Traffic Sign Recognition

Writeup

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

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 and pandas to calculated the summary statistics of the dataset and used matplotlib to display the images

* The size of training set is - 34799

* The size of the validation set is - 4410

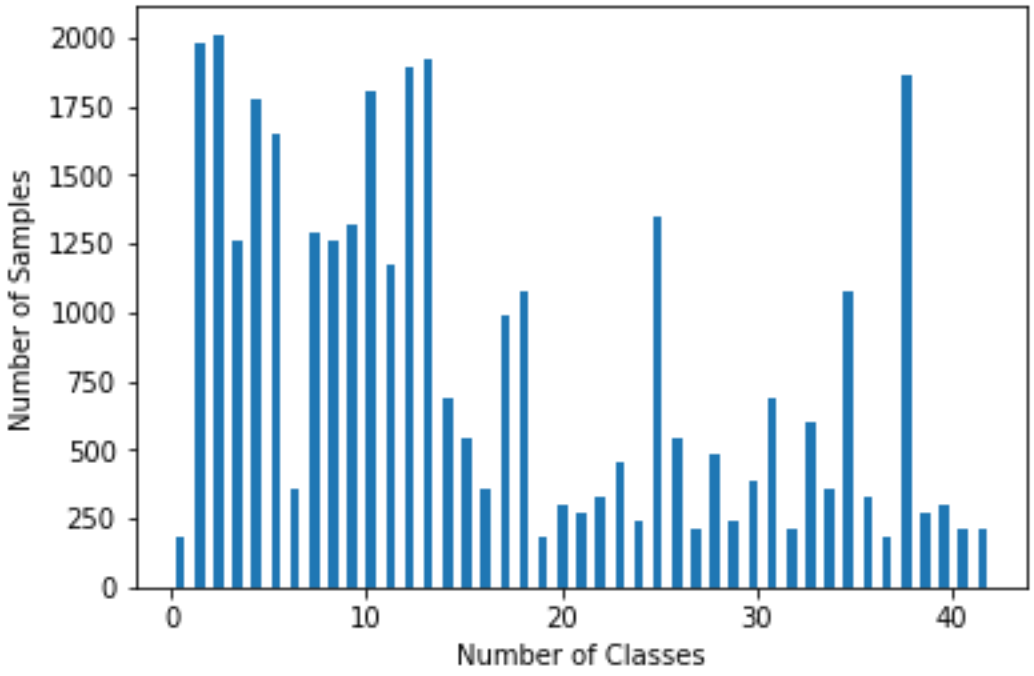
* The size of test set is - 12630

* The shape of a traffic sign image is - 32x32x3

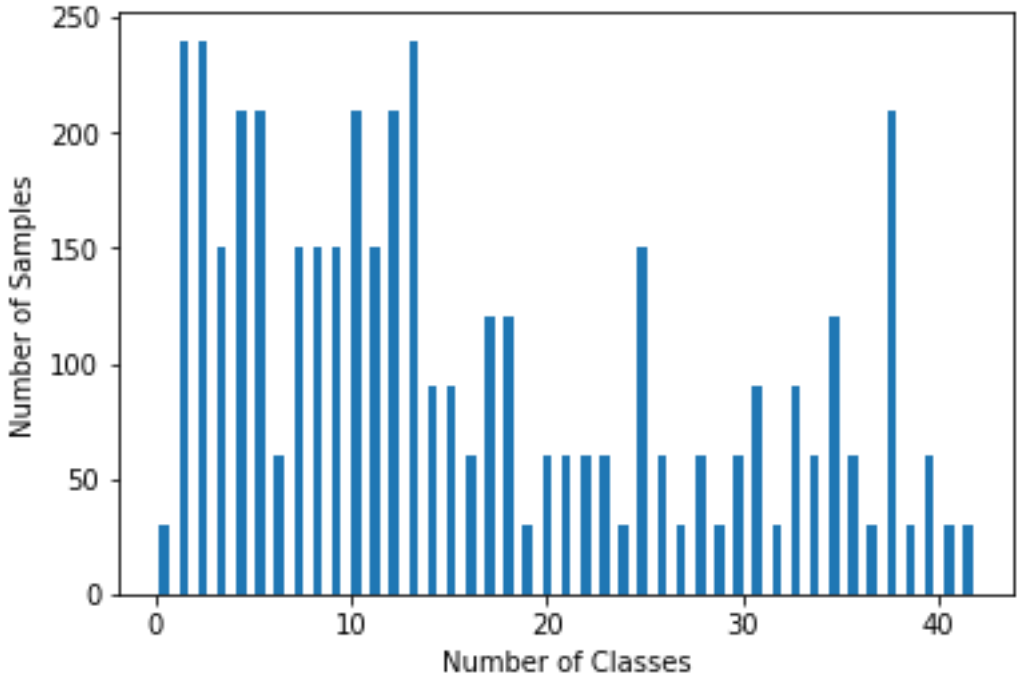
* The number of unique classes/labels in the data set is - 43

