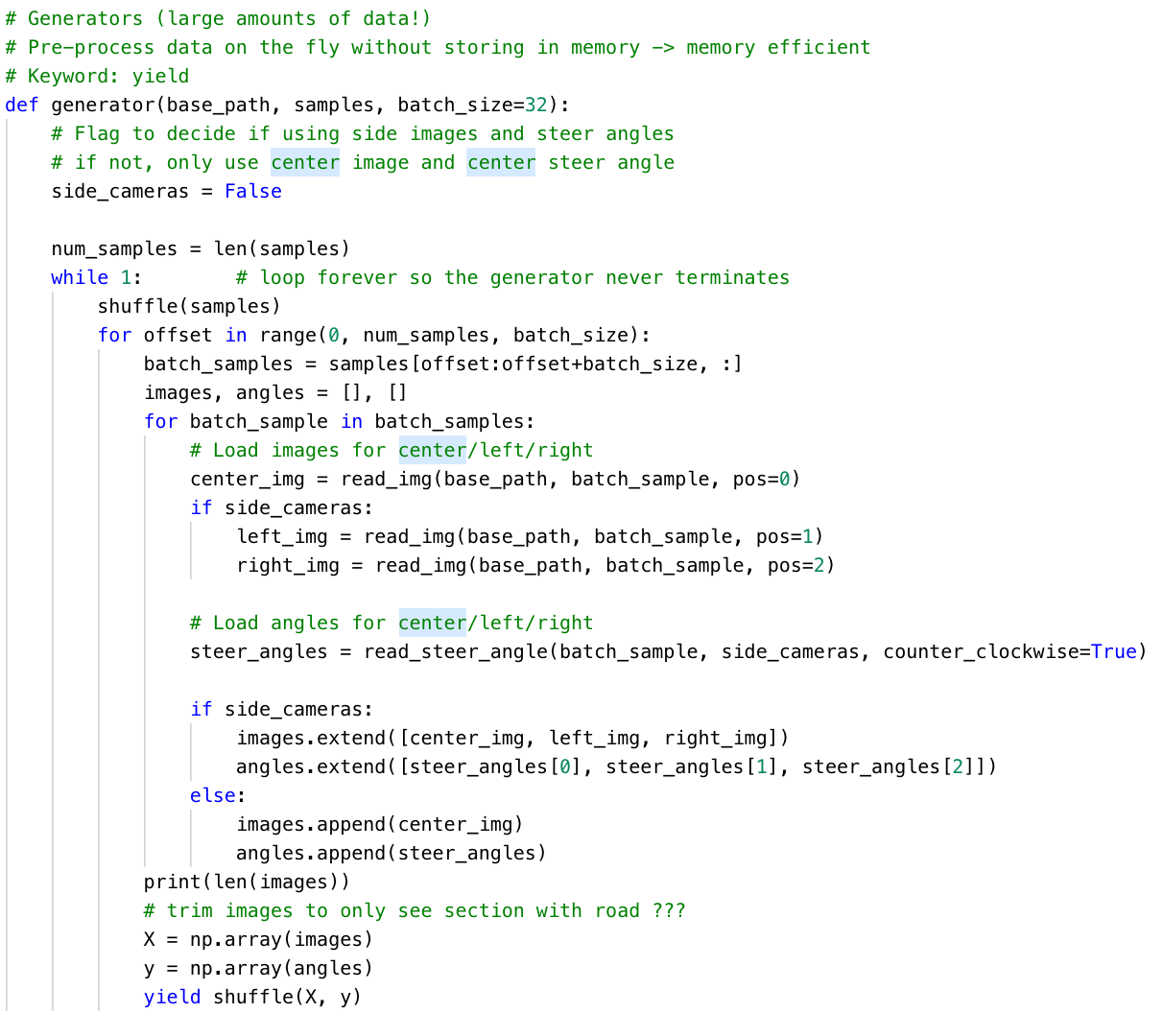


*Figure 3: Implementation of NVDIA Model Architecture in Keras (Code Snippet)*

As shown in the code snippet above, I implemented the model using Keras framework. Initially, the **Lambda** layer is used to normalize and mean-centering of the images. Then a **Cropping** layer is used to crop the original image of size (160x320x3) to (75x320x3) in order to crop out the unnecessary parts of the images such as trees and sky. Following the Cropping layer is the **convolutional** layers with usage of **MaxPooling** and **Dropout** layer with dropping rate of 20%. After that, four **fully-connected** layers are used to finish the model.

## Training Strategy

For the training purposes, I first created a generator so that it can generate batches of images on the fly without the need of storing pre-processed images in memory. This is faster and much more memory-efficient. The code snippet below shows how the generator is implemented. More information of helper functions can be seen in the code.



*Figure 4: Batch-Generator Implementation*

As discussed in section 1 above, since I have collected 4 datasets to generalize the model in different situations, I also need to train the model with these 4 datasets. To do this, I first trained the model with data 1, then saved the model. To train for data 2, I re-loaded the saved model and continued training on it and kept doing so for the remaining datasets.

Finally, since this is a regression model as we want to predicts the steering angle based on the camera images, I used mean squared error (MSE) as the loss function and Adam as the optimization algorithm.

## Test the model on Autonomous Driving

After training the model, we can test it by running the model in autonomous mode from the simulation GUI. In this mode, we input our trained model and it will output the steering angles at every scene and the car will drive autonomously. The car did a pretty good job as no tire left the drivable portion of the track and did not pop up onto ledges or roll over any surfaces other than the main road.

# **Discussions (Further improvements)**

Further improvements can be done such as:

* Generate more training examples with track 2 since track 2 is more difficult and can further generalize the model in different situations, that may be even tough for human driving
* Tuning parameters such as number of batches, number of epochs, learning rate
* Try different model architectures