# Traffic Sign Recognition

## Writeup

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

Build a Traffic Sign Recognition Project

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
- Summarize the results with a written report

## Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

### Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

I'm sending you my code attached to the zip file.

## 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:

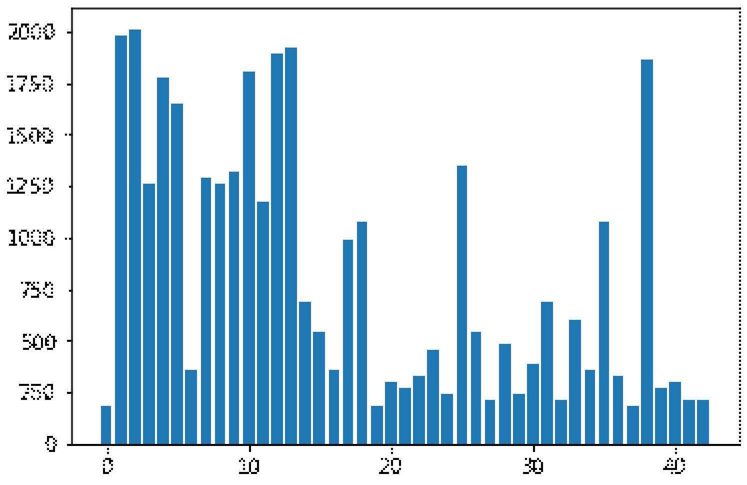
- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

### 2. Include an exploratory visualization of the dataset.

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

x_axis refers to the indel of the dataset.

y_axis is the number of datas included in each labels.



Shown below are the images randomly picked from the data set. You can see how's it seems like.

## Design and Test a Model Architecture

. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

As a first step, I decided not to convert the images to grayscale because The data of one actual color image is three-dimensional, and in order to learn it, it must be expressed in one dimension by flattening. In the process, since the size must be adjusted, there is a concern that data on the space may be lost. Due to this risk of data loss, problems arise in the learning and prediction processes. CNN method is designed to eliminate the risk of loss of spatial data and to extract the characteristics of the image to facilitate learning. CNN minimizes the loss of spatial data, derives features, and derives several features in one chapter. CNN proceeds before the deep learning process, and the value is added to the FC layer to proceed with learning and prediction. Lenet, a method that will be used later, uses the CNN method.I don't feel the need to convert to black and white.
I normalized the image data because It is pointed out that when the feature of the

dataset is very large, the prediction is not accurate due to the large effect of the feature with a large scale.
The average value of pixel before normal size was 82.677589037, but it could be verified that it decreased to -0.354082 after that.

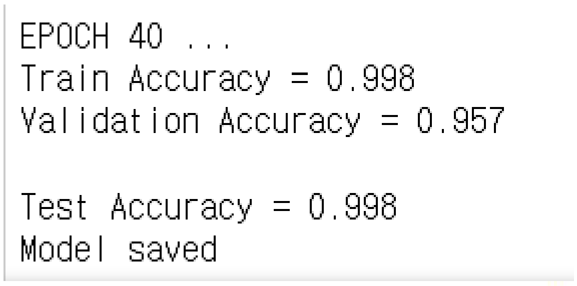
2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.
My final model consisted of the following layers:

| | |
|---|---|
| Input | 32x32x3 RGB image |
| Layer1 | Convolutional. Input = 32x32x3. Output = 28x28x6. |
| Activation | RELU |
| Max pooling | The output shape should be 14x14x6. |
| Layer2 | Convolutional. Input = 14x14x6. Output = 10x10x16. |
| Activation | RELU |
| Max pooling | The output shape should be 5x5x6. |
| Flatten | Flatten the output shape of the final pooling layer such that it's 1D instead of 3D. |
| Layer 3 | Fully Connected. shape(Input = 400. Output = 120.) |
| Layer 4 | Fully Connected. Input = 120. Output = 84. |
| Layer 5 | Fully Connected. Input = 84. Output = 10. |
| Softmax | + loss_operation |
| AdamOptimizer | A little more sophisticated than the stochastic gradient descent |

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.
I set the same parameter as what I learned in class. Initially, epochs was set to 10. However, the accuracy did not reach the desired value, so only epochs were raised to 40.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.
My final model results were:

training set accuracy of 0.998
validation set accuracy of 0.957
test set accuracy of 0.998
What architecture was chosen?
Architecture of Le Net



```
EPOCH 40 ...
Train Accuracy = 0.998
Validation Accuracy = 0.957

Test Accuracy = 0.998
Model saved
```

