**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

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

---

You're reading it! and here is a link to my project code

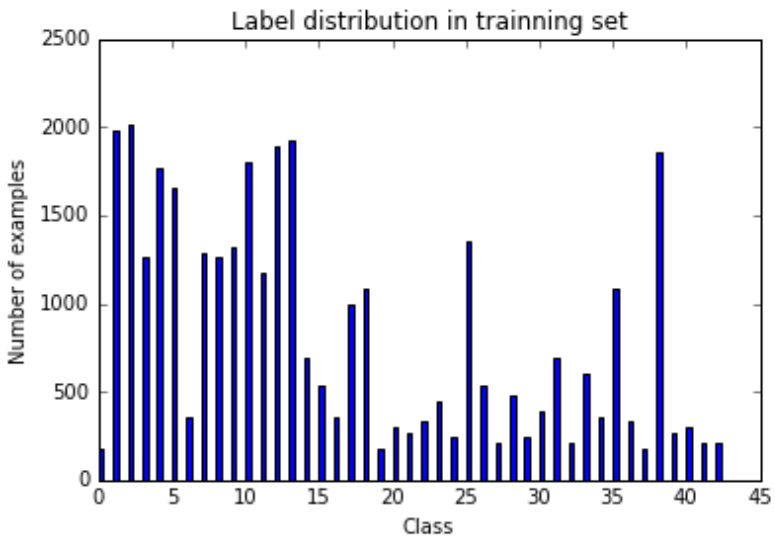
Data Set Summary & Exploration

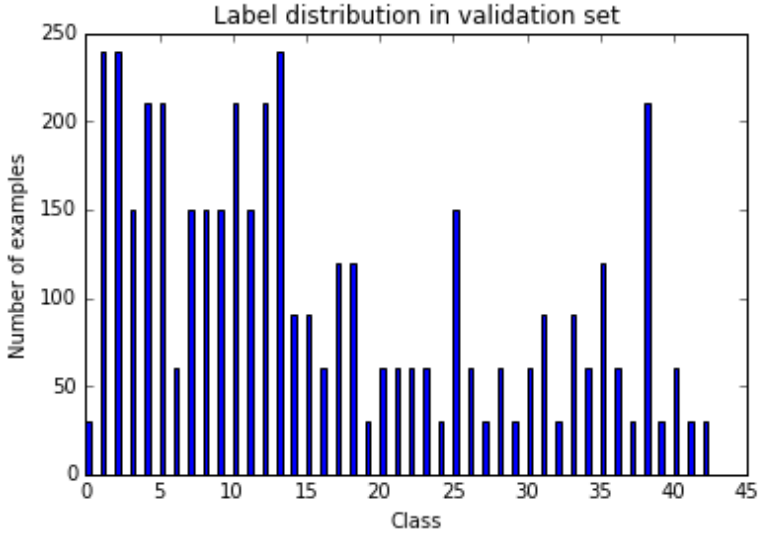
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

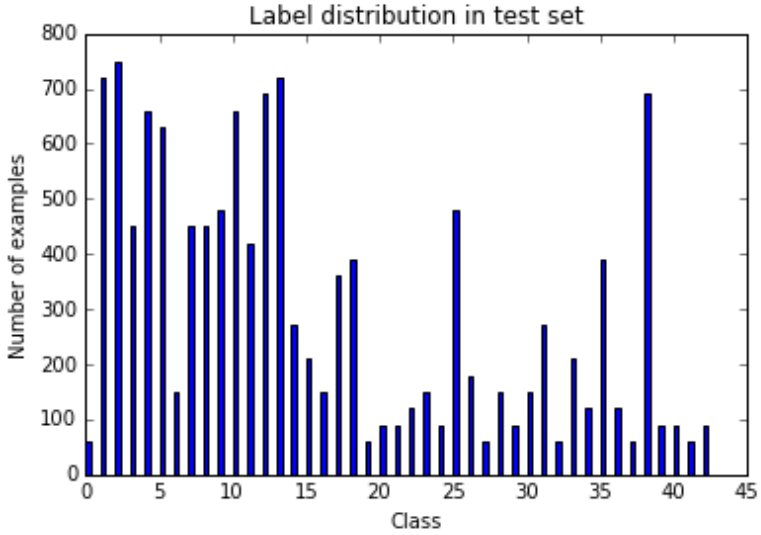
I used the pandas library to calculate summary statistics of the traffic signs data set:

- 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. Include an exploratory visualization of the dataset.


