Self-Driving Car - Behavioural Cloning Project

Vitali Mueller – http://www.vitalimueller.com/

Behavioural Cloning Project

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behaviour
- 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

Introduction

The objective of this project is to build an end-to-end control system (supervised agent) who is capable to clone human behaviour using Convolutional Deep Neural Network (CNN). Udacity has provided a simulator environment with closed loop tracks for students to practice and build autonomous self-driving agent. Three viewpoints were captured from the cameras which are placed cantered on top of the car.

The process of building an autonomous agent consists of two phases:

1. Training – Manual Driving
2. Validating – Autonomous Driving

During the training phase, we are going to drive a car in the simulator using the keyboard "W", "A", "S", "D" to steer and accelerate the car. While we are driving the car, the simulator is capturing information such as steering angle, throttle position and images from left, centre and right position.

Left	Center	Right
		

Once the capturing is completed we are using the captured data to train the model using Convolutional Neural Network (CNN).

The Goal:

Develop an end to end control system using Deep Neural Network.

Solution:

The CNN architecture for solving this project was taken from the paper by Bojarski, Mariusz, et al [1]. The authors have proved that it is possible to drive a real car autonomously, using cameras mounted on top of the vehicle. Images were used as an input value. The output value is steering angle of the end to end control system. The system is an open loop in validation mode and only a closed loop when it is in training mode.

Further Information:
This simulator is built with Unity. Unity is a free game engine.
Both the simulator and Unity is free available here:

- Unity
- Self-Driving Car Simulator

Udacity has provided two different files in order to run and validate the model.

- Drive.py

- Video.py

Drive.py: The script is used to drive the car in autonomous mode and save images seen by the agent. Drive.py file had to be modified in order to work. Image dimensions need to be the same as it was used for training the end to end system.

Video.py: Creates a chronological Video of the agent driving

Dependencies

Python 3.5 and the following Python libraries are required:

- Keras
- NumPy
- SciPy
- TensorFlow
- Pandas
- OpenCV
- Matplotlib
- Jupyter

In this project, we need to do:

1. ***Data Generation***
2. ***Behaviour Cloning – Training our Convolutional Neural Network (CNN)***
3. ***Testing\Validation Mode – Autonomous Driving Mode***

Data Generation:

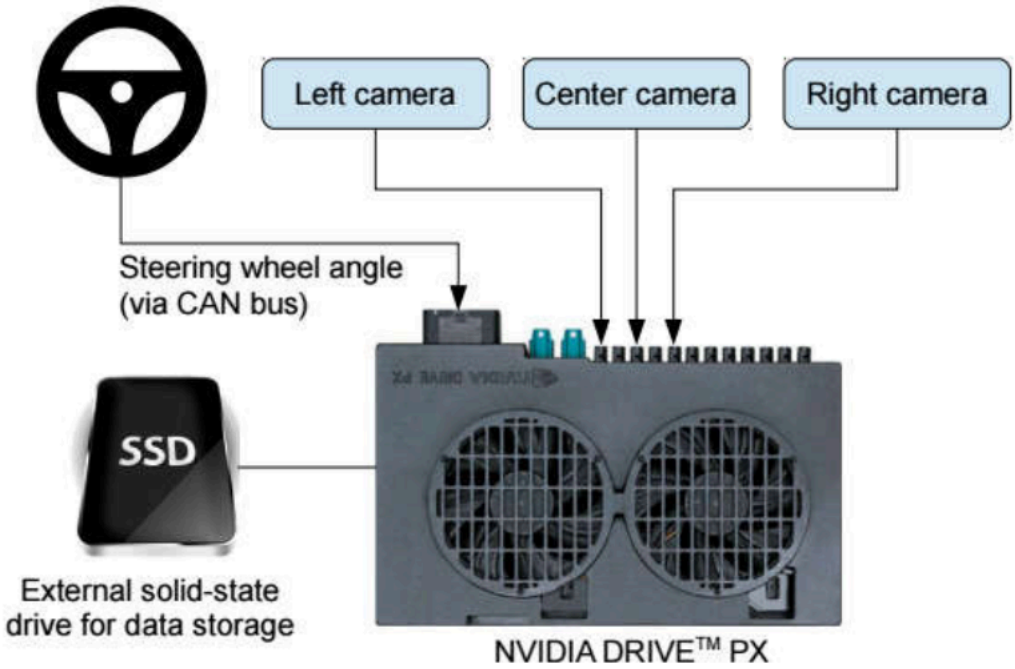
In order to generate data, you will need to start the driving simulator and select training mode and track.

Once the Simulator is running click on the keyboard “R” and it will start to record your data.

Now you will need to drive your car around the track approximately 5 rounds. Once you have completed your 5 rounds click again on “R” and it will pre-process your data and store it in the selected Folder.

