Traffic Sign Classifier

Introduction

Traffic Sign Classifier is trained on German Traffic signs dataset and classifies/recognizes new traffic images as one of the 43 traffic signs. Below are the steps to build Traffic Sign Classifier pipeline:

- Exploratory Data Analysis
- Data Augmentation
- Data Pre-processing
- Build Convolutitional Neural Network Architecture
- Train, validate, test and fine tune model and Network architecture
- Predict sign for new images and evaluate performance
- Visualize convolutional neural network to understand what each layer learns

Data Exploration and Summary

Data set consists of 34K training samples, 4.4K validation and 12.6k. There are 43 distinct traffic signs in the dataset.

Dataset Summary

```
Number of training examples      = 34799 (67%)
Number of validation examples    = 4410  (8.5%)
Number of testing examples       = 12630 (24.36%)
Image data shape = (32, 32, 3)
Number of classes = 43
```

Let's examine few of the traffic sign classes:

Observations

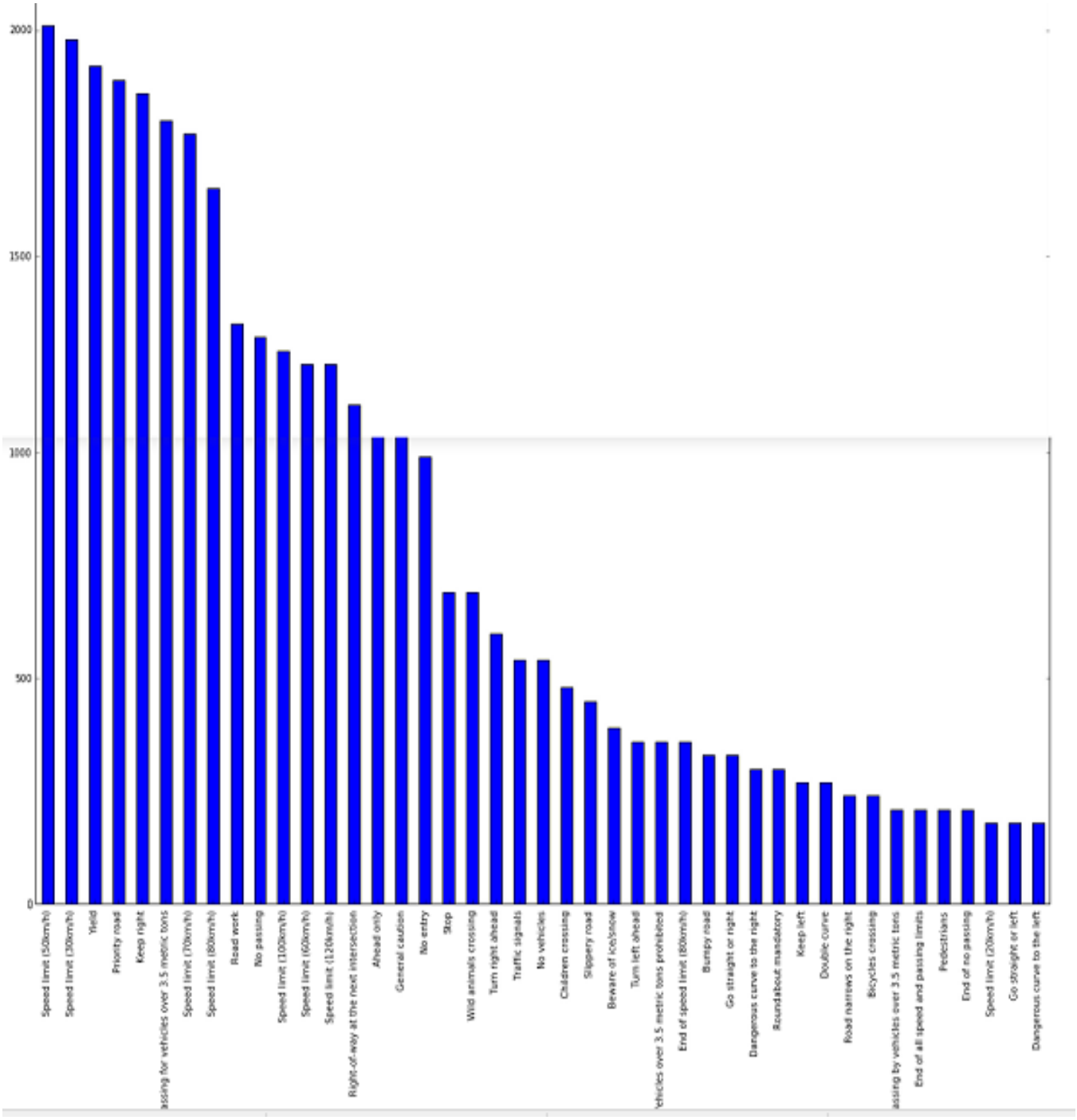
We can observe that traffic signs of the same appear differently due to following aspects

