Build a Traffic Sign Recognition Classifier

The goals / steps of this project are the following:

- Load the 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, based on the associated template.

Data Set Summary & Exploration

1. Basic summary of the data set

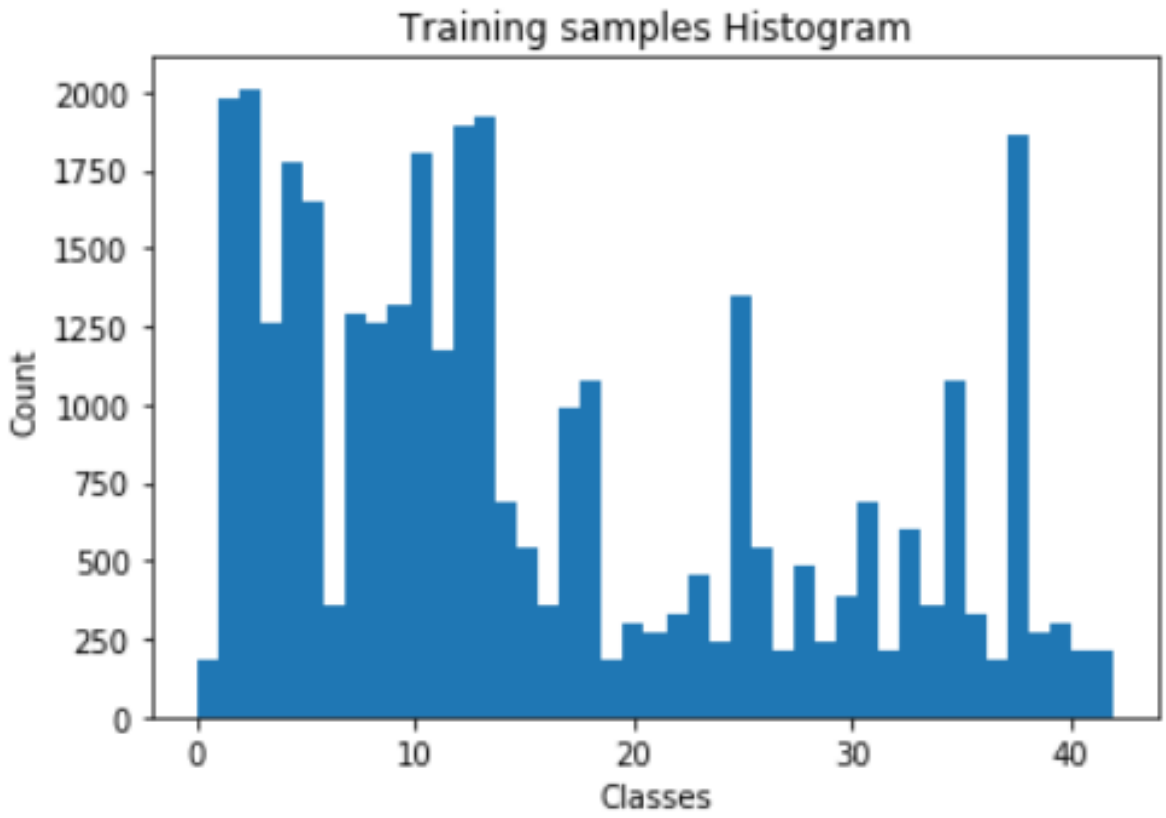
I used the numpy library to calculate summary statistics:

- The size of training set is 34799.
- The size of the validation set is 4410.
- The size of test set is 12630.
- The shape of a traffic sign image is (32,32,3).
- The number of unique classes/labels in the data set is 43.

2. Exploratory visualization of the dataset

First I explore 15 random images

Then, I show a histogram for the 43 types of signs to get a sense on how the samples are distributed.



Design and Test a Model Architecture

1. Image processing

I directly doubled the training set rotating the images a range of -20 to 20 degrees, some samples are:

The resulting data set is 69598 images.

I discarded the following techniques because later on checking the pipeline I got good results:

- Grayscale
- Normalization
- Contrast

A possible improvement is to augment data for those classes with fewer samples only. Anyway as the more data you have the better I decided to double the data set.

