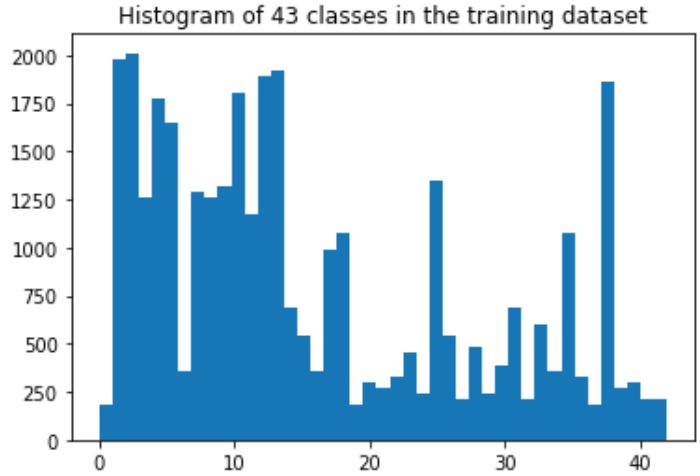
Traffic Sign Classification

Data Set Summary & Exploration

I used the pickle library to import the data and calculate summary statistics of the traffic signs data set:
The size of training set: 34799
The size of the validation set: 4410
The size of test set: 12630
The shape of a traffic sign image: (32, 32, 3)
The number of unique classes/labels in the data set:
[180 1980 2010 1260 1770 1650 360 1290 1260 1320 1800 1170 1890 1920 690
  540 360 990 1080 180 300 270 330 450 240 1350 540 210 480 240
  390 690 210 599 360 1080 330 180 1860 270 300 210 210],
which mean the number of the first class is 180,
the number of the second class is 1980, and so on…

Here is an exploratory visualization of the data set. It is a histogram chart to show the distribution of training dataset:



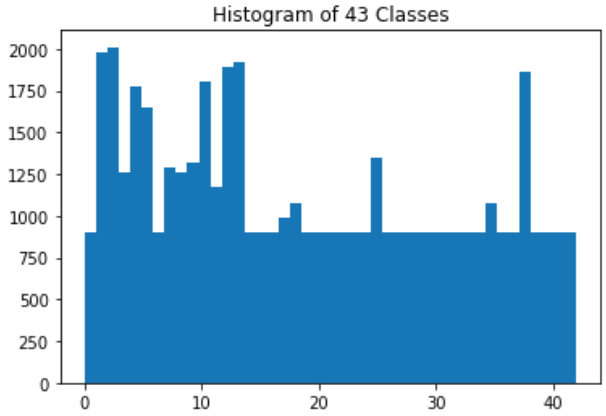
The median of the number of unique classes/labels in the data set: 540.

Design and Test a Model Architecture

As a first step, I used the three channels images, do not convert them to gray color, for the Traffic Sign have the different color which of red, blue, yellow, white, black, and the three channel can stay more information.

The second step, I normalized(/255.0) the image data because it can help model to converge more quickly.

Based on the above two points, after ran the model, I got the 89% accuracy rating. After review the dataset, I found that it is very maldistribution, so I generated additional data for training set, the histogram chart of the distribution of training dataset with additional data:



To prevent overfitting, I also decided to augment the dataset with the random rotation technics, the angle of random rotation is [-15, 15]

The final model architecture:

My final model consisted of the following layers:

Layer	Description
input	32*32*3 RGB image
Convolution 5*5	1*1 stride, valid padding, outputs 28*28*40
RELU	
Max pooling	2*2 stride, valid padding, outputs 14*14*40
Convolution 3*3	1*1 stride, valid padding, outputs 12*12*80
RELU	
Dropout	keep_prob is 0.7
Convolution 3*3	1*1 stride, valid padding, outputs 10*10*160
RELU	
Max pooling	2*2 stride, valid padding, outputs 5*5*160
Flatten	outputs 4000
Fully connected	inputs 4000, outputs 120
RELU	
Fully connected	inputs 120, outputs 84
RELU	
Fully connected	inputs 84, outputs 43
output	43 classes

To train the model, I used an improved LeNet model that add a convolutional layer and a dropout layer, and the default parameters :
The optimizer: Adam
The batch size: 128
The number of epochs: 20
The learning rate: 0.001

My final model results were:
the average accuracy of training dataset: 0.704
the average accuracy of validation dataset: 0.936(offset approximation 0.06)
the test set accuracy: 0.940(offset approximation 0.01)

What architecture was chosen?
LeNet-5 model, and add some improvement that add a convolutional layer and a dropout layer.