2. Include an exploratory visualization of the dataset.

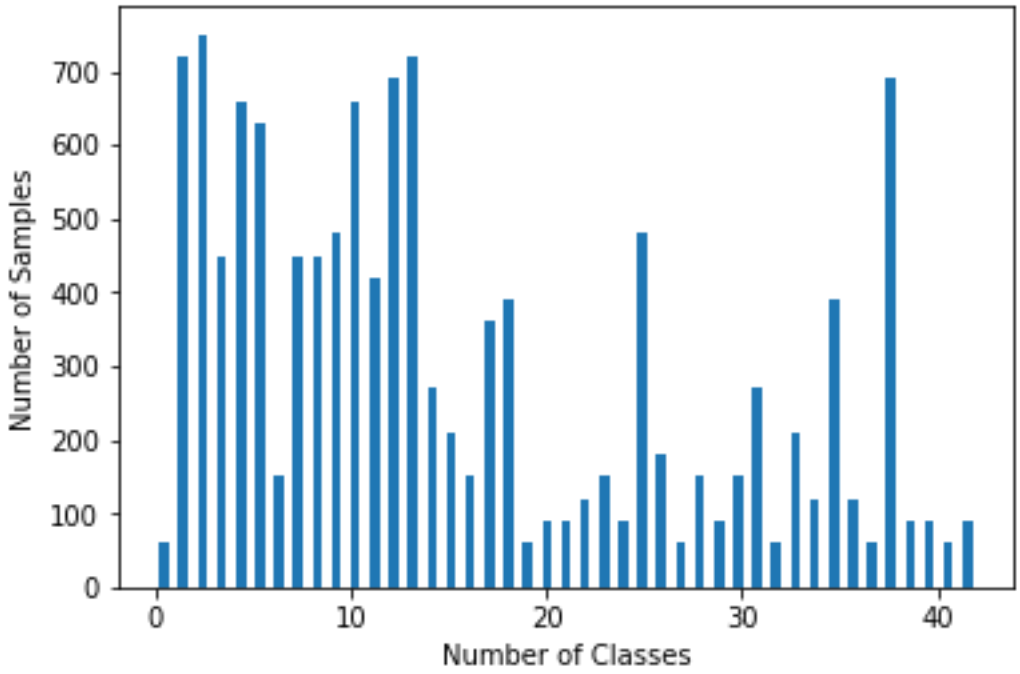
Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed. X axis shows the number of classes and Y axis shows the number of samples per class. Below three bar charts shows the data distributions in train, validation and test set.



