UDACITY

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

Use Deep Learning to Clone Driving Behavior

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

This was an excellent effort!
Congratulations on successfully completing this project! ✨

You may also want to keep the following resources as a reference -

- https://www.youtube.com/watch?v=rpxZ87YFg0M
- http://selfdrivingcars.mit.edu/
- http://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf
- http://jacobgil.github.io/deeplearning/vehicle-steering-angle-visualizations
- http://medium.com/udacity/teaching-a-machine-to-steer-a-car-d73217f2492c
- http://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9
- Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

Keep up the good work and all the best for your future projects!

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.

Great job! The car runs successfully in the simulator!

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.

- Well done using a generator. This is especially helpful when training using a CPU.
  In case you want to gain a better intuition about Python generators, you may refer to
  https://www.youtube.com/watch?v=bD05uGo_sVI
- The code is clearly organized and comments have been used where appropriate.

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.

- The network has an adequate number of convolution layers.
- Activation function has been used to introduce nonlinearity in the model. Please note that a faster alternative to the `relu` activation is the `elu`. You may refer to https://arxiv.org/pdf/1511.07289v1.pdf
- Well done normalizing the data in the model.

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

- The dataset has been split appropriately.
- Well done using dropout to reduce overfitting. For a detailed analysis of dropouts, you may refer to
  https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/

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

Good job using an Adam optimizer.

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

The appropriate training data has been chosen to achieve the desired result.

- Well done using the appropriate data augmentation techniques.
- Using a correction factor of 0.1 is justified here.
- For further examination of the images, you may plot all independent channels of information in the color spaces RGB, YUV, HSV, grayscale.

Architecture and Training Documentation

The README thoroughly discusses the approach taken for deriving and designing a model architecture fit for solving the given problem.

Great job discussing your approach thoroughly.

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. Here is one such tool for visualization.

Using the NVIDIA based architecture is perfectly justified in this situation.
You may also use Keras to visualize your model - https://keras.io/visualization/

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.

The training of the model has been adequately described alongwith appropriate images. Nice work here!

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).

You've done a great job of coming up with an architecture that drives the car safely in the simulator!

DOWNLOAD PROJECT

RETURN TO PATH