Write Up for P3: Traffic Sign Classifier

Introduction:

This document is a write up for the 3rd project. As mentioned in the write up template, the objectives are:

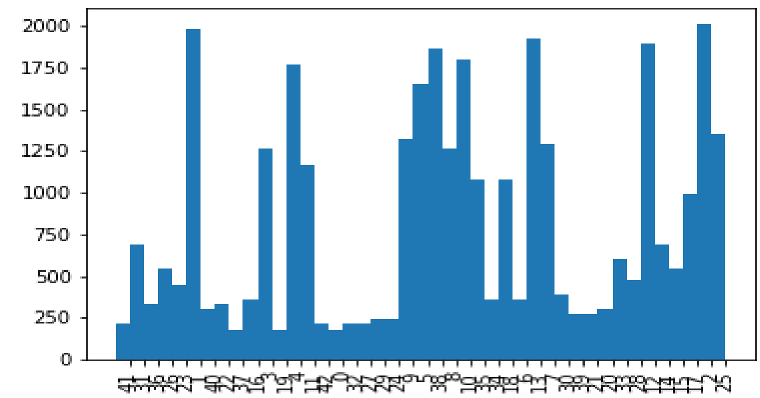
- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

The submission includes this writeup + code in a single file ***Traffic_Sign_Classifier*** in ipynb and html format.

