Traffic Sign Classifier

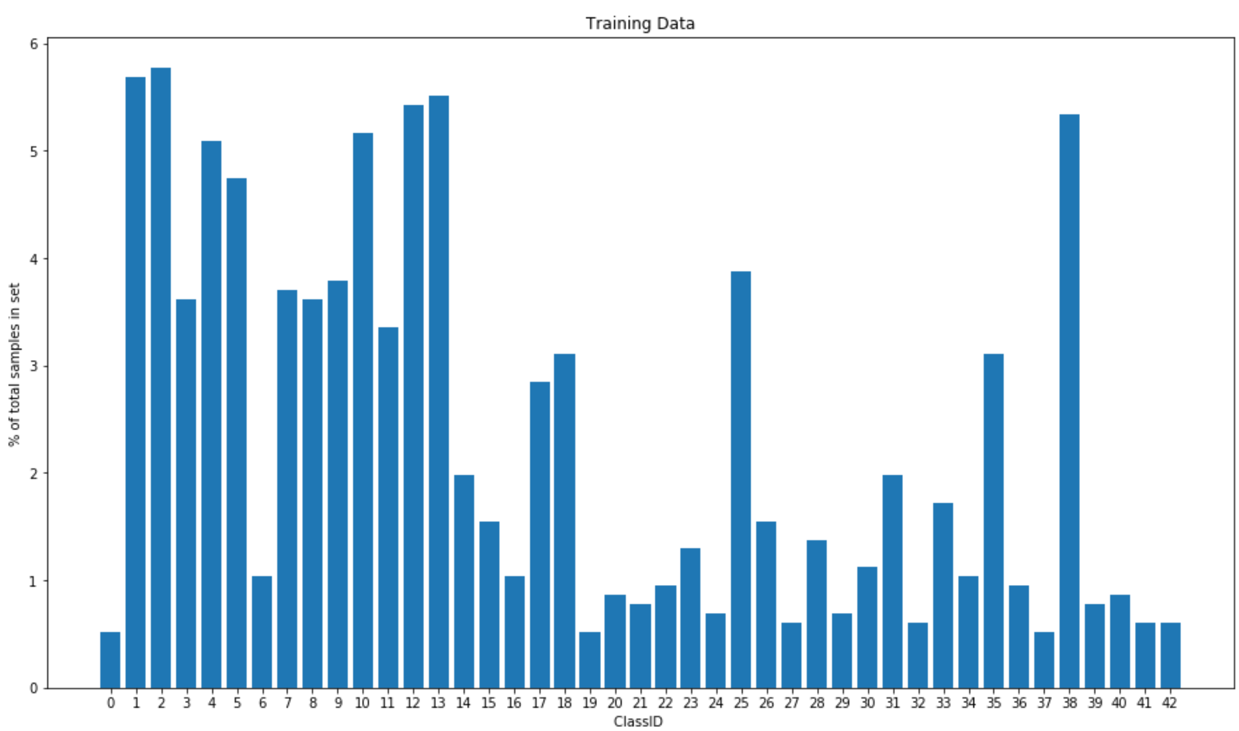
I loaded the dataset from files provided in the traffic-signs-data.zip folder and used basic python functions to extract the dataset summary.

- 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
- Number of total classes = 34799

For the purpose visualization, I used matplotlib.pyplot to graph the occurances of each of the 43 different traffic signs in the dataset. The dataset is extracted into a dictionary format using numpy, and then passed to pyplot to create bar charts. I scaled the y-axis of the graph from # to % so that all 3 datasets are directly comparable for relative rate of occurance.

For example, the output of the training data in dictionary format looks like this.

{0: 180, 1: 1980, 2: 2010, 3: 1260, 4: 1770, 5: 1650, 6: 360, 7: 1290, 8: 1260, 9: 1320, 10: 1800, 11: 1170, 12: 1890, 13: 1920, 14: 690, 15: 540, 16: 360, 17: 990, 18: 1080, 19: 180, 20: 300, 21: 270, 22: 330, 23: 450, 24: 240, 25: 1350, 26: 540, 27: 210, 28: 480, 29: 240, 30: 390, 31: 690, 32: 210, 33: 599, 34: 360, 35: 1080, 36: 330, 37: 180, 38: 1860, 39: 270, 40: 300, 41: 210, 42: 210}



From the visualization, it's possible to conclude that the rate of occurance of each type of signs in all 3 datasets are more or less about the same. This suggest that the distribution is balanced and we should expect the validation accuracy to be comparable to the test accuracy.

