Udacity P3 Project Cloning

1.Model Architecture and Training Strategy

Model uses the architecture proposed by Nvidia team. Model uses a mix of convolution layer, maxpooling layer and finally uses a feed forward layer as a classifier.

The activation layer used in the case is ReLU which makes the system faster and better than sigmoid, tanh or logistic activation function.

There is a use of dropout feature to improve generalization and reduce overfitting.

Normalization is used before the input is fed to the convolution neural network model.

2. Attempts to reduce overfitting in the model

There are use of dropout layer which is used for improving generalization error and overfitting.

Number of epochs is restricted to 5 to avoid overfitting

There are two sets of data sets, one for training and one for validation.

To prevent overfitting, validation set was not used for training purpose.

3. Model parameter tuning

The optimizer used in the case is Adam optimizer.

The learning rate used is 1e-03.

The number of epochs used were 5.

4. Appropriate training data

I used a mix of center lane driving image, left driving image and right driving image.

Images are horizontally flipped with a probability of 20 percentage.

Randomly images are chosen from center image, left image and right image with a probability of 0.33 each.

The left and right image were offset by a margin of +/- 0.2 respectively.

As mentioned earlier, validation set was used to test the model.

Model Architecture and Training Strategy

1. Solution Design Approach

The data used was the data obtained from udacity site. This data was split into training and validation data. mean square error was used as measure for model accuracy. to check for overfitting/ underfitting, the model was tested for validation error and similar performance was obtained for both the data sets. There was no need for additional data as the udacity data was sufficient.

The system performs god for the test environment.

2. Final Model Architecture

The code of the model is given below:

 # Model architecture

model = models.Sequential()

model.add(layers.core.Lambda(lambda x: (x / 127.5 - 1.), input_shape = (160,320,3)))

model.add(layers.convolutional.Convolution2D(conv_layer_1, 5, 5, activation=activation_func))

model.add(layers.pooling.MaxPooling2D(pool_size=pooling_size))

model.add(layers.convolutional.Convolution2D(conv_layer_2, 5, 5, activation=activation_func))

model.add(layers.pooling.MaxPooling2D(pool_size=pooling_size))

model.add(layers.convolutional.Convolution2D(conv_layer_3, 5, 5, activation=activation_func))

model.add(layers.pooling.MaxPooling2D(pool_size=pooling_size))

model.add(layers.convolutional.Convolution2D(conv_layer_4, 3, 3, activation=activation_func))

model.add(layers.pooling.MaxPooling2D(pool_size=pooling_size))

model.add(layers.core.Flatten())

model.add(layers.core.Dense(neuron_100, activation=activation_func))

model.add(layers.core.Dropout(dropout_rate))

model.add(layers.core.Dense(neuron_50, activation=activation_func))

model.add(layers.core.Dropout(dropout_rate))

model.add(layers.core.Dense(neuron_10, activation=activation_func))

model.add(layers.core.Dense(neuron_1))

model.compile(optimizer=optimizers.Adam(lr=learning_rate), loss=loss_type)

Total number of parameters involved is 861,691

The training accuracy and validation accuracy is given below:

Epoch 1/5

19987/20000 [==============================>.] - ETA: 3s - loss: 0.0210

Epoch 00000: saving model to C:/Users/NIKHIL XAVIER/git/Self-Driving-Car/P3/CarND-Behavioral-Cloning-P3-master/checkpoint2/check-00-0.0118.hdf5

20043/20000 [==============================] - 5097s - loss: 0.0210 - val_loss: 0.0118

Epoch 2/5

19971/20000 [==============================>.] - ETA: 8s - loss: 0.0142 Epoch 00001: saving model to C:/Users/NIKHIL XAVIER/git/Self-Driving-Car/P3/CarND-Behavioral-Cloning-P3-master/checkpoint2/check-01-0.0110.hdf5

20029/20000 [==============================] - 6355s - loss: 0.0141 - val_loss: 0.0110

Epoch 3/5

19986/20000 [==============================>.] - ETA: 3s - loss: 0.0129 Epoch 00002: saving model to C:/Users/NIKHIL XAVIER/git/Self-Driving-Car/P3/CarND-Behavioral-Cloning-P3-master/checkpoint2/check-02-0.0126.hdf5

20044/20000 [==============================] - 4610s - loss: 0.0129 - val_loss: 0.0126

Epoch 4/5

19996/20000 [==============================>.] - ETA: 1s - loss: 0.0124 Epoch 00003: saving model to C:/Users/NIKHIL XAVIER/git/Self-Driving-Car/P3/CarND-Behavioral-Cloning-P3-master/checkpoint2/check-03-0.0116.hdf5

20051/20000 [==============================] - 6308s - loss: 0.0124 - val_loss: 0.0116