- Lighting conditions
- Brightness
- Angle or position
- Sharpness
- Size
- Color

It will be interesting to observe the distribution of the traffic sign classes.

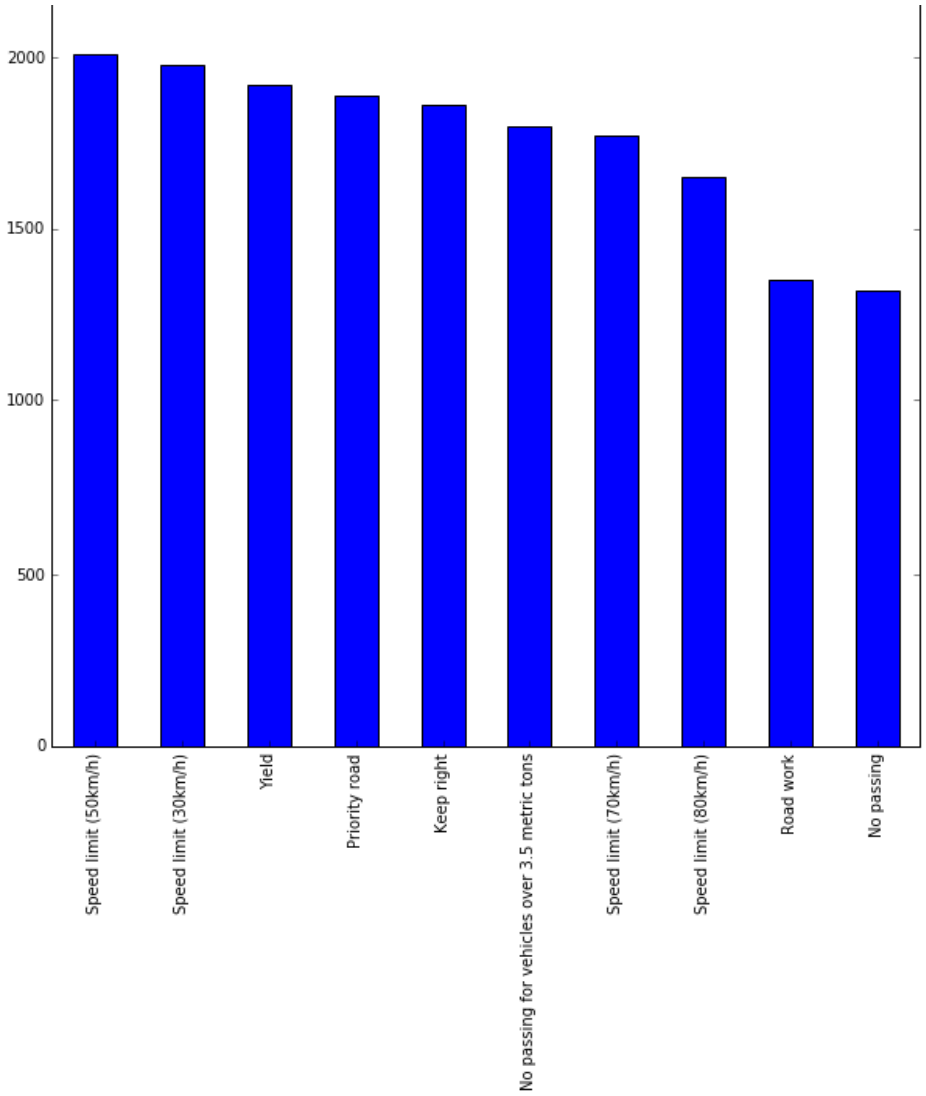
Traffic Sign Distribution

Below is the distribution of the traffic signs. Obviously, the classes are skewed with many traffic sings with less than 500 traffic signs.



