Traffic Sign Recognition

Writeup

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the 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.

Files Submitted

The project submission includes all required files as shown below:

1. Ipython notebook with code (file name: Traffic_Sign_Classifier_WX.ipynb)
2. HTML output of the code (file name: Traffic_Sign_Classifier_WX.html)
3. A writeup report (You are reading it! file name: writeup_WX.md)
4. Other supporting files (Appendix.ipynb/.html, utilities_tf1.py and images in the test_images folder)

Dataset Exploration

1. A basic summary of the data set.

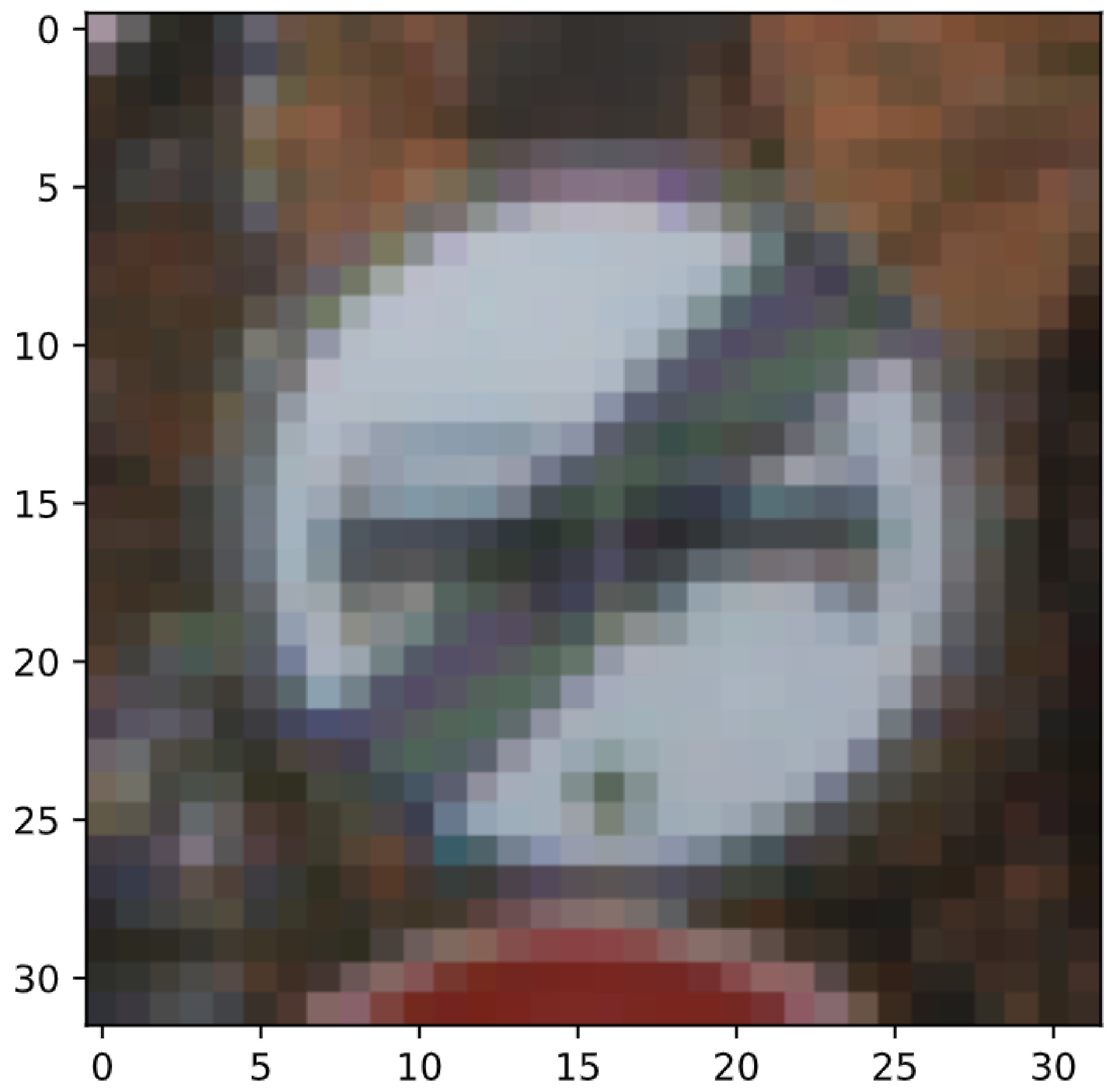
I used the pandas library to calculate summary statistics of the traffic signs data set:

- 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 (32, 32, 3)
- The number of unique classes/labels in the data set is 43

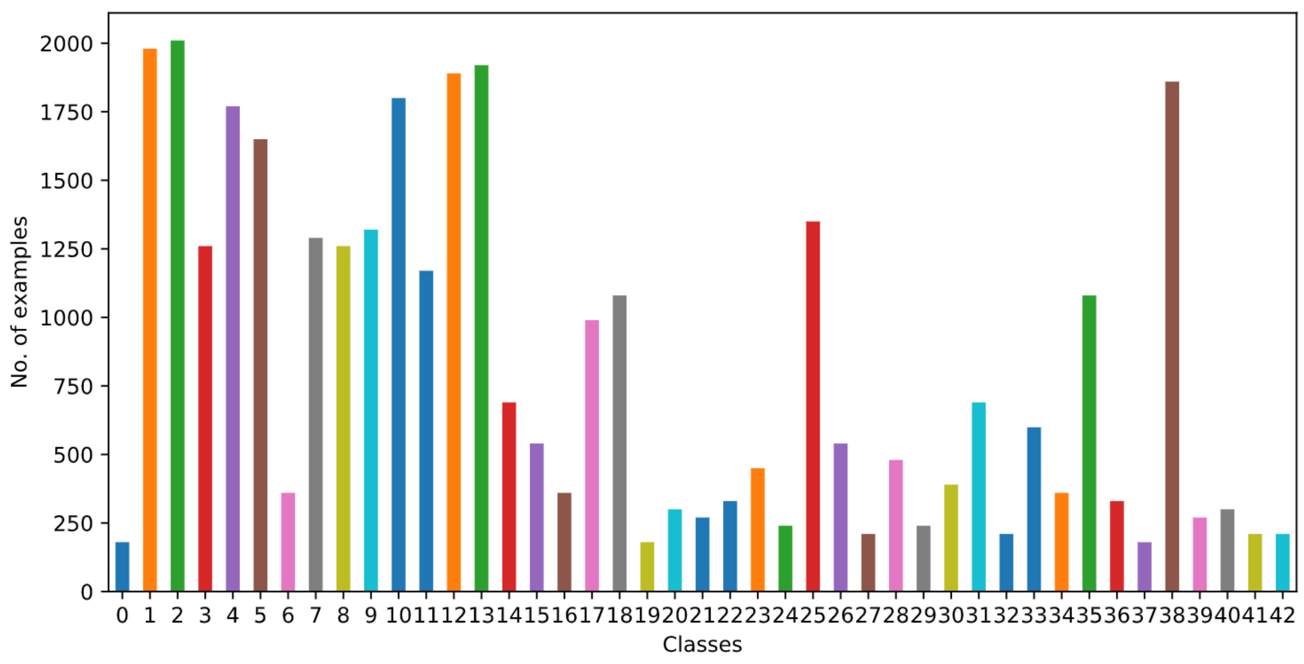
2. Include an exploratory visualization of the dataset.

Here is an plotting of the traffic sign images, which is under `class 41: "End of no passing"`:

```
y= 41
```



Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed in terms of different classes:



Design and Test a Model Architecture

0. re-arranging the Training/Test datasets

I have re-arranged the Training/Test sets through the following method:

> - Taken out around 75% of the Test set data then appended it to the Training set.
> - Left the rest 25% of the Test set data for evaluation purpose.
> - the orignal validataion set has not been used in this project.

