Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Rubric Points

Data Set Summary & Exploration

1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the **numpy** library to calculate summary statistics of the traffic signs data set:

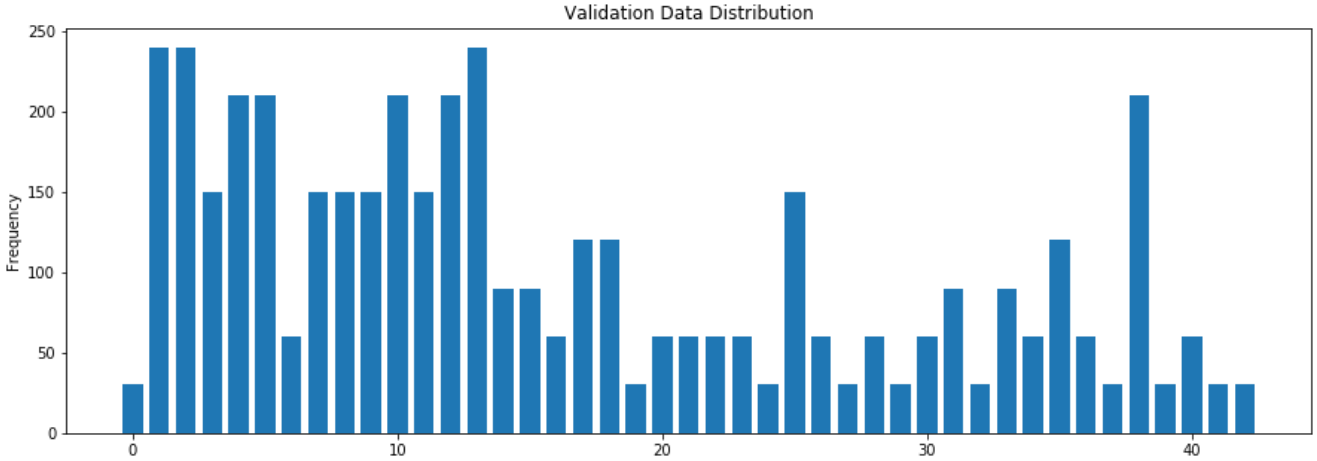
Number of training examples = **34799**

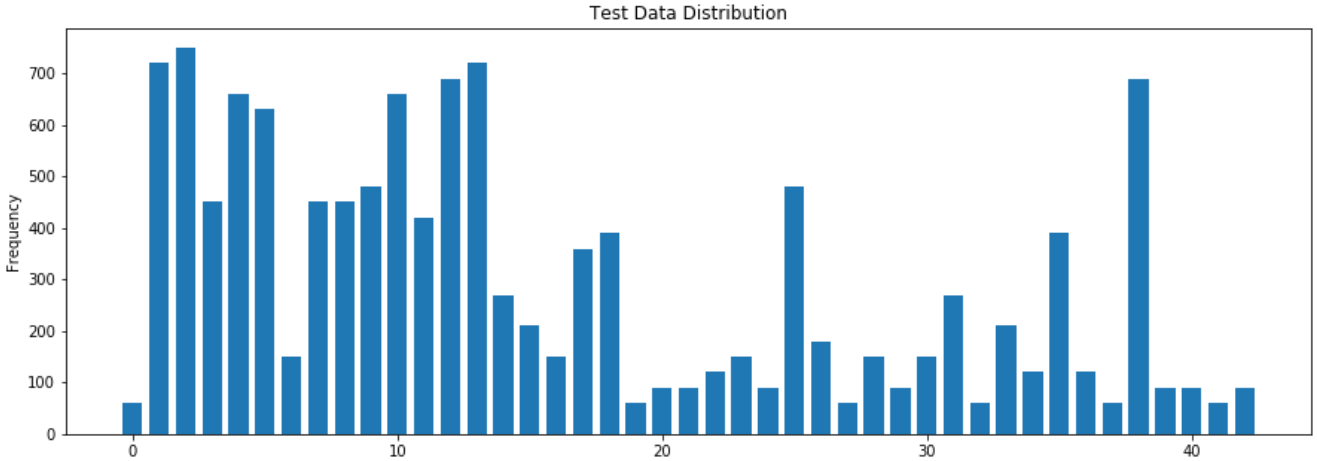
Number of validation examples = **4410**

Number of testing examples = **12630**
Image data shape = **(32, 32, 3)**
Number of classes/labels = **43**