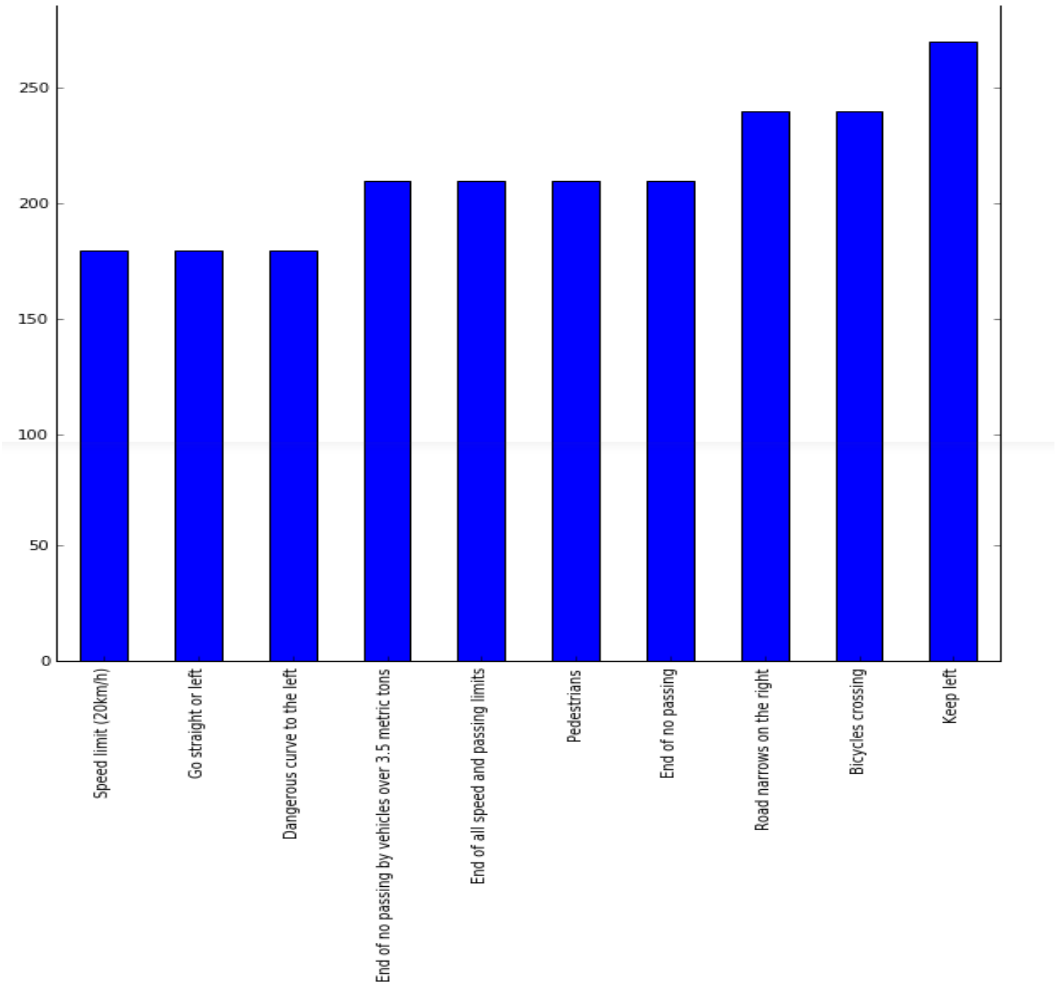
Top 10

Below are the top 10 traffic signs by number of examples:



Speed Limit 50 kmph has the highest number of example images (2000).

Bottom 10



Speed Limit 20kmph has the lowest number of example images (~ 180).

