Behavioral Cloning Project Approach

Behavioural Cloning Project is to demonstrate the reuse a pre-trained model and its trained weights to use it in real-time situations to predict and thereby demonstrate the behaviour learned. From the overall point of view, it is cloning the behaviour of humans by machines.

As part of the project, a simulator is provided by udacity to generate the image data and images, and evaluate the model trained.

Approach to solution steps as follows :

* Collect the image data of good driving behavior using simulator. Image data used for this exercise are generated by me using simulator in Training mode.

* Build a model (using python codes), using a convolution neural network in Keras that predicts appropriate model (to be stored in model.h5 file) to ensure smooth autonomous driving.

* Train and validate the model with a training and validation sets, and Test the model successfully that drives around track one without leaving the road.

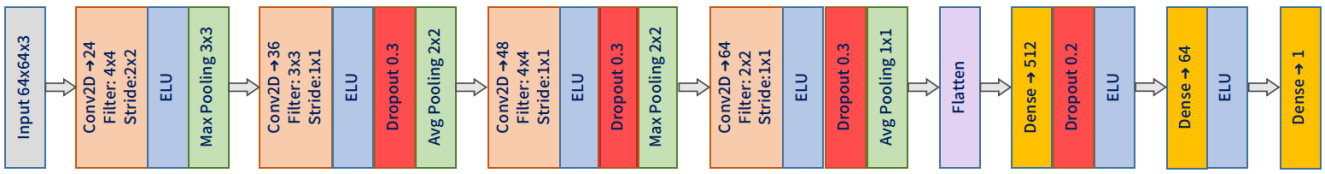
* Summarize the results with detailed report per rubric.

