Traffic Sign Classification

Summary

The objective is to create a classifier to correctly classify German traffic signs via deep learning. The project includes a German traffic sign dataset. The classifier that I design have an accuracy of 93.6% on a test dataset.

Data Set Summary & Exploration

The German traffic sign dataset are broken down into training set, validation set, and test set. In all sets, there are images and their corresponding labels. Here are the data of the dataset.

Number of training examples = 34799

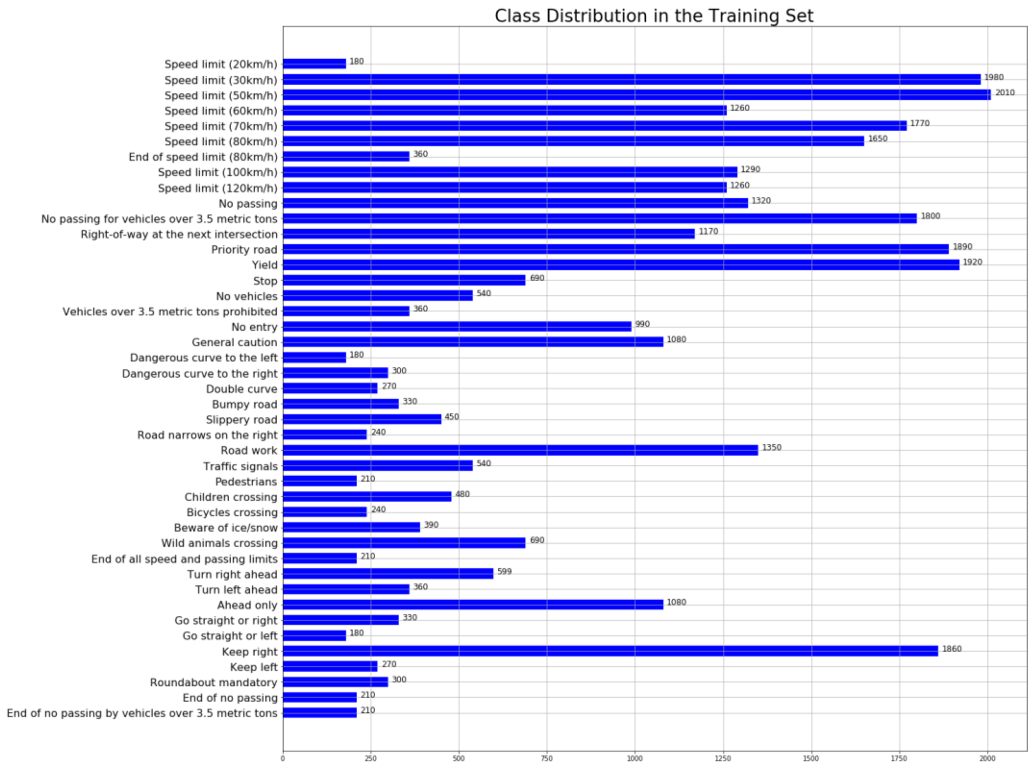
Number of Validation examples 4410

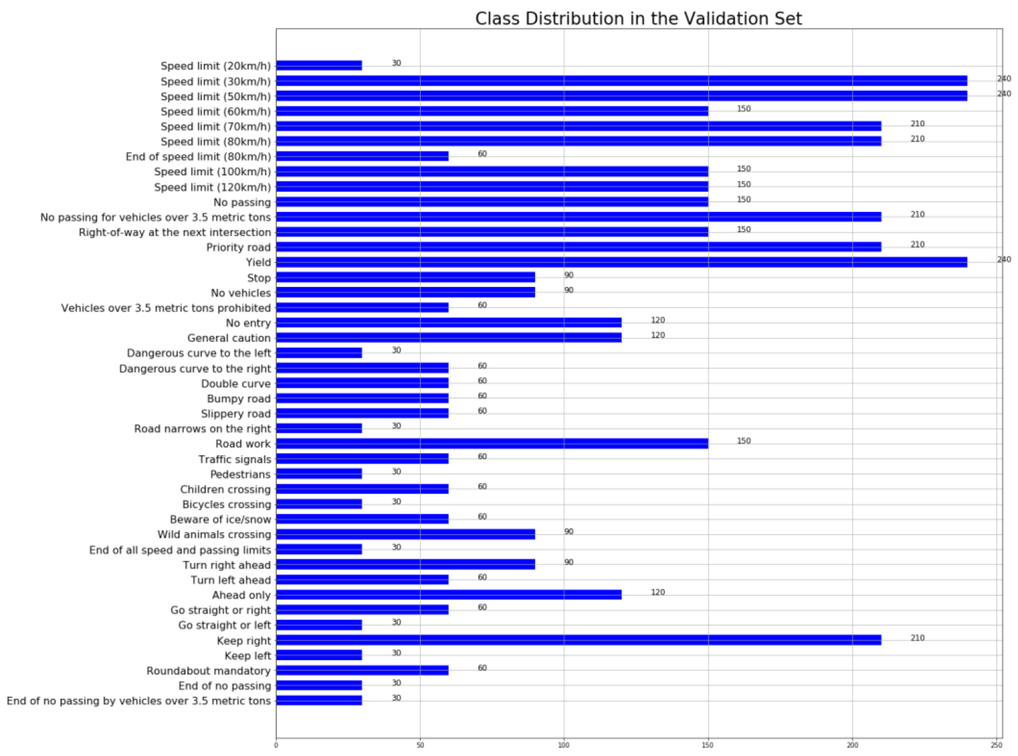
Number of testing examples = 12630

Image data shape = (32, 32, 3)

Number of classes = 43

I plot all the classes and their number of occurrences in the training set and validation set.





The dataset turns out to be unbalance in both training set and validation class. The classifier when train on this dataset may become bias towards the classes with more occurrences.

Design and Test a Model Architecture

First the dataset needs to be preprocessed so the neural network classifier can learn efficiency. I compared the validation result when the image is in color or in grayscale and I find grayscale to work better. While the neural network will gain a bit more information with the dataset is in color, the features size of the dataset increases 3 times when compare to grayscale. This will result in the curse of dimensionality problem. Therefore, I convert all the images to grayscale. I also scale the image to 0-1 instead of 0-255. This will allow for neural network to learn efficiency.