Summary Statistics

count	43.000000
mean	809.279070
std	626.750855
min	180.000000
25%	285.000000
50%	540.000000
75%	1275.000000
max	2010.000000

Given the that lower quartile has only 285 images, we can augment the number of examples for these classes by faking the data. I applied rule of augmenting the traffic sign class by adding 400 images per class for the signs with less than 1000 example images. Initially I had planned to augment all the classes but the net result could be that the distribution would remain the same. Combinations of random rotation, translation, sharpening or histogram equalization.

Training vs Testing vs Validation Distribution

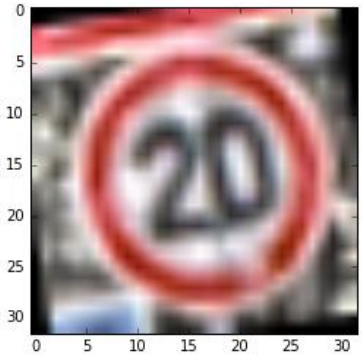
Data Augmentation

Given the imbalanced data set, we can augment the dataset by generating fake images. Images can be generated through following operations:

- Rotation [-15, 15]
- Translation [-2, 2]
- Histogram Equalization – it is technique for adjusting image intensities by enhancing the contrast
- Sharpening

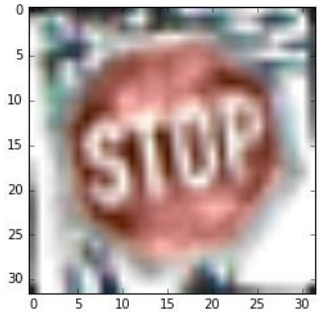
I have generated fake images by transforming the images through series of few of the above set of operations. Below are few examples of such augmented images:

Original Image



Augmented Image

Original Image



Augmented Image

Data Pre-processing

Below were the data pre-processing tests done:

a) **Convert to gray scale**

Color Image

Gray Scale Image

b) **Normalize data set to be in the range of [0, 1]**

As per the Le-Net 5 implementation, color channels would not help in increasing the accuracy, which was my experience as wel. Thus, gray scale conversion was applied. Further I have experimented with pixel ranges such as -5 to +5, -1 to +1 but found that simple division by 255 yielding a [0,1] range results in better accuracy. Since in each layer we do dot products with the weights on the input vectors and with deeper networks, it's possible that the gradients might explode. Thus, normalizing the data to a fixed range may avoid such explosion of gradients.

Model Architecture

I have started with Le-Net 5 which was proven to be useful architecture for Traffic signs as a starting point. However, I did not strictly follow the model and did my own experiments with the architecture. The baseline architecture provided only about 85% accuracy. Initial architecture did not seem to capture the complexities of the images very well.

On this architecture, I have experimented with filter sizes, number of filters and droputs. As per the latest finding dropouts perform better than max pools. Dropouts perform better regularization than maxpool layers as per the latest findings. Thus, I have added as two Droput layers – one in Conv 2 layer and the other in Conv – 3 layers while still retaining max pool layers in layer 1 and 3. Biggest jump in accuracy came from adding additional convolutional layer.