If everything went correctly for recording data, you should see the following in the directory you selected:

- IMG folder – this folder contains all the frames of your driving
- driving_log.csv – each row in this sheet correlates your image with the steering angle, throttle, brake, and speed of your car. You will mainly be using the steering angle
- images are recorded from centre, left and right.

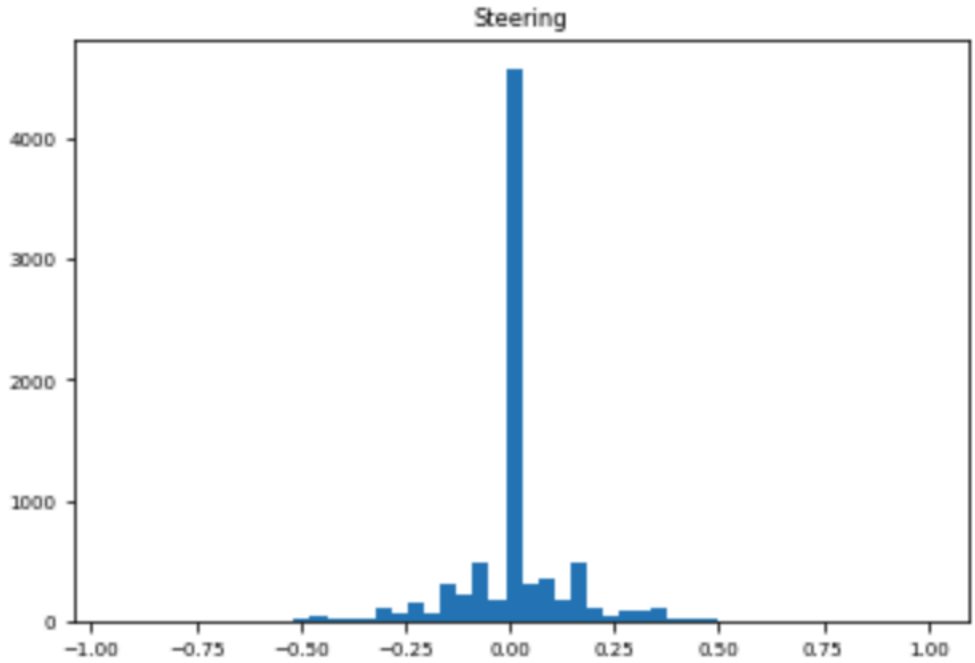


Source: http://www.i-programmer.info

Data Augmentation – Data Pre-processing

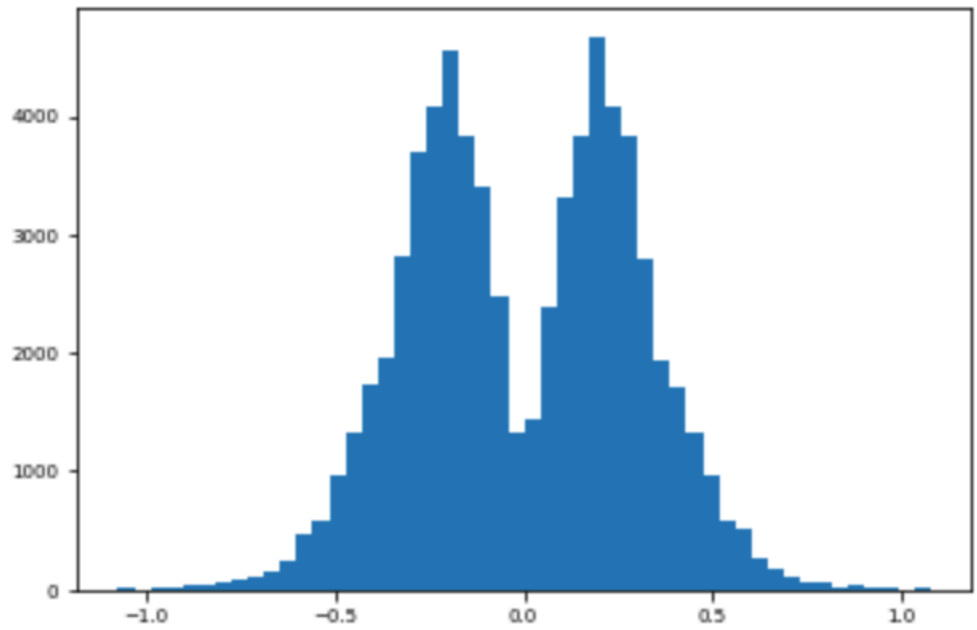
It is important to look at the generated data and analyse it before using it for training phase. Datasets for the training mode were provided by Udacity. Using the data without pre-processing or augmenting leads to driving off the road in curvy situations.

Analyses shows that there are too many data samples centred around zero steering angle and just few of them had extreme steering angle.