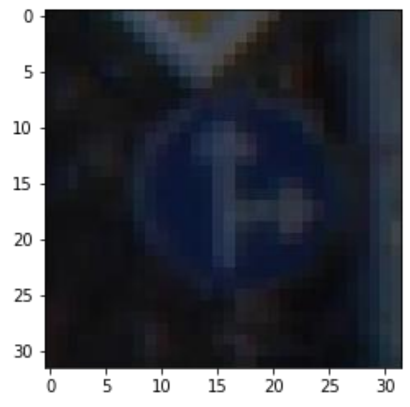


Image Before Preprocess (32,32,3)

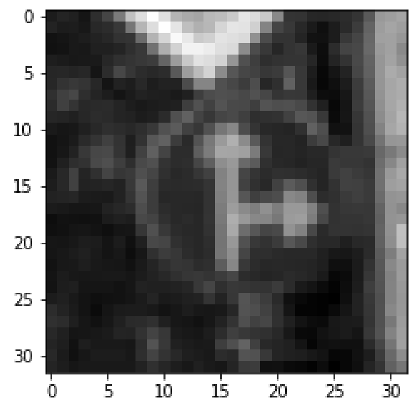


Image After Preprocess (32,32,1)

Design and Test a Model Architecture

I trained and validate both LeNet and AlexNet architecture and I find LeNet to achieve better result. When I preprocess the dataset by changing to grayscale and normalizing the dataset to 0-1. The LeNet network can achieve the validation accuracy greater than 93%. Therefore, I selected LeNet to the architecture for this classifier.

The LeNet consist of the following layers

Layer	**Description**
Inputi	32X32X1 Grayscale Image
Convolution 5X5	Convolution with 5X5 Kernel, 1X1 Stride and 6 filters Output: 28X28X6
Leaky Relu Activation	Max(a*x,x) where a = 0.001 Same Output
Max Pooling Layer	2X2 with 2X2 Stride Output: 14X14X6
Convolution 5X5	Convolution with 5X5 Kernel, 1X1 Stride and 16 filters Output: 10X10X16
Leaky Relu Activation	Max(a*x,x) where a = 0.001 Same Output
Max Pooling Layer	2X2 with 2X2 Stride Output: 5X5X16
Flatten	Output: 400
Fully connected	Output:120
Leaky Relu Activation	Max(a*x,x) where a = 0.001 Same Output
Dropout	Make 50% of the previous value to 0 during training time.
Fully connected	Output:84
Leaky Relu Activation	Max(a*x,x) where a = 0.001 Same Output
Dropout	Make 50% of the previous value to 0 during training time.
Fully connected	Output: number of classes(43)
Softmax	Convert all classes to probability so that the sum is equal to 1.

For the training I used the Adam optimizer with the following hyper parameters.