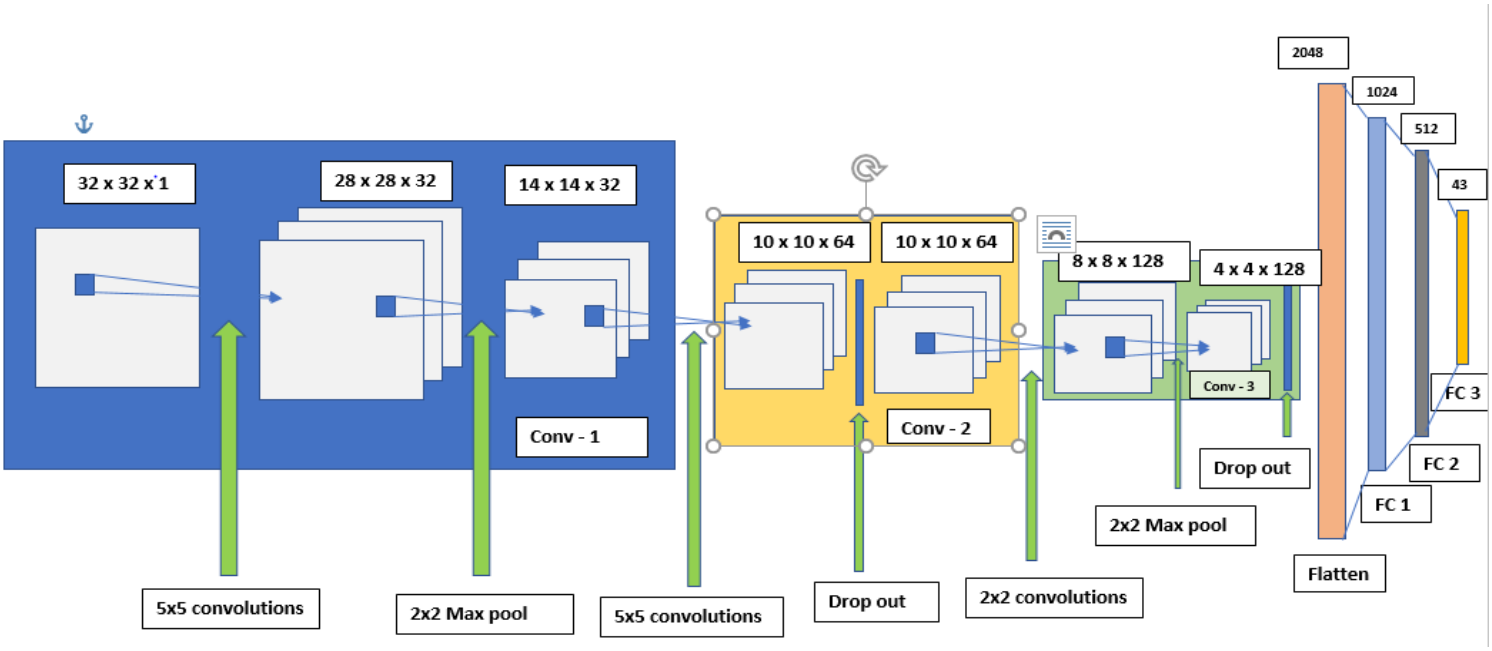


Figure: 6-layer Convolutional Neural Network

6 – Layer Convolutional Network

Layer	Description	Input	Output
Input Layer	32 x 32 x 1 after conversion to grayscale	32x32x3	32x32x1
Conv - 1	32 5x5 filters	32x32x1	28x28x32
Max pool - 1	2x2 Maxpool	28x28x32	14x14x32
Conv - 2	64 5x5 filters	14x14x32	10x10x64
Droput	Dropout layer (Keep 80%)	10x10x64	10x10x64
Conv - 3	128 2x2 Filters	10x10x64	8x8x128
Max pool - 2	2x2 Maxpool	8x8x128	4x4x128
Droput	Dropout layer (Keep 80%)	4x4x128	4x4x128
Flatten Layer	Flatten droput layer	4x4x128	2048
FC 1 Layer	Fully connected layer 1	2048	1024
FC 2 Layer	Fully connected layer 2	1024	512
FC 3 Layer	Fully connected layer 3	512	43

Fine Tuning

I have experimented with various values for Learning rate, number of epochs, droput and number of layers as shown below:

Parameters	Values
Learning Rate	[0.0001, 0.0005, 0.0003, 0.001]
Number of epochs	[20, 30, 40, 50]
Drop out percentage	[0.6, 0.7, 0.75, 0.8]
Number of layers	[5, 6]

Eventually, Learning rate of 0.0003, number of epochs = 50, droput percentage of 0.8 and 6-layer network provided the best performance.

Training, Validation and Testing

Results

Using the above parameters above, my validation accuracy is 96.8% and testing accuracy is 95.6%.

