Traffic Sign Classifier

Tim Kaufmann   timothy.w.kaufmann@gmail.com

Rubic Points

https://review.udacity.com/#!/rubrics/481/view

the Rubic points are in *italic* font

Submission Files

The project submission includes all required files.

Ipython notebook with code

HTML output of the code

A writeup report (either pdf or markdown

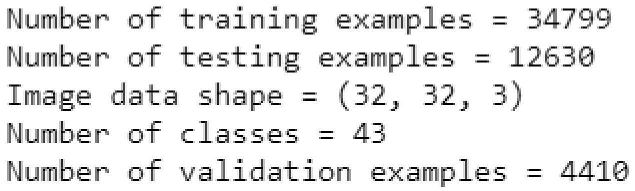
Dataset Exploration

Dataset Summary

The submission includes a basic summary of the data set.

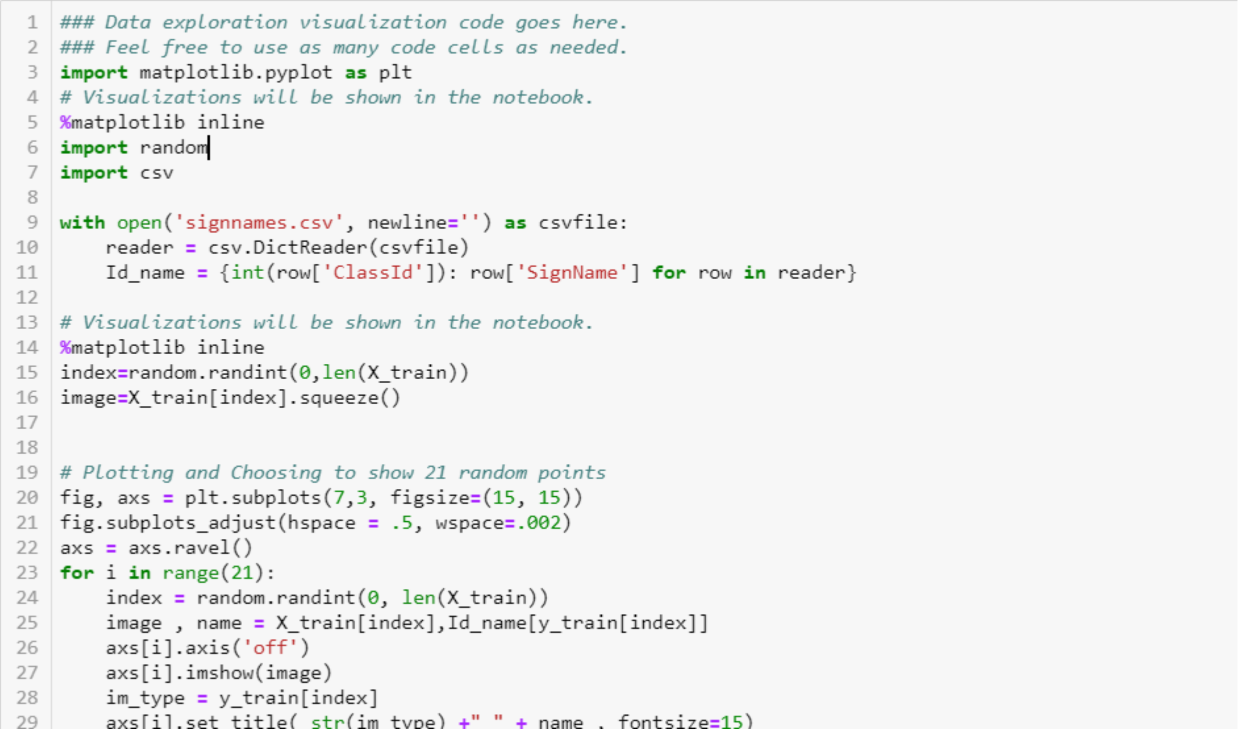
Exploratory Visualization

The submission includes an exploratory visualization on the dataset



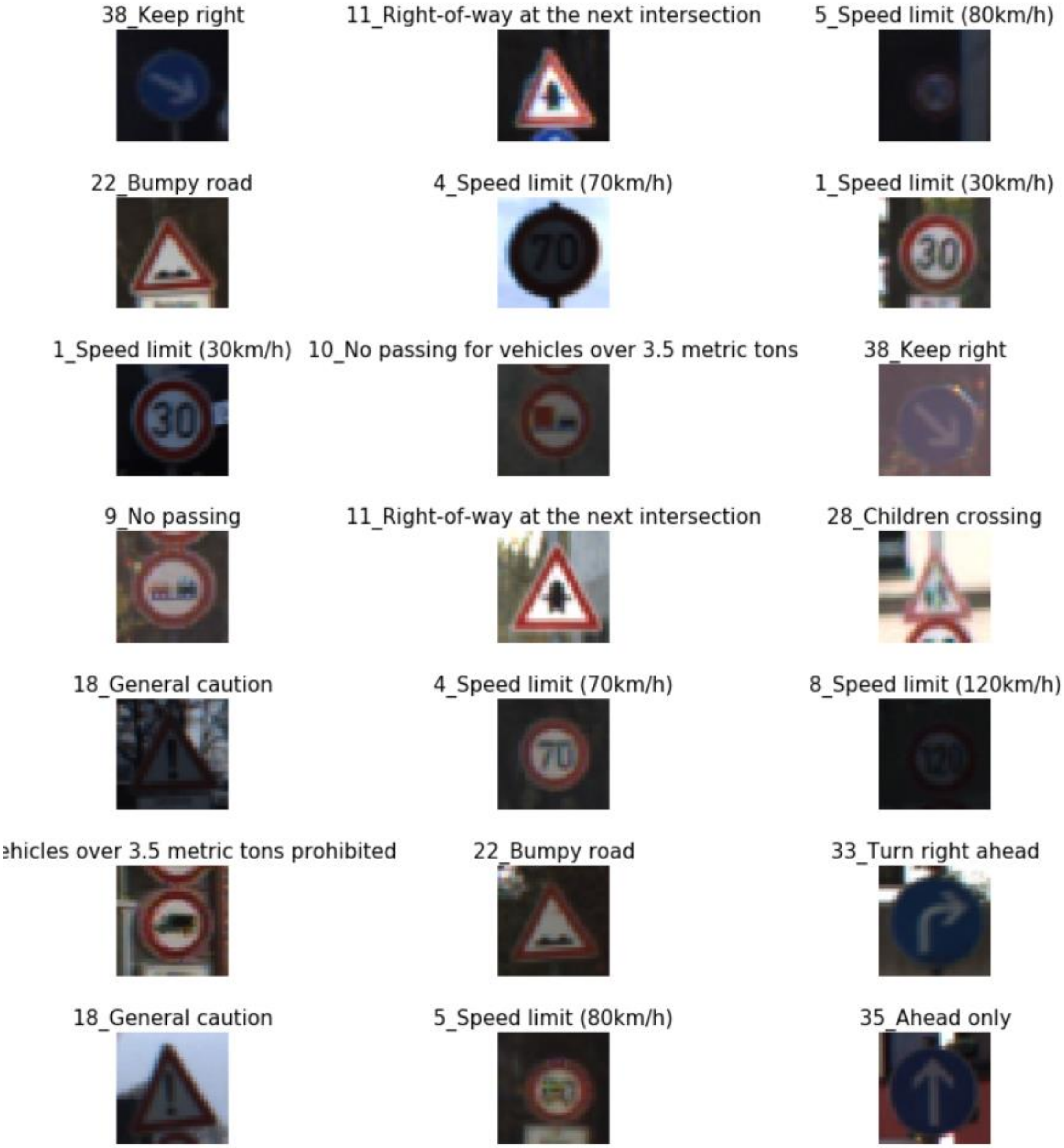
```
Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
Number of validation examples = 4410
```

