PROJECT 3: Traffic Sign Classification

README of Traffic_Sign_Classifier.ipynb

Autonomous Car Nanodegree program's third project, to design a deep learning framework based on Convolutional Neural Networks was a great learning experience. The foremost objective of this project was to classify various traffic signs belonging to the German Traffic Signs database, with an accuracy of at least 93%.

1. Dataset Exploration

The primary step was to load the image from the dataset, which was readily possible in the notebook from the location, '../data/'. After loading the training, validation and testing files, the shape of each set of data was printed. There are 34799, 4410 and 12630 distinct images in training, validation and testing datasets respectively, with each image having a dimension of 32x32x3. Different classes and their names are shown in table 1

Table 1: Different classes of traffic signs

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

Randomly, and image from the dataset was plotted as shown in the figure 1. Apart from that the histogram of distribution of different classes of images in the dataset was plotted and is shown in figure 2. From histogram it is evident that, the dataset is highly uneven and unbalanced. Few classes of traffic signs have only feeble representation in the dataset.

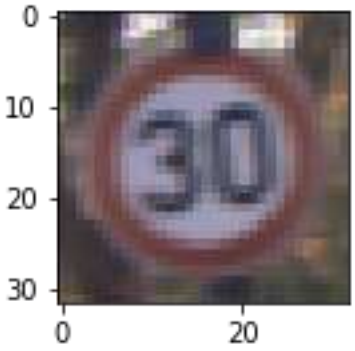

Figure 1: Sample image from dataset

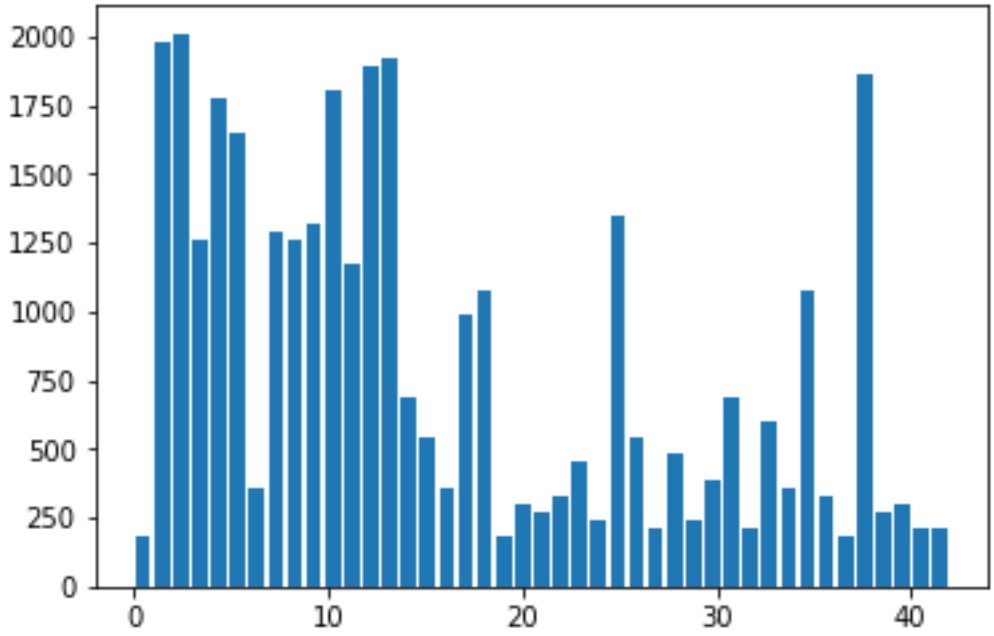

Figure 2: Histogram of classes of images in the dataset

2. Design and Test Model Architecture

2.1. Pre-processing

The images in the dataset are color images with 3 channels. At first the unprocessed image was used for classification and the details will be discussed in the discussion and conclusion section. The color channels just add computational complexity to the algorithm as the color channels doesn't carry any information about the shape of the sign, which is the main distinctive feature for classification. So the image was converted to single channel. Then the grayscale image is normalised to a scale of [0,1] instead of [0,255]. Apart from that, image histogram equalisation is done for enhance the contrast of the image. The algorithm was obtained from reference [1]. A sample pre-processed image is shown in figure 3.

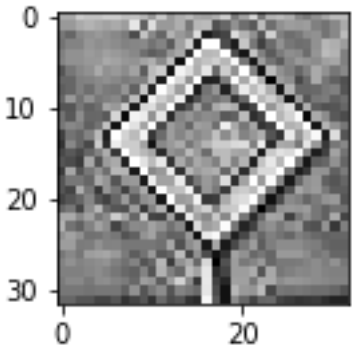

Figure 3: Pre-processed image

The pre-processing step consumes a lot os

[1] Traffic signs classification with a convolutional network, Alex Staravoitau, https://navoshta.com/traffic-signs-classification/

2.2. Model Architecture

The model architecture essentially depends on LeNet 5 and the model discussed in class, Convolutional Neural Network for TensorFlow. The model consists of two convolutional and fully connected layers. The two convolutional layers are made using **'tf.nn.conv2d'** function in TensorFlow, with a stride of 1 and 'SAME' padding. Hence, there is no change in size for the output. The biases are added using **'tf.nn.bias_add'** function. The result of this layer is made nonlinear by applying a relu activation function. Moreover, a Max-pooling operation, having stride 2 and 'SAME" padding, is performed on the result of the relu activation function. Thus a reduction of dimension from 32x32 to 16x16 will occur at the output of first convolutional layer after maxpooling and reduction from 16x16 to 8x8 at the output of second layer.