a) Without Data Augmentation

Stage	Accuracy
Training	1.000
Validation	0.968
Testing	0.956

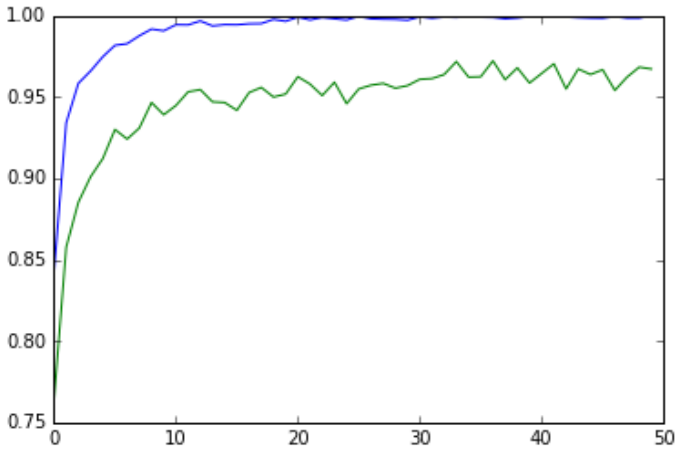
b) With Data Augmentation

Stage	Accuracy
Training	0.990
Validation	0.963
Testing	0.950

Data augmentation does not seem to particularly improve the performance. Perhaps a different combination of image transformations or lesser number of transformed images per traffic sign might help in improving the accuracy.

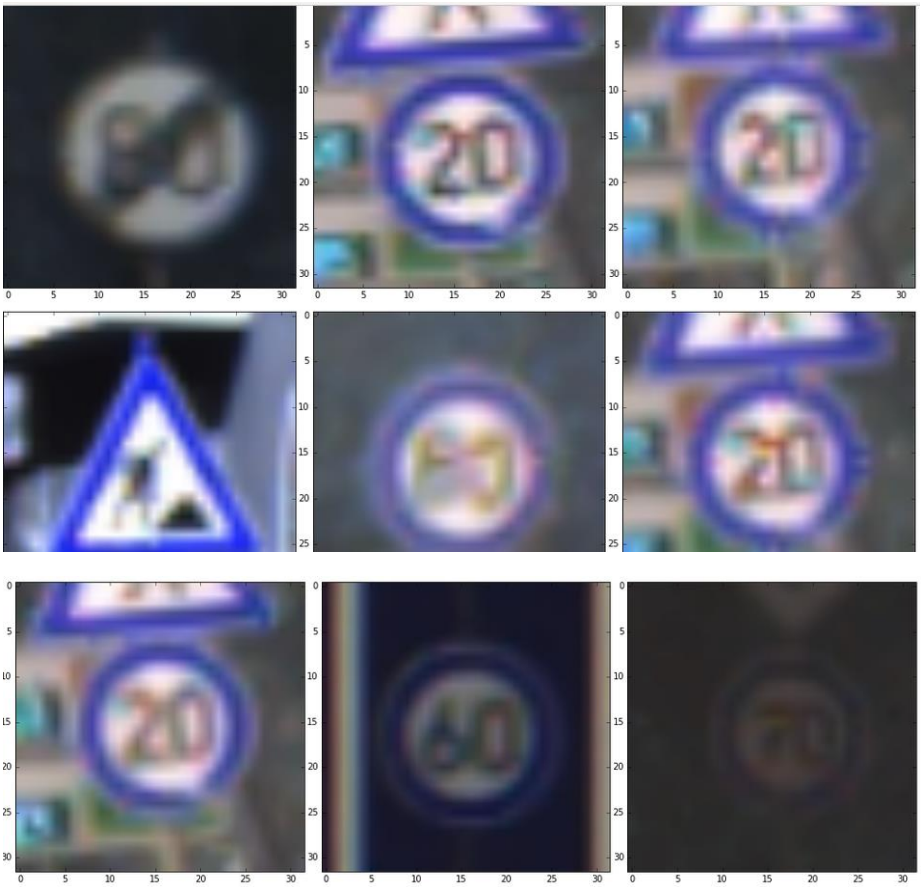
Training vs Validation Accuracy

Below is the plot for training vs validation accuracy:



Predicting Sign of New Images

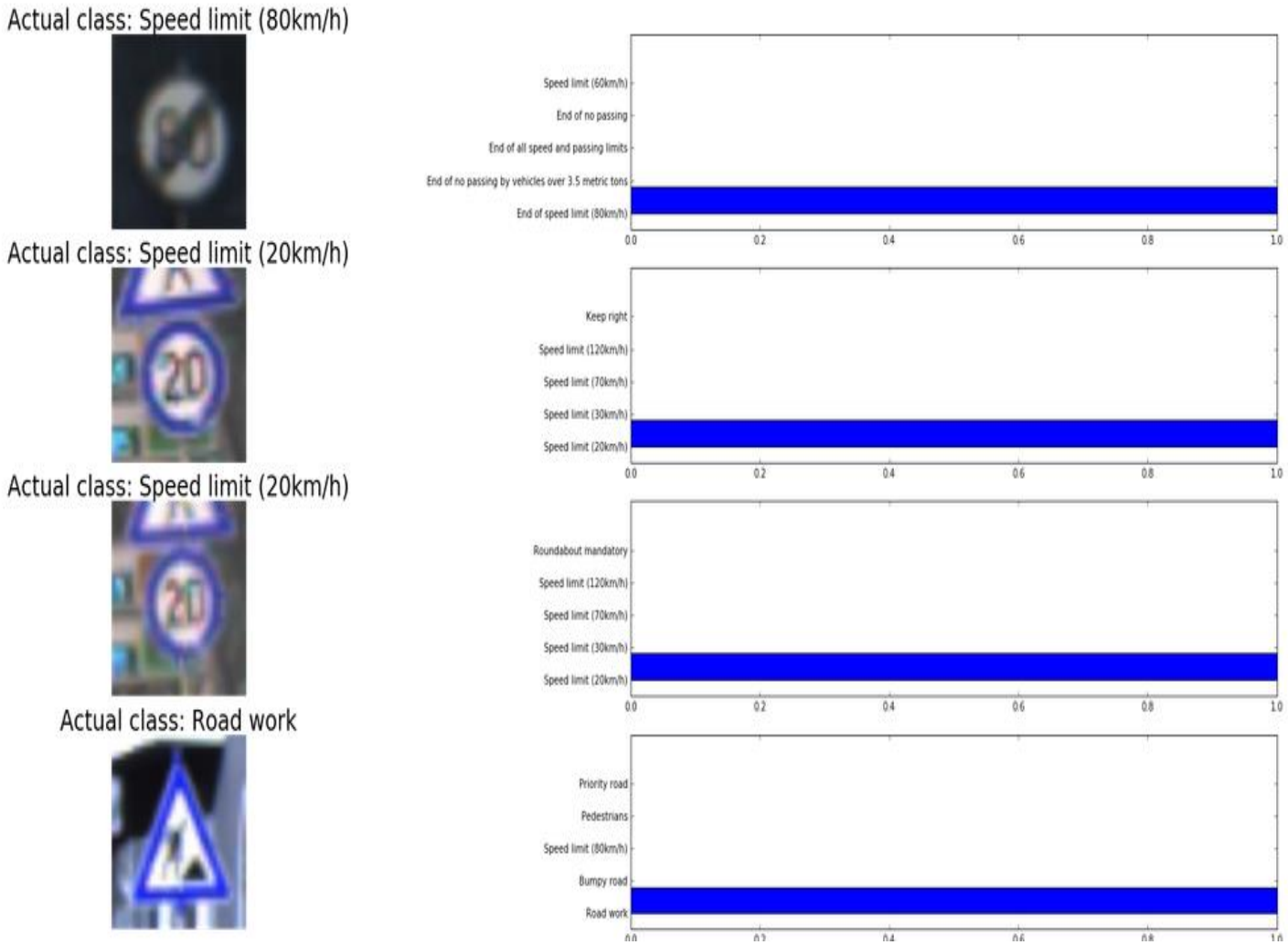
I have selected below set of random images from the web and tested in on my network:

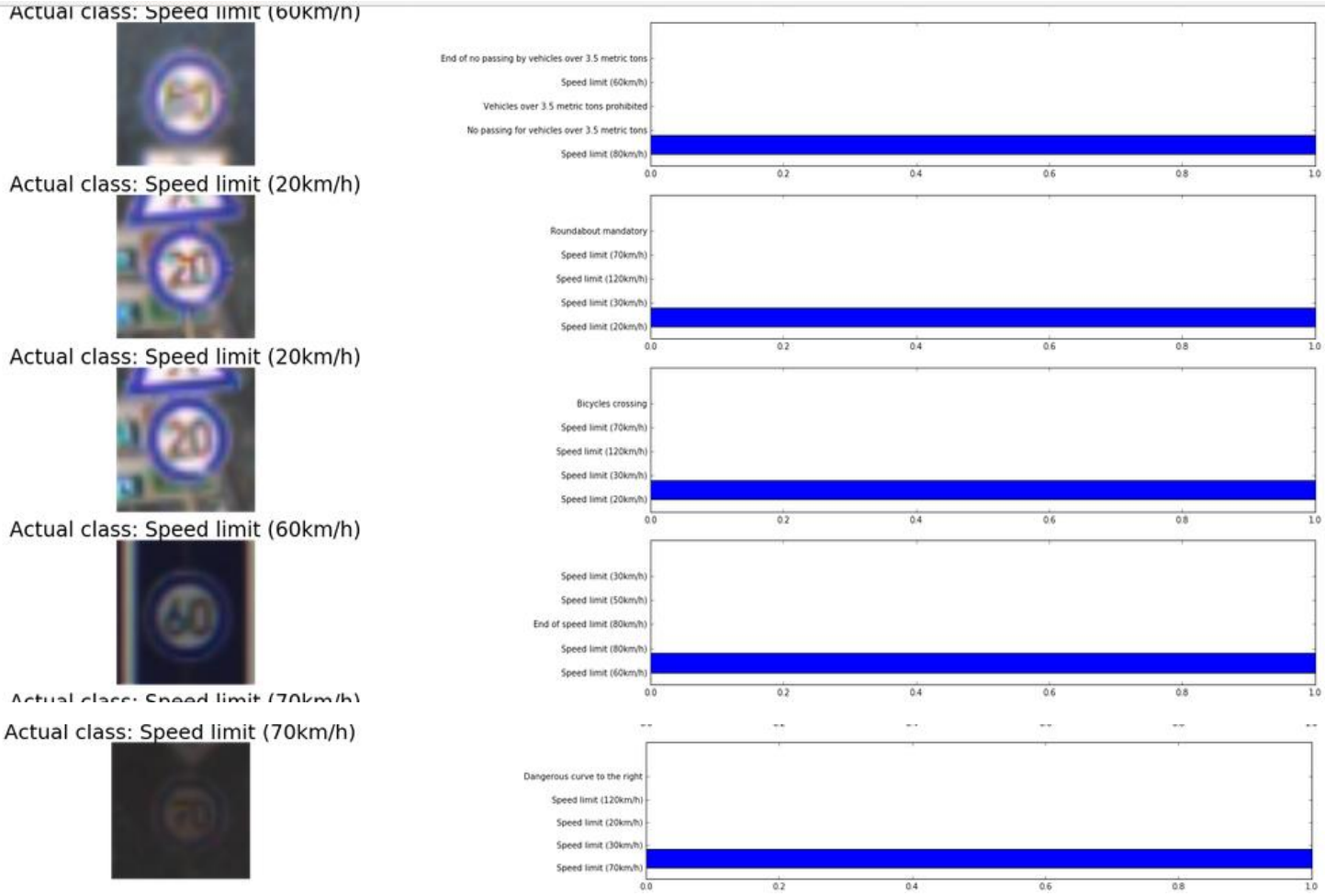


First image seems to be difficult to classify as casting of shadow seems to suggest that the traffic sign is End of Speed Limit 80 instead of traffic sign speed limit 80. Also, another curve ball for the network is traffic sign speed limit 60. It almost looks like speed limit 80, which is even confusing for the human 😊 As anticipated, network fails to recognize this image as speed limit 60. Despite the brightness or the lack of it, network manages to recognize the images as speed limit sign 70. Next session displays the predicted sofmax probabilities of the network.

Top 5 Softmax probabilities

Below is the listing of the actual class and plot of predicted softmax probabilities:





Model could correctly classify 7/9 images correctly resulting in a 77.78% accuracy. This is lower than that of Test set because of the smaller sample size and curve ball images in the test set. Looking at the softmax probabilities it is quite evident that model is quite confident even when making mistakes 😊

Visualizing the Network

I have tried to visualize the features learnt by the network in each layer. At the time of writing, author is facing issues with manipulating tensors to generate the visuals. Expecting to fix this issue soon.