2. Model Architecture

The model consists of the following layers:

Layer	Description
l2_normalize	Use the standard L2 norm
Input	32x32x3 RGB image
Convolution 3x3	1x1 stride, Valid padding, outputs 28x28x6
ELU	

Layer	Description
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	1x1 stride, Valid padding, outputs 10x10x16
ELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	outputs 400
Fully Connected	outputs 120
ELU	
Dropout	
Fully Connected	outputs 84
ELU	
Fully Connected	outputs 43

3. Describe how you trained your model

To train the model, I used the suggested softmax operation with a minor tweak in favor of the softmax_cross_entropy_with_logits_v2 operation as was hinted on the execution output (tensorflow 1.5rc1).

The same optimizer AdamOptimizer was used.

I started with a 1 epoch and moved to 5, 10 and finally 30.

The batch size was kept the same to 128 images.

I tested the following values for the learning rate parameters (0.001, 0.005 0.0007, 0.01, 0.1), finally selecting 0.005.

When introducing a dropout operation the 0.5 and 0.75 values were tested finally selecting the later.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93

Initially I used the proposed pipeline based on the LeNet architecture with the necessary adjustments.

Doing minor modifications on the hyper-parameters didn't get better has the training accuracy was around 0.80.

I introduced the following modifications in a sequence of steps

- l2_normalize
- average vs max pooling
- dropout
- other activations
- increate training set

Normalize

Based on the external material provided on the course, I decided to try L2 normalization; I got some improvements so decided to keep it.

Pooling

I tested average vs max pooling; selected the latter as it got better results.

Activations

I incrementally tested several dropout operations to prevent overfitting. At the end of the pipeline, finally kept only one operation after the first fully connected network keeping 0.75.

I decided to try different activation operations; using an activation of exponential linear units tf.nn.elu instead of the proposed tf.nn.relu it got much better results.

Training Set

I decided to increate the training set augumenting the available data with random image rotations, it improved the results.

Results

My final model results were the following.

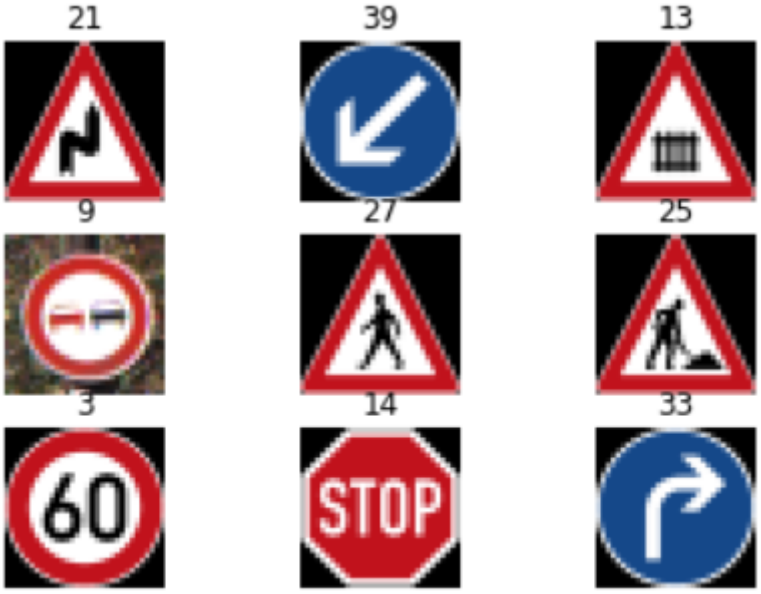
Training...
EPOCH 1 ... Validation Accuracy = 0.834
EPOCH 2 ... Validation Accuracy = 0.914
….
EPOCH 29 ... Validation Accuracy = 0.958
EPOCH 30 ... Validation Accuracy = 0.953

Test Accuracy = 0.94

Test a Model on New Images

1. Choose five German traffic signs found on the web and provide them in the report

Here are 9 German traffic signs that I found on the web:



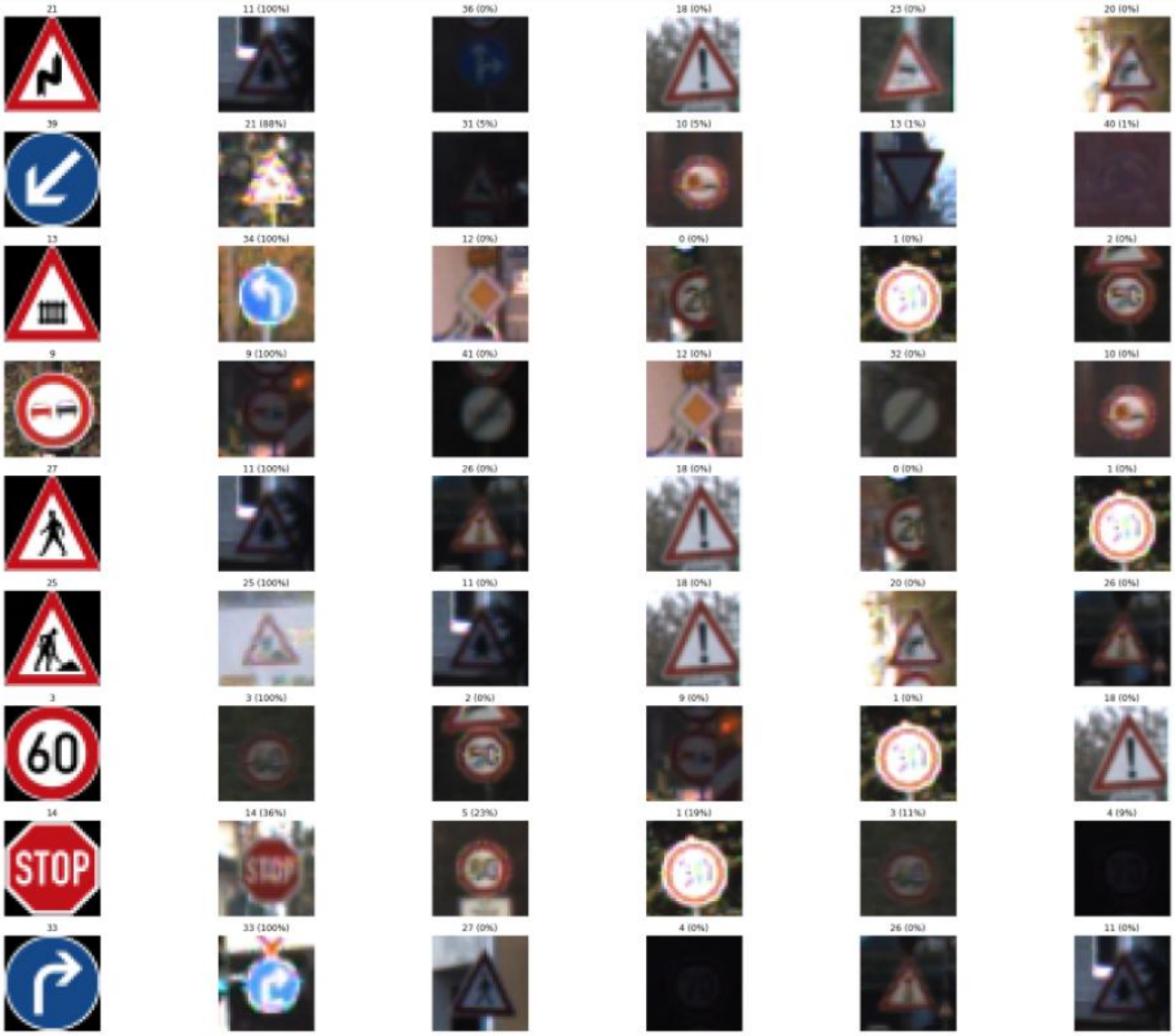
2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set

The model was able to correctly guess 5 of the 9 traffic signs provided, which gives an accuracy of 55.56%.

Honestly I did not expect to get such a bad result considering the augmentation phase adding more samples to the training set and the results on the provided dataset.

3. Describe how certain the model is when predicting on each image by looking at the softmax probabilities for each prediction

The selected images and top 5 softmax probabilities for each image are displayed below:



The first column details the original image with the class on top, then the top 5 predictions and the associated percentage.

Image	Comments
21(Double curve)	0 match, all suggestions wrong
39(Keep left)	0 match, all suggestions wrong
13(Yield)	0 match, all suggestions wrong
9(No passing)	100% match

Image	Comments
27(Pedestrians)	0 match
25(Road work)	100% match
3(Speed limit 60km/h)	100% match
14(Stop)	100% match
33(Turn right ahead)	100% match

Based on the histogram of classes the following ones have fewer samples so it may be difficult to get an accurate prediction:

- 14, 27, 33

One option would be to add testing data with new cases for those classes.

(Optional) Visualizing the Neural Network

Not achieved yet…