Project: Build a Traffic Sign Recognition Program

Overview

The main objective of the project is to use deep learning to classify traffic sign images from the German Traffic Signs Data set. Training, validation and test is done using the dataset available for the project. Furthermore, after the best performing model has been chosen, it is used to classify images downloaded from the Internet.

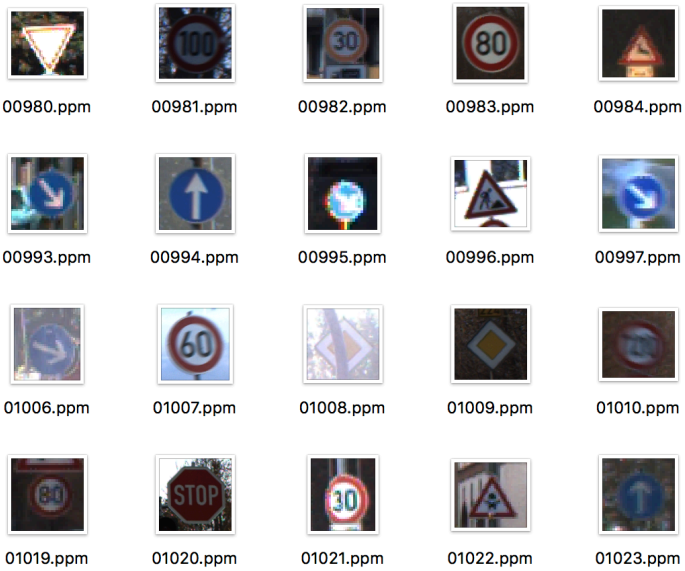
Dataset Exploration

a basic summary of the data set

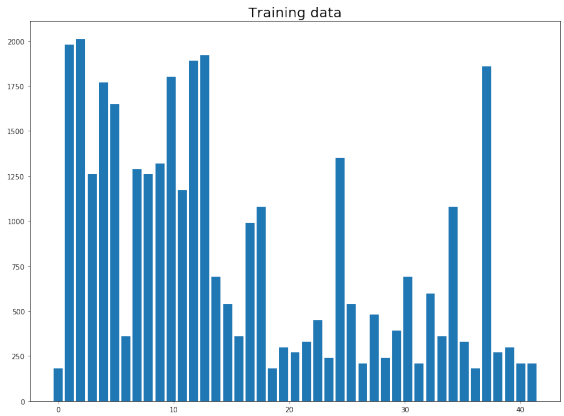
In this project, the intention is to build a CNN architecture to read in 32x32 images to recognise images. The Dataset contains 34799 images for training, 4410 for validation and 12630 for testing. The dataset has 43 classes of image

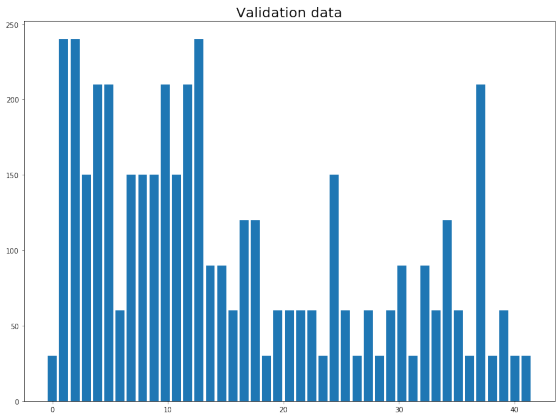
an exploratory visualization on the dataset

The following figure shows 20 random images from the training data set.



Here is an exploratory visualization of the data set. It is a bar chart showing how the data distribute among each class





Design and Test a Model Architecture

preprocessing techniques

The images were converted to grayscale and normalization was performed on the training data.

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.

Layer	Input	Output	Details
Input	32x32x1	-	Input size
Layer 1 Convolutional + activation	32x32x1	28x28x6	Valid padding with ReLU activation
Layer 1 Pooling	28x28x6	14x14x6	Max pooling
Layer 2 Convolutional + activation	14x14x6	10x10x16	Valid padding with ReLU activation
Layer 2 Pooling	10x10x16	5x5x16	Max pooling
Layer 3 Fully connected	400	120	Flattened output from previous layer as input with ReLU activation
Layer 3 Dropout			With 50% keep probability
Layer 4 Fully connected	120	84	With ReLU activation
Layer 4 Dropout			With 50% keep probability
Layer 5 Fully connected	84	43	Output logits

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

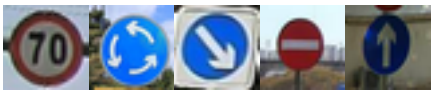
- Learning Rate : 0.001
- Keep probability : 0.5
- Batch Size : 128
- Epochs : 100
- Valid Accuracy : 0.965
- Test Accuracy : 0.947

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

First, I use sklearn.model_selection train_test_split to diverse 20% training dataset as validation set and training data were imported directly to the model without pre-processing. Epochs and batch size were 10 and 128 respectively. And I got 0.954 accuracy on the validation set. Second, I apply pre-processing techniques to the input. the images were converted to grayscale and normalization was performed on the training data. Epochs and batch size were the same as the first time. This time I got 0.972 accuracy on the validation set. But I think the validation set split from training dataset cannot generalize the model to reality because it was so much like training dataset. So the third time I read in valid.p as validation set and drop-out layers were introduced after each activation layer to improve performance. I got 0.904 accuracy. The last time I increased epochs to 100, and finally got 0.965 accuracy on the validation set and 0.947 on test set, which was satisfactory.