A Random peek at the images in the data set, the following is the python code to do this:



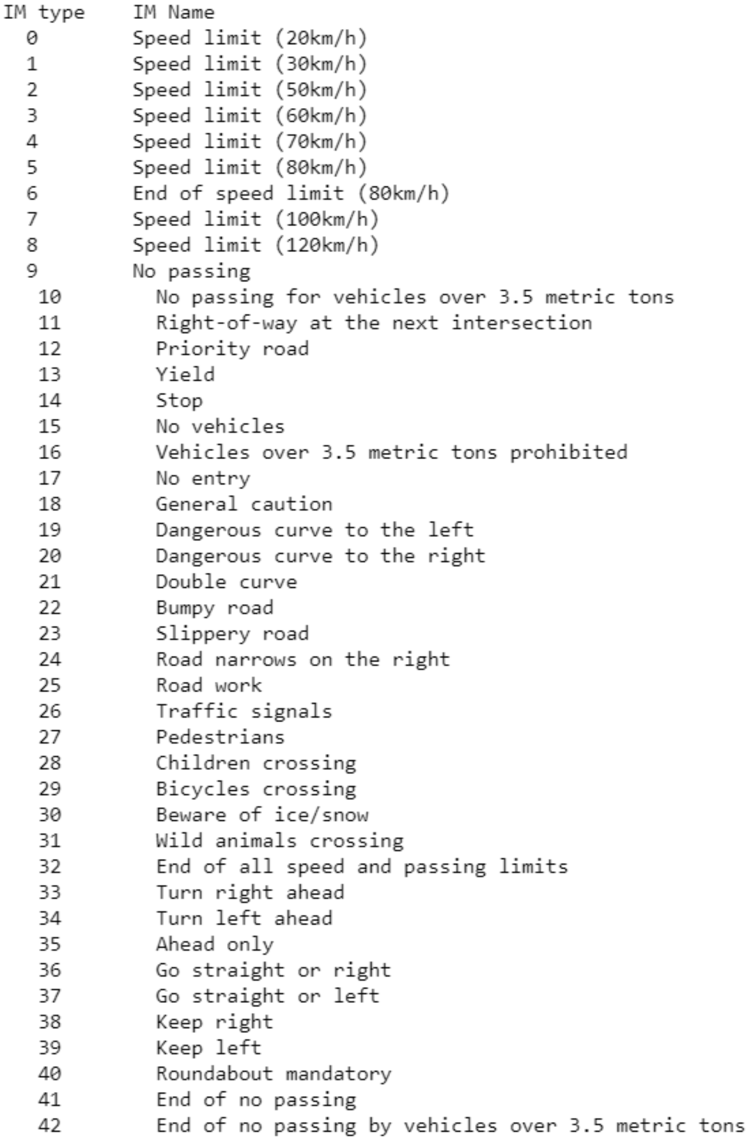
```python
### Data exploration visualization code goes here.
### Feel free to use as many code cells as needed.
import matplotlib.pyplot as plt
# Visualizations will be shown in the notebook.
%matplotlib inline
import random
import csv

with open('signnames.csv', newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    Id_name = {int(row['ClassId']): row['SignName'] for row in reader}

# Visualizations will be shown in the notebook.
%matplotlib inline
index=random.randint(0,len(X_train))
image=X_train[index].squeeze()


# Plotting and Choosing to show 21 random points
fig, axs = plt.subplots(7,3, figsize=(15, 15))
fig.subplots_adjust(hspace = .5, wspace=.002)
axs = axs.ravel()
for i in range(21):
    index = random.randint(0, len(X_train))
    image , name = X_train[index],Id_name[y_train[index]]
    axs[i].axis('off')
    axs[i].imshow(image)
    im_type = y_train[index]
    axs[i].set_title( str(im_type) +" " + name , fontsize=15)
```

The following is the random images displayed:



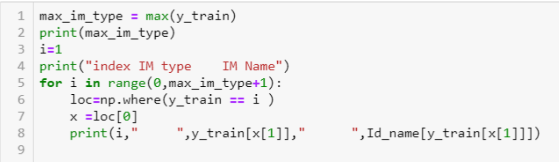
The first number is the image type(truth) in the dataset on the title of each image, second number is a description of the image.

