Eduardo Ribeiro de Campos - Udacity / SDCND - May./2019.

Self-Driving Car Engineer Nanodegree

Project3: Traffic Sign Recognition

The goals / steps of this project are the following according to the rubric points (https://review.udacity.com/#!/rubrics/481/view)

1. **Dataset Exploration**
   1.1 Dataset summary.
   1.2 Exploratory Visualization.
2. **Design and Test a Model Architecture**
   2.1 Preprocessing.
   2.2 Model Architecture.
   2.3 Model Training.
   2.4 Solution Approach.
3. **Test a Model on New Images**
   3.1 Acquiring New Images.
   3.2 Performance on New Images.
   3.3 Model Certainty - Softmax Probabilities.

Here is a link to code file (./Advanced_Lane_Finding.ipynb)

---

1 - Load the data set.

1.1 - Dataset sumary.

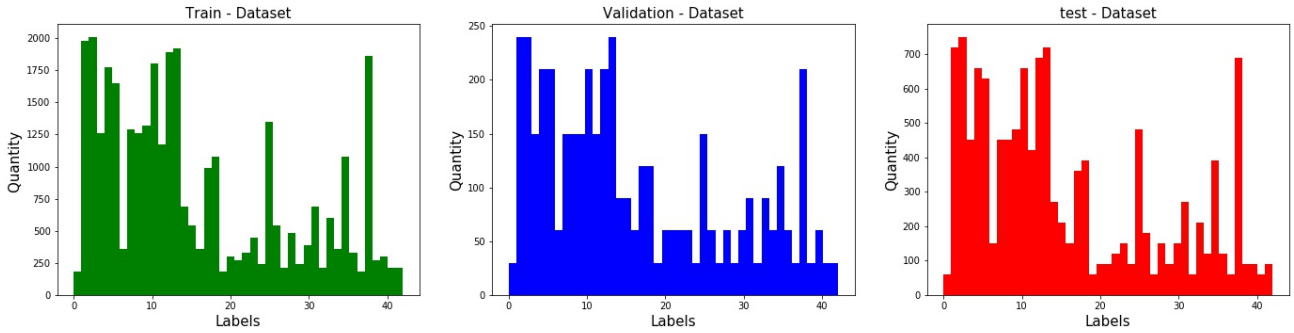
The dataset was analysed using dictionary function from python. The table below shows the summary.

Features	quantity
training	34799
testing	12630
Validation	4410

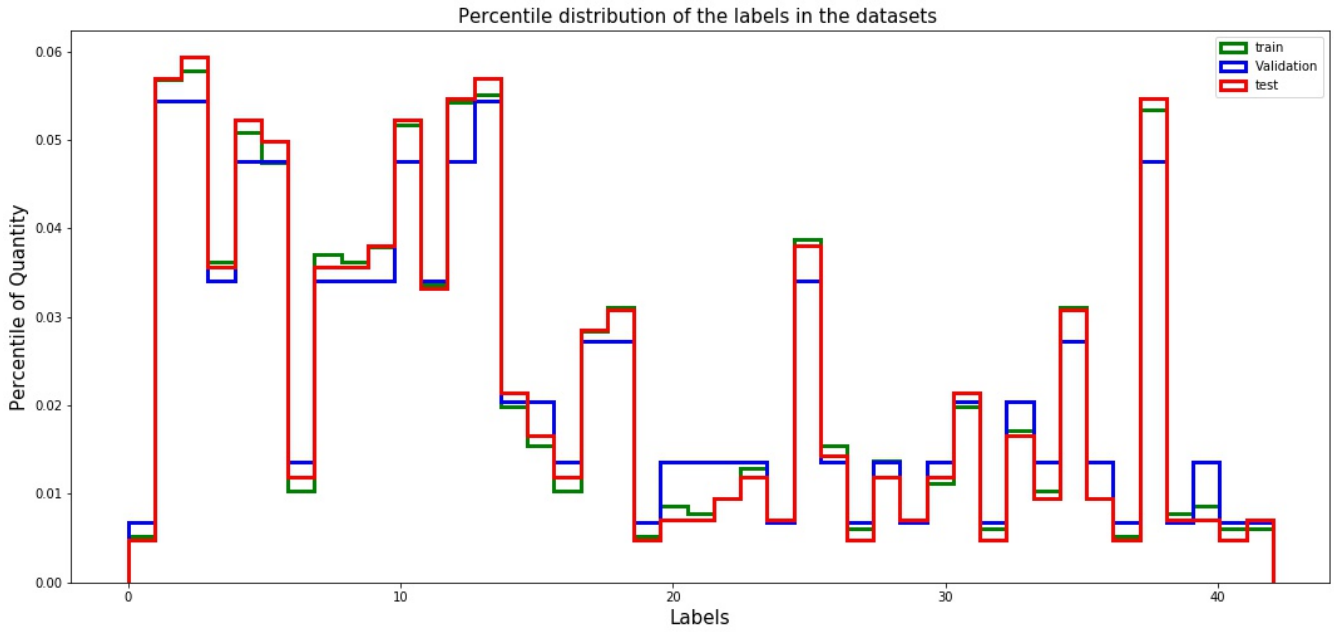
The image shape for the features is: 32x32x3