I have created another note in the project folder named "Appendix.ipynb/.html" to explain why I did this, in summary:

> - After the initial Training/Testing of the CNN model, I realised that the given Traing/Validation/Testing sets might have data mis-matching issue.
> - In order to verify my guessing, I have defined a new subset called `training-validation set`. This new subset has the same distribution as the training set, but it is not used for training the neural network. The result shows that the training set and validation/test sets are very likely from diffierent distributions, which indicates that there might be a data mis-maching issue.
> - Therefore, I have decided to add a portion of the Test set data to the Training set which is to enable the model to learn features of the Test set.
> - The new trained CNN model shows a good performance, which was proved by predicting the on-line sample images correctly with an accuracy of 100%. (Please refer to a later Section "Test a Model on New Images" for more details)

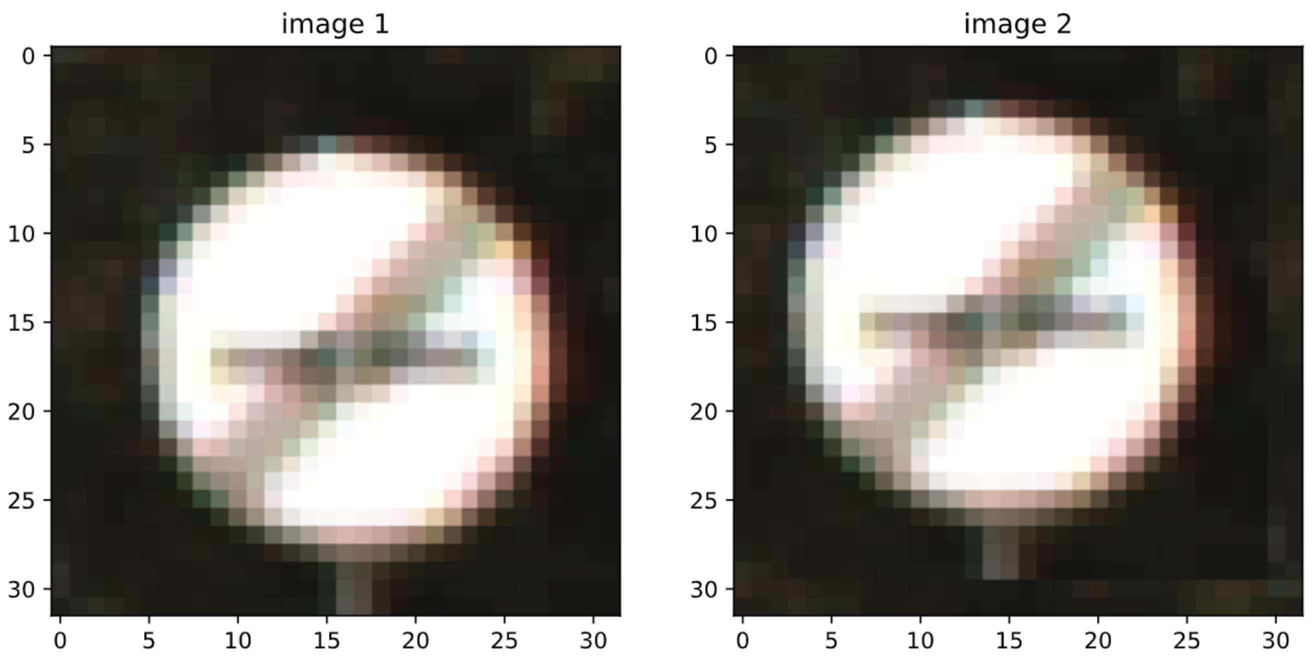
1. Data pre-processing

I decided to generate additional data to improve the model accuracy. To add more data to the the data set, I used the following techniques which was referenced from the paper by Yann LeCun:

1. Filtered out images/labels with low sample sizes, e.g., number of examples less than 1000.

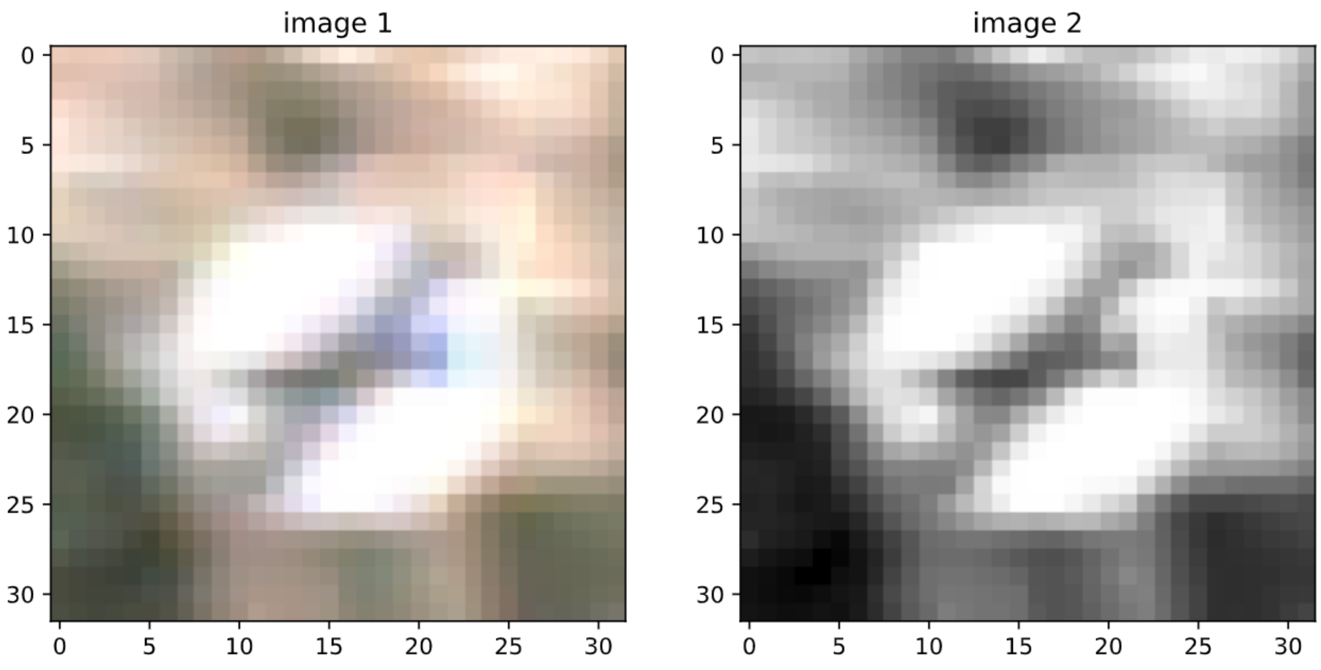
2. Generate fake data by adjusting the position of the images with [-2, 2] pixels.
3. Generate fake data by adjusting the ratio of the images with factors of [0.9, 1.1]

Here is an example of an original image and an augmented image (adjusted image position by (2, 2) pixels):



Then, I decided to convert the images to grayscale because in the later stage when I tried to predict on-line traffic sign images, I observied that their resolutions/brightnesses is quite different from the trainging data set.

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



As a last step, I normalized the image data to enable a more effective training of the project.

2. Model Architecture

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale 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	
Fully connected	outputs n_classes=43
Softmax	

3. Model Training.

To train the model, I used the settings/ hyperparamters as shown below:

Items	Value
Number of epochs	20
Batch size	128
Learning rate	0.001
Optimizer	Adam

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

My final model results were:

- Training set accuracy of 98.6%
- Test set accuracy of 93.6%

(note: the orignal validation set was not included as explained in Section "0. re-arranging the Training/Test datasets")

Below illustrates the step-by-step approach which leads to achieving the abovementioned final model result:

1. Initial Run: with the orignal training/validation/test datasets (pre-processed with grayscale and normalization) and LeNet model with intial parameters introduced in the course, I have trained the CNN model with the following results:

Dataset	**Accuracy (%)**
Training set	97.2%
Validation set	75.8%
Test set	76.2%

2. From the results of the inital run it seems that the model has an "overfitting" issue, since the accuracy difference between the training set and validation/test sets is as large as 21.4%. Therefore, I have tried a few improvement measures as shown below:

Improvement measures	**Validation/test set Accuracy (%)**
number of epochs increated to 20	70%
drop-out one/two layers (keep = 0.75)	72%
Generate more fake images to the training set by approx. 65%	75%

3. The results above show that the Accuracy of the given Validation/test set is not imporved, then I realised that there might be a data mis-matching issue.