Label distribution in trainning set
Number of examples
Class
0
500
1000
1500
2000
2500
0
5
10
15
20
25
30
35
40
45


Label distribution in validation set
Number of examples
Class
0
50
100
150
200
250
0
5
10
15
20
25
30
35
40
45

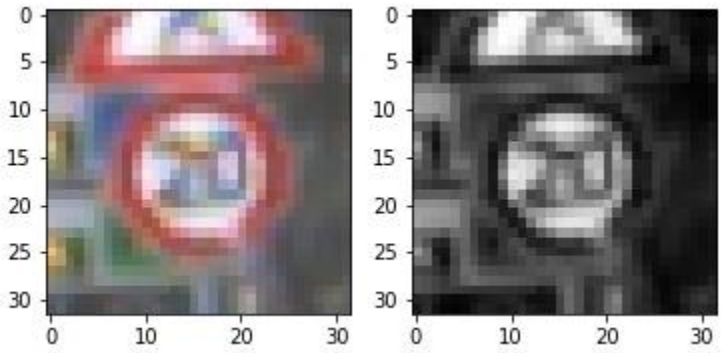

Label distribution in test set
Number of examples
Class
0
100
200
300
400
500
600
700
800
0
5
10
15
20
25
30
35
40
45

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. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

**Preprocessing 1**: Normalize the pixel value from [0, 255] to [-1, 1] via (pixel value / 255.0 * 2.0 - 1.0) in order to obtain 0 mean and equal variance across all color channels and to help the model coverage faster (with less training examples/ less epochs).

**Preprocessing 2**: Add grayscale channel by averaging 3 color channels, since certain traffic sign labels are independent from color, and grayscale may help the model learns faster.




**Other considerations**: I did not add noise to the image, since I have already added drop out layer in my multilayer perceptron. Adding noise or drop out may lead similar results.

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:

| Layer | Description |
|---|---|
| Input | 32x32x4 Red, Green, Blue, Grayscale channels |
| Convolution | Filter size: 5x5, 1x1 stride, same padding, outputs 28x28x32 |
| RELU | |
| Convolution | Filter size: 5x5, 1x1 stride, same padding, outputs 24x24x32 |
| RELU | |
| Max pooling | 2x2 stride, outputs 12x12x32 |
| Convolution | Filter size: 5x5, 1x1 stride, same padding, outputs 8x8x64 |
| RELU | |
| Max pooling | 2x2 stride, outputs 6x6x32 |
| Fully connected | 3 hidden layers of 512 each. Drop out with probability of 50% after each layer |
| Softmax | |

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.

20 epochs

Batch size of 128

Learning rate of 0.001

Adam Optimizer

4. Describe 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 0.987
- validation set accuracy of 0.951
- test set accuracy of 0.934

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?

  LeNet model that was implemented in the previous model, since the tasks are identical classifying numbers vs classifying traffic signs

- What were some problems with the initial architecture?

  The model size is too small since the classifying numbers problem is simpler.

- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

  I increase the size of the model by adding more convolution layers and increase multilayer perceptron layers. I observe some discrepancy between train accuracy and validation accuracy, and then add drop out.

I select epoch as 20 but not higher, since the model will become overfit and not perform well on validation set.

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

  Model size (number of convolution layers, number of hidden dimensions in multilayer perceptron's, drop out): The bigger the model, the more likely it overfit the training data, and underfit test data.

  Epoch: more epoch, more overfit to training data.

- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

  Drop out helps as regularize to prevent the model from overfit and over dependent on certain pixels.