The number of unique classes (Labels) is: 43

The next histograms show the distribution for the labels for each set. (Train,Validation and Test).

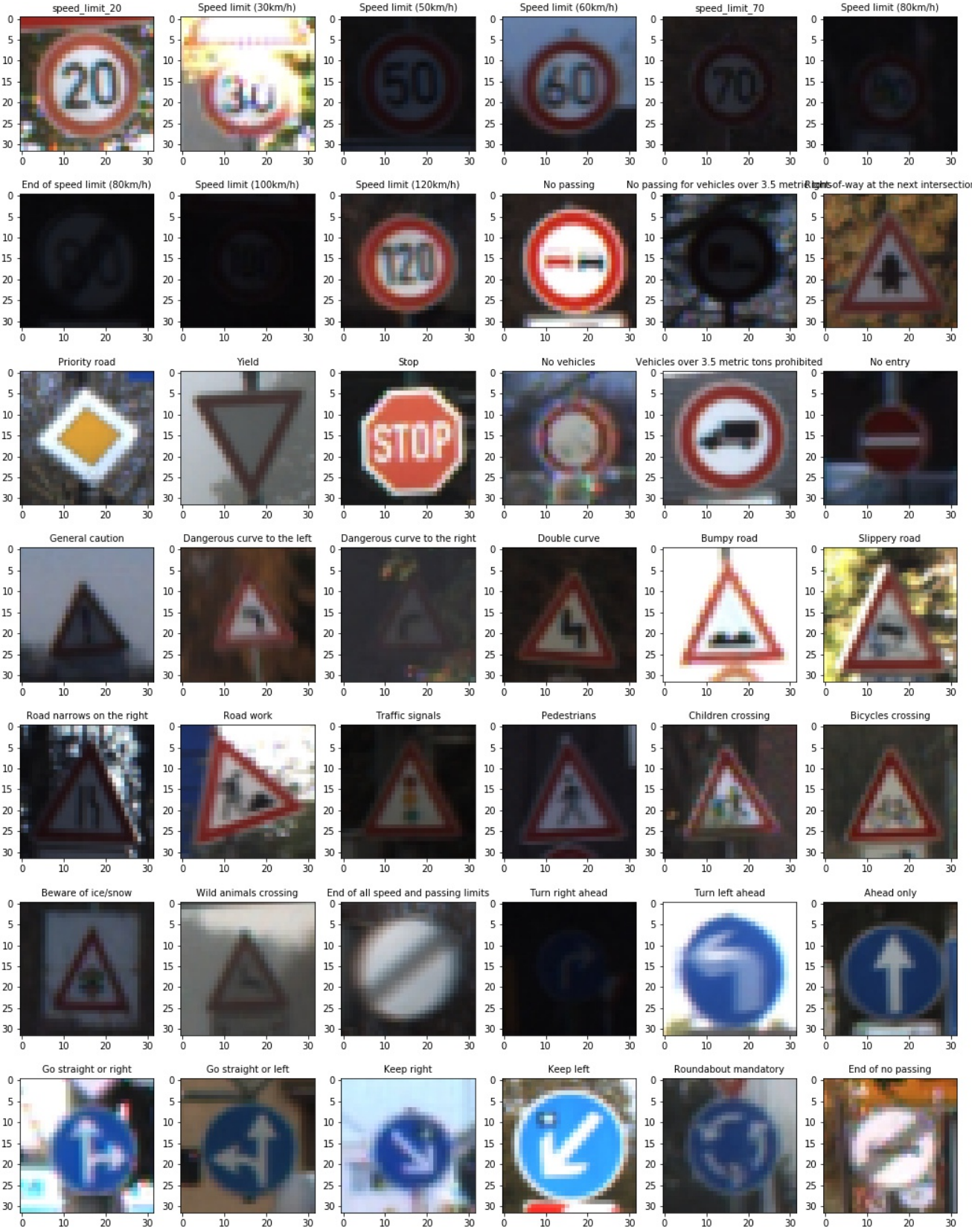


Regarding the label distribution for the datasets, it is possible identify that we have the approximately the same distributions from the percentiles of the labels comparing with the total for the each Dataset. The next histogram show the distribution:



1.2 - Exploratory Visualization.