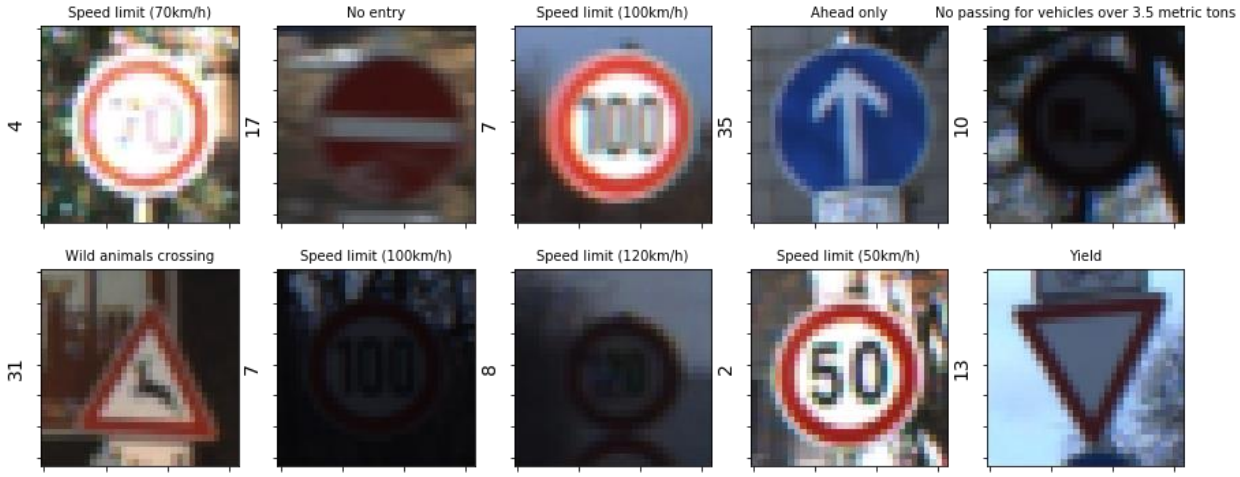
Train Set



Valid Set



Test Set



Training Data Set – Random Images with their respective classes on Y axis and Label Names on X axis

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.)

Techniques used for preprocessing the images

1. Gray scaling

   The images in all three(test, validation and train) sets were converted from 3 channels to one channels i.e converting color images to gray scale. This was done by taking the mean along three channels in the image. This will contribute for reduction of training time and help the neural network to classify an image in the training set faster. Reducing the channels also mean less computation which makes the whole program lighter and can be run on a decent CPU.

2. Normalizing

   Normalization was done by taking the image data and subtracting with 128 and dividing it with 128. Normalization will further contribute for faster backpropagation and gradient descent as the mean approaches zero and the variance is equally distributed among the image data.

3. Data Augmentation

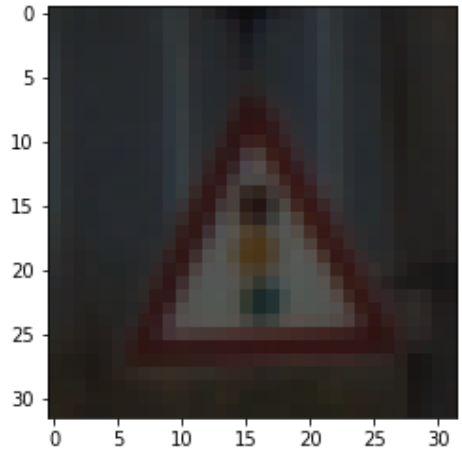
   Data Augmentation performed on the original set. This was done by adding random variations into the images. I used rotation, warping, random brightness, scaling and transformation as the image processing techniques from opencv library to create the extra data. This data was then mixed with the original data and shuffled with respective labels. Once the data is shuffled, 20 percent of the new data was distributed into validation set.
   ‘

* The size of new training set is - 55678

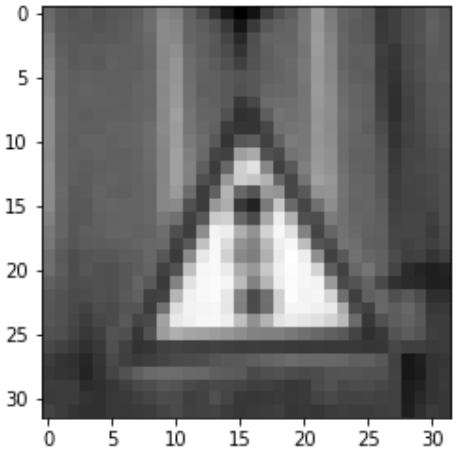
* The size of the new validation set is - 13920

