Driving behavior cloning project report

Overview

This project utilizes deep learning network to drive a car in virtual enviroment.

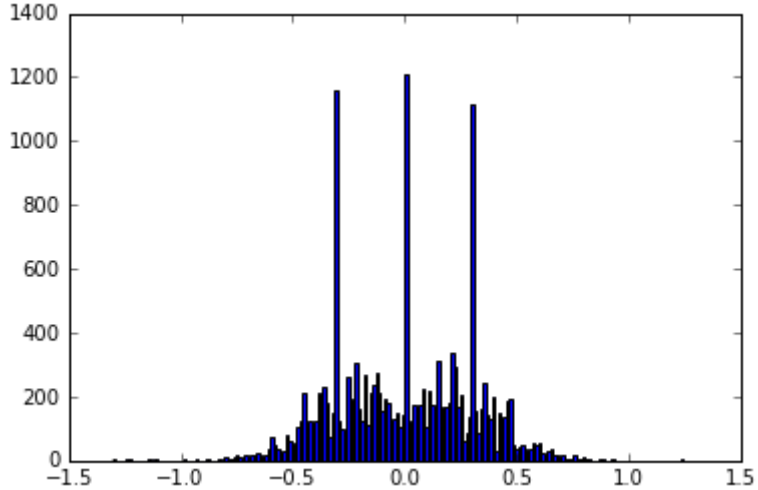
1. Training samples was downloaded from Udacity's website. I couldn't make it on my own data.
2. Nvidia's model was used. Small modification was made by adding a dropout layer.
3. Two tracks were tested and successful(not with the beta simulator but with the default simulator). I included a video recording track 2.
4. Drive.py was modified to crop the top and bottom image before sending it to prediction.
5. To run the program, one need to run the generate_img.py first. This will add some modified data to the IMG folder and output two csv file, one for training another for validation.

Files

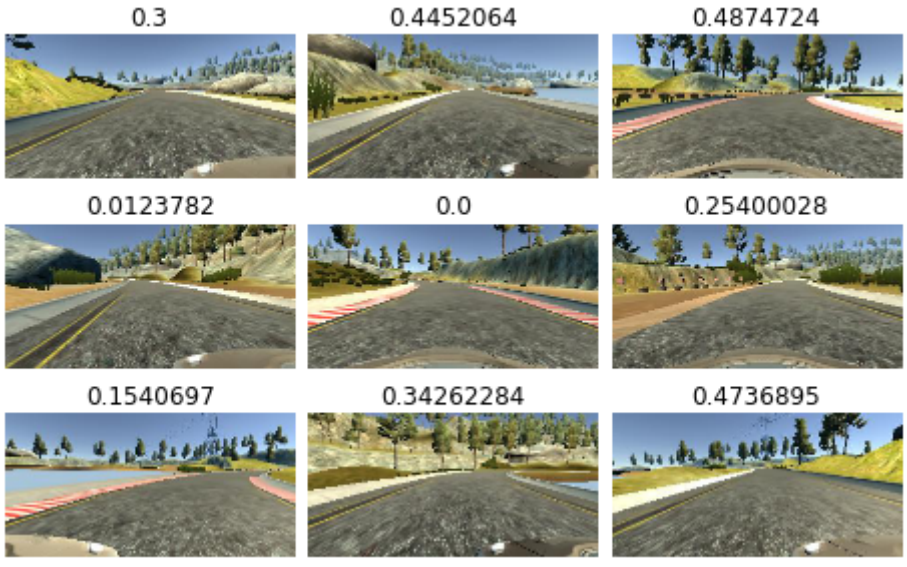
1. drive.py
2. model.py
3. generate_img.py
4. model.h5
5. run2.mp4

