Project: Build a Traffic Sign Recognition Program

In this project, I used deep neural networks and convolutional neural network architecture (LeNet) to classify traffic signs. Model is trained and validated for classifying "German traffic sign dataset" and later the trained model is applied on traffic sign images taken from internet.

Steps followed:

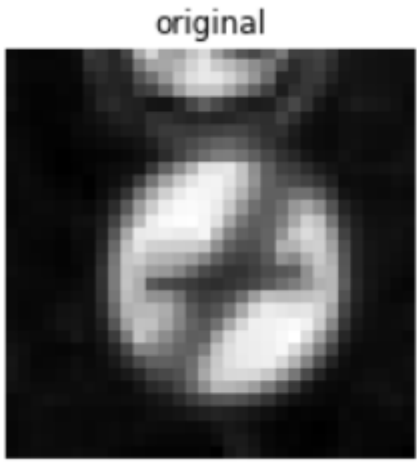
i. Data loading for training, validation and testing

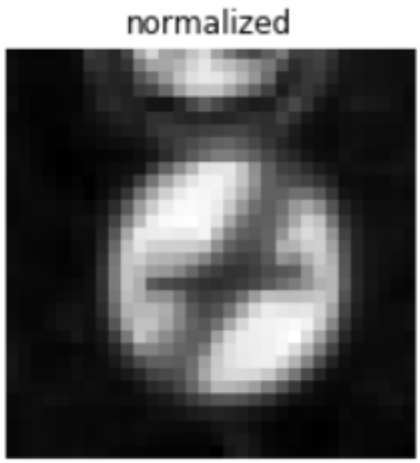
```
Image Shape: (32, 32, 3)
Training Set:   34799 samples
Validation Set: 4410 samples
Test Set:       12630 samples
```

ii. Initially I tried to train the model without any pre-processing of images, and couldn't be able to achieve a satisfactory accuracy. Therefore, I converted the image (RGB) to grayscale and further normalized. By converting RGB to Gray, I am reducing the color channels from 3 to 1. Further my gray scale image features are distributed in a wide range (in range of 0 to 255), which leads to inaccuracy while applying gradient descent optimization with learning rate, hence need to scale it, to minimize the errors. If we did not scale our input training vectors, the ranges of our distributions of feature values would likely be different for each feature, and thus the learning rate would cause corrections in each dimension that would differ (proportionally speaking) from one another. We might be overcompensating a correction in one weight dimension while undercompensating in another.

Also, its been preferred to centering the data near to origin/ zero-center, which has been achieved by mean.

```
RGB shape for train: (34799, 32, 32, 3)
RGB shape for valid: (4410, 32, 32, 3)
RGB shape for test: (12630, 32, 32, 3)
Grayscale shape for train: (34799, 32, 32, 1)
Grayscale shape for valid: (4410, 32, 32, 1)
Grayscale shape for test: (12630, 32, 32, 1)
```





Mean value before preprocessing:

```
82.677589037
83.5564273756
82.1484603612
```

Mean value after preprocessing:

```
-0.354081335648
-0.347215411128
-0.358215153428
```

Further to enhance the image which will certainly improve the efficiency of the architecture, image warping and histogram equalization (on gray scale image) can be done (although I tried, but didn't applied later).

iii. LeNet architecture is applied

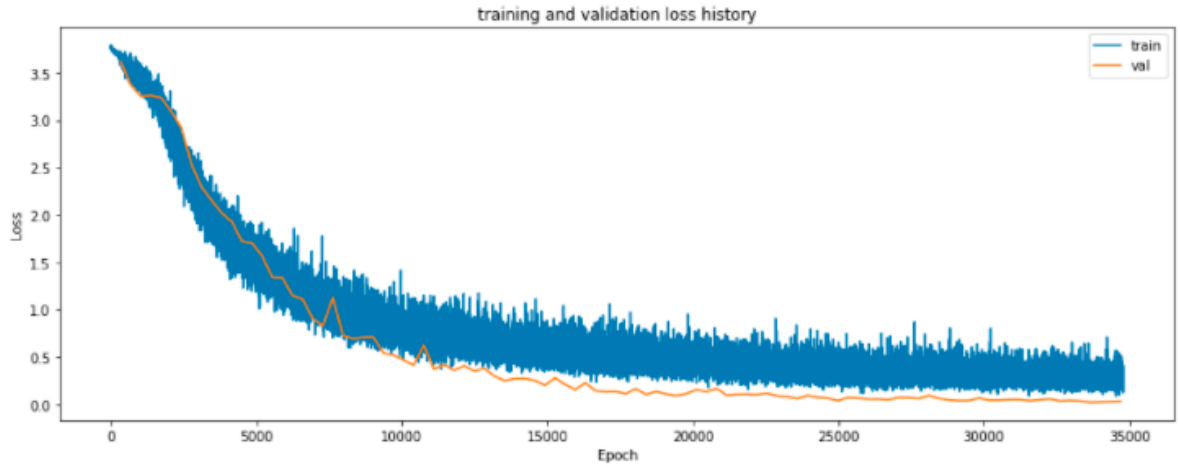
Layer	Description		Input	Output
1	Convolution	kernel: 5x5; stride:1x1; padding: valid	32x32x3	28x28x6
	Max pooling	kernel: 2x2; stride:2x2;	28x28x6	14x14x6
2	Convolution	kernel: 5x5; stride:1x1; padding: valid	14x14x6	10x10x16
	Max pooling	kernel: 2x2; stride:2x2;	10x10x16	5x5x16
	Flatten	Input 5x5x16 -> Output 400	5x5x16	400
3	Fully connected	connect every neural with next layer	400	120
4	Fully connected	connect every neural with next layer	120	80
5	Fully connected	output 43 probabilities for each label	80	43

