

```
1  ***Behavioral Cloning**
2
3  The goals / steps of this project are the following:
4  * Use the simulator to collect data of good driving
   behavior
5  * Build, a convolution neural network in Keras that
   predicts steering angles from images
6  * Train and validate the model with a training and
   validation set
7  * Test that the model successfully drives around track one
   without leaving the road
8  * Summarize the results with a written report
9
10
11 [//]: # (Image References)
12 [image0]: ./models/model_1/model_1_summary.PNG "Model
   Summary"
13 [image1]: ./models/model_1/tb_losses.png "Final loss"
14 [image2]: ./models/model_1/equalized_shadowed.png "
   Equalized with shadows"
15 [image3]: ./models/model_1/high_validation_loss.png "High
   validation loss"
16 [image4]: ./models/model_1/cropping.png "Image Croppoing"
17 [image5]: ./models/model_1/flipped.png "Flip Image"
18 [image6]: ./models/model_1/shifted.png "Shift Image"
19 [image7]: ./models/model_1/hueadjusted.png "Hue adjusted"
20 [image8]: ./models/model_1/shadowed.png "Shadowed image"
21
22
23 ## Rubric Points
24 ###Here I will consider the [rubric points](https://review
   .udacity.com/#!/rubrics/432/view) individually and
   describe how I addressed each point in my implementation.
25
26 ---
27 ###Files Submitted & Code Quality
28
29 ####1. Submission includes all required files and can be
   used to run the simulator in autonomous mode
30
31 My project includes the following files:
32 * main.py the script to train the model
33 * model.py containing the script to create the model
34 * my_model.py holding various models tested
```

```

35 * drive.py for driving the car in autonomous mode
36 * model_1.h5 containing a trained convolution neural
    network
37 * model_1_weights.h5 the weights from the trained cnn
38 * model_1.mp4 record of driving track1 2 rounds in
    autonomous mode
39 * prepare.py a class for reading and preprocessing images
40 * visual.py a helper class for visualizing results
41 * conv_visualization a class for generating activation
    images for layers
42 * video.py for generating videos from recorded autonomous
    driving images
43 * evaluate.py for running the conv_layer visualizations
44 * generator.py the generator class for training and
    validation set
45 * writeup_report.md this file - summarizing the results
46
47 Some files are only for experimental uses like conv_
    visualization.py/visual.py
48
49 ####2. Submission includes functional code
50 Using the Udacity provided simulator and my drive.py file,
    the car can be driven autonomously around the track by
    executing
51 ```
52 python drive.py models\model_1\model_1.h5 30
53 ```
54
55 The quality chosen was fantastic in window mode on
    1280x960 display size. Steering speed desired was set to
    30.
56
57 ####3. Submission code is usable and readable
58
59 The model.py file contains the code for training and
    saving the convolution neural network.
60 The file shows the pipeline I used for training and
    validating the model, and it contains comments to explain
    how the code works.
61
62 ###Model Architecture and Training Strategy
63
64 ####1. An appropriate model architecture has been employed
65
66 My model is based on the nvidia model and consists of a

```

```
66 convolution neural network with 5x5 and 3x3 filter sizes
   and depths between 24 and 128 (my_model.py lines 59-84)
67
68 The model includes ELU activation layers on each layer to
   speed up learning based on following a discussion on [ML
   Reddit] (https://www.reddit.com/r/MachineLearning/
   comments/3u6ppw/exponential_linear_units_yielded_the_best
   /?st=izx2u5u9&sh=010a4b84)
69 and [paper] (https://arxiv.org/abs/1511.07289)
70
71 The data is normalized in the model using a Keras lambda
   layer (my_model.py code line 63).
72 Instead of dropout layers i used batch normalization on
   channels axis for the first two convolutional layers to
   make the network more robust to bad initialization.
73
74 #####2. Attempts to reduce overfitting in the model
75
76 The model contains 2 batch normalization layers (my_model
   .py lines 64,66) which should overfitting less likely.
77
78 The model was trained and validated on different data
   sets to ensure that the model was not overfitting (model.
   py line 102) by splitting
79 from the test set wit a factor of 0.2.
80 The model was tested by running it through the simulator
   and ensuring that the vehicle could stay on the track for
   multiple laps.
81
82 #####3. Model parameter tuning
83
84 The model used an adam optimizer, so the learning rate
   was not tuned manually (my_model.py line 82).
