Traffic Sign Recognition Writeup

Student Name: Ying Li
Email Address: liying2905@gmail.com

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.

You're reading it!

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 hard coding results manually.

I used the basic numpy library (len and shape) to calculate summary statistics of the traffic signs data set:

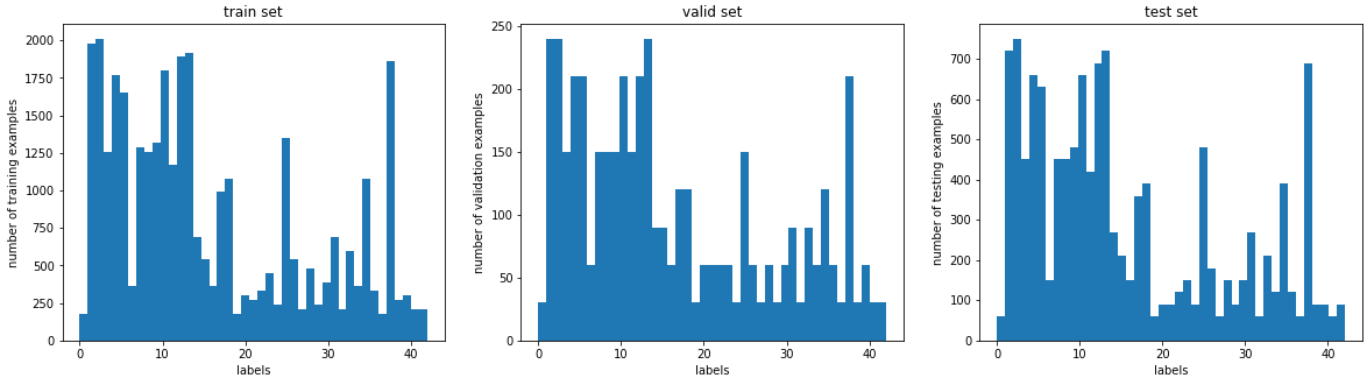
The size of training set is 34799
The size of the validation set is 4410
The size of test set is 12630
The shape of a traffic sign image is (32, 32, 3)
The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set.

Example of images:

Classification distribution:

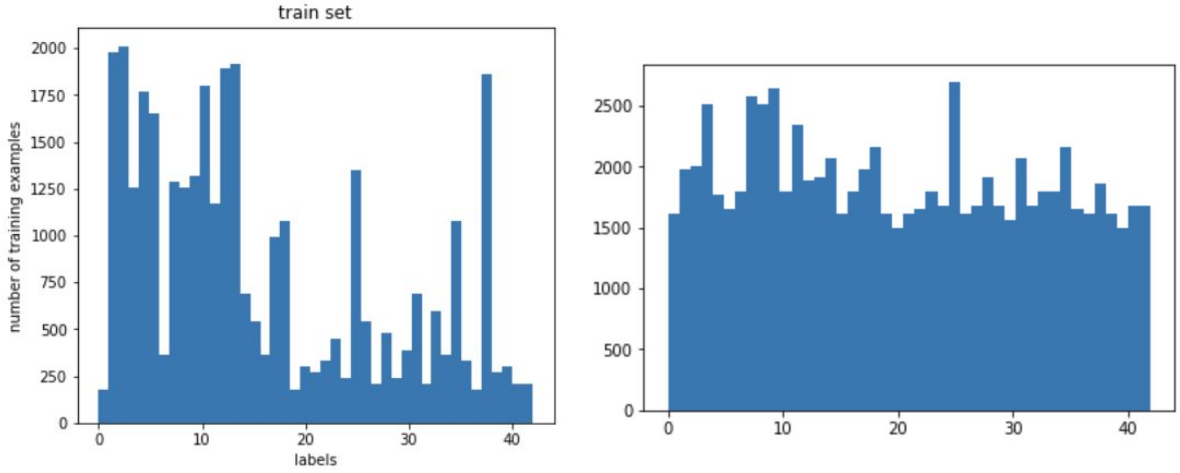


Design and Test a Model Architecture

1. 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.)

First, I augmented the training data set in order to make each class contain at least 1500 data. Because I found the number of data in each traffic sign class is seriously unbalanced and this could affect the training results. The augmentation approach I used is a simple duplication of its existing data set, until the total number reaches 1500.

Before augmentation: After augmentation:



Second, I normalized all the input data in the training set, validation set, and testing set. Because the range of each pixel data is [0, 255]. I use the formula (pixel - 128) / 128 to normalize the data.

Then, I decrease the channel of the image from 3 to 1. Because the training process shows that 1 channel input is more effective than the 3 channel input image. The approach I used is a simple mean operation on 3 channels.

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:

Layer	Description	Input	Output

Input	32x32x1 Gray image		
Convolution 5x5	1x1 stride, VALID padding	32x32x1	28x28x6
RELU			
Max pooling	2x2 kernel, 2x2 stride	28x28x6	14x14x6
Convolution 5x5	1x1 stride, VALID padding	14x14x6	10x10x6
RELU			
Max pooling	2x2 kernel, 2x2 stride	10x10x6	5x5x6
Fully connected	Input: the flatten of conv1 and conv2 drop out rate 0.5	14x14x6 + 5x5x6	480
Fully connected	drop out rate 0.5	480	84
Fully connected	drop out rate 0.5	84	43
Output logits	[0, 42] logits representing 43 signs		

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.

To train the model, I used an Adam Optimizer. The batch size is 128; the number of epochs is 90, the learning rate is 0.001. I used a dropout approach to alleviate the overfitting problem, with a keep probability of 0.5.