Hyper-parameter's passed:

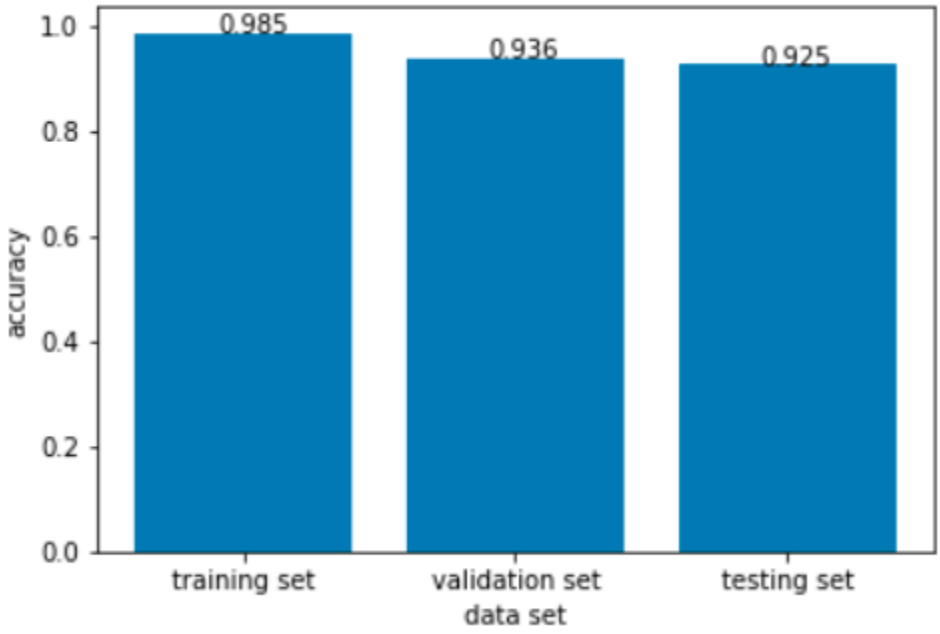
LEARNING_RATE = 0.01
EPOCHS = 100
BATCH_SIZE = 100
Dropout keep probability 'keep_prob' = 0.5
mu = 0
sigma = 0.1

I started with pre-defined LeNet architecture with gradient descent optimizer and almost all of the tweaking from there was a process of trial and error. Although my guesses were educated, as in previous lectures it's been mentioned which parameter to work on first and how to proceed if it didn't worked. Although trial and error is pretty much a losing strategy when it comes to building the neural net, so I used to keep a note of the performance w.r.t to hyper-parameters passed. Initially I changed the learning rate and started its value from 0.00001 with higher batch size and lower epochs, but that didn't worked and then keeping the same learning rate I tried for few other lower values batch size and higher epochs. I repeat this process for quite some time and then I changed the learning rate to some other value and finally ended up with 0.01.
In general there are three commonly used techniques are used for learning rate: step decay, exponential decay and 1/t decay

```
EPOCH 100 : Validation Accuracy = 0.936
Model saved
```




```
CPU times: user 4min 9s, sys: 1min 11s, total: 5min 20s
Wall time: 4min 1s
```



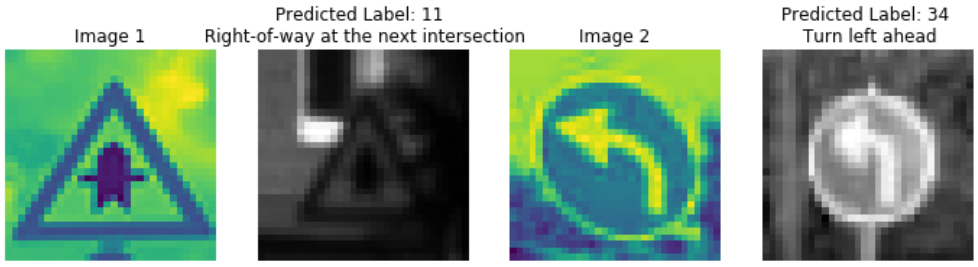
A validation set can be used to assess how well the model is performing. A low accuracy on the training and validation sets imply underfitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting.