Test a Model on New Images

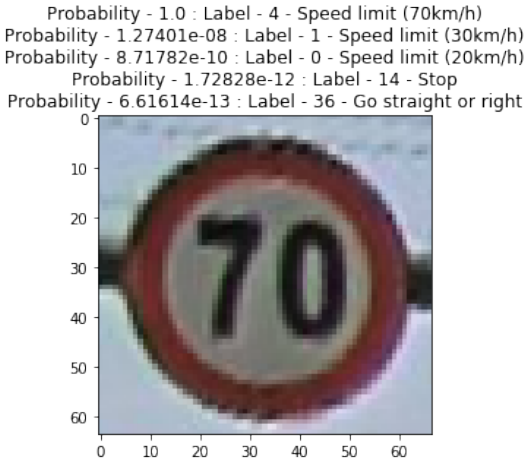
Five images were downloaded from the Internet by searching for "traffic signs" in Google Images Search. I randomly picked images from signs that were in the list in signnames.csv. The images size were adjusted to 32x32 as required by the model. Here are all the images that have been resized.

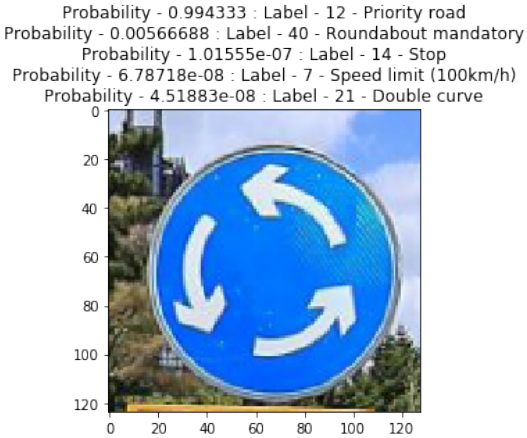


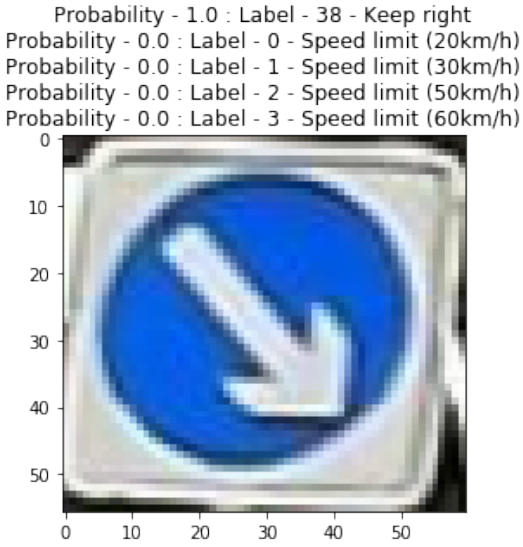
The top five softmax probabilities of the predictions on the captured images are outputted. The submission discusses how certain or uncertain the model is of its predictions.

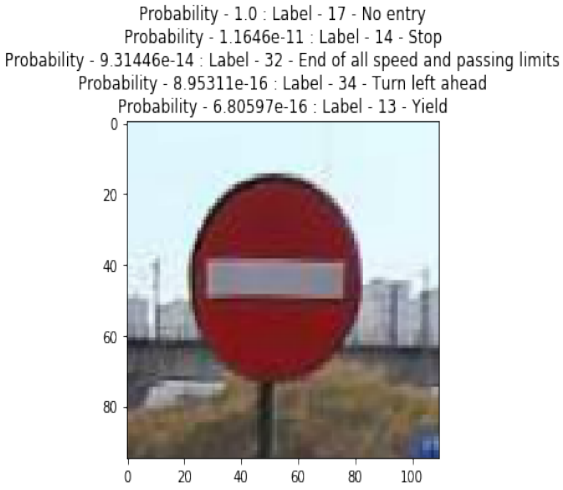
Actual Traffic Sign	Predicted Sign	Probability	Match
Speed limit (70km/h)	Speed limit (70km/h)	1.0	Yes
Roundabout mandatory	Priority road	0.99	No
Keep right	Keep right	1.0	Yes
No entry	No entry	1.0	Yes
Ahead only	Ahead only	0.99	Yes

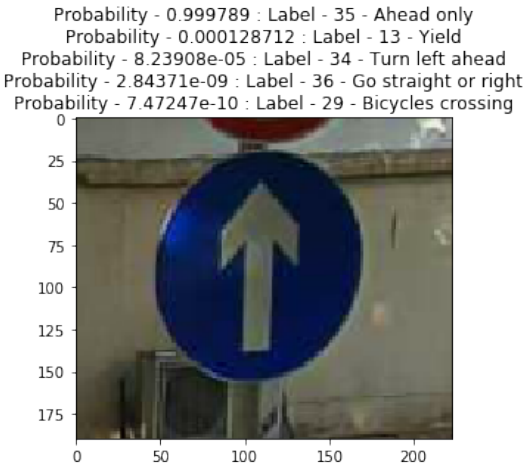
The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of 94.7%. The top 5 predictions show the signs that are accurately predicted(speed limit, keep right, no entry and ahead only), but gives no obvious indication as to why the roundabout mandatory sign is performing poorly. And nearly all signs are predicted with 100% accuracy for the first value. This could have been due to mismatch in quality of the signs in the training dataset and the test image from the Internet. And perhaps the classifier is affected by brightness or color saturation within the image, rather than picking up the features.











Reference

1. LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann.lecun.com/exdb/lenet/
2. Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale Convolutional Networks." IJCNN. 2011. URL: http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf
3. https://github.com/infinityplusb Traffic_Sign_Classifier Repository
4. https://github.com/saksham CarND-Traffic-Sign-Classifier-Project
5. https://github.com/archit69 Traffic_Sign_Classifier Repository
6. https://github.com/gmnvh Self-Driving Cars Repository