2. *Include an exploratory visualization of the dataset.*

Random images with the corresponding labels. Histograms are plotted for all the three data sets, and the distribution is consistent which makes a good data set to work with in training, validating and testing





Design and Test a Model Architecture

1. *Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.*

As the data has **consistent distribution** across the training, validation and test sets, no effort is spent in augmenting the training data sets. Also, the images were of the size 32x32x3 (small enough to not warrant grayscaling). However, all the images in each of the data set are normalized by their means and sigmas so that the normalized mean is zero. Such normalized data sets provide better numerical stability and faster convergence. The simple mean normalization, in this case, results in an exact copy of the image (no change)

There is image processing done on the new test images later in the section. These images are pre-processed to have the same size as the images in the training set. The model architecture designed in this section works for only 32x32x3 images.

2. *Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.*

My final model consisted of the following layers (**classic LeNet with max pooling and drop-outs** to prevent over-fitting)

I used Max pooling for convoluted layers and drop-outs for FC layers

Layer	Description
Input	32x32x3 RGB image
Convolution 5x5x3x6	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5x6x16	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16

Layer	Description
Fully connected	400 to 120
RELU and then drop-out	
Fully connected	120 to 84
RELU and then drop-out	.
Fully connected	84 to 43
Softmax, cross entropy and minimize loss using optimizer	.

3. *Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.*

The model was trained with the following parameters:

Learning rate = 0.001

Batch-size = 100

Epochs = 20 (it started to saturate, and I got around 0.5% increase in accuracy of validation set by increasing Epochs to 60).

keep_prob = 0.5 for training the model where drop-outs are used.

To train the model, I used **Adam optimizer** as discussed in LeNet lab. It looks like this optimizer has combined the benefits of both momentum and adagrad together and in a memory efficient way

4. D*escribe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.*

My final model results were:

- training set accuracy of **99.7%**
- validation set accuracy of **96.1%**
- test set accuracy of **95%**

After training the model and saving it, all the 3 accuracies are calculated so that they are at one go. Test set accuracy is not used at all in refining the model. Only validation accuracy was used as a metric in training the model.

Screenshot is included here:

```
EPOCH 17 ...
Validation Accuracy = 0.958

EPOCH 18 ...
Validation Accuracy = 0.957

EPOCH 19 ...
Validation Accuracy = 0.959

EPOCH 20 ...
Validation Accuracy = 0.961

Model saved
```

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    saver = tf.train.import_meta_graph('./lenet.meta')
    saver.restore(sess, "./lenet")
    Training_accuracy = evaluate(X_train_norm, y_train)
    print("Training Accuracy = {:.3f}".format(Training_accuracy))
    Validation_accuracy = evaluate(X_valid_norm, y_valid)
    print("Validation Accuracy = {:.3f}".format(Validation_accuracy))
    test_accuracy = evaluate(X_test_norm, y_test)
    print("Test Set Accuracy = {:.3f}".format(test_accuracy))
```