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 100%
- validation set accuracy of 97.1%
- test set accuracy of 95.7%

code position:
training pipeline: code snippet 12
evaluation pipeline: code snippet 13
training and validation process: code snippet 14
testing process: code snippet 15

Here's the internal results for every 10 epochs:
EPOCH 0 ...
Training Accuracy = 0.907
Validation Accuracy = 0.814

EPOCH 10 ...
Training Accuracy = 0.996
Validation Accuracy = 0.962

EPOCH 20 ...
Training Accuracy = 0.999
Validation Accuracy = 0.960

EPOCH 30 ...
Training Accuracy = 0.999
Validation Accuracy = 0.965

EPOCH 40 ...
Training Accuracy = 1.000
Validation Accuracy = 0.970

EPOCH 50 ...
Training Accuracy = 1.000
Validation Accuracy = 0.968

EPOCH 60 ...
Training Accuracy = 1.000
Validation Accuracy = 0.969

EPOCH 70 ...
Training Accuracy = 1.000
Validation Accuracy = 0.970

EPOCH 80 ...
Training Accuracy = 1.000
Validation Accuracy = 0.971

An iterative approach was chosen to find the solution:

- What was the first architecture that was tried and why was it chosen?
  I tried the LeNet-5 architecture at the end of the CNN lesson.
- What were some problems with the initial architecture?
  It gained an accuracy of 94.5% in the validation set. I feel there's room for improvement.
- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common

justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

I read the paper *Traffic Sign Recognition with Multi-Scale Convolutional Networks* by Pierre Sermanet and Yann LeCun and borrowed some ideas from there:
1) Results from all convolutional layers are fed into the fully connected layer, instead of only feeding the result of the last conv layer. This approach takes consideration of all features, not just high-level features.
2) The input images keep only 1 channel instead of 3 channels. Opposed to our common sense, 1 channel images make the training accuracy higher

So the accuracy increased to more than 95% on the validation set. To further improve it. I use a dropout approach to lessen the overfitting problem of the model, with a keep probability of 0.5. By now the accuracy is raised to around 97% on the validation set.

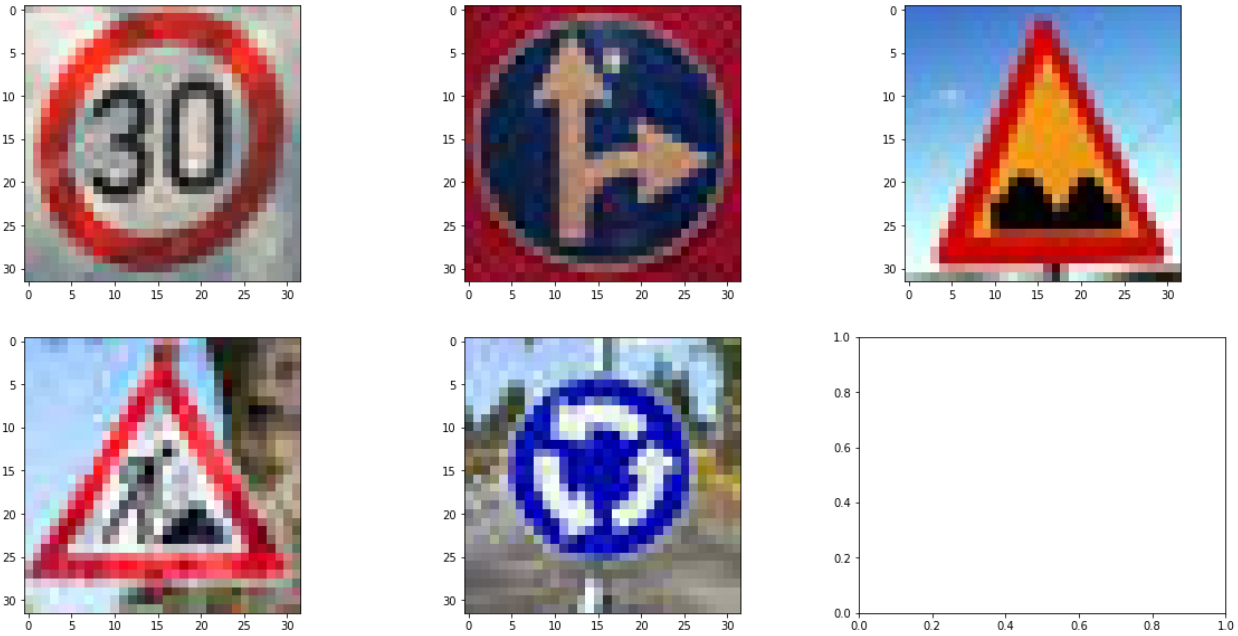
- Which parameters were tuned? How were they adjusted and why?
  I first tuned the learning rate. I use 0.1 as an initial learning rate. The model accuracy does not improve during the training process. I realized that it doesn't converge due to the large learning rate. So I gradually decrease the learning rate, and find 0.001 is an optimum value to make the model learn fast while still keeping the model converging.
  I also tuned the number of nodes in the fully connected layers. I found that when gradually decreasing the number of nodes layer by layer, the effect is better.
  Then I tuned the epochs number, after 90 epochs, the model doesn't improve on accuracy. So I set the epochs as 90.

- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?
  1) The biggest improvement comes from using the results of all conv layers as the input of the fully connected layer. This approach takes consideration of all features, not just high-level features, thus increasing the accuracy significantly.
  2) Using pooling layers is also important because they make the model robust to small shifts and geometric distortions.
  3) Dropout further alleviates the overfitting problems.

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 five German traffic signs that I found on the web:



The first image might be difficult to classify because it is seriously distorted: the circle boundary of the sign becomes an oval.

The second image might be difficult to classify because its colors are dark and the contrast between different colors is small.