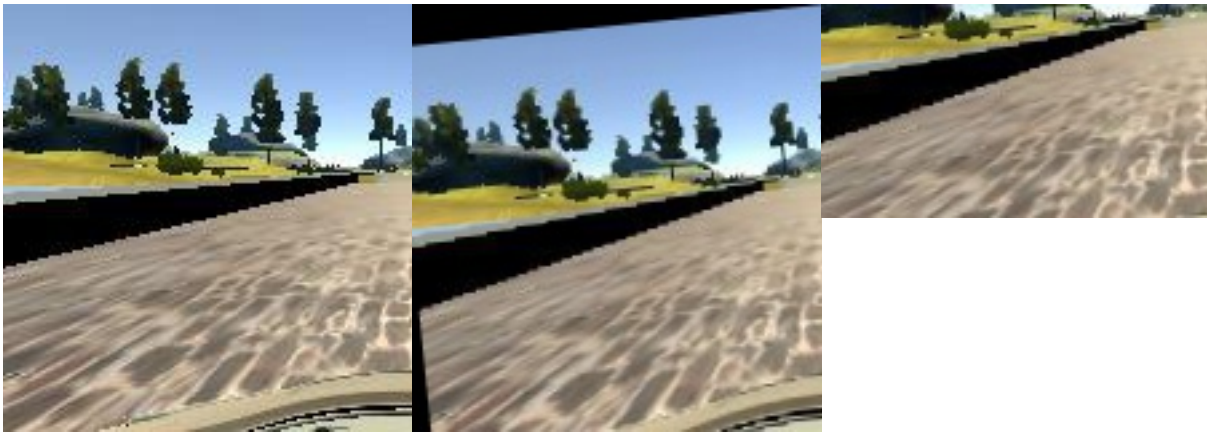
Original Dataset

The key method to have a stable driving agent is to reduce the access information for the centre region and provide more information to the left or right regions. To do this we will would need to copy the data in that regions by shifting images captured by the centre camera both to the left and to the right and correcting the steering accordingly. Also, the image was normalised for values between -0.5 and 0.5.