If a well known architecture was chosen:

- What architecture was chosen? NA
- Why did you believe it would be relevant to the traffic sign application? NA
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well? NA

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:


All the images are relative simple since they may be very similar to the training example except the blue circle (as it is not a true circle since the picture is taken from an angle) and the “road narrows on the right” (as it have some noise level in the sign itself).

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:

| Image | Prediction |
|---|---|
| Road narrows on the right | Road narrows on the right |
| Roundabout mandatory | Roundabout mandatory |
| Pedestrians | Pedestrians |
| Double curve | Double curve |
| Bicycles crossing | Bicycles crossing |
| Right-of-way at the next intersection | Right-of-way at the next intersection |
| Beware of ice/snow | Beware of ice/snow |
| Speed limit (70km/h) | Turn right ahead |
| Speed limit (20km/h) | Speed limit (20km/h) |
| Children crossing | Children crossing |

The model was able to correctly guess 9 of the 10 traffic signs, which gives an accuracy of 90%. This compares favorably to the accuracy on the test set of 0.934

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)

The code for making predictions on my final model is located in the 17th cell of the Ipython notebook.

The model is highly curtained about the predictions with softmax probabilities very close to 1.0:

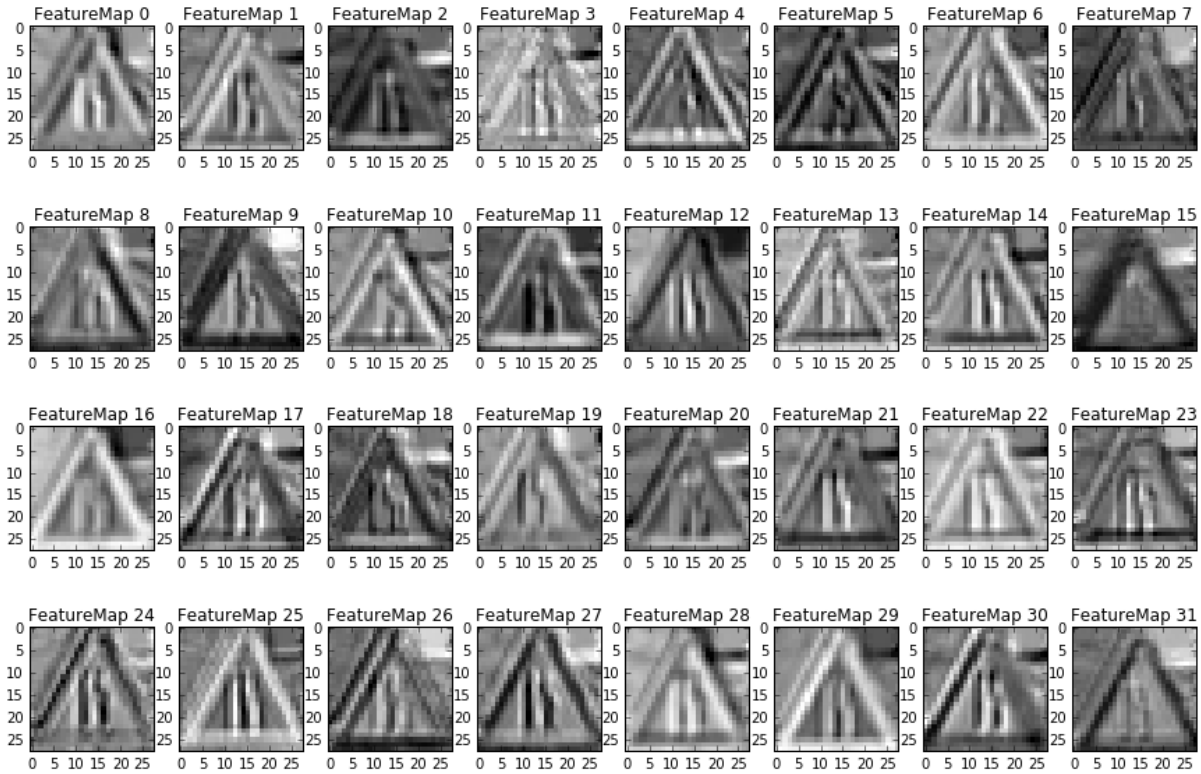
```
Label:      24 - Road narrows on the right
Prediction: 24 - Road narrows on the right
Top 5 predictions: [24 27 21  1 18]
Top 5 probabilities: [  9.99999642e-01   2.40124649e-07   7.38643067e-08
1.79673049e-13
   1.28243783e-13]

Label:      40 - Roundabout mandatory
Prediction: 40 - Roundabout mandatory
Top 5 predictions: [40 38 42  0  1]
Top 5 probabilities: [  1.00000000e+00   7.70347890e-37   1.55422844e-37
0.00000000e+00
   0.00000000e+00]

Label:      27 - Pedestrians
Prediction: 27 - Pedestrians
Top 5 predictions: [27 24 37 11  4]
Top 5 probabilities: [  9.45566237e-01   5.44333495e-02   1.57735528e-07
1.26465139e-07
   6.58837536e-08]

Label:      21 - Double curve
