Data Set Summary & Exploration

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

The code for this step is contained in the second code cell of the IPython notebook.
I used the len( ) and shape function from python to calculate summary statistics of the traffic signs data set:
• The size of training set is 34799
• 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 and identify where the code is in your code file.

The code for this step is contained in the third code cell of the IPython notebook.

Here is an exploratory visualization a random sample from the traning data with the correspondingly correct training label being 14 (identified as the stop sign label in the signnames.csv):

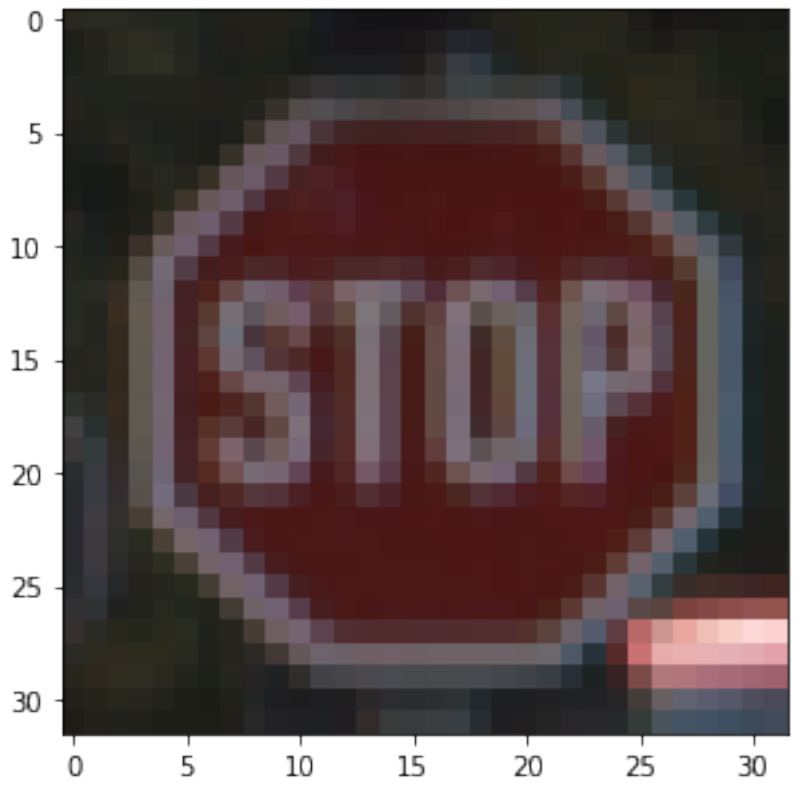

Figure: Data Visualization

Design and Test a Model Architecture

1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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.

The code for this step is contained in the fourth code cell of the IPython notebook.

As a first step, I convert the images to grayscale because grayscale images seem to ease edge detection from a neural network with a moderate size at the bottom layers. Although by doing so, we will lose some color information, it seems to facilitate training of the neural network based on the LeNet architecture for this particular problem of traffic sign classification.

Then, I perform the shifting and scaling of the greyscale image by first dividing each of the 32 by 32 pixel value it by 127.5 and then subtracted by 1 so as to make the range of each pixel being -1 to 1. It is well-known that such operations help ameliorate the problem of vanishing and exploding gradients in the gradient descent training.

