Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals/steps of this project are the following:

- Load the 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.

Data Set Summary & Exploration

1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the pandas library 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 12360.
- The shape of a traffic sign image is (32, 32, 3).
- The number of unique classes/labels in the data set is 43

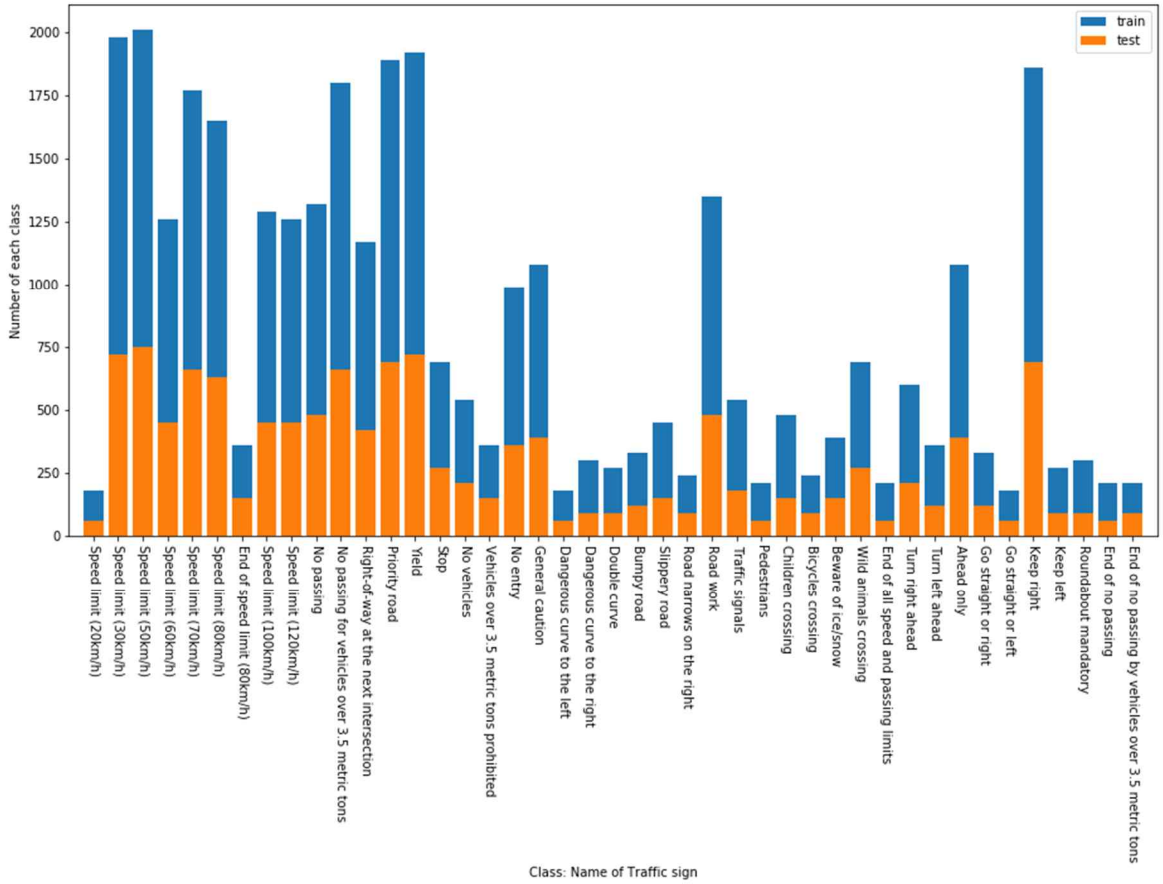
2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the dataset.