The following is a list of all types of images in the data set:



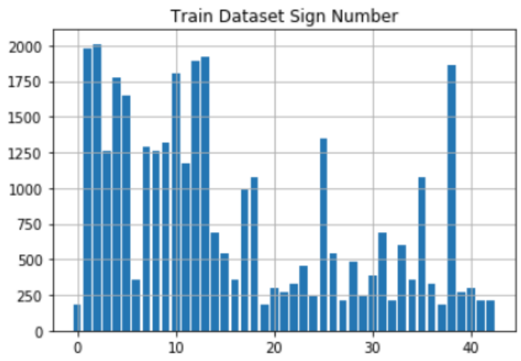
```
IM type    IM Name
  0        Speed limit (20km/h)
  1        Speed limit (30km/h)
  2        Speed limit (50km/h)
  3        Speed limit (60km/h)
  4        Speed limit (70km/h)
  5        Speed limit (80km/h)
  6        End of speed limit (80km/h)
  7        Speed limit (100km/h)
  8        Speed limit (120km/h)
  9        No passing
   10        No passing for vehicles over 3.5 metric tons
   11        Right-of-way at the next intersection
   12        Priority road
   13        Yield
   14        Stop
   15        No vehicles
   16        Vehicles over 3.5 metric tons prohibited
   17        No entry
   18        General caution
   19        Dangerous curve to the left
   20        Dangerous curve to the right
   21        Double curve
   22        Bumpy road
   23        Slippery road
   24        Road narrows on the right
   25        Road work
   26        Traffic signals
   27        Pedestrians
   28        Children crossing
   29        Bicycles crossing
   30        Beware of ice/snow
   31        Wild animals crossing
   32        End of all speed and passing limits
   33        Turn right ahead
   34        Turn left ahead
   35        Ahead only
   36        Go straight or right
   37        Go straight or left
   38        Keep right
   39        Keep left
   40        Roundabout mandatory
   41        End of no passing
   42        End of no passing by vehicles over 3.5 metric tons
```

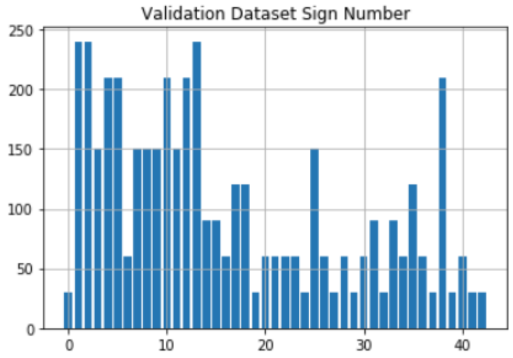
Code to do the above:

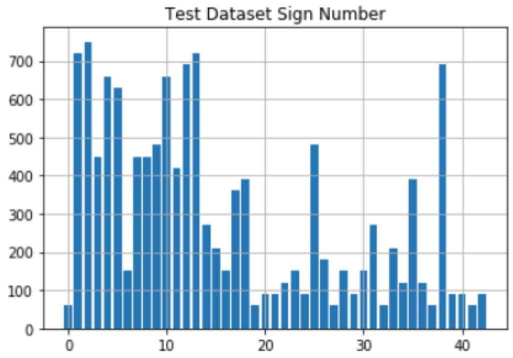


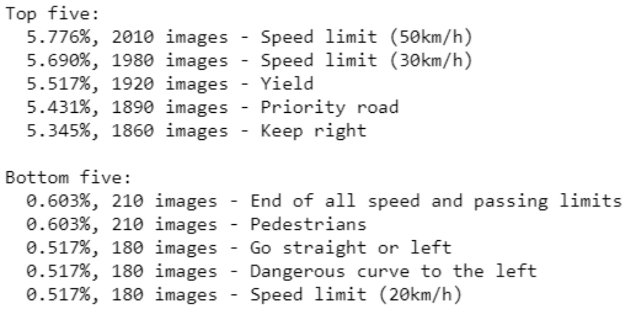
```python
max_im_type = max(y_train)
print(max_im_type)
i=1
print("index IM type    IM Name")
for i in range(0,max_im_type+1):
    loc=np.where(y_train == i )
    x =loc[0]
    print(i,"     ",y_train[x[1]],"      ",Id_name[y_train[x[1]]])

```

Some statics of the datasets:









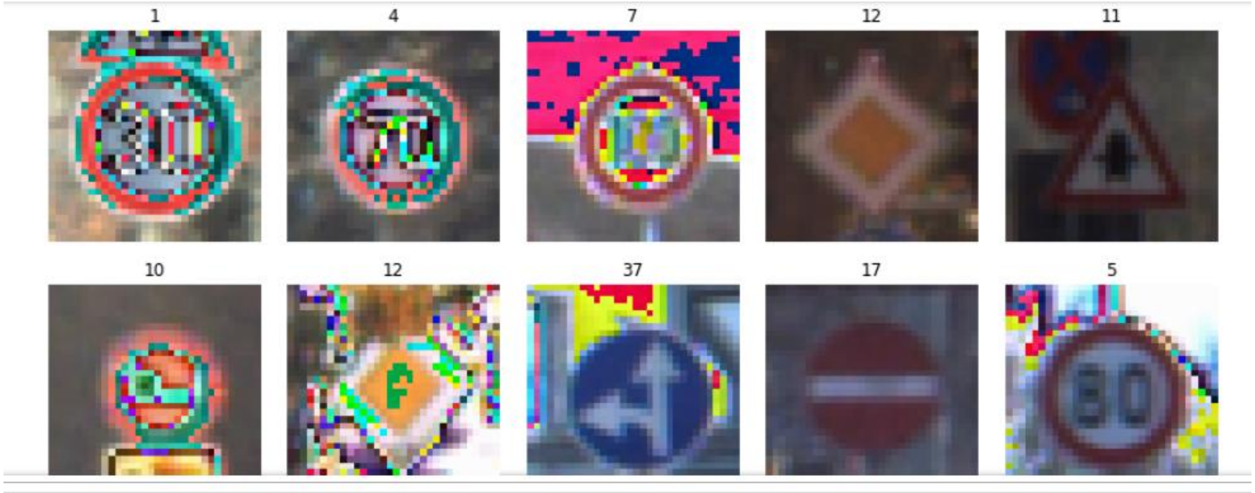
```
Top five:
  5.776%, 2010 images - Speed limit (50km/h)
  5.690%, 1980 images - Speed limit (30km/h)
  5.517%, 1920 images - Yield
  5.431%, 1890 images - Priority road
  5.345%, 1860 images - Keep right

Bottom five:
  0.603%, 210 images - End of all speed and passing limits
  0.603%, 210 images - Pedestrians
  0.517%, 180 images - Go straight or left
  0.517%, 180 images - Dangerous curve to the left
  0.517%, 180 images - Speed limit (20km/h)
```

We can see the data mix is between Train Validation and Test is the same shape, therefore the distribution of sign types is generally the same.

Design and Test a Model Architecture

Preprocessing

The submission describes the preprocessing techniques used and why these techniques were chosen



The following results are after normalizing the images, I chose to normalize by dividing by 128, this is a bit different then was in the class notes. Which was (img -128)/128, I kept having negative values and python was complaining about it, so I chose to have the normalized images between 0 to 2.0. I also found out the my training accuracy was the best with this method 0.939 .