The Next picture shows the 43 Labels used as traffic signs dataset and the image of them.



From the histogram, below it is possible see 7 differents images from the same Label. The Labels 2, 1 and 13 are the labels with more number of examples and the Labels 0, 19 and 37 are the labels with less number of examples.

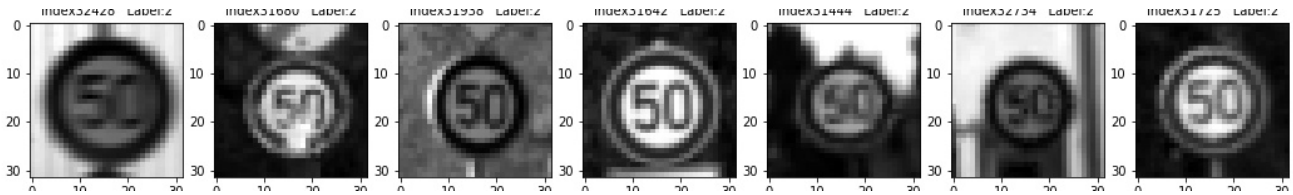
2 -Design and Test a Model Architecture.

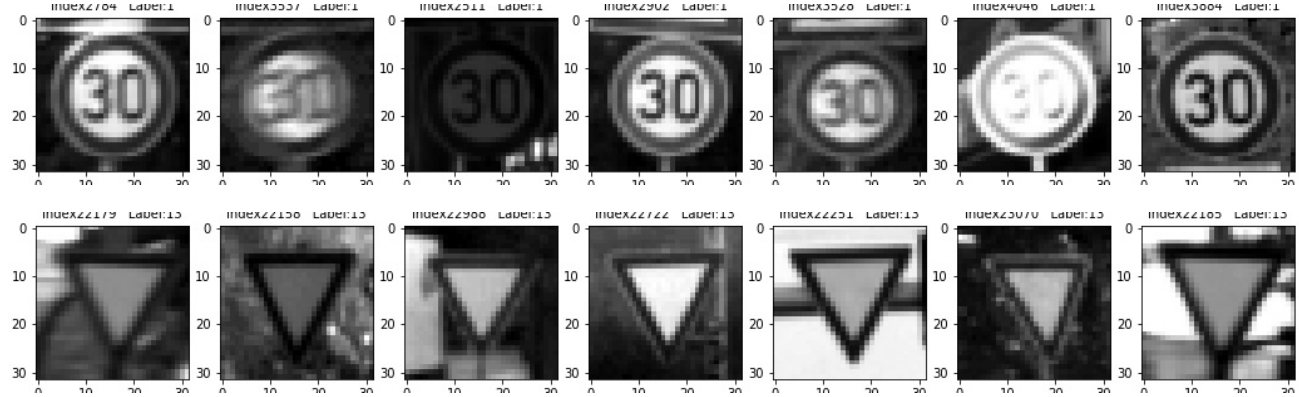
2.1 - Preprocessing.