Prediction: 21 - Double curve
Top 5 predictions: [21 11 30  1 31]
Top 5 probabilities: [  1.00000000e+00   1.58976547e-08   4.32268776e-10
1.30409253e-10
   9.47639675e-11]

Label:      29 - Bicycles crossing
Prediction: 29 - Bicycles crossing
Top 5 predictions: [29 28 24 21 19]
Top 5 probabilities: [  9.99999762e-01   2.14709203e-07   2.81859369e-10
1.63469363e-10
   3.48181472e-13]

Label:      11 - Right-of-way at the next intersection
Prediction: 11 - Right-of-way at the next intersection
Top 5 predictions: [11 28 29 25  2]
Top 5 probabilities: [  1.00000000e+00   2.68028804e-14   2.45589072e-14
3.66172894e-15
   1.54433472e-15]

Label:      30 - Beware of ice/snow
Prediction: 30 - Beware of ice/snow
Top 5 predictions: [30 28 23  0  3]
Top 5 probabilities: [  9.56343472e-01   4.36564013e-02   1.37530535e-07
1.23332313e-08
   2.23597363e-09]

Label:      4 - Speed limit (70km/h)
```

```
Prediction: 33 - Turn right ahead
Top 5 predictions: [33 37 39 22  1]
Top 5 probabilities: [  9.75890338e-01   1.99277438e-02   4.06010076e-03
5.46856027e-05
   1.88839222e-05]

Label:      0 - Speed limit (20km/h)
Prediction: 0 - Speed limit (20km/h)
Top 5 predictions: [0 1 2 5 4]
Top 5 probabilities: [  9.99994040e-01   5.96101472e-06   1.88894465e-08
1.45276386e-08
   2.18492757e-09]

Label:      28 - Children crossing
Prediction: 28 - Children crossing
Top 5 predictions: [28 29 20 37 23]
Top 5 probabilities: [  1.00000000e+00   2.77638508e-13   1.45960690e-13
1.76152817e-14
   5.34718813e-15]
```

## (Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

####1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

The feature maps noticibly show the outer triangular and the inner vertical edges, i.e. the model has learnt to detect the triangular edges of the traffic sign and the 2 edges of inner vertical lines.


FeatureMap 0
FeatureMap 1
FeatureMap 2
FeatureMap 3
FeatureMap 4
FeatureMap 5
FeatureMap 6
FeatureMap 7
FeatureMap 8
FeatureMap 9
FeatureMap 10
FeatureMap 11
FeatureMap 12
FeatureMap 13
FeatureMap 14
FeatureMap 15
FeatureMap 16
FeatureMap 17
FeatureMap 18
FeatureMap 19
FeatureMap 20
FeatureMap 21
FeatureMap 22
FeatureMap 23
FeatureMap 24
FeatureMap 25
FeatureMap 26
FeatureMap 27
FeatureMap 28
FeatureMap 29
FeatureMap 30
FeatureMap 31