I preprocessed the data by applying a basic normalization scheme to all 3 color channels. The algorithm is (pixel - 128)/ 128. I chose not to grayscale the image because I believe that color information is useful in helping to distinguish certain signs (for example, the stop sign is always red).

The network architecture I selected is based on LeNet with very minor modifications.

Layer	**Description**
Input	32x32x3 RGB image
Layer 1 Convolution	1x1 stride, 5x5 filter, VALID padding, 28x28x6 output
Layer 1 Activation	RELU function
Layer 1 Pooling	2x2 stride, MAX poling, 14x14x6 output
Layer 2 Convolution	1x1 stride, 5x5 filter, VALID padding, 10x10x16 output
Layer 2 Activation	RELU function
Layer 2 Pooling	2x2 stride, MAX poling, 5x5x16 output
Flatten	400 output
Fully connected Layer 3	120 output
Layer 3 Activation	RELU function
Layer 3 Dropout	Prevent overfitting
Fully connected Layer 4	84 output
Layer 4 Activation	RELU function
Layer 4 Dropout	Prevent overfitting
Fully connected Layer 5	43 output
Logits	Logits

The rest of the code is directly ported from the LeNet lab. I did not change the hyper parameter settings

Initialized random weight and bias distribution

- mu = 0
- sigma = 0.1

LeNet was chosen because the complexity of the MNIST dataset images are comparable to the traffic sign images in pixel dimensions, so it's a natural starting point. For training, I used the following hyperparameter settings to achieve the required accuracy. During training, I set the drop out to 50% to prevent overfitting the training set. Once I see that the validation accuracy has stabilized above 93%, I ran it through the test dataset once.

- EPOCHS = 20
- BATCH_SIZE = 128
- rate = 0.001

The final results are as follows

- After the last epoch, validation accuracy is 95.6%
- Test accuracy is 93.6%

I found 5 German traffic signs on the web and scaled the image down to 32 by 32 pixels.

Image					
Sign	Speed limit (30km/h)	Yield	Stop	Ahead only	No vehicles
Label	1	13	14	35	15
Predicted Label	1	13	33	28	12
Classification Challenges	The sign is not perpendicular to the axis of the camera lens. Visible tilt and rotation	Over cropping the edges might make it difficult to identify	Relatively few samples in datasets (only ~ 2%)	Background color is noisy	Relatively few samples in datasets (only ~ 2%)
	Color palette is different from training set due to lighting,	Color palette is different from training set due to lighting,	Color palette is different from training set due to lighting,	Color palette is different from training set due to lighting,	Color palette is different from training set due to lighting,

	shade, exposure, contrast, etc.	shade, exposure, contrast, etc.	shade, exposure, contrast, etc.	shade, exposure, contrast, etc.	shade, exposure, contrast, etc.

Prediction based on the new sets of images is 40%, compared to 93.6% on test set. While this is less than I would have hoped, it shows that the model is working. The abnormally low prediction accuracy is likely due to color differences, and the fact that the new set has a very small number of samples. If the new set had a comparable number of samples compared to the test set, I would expect the overall prediction accuracy to be much closer to that of the test set.

Softmax probability output

Image 1	Probability	0.6657	0.3323	0.0018	0.0002	0.0000
	ClassID	1	37	4	39	18
	Label	Speed limit (30km/h)	Go straight or left	Speed limit (70km/h)	Keep left	General caution
Image 2	Probability	1.0000	0.0000	0.0000	0.0000	0.0000
	ClassID	13	12	35	14	36
	Label	Yield	Priority road	Ahead only	Stop	Go straight or right
Image 3	Probability	0.9829	0.0110	0.0037	0.0021	0.0002
	ClassID	33	35	40	34	37
	Label	Turn right ahead	Ahead only	Roundabout mandatory	Turn left ahead	Go straight or left
Image 4	Probability	0.8729	0.0865	0.0184	0.0134	0.0030
	ClassID	28	26	12	15	32
	Label	Children crossing	Traffic signals	Priority road	No vehicles	End of all speed and passing limits
Image 5	Probability	1.0000	0.0000	0.0000	0.0000	0.0000
	ClassID	12	15	14	26	17
	Label	Priority road	No vehicles	Stop	Traffic signals	No entry

Image 1 – model is 66% sure that it is a Stop sign (correct!)

Image 2 – model is 100% sure that it is a Yield sign (correct!)

Image 3 – model is 98% sure that it is a Turn right ahead sign (incorrect!)

Image 4 – model is 87% sure that it is a Children crossing sign (incorrect!)

Image 5 – model is 100% sure that it is a Priority road sign (incorrect!)