Traffic Sign Recognition

目录

[Traffic Sign Recognition 1](#_Toc535345266)

[The goals / steps of this project are the following: 2](#_Toc535345267)

[Data Set Summary & Exploration 2](#_Toc535345268)

[1. A basic summary of the data set. 2](#_Toc535345269)

[2. Exploratory visualization of the dataset. 2](#_Toc535345270)

[Design and Test a Model Architecture 4](#_Toc535345271)

[1. Preprocessed the image data. 4](#_Toc535345272)

[1.1 As a first step, I decided to convert the images to grayscale. 4](#_Toc535345273)

[1.2 Normalized the image data. 4](#_Toc535345274)

[2. Final model architecture looks like. 5](#_Toc535345275)

[3. Trained model. 6](#_Toc535345276)

[4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. 6](#_Toc535345277)

[Test a Model on New Images 7](#_Toc535345278)

[1. Choose ten German traffic signs found on the web and provide them in the report. 7](#_Toc535345279)

[2. The model's predictions on these new traffic signs. 8](#_Toc535345280)

# The goals / steps of this project are the following:

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

# Data Set Summary & Exploration

## 1. A basic summary of the data set.

I used the pickle library to load the dataset ,then use numpy 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:12630

The shape of a traffic sign image is:(32,32,1)

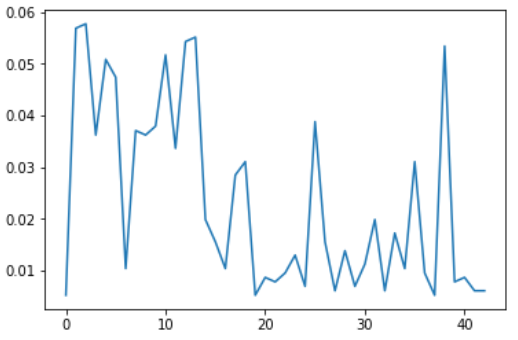
The number of unique classes/labels in the data set is:43

## 2. Exploratory visualization of the dataset.

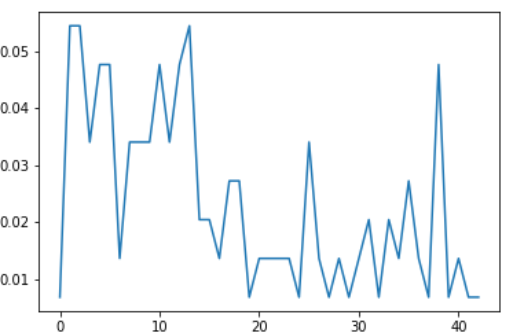
Random input images in X\_train dataset:



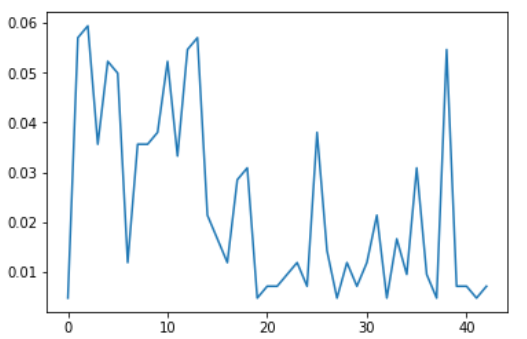
Class distribution in X\_train dataset:



Class distribution in X\_valid dataset:



Class distribution in X\_test dataset:



# Design and Test a Model Architecture

## 1. Preprocessed the image data.

### As a first step, I decided to convert the images to grayscale.

X\_train\_original = X\_train

X\_train\_gray = np.sum(X\_train\_original/3, axis=3, keepdims=True)

X\_valid\_original = X\_valid

X\_valid\_gray = np.sum(X\_valid\_original/3, axis=3, keepdims=True)

X\_test\_original = X\_test

X\_test\_gray = np.sum(X\_test\_original/3, axis=3, keepdims=True)