I also performed a grey scale conversion on all of the images a the shape and other features on the sign. This also reduces the complexity of handling a 32 x 32 x 3 to a 32 x 32 x 1.

Model Architecture

The submission provides details of the characteristics and qualities of the architecture, including the type of model used, the number of layers, and the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.

Layer 1 Convolution 5 x 5 stride 1 x 1, 32 x 32 x 1, out 28 x 28 x 6

Layer 2 RELU/Dropout layer, RELU activation/Dropout layer over fitting trends

Layer 3 Max Pooling 2 x 2 stride, out 14 x 14 x 6

Layer 4 Convolution 5 x 5 1 x 1 stride,

Layer 5 RELU/Dropout layer, RELU activation/Dropout layer over fitting trends

Layer 6 Max Pooling, 2 x 2 stride, out 5 x 5 x 16

Layer 7 Flattening Layer,   out 400

Layer 8 Fully connected out 120

Layer 9, RELU/Dropout layer, RELU activation/Dropout layer over fitting trends

Layer 10, Fully Connected, out 84

Layer 11, RELU/Dropout layer, RELU activation/Dropout layer over fitting trends

Layer 12, Fully Connected, out 43

Started with LENET5, config, then added dropout   to understand overfitting.

Model Training

The submission describes how the model was trained by discussing what optimizer was used, batch size, number of epochs and values for hyperparameters.

Optimizer AdamOptimizer

Epocs 40

Batch Size 128

Learn Rate 0.001

Dropout Probability 0.75

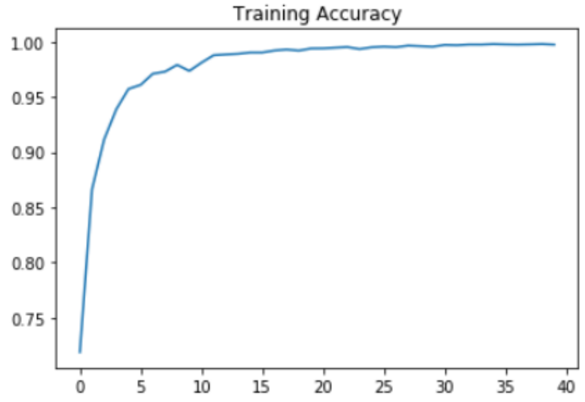
Solution Approach

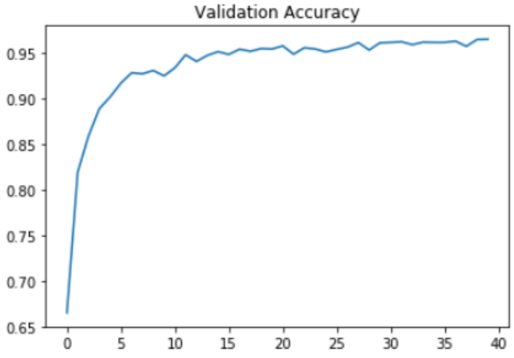
The submission describes the approach to finding a solution. Accuracy on the validation set is 0.93 or greater.

Training accuracy 0.998

Validation accuracy 0.965

Test set accuracy 0.939



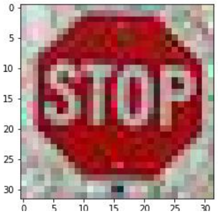


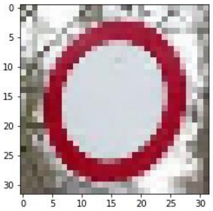
Added normalization to improve accuracy of the validation dataset and fray scalling so color variations did not affect the results.

