TRAFFIC SIGN CLASSIFICATION

• ***The goals of this project is to build a CNN model that classifies German traffic signs.***

PROJECT OVERVIEW

In this project, I have used convolutional neural networks to classify traffic signs. The steps of this project are the following:

1. Load the data set (German traffic signs dataset)
2. Explore, summarize and visualize the data set
3. Design, train and test a model architecture
4. Use the model to make predictions on new images from the web
5. Analyze the softmax probabilities of the new images

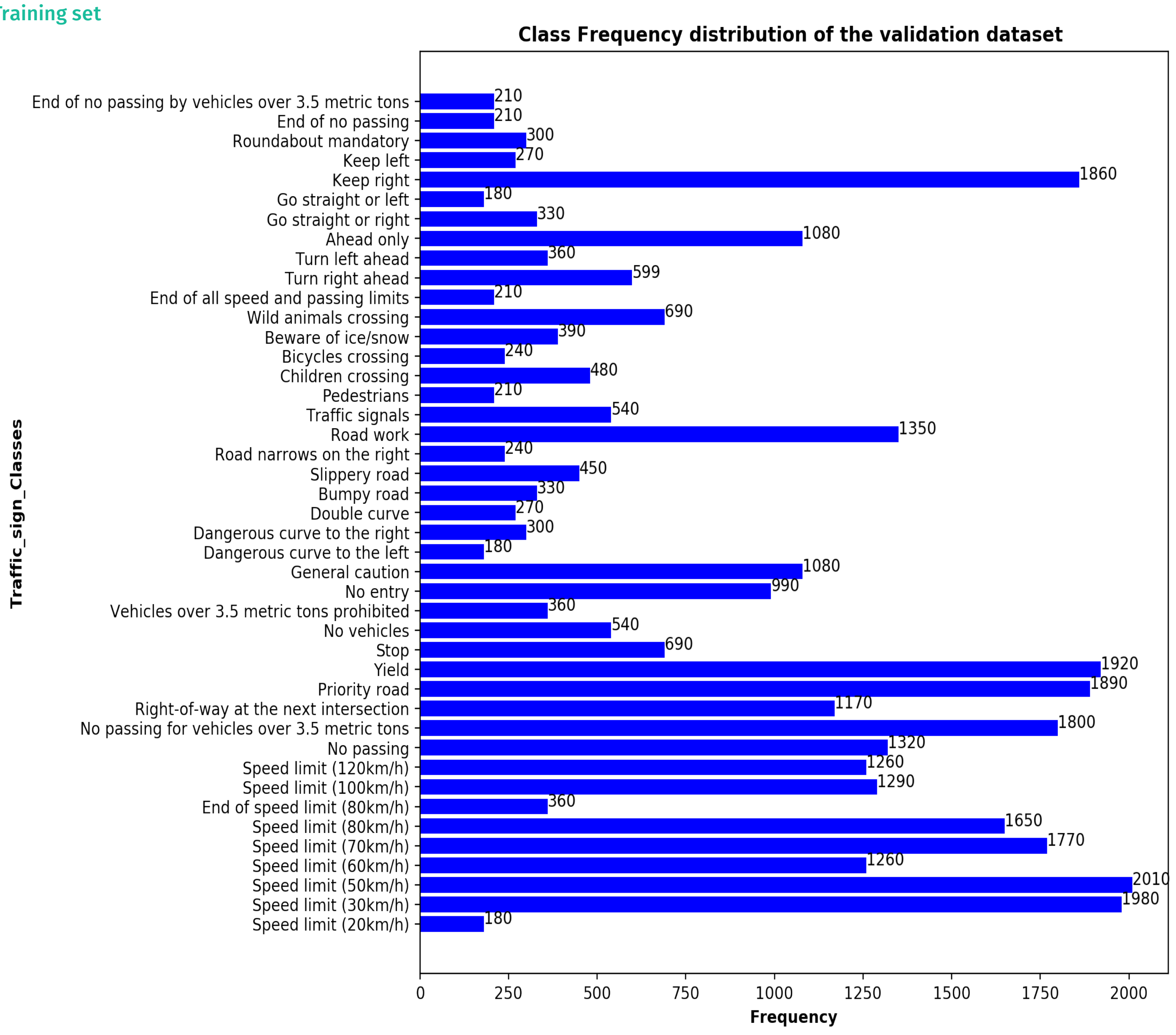
1. DATA SET SUMMARY & EXPLORATION

I used the numpy and/or pandas methods to calculate summary statistics of the traffic signs data set:

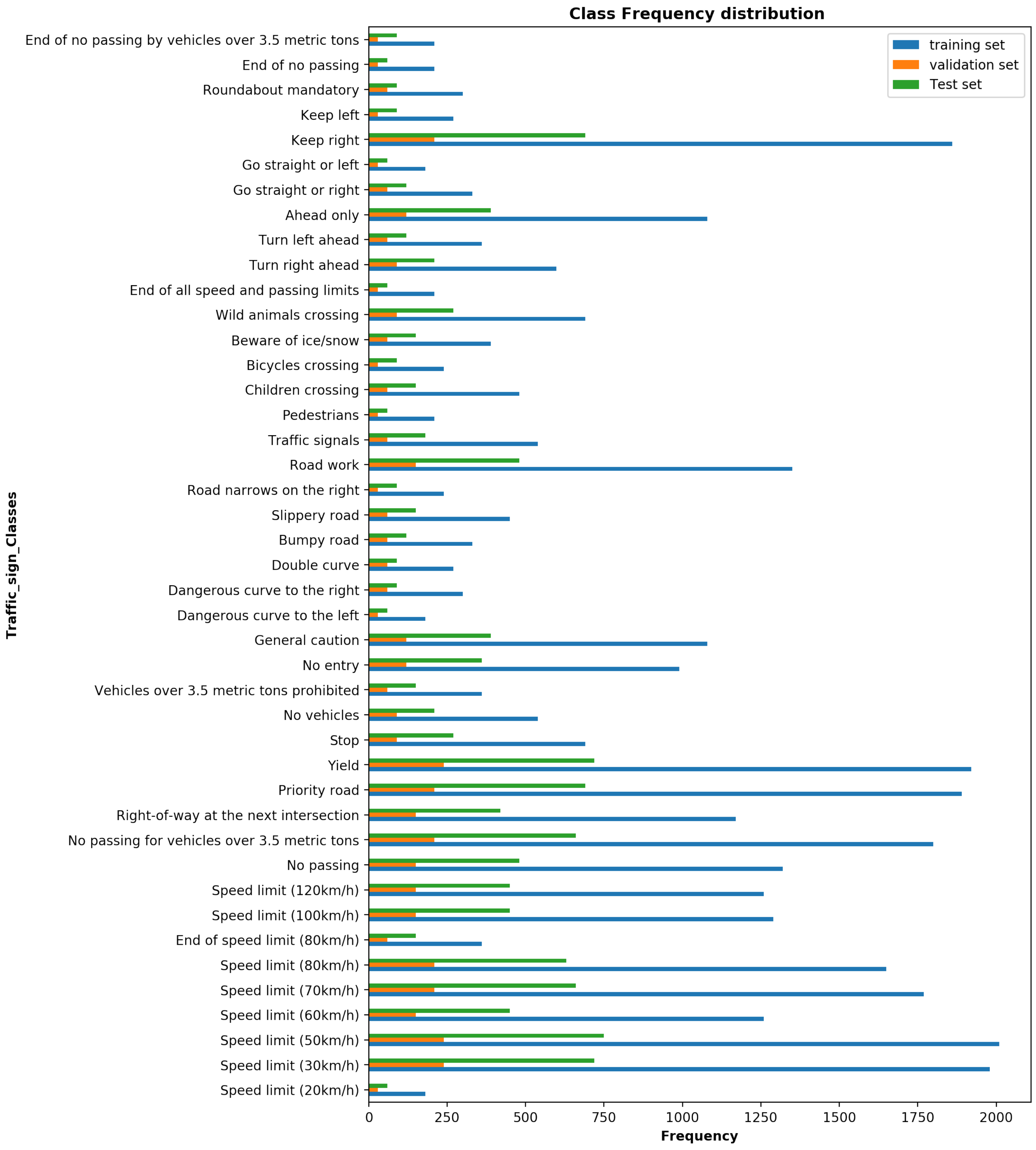
The size of training set is : 34799
The size of the validation set is : 4410
The size of test set is : 12630
The shape of a traffic sign image is : 32x32x3
The number of unique classes/labels in the data set is : 43