4. In order to verify my guessing, I have defined a new subset called `training-validation set`. This new subset has the same distribution as the training set, but it is not used for training the neural network. The result belwo shows that the original training set and validation/test sets are very likely from diffierent distributions, which indicates that there might be a data mis-maching issue.

Dataset	**Accuracy (%)**
Training set	97.7%
Training-Validation set	92.0%
Validation set (from previous run)	75.8%
Test set	73.4%

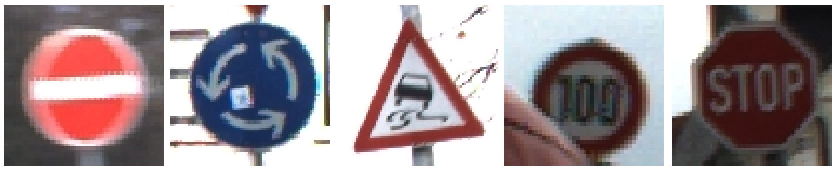
5. Therefore, I have decided to add a portion of the Test set data (around 75%) to the Training set which is to enable the model to learn features of the Test set.

6. The new trained CNN model shows a good performance which achieved 93.6% accuracy for the testing set. It was also proved by predicting the on-line sample images correctly with an accuracy of 100%. (Please refer to a later Section "Test a Model on New Images" for more details)

Test a Model on New Images

1. Choose five German traffic signs found on the web

Here are five German traffic signs that I found on the web:



All of the five images look quite different from the training dataset since their resolutions and brightnesses are much better than those images from the training dataset. However, since all images need to be converted to grayscle, such difference could be tackled by the model.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set.

Here are the results of the prediction:

Image	Prediction
Speed limit (100km/h)	Speed limit (100km/h)
Stop sign	Stop sign
No Entry	No Entry
Slippery Road	Slippery Road
Roundabout	Roundabout

The model was able to correctly guess 5 out of the 5 traffic signs, which gives an accuracy of 100%.

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.

For four out of the five images (Images 1, 2, 3 and 5), the model predictions are quite certain about their actual classes with the probability 1 or close to 1. For image 4 which is a `speed limit (100km/h)`, the predicted probability is 0.854 which is lower than those of the other four images. This might be due to the reason that all speed limit signs look quite similar to each other, which is also supported by the other softmax probalbilities for the same image with most of them identified are speed limit signs as well.

the softmax probabilities for eacch predictions are shown below:

Image 1:

Probability	Predicted class	Prediction
1.0	17	No entry
${7.02}\times 10^{-13}$	9	No passing
${2.73}\times 10^{-13}$	14	Stop
${1.82}\times 10^{-17}$	41	End of no passing
${1.80}\times 10^{-17}$	12	Priority road

Image 2:

Probability	P redicted class	Prediction
0.999999	40	Roundabout
${5.96}\times 10^{-8}$	12	Priority road
${1.71}\times 10^{-8}$	34	Turn left ahead
${8.49}\times 10^{-9}$	14	Stop
${3.05}\times 10^{-11}$	37	Go straight or left

Image 3:

Probability	Predicted class	Prediction
0.992	23	Slippery road
${5.61}\times 10^{-4}$	11	Right-of-way at the next intersection
${1.95}\times 10^{-4}$	20	Dangerous curve to the right
${5.92}\times 10^{-5}$	19	Dangerous curve to the left
${2.44}\times 10^{-6}$	25	Road work

Image 4:

Probability	Predicted class	Prediction
0.854	7	Speed limit (100km/h)
${9.40}\times 10^{-2}$	5	Speed limit (80km/h)
${1.84}\times 10^{-2}$	15	No vehicles
${1.55}\times 10^{-3}$	8	Speed limit (120km/h)
${4.51}\times 10^{-4}$	2	Speed limit (50km/h)

Image 5:

Probability	Predicted class	Prediction
1.0	14	Stop
${2.17}\times 10^{-10}$	33	Turn right ahead
${1.28}\times 10^{-11}$	4	Speed limit (70km/h)
${5.44}\times 10^{-12}$	39	Keep left
${4.23}\times 10^{-3}$	15	No vehicles