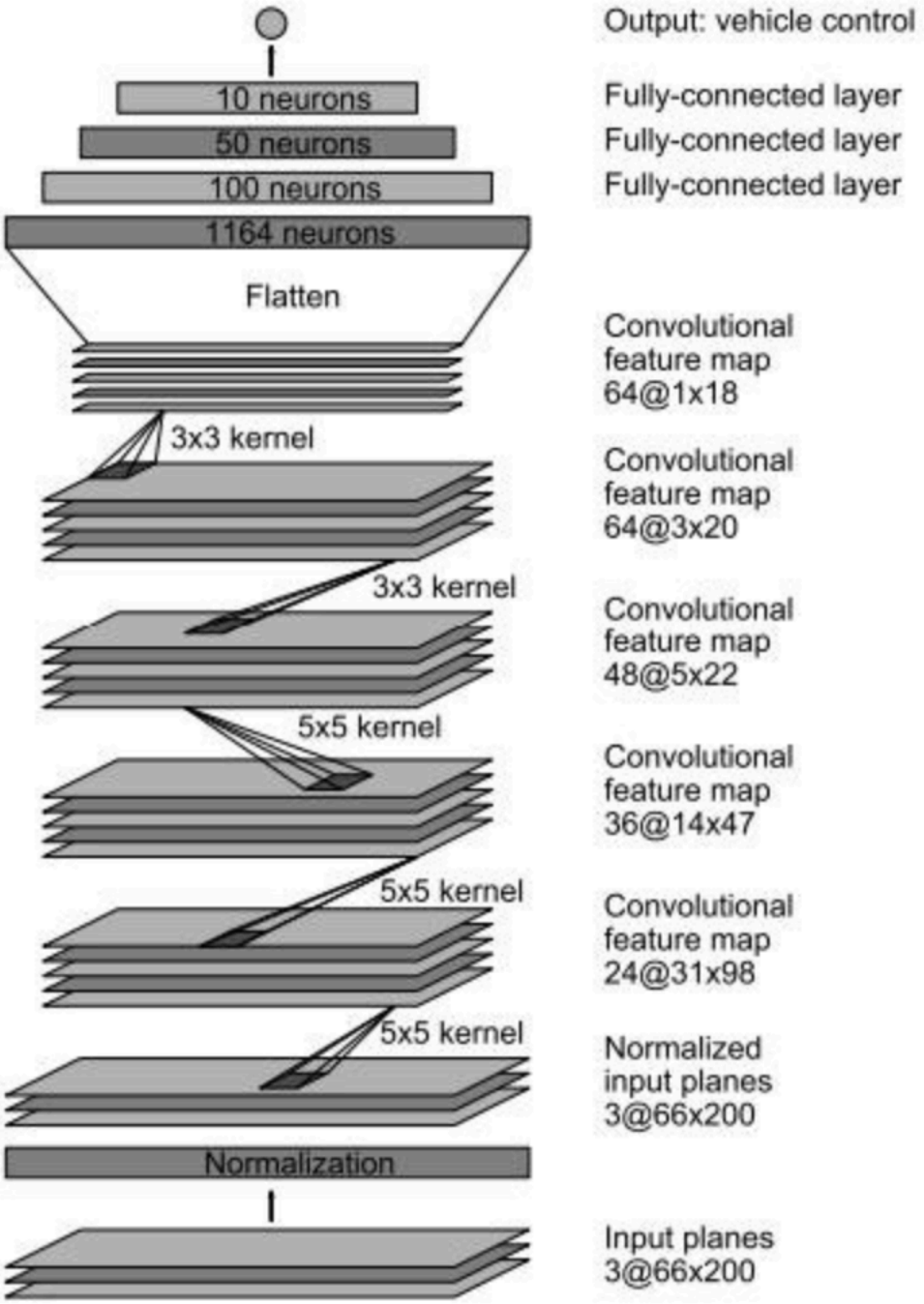
Pre-processed Dataset

Also, the origin image with size 320x160 were cropped to 300x80 to remove useless information.



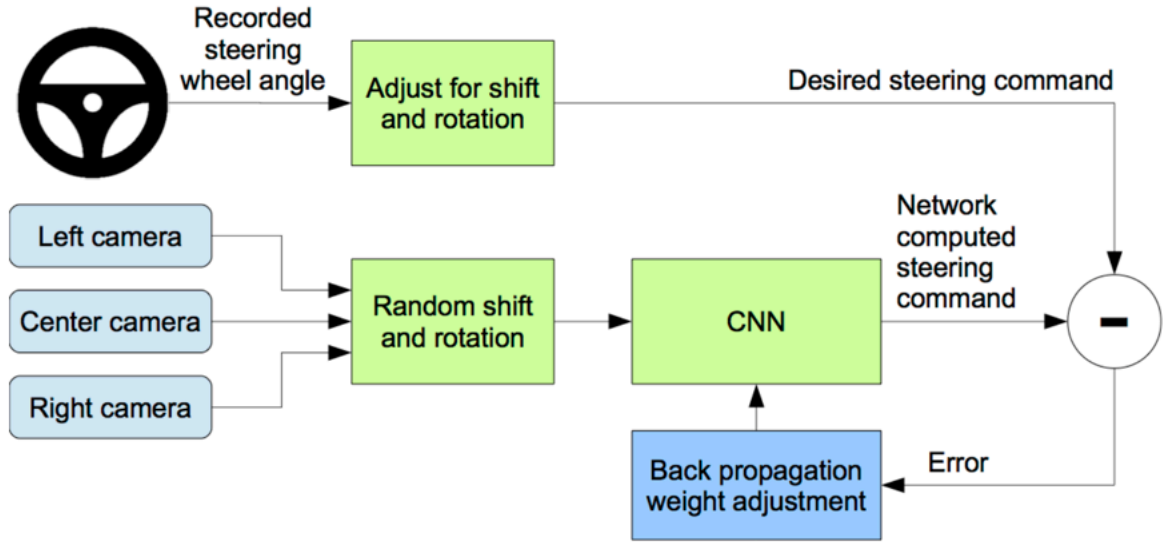
Behaviour Cloning – Training our Convolutional Neural Network (CNN)

We will implement the 9 layer convolutional network which was presented by NVidia self-driving car paper.



Source: http://www.i-programmer.info

Software Design – Supervised Learning



Testing Mode – Autonomous Driving Mode

Once the training is complete and your trained model is saved to "model.h5"

Run the following command: `python drive.py model.h5 run1\`

The Server will start and it will connect to your simulation studio once you have selected the autonomous mode. At this point you should see the agent will start to drive the car if everything is correct.

Furthermore, the second argument run1 will capture images from the autonomous driving. In this case it will save all images with timestamp into run1 directory.

This information is later used by video.py scripting file to create a chronological video of the agent driving. Just run the following command:

```
python video.py run1 –fps 48
```

The video will run at 48 FPS. The default FPS is 60

Conclusion:

I was surprised that in the I was able to use 30 epochs to train the system with CPU under 10 minutes and have a stable driving agent.

Architecture provided by Bojarski, Mariusz, et al, proved to be very useful for this kind of problem. The architecture is very small and can be trained and make it work with relative small datasets. That is incredible!

The provided model solves the given problem and the agent is driving stable on the first track. For the second track, more data information will be needed.

Source Code and Video of the validation is provided on GitHub.

Paper:

References: [1] “Bojarski, Mariusz, et al. "End to End Learning for Self-Driving Cars." arXiv preprint arXiv:1604.07316 (2016)