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed.

Training set



Training, Validation and Test datasets



2. PRE-PROCESS THE DATA SET

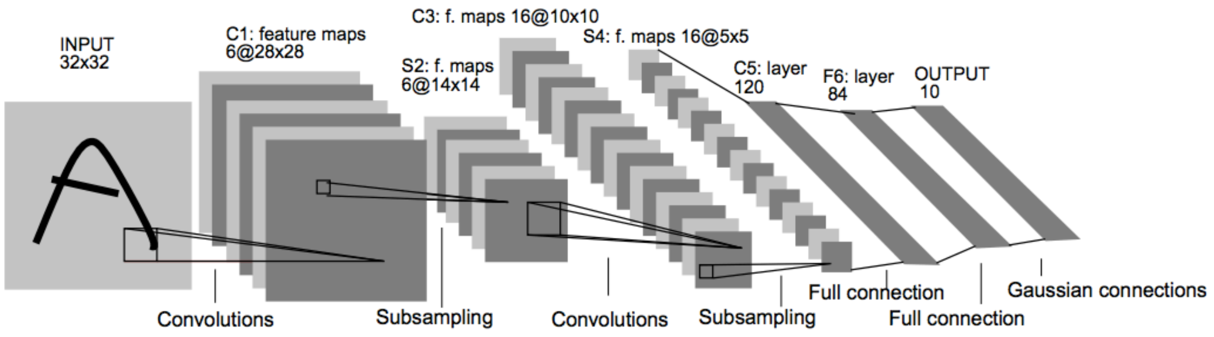
Traffic signs are easily distinguishable by their colors, that's why I kept the images in RGB, because the color information is very important.
I pre-processed the dataset using only 2 steps : Shuffling to serves the purpose of reducing variance and making sure that models remain general and overfit less. Then Normalizing by dividing the dataset images by 255.

I also performed a local standardization per-channel, by standardizing the data, we obtain a mean (μ) = 0 and a standard deviation (σ) = 1.

I decided to perform some data augmentation, but I couldn't because the workspace came with an old version of Tensorflow.

3. MODEL ARCHITECTURE

I have tried two configurations of LeNet :
In the first one, I used the classic CNN Architecture of LeNet-5 with 2 dropouts on the fully connected layers before the output.
and in the second, I made some changes to the classic CNN Architecture of LeNet-5, I first applied 2 convolutions layer before performing a pooling and then convolve with a filter with a size kernel of 4 and 32 feature map (4x4x32), and finally apply 2 dropouts on the fully connected layers before the output.



I adopted the second config due to high accuracies during the testing and this my model consisted of the following layers in the table below. I used convolutions, activations, maxpooling, and linear combinations to classify the 32x32 color images into one of 43 traffic sign categories.