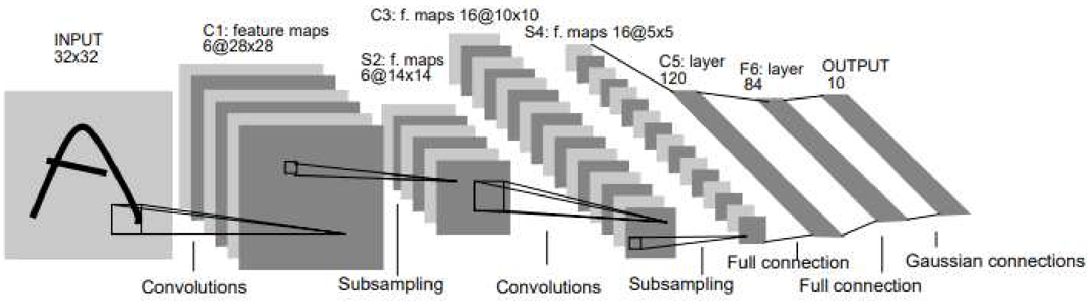


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

## Why did you believe it would be relevant to the traffic sign application?

LeNet-5 proposed a modern CNN structure that not only shows high performance using a complex network, but also repeats the combination of convolutional layers and pooling. Compared to MNIST, colored and relatively complex traffic signs had to be analyzed. This architecture was judged to be appropriate.

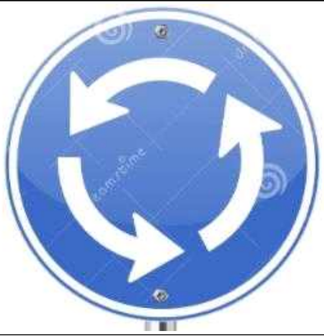
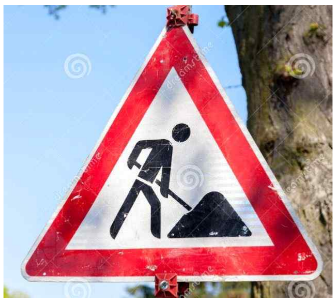
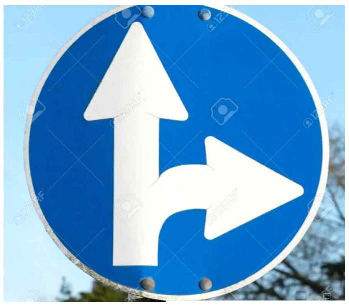
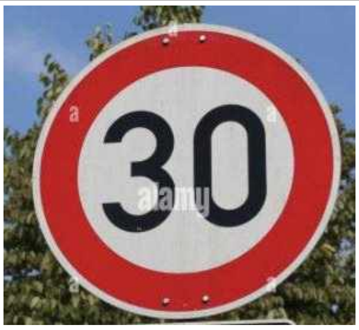
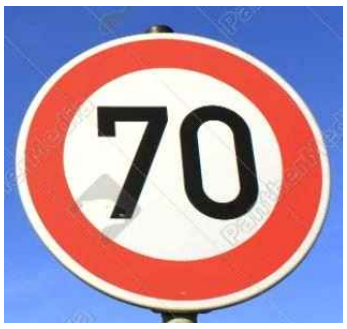
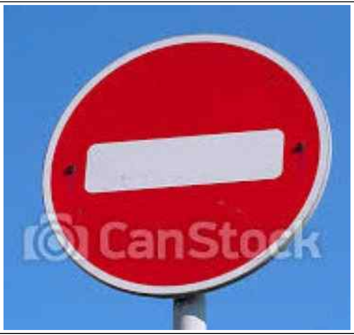
## How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

Looking at the results of the training presented above, it shows almost 100% accuracy. This proves that the use of LeNet is an appropriate choice.

