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.

###Writeup / README

####1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

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.

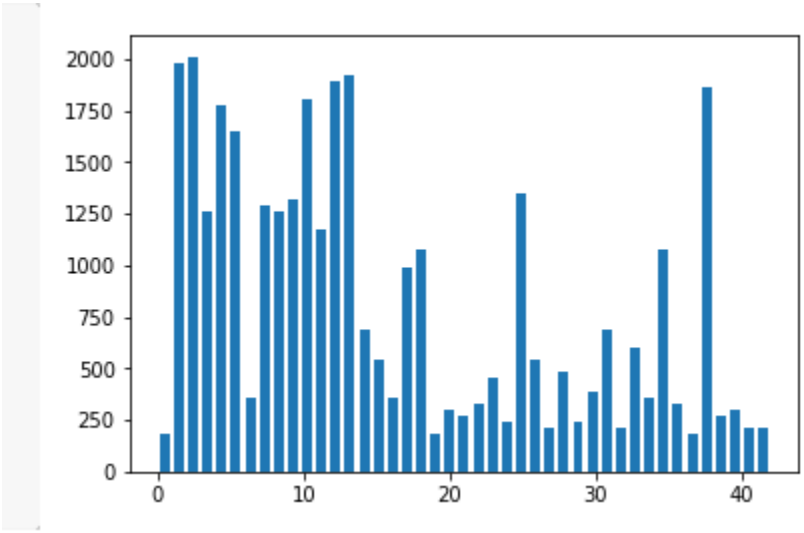
I used numpy methods to calculate summary statistics of the traffic signs data set:

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

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

Here is a random sample of five images – in color with the sign class ID assigned to them – they are 32 x 32 pixel images in RGB – and very low resolution/blurry.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is not equally distributed among the 43 sign classes – some sign classes have many more (2000) data points than others (200) by a factor of 10.



Classes of Sign Type (43 total)

###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.)

As a first step, I decided to convert the images to grayscale because grayscale makes it easier to detect edges and shapes – the contrast between dark and light is sharper.

I also normalized the images to (-1, 1) using the Normalized Image = (pixel-128)/128 formula.

I decided to generate additional data because some glasses only had ~200 data points – while others had ~2000 images.

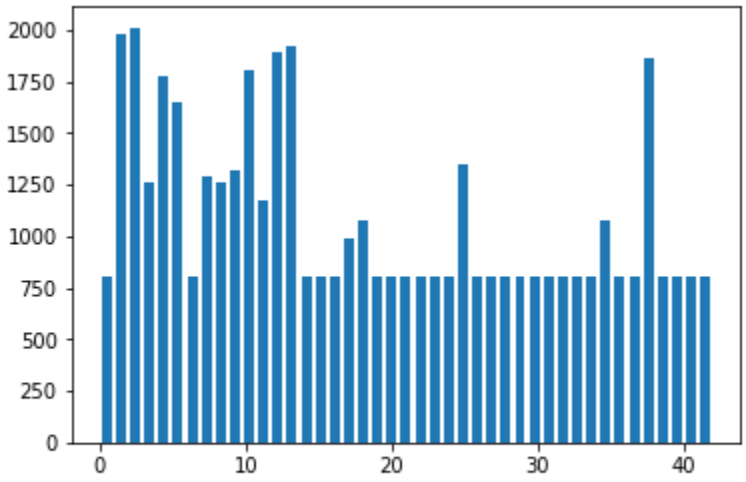
To add more data to the data set, I used

- Translation – shifting image by 2 pixels max
- Brightness – creating darker and lighter image versions
- Warping – changing viewing angle
- Scaling – changing overall image size

Here is an example of a traffic sign image before and after warping:

Here is an example of an original image and scaled image:

The difference between the original data set and the augmented data set is that now every class has at least 800 images – I believe this has a major impact on the ability of the model to recognize signs from all 43 classes.




```
X, y shapes: (46480, 32, 32, 1) (46480,)
```

####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.