For the Pre- process was applied 3 steps to the image data.
1^{st} - Converted to gray scale.
2^{nd} - Normalized the image data using the formula (pixel - 128)/ 128
3^{rd} - reshape image to add channel and reach the shape: 32x32x1

These techniques were chosen to normalize the images, so that the data has mean zero and equal variance and ensure a shape to be correctly in the next steps. (Tensorflow functions).

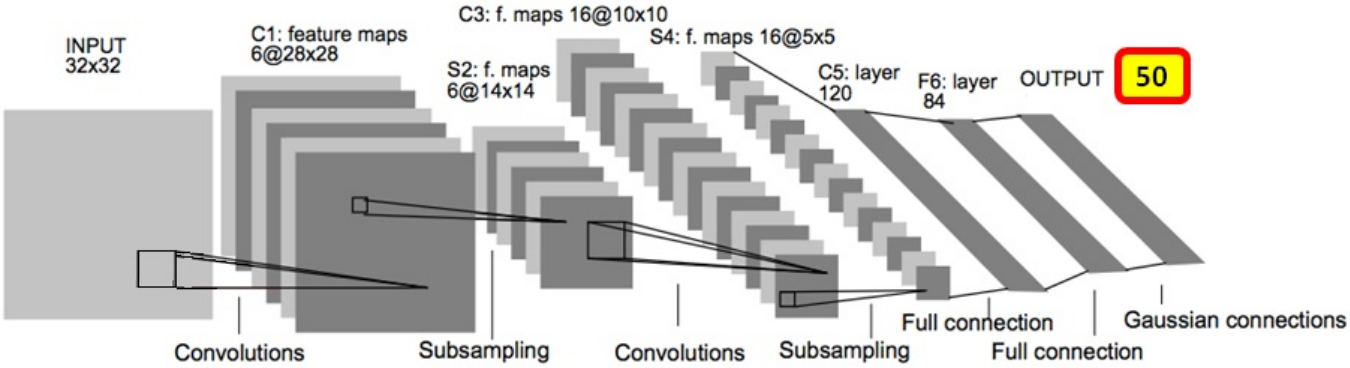
The final results are shown below in 7 pictures as example for the labels 2,1 and 13:





2.2 - Model Architecture.

The Convolutional Neural networks used have the same architecture of the example for Lenet-5. Used in the Udacity class, but the difference is that the last Linear Layer was chane the value from 10 to 50. The architeture is: Two convolutional layers followed by one flatten layer, drop out layer, and Three fully connected linear layers.As the follow picture.



Source: Yan LeCun / edited to adjust the new value for the last layer.

Details about the architeture:

Input: Image in GrayScale with the size: 32x32x1

1. convolution 1: 32x32x1 -> 28x28x6 -> relu -> 14x14x6 (pooling)
2. convolution 2: 14x14x6 -> 10x10x16 -> relu -> 5x5x16 (pooling)
3. flatten: 5x5x16 -> 400
4. drop out: 400 -> 120
5. linear: 120 -> 84
6. linear: 84 -> 50

Output: Return the result of the 2nd fully connected layer. Size = 50

It was canceled the last layer to check the accuracy for the validation Set, the results is very similar , but the results to predict the label for the images from the internet it was not good. For this reason was kept the architeture as the LeNet-5 example.