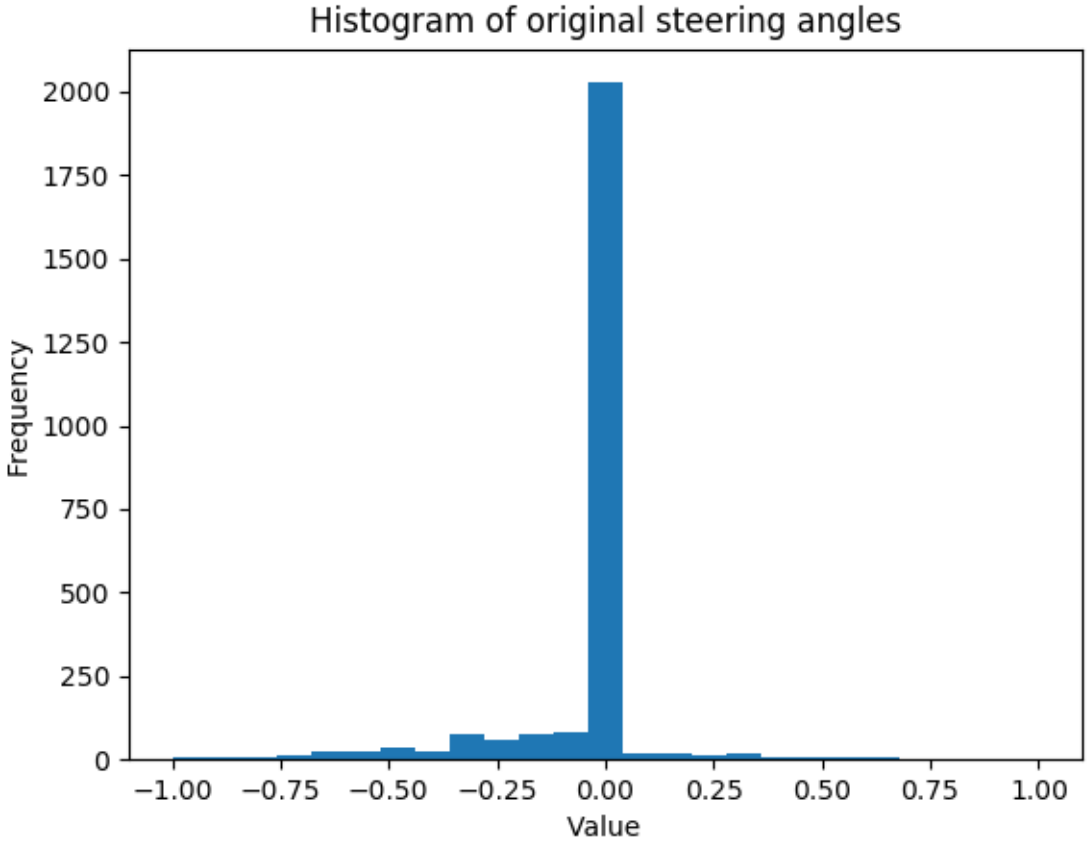
PROJECT SPECIFICATION

Use Deep Learning to Clone Driving Behavior

CRITERIA	MEETS SPECIFICATIONS
Required Fields	
Are all required files submitted?	My project includes the following files: * model.py containing the script to create and train the model * drive.py for driving the car in autonomous mode * model.h5 containing a trained convolution neural network * writeup_report.pdf summarizing the results * video.mp4 - Video recording of my vehicle driving autonomously at least one lap around the track
Quality of Code	
Is the code functional?	Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the track by executing : python drive.py model.h5
Is the code usable and readable?	The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for

	training and validating the model, and it contains comments to explain how the code works.
Model Architecture and Training Strategy	
Has an appropriate model architecture been employed for the task?	Adopted the NVIDIA architecture to use it with the current data, and trained the model and evaluated. My model consists of 4 convolution layers followed by 3 fully-connected layers. The input data is normalised and cropped (model.py line 83). I have changed Relu to ELU (Exponential Linear Unit), which is found to be superior than Relu to introdue non-lineraity. The final architecture arrived is described below.
Has an attempt been made to reduce overfitting of the model?	The convolution layers are followed by dropout layers in order to reduce overfitting. The dataset was balanced and augmented to ensure that the model was not biased towards low steering angles and also generalised to extreme positions. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.
Have the model parameters been tuned appropriately?	The model used an adam optimizer, so the learning rate was not tuned manually (model.py line XXX)
Is the training data chosen appropriately?	I trained using the simulator and collected data.
Architecture and Training Documentation	
Is the solution design documented?	Convolution Neural Networks are the best deep learning models to analyse and learn from images. Details of Architecure discussed below. Initially the Model did not perform well. Based on known architectures (like LeNet, AlexNet), went through the NVIDIA model architecture from the link provided. Adopted the NVIDIA architecture to use it with the current data, and trained the model and evaluated. Then split the existing data into training and validation sets, and trained again. In some of the epochs, the validation accuracy came to 1.000, which is an indication of over fitting the images. So, added the drop out layers in the middle of architecture and able to train and evaluate again. This time, there was some more improvement.

	In the same manner, changed the activation from Relu to ELU (Exponential Linear Unit), which is found to be superior than Relu to introdue non-lineraity. (Refer the article refered about ELU in the references). Similarly, on the pooling layer also, tried Max and Average pooling. And finally changed the optimizer to Adam optimzer, which is considered to be the best optimizer for image processing. All these changed improved the output of the autonomous driving.
Is the model architecture documented?	**Layer-1:** Convolution layer-from 3 dimention to 24 dimension using 4x4 filter and 2x2 stride. Used ELU activation and Max Pooling with 3x3 filter **Layer-2:** Convolution layer-from 24 dimention to 36 dimension using 3x3 filter and 1x1 stride. Used ELU activation, Dropout of 0.3 and Average Pooling with 2x2 filter **Layer-3:** Convolution layer-from 36 dimention to 48 dimension using 4x4 filter and 1x1 stride. Used ELU activation, Dropout of 0.3 and Max Pooling with 2x2 filter **Layer-4:** Convolution layer-from 48 dimention to 64 dimension using 2x2 filter and 1x1 stride. Used ELU activation, Dropout of 0.3 and Avg Pooling with 1x1 filter **Layer-5:** Flatten the data from multi-dimension tuple to 1D tuple **Layer-6:** Fully Connected Layer with the output size of 512, followed by a Dropout of 0.2 and ELU activation **Layer-7:** Fully Connected Layer with the output size of 64, followed by ELU activation **Layer-8:** Fully connected layer with output size of 1 as the final out layer. Since it is a regression problem, no activation function is attached. The final model architecture (model.py lines 134-176) consisted of a convolution neural network with the following layers and layer sizes .. Here is a visualization of the architecture (as below)
	