## 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.

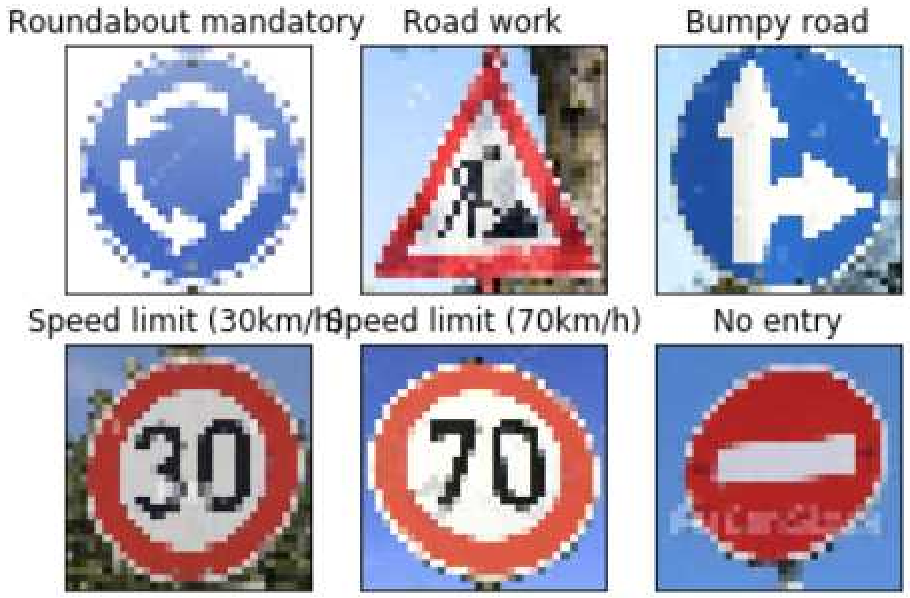
Here are six German traffic signs that I found on the web:

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

The first image will be difficult to classify. This is because the watermark may cause an error because it looks like the content on the sign. In addition, the second row's pictures are expected to have a low detection probability because they are looking in different directions without looking forward.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set
Here are the results of the prediction:

The pictures below are the results of detecting the sign. Except for the third picture, it was accurately detected.



| Image | Prediction |
|---|---|
| Roundabout mandatory | Roundabout mandatory |
| Road work | Road work |
| Go straight or right | Bumpy road |
| Speed limit(30km/h) | Speed limit(30km/h) |
| Speed limit(70km/h) | Speed limit(70km/h) |
| No entry | No entry |

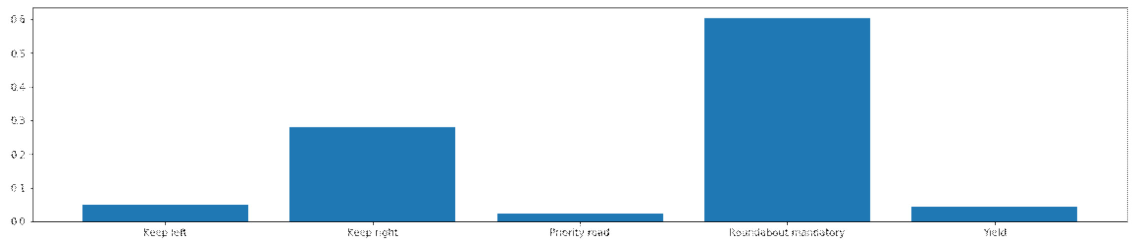
The model was able to correctly guess 5 of the 6 traffic signs, which gives an accuracy of 83.33%.

Accuracy: 83.33333333333334 %

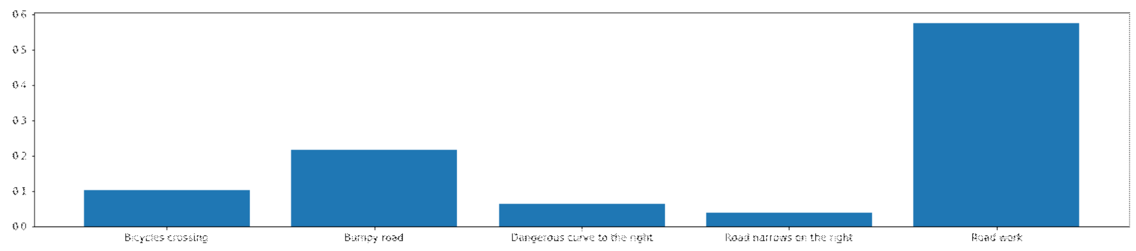
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability
The code for making predictions on my final model is located in the 15th cell of the Ipython notebook.