The number of filters will increase the features extracted from the image. Thus in the first convolutional layer we used a 64 layer filter, making the output dimension after maxpooling 16x16x64. On the second convolutional layer we use a 32 layer filter is used making the resultant shape 8x8x32.

Now this matrix is flattened to form single dimensional tensor of size 2048, which is in turn densely connected to a layer having 256 neurons. Moreover, this layer is connected with the output layer having 43 output neurons.

2.3 Model Training and Solution Approach

The parameters chosen are learning rate, number epochs, dropout probability, and the batch size. Kernel size for filters and strides are given as constant values. The training pipeline is as discussed below.

The **tf.nn.softmax_cross_entropy_with_logits** function is used to find the cross entropy between logits (the predicted output) and the label using softmax function. **tf.reduce_mean** function is used to find the mean of error, which is fed to an optimizer. Adam Optimizer (**tf.train.AdamOptimizer**) implements an extension of stochastic gradient descent.

For evaluation we find the right predictions and find the mean of it to get the accuracy of prediction. To find the accuracy during each epoch the process is repeated for every epoch.

The final validation accuracy = 98%

The test accuracy = 96.8%

3. Test Model on New Images

3.1 Acquiring New Images

Seven different images are obtained by searching 'German Traffic Signs' in google.com. The classes of images and their corresponding labels are as shown in table 2. Also the acquired images are displayed in figure 4. The acquired images are saved in folder named Test_images.

Table 2: Acquired Image details

Sl.No	Label	Sign Category
1	4	70km
2	12	Priority Road
3	13	Yield
4	14	Stop
5	17	No Entry
6	18	General Caution
7	25	Road Work

Figure 4: Acquired images

The obtained images are pre-processed prior to feeding into the prediction pipeline. The pre-processed stop sign image is shown in figure 5.

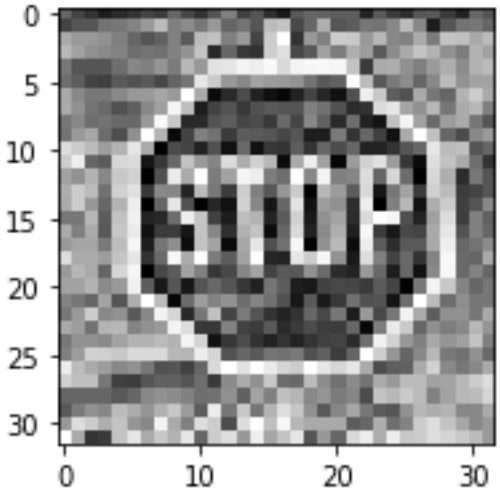

Figure 5: Pre-processed acquired image

3.2. Performance on New Images

The captured image is tested using the created model and except one all other images were classified correctly. The prediction performance matrix is shown in table 3, in which the predicted class number is given in the first raw and actual label id is given in the later row. The 70 km sign was miss classified as 60 km sign.

Table 3: Prediction performance on captured image

Predictions	3	12	13	14	17	18	25
Actual	4	12	13	14	17	18	25

Test Accuracy = 85.7 %

3.3 Model Certainty - Softmax Probabilities

Top 5 softmax probabilities as predicted by the network is analysed in this section. The top five predictions are shown in table 4. The top row shows the most confident prediction made by the network. Those marked in green are correct predictions and the one with red is wrong prediction.

Table 4: Top five predictions

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7
Prediction_1	3	12	13	14	17	18	25
Prediction_2	4	13	12	8	9	27	22
Prediction_3	0	32	17	22	20	26	20
Prediction_4	16	2	14	33	39	11	30
Prediction_5	1	17	33	13	32	28	29

It is evident from the softmax probabilities that all the predictions made are with high confidence. 70 km sign is misclassified as 60 km sign with a very high probability of 0.998,

which is followed by the correct classification with a probability of only 0.0019. The reason might be change of font in actual German sign and that of captured image. Following 5 correct predictions are made with very high probability of 1. The last image also was correctly classified with a very high confidence of .997 softmax probability. The probability matrix is shown in figure 6.

```
probability for top 5 predictions for each image:
0 [  9.98971581e-01   1.01942418e-03   6.16052375e-06   1.08990844e-06
   1.02304864e-06]
1 [  1.00000000e+00   6.70145606e-11   6.85896001e-12   5.66841767e-13
   4.81612254e-13]
2 [  1.00000000e+00   2.63382914e-12   3.75267063e-15   2.67558756e-15
   2.32520264e-15]
3 [  1.00000000e+00   1.86945659e-09   1.34424538e-09   1.98159378e-10
   7.82537229e-11]
4 [  1.00000000e+00   3.47748874e-09   2.47369081e-09   1.87523841e-09
   1.45925916e-09]
5 [  1.00000000e+00   9.22400499e-15   8.21050794e-16   4.31453596e-16
   2.68609897e-16]
6 [  9.97228563e-01   1.43761758e-03   9.09637485e-04   2.03599164e-04
   1.67833306e-04]
INFO:tensorflow:Restoring parameters from ./model
Test Accuracy = 0.857
```

Figure 6: Probability of top 5 predictions for acquired images

The entire process described in the readme can be viewed in **Traffic_Sign_Classifier.html**

THANK YOU