- plotting sample of the traffic sign images is as bellows:

- plotting the count of each sign is as bellows:

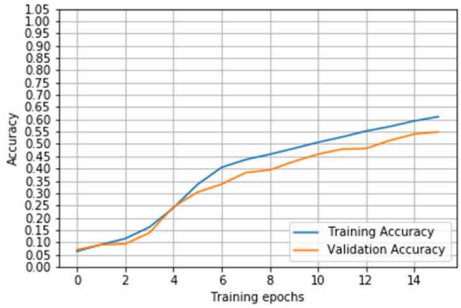
It is a bar chart showing how the data**.**

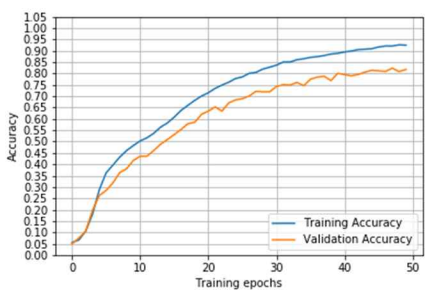


Design and Test a Model Architecture

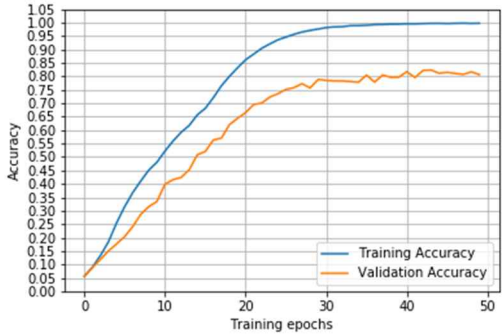
1. Pre-processed image data

- 1st: Color channel image & normalize the image data
  - I tried generated model several times based on color channel. But results are not satisfying the requirement of this project. (The validation set accuracy will need to be **at least 0.93**)

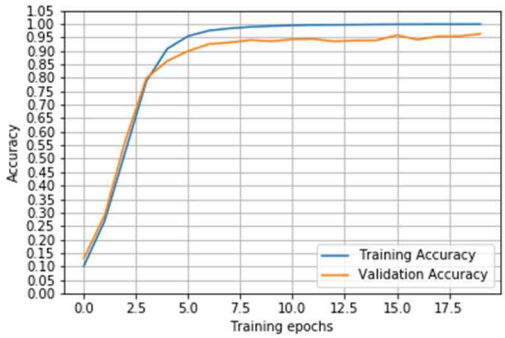




- I tried generated model increase the number of the epoch and more deeper and using more filters. but, this is not good idea because best of the accuracy of the validation set is just about 80%.

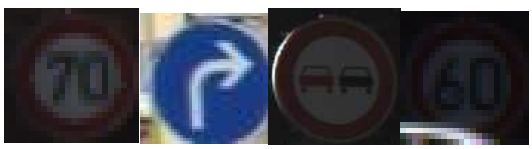


- 2nd: Converting into grayscale images & normalize the image data
  - I searched on Udacity Forum which might be helpful as well.
    - The result of the accuracy of the validation set is about 96%.



- 3th: Converting CLAHE (Contrast Limited Adaptive Histogram Equalization) & normalize the image data
  - I found the paper "A Committee of Neural Networks for Traffic Sign Classification" while searching the new dataset of traffic sign.
    - I applied the CLAHE algorithm because the dataset included below images:

    - Input images



    - The result of the Converting into grayscale images & normalize the image data

- The result of the Converting into CLAHE images & normalize the image data

2. The model architecture

My final model consisted of the following layers:

(Refer: A Committee of Neural Networks for Traffic Sign Classification)

Layer	Description
Input	32x32x1 (CLAHE & Normalize)
Convolution 3x3	1x1 stride, same padding, outputs 32x32x96
RELU	
Max pooling	2x2 stride, outputs 16x16x96
Convolution 4x4	1x1 stride, same padding, outputs 16x16x128
RELU	
Max pooling	2x2 stride, outputs 8x8x128
Convolution 3x3	1x1 stride, same padding, outputs 8x8x256
RELU	
Max pooling	2x2 stride, outputs 4x4x256
Convolution 4x4	1x1 stride, same padding, outputs 4x4x256
RELU	
Dropout	0.5
Flatten	4x4x256 = 4096
Fully connected	(4096, 1024)
Dropout	0.5
Fully connected	(1024, 256)
Dropout	0.5
Fully connected	(256, 43)