Why did you believe it would be relevant to the traffic sign application?
The LeNet-5 model was used to the letter classifier, and the structure of traffic signs are similar with the structure of 26 letters, so I think the model also can be fit it.

How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
The original model's is not good, but the final model can reach the baseline.
The average accuracy of training dataset: 0.704
The average accuracy of validation dataset: 0.936
The test set accuracy: 0.940

How was the architecture adjusted and why was it adjusted?
When I used the default structure of LeNet-5, the model only can get the 0.89 accuracy. To reach the baseline, I think two points, the first point is that get more training dataset, meanwhile, the second point is that improve the model, so I did the below:
1) add a new convolutional layer that just try it, it is maybe work to add the depth .
2) add the dropout that to prevent overfitting
3) raise the number of input and output for the last 3 fully connected layer.

After ran the dataset,
1) it is good idea to add a new convolutional layer.
2) it can prevent overfitting to add the dropout layer, and I try the parameter dropout_ratio between 0.6 to 0.75, the step is 0.05, the effect of 0.70 parameter is the best.
3) raise the number the input and output for the fully connected layer, so I set the numbers: (Input = 4000. Output = 800) is for the first fully connected layer, (Input = 800. Output = 120) is for the second fully connected layer, (Input = 120. Output = 43) is for the third fully connected layer. the effect can work to reach the baseline, **but it is too slow**. After used the default paremeter of LeNet-5, it can also reach the baseline, so finally, I used the (Input = 4000. Output = 120), (Input = 120. Output = 84) and (Input = 84. Output = 43).

Test a Model on New Images

Here are five German traffic signs(from GTSRB) that I found on the web:

Their shape are (54, 53, 3), (45, 48, 3), (97, 100, 3), (105, 98, 3) and (48, 50, 3), for aligning the (32, 32, 3), resized them, and convert them from BRG to RGB color channels.

The first image might be difficult to classify because it is similar with the “Speed limit (20km/h)”. The second image might be difficult to classify because it is similar with the “Dangerous curve to the right”.

Here are the results of the prediction:

Image	Prediction
Speed limit (70km/h)	Speed limit (70km/h)
Turn right ahead	Turn right ahead
Stop	Stop
Speed limit (60km/h)	Speed limit (60km/h)
Go straight or right	Go straight or right

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%.

Output Top 3 Softmax Probabilities For Each Image Found on the Web

The code for making predictions on my final model is located in the 17th cell of the Ipython notebook, I did not use the top 5 soft max, because I think the top 3 is enough to explain the question.

For the all image, the model can correctly predict them, but the probability is not high.

For the 1st image, the top 3 soft max probabilities were:

Probability	Prediction
0.286	Speed limit (70km/h)
0.115	Speed limit (20km/h)
0.467	Speed limit (120km/h)

For the 2nd image, the top 3 soft max probabilities were:

Probability	Prediction
0.308	Turn right ahead
0.052	Turn left ahead
0.032	Ahead only

For the 3rd image, the top 3 soft max probabilities were:

Probability	Prediction
0.119	Stop
0.262	No vehicles
0.082	Wild animals crossing

For the 4th image, the top 3 soft max probabilities were:

Probability	Prediction
0.197	Speed limit (60km/h)
0.031	Speed limit (50km/h)
0.021	Speed limit (80km/h)

For the 5th image, the top 3 soft max probabilities were:

Probability	Prediction
0.166	Go straight or right
0.051	Dangerous curve to the right
0.021	Traffic signals

(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)
It hit the “ValueError: num must be 1 <= num <= 48, not 49” issue. For the time, I did not dive into it.