This section will focus on a very useful technique called behavioral cloning, This section will deal with deep neural networks, feature extraction with convolutional neural networks as well as continuous regression. In summary we are essentially going to be loading a self-driving car simulator provided by open-source by Udacity. We are then going to use the simulator to create our very own training data for our model, by driving a car through the training track inside the simulator. As we drive the car through the simulator, we are going to be taking images at each instance of the drive. These images are going to represent our training dataset and the label for each specific image is going to be the steering angle of the car at that specific instance. We will then show all of these images to our CNN and allow it to learn how to drive autonomously by learning from our behavior as the manual driver. This main variable that our model will learn to adjust is the steering angle of the car at any given instance. it will effectively learn to adjust the steering angle to an appropriate degree based on the situation that it finds itself in. Now after we train our model we are going to evaluate its performance on a completely different testing track where the car will be made to run autonomously if we are able to train the car properly it will perform very well on our second track and will drive on its own.

This behavioral cloning technique is incredibly useful and plays a big role in real life self-driving cars as well. Cras are typically driven around and trained on real roads by manual drivers and they are then trained on the data that they collected on this drive to then clone the behavior of their manual drivers. Therefore after learning this behavioral cloning technique you will effectively be able to understand and even apply the science of self-driving cars.

Collecting Data

We are downloading simulator that will enable us to begin our behavioral cloning process, we are going to start by driving the car through the simulator using our keyboard keys. That way we are able to then ultimately train a CNN to monitor the controlled operation and movement of the vehicle, and depending on how you are driving, it copies of the behavior to then drive on its own in autonomous mode. Cloning your behavior hence the term Behavior cloning.

That being said since it is watching your behavior and copying the data that you are providing it, how will the NN drives is determined by how well you are able to drive the car yourself, To then essentially take your driving skills and reflect it onto your NN.

To download the simulator, go to

github.com/Udacity/self-driving-car-sim

Inside of the GitHub you will find 2 version of it, down load the v1. This simulator is self-executable, so you don’t need to install it.

First we run the simulator in training mode to gather data. Since your vehicle’s driving is only as good as the behavior of the driver who supplied it the data obviously this is not realistic as in real life. A car would be either in a left or a right lane not in the middle but for this project we aim for a successful neural model that drives down the center to ensure a balanced dataset. Once you have finished 3 laps you will do another 3 laps in the reverse direction to gather more data thus helping the model to generalize. Another reason for this is we need to guarantee our data is balanced since this track when you first start has a left turn bias meaning when driving forwards you are mostly steering left and thus driving the car in one direction around the track which will skew our data to one side. The collective data will be biased towards left turns. This creates a problem for the NN as it could bias the model towards always predicting left turns which would have the car become biased towards driving left all the time crashing into the edges. But if we then drive in the opposite direction where we would now be taking mostly right turns the left and right steering angles in our data would become more balanced giving the model a new track to learn from helping the model generalize better another way to eliminate data imbalance is by flipping images, but this is a concept we will talk about later on.

Why we are doing the 3 laps in both directions? Why would this be enough to collect sufficient data?

Well developing accurate machine learning algorithms involves trying and testing with different sets of data until we come up with something that reaches the intended target and so by analyzing the loss and accuracy plots and determining if your model is under fitting or overfitting and then adjusting it accordingly.

You will see that this is a regression type example such that the error metric is the means squared error. If the means squared error is high on both the training and validation sets, then as you know we would be dealing with an underfitting problem, otherwise if the means squared is low on the training but high on the validation then the model would be overfitting. But in such a case a larger dataset can help improve the model as now it has to be flexible enough to fit more data and thus further inhibiting memorization. In other cases you realize that when testing your model whenever it encounters a turn it falls off the track, in such case you can choose to add more specific helpful data by only rerecording specific turns providing the model with more turning data so that it knows the turn when encountering these sharp turns.

We will only train the data on the first track and before we use this to train our model we will do some preprocessing techniques that can help to maximize accuracy.

But what we are exactly recording? While the simulated car is equipped with 3 cameras:

Camera mounted on the left

Camera mounted on the middle

Camera mounted on the right

Of the windshield. Each one recording video footage, where for each image, it collects a value for the steering angle, speed, throttled and break at the current image. The steering angle corresponds to radian values that have been arranged from -1 to +1. Training data is the set of images in each training dataset each set of images has a label, steering angle. This is going to help our network, learn features for each image and associate each image feature with a steering angle with a label. Whenever there is a straight path most likely the steering angle is zero since it is going straight, whereas whenever there is a curvature and the steering angle would be -ve so as to denote a left turn and +ve which denotes a right turn that way when we want to test our model in autonomous mode when the car is located somewhere within the track based on the features that are present within the location within that specific location is it a curved road, is it a straight path. What type of borders are we dealing with. It is going to know what the appropriate angle to steer it and thus appropriately steering the car based on which part of the track it is. In much simpler terms we are using the training images in order to train the NN to predict the appropriate steering angles for when we test the car to drive on its own.

To summarize, this is not a classification problem rather it is a regression based problem as we are trying to predict the steering angle based on a continuous spectrum, hence why we introduced the regression example, which is unsually as normally we would use CNN for classification purposes, such as trying to classify various traffic signs or vehicular detection which are still subjects that are very relevant to self-driving cars but for the purposes of the simulation we are dealing with a regression based problem.