UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Use Deep Learning to Clone Driving Behavior

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hi,

This was an excellent submission! The pipeline performs exceptionally well as the car can navigate the entire track without any errors. Keep it up!

Required Files

The submission includes a model.py file, drive.py, model.h5 a writeup report and video.mp4.

Quality of Code

The model provided can be used to successfully operate the simulation.

The code is functional and I could run it on the first track successfully.

The code in `model.py` uses a Python generator, if needed, to generate data for training rather than storing the training data in memory. The `model.py` code is clearly organized and comments are included where needed.

Good job using a `fit_generator` here. The code is well organized and commented clearly.

Model Architecture and Training Strategy

The neural network uses convolution layers with appropriate filter sizes. Layers exist to introduce nonlinearity into the model. The data is normalized in the model.

Nice job in normalizing the data and using the appropriate layers and hyperparameters for the model architecture.

Train/validation/test splits have been used, and the model uses dropout layers or other methods to reduce overfitting.

Train/test splits have been used along with dropout layers to prevent overfitting in the model. You could also try other techniques like adding pooling layers, batch normalization, L2 regularizers, or early stopping.

Learning rate parameters are chosen with explanation, or an Adam optimizer is used.

Good job using an Adam optimizer.

Further Reading
You can read more on gradient descent optimisation algorithms here:
http://sebastianruder.com/optimizing-gradient-descent/

Training data has been chosen to induce the desired behavior in the simulation (i.e. keeping the car on the track).

Nice work collecting the data yourself and describing the data collection process.

Architecture and Training Documentation

The README thoroughly discusses the approach taken for deriving and designing a model architecture fit

for solving the given problem.

You've done very well in discussing how the model was derived and designed. You can also take a look at the architecture given by comma.ai.

The README provides sufficient details of the characteristics and qualities of the architecture, such as the type of model used, the number of layers, the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.

The model architecture is easy to understand and is presented very well.

The README describes how the model was trained and what the characteristics of the dataset are. Information such as how the dataset was generated and examples of images from the dataset must be included.

This is a thorough discussion of the process that was followed to collect the training data, and it is well supported with example images. Apart from flipping the images, you can also consider doing brightness augmentation:

```
temp = cv2.cvtColor(im, cv2.COLOR_RGB2HSV)

# Compute a random brightness value and apply to the image
brightness = 0.25 + np.random.uniform()
temp[:, :, 2] = temp[:, :, 2] * brightness

im = cv2.cvtColor(temp, cv2.COLOR_HSV2RGB)
```

You'll find some other augmentation techniques on this link.

Simulation

No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle).

The simulation is very impressive as the car could drive around the track smoothly without going over the lane lines even once. It would be a good idea to evaluate the pipeline on the other track to see how well it can generalize.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review