* The shape of a traffic sign image is - 32x32x1

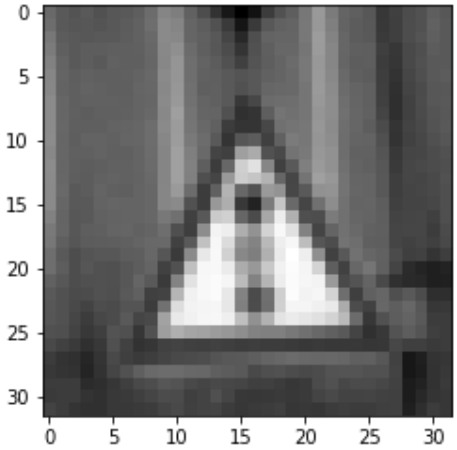
* The number of unique classes/labels in the new data set is - 43



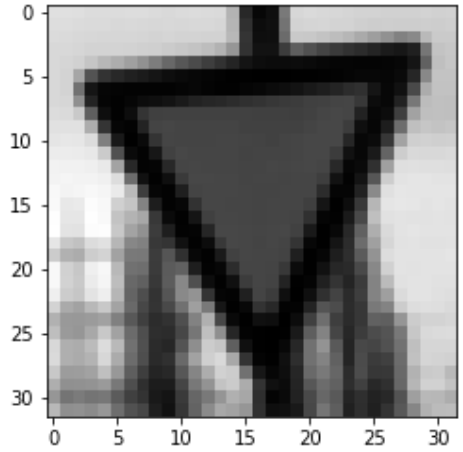
Train Set RGB Image



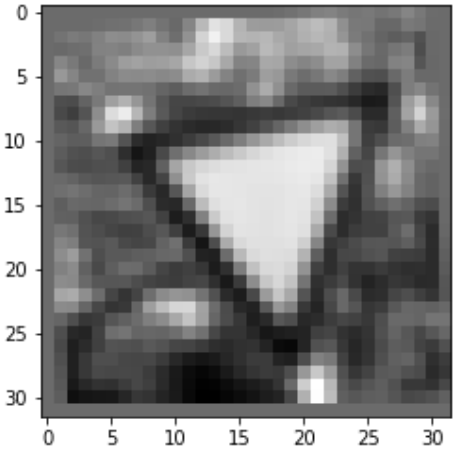
Gray Image



Normalized Image



Before Augmentation

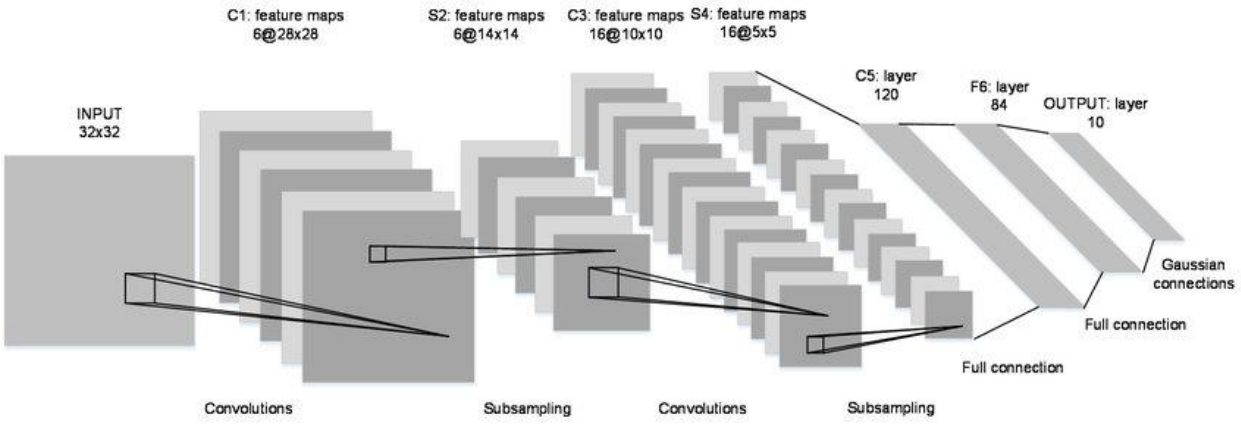


Augmentation(Notice the gap in the left corner and rotations in the image)

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:

I used LeNet – 5 Architecture for my neural network model. Below is the diagram of the architecture and list of layers and their corresponding inputs and outputs.