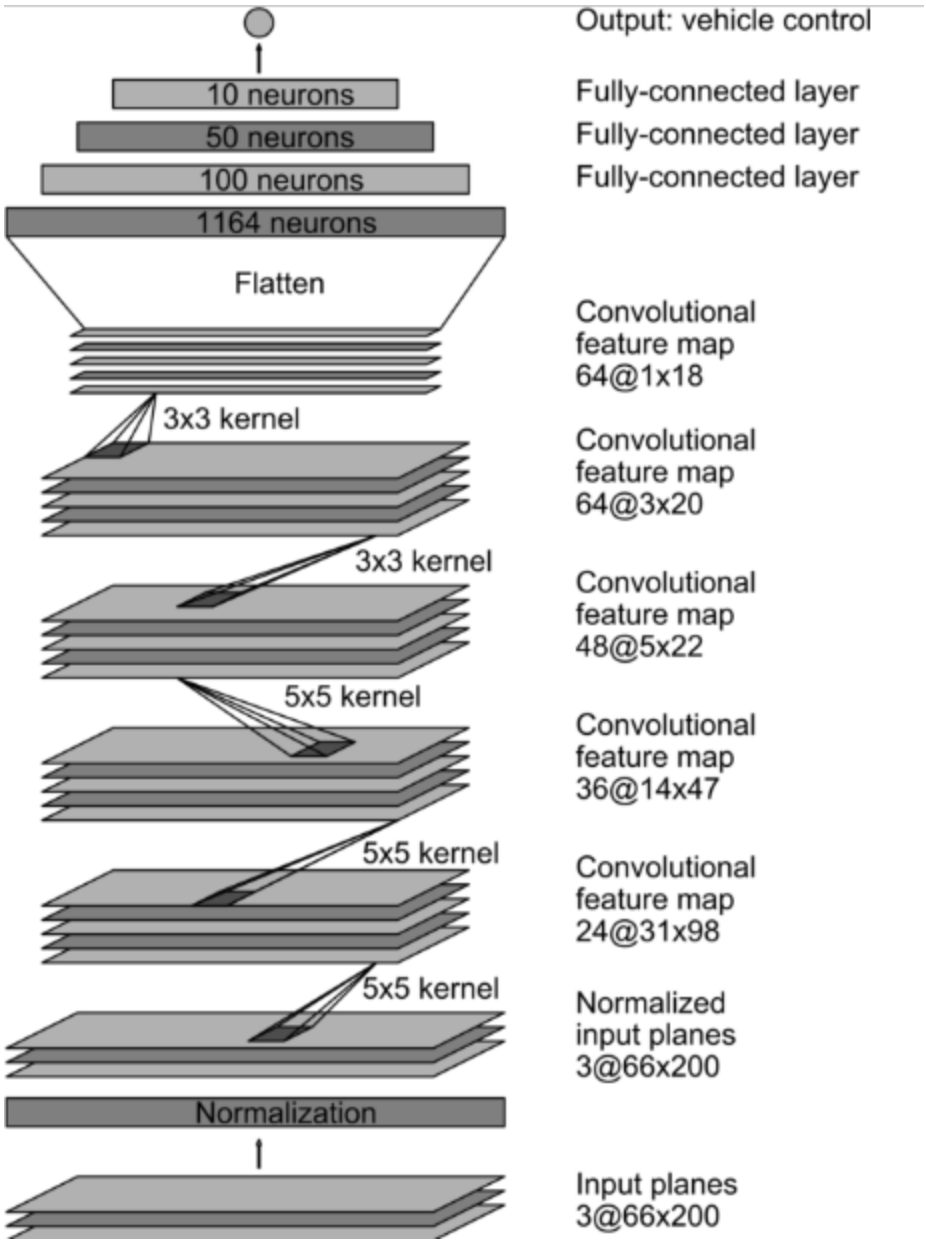
Epoch 5/5

19965/20000 [==============================>.] - ETA: 9s - loss: 0.0116 Epoch 00004: saving model to C:/Users/NIKHIL XAVIER/git/Self-Driving-Car/P3/CarND-Behavioral-Cloning-P3-master/checkpoint2/check-04-0.0101.hdf5

20024/20000 [==============================] - 5492s - loss: 0.0116 - val_loss: 0.0101

Saved model to disk

The image for the system architecture is given below.



3. Creation of the Training Set & Training Process

The training data is normalized before the data is fed into the CNN model. Data from center lane, right lane and left lane are used with +/- 0.2 offset and fed into the model. Data is randomly sampled from center lane, left lane and right lane by a probability of 33 percentage.

INput data is augmented to generate more data samples. Data is shiffled before being used. Additional data is created by randomly flipping images

horizontally. There was also an option to shear image with 50 percentage probability.

Finally, image is trimmed to use only relevant information for training the model.

Image size is 160x320x3 and resized to a particular fixed size. Finally validation dataset was used for testing with optimizer used as Adam optimizer. Shuffling is also performed so as to lose the inherent trend in the training dataset.

_____

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 160, 320, 3)	0	lambda_input_1[0][0]
convolution2d_1 (Convolution2D)	(None, 156, 316, 24)	1824	lambda_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 78, 158, 24)	0	convolution2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 74, 154, 36)	21636	maxpooling2d_1[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 37, 77, 36)	0	convolution2d_2[0][0]
convolution2d_3 (Convolution2D)	(None, 33, 73, 48)	43248	maxpooling2d_2[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 16, 36, 48)	0	convolution2d_3[0][0]

convolution2d_4 (Convolution2D)	(None, 14, 34, 64)	27712	maxpooling2d_3[0][0]
maxpooling2d_4 (MaxPooling2D)	(None, 7, 17, 64)	0	convolution2d_4[0][0]
flatten_1 (Flatten)	(None, 7616)	0	maxpooling2d_4[0][0]
dense_1 (Dense)	(None, 100)	761700	flatten_1[0][0]
dropout_1 (Dropout)	(None, 100)	0	dense_1[0][0]
dense_2 (Dense)	(None, 50)	5050	dropout_1[0][0]
dropout_2 (Dropout)	(None, 50)	0	dense_2[0][0]
dense_3 (Dense)	(None, 10)	510	dropout_2[0][0]
dense_4 (Dense)	(None, 1)	11	dense_3[0][0]

Remarks:

I was able to run the car twice till 90 percent of the track. Finally I was able to run it once completely autonomously through out the track.

Reference:

https://github.com/priya-dwivedi