LAYER	DESCRIPTION
Input	32x32x3 RGB image
Convolution + RELU	1x1 stride, same padding, size 5x5x6, Output = 28x28x6
Convolution + RELU	1x1 stride, same padding, size 5x5x16, Output = 24x24x16
Max pooling	2x2 stride, Output = 12x12x16
Convolution + RELU	1x1 stride, same padding, size 4x4x32, Output = 8x8x32
Max pooling	2x2 stride, Output = 4x4x32
Flatten	Flatten the space to one dimension of size 512
Fully Connected layer +RELU+ dropout (60%)	Output = 120
Fully Connected layer + RELU+ dropout (60%)	Output = 84
Fully Connected layer	Output = 43

Training the Model

After customizing the architecture, I built the training pipeline and tested it on the architecture. I one-hot encoded the labels, used softmax cross entropy to determine loss, and used the Adam optimizer.

The learning rate was set to 0.001, the batch size to 128, and the number of epochs to 10.

My final model results were:

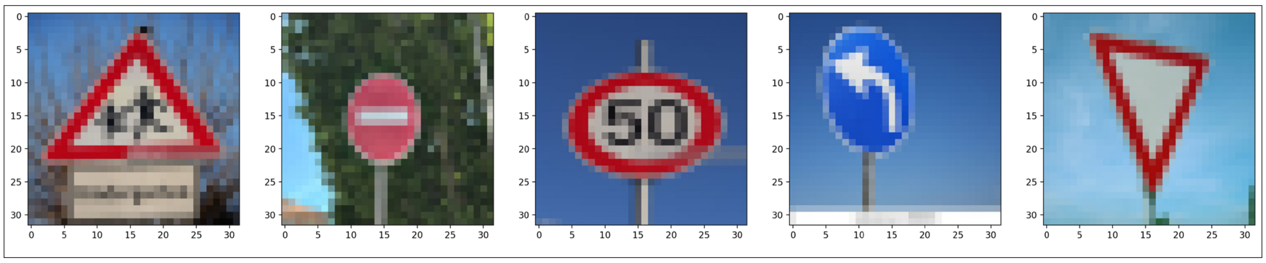
Training set accuracy of 100%
Validation set accuracy of 98%
Test set accuracy of 96.9%

the 1% difference between the test and validation set probably indicates a bit of overfitting in the model.

The initial architecture of LeNet was quite good, but I thought I could make it even better if I add a third convolution layer and apply a regularization technique, I tweaked the hyperparameters and I ended up with high accuracy.

4. TEST A MODEL ON NEW IMAGES

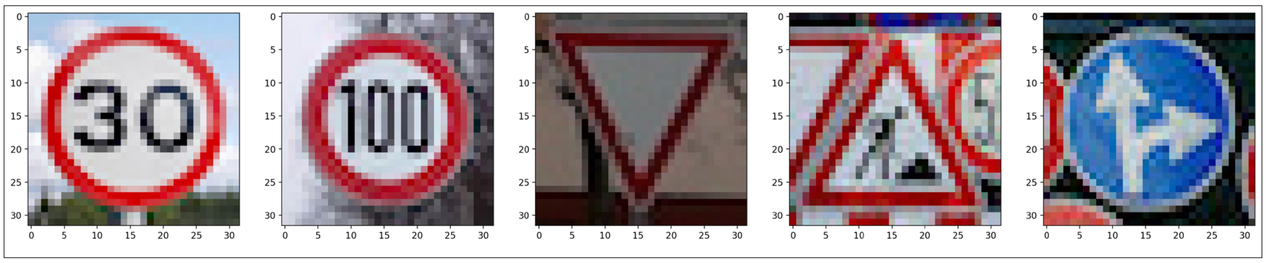
To have more insight into how my model is working, I downloaded a bunch of images of the German traffic signs from the web and I used my model to predict the traffic sign type on 4 different web datasets.



