Traffic Sign Recognition

Link to project code:

https://github.com/sethbunke/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

https://github.com/sethbunke/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.html

Summary of training and testing data:

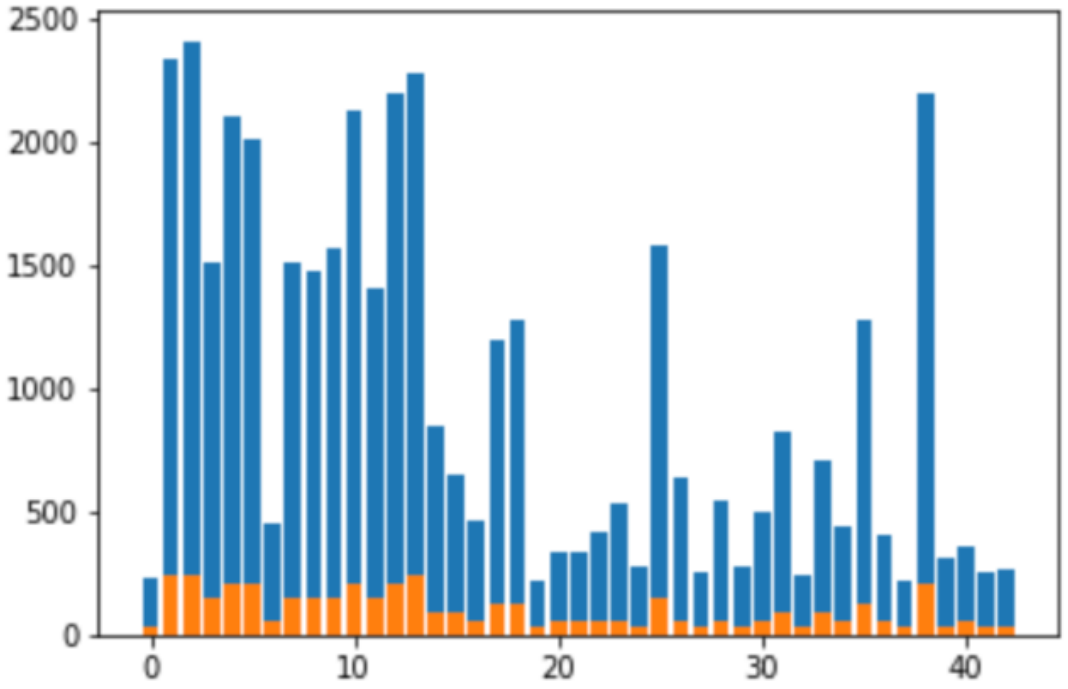
Number of training examples = 34799

Number of testing examples = 12630

Image data shape = (32, 32, 3)

Number of classes = 43

Exploratory visualization of dataset. Initial research of the data sets that were provided indicated very imbalanced data. To attempt to address this I concatenated the training, testing, and validation sets, shuffled that data and split that data between training and validation sets; 80% and 20%, respectively. Please see the notebook for percentages.



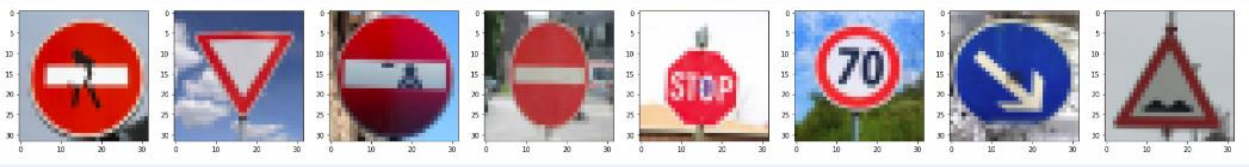
Preprocessing of image data. The preprocessing of the data was limited to min-max normalizing of the data. When working with neural networks data normalization is crucial as one learning rate is used throughout the network and if the values are not scaled consistently the learning rate could potentially cause over or under-corrections of the weights for each feature. While I did explore converting the images to grayscale as well as augmenting the images (scale and position/translate – see submitted notebook), I did not apply the techniques to the dataset as I was ultimately able to get ~98% accuracy through a combination of tweaking hyper-parameters and changes to the network architecture. The most significant changes that I made to the architecture (which was based on LeNet), was to increase the depth of the filter on the second convolutional layer to 48 instead is 16 – meaning it would be able pick up more details in every convolution. Also, adding two drop-out layers with 80% keep probabilities.

