Rubric Points

This will be an overview of my project located here for the ipython notebook or here for the html version.

Data Set Summary & Exploration

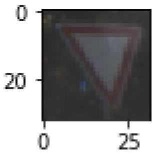
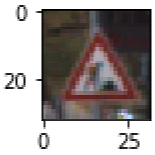
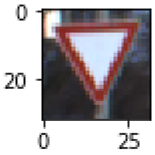
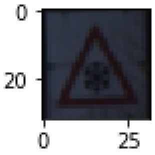
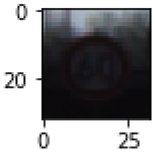
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

- The size of training set is 34,799
- The size of the validation set is 4,410
- The size of test set is 12,630
- The shape of a traffic sign image is 32x32x1
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. Five images were randomly selected and displayed with their labels:

Class	**Image**
13	
25	
13	
30	
3	

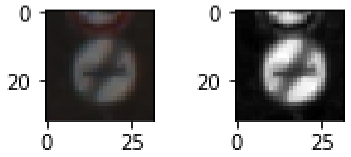
Design and Test a Model Architecture

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 I believe that having colors hurt the training algorithm by reducing its accuracy.

Here is an example of a traffic sign image before and after grayscaling.



As a last step, I normalized the image data in order to reduce the amount of errors the computer makes when doing millions of calculations.

The difference between the original data set and the augmented data set is the augmented dataset contained the same images as the original just grayscaled then normalized.

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	32x32x1 Gray image
Convolution 3x3	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	etc.
Fully connected	etc.
Softmax	etc.

Layer	Description
Input	32x32x1 Gray image

Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	Activation Function
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	Activation Function
Max pooling	2x2 stride, outputs 14x14x6
Flatten	Take a 5x5x16 and output a 400
Fully connected	Input 400, output 120
RELU	Activation Function
Fully connected	Input 120, output 84
RELU	Activation Function
Fully connected	Input 84, output 43
Softmax	Final Activation Function

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 a batch size of 120 with 3,000 Epochs. The learning rate was set at a constant rate of 0.001. The mu and sigma was set to 0 and 0.1 respectively. Finally the optimizer used was tf.train.AdamOptimizer().

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:

- validation set accuracy of 94.7%
- test set accuracy of 93%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?
    - I chose to use the LeNet architecture because it was very familiar to me and seemed to be a good fit for traffic signs.
- What were some problems with the initial architecture?
    - The initial architecture worked great. In fact the only issue was there the accuracy was extremely low and more layers had to be added to increase the accuracy.
- 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.
    - The LeNet architecture really wasn't modified at all. The only difference were adjusting some of the convolutional filter sizes to adjust for the increase in labels.
- Which parameters were tuned? How were they adjusted and why?
    - The Epochs were really the only thing that was tuned outside the standard LeNet architecture.
- 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?
    - I tried using a drop out and I got a worst accuracy so I didn't include it in the final build. I would assume, if implemented correctly, dropout would prevent overfitting as the model won't be able to rely on patterns as much and will have to be redundant.

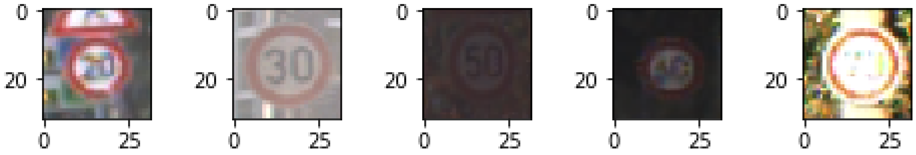
If a well known architecture was chosen:

- What architecture was chosen?
    - The LeNet Architecture.
- Why did you believe it would be relevant to the traffic sign application?
    - The LeNet Architecture is extremely good at classifying handwritten letters and number so it is really good at recognizing patterns with lines which is what a traffic sign is, a bunch of lines.
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
    - The validation set started out on 73% and after the first 1000 epochs it reached 93%. After another two thousand epochs the accuracy only jumped .2% meaning that it was taking longer to see any improvement.

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 might have some issues because there is part of another sign on top of it. Also the last two images might have problems because they are even hard for human eyes to see. The 5th image is extremely bright almost making the dark (black) part of the sign unreadable and the 4th image is extremely pixelated distorting the markings inside the sign.

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
Speed limit (20km/h)	Speed limit (20km/h)
Speed limit (30km/h)	Speed limit (30km/h)
Speed limit (50km/h)	Speed limit (50km/h)
Speed limit (60km/h)	No vehicles
Speed limit (70km/h)	Speed limit (70km/h)

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of 93% as to be expected. This did well considering the fact that these images were especially of low quality making them more of a challenge set (I even had a hard time reading the last two signs as a human!).

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 bottom half of the notebook under **Output Top 5 Softmax Probabilities For Each Image Found on the Web**.

The first and last image seemed to have extremely high softmax values meaning that the model was extremely confident in classifying them.

Softmax Value	Prediction
1269.0	Speed limit (20km/h)
571.54	Speed limit (30km/h)
101.66	Speed limit (50km/h)
136.66	No vehicles
1277.84	Speed limit (70km/h)

For the second image ...

(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?