2.3 - Model Training.

Here working with a lower value for 'Batch_size', I have realized an opportunity to increase the accuracy over than 0.98 in the validation data set, but the results accuracy with the Dowloaded images is very bad.So it was necessary work with a high value for batch size. The Number of EPOCHS greather than 10 did not show advantages to justify use values higher than 10. the Learning rate have contributed working with values less than 10^{-3}. Below the parameter values:

EPOCHS = 10
BATCH_SIZE = 200
learning rate 0.001

The training process was executed using the Tensor flow library. and could be checked from cells 20 to 26 in the code file (./Advanced_Lane_Finding.ipynb)

2.4 - Solution Approach.

The picture below shows the accuracy variation over each EPOCH:

```
Training...

EPOCH 1 ...
Validation Accuracy = 0.632

EPOCH 2 ...
Validation Accuracy = 0.844

EPOCH 3 ...
Validation Accuracy = 0.905

EPOCH 4 ...
Validation Accuracy = 0.929

EPOCH 5 ...
Validation Accuracy = 0.951

EPOCH 6 ...
Validation Accuracy = 0.959

EPOCH 7 ...
Validation Accuracy = 0.959

EPOCH 8 ...
Validation Accuracy = 0.962

EPOCH 9 ...
Validation Accuracy = 0.969

EPOCH 10 ...
Validation Accuracy = 0.971

Model saved
```

.

The Validation accuracy have accomplished the value above the target of the project 0.93.
The parameter and the architeture were checked in order to reached a high accuracy with the new images from the internet. The results will be shown on the next part of this write up.

---

3 - Test a Model on New Images.

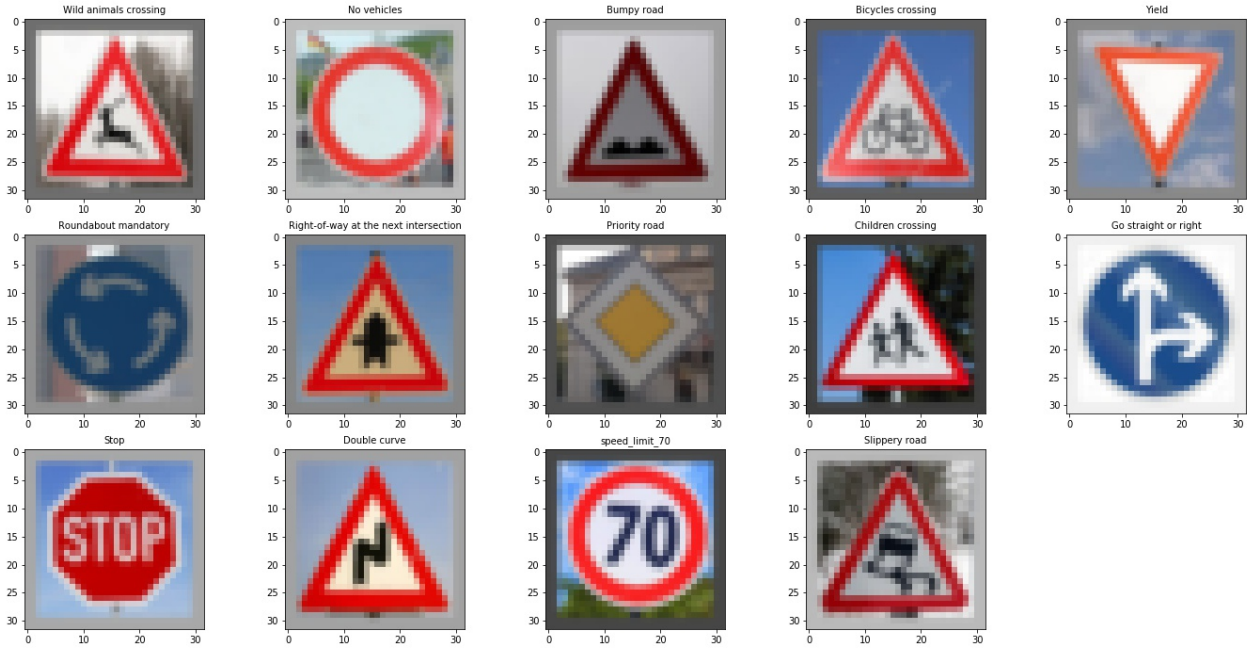
3.1 - Acquiring New Images.