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

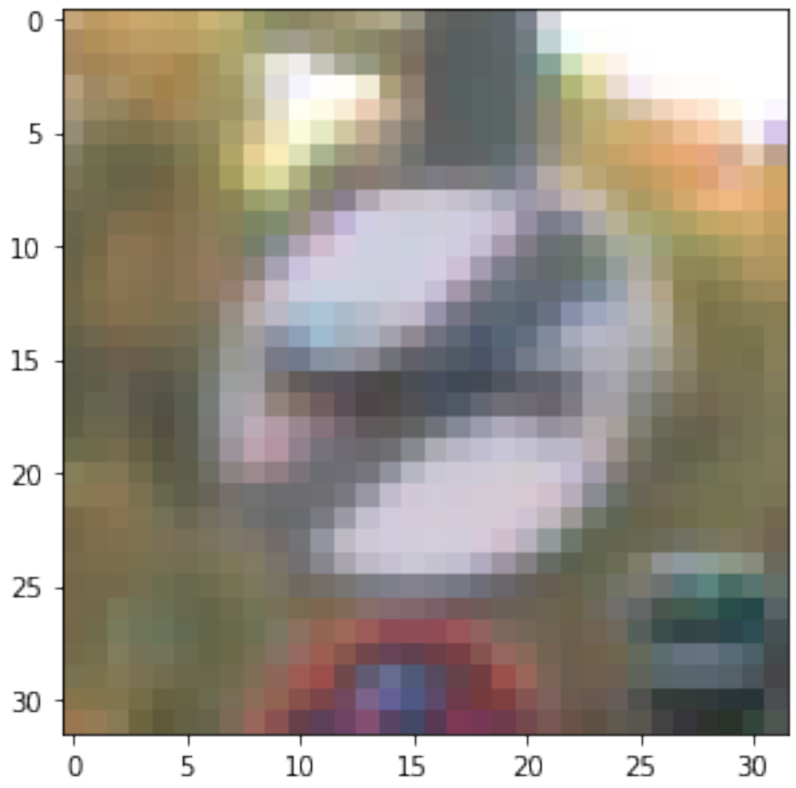

Figure: Sample Image before Preprocessing

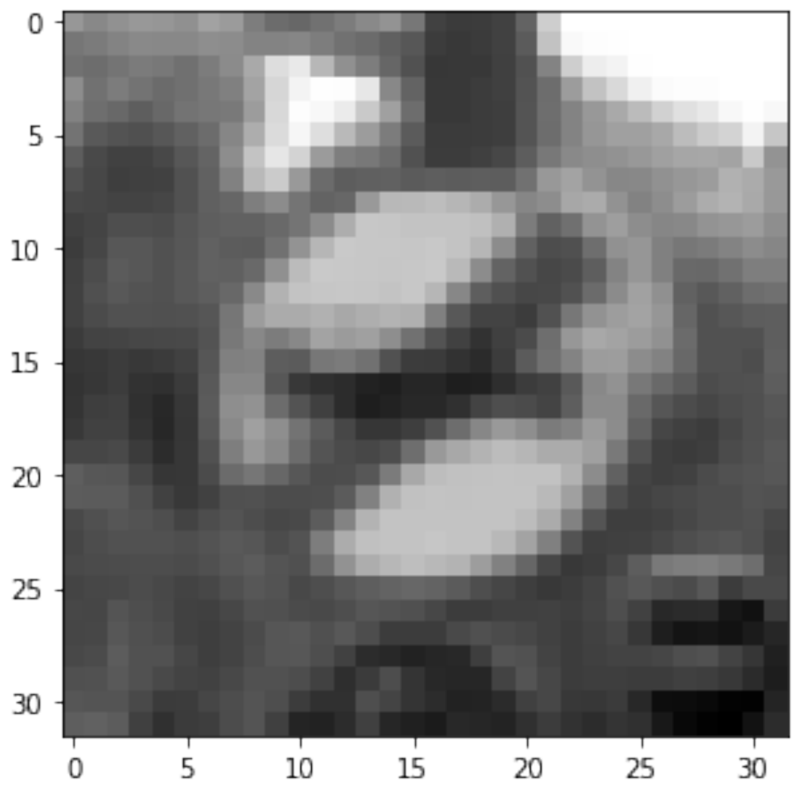

Figure: A Sample Image after Preprocessing

2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The code for splitting the data into training and validation sets is contained in the seventh to tenth code cells of the IPython notebook.

The picked data that comes with the project package has already been split into training, validation and testing sets. We only used the training and validation sets, and we do the final testing on a separate set of images download from the server (see below).

My final training set had 34799 number of images. My validation set and test set had 4410 and 5 number of images, respectively.

3. Describe, and identify where in your code, 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.

The code for my final model is located in the sixth cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 grascale preprocessed image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Outputs 400
Fully connected	Outputs 120
RELU	
Fully connected	Outputs 84
RELU	
Drop out	Keep probability of 0.5
Fully connected	Outputs 43

4. Describe how, and identify where in your code, 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 code for training the model is located in the seventh to tenth cells of the ipython notebook.

To train the model, I used the Adam optimizer to minimize the cross entropy cross function with batch size, number of epochs and learning rate being 128, 75, 0.001, respectively.

5. Describe the approach taken for finding a solution. 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.

The code for calculating the accuracy of the model is located in the tenth cell of the Ipython notebook.

My final model results were with the validation set accuracy of about 93-94%.

Our final model is based on appropriately refining and retuning the LeNet architecture (originally for handwritten digit classification for the MNIST data) to make it work well for the German traffic sign classification. The notable changes from the original LeNet architecture are as follows:

- Preprocessing the grayscale image with shifting and normalizing
- Expanding the dimension of the final logits from 10 (10 possible digits) to 43 (classes of the German traffic signs)
- Adding a dropout after the final RELU but before the final fully connected layer producing the final logits. We later optimize (by maximizing the validation accuracy in the training) the keep probability of the dropout to be about 0.5.
- The validation accuracy computed during training was boosted substantially after these amendments were made to the original LeNet architecture.