I used a modified LeNet Architecture – same as the LeNet model in the lessons, but I added Dropout layers after the Fully connected Layers 3 and 4.

My final model consisted of the following layers:

Layer	**Description**
Input	32x32x1 Grayscale Image
Convolutional	Output 28x28x6 Valid Padding
Relu	Activation
Max Pooling	2x2 Stride Output 14x14x6
Convolutional	Output 10x10x16 Valid Padding
Relu	Activation
Max Pooling	2x2 Stride Output 5x5x16
Flatten	Output 400
Fully Connected	Output 120
Relu	Activation
Dropout	Keep probabilities
Fully Connected	Output 84
Relu	Activation
Dropout	Keep probabilities
Fully Connected	Output 43

####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.

To train the model, I used the AdamOptimizer with a learning rate of 0.0009. Since I was using a laptop without a GPU, I tried to keep the EPOCHs to 20 or less. My final solution used a batch size of 100 and 20 Epochs. I did not change the suggested mu of 0, or the suggested sigma of 0.1. For the added dropout layers, I used a keep probability of 0.5.

####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 started at 62.9% and increased to 97.5% over 20 Epochs.
- validation set accuracy of 94.5%
- test set accuracy of 92.8%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
    - First tried LeNet as used in the lab. Wanted to use it just to get the initial code to run – and to see how close it could get to 93% without changes.
- What were some problems with the initial architecture?
    - Would only get to 87/88% accuracy (using only Grayscale pre-processing)
- How was the architecture adjusted and why was it adjusted?
    - Added two dropout layers – after the first two fully connected layers. These layers seemed to be in "Earlier" versions of LeNet – and the model seemed to perform very well on the MNIST dataset – so I added them back in.
- Accuracy on the training set was close (within 3%) to accuracy on the validation set – so I don't think the model was underfitting or overfitting.
- Which parameters were tuned? How were they adjusted and why?
    - Adjusted learning rate from 0.0010 to 0.0005 – found best results with 0.0009.
- What are some of the important design choices and why were they chosen? I believe adding a dropout layer helped the model be successful by forcing it to develop redundant activations – and forced it to recognize sign shapes and symbols in multiple layers.

If a well known architecture was chosen:

- What architecture was chosen?
    - Modified LeNet with two dropout layers
- Why did you believe it would be relevant to the traffic sign application?
    - LeNet seemed to work well with character/shape recognition (MNIST) and traffic sign recognition seems to be a more complicated version of the same basic task – recognizing shapes and outlines of objects.
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
    - The model reached 97.5% accuracy after only 20 Epochs – I feel this is strong evidence of the model working well, with a limited number of Epochs. One trial used 40 Epochs – and that did not improve validation accuracy above 97.5%.

###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 five German traffic signs that I found on the web:

The first image was probably easy to classify – as the outside shape and inner triangle are unique to a yield sign.

The second image was probably slightly more difficult as the outside shape is shared with other signs – but there is good contrast between the inner and outer trapezoidal shape.

I expected the third image (60 KPH) to be difficult to classify as the shape is identical to other speed limit signs – only the number in the inner circle is different.

The fourth image also has a common round outer shape that could be confused with other signs – and I expected that the white arrows could be difficult to find in the blue background. (It was – as the model failed on this sign – giving it only a 2% rating).

The Fifth sign had a similar outside shape to other signs – but the model was able to pick out the pedestrian image as there is good contrast between the black image and the white background – and this contrast holds in a grayscale image.

####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).