The New images from the internet for German traffic signs could be see in this link (./German_traffic_Signs_Download/original_images).

It was neccesary crop the images to isolate the traffic signs and this step could be see in this link (./German_traffic_Signs_Download/Manual_crop_images).

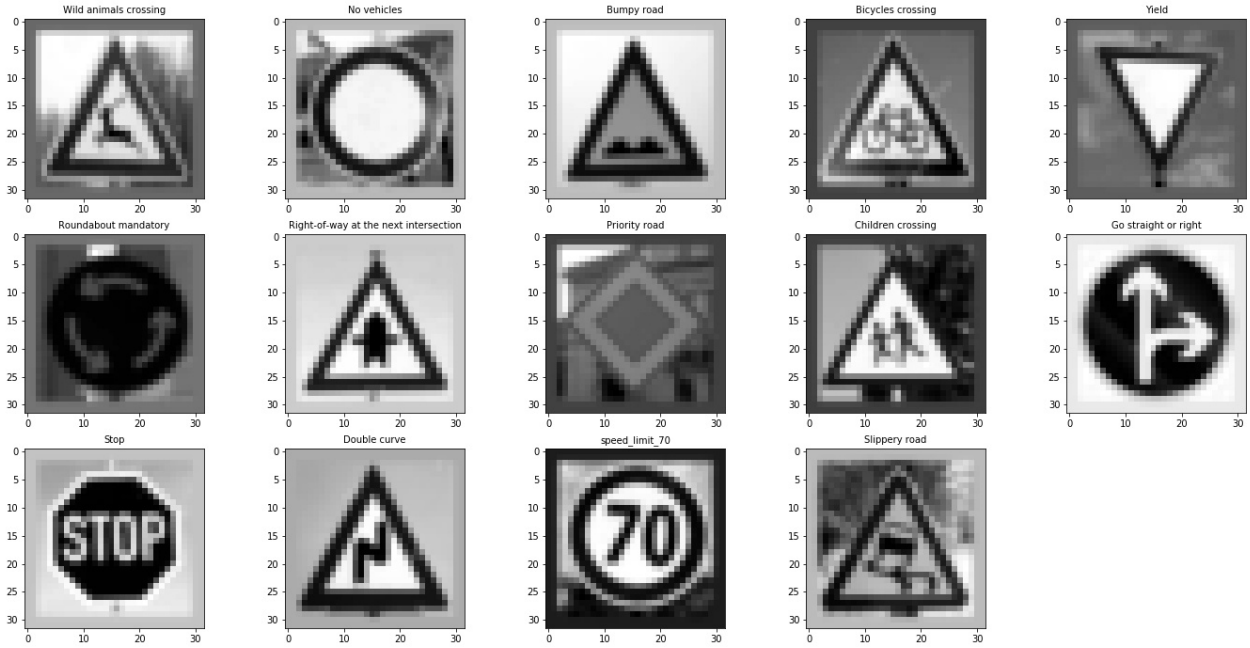
To use the pictures to check the accuracy using the model trained and already explained previously, it was necessary load the data , add margins do adapt the traffic signs image to similar of the data set used ,it was necessary also apply the `blur function` from the cv2 library ,before create the data set with the new images.

Below it is possible to see that it was capture 15 pictures from the internet.