### Normalized the image data.

X\_train\_normalized = (X\_train\_gray - 128)/128

X\_valid\_normalized = (X\_valid\_gray - 128)/128

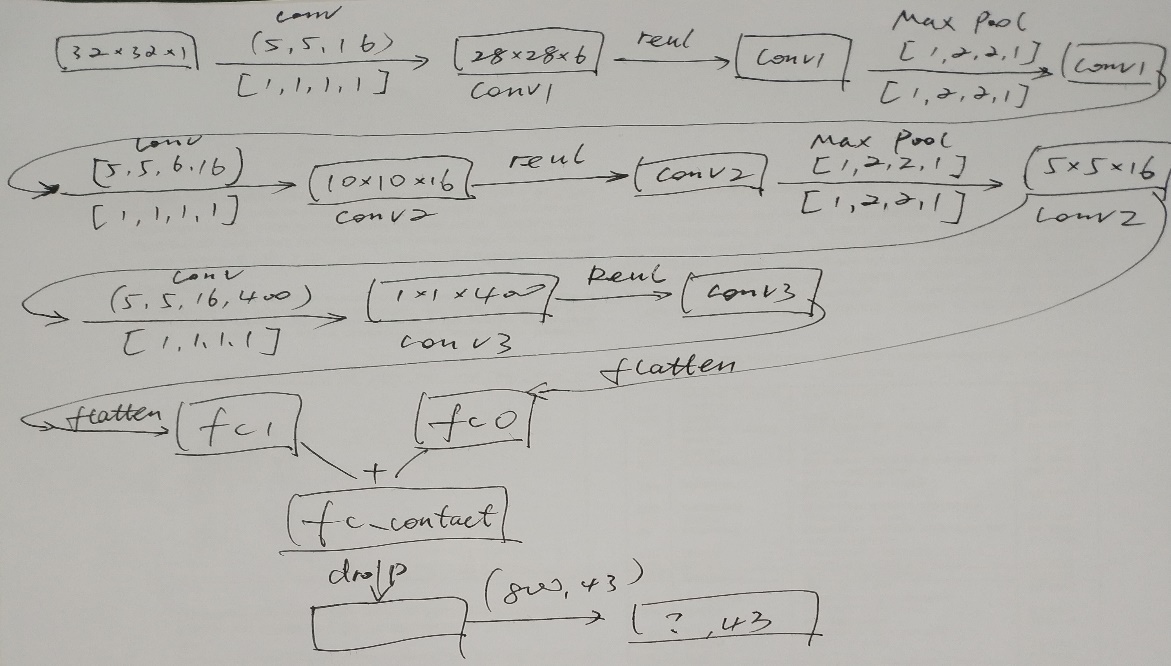
X\_test\_normalized = (X\_test\_gray - 128)/128

X\_train = X\_train\_normalized

X\_valid = X\_valid\_normalized

X\_test = X\_test\_normalized

## 2. Final model architecture looks like.



My final model consisted of the following layers:

|  |
| --- |
| conv1 shape: (?, 28, 28, 6)  conv1 after activation shape: (?, 28, 28, 6)  conv1 after pooling shape: (?, 14, 14, 6)  conv2 shape: (?, 10, 10, 16)  conv2 after activation shape: (?, 10, 10, 16)  conv2 after pooling shape: (?, 5, 5, 16)  fc0 shape: (?, 400)  conv3 shape: (?, 1, 1, 400)  conv3 after activation shape: (?, 1, 1, 400)  fc1 shape: (?, 400)  fc\_contact fc0+fc1 shape: (?, 800)  fc\_contact\_drop shape: (?, 800)  logits shape: (?, 43) |

|-----------------------|-----------------------------------------------|

| Layer | Description |

|:---------------------:|:---------------------------------------------:|

| Input | 32x32x1 |

| Convolution (5,5,1,6 | 1x1 stride, valid padding, outputs 28x28x6 |

| RELU | |

| Max pooling | 2x2 stride, outputs 14x14x6 |

| Convolution (5,5,6,16)| 1x1 stride, outputs 10x10x16 |

| RELU | |

| Max Pooling | 2x2 stride, outputs 5x5x16 |

| Conv (5,5,16,400) | 1x1 stride, outputs 1x1x400 |

| Flatten | fc0, fc1 |

| contact | fc\_contact |

| Drop | |

| Full connect | logits |

| | |

| | |

| | |

|-----------------------|-----------------------------------------------|

## 3. Trained model.

To train the model, I used ....