LeNet-5 Diagram

1. Input Layer – 32x32x1
2. Convolution (5x5) – Input (32x32x1), Output(28x28x6)
3. ReLU
4. Max Pooling - Input (28x28x6), Output(14x14x6)
5. Convolution (5x5) – Input (14x14x6), Output(10x10x16)
6. ReLU
7. Max Pooling - Input (10x10x16), Output(5x5x16)
8. Flatten - Input (5x5x16), Output (400)
9. Fully Connected – Input (400), Output (120)
10. ReLu
11. Dropout – 50%
12. Fully Connected - Input (120), Output (84)
13. Relu
14. Dropout – 50%
15. Fully Connected - Input (84), 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 LeNet – 5 architecture. I followed the previous LeNet lectures and followed the approach as instructed in the lessons to create the neural network. I added two dropout layers in the LeNet-5 model to achieve better validation accuracy. I randomly experimented with the hyperparameters and observed the below changes.

- Decreasing batch size resulted in better accuracy, however the epochs were very slow
- Increasing the epochs after a certain level (around 20) didn't make the validation accuracy any better.
- Increasing the learning rate with constant batch size made the validation accuracy to fluctuate up and down by 0.5 to 1%.
- Decreasing the learning rate beyond 0.001 didn't show any effect on the validation accuracy.

Final Hyperparameters used in the model

Batch Size – 128

Epochs – 20

Learning Rate – 0.001

Dropout Rate – 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.

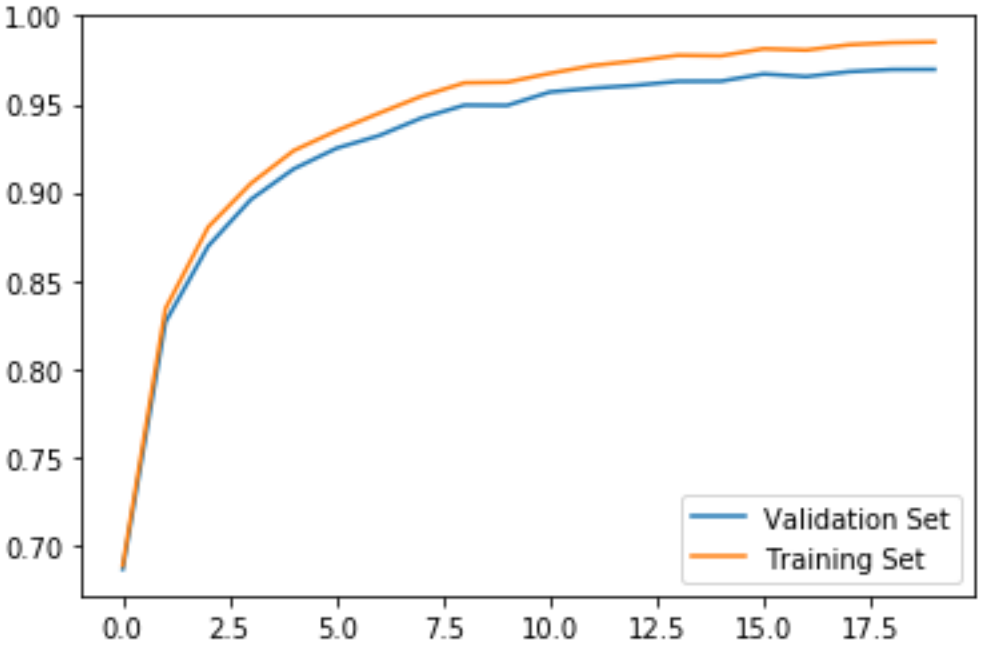
I used iterative approach to find the solution for this problem and achieve 97% of validation accuracy.