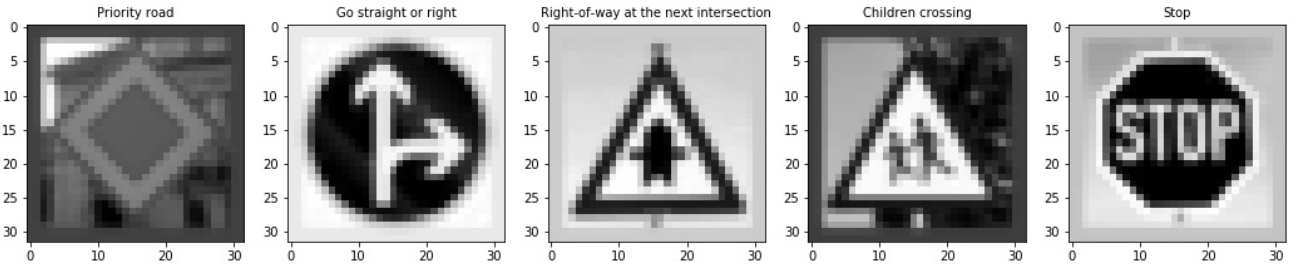
.

It was applied the same preprocessing function to match the new images to the others available on the data-set reference.



.

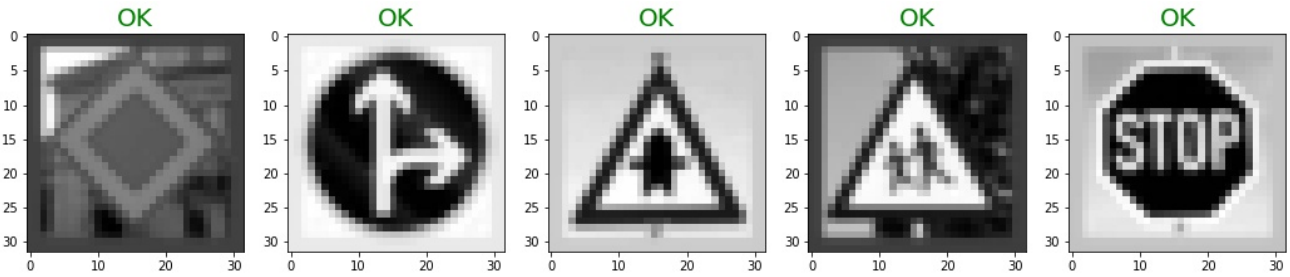
Using the `random.sample` function , it was selected 5 picures to check the accuracy. Below the pictures selected.



.

3.2 - Performance on New Images.

Below we have the same five images and the OK title where the prediction have reached success.



.

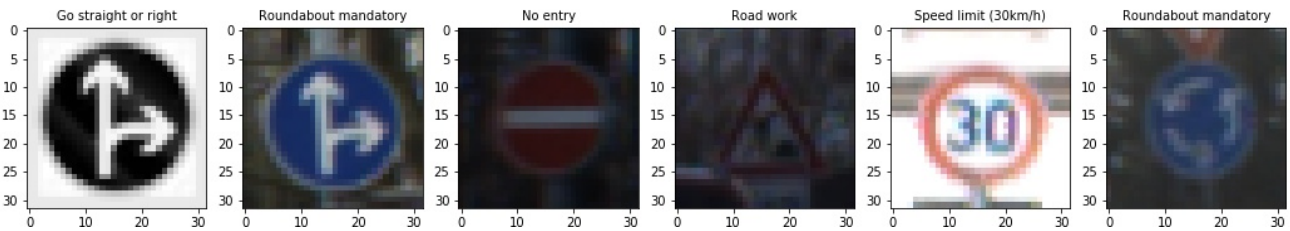
It is possible to see that our model under a small data set with only 5 images reach an accuracy value equal 1 (100%).

3.3 - Model Certainty - Softmax Probabilities.