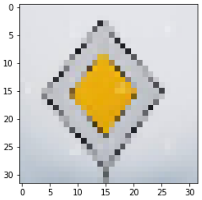
Drop out layers added to combat over training concerns

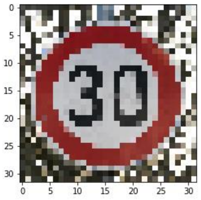
Test a Model on New Images

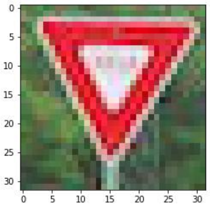
Acquiring New Images











Type	14	15	12	1	13
Sign	Stop	No vehicles	Priority Road	30 kph	Yield

The submission includes five new German Traffic signs found on the web, and the images are visualized. Discussion is made as to particular qualities of the images or traffic signs in the images that are of interest, such as whether they would be difficult for the model to classify.

Added web images stop sign, no vehicles, priority road, 30 kph, and yield signs.

Comments on each sign:

Stop sign, roughly 275 of this type of signs in data base, in the middle as far as number samples

No vehicles sign, slightly over 200 of this type of signs in data base, in the middle as far as number samples.

Priority Road, 1890 samples of this type of sign, in the top five as a percentage of total signs (5.431 %).

30 kph speed limit sign, , 1980 samples of this type of sign, in the top five as a percentage of total signs(5.69 %).

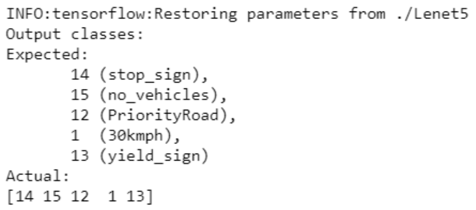
Yield sign, 1920 samples of this type of sign, in the top five as a percentage of total signs (5.69 %).

The images are good quality signs, fairly noise free, good focus ,straight image and not viewed from and angle, and with good information when in 32 x 32 format.

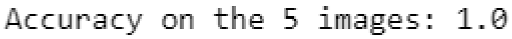
Performance on New Images

The submission documents the performance of the model when tested on the captured *images. The performance on the new images is compared to the accuracy results of the test set.*

All of the images were identified correctly, 100% accuracy for the signs chosen. Program output clips shown below :