My final model results were:

* training set accuracy of 98.5

* validation set accuracy of 97.0

* test set accuracy of 93.9



Training vs Validation Accuracy

If an iterative approach was chosen:

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

I used the sample code provided by Udacity lectures in the LeNet-5 module and made changes as suggested in the lectures by changing the input channels from 3 to 1 and final layer output from 10 to 43. This was chosen to experiment and see how the model will work with minimal modifications.

* What were some problems with the initial architecture

-There were not enough validation samples, although I tried to shuffle the data with initial 34K training samples, it did not increase any validation accuracy.

- I tried just the grayscaling sample for training the network and the accuracy was bad, so I later added normalization. With normalization I was achieving around 93% but the graph showed there is still lot of variance between training and validation.

- Without including small variations in the data, the network had a hard time to identify some of the images which had less brightness, slightly rotated or warped in the sample downloaded from internet.

* 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 architecture was adjusted by adding dropout layers, which increased the fastness of the training and achieved good validation accuracy which was consistent in terms of variance with respect to test set. There was overfitting initially with higher learning rate or higher batch size. Data augmentation did help in achieving 97% accuracy, however by tuning in the learning rate and batch size further the training and validation graph became much smoother and couldn't find any random jumps (overfitting) in the validation graph.

Also, with tuning hyperparameters and data augmentation I was able to achieve more than 90% of training and validation accuracy within first 10 epochs.

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

Check above answer.

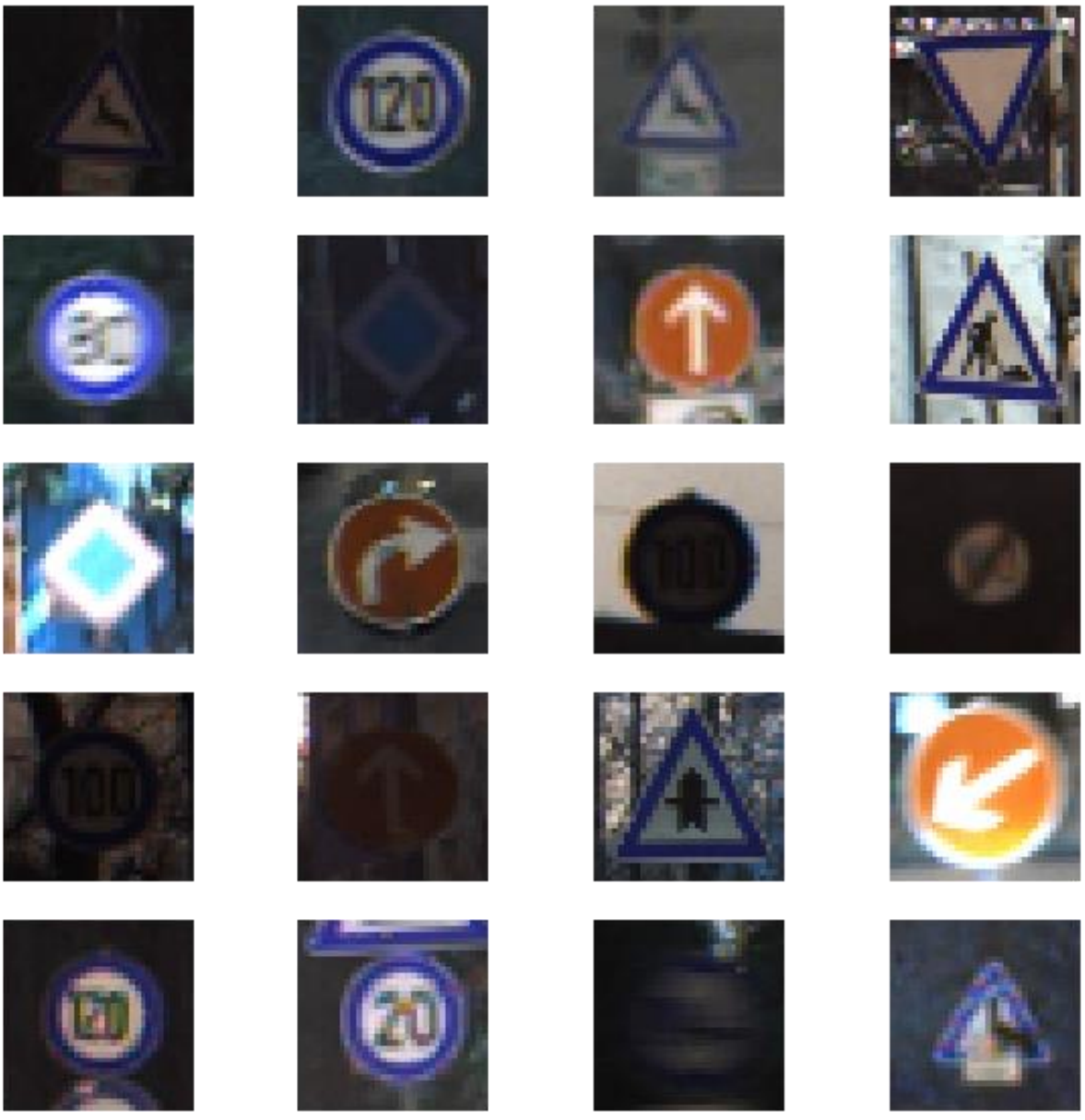
* **What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?**