EPOCHS = 50

BATCH\_SIZE = 200

mu = 0

sigma = 0.1

rate = 0.001

|  |
| --- |
| C:\Users\stron\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\10C2DE18.tmp |
| EPOCH 1 ...  Validation Accuracy = 0.746  EPOCH 2 ...  Validation Accuracy = 0.836  EPOCH 3 ...  Validation Accuracy = 0.883  EPOCH 4 ...  Validation Accuracy = 0.903  EPOCH 5 ...  Validation Accuracy = 0.913  EPOCH 6 ...  Validation Accuracy = 0.922  EPOCH 7 ...  Validation Accuracy = 0.925  EPOCH 8 ...  Validation Accuracy = 0.928  EPOCH 9 ...  Validation Accuracy = 0.942  EPOCH 10 ...  Validation Accuracy = 0.939  EPOCH 11 ...  Validation Accuracy = 0.945  EPOCH 12 ...  Validation Accuracy = 0.936  EPOCH 13 ...  Validation Accuracy = 0.944  EPOCH 14 ...  Validation Accuracy = 0.947  EPOCH 15 ...  Validation Accuracy = 0.944  EPOCH 16 ...  Validation Accuracy = 0.939  EPOCH 17 ...  Validation Accuracy = 0.951  EPOCH 18 ...  Validation Accuracy = 0.943  EPOCH 19 ...  Validation Accuracy = 0.953  EPOCH 20 ...  Validation Accuracy = 0.949  EPOCH 21 ...  Validation Accuracy = 0.952  EPOCH 22 ...  Validation Accuracy = 0.953  EPOCH 23 ...  Validation Accuracy = 0.949  EPOCH 24 ...  Validation Accuracy = 0.957  EPOCH 25 ...  Validation Accuracy = 0.959  EPOCH 26 ...  Validation Accuracy = 0.958  EPOCH 27 ...  Validation Accuracy = 0.959  EPOCH 28 ...  Validation Accuracy = 0.954  EPOCH 29 ...  Validation Accuracy = 0.952  EPOCH 30 ...  Validation Accuracy = 0.960  EPOCH 31 ...  Validation Accuracy = 0.960  EPOCH 32 ...  Validation Accuracy = 0.963  EPOCH 33 ...  Validation Accuracy = 0.952  EPOCH 34 ...  Validation Accuracy = 0.958  EPOCH 35 ...  Validation Accuracy = 0.960  EPOCH 36 ...  Validation Accuracy = 0.958  EPOCH 37 ...  Validation Accuracy = 0.955  EPOCH 38 ...  Validation Accuracy = 0.960  EPOCH 39 ...  Validation Accuracy = 0.949  EPOCH 40 ...  Validation Accuracy = 0.951  EPOCH 41 ...  Validation Accuracy = 0.955  EPOCH 42 ...  Validation Accuracy = 0.962  EPOCH 43 ...  Validation Accuracy = 0.963  EPOCH 44 ...  Validation Accuracy = 0.952  EPOCH 45 ...  Validation Accuracy = 0.954  EPOCH 46 ...  Validation Accuracy = 0.959  EPOCH 47 ...  Validation Accuracy = 0.960  EPOCH 48 ...  Validation Accuracy = 0.958  EPOCH 49 ...  Validation Accuracy = 0.961  EPOCH 50 ...  Validation Accuracy = 0.956 |

## 4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

My final model results were:

training set accuracy of:1.000

validation set accuracy of:0.956

test set accuracy of:0.941

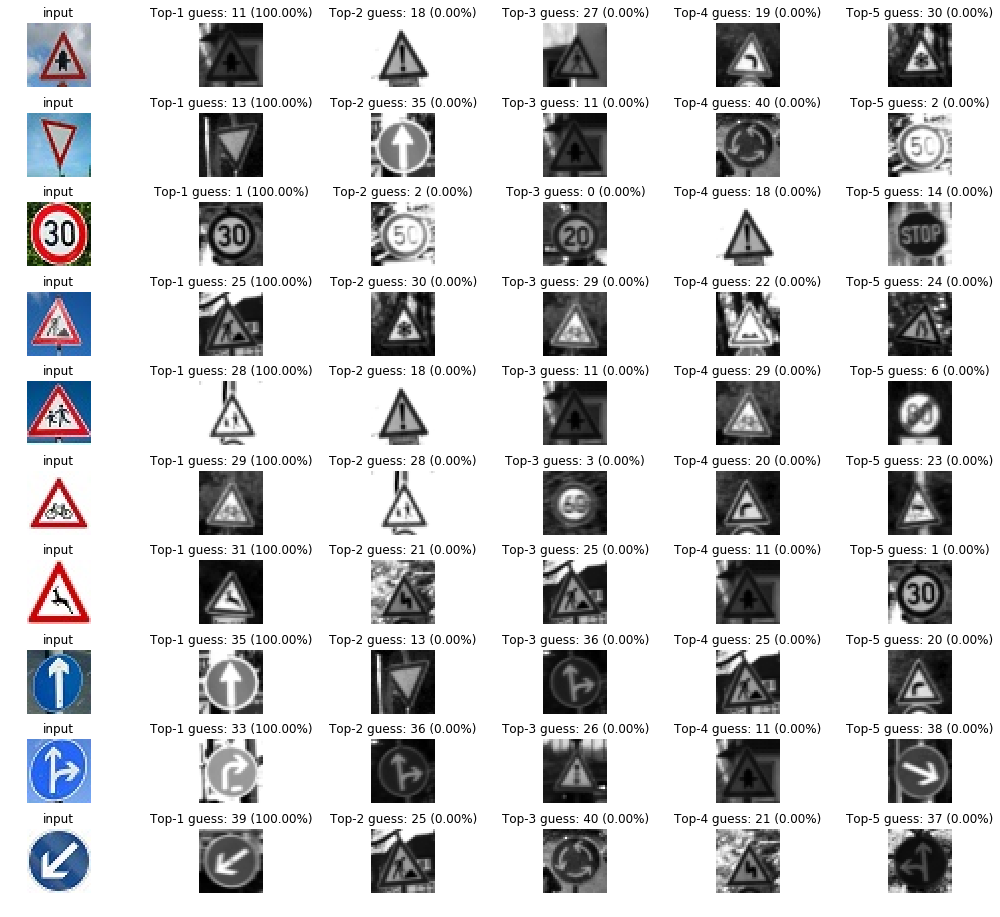
# Test a Model on New Images

## 1. Choose ten German traffic signs found on the web and provide them in the report.

Here are ten German traffic signs that I found on the web:



## 2. The model's predictions on these new traffic signs.



The model was able to correctly guess 9 of the 10 traffic signs, which gives an accuracy of 90%.