Project: Behavioral Cloning

Overview

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 the collected dataset
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

Model Architecture and Training Documentation

My model is mostly based on NVIDIA architecture, which consists of 5 CNN layers and 4 fully connected layer. Among the 5 CNN layers, 3 have 5x5 filter sizes and 2 have 3x3 filter size. RELU is the activation function of these layers to introduce nonlinearity. I reduce overfitting by controlling epochs so dropout is no included in model architecture.

At the beginning of the model, data is normalized in the model using a Keras lambda layer and cropped by removing top 70 lines and bottom 25 lines of image data.

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

Layer	Input	Output	Details
Cropping Layer	160x320x3	65x320x3	Trim image to only see section with road
Layer 1 Convolutional + activation	65x320x3	31x158x24	Valid padding with ReLU activation (2, 2)subsample
Layer 2 Convolutional + activation	31x158x24	14x77x36	Valid padding with ReLU activation (2, 2)subsample
Layer 3 Convolutional + activation	14x77x36	5x37x48	Valid padding with ReLU activation (2, 2)subsample
Layer 4 Convolutional + activation	5x37x48	3x35x64	Valid padding with ReLU activation (1, 1)subsample

Layer 5 Convolutional + activation	3x35x64	1x33x64	Valid padding with ReLU activation (1, 1)subsample
Layer 6 Fully connected	2112	100	Flattened output from previous layer as input without activation
Layer 7 Fully connected	100	50	None activation
Layer 8 Fully connected	50	10	None activation
Layer 9 Fully connected	10	1	Output

The writeup report 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.

My final model uses the official data provided by Udacity. As for training strategy, I have tried these things:

- generate my own dataset by playing the game more than 3 laps
- augment data by flipping images
- use a generator to load data and preprocess it on the fly, in batch size portions to feed into my model

The 1st sharp turn after the bridge is really hard to deal with. No mater how I finetune the parameters (epochs and batch size), the car still has high possibility to crash on the sharp turn.

And after browsed Student Hub to see if other people encounter problems the same as me, I came to realize that drive.py sends RGB images to the model but cv2.imread() reads images in BGR format. So I use matplotlib.image.imread() instead and the car was able to stay on the track.

Reference

1. Udacity Self-Driving Car Engineer Project: Behavioral Cloning (15) Even More Powerful Network
2. https://github.com/jychstar NanoDegreeProject Repository