Methodology

1. At first I tried to record my own data and tested on comma.ai's model. I didn't do any image processing or image augmentation. The car almost always runs off the track. I tried to increase the epoch and started adding additional data to the generator to see whether it would improve my result. As the data increases, the generator loses efficiency. I then realize it is necessary to build a file to handle image augmentation seperately.
2. I then build generate_img.py for image augmentation only. I included the left and right images from the samples. I've added 0.3 to the left image and -0.3 to the right image.
3. Through trial and error I found out that I have too many zero steerings in my samples. I then filtered 75% of the images with zeros.
4. To make my model not biased towards one direction, I also augmented the samples with horiziontal flipping.
5. I've also found out that lowering the learning rate to 0.0001 helps. I'm using the Adam optimizer with 10 epochs.
6. After some testing, I found out that the sequence of the image augmentation is important as well. Previously I do tripling the sample first, then removing the zeros, then flipping the images at the end. Now I do removing the zeros at first, then tripling the data, then flipping the images at the end. The reason I do this is because it expedite the process of generating new data.
7. I've also added cropping and resizing later to enhance my pipline. Cropping is need because the model doesn't need to learn the sky nor the bonnet. Resizing to (64,64) helps the training process but loses some information.
8. The training samples histogram distribution is shown below:



9. Here's 9 samples I included as visualization:

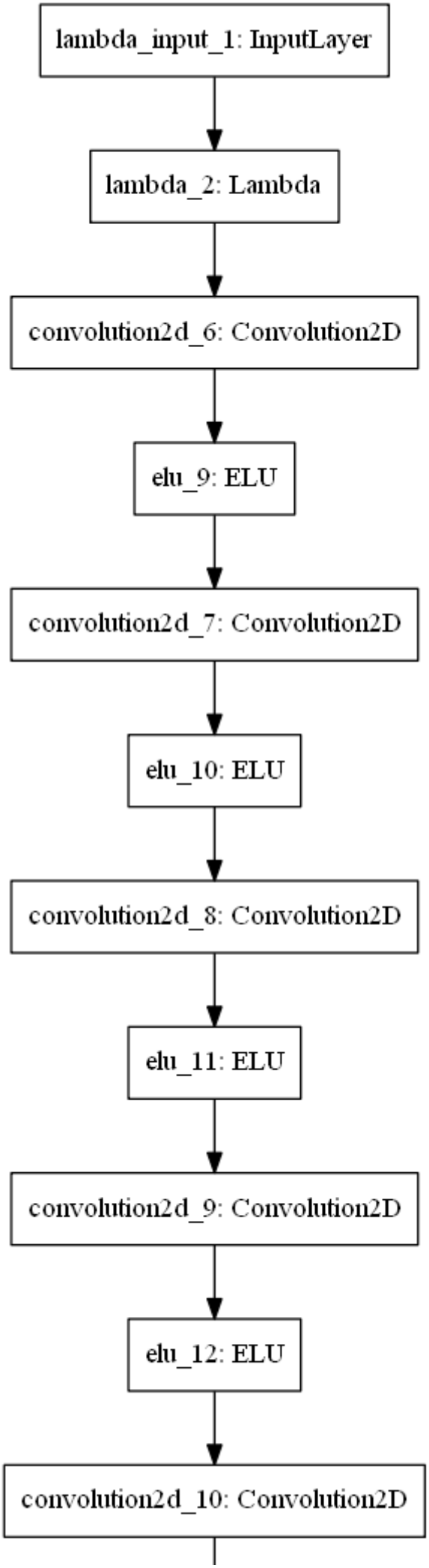


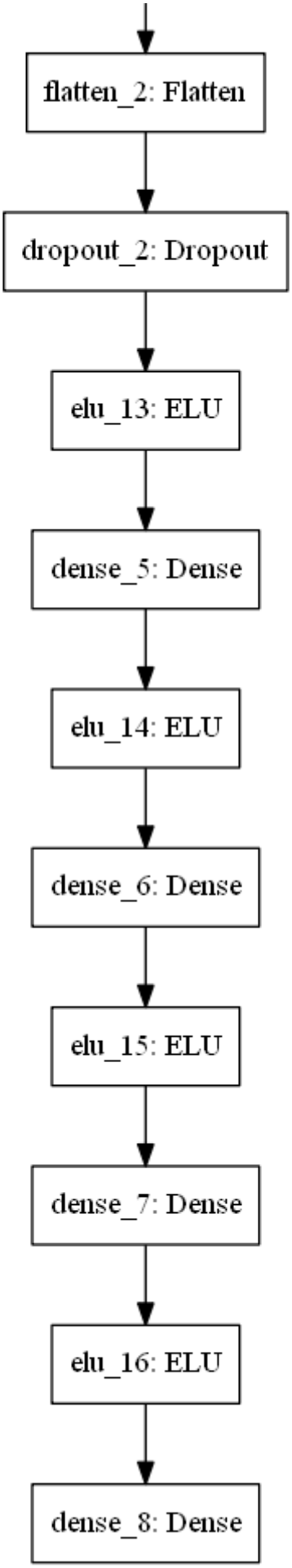
10. Next thing I did was changing to Nvidia's model, and it performs a lot better than comma.ai's model. I added a dropout layer between the first and second fully connected layers.