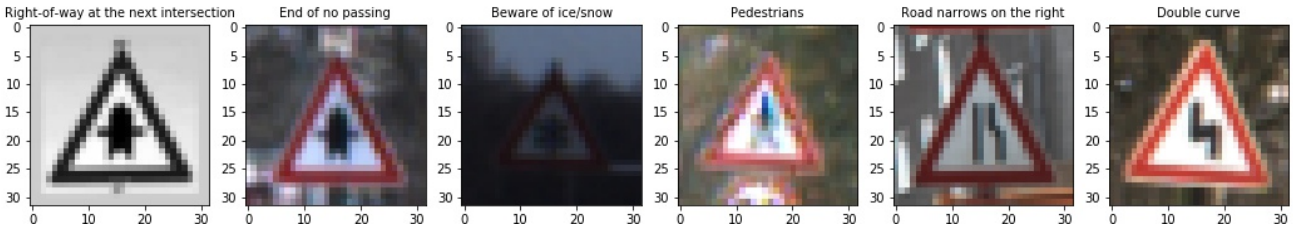
In our example, using the Tensor flow function `tf.nn.top_k` and K value equal to 5. it is possible see 5 probabilities for each image.

From the cells 45 to 49 were applied this function and the final result is an image showing in the left side the original image and the others 5 images in the same row are the probability to the model reach the success to predict the traffic sign. It is possible have a look that in the 5 rows number the model hit the correct label. It is the same result shown in the performance on New Images (item 3.2) confirming the accuracy = 1.

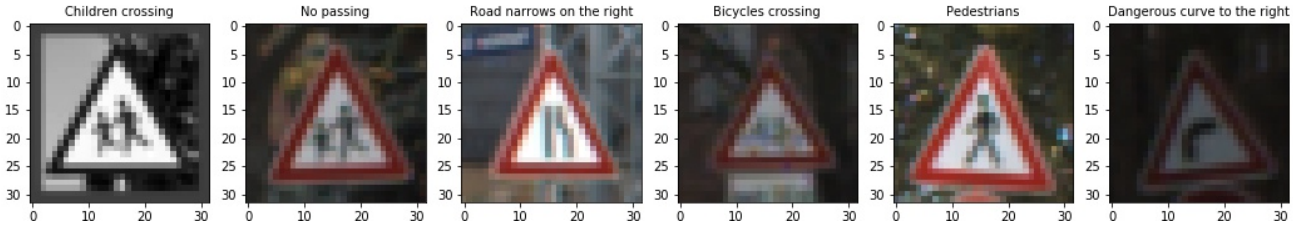
.



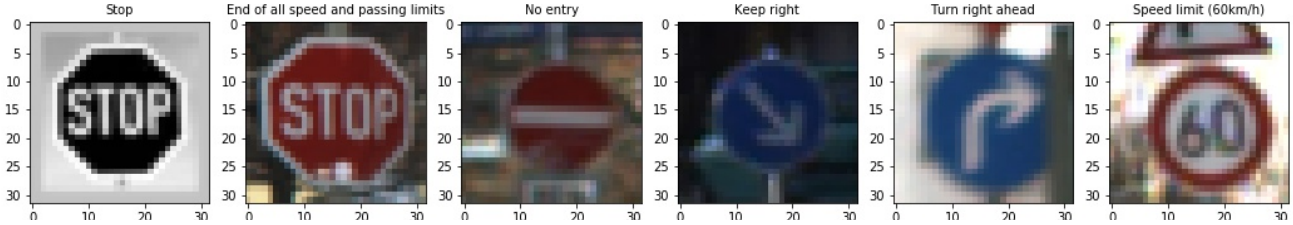
.


Right-of-way at the next intersection
End of no passing
Beware of ice/snow
Pedestrians
Road narrows on the right
Double curve

.


Children crossing
No passing
Road narrows on the right
Bicycles crossing
Pedestrians
Dangerous curve to the right

.


Stop
End of all speed and passing limits
No entry
Keep right
Turn right ahead
Speed limit (60km/h)
STOP
STOP
60