You can see from the figure below how the results of each picture were derived.
−934.93902588]]

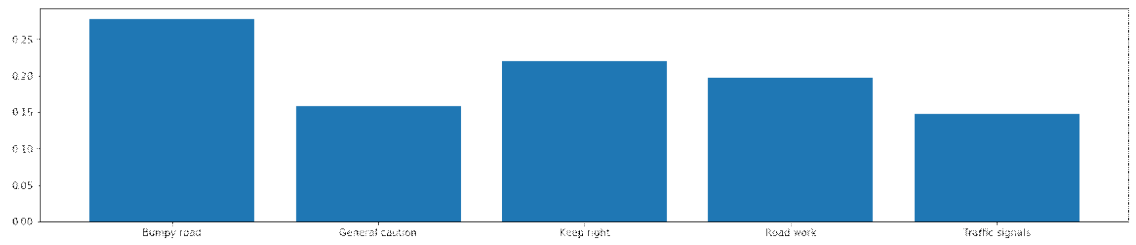
Sum_of_probability 1.0



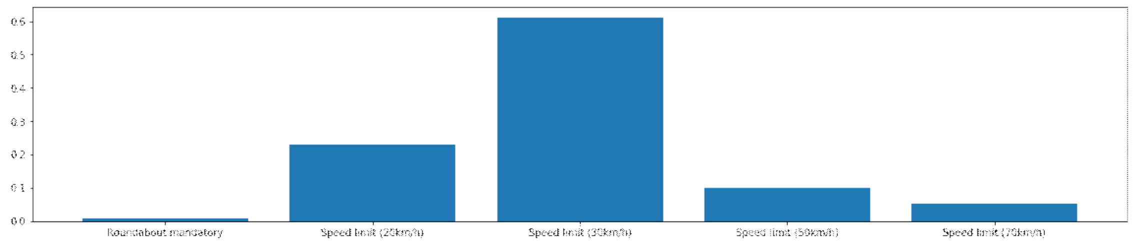
Sum_of_probability 1.0

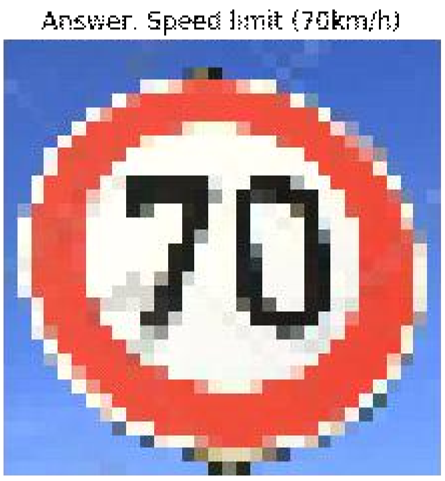


Sum_of_probability 1.0

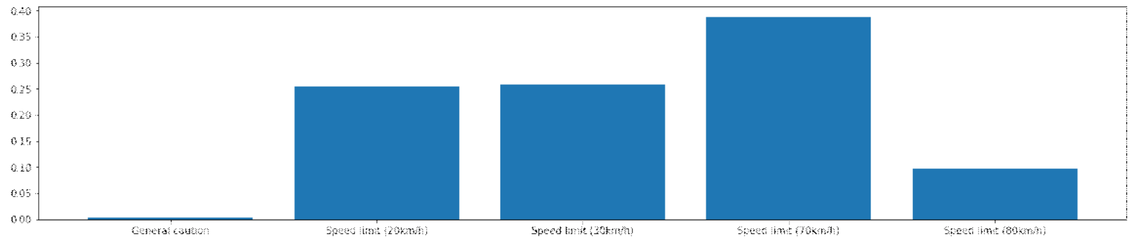


Sum_of_probability 1.0

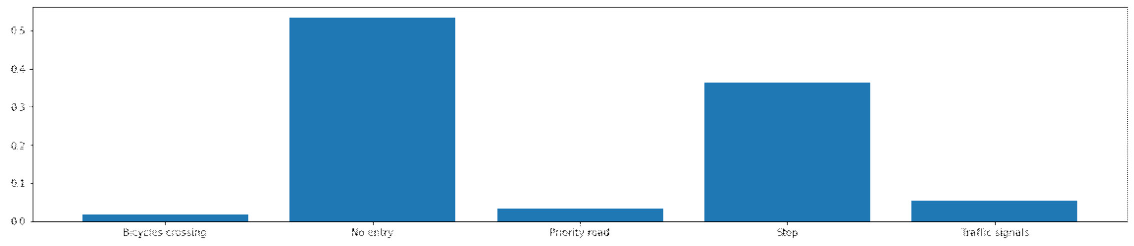




Sum_of_probability 1.0



Sum_of_probability 1.0



0.5
0.4
0.3
0.2
0.1
0.0
Bicycles crossing
No entry
Priority road
Stop
Traffic signals