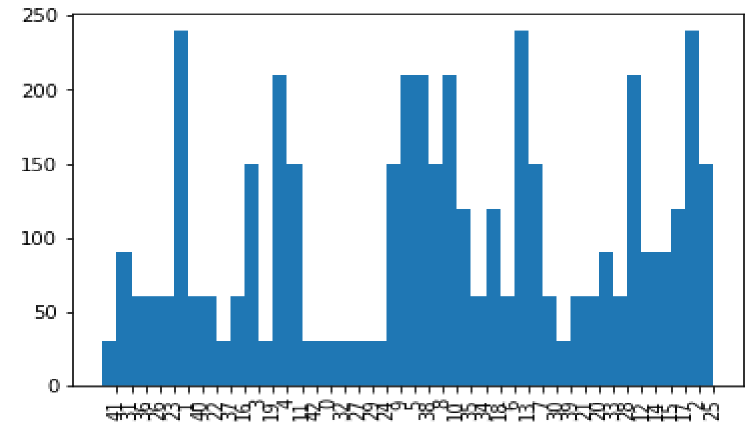
Data set summary and exploration

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

Training classes histogram



Validation classes histogram

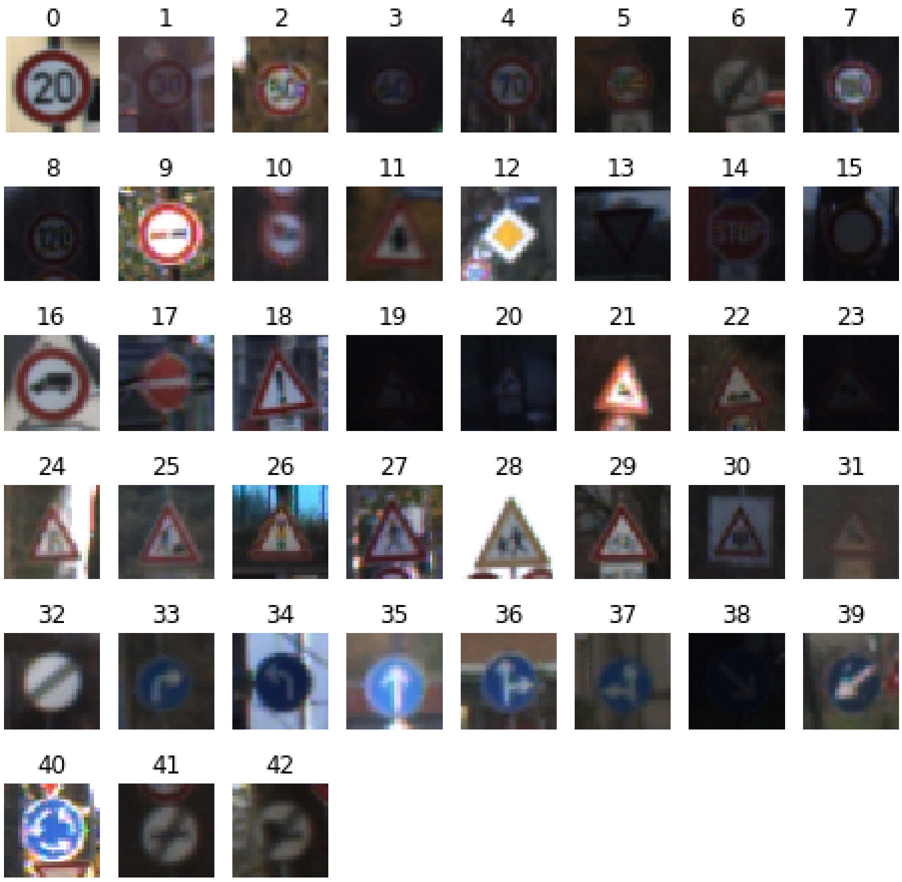


Remarks concerning distribution:
Classes are not equally distributed. On the other side, distribution is similar between training and validation. So as a first approximation, we can use the current training distribution to teach our model.

Design and test a model architecture

Preprocessing

Here're the 43 classes, it is interesting to notice that even though color is not a differentiating factor (still an important factor though). That is, no 2 signs are identical in all aspects except color.
⇒ For that reason I change signs to grayscale
Additionally, I scale the data between -1..1 for reasons that are now obvious and that were discussed in the course and many papers (example: “Efficient Backprop” by Yahn Lecun).



Model architecture

Layer	Description	Output Shape
Input	32x32x1 scaled	-
CONV 3x3 -> ReLU	Stride: 1x1, same padding, filter depth: 16	N_train x 32x32x16
Max POOL 2x2	Stride: 2x2	N_train x 16x16x16
CONV 3x3 -> ReLU	Stride: 1x1, same padding, filter depth: 32	N_train x 16x16x32
Max POOL 2x2	Stride: 2x2	N_train x 8x8x32
FC -> ReLU		N_train x 120
Dropout 0.8		N_train x 120
FC -> ReLU		N_train x 120
Dropout 0.8		N_train x 84
FC -> Softmax		N_train x 43

Model training

Learning rate: 1e-3
Batch size: 64
Optimization: Adam (did not tweak parameters)
Epochs: 15

Approach discussion

To select these parameters, I did several runs displaying both validation error and training error. My approach was first to reach near 100% training accuracy then apply some regularization to increase the validation accuracy.

Training error optimization

Tried increasing the number of layers and filter depth in each layer. For simplicity, I chose a SAME padding followed by Max POOL layer which would divide size by 2. It’s easier for me to keep track of image size (in case it gets big!).
⇒ I found that 3 convolutional layers overfit the model so I settled for 2 layers.
I also tried different learning rates and found that 1e-3 was best.

Validation error optimization

Whenever I got satisfactory training error with minimum configuration, I tried 2 measures to improve validation error: Early stopping and dropout.

I found that 15 epochs coupled with dropout=0.8 provided good results !

