Behavrioal Cloning Project

The goals / steps 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

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_report.pdf summarizing the results

Code

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

Note: My computer's os is Windows. I can run the simulator on my computer but I cannot run my code because my Windows version is not able to run a docker. All my code were test and compile on AWS, which only has command line. But I certainly build an efficient model. The model.h5 is my stored model.

Model result :

The final validation metrics result is :[136.04831166880754, 0.010788381742738589]

The result is reasonable.

Model Architecture and Training Strategy

An appropriate model arcthiecture has been employed

Layer1: convolutional neural network, input_shape=(160,320,3)

Layer2: relu activation

Layer3: convolutional neural network

Layer4: relu activation

Layer5: convolutional neural network

Layer6: relu activation

Layer7: convolutional neural network

Layer8: relu activation

Layer9: Flatten layer, flatten the network

Layer10: Dropout layer, prob=0.5, to prevent overfitting

Layer11: Fully connected layer, dense the net to 100

Layer12: relu activation

Layer13: Dropout layer, prob=0.5, to prevent overfitting

Layer14: Fully connected layer, dense the net to 50

Layer15: relu activation

Layer16: Fully connected layer, dense the net to 10

Layer17: relu activation

Layer18: Fully connected layer, dense the net to 1

Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually. [model.py line156]

The reason why I choose adam optimizer is that I used adam optimizer before and it performs well. It also appears in the lecture.

Appropriate training data

I recorded some files by using the simulator. However, I found their quality were not good. Then I use the sample data to train my model. Before training, I fetched the data from driving_log.csv and the IMG folder. After randomly

shuffled the data, I did a normalization operation to those training samples. [model.py line 80 to 86, line 96]

At first ,I choose 80% of the img data as my training data and 20% as my validation data. However, this would course an memoryError on AWS when reading the validation file. So I changed the the ratio to 9:1 (model.py line 96 &line 176)

A small but an important point is that the driving_lot.csv has a header row, which would cause an error if we did not noticed that. Before load the data to the model, this line should be deleted.[model.py line 41]

Solution Design Approach

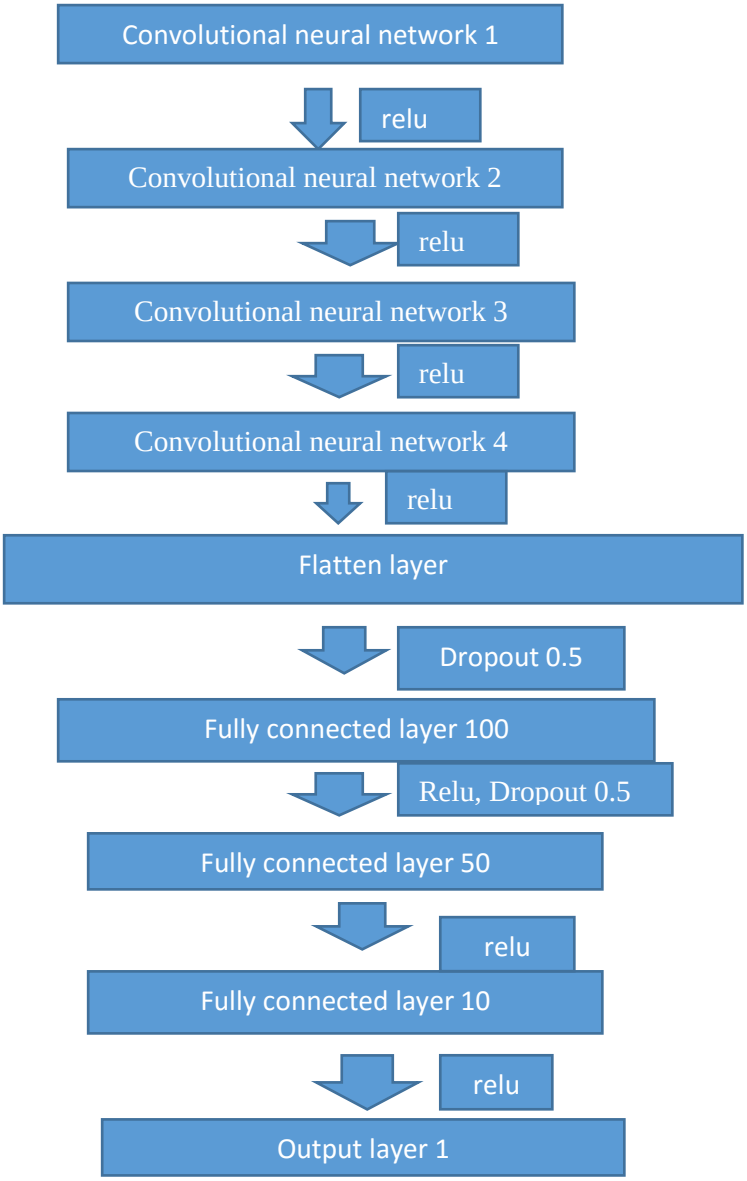
1. Based on the recommendation from other cohorts, I choose to train 5 epochs. And my batch size is 120, which is a reasonable value when training images. Since the training is just like the traffic sign classifier project, I did not change so much on those parameters. As a result, my code successfully operates on the simulation samples.

2. The filters I used in the convolutional neural net are 5*5(conv 1,2,3) and 3*3(conv4)

3. The model uses dropout layers or other methods to reduce overfitting.[line 140, line 144]

4. The generator. This is an essential part for the whole project. Firstly, I have no idea what to do about this. After going through those questions on the forum, I have some deeper understanding upon this. The mechanism of the my generator is actually continuously providing data to the training pipline. Each time the amount it provide is a batch size number. It just like feeding data to GPU, which process data at the same time as much as possible.

Final Model Architecture



Some reflections:

1. The most challenge part in this project is actually the generator part I think. At first, I did not realized the importance of generator. My first generator went wrong but the I get an error in other part, which confused me for a while.

2. I did not handle the image distortion before I trained them. This might be a inappropriate point.

3. The whole project can be improved on the architecture and data preprocessing part. I have seen some other cohorts posted their result on YouTube and their results are accurate.

4. The softmax activation is not suitable for this project. When I was trying to apply softmax activation in this project. It would caused error, which is about the tensor dimension. But when I choose relu, this error was gone.