Model Architecture:

Layer	Description
Input	**32x32x3 RGB**
Conv 5x5	**28x28x6**
Relu	
MaxPool	**2x2**
Conv 5x5	**10x10x48**
MaxPool	**5x5x48**
Flatten	**1200**
Fully Connected	**120**
Relu	
Drop-out	**Keep 80%**
Fully Connected	**84**
Relu	
Drop-out	**Keep 80%**
Fully Connected	**43**

Training the Model. Much of what was provided by LeNet was used. Through the process of trial and error I determined that the optimal number of epochs was 15; interestingly, even though the accuracy starts to dip after the 13th epoch going to 15 epochs does result in an approximately 7% increase in accuracy against the new images from the internet which ended up being at ~88%. Additionally, while Keras offer the functionality of "early termination" of training when the accuracy begins to drop, I did not find that TensorFlow offers this functionality "out of the box". I also found that changing batch sizes did not produce an appreciable difference in accuracy when going above or below 128 images per batch.

Testing on New Images. The 8 images included below were obtained from the internet and used to test the model.



A number of these images could be difficult to classify as from what I am finding many of the signs contain very similar features – with very subtle differences. For instance image 1, 3, and 4 (from left to right) are extremely similar and could be difficult even for humans to distinguish based on the sign's shape, the colors, and markings; images 2 and 8 could be equally difficult to classify for similar reason. Also, it's easy to image that many images in the dataset closely resemble image 6 – for instance a speed limit sign for 10 KPH. See the actual results discussed below.

Test Results on New Images. The model was 75% accurate on the 8 images.

image1.png:

Vehicles over 3.5 metric tons prohibited: 100.00%

SignName: 0.00%

Speed limit (100km/h): 0.00%

Wild animals crossing: 0.00%

Yield: 0.00%

image2.png:

Priority road: 100.00%

No passing: 0.00%

Speed limit (20km/h): 0.00%

Keep right: 0.00%

Yield: 0.00%

image3.png:

Vehicles over 3.5 metric tons prohibited: 100.00%

Wild animals crossing: 0.00%

SignName: 0.00%

Speed limit (100km/h): 0.00%

No entry: 0.00%

image4.jpg:

Vehicles over 3.5 metric tons prohibited: 51.66%

Bicycles crossing: 46.99%

Pedestrians: 0.95%

Dangerous curve to the left: 0.38%

Turn right ahead: 0.01%

image5.png:

Yield: 100.00%

Vehicles over 3.5 metric tons prohibited: 0.00%

Speed limit (70km/h): 0.00%

Speed limit (20km/h): 0.00%

Speed limit (100km/h): 0.00%

image6.png:

Priority road: 100.00%

Speed limit (50km/h): 0.00%

Speed limit (120km/h): 0.00%

Road narrows on the right: 0.00%

Children crossing: 0.00%

image7.png:

Go straight or left: 100.00%

Keep left: 0.00%

Road narrows on the right: 0.00%

Turn right ahead: 0.00%

Double curve: 0.00%

image8.jpg:

No entry: 99.37%

Road narrows on the right: 0.31%

Beware of ice/snow: 0.17%

Traffic signals: 0.12%

No passing for vehicles over 3.5 metric tons: 0.02%

Visualizing the Neural Network. Unfortunately I was not able to get the visualization to work. Any feedback on what I am doing incorrectly will be appreciated.