UDACITY

PROJECT

Use Deep Learning to Clone Driving Behavior

A part of the Self Driving Car Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

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 submitted model can be successfully used to navigate a car in the simulated environment.

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.

You did a good job implementing the model.

The generator is not used what may lead to memory overload. Moreover, it would be good to separate blocks of code into reusable methods and provide Docstring documentation that conforms to Python conventions:

Docstring Conventions
Example Google Style Python Docstrings
Google Python Style Guide

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.

You used convolutional layers with input normalised in the Lambda layer. Rectified Linear Units were built into the architecture of the neural network as a nonlinear activation function.

You may want to read about other activation methods:

- Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)
  https://arxiv.org/pdf/1511.07289v1.pdf
- Commonly Used Activation Functions
  http://cs231n.github.io/neural-networks-1/#actfun

- Advanced Activations Layers
  https://keras.io/layers/advanced_activations/

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

Dropout layers were used to reduce overfitting. The dataset was split into train/validation sets to make parameter choices and monitor overfitting.

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

The Adam optimiser was used for stochastic optimisation.

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

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 described the process that led you to the final implementation of your model including the range of evaluated parameters, the reasoning for choosing these sets of parameters for testing, and preprocessing that you applied.

I recommend watching the following talk given by Andrej Karpathy at Deep Learning School in Stanford to revise the design of convolutional neural network architectures if needed: https://www.youtube.com/watch?v=u6aEYuemt0M

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 final model is presented.

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 should be included.

The README contains all required information regarding the training and the dataset including a source of the dataset and number of data points.

Optional video about training neural networks (and much more):

- Nuts and Bolts of Applying Deep Learning (Andrew Ng)
  https://www.youtube.com/watch?v=F1ka6a13S9I

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 car drives around the track without any issues. Well done.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