3. Train the model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.

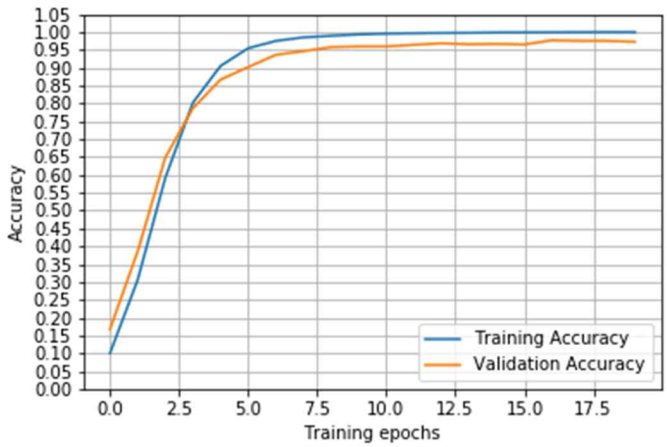
- Optimizer: AdamOptimizer

- Batch size: 256
- Number of epochs: 20
- Learning rate: 0.001

4. Training process

My final model results were:

- Training set accuracy of 99%
- Validation set accuracy of 97.3%
- Test set accuracy of 95.6%



I started with color image data, but I could not create a model that could reach the requirements.

Tried like as follows:

1. **With LeNet. (refer lecture)**
2. **Small number of epochs such as 5, 10 and then, try increase the number of epochs (until 50) (learning rate: 0.001 and checking the Loss)**
3. **Change the architecture of Net (more deeper & wider and using more filters)**
4. **Batch size: 128 or 256 (this depends on the computer memory)**

So, I have to change the training method.

1. **Converting into grayscale images and normalize the image data**
2. **And tried same as sequence like color channels (Base LeNet, change number of epochs and then, more deeper & wider and using more filters)**
3. **if having big gap between training and validation accuracy, this case means over-fitting. Need to apply DROPOUT. Dropout is usually added after Relu activation**

layer. It is not necessary to apply the dropout method if the layer has MAX pooling because the Max pooling method is same purpose.

4. **I checked the accuracy of validation & test dataset. (Since the accuracy is 96 & 95%, I confirmed that it satisfies the requirements.)**

And then, I want to improve the accuracy. So, I decided to work on dataset preprocessing using CLAHE algorithm, because adding layers or filters no longer had a significant impact on performance. (this is just my experience)

Test a Model on New Images

1. **Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.**

   A total of 60 German traffic signs were found, with the first 10 applying the model to relatively clean test data sets.

   Average Accuracy: about 90%

   Input Data #1

   Result #1: Accuracy is 100%

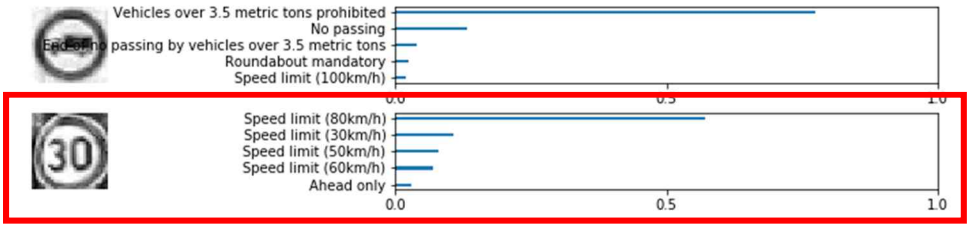
   Input Data #2

   Result #2: Accuracy is 90%

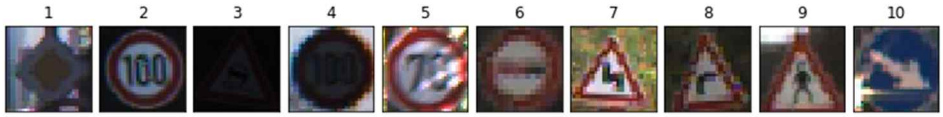
   Test: Speed Limit (30km)

   Prediction: Speed Limit (80km)

   Look like similarly



Input Data #3



Result #3: Accuracy is 90%

Test: I don't know exactly.

Prediction: Right -of-way at the next intersection

Input Data #4

Result #4: Accuracy is 100%

Input Data #5

Result #5: Accuracy is 90%

Test: Speed Limit (50km)

Prediction: Speed Limit (30km)

Although the prediction value is wrong, the output value of the Softmax function shows that 50 km is the second one.

Input Data #6

Result #6: Accuracy is 70%