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:

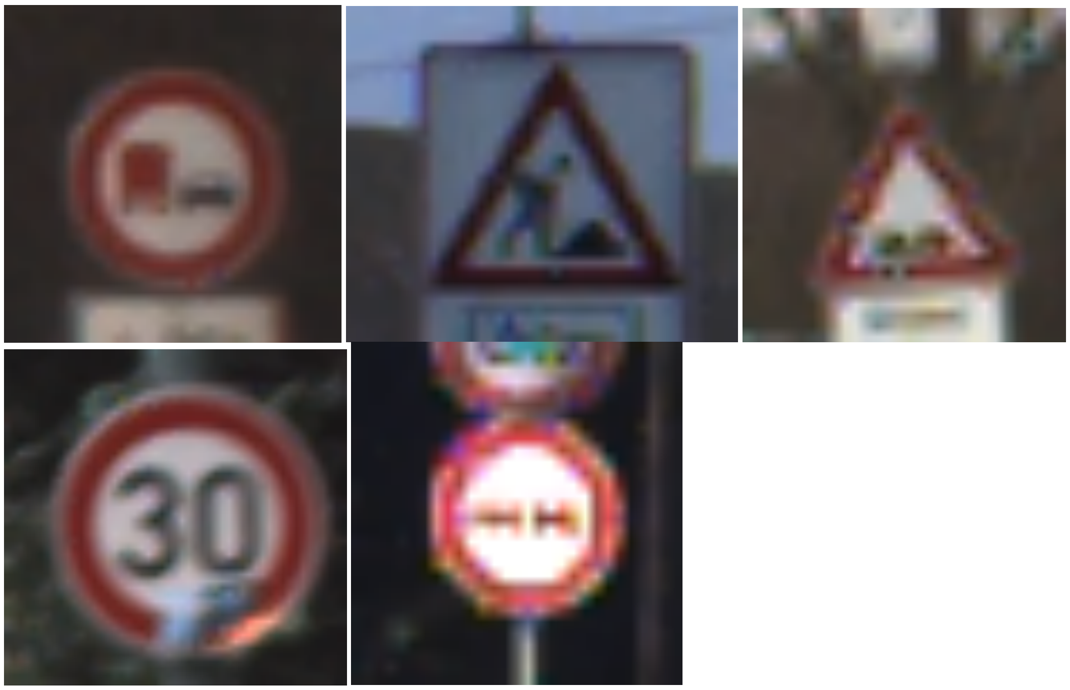

Figure: Five German Traffic Sign Images on the Web (1, 2 and 3 on the top row and 4 and 5 on the bottom row)

The third and fifth image should be the hardest to classify due to unclear representations of the bumpiness in the third and of the the two cars with different colors in the fifth image.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

The code for making predictions on my final model is located in the 12th cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
No passing for vehicles over 3.5 metric tons	No passing for vehicles over 3.5 metric tons
Road work	Road work
Bumpy road	Bumpy road
Speed limit (30km/h)	Speed limit (30km/h)
No passing	No passing

The model was able to correctly all the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the validation set during training of 93-94% as mentioned earlier.

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 and identify where in your code softmax probabilities were outputted. Provide the top 5 softmax probabilities for each image along with the sign type of each probability.

The code for evaluating the confidence level of the predictions on my final model is located in the 12th-14th cell of the Ipython notebook.

As can be seen from the top softmax probabilities that my final model makes the prediction of each of the five testing images form the web with overwhelming confidence, wherein the next confusable hypothesis has an extremely small probability < 5*10^(-10), which suggests that the final model is able to generalize very well to unseen data.

The top five soft max probabilities of all five images are as follows (see the 13th and 14th code cells in the Ipython notebook).

Probability	Prediction
1.00000000e+00	No passing for vehicles over 3.5 metric tons
4.09746532e-25	Vehicles over 3.5 metric tons prohibited
6.87358340e-36	End of no passing by vehicles over 3.5 metric tons
0.00000000e+00	Speed limit (20km/h)
0.00000000e+00	Speed limit (30km/h)

Figure: Top five soft max probabilities for predicting the traffic sign for No passing for vehicles over 3.5 metric tons

Probability	Prediction
1.00000000e+00	Road work
9.91191463e-26	Turn right ahead
5.17616321e-26	Stop
2.61124456e-31	Dangerous curve to the right
2.21581320e-31	Bumpy road

Figure: Top five soft max probabilities for predicting the traffic sign for Road work

Probability	Prediction
1.00000000e+00	Bumpy road
4.90310736e-10	Dangerous curve to the right
1.74539064e-11	Bicycles crossing
7.02026284e-12	Traffic signals
7.26596463e-13	Go straight or right

Figure: Top five soft max probabilities for predicting the traffic sign for Bumpy road

Probability	Prediction
1.00000000e+00	Speed limit (30km/h)
0.00000000e+00	Speed limit (20km/h)
0.00000000e+00	Speed limit (50km/h)
0.00000000e+00	Speed limit (60km/h)
0.00000000e+00	Speed limit (70km/h)

Figure: Top five soft max probabilities for predicting the traffic sign for Speed limit (30km/h)

Probability	Prediction
1.00000000e+00	No passing
7.91261280e-16	No passing for vehicles over 3.5 metric tons
8.06010670e-20	Vehicles over 3.5 metric tons prohibited
1.01400657e-28	Dangerous curve to the left
5.42730334e-34	Speed limit (60km/h)

Figure : Top five soft max probabilities for predicting the traffic sign for No passing