#**Traffic Sign Recognition**

**Build a Traffic Sign Recognition Project**

The goals / steps of this project are the following:
* Load the data set (see below for links to the project data set)
* Explore, summarize and visualize the data set
* Design, train and test a model architecture
* Use the model to make predictions on new images
* Analyze the softmax probabilities of the new images
* Summarize the results with a written report

###Data Set Summary & Exploration

####1. Provide a basic summary of the data set and identify where in your code the summary was done. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

1. I imported the signnames.csv to store all the indices and sign names.
2. Using basic python commands, found the size of the training and test set
3. Using pandas series I converted all labels into a series to find the number of unique values

Summary Statistics obtained as below:
Number of training examples = 4
Number of testing examples = 4
Image data shape = (32, 32, 3)
Number of classes = 43

####2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

The code for this step is contained in the third code cell of the IPython notebook.

I used the matplotlib library to plot an image of the traffic sign and print the corresponding label

###Design and Test a Model Architecture

####1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.

The code for this step is contained in the fourth code cell of the IPython notebook.

I tried various techniques, converted the images to grayscale and stored it as a different dataset as well as tried concatenating the grayscale values with RGB values.

These efforts gave me a good understanding of the technique but at the end, I used 1 by 1 convolution on the first layer for CNN to decide the color scale it would like.

####2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, identify where in your code, and provide example images of the additional data)

My final training set had 34799 number of images. My validation set and test set had 4410 and 12630 number of images.

####3. Describe, and identify where in your code, what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

The code for my final model is located in the seventh cell of the ipython notebook.

My final model consisted of the following layers:

| Input | Layer | Stride | Kernel | Padding | Output |
|---|---|---|---|---|---|
| 32x32x3 | Image | | | | 32x32x3 |
| 32x32x3 RGB | Convolution (1by1) | 1x1 | 1x1 | VALID | 32x32x1 |
| 32x32x1 | Convolution (First) | 1x1 | 3x3 | VALID | 30x30x14 |
| | Activation (Relu) | | | | |
| 30x30x14 | Max Pooling | 2x2 | 2x2 | VALID | 15X15X14 |
| 15X15X14 | Convolution (Second) | 1x1 | 3x3 | VALID | 13x13x18 |
| | Activation (Relu) | | | | |
| 13x13x18 | Max Pooling | 1x1 | 2x2 | VALID | 12x12x18 |
| 12x12x18 | Convolution (Third) | 1x1 | 3x3 | VALID | 10x10x25 |
| | Activation (Relu) | | | | |
| 10x10x25 | Max Pooling | 2x2 | 2x2 | VALID | 5X5X25 |
| 5x5x25 | Flatten | | | | 625 |

| | | | | | |
|---|---|---|---|---|---|
| 625 | Fully Connected (First) | | | | 440 |
| | Activation (Relu) | | | | |
| 440 | Fully Connected (Second) | | | | 120 |
| | Activation (Relu) | | | | |
| 120 | Fully Connected (Third) | | | | 84 |
| | Activation (Relu) | | | | |
| 84 | Fully Connected (Fourth) | | | | 43 |
| | Softmax | | | | |

####4. Describe how, and identify where in your code, you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

The code for training the model is located in the seventh cell of the ipython notebook.

To train the model, I used
1. Learning rate of 0.001,
2. Optimizer as AdamOptimizer
3. EPOCHS =15
4. Batch_size=128

####5. Describe the approach taken for finding a solution. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

The code for calculating the accuracy of the model is located in the ninth cell of the Ipython notebook.

My final model results were:
* validation set accuracy of 94%

* test set accuracy of 92.7%

I started of training my model using LeNet architecture and achieved a 88% accuracy. This gave me confidence that LeNet could be modified to give better accuracy.
The initial problems for low accuracy was the depth of the CNN and the fact that I was using raw images without preprocessing (I tried grayscale, but did not have much of an effect on the accuracy)
Using the basic LeNet, I added multiple convolution networks as well as a fully connected layer.
After doing some research, I found out that adding a 1by1 convolution at the start could help the CNN to pick out the color scale it would like from RGB.
I used Max Pooling which was very effective in deepening the architecture. In different layers of convolution and max pooling I extracted a good bit of features so as to increase the accuracy of the model.
I tried tuning the learning rate, but it had a negative effect. Maybe I needed to increase the EPOCHS while tuning the learning rate.
The most important design choice was the 1by1 convolution layer.
Implementation of dropout in this architecture would help in preventing overfitting. I might implement it some other time.

###Test a Model on New Images

####1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

I used 8 german signs to test my architecture. I have displayed them below:


Image 1- The angle of the picture could have been an issue
Image 2- copyright picture at the center could be an issue
Image 3- As this picture is compressed into 32x32, the figures would be difficult to picturize
Image 4- copyright picture at the center could be an issue

Image 5- This is a clean image
Image 6- The scratches and the inclination could be an issue
Image 7- copyright picture across the image could be an issue
Image 8- The brightness of the image could be an issue

####2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

The code for making predictions on my final model is located in the eight cell of the Ipython notebook.

Here are the results of the prediction:

| Image No | Prediction | Outcome |
|---|---|---|
| 1 | No entry | Correct |
| 2 | Speed limit (30km/h) | Correct |
| 3 | Children crossing | Correct |
| 4 | General caution | Correct |
| 5 | Stop | Correct |
| 6 | No passing | Correct |
| 7 | Bumpy road | Correct |
| 8 | Road work | Correct |

The model was able to correctly guess 8 of the 8 traffic signs, which gives an accuracy of 100%. Test Set accuracy was 92.7%.
The model varies each time I run it, which is expected, I get an accuracy of about 75% to 100% on web based images which relates to Test set accuracy

####3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction and identify where in your code softmax probabilities were outputted. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The code for making predictions on my final model is located in the 11th cell of the Ipython notebook.

| Image | Softmax |
|---|---|
| No entry | 1 |
| Speed limit (30km/h) | 1 |
| Children crossing | 1 |
| General caution | 1 |
| Stop | 1 |
| No passing | 1 |
| Bumpy road | 1 |

| | |
|---|---|
| Road work | 0.94 |

The model predicts well on the web based images