Here are the results of the prediction:

IMAGE	PREDICTION	SOFTMAX PROBABILITY
Children_crossing	No entry	78.7%
No_Entry	No passy	22.3%
Speed_limit_50km/h	Speed_limit_50km/h	100%
Turn_left_ahead	Go straight or right	97.2%
Yield	yield	100%

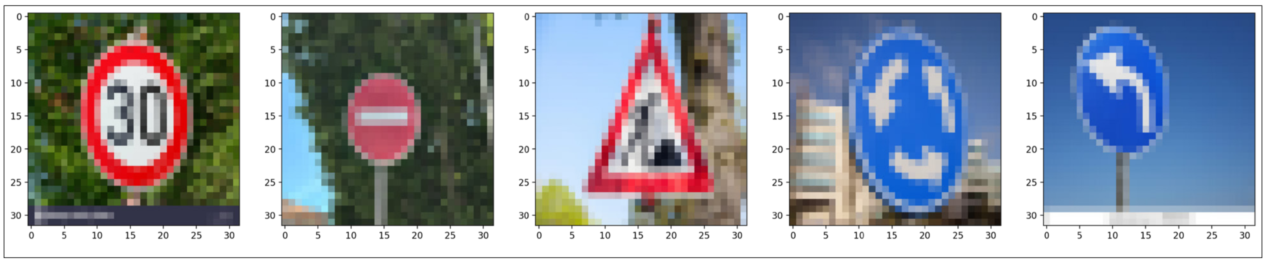
The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%. This compares unfavorably to the accuracy on the test set of 97%



Here are the results of the prediction:

IMAGE	PREDICTION	SOFTMAX PROBABILITY
Speed_limit_30km/h	Speed_limit_30km/h	100%
Speed_limit_100km/h	Speed_limit_100km/h	100%
Yield	Yield	100%
Road work	Road work	100%
straight or right	straight or right	100%

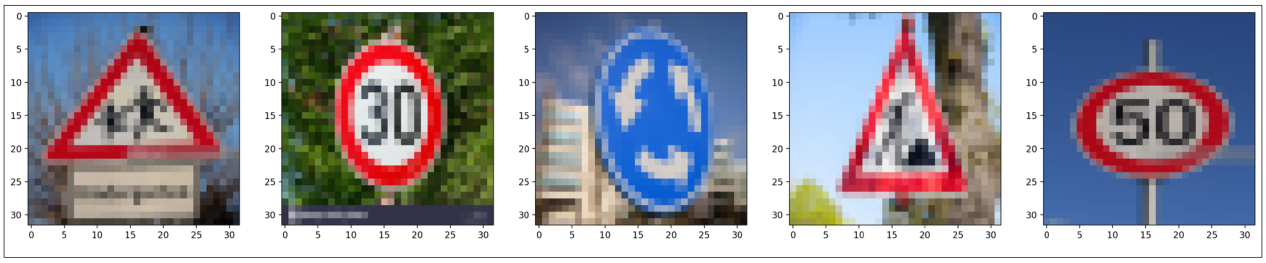
The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 97%



Here are the results of the prediction:

IMAGE	PREDICTION	SOFTMAX PROBABILITY
Speed_limit_30km/h	Speed_limit_30km/h	99.4%
No_Entry	No passy	22.3%
Road work	Road work	100%
Roundabout_mandatory	Roundabout_mandatory	97.9%
Turn_left_ahead	Go straight or right	97.9%

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. This compares less favorably to the accuracy on the test set of 97%



Here are the results of the prediction:

IMAGE	PREDICTION	SOFTMAX PROBABILITY
Children_crossing	No entry	78.7%
Speed_limit_30km/h	Speed_limit_30km/h	99.4%
Roundabout_mandatory	Roundabout_mandatory	97.9%
Road work	Road work	100%
Speed_limit_50km/h	Speed_limit_50km/h	100%

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 97%

Quick debriefing

The model classifies some signs with almost 100% certainty. The rest are negative test cases where the model is expected to be not certain as these traffic signs are not in the training data. Deep learning models are data hungry and I think if we apply some feature engineering techniques like data augmentation would improve the accuracy of the model, there is also the number of samples for a given class.