I am not entirely sure why a convolution layer work well with this problem and why not others as I haven't experimented the model with a different architecture. Drop layer helped to remove the overfitting issue that I saw initially when the model didn't include any dropout layers.

TEST A MODEL ON NEW IMAGES

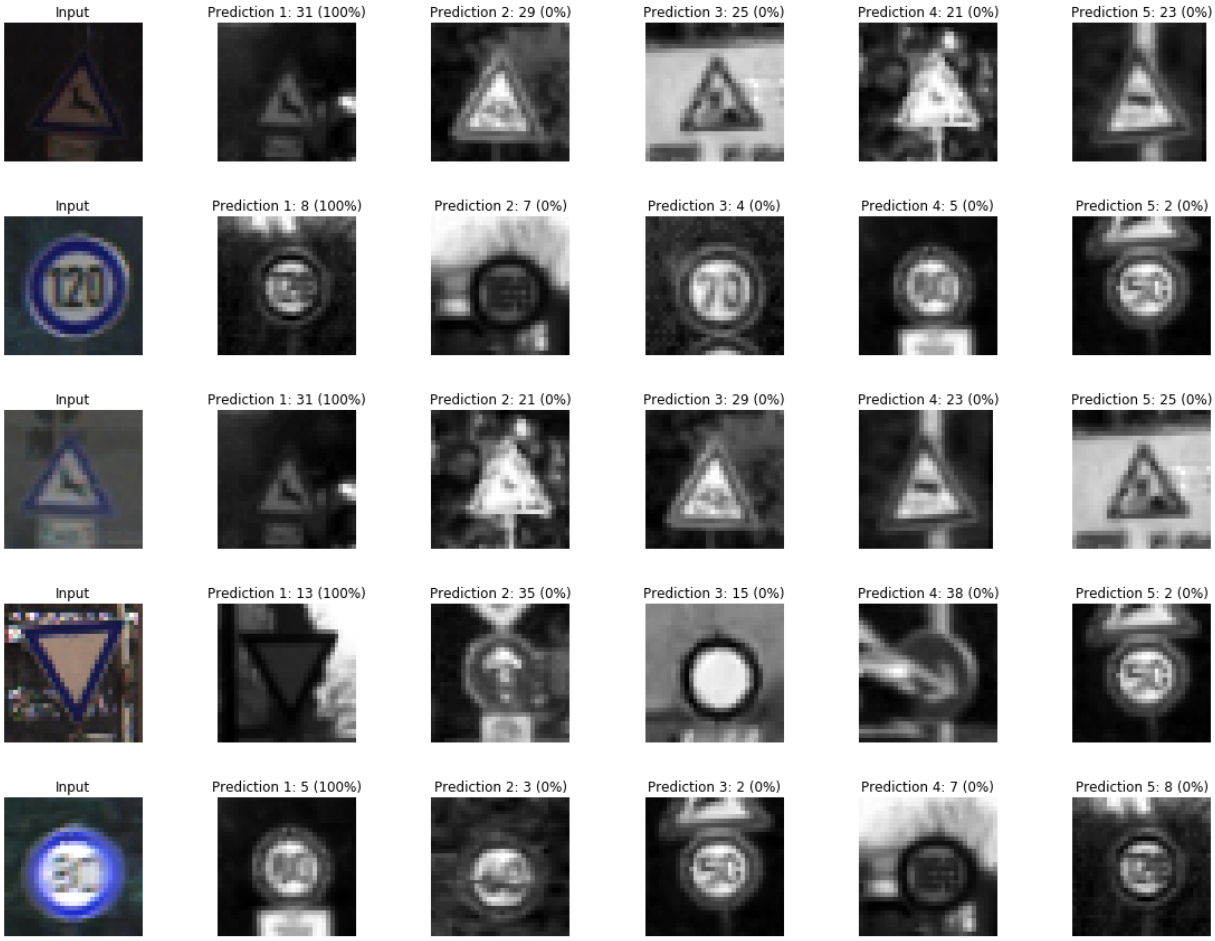
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

