CarND Term1 - Project2: Traffic Signal Classifier Project

Training and Testing on Given data (Train, Valid, Test):

Preprocessing**:**

a) First the data was normalized (divided by 255 as the data is images in this case) to avoid dealing with very high numbers which could be a result of matrix multiplications, and to make it easier for the NN to converge to an optimal solution.

b) Then the images were converted to grayscale using weighted average values of R=0.21, G=0.72 and B=0.07. Source link: https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

As it turned out, the weighted average data resulted in at least a 2% increase in validation accuracy. This grayscale conversion was done as the image shape doesn't change becasue of colors and it would decrease the amount of variables.

c) I tried combining the grayscale data with the original data to make it into a 32x32x4 size input so that the NN can choose which data to take for implementation, but it didn't have any significant impact on validation accuracy. And as expected it almost doubled the processing time, so kept the input at 32x32x1.

d) Data Augmentation: Produced gamma adjusted images and combined them with training data. Improved the accuracy by around 2%. Also based on the number of samples in each class, produced rotated (by 30 degrees) versions of the images that have less than 550 samples. Although, I could have used any random angle for each image, for example between -30 and 30. Combined these two new sets with the original training data.

Model Architecture:

a) Used the standard LeNet architecture with ReLu activation functions.

Added three dropout layers between the fully connected layers with a keep probability of 0.7. Tried different probability numbers but 0.7 seemed to work well of them all. Adding dropout layers also increased the accuracy by 1 to 2%.

b) Adding dropout layers between convolutional layers (with 0.5 prob) reduced the accuracy probably becuase of underfitting. So, removed them.

c) Tried implementing a partial inception method by inputting both conv1 and conv2 into the first fully connected layer, but it didn't make that big of difference in the accuracy and obviously increased processing time. Something I need to look into more detail.

Model Training:

a) AdamOptimizer was used for the training with a learning rate of 0.002, batch size of 128 and 30 epochs. Tried different learning rates between 0.0005 and 0.01, 0.002 seemed to do well eventhough it was still making accuracy slightly go up and down during the epochs.

Solution:

Validation accuracy: 96.5%

Test accuracy: 93.2%

Testing on New images:

a) Downloaded 5 images(included in the submitted zip folder) of German traffic signals online, preprocessed them and ran them with the final weights of the trained model. Accuracy on these new images was 80%. It was noticed that inputting the new images without any irrelevant objects around the traffic sign made it easier for the network to identify them. This step can be included in the pre-processing stage of the NN if the data is being collected directly from the road.

Test accuracy on new images is 80% compared to the original test accuracy of 93.2%

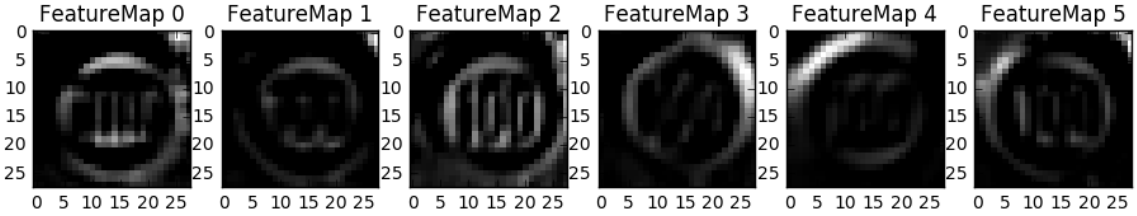
b) The 'Traffic signals' image shown below seemed to be the toughest

one of the 5 to analyze becasue of a bright light above it and its tilted orientation.

c) The top five softmax probabilities were outputted for these 5 images. Except for the last image (above traffic signal), the top softmax probability of the other 4 images turned out to be above 98%. For the above image, correct prediction didn't feature in the top 5. Inputting the image with signal centered in the image and removing the other objects would help, but left this as is to understand the limitations of this particular NN.

Visualizing layers of the Neural Network:

a) Outputted images of the trained network's feature maps after the first convolutional layer. It can be seen that NN has learned to concentrate on the outer and inner boundaries of the circle and the number in the middle.



Speed: 100