Behavioral Cloning Project

By Thomas Tartière

Behavioral Cloning Project

In this project we implement a deep learning model to learn the correct steering angle for a car, using images taken from a camera at the front of the car.

The goals / steps of this project are the following:

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

1. Files Submitted & Code Quality

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.pdf summarizing the results

2. Model Architecture and Training Strategy

I use the NVIDIA architecture as my deep learning model. It consists of a normalization layer, followed by 5 convolution layers, followed by 4 fully connected layers.

The model includes RELU layers to introduce nonlinearity and the data is normalized in the model using a Keras lambda layer.

3. Attempts to reduce overfitting in the model

The data was randomly split between training (80%) and cross validation (20%) datasets in order to prevent orverfitting. I also trained the model on images coming from the two tracks available in order to help it to generalize.

4. Model parameter tuning

The model used an Adam optimizer, so the learning rate was not tuned manually.

5. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used the following combination:

- 2 loops of center lane driving
- 1 loop of "reverse" driving
- 1 loop recovering from the left and right sides of the road
- 2 loops of center lane driving with the second track available

For details about how I created the training data, see the next section.

6. Model Architecture and Training Strategy

a. Solution Design Approach

The chosen model architecture was established by NVIDIA for their Self-Driving Car program.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set.

The first experiments led to a good cross validation error but the car was driven properly through some of the steep curves. In order to improve the driving behavior, I spent time enhancing the training data set by focusing on curves and recovery from the sides of the road.

At first, using images from the left and right side cameras did not improve the driving so I had to optimize the correction factor and find the right value in order to use this additional data and improving the car driving ability.

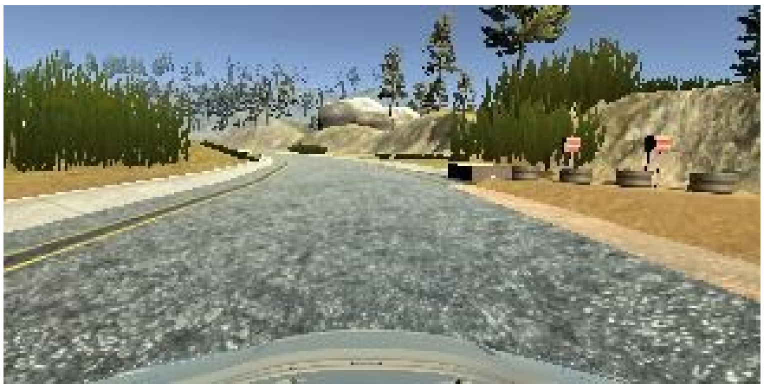
At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

b. Final Model Architecture

The final model architecture the NVIDIA architecture as my deep learning model. It consists of a normalization layer, followed by 5 convolution layers, followed by 4 fully connected layers.

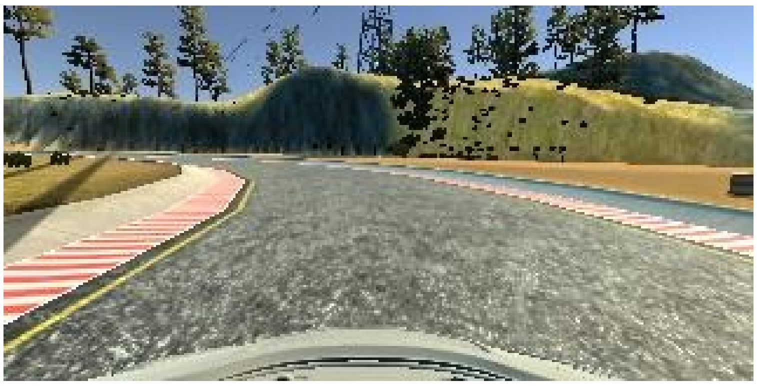
c. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:

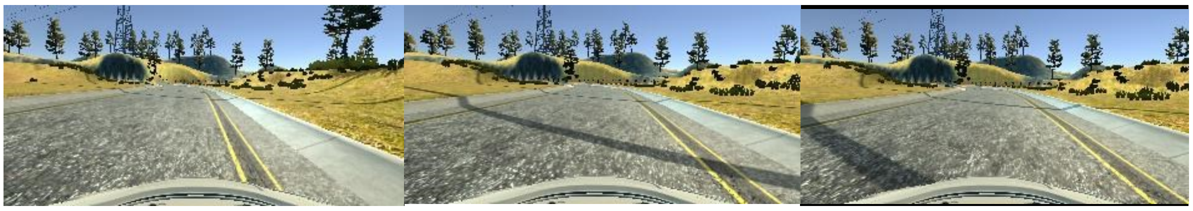


As the track as a bias towards turning left, I then recorded one lap of center lane driving going in the opposite direction.

As the car wasn't driving properly through curves, I then generated additional images focusing on center lane driving in curves, as shown below.



I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to get back to the center. These images show what a recovery looks like starting from the right side of the road to the center:



Then I repeated this process on track two in order to get more data points.

To augment the data set, I also flipped images and angles thinking that this would help the model to generalize

After the collection process, I had 21130 data points, from 3 cameras (left, right, center). Considering the flipping, this gave 84,520 images for training and cross validation. I then preprocessed this data by cropping the top part of the images in order to reduce the image of changing landscape.

I finally randomly shuffled the data set and put 20% of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 3 as evidenced by the cross validation score increasing after 3 epochs. I used an Adam optimizer so that manually training the learning rate wasn't necessary.