Machine Learning Engineer Nanodegree

Capstone Proposal

Abhishek Kumar Dubey

July 14th, 2018

Proposal

Domain Background

I work in the automotive domain. The fascinating topic in the automotive domain is autonomous vehicle. The vehicle which can run without any driver. Many attempts are made to bring the dream to the reality.

The autonomous vehicle has many functions like LWD (lane departure warning), LKS (Lane keeping support), PD (Pedestrian detection) etc. This is my first attempt to contribute in the era of AD (Autonomous driving). I will design the basic functionality which the base of LDW and LKS – Lane marking detection.

I have designed the lane marking detection function using State of Art. The first Lane marking detection function I made was using Hough line Transform, Second attempt was made using regression technique. I made use of np.polyfit function to use the regression in my code. This time I want to implement the functionality using machine learning.

Problem Statement

A lot of accident happens because of manual driving. Humans can identify the lane lines with a lot of accuracy but we are very porn to commit the mistakes. Sometimes we fail to apply the brake or sometimes we cannot decide in which direction we should steer our vehicle. We take time to make decision and meet the accident. The computer can take the decision quickly and it can save us from the accident provided we make the computer able to identify the lane markers.

The State of Art code does not perform well. If the situation varies, like rainy season, night time, snow fall etc. These situations changes the intensity and brightness of the image. The state of Art logic works well in some conditions only and fails to work in special condition. If there is bad lane marking on the road the function performance reduces drastically.

Datasets and Inputs

I searched online resource for the data set. But I did not find any properly labeled data. There are many data sets available online it has labels of pedestrian, road signs, vehicle and all but no labels for lane marking. There are some data set available with lane marking also but not with all the conditions

I took online dataset and decided to label the data myself. I used state of art method to label the data. Some of them was not labeled correctly. I used Matlab ground truth labeler to label them. At last some of the data cannot be labeled by any algorithm. I labeled those data manually.

I used my regression logic to label the data. I had followed Udacity Self driving nanodegree program in which I had implemented advanced lane detection function. I did the same to label the data automatically. After Canny edge detection I applied bird eye view and then computed histogram. In the next step I applied the moving average and window technique to filter the pixels which are good candidates for lanes. I applied the regression on those pixels to compute a 2-D polynomial equation. It was done for left and right lane. The area was filled with white colors and I got the level for the dataset.

Solution Statement

The solution for the Lane detection with a good performance is to use CNN. The CNN network can learn for all the environment conditions. As described above the state of art method was not able to perform well in all the condition. It the CNN which can perform will if fed with all the data of all the possible conditions. A well trained and a well-designed CNN will work well in all the conditions. Once the lane is detected the computer can take the further decision and the Autonomous driving can be converted into reality.

Benchmark Model

As discussed earlier the State of Art method is used for lane detection which does not perform well. The previous lane detection algorithm which is purely based on the image processing technique can be used for the comparison. The state of Art based lane detection method is best for benchmarking as we can clearly see the difference in the performance. The benchmarking evaluation shall be done in those conditions in which the previous model does not work well.

Evaluation Metrics

The output of the State of Art model is the polynomial coefficient which is used to draw the line. The color is filled in the space between the lane lines. These colors are nothing but a particular numbers.

The output of the CNN model is same. It is also the colors filled between the detected lines and these are also a particular number.

A new label for the same image will be designed which will be hand engineered by me which can be considered as the perfect output or an ideal output.

The loss function (mean squared error) will be calculated for both the model (state of art and CNN) with respect the hand engineered ideal output.

The loss can be compared and can be used as evaluation matrices.

Project Design

As the labels and dataset is already created. I will see if I can improve my levels. I will try to add the data for all possible conditions.

After the data set is prepared it is the time to design a good CNN network. To design a good CNN I will take reference of the previous project (CFAR -10 image classification).

As the output of my network is not a specific number it does not come under classification problem. The final layer output should be the same dimension as the validation image dimension. The image is resized to (80 *160) for the computational efficiency.

I will use first few convolution layers with increasing filter size. The filter size I will start from 8 and may go up to 64. These filters helps recognizing the different features in the image. I will use Keras visualization tool to see how the network is learning. The strides will be used as (1*1). So if my image dimension is (80*160*3) with a valid stride of (1*1) and kernel size (3*3) and with 8 filters, the dimension of the output of this layer will be (78*158*8).So we see here the dimension of the image is decreasing. So to match the size of the output dimension to the validation set dimension the deconvolution and up sampling shall be used. The validation set dimension is (80*160*1) so the number of filter in the output layer should be 1.

The first layer will be batch normalization. And next convolution with activation function followed by some drop out layer. There will be max pulling after some convolution laver, after that I will start increasing the dimension so I will use up sampling and deconvolution layers.

The deconvolution is called as Conv2DTranspose in Keras which is quite confusing. It is actually a convolution but in opposite direction so called as deconvolution it is used to up sample the image.

I have already designed the network architecture prototype. Below is the prototype but it might be changed when I will tune the hyper parameter.

batch_normalization	(None, 80, 160, 3)
Conv	(None, 78, 158, 8)
Conv	(None, 76, 156, 16)
Conv	(None, 74, 154, 32)
Dropout	(None, 74, 154, 32)
Conv	(None, 72, 152, 32)
Dropout	(None, 72, 152, 32)
MaxPooling	(None, 36, 76, 32)
Conv	(None, 34, 74, 64)
Dropout	(None, 34, 74, 64)
Deconv	(None, 36, 76, 64)
Dropout	(None, 36, 76, 64)
UpSampling	(None, 72, 152, 64)
Deconv	(None, 74, 154, 32)
Dropout	(None, 74, 154, 32)
Deconv	(None, 76, 156, 32)
Dropout	(None, 76, 156, 32)
Deconv	(None, 78, 158, 16)

Dropout	(None, 78, 158, 16)
Deconv	(None, 80, 160, 1)

As we can see the output dimension is (80*160*1) which is required to validate it against the validation set.

The output of the CNN is also an image in which the bits are one in the area between the lane lines. The output of the CNN will be drawn back on the original image using the Image processing technique.