```
Training Accuracy = 0.997
Validation Accuracy = 0.961
Test Set Accuracy = 0.950
```

- *What architecture was chosen?*

Well known **LeNet architecture as discussed in LeNet lab was chosen**. Max pooling was done after convnets/RELU as well as drop-outs post fully-connected layers and RELU.

- *Why did you believe it would be relevant to the traffic sign application?*

This is a good model, as our data set has images **32x32x3 close to classic LeNet dataset (32x32x1)**. The frame work for 2 convnets and 3 FC layers is suitable for this traffic sign application.

- *Which parameters were tuned? How were they adjusted and why?*

Learning rate (0.001) was chosen same as LeNet model. **Batch size was chosen as 100**. Number of epochs was a hyper-parameter, and it was noticed that the validation accuracy topped out around **20 Epochs**.

Validation accuracy (~84%)was much lower compared to training accuracy, which implies that there is over-fitting. Enabling drop-out with k**eep_prob as a hyper-parameter (0.5)** helped to provide a better training model that resulted in improved validation accuracy.

Also, I attempted to use drop-out post convnet/RELU instead of max pooling. This resulted in too many parameters for FC layers with little to no improvement in validation accuracy. So, the decision was made to stick to the classic LeNet model.

- *How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?*

We expect high training accuracy, as the model is being trained on the training data. Model refinement (with max pool and dropout) yielded validation accuracy of 96%, and the test-set that is completely blind to the model has yielded similar accuracy of 95%

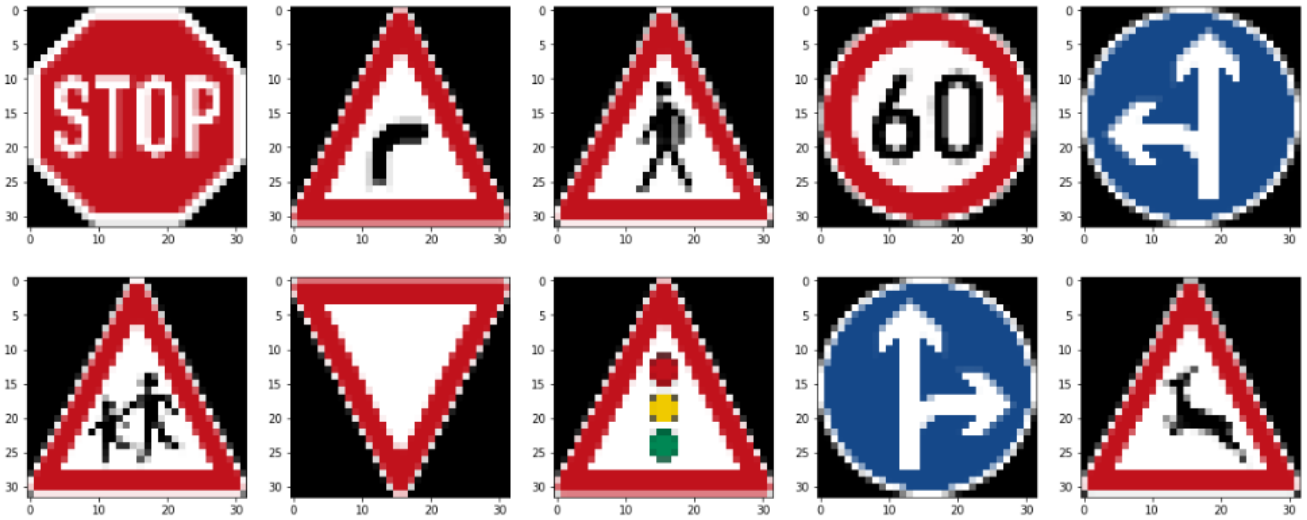
Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are ten German traffic signs that I found on the web:

https://en.wikipedia.org/wiki/Road_signs_in_Germany

I picked the images where the number of instances of the image label in the training set are mixed. Also, these are much cleaner images with no background.



I expect the accuracy of these test images close to 100%. Also, I expect soft-max probabilities to be lower for images 2,3,6,8,10 (counting along the rows, top row first and then bottom row), as there is finer detail in these images (could be mis-classified or classified right with lower probability). The rest of the images (1,4,5,7,9) should not only be accurate with softmax probabilities close to 1 for the correct label.

Labels for the above test-images = [14,20,27,3,37,28,13,26,36,31]

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:

Label	Image	Prediction	Max Prob
14	Stop	Stop	0.99997
20	Dangerous curve to the right	Dangerous curve to the right	0.71383
27	Pedestrians	Pedestrians	0.97227
3	Speed limit (60km/h)	Speed limit (60km/h)	0.97334
37	Go straight or left	Go straight or left	0.99048
28	Children crossing	Children crossing	0.6485
13	Yield	Yield	1
26	Traffic signals	Traffic signals	0.99997
36	Go straight or right	Go straight or right	1
31	Wild animals crossing	Wild animals crossing	0.64595

The model was able to correctly guess 10 of the 10 traffic signs, which gives an accuracy of 100%. This is also partly because of the cleaner data set with minimum background effects on my test images. I have also included the top softmax probability for each of the images

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

Top 5 softmax probabilities are printed in the ipynb along with relevant indices are printed in the attached ipynb.

The highlighted images even with current prediction has lower top softmax probability compared to others. Focus will be on these three images rather than all the 10 images.

In addition, the top 2 softmax probabilities add up close to 1. Only the top 2 labels are given in the following tables.

Prob	Label	Traffic Sign
7.14E-01	20	Dangerous curve to the right
2.86E-01	23	Slippery road
3.51E-07	25	
5.81E-08	41	
5.33E-08	30	

Prob	Label	Traffic Sign
6.49E-01	28	Children crossing
3.51E-01	29	Bicycles crossing
1.45E-05	0	
3.06E-06	8	
2.23E-06	20	

Prob	Label	Traffic Sign
6.46E-01	31	Wild animals crossing
3.54E-01	25	Road work
2.51E-06	29	
1.26E-06	24	
7.78E-08	21	

The top 5 softmax probabilities and the relevant indices are printed for each of the images in the ipynb.