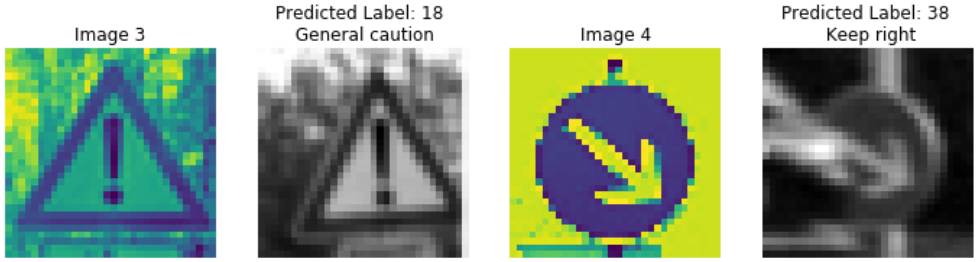
iv. Trained model is saved in the workspace inside the “model” folder.

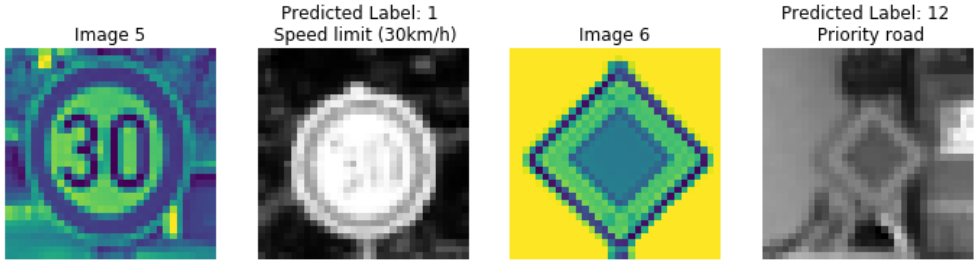
v. Further the trained model is tested on traffic sign images available on internet, which are stored under the “found-images” folder.

```
Input Image 0 (32, 32, 3)
Input Image 1 (32, 32, 3)
Input Image 2 (32, 32, 3)
Input Image 3 (32, 32, 3)
Input Image 4 (32, 32, 3)
Input Image 5 (32, 32, 3)
After Processing Image: (6, 32, 32, 1)
```

And the trained model successfully able to correctly guess the 6 of 6 traffic signs.







vi. the Top 5 Softmax Probabilities of predicted labels by our model for each input image is as follows:

```
File Name: [ './found-images/1.png' ]
Right-of-way at the next intersection: 0.06075329
Beware of ice/snow: 0.02237554
Speed limit (20km/h): 0.02236271
Speed limit (30km/h): 0.02236271
Speed limit (50km/h): 0.02236271


File Name: [ './found-images/6.png' ]
Turn left ahead: 0.06078671
Keep right: 0.02236226
Speed limit (60km/h): 0.02236222
Go straight or right: 0.02236222
Speed limit (20km/h): 0.02236222


File Name: [ './found-images/8.png' ]
General caution: 0.06078572
Traffic signals: 0.02236265
Speed limit (20km/h): 0.02236224
Speed limit (30km/h): 0.02236224
Speed limit (50km/h): 0.02236224


File Name: [ './found-images/5.png' ]
Keep right: 0.06078682
Speed limit (20km/h): 0.02236222
Speed limit (30km/h): 0.02236222
Speed limit (50km/h): 0.02236222
Speed limit (60km/h): 0.02236222


File Name: [ './found-images/2.png' ]
Speed limit (30km/h): 0.06078500
Speed limit (50km/h): 0.02236284
Speed limit (20km/h): 0.02236233
Speed limit (70km/h): 0.02236227
Speed limit (60km/h): 0.02236225


File Name: [ './found-images/3.png' ]
Priority road: 0.06078682
Speed limit (20km/h): 0.02236222
Speed limit (30km/h): 0.02236222
Speed limit (50km/h): 0.02236222
Speed limit (60km/h): 0.02236222
```

vii. Improvement/ Future Scope:

I have processed the data but not enhanced its readability, which can further be done using image wrapping and angular tilting. That might help in lesser time in training. Should have tried other CNN architectures like AlexNet and GoogleNet.