Is the creation of the training dataset and training process documented?	Generation of data using the simulator was a tricky part in this project. As indicated in the NVIDIA paper referenced, there is a bias towards turning left and being straight with 0.0 steering ange. In order to compensate the poor distribution, enormous amount of data needed to be augmented using pre-processing techniques. In my pre-processing, additional images are generated by flipping images horizontally, adjusting the brightness of the images, cropping the images to remove sky & vehicle bonnett - they are pretty much same and thereby can be called in-variants and not very useful features, resizing the image to 64x64 and normalizing the images. Histogram of original steering angles as given below.

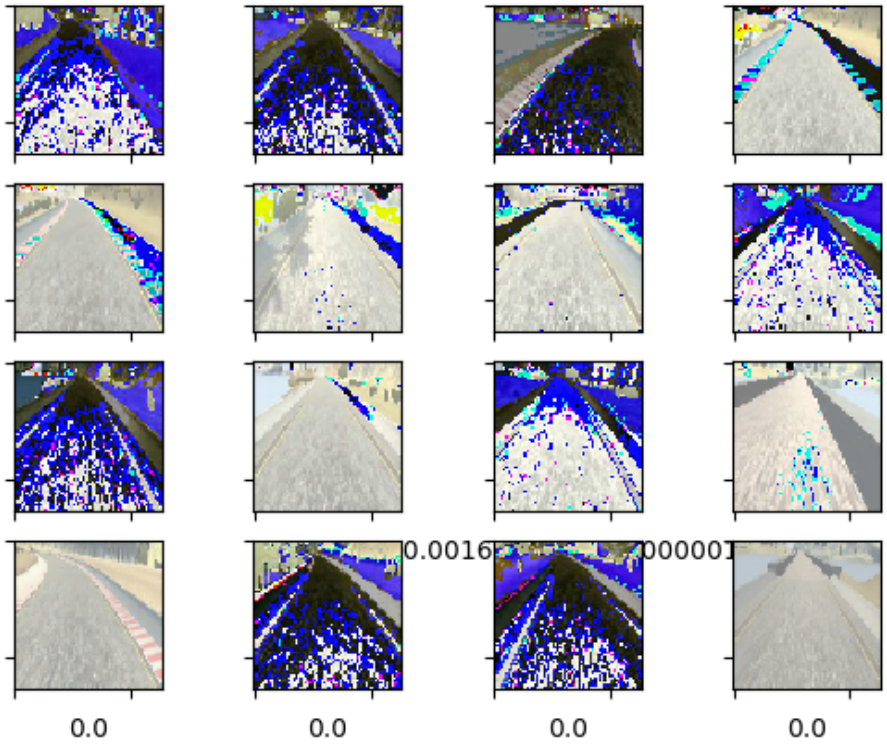


Here is example image of center lane driving:

Here is an example image of left lane driving:

Here is an example image of right lane driving:

Examples of cropAndResize, adjustBrightness, normalizeImage as below



Training: As explained in the dataset generation and model architecture sections, training started with a minimal set of data and a simple model that could run on my laptop . Once I got a reasonably performing model & dataset, the real tweaking started. Played around with following parameters to get the final solution.

- •Batch Size, Epoch size, Validation data size, Learning Rate
- •Optimizers, Activation Functions, Max/AvgPooling, Dropout percentage and even number of layers
- •Dataset sizes, augmentations, number of samples per epoch, number of samples for validation, data crop/resize, etc.

Used fit_generator method of keras to train the model efficiently with available computing resource.

Evaluation: After having the model tested and trained on a reasonable set of training & validation data, the simulation car was able to run on Autonomous mode on its own with out any off-roading. With the current trained model, the simulator was able to run on the track with out any flaws.

The drive.py given in the kit was edited for handling the following changes

- •Included a step to Crop & Resize the input images to fit the images to model's expectaion
- •Included a step to adjust the brightness & normalize the input images
- •Added a logic to adjust 'throttle' to limit/maintain the speed of simulated vehicle.

Simulation

Is the car able to navigate correctly on test data?	Yes. Please check run1.mp4 File as attached in this bundle.