11. The last thing I did was to change my training samples to samples provided by Udacity. And it finally works !
12. Updated, using ELU instead of RELU made it work on the second track. Method not used: Shifting augmentation implemented but not used.

Model Architecture

Here's an output image of my model:


lambda_input_1: InputLayer
lambda_2: Lambda
convolution2d_6: Convolution2D
elu_9: ELU
convolution2d_7: Convolution2D
elu_10: ELU
convolution2d_8: Convolution2D
elu_11: ELU
convolution2d_9: Convolution2D
elu_12: ELU
convolution2d_10: Convolution2D



Description of the model:

1. 5 convolutional layers to increase the feature depth
2. ELU for faster training and accurate result

3. Three fully connected layer, outputting steerning angle
4. Dropout layer of 0.5 keep rate at the middle
5. Adam optimizer with learning rate 0.0001

Here's a detailed explaination of the model's layers

Layer (type)	Output Shape	Param #	Connected to
lambda_2 (Lambda)	(None, 64, 64, 3)	0	lambda_input_1[0][0]
convolution2d_6 (Convolution2D)	(None, 30, 30, 24)	1824	lambda_2[0][0]
elu_9 (ELU)	(None, 30, 30, 24)	0	convolution2d_6[0][0]
convolution2d_7 (Convolution2D)	(None, 13, 13, 36)	21636	elu_9[0][0]
elu_10 (ELU)	(None, 13, 13, 36)	0	convolution2d_7[0][0]
convolution2d_8 (Convolution2D)	(None, 5, 5, 48)	43248	elu_10[0][0]
elu_11 (ELU)	(None, 5, 5, 48)	0	convolution2d_8[0][0]
convolution2d_9 (Convolution2D)	(None, 3, 3, 64)	27712	elu_11[0][0]
elu_12 (ELU)	(None, 3, 3, 64)	0	convolution2d_9[0][0]
convolution2d_10 (Convolution2D)	(None, 1, 1, 64)	36928	elu_12[0][0]
flatten_2 (Flatten)	(None, 64)	0	convolution2d_10[0][0]
dropout_2 (Dropout)	(None, 64)	0	flatten_2[0][0]
elu_13 (ELU)	(None, 64)	0	dropout_2[0][0]
dense_5 (Dense)	(None, 100)	6500	elu_13[0][0]
elu_14 (ELU)	(None, 100)	0	dense_5[0][0]
dense_6 (Dense)	(None, 50)	5050	elu_14[0][0]
elu_15 (ELU)	(None, 50)	0	dense_6[0][0]
dense_7 (Dense)	(None, 10)	510	elu_15[0][0]
elu_16 (ELU)	(None, 10)	0	dense_7[0][0]
dense_8 (Dense)	(None, 1)	11	elu_16[0][0]

Total params: 143,419
Trainable params: 143,419

Drive.py

I follow my reviewer's advice and added a simple proportional feedback control. I set the reference speed to be 20 and Kp to be 0.3.

Acknowledgement

This is overall a very challenging project, and it was a lot of fun completeing it. I had a lot of help by reading these two article:

1. https://medium.com/@billzito/my-first-self-driving-car-e9cd5c04f0f2#.ahi9x4nfm (https://medium.com/@billzito/my-first-self-driving-car-e9cd5c04f0f2#.ahi9x4nfm) by Bill Zito
2. https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9#.vobdrpdw9 (https://chatbotslife.com/using-augmentation-to-mimic-human-driving-496b569760a9#.vobdrpdw9) by Vivek Yadav