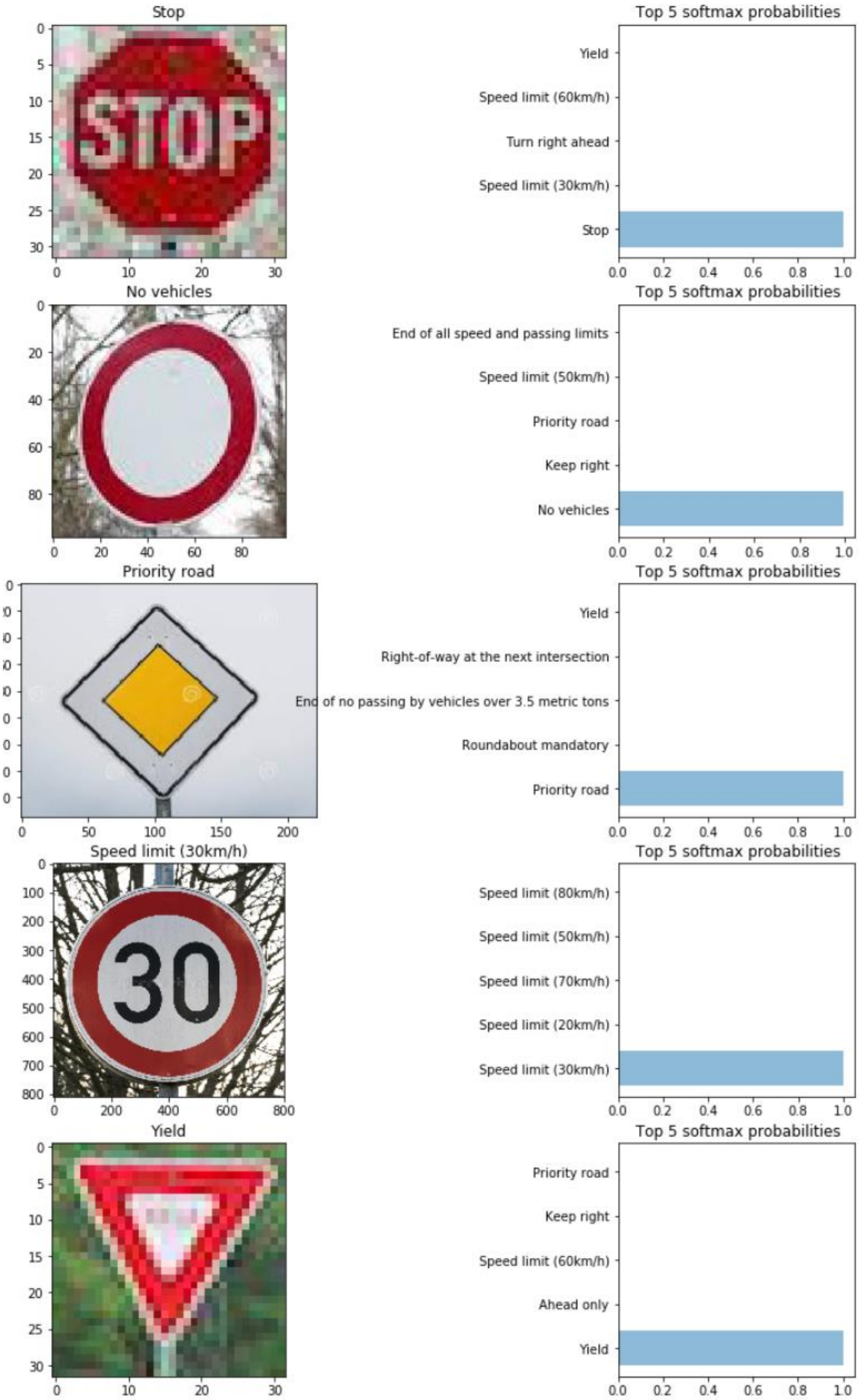
```
INFO:tensorflow:Restoring parameters from ./Lenet5
Output classes:
Expected:
       14 (stop_sign),
       15 (no_vehicles),
       12 (PriorityRoad),
       1  (30kmph),
       13 (yield_sign)
Actual:
[14 15 12  1 13]
```



```
Accuracy on the 5 images: 1.0
```

Model Certainty - Softmax Probabilities

The top five softmax probabilities of the predictions on the captured images are outputted. The submission discusses how certain or uncertain the model is of its predictions.


Stop
0
5
10
15
20
25
30
0
10
20
30
Top 5 softmax probabilities
Yield
Speed limit (60km/h)
Turn right ahead
Speed limit (30km/h)
Stop
0.0
0.2
0.4
0.6
0.8
1.0
No vehicles
0
20
40
60
80
0
20
40
60
80
Top 5 softmax probabilities
End of all speed and passing limits
Speed limit (50km/h)
Priority road
Keep right
No vehicles
0.0
0.2
0.4
0.6
0.8
1.0
Priority road
0
50
100
150
200
Top 5 softmax probabilities
Yield
Right-of-way at the next intersection
End of no passing by vehicles over 3.5 metric tons
Roundabout mandatory
Priority road
0.0
0.2
0.4
0.6
0.8
1.0
Speed limit (30km/h)
0
100
200
300
400
500
600
700
800
0
200
400
600
800
Top 5 softmax probabilities
Speed limit (80km/h)
Speed limit (50km/h)
Speed limit (70km/h)
Speed limit (20km/h)
Speed limit (30km/h)
0.0
0.2
0.4
0.6
0.8
1.0
Yield
0
5
10
15
20
25
30
0
10
20
30
Top 5 softmax probabilities
Priority road
Keep right
Speed limit (60km/h)
Ahead only
Yield
0.0
0.2
0.4
0.6
0.8
1.0