1) **Dataset Summary**: I used Numpy to get a summary of the data set.
   Number of original training examples = 34799
   Number of training examples = 104397 → this is after augmentation of the data
   Number of validation examples = 4410
   Number of testing examples = 12630
   Image data shape = (32, 32, 3)
   Number of classes = 43 → this represents the total different types of traffic sings included in the data set.

2) **Exploratory Visualization**: I used the matplot library to plot a subset of the training data. 50 images were chosen at random and were plotted. As a second step, I used the Seaborn library to get a histogram of all the data sets (training/validation/test). From the histogram it seems that data is unevenly distributed across the classes. However, the distribution across the training/validation/test sets is the same.

3) **Preprocessing:** I started off the preprocessing with a simple min-max normalization to bring the data set into the range of 0.1 to 0.9. However, after training the network with this I found that the validation accuracy was lower than 0.93. This indicated a lack of data to train the network. Therefore, I augmented the data by applying a rotation to the image matrix, first by 10 degree and then by -10 degrees. This step logically makes sense since a human eye can identify say a stop sign even if it is titled/rotated. Therefore, our network should be able to do the same. I also tried to reduce the image from RGB to grey scale, however this gave a worse performance than with color. Here, I think for the below mentioned leNet model, a gray scaled image loses too much information for proper training of the weights, hence I stuck with the RGB images.

4) **Model Architecture:** I started off with the classic LeNet model, i.e. 2 convolution layers, where each layer comprised off convolution-relu-max pooling, followed by 3 fully connected layers. This network however did not reach the desired validation accuracy of > 0.93. As a next step I optimized the number of nodes in the fully connected layer and landed with the following final configuration (nodes marked in bold were tuned w.r.t. to the classical LeNet model):

Layer	Input	Output
Convolution -1	32x32x**3**	32x32x6
Max pooling-1	28x28x6	14x14x16
Convolution -2	14x14x16	10x10x16
Max pooling-2	10x10x16	5x5x16
Fully connected -1	400	**300**
Fully connected -2	**300**	**200**
Fully connected -3	**200**	**43**

To avoid over fitting, I introduced **drop out layers** after every layer of the LeNet network. Further, I tuned the optimal value of keep probability by training the model with different values from 0.5 to 1 and found that **0.9** maximized the accuracy in validation set for the above model

5) **Model training:** I trained the model using Adam optimizer. I did not get a chance to try out different optimizers, such a Stochastic Gradient Descent. Next, I tuned the batch size, epoch and learning rate hyper parameters. I found that increasing the batch size too much, while decreasing the epoch gave a worse performance than increasing the epochs and decreasing the batch size. After performing a joint optimization of all 3 parameters, I found that batch size == 128, epoch = 30, learning rate = 0.001, gave validation accuracy of > 0.93 while keeping the training time of the network under control. Also data augmentation done in the preprocessing stage dramatically improve the training results.

6) Solution Approach:

The problem at hand here requires that the classifier should be (1) Quick to train (2) attain high test accuracy and (3) Generalize to new images. Since, we are dealing with images, convolution neural network seems to be the right choice for the job. This is because convolution neural networks are an excellent tool for extracting complex features from the images layer by layer i.e. each layer extracts a higher level of feature. Next, since we need to obtain a high validation/test accuracy, the network would probably need to have multiple layers i.e. be a deep network. However, we can't arbitrarily keep extending the number of layers as well since the training time would exponentially increase. Therefore, we need to have the right balance on the number of layers and their dimensions we choose. LeNet is a model which meets our requirements here. The model is quite powerful in terms of the features it can extract while being trainable in a reasonable time.

I followed an iterative approach to find the optimal solution. First I started off with a base LeNet model and found its validation accuracy. Next, I improved this accuracy by training the model better by data augmentation. Then, I optimized the hyper parameters, epoch, batch size and learning rate in a joint manner. As a final step, to ensure model is not over fitting the data, I tuned the keep probability of the drop out layer. As a final result, the validation accuracy was **0.95** while the test accuracy was **0.937.** This indicates that the model was slightly over fitted.

7) **Images found on the web:**

The images found on the web were typically of much higher resolution than what the LeNet model can process. Due to this, I have resized the images to be of dimensions 32x32. This would then lead to loss of information and therefore could have an impact on the performance of the classifier and lead to miss-classifications.
On the other hand side, most of these images have a sharp and clearly defined edge which distinguishes the sign from the background. Moreover, the images don't seem to have too much noise as well. Both of these qualities should then lead to a better prediction.

After the preprocessing, I ran the prediction model to output the class and found an accuracy of **0.8.** The image which shows a mismatch was of 70Kmph which the network predicted to be 20Kmph. Both of these images share lot of characteristics, therefore further training of the model is required to resolve these conflicts as well. The softmax shows that the original class,

70Kmph, was in the top 5 classes, which shows that further training could improve the results. The test set had an accuracy of 0.93 while the new images from web had an accuracy of 0.8. I believe the new images performed as expected but we should not directly compare the two accuracies since the test set comprised of more than 12,000 images while the new images from the web were just 5.

Looking at the Softmax probabilities of all the outputs, it is evident that the network gave correct prediction with a very high confidence i.e. the probability of the correct class is almost 1 and all the other classes have a probability of less than 10^-6. On the other hand side, the first sample, which fails to be recognized correctly has a much more uniform distribution of the probabilities. This then seems consistent with the image itself (a speed sign) since there are multiple much sign with very similar characteristics leading to a closer distribution of probabilities.