I did not assign any labels to the signs. I used 20 random signs from the set downloaded from internet. I did not find any difficulty in classifying the initial 5 images.



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:



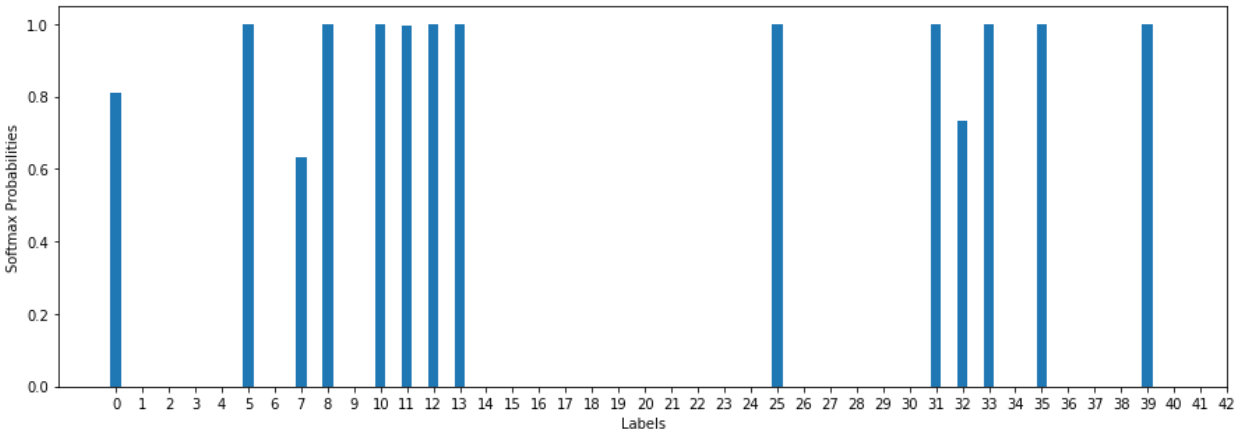
German Traffic Signs- Final Predictions

The model was able to predict all 5 samples with 100% accuracy.

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 above image (German Traffic Signs-Final Predictions) will show the top 5 predictions of the test images downloaded from internet. The top 5 were selected randomly from 20 test images. The softmax probabilities for all 20 images were shown in the below histogram. X-axis is the corresponding label for the prediction and Y axis shows the SoftMax prediction in percentage. Except for label 0,7 and label 32, the rest are 100 percent predicted correctly.

Averaged all the predictions for 20 samples, test accuracy was 94% which is similar to the result that I got during the initial testing phase using test images from test.p file.



Softmax Probabilities for 20 samples Images downloaded from web