85
86 #####4. Appropriate training data
87
88 Training data was chosen to keep the vehicle driving on
   the road. I used a combination of center lane driving and
   additional weak spot image recording.
89 For details about how I created the training data, see
   the next section.
90
91 ###Model Architecture and Training Strategy
92
93 #####1. Solution Design Approach
```

```
94
95 The overall strategy for deriving a model architecture
   was to implement a well known model like nvidia and
   finetune where it's necessary.
96 As dataset at first only the provided udacity dataset was
   used.
97
98 In order to gauge how well the model was working, I split
   my image and steering angle data into a training and
   validation set.
99 I chose to validate on only the center images where i had
   exact labels and not on the steering corrected
   additional left and right images.
100
101 I found that my first model had a low mean squared error
   on the training set but a high mean squared error on the
   validation set. This implied that the model was
   overfitting.
102
103 ![High validation loss][image3]
104
105 To combat the overfitting, I modified the model and
   introduced batch normalization layers
106
107 The final step was to run the simulator to see how well
   the car was driving around track one. As some spots were
   difficult and the car left the road i
108 created additional datasets made on my own.
109
110 At the end of the process, the vehicle is able to drive
   autonomously around the track without leaving the road on
   an endless loop.
111
112 ####2. Final Model Architecture
113
114 The final model architecture (my_model.py lines 59-84)
   consisted of a convolution neural network with the
   following layers and layer sizes:
115
116 ![Model summary][image0]
117
118 ![Final loss][image1]
119
120 ####3. Creation of the Training Set & Training Process
121
```

```
122 I used the provided udacity dataset. After identifying a
    few weak spots where the car left the track, i decided to
    record additonal recovery images.
123 Doing so with the keyboard or gamecontroller failed badly
    as the steering angle was too fixed and could not be so
    well controlled, which was
124 in the end the most important point of the whole project.
    Only using the mouse steering control on the beta
    simulator let me
125 drive the track in a way that i good continous steering
    result on lots of small degree steps
126
127 Creating recovery images from off track on the road again
    was in my set not necessary for track one. Instead i
    cherry picked good steering angles for the weak spots
128 and added them to the training data.
129
130 To augment the data set, i randomly flipped the images
    and measurements as well as shifted them vertically or
    taking left or right camera image instead
131 of center images. when taking left or right camera images
    , the steering angle has been corrected by a value of 0.
    25 for right images and -0.25 for left images.
132
133 Example of cropping the image from 160x320 to 66x200:
134
135 ![Cropping][image4]
136
137 Example of flipping the image vertically:
138
139 ![Flipping][image5]
140
141 Example of shifting the image:
142
143 ![Shifting][image6]
144
145 Example of adjusting the hue of the image:
146
147 ![Hue adjust][image7]
148
149 Example of adding a shadow section to the image:
150
151 ![Shadowing][image8]
152
153 The image input itself has been 0/1 normalized on the
```

153 keras lambda layer (my_model.py line 63).
154 Additionally i added hue randomness and randomly
generated shadow sections to overcome the hurdles in the
challenge videos. at the end this did not succeed,
155 so i only was able to make track 1 working.
156
157 A test and validation set generator (generator.py) has
been created to batch work on the fly all data. Since
using the nvidia model quickly ran out of memory
158 even on my gtx 1080. Also i resized the images to the
nvidia chosen size of 66x200, as using 100x320 made out
about 12 million parameters, where the resized one
159 needed only 1.6 million which lowered the size of the
model from about 140MB to 14MB.
160
161 I finally randomly shuffled the data set and put 0.2 of
the data into a validation set.
162
163 I used this training data for training the model. The
validation set helped determine if the model was over or
under fitting.
164 The ideal number of epochs was 3 as evidenced by an
introduced early stopping layer with patience 1.
165
166 I used an adam optimizer so that manually training the
learning rate wasn't necessary.
167
168 additionally i added also a tensorboard layer (model.py
line 111) to experiment with it's logging capabilities
and generated a loss diagram out of it.
169 The model is unfortunately so big that the tensorboard
dashboard was not able to handle it's size.
170
171 To get always the best checkpoint i added a checkpoint
callback which always compared the current with the last
run and saved the best out of it, as
172 well as a checkpoint file in addition to the best fit.
173
174