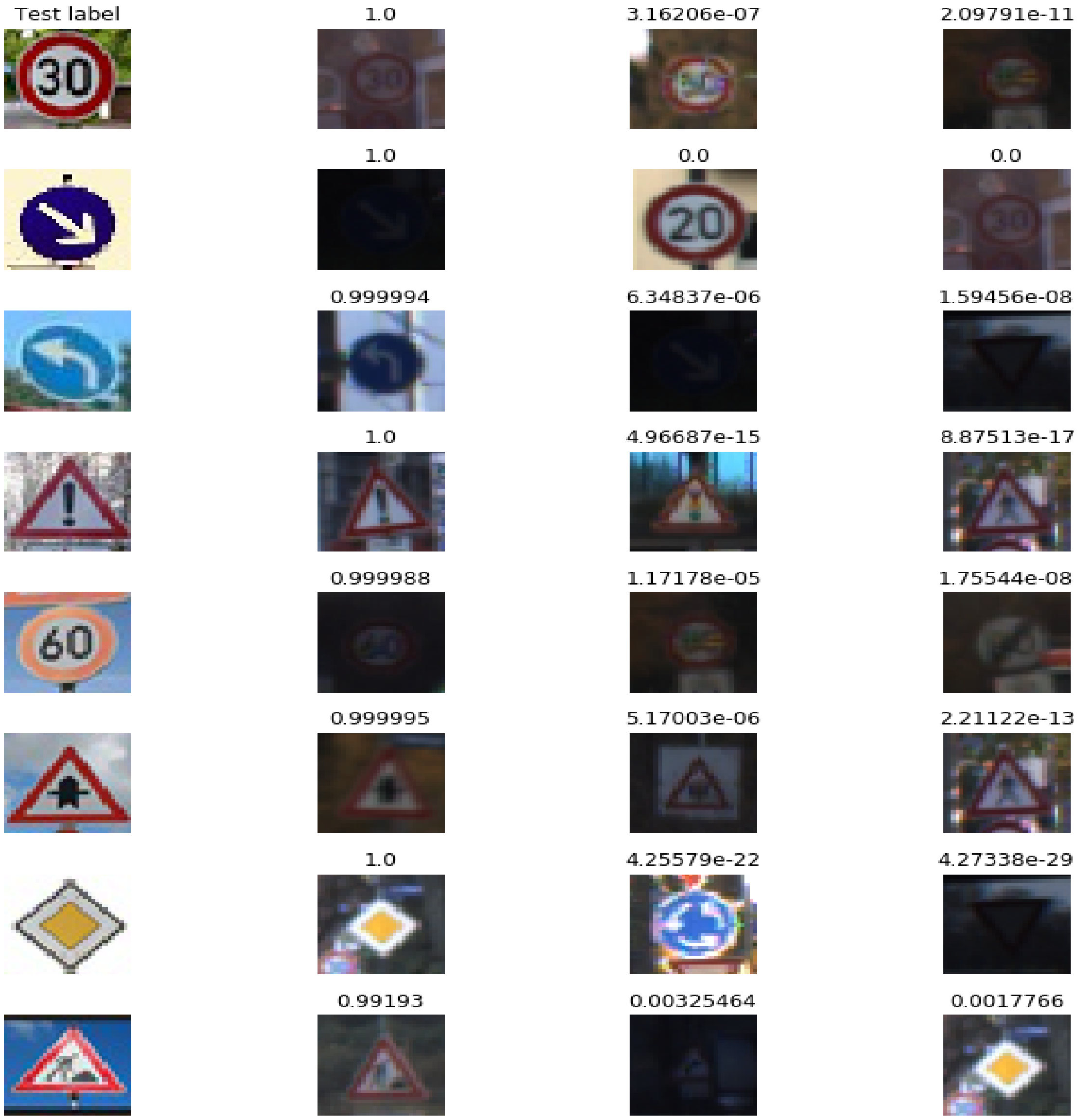
Errors report

Training error: 0.999
Validation error: 0.962
Test error: 0.938

Test model on new images

Here're the images found on the web for the german dataset. Corresponding classes from left to right: [1,38,34,18,3,11,12,25]

Model accuracy =100%
NB: I am not sure if some (or all) of these images exist inside the set.
As can be seen from below table, the model is very certain of the class of each class. I have plotted top 3 candidates for each class.



Discussion

Keeping the same architecture, I believe the model can be further improved in at least 2 ways:

1. Batch normalization
2. Data augmentation in the training set to balance badly classified classes (confusion matrix)

Additionally, one can try a different architecture (inception for example) or transfer learning.

NB: I have written wrote my own NN and CNN from scratch but I was not familiar with tensorflow. After taking this course, I appreciate the ease and efficiency with which one can program such complicated models, thus allowing more time for prototyping!