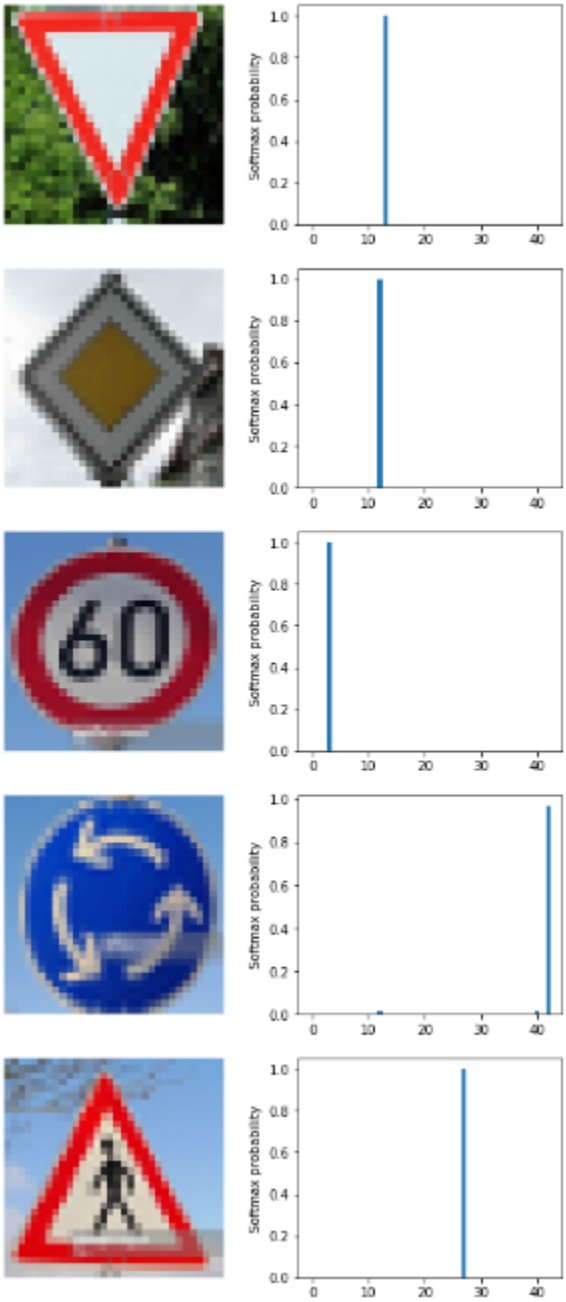
Here are the results of the prediction (80% correct)

Image	**Prediction**
Yield	Yield (100%)
Main Arterial Road	Main Arterial Road (100%)
60 KPH	60 KPH (100%)
Roundabout Ahead	No trucks in left lane (97%)
Pedestrian Crossing	Pedestrian Crossing (100%)

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This is lower than the accuracy on the test set. My roundabout image has a “Getty Images” tag on it – and this could have confused the model.

####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)

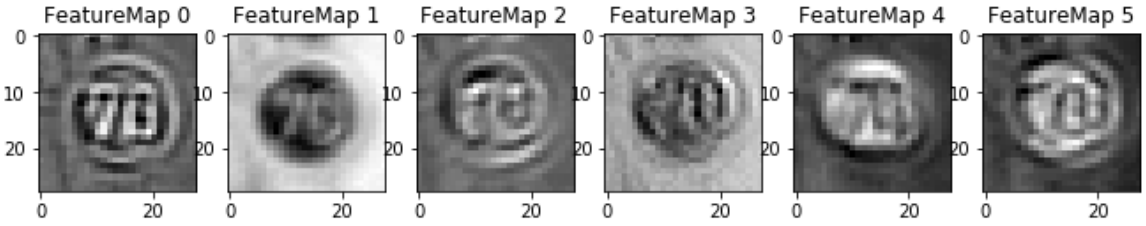
The model was 100% sure in the 4 cases with correct identification (Yield, Arterial Road, 60 KPH, and Pedestrian Crossing). On the Roundabout sign – it was 97% sure of the incorrect choice (No Trucks in Left Lane), 2% of the Correct guess (Roundabout) and 1% sure it was an Arterial road sign.



(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 example image is a 70 KPH sign – the feature map images are shown below:



I was surprised how clear the sign image is in the first 2 layers – then it seems to get distorted. I would expect that each layer would activate on different parts of the image – and-maybe-that's happening in Feature Map 2 and 3 (the edge of the numbers appears to be in negative then positive relief)…but it's hard to say what Maps 4 and 5 are activated by. Maybe the added Dropout layers (after layers 3 & 4) are distorting those Feature Maps?