
Behavioral Cloning Project

The goals 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

Writeup

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Files and Code Submitted

1. Required files to run the simulator in autonomous mode

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.md and writeup.pdf summarizing the results
- data points generated in training mode, in the "./data/IMG" directory and in "./driving_log.csv"
- images generated in autonomous mode of one run around the track, in the "./run1" directory
- video of the images in "./run1.mp4"

2. How to run the code

Using the Udacity provided simulator and drive.py file, the car can be driven autonomously around the track by executing

```
python drive.py model.h5
```

while in autonomous mode in the simulator. To generate the model.h5 file, the code

```
python model.py
```

has to be run after a training mode session in the simulator, typically 2-3 times around the track.

3. Code description

The model.py file contains the code for training and saving the convolution neural network in model.h5. The pipeline I used for training and validating the model is in code lines 43 to 56. The code uses the NVIDIA model as shown in the comment above the pipeline.

Model Architecture and Training Strategy and Documentation

1. Model architecture

The model consists of a convolution neural network with 5x5 filter sizes and depths between 24 and 1164 (model.py lines 43-56). The model includes RELU layers to introduce nonlinearity (code lines 46-50), and the data is normalized in the model using a Keras lambda layer (code line 44).

2. Reducing overfitting in the model

The model was split into training and validation sets (code line 59). From initially running 7 epochs on a simple dense network with an 80%-20% training/validation split, I tried various choices of number of epochs from 1 to 5 on the Le Net network and NVIDIA network. I also tried a few training-validation splits between 70%-30% and 80%-20%. The final choice was 2 epochs on the NVIDIA network with a split of 70%-30%. This avoids a validation loss going back up after going down initially, and worked better for the autonomous run. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

The model used an adam optimizer, so the learning rate was automatically tuned (model.py line 58). After some tuning as explained above, finally the number of epochs was set at 2 and the training-validation split was set at 70%-30%.

4. Training data

I generated training data by running the simulator in training mode and driving around the track about three times, keeping the vehicle on the road.

To capture good driving behavior, I recorded three laps on track one using center lane driving and some recovering from the left and right sides of the road. An example image of center lane driving is in figure “center lane” below.

Figure 1: center lane

I recorded the vehicle recovering from the right and left sides of the road back to center so that the vehicle would learn to get back on track. Example images are in figures “recovery right” and “recovery left” below. The result of the recovery is in the figure “recovered” below.

After the collection process, I had 13,605 data points that I randomly split into a training set and a validation set in the proportion 70%-30% respectively. The validation set was used to observe if the parameters chosen (see above) resulted in over- or under-fitting.

Simulation

Link to video of autnoumous run around the track.

Figure 2: recovery right

Figure 3: recovery left

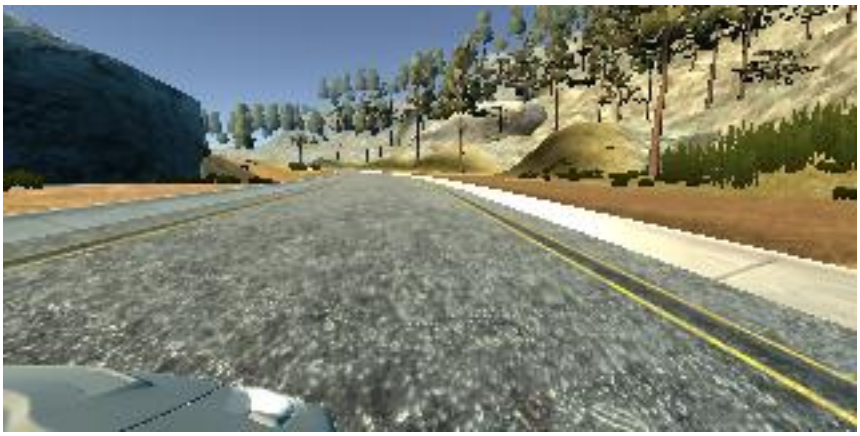

Figure 4: recovered