EPOCHS = 100

BATCH_SIZE = 128

LEARNING RATE = 0.0005

Keep Probability = .5

Using these parameters, the network obtained the following results:

Train Accuracy = 0.999

Validation Accuracy = 0.968

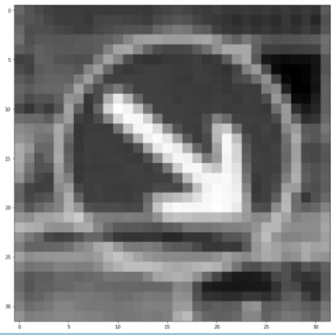
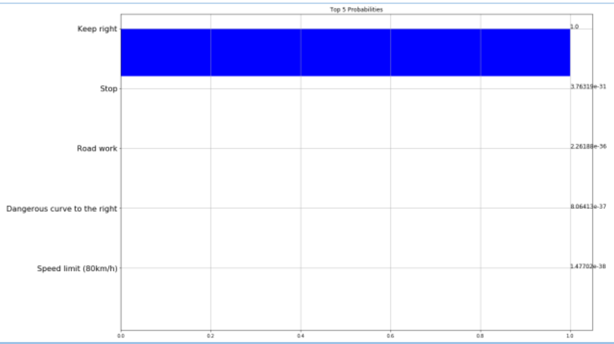
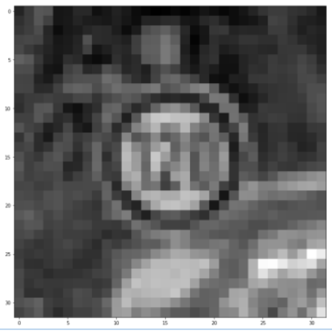
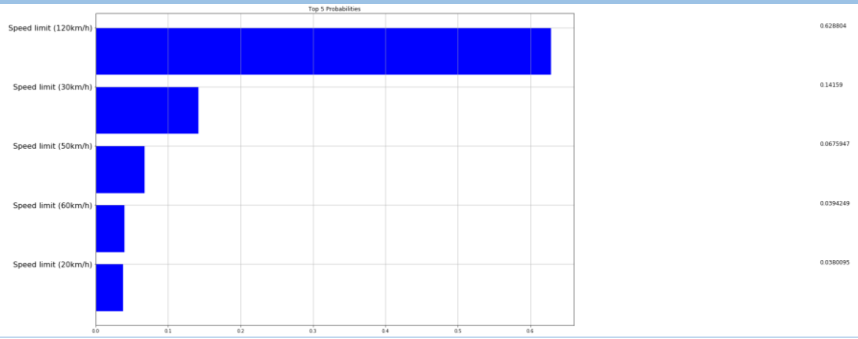
Test Accuracy = 0.936

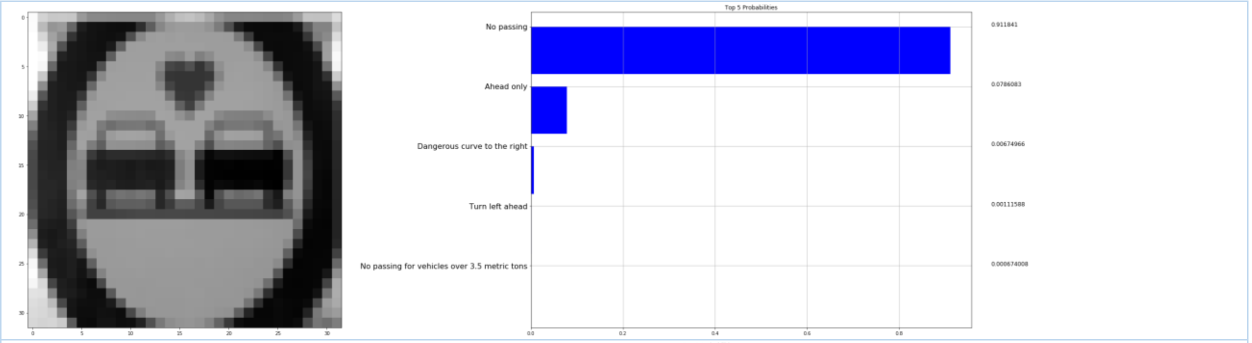
Test a Model on New Images

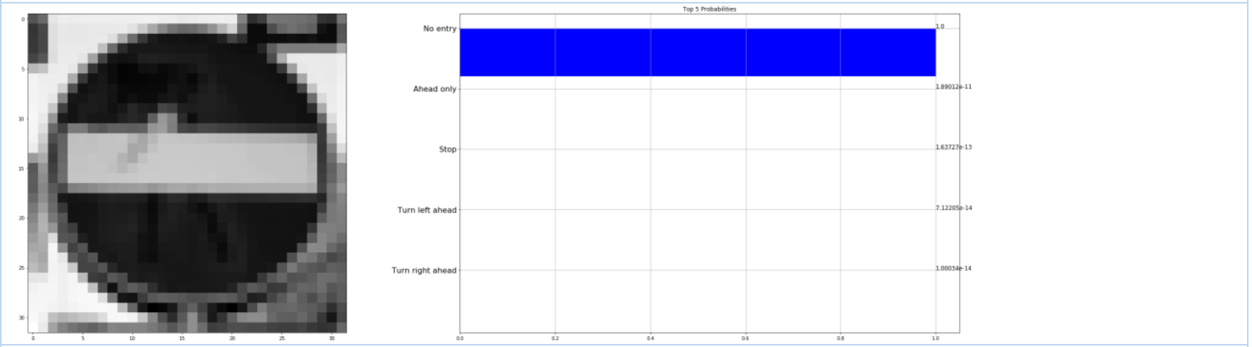
The following images where updated by google earth street view and by web pages.

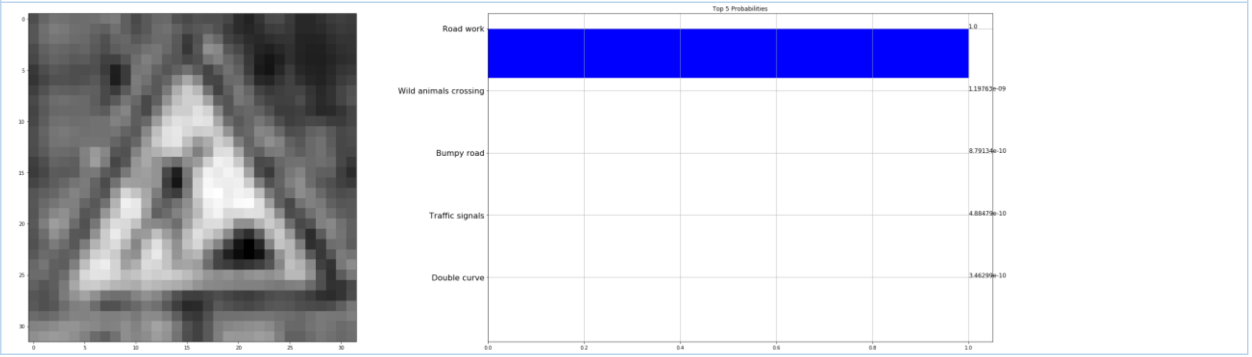
120km/h	**No Passing**	**No Entry**	**Road Work**	**Keep Right**

Notices that some of the sign were altered by people. I want to see what kind of affect the alterations have on the network. The following table below shows the result of each image. It shows the top 5 labels along with the probabilities of the top 5 labels.







In all 5 images, the network was able to identify correctly. Therefore, it has an accuracy of 100% for these 5 tests case.

Since I only test on only 5 images, it's not enough to judge the performance of the network. Also, accuracy doesn't give a complete picture of the network performance. Therefore, I want to compute the Recall and Precision for each class.

Recall and Precision

Recall and Precision are two metric that can be computed. Recall is defined by the probability of the network giving a true positive (correctly identify the class) giving that the ground truth of the test data is of a specify class. Precision is defined by the probability of the a true positive (correctly identify the class) when the classifier predicts the result of the specified class.

Speed limit (20km/h) Recall: 0.95 | Precision: 1.00Speed limit (30km/h) Recall: 0.98 | Precision: 0.91

Speed limit (50km/h) Recall: 0.96 | Precision: 0.95Speed limit (60km/h) Recall: 0.98 | Precision: 0.92

Speed limit (70km/h) Recall: 0.93 | Precision: 0.97Speed limit (80km/h) Recall: 0.91 | Precision: 0.96

End of speed limit (80km/h) Recall: 0.84 | Precision: 0.98

Speed limit (100km/h) Recall: 0.88 | Precision: 0.99

Speed limit (120km/h) Recall: 0.96 | Precision: 0.94

No passing Recall: 0.99 | Precision: 0.96

No passing for vehicles over 3.5 metric tons Recall: 0.99 | Precision: 0.99

Right-of-way at the next intersection Recall: 0.93 | Precision: 0.80

Priority road Recall: 0.98 | Precision: 0.98

Yield Recall: 0.99 | Precision: 0.99

Stop Recall: 0.93 | Precision: 0.89

No vehicles Recall: 0.99 | Precision: 0.92

Vehicles over 3.5 metric tons prohibited Recall: 0.99 | Precision: 0.99

No entry Recall: 0.99 | Precision: 0.99

General caution Recall: 0.86 | Precision: 0.89

Dangerous curve to the left Recall: 1.00 | Precision: 0.87

Dangerous curve to the right Recall: 0.82 | Precision: 0.82

Double curve Recall: 0.58 | Precision: 0.80

Bumpy road Recall: 0.86 | Precision: 0.98

Slippery road Recall: 0.96 | Precision: 0.88

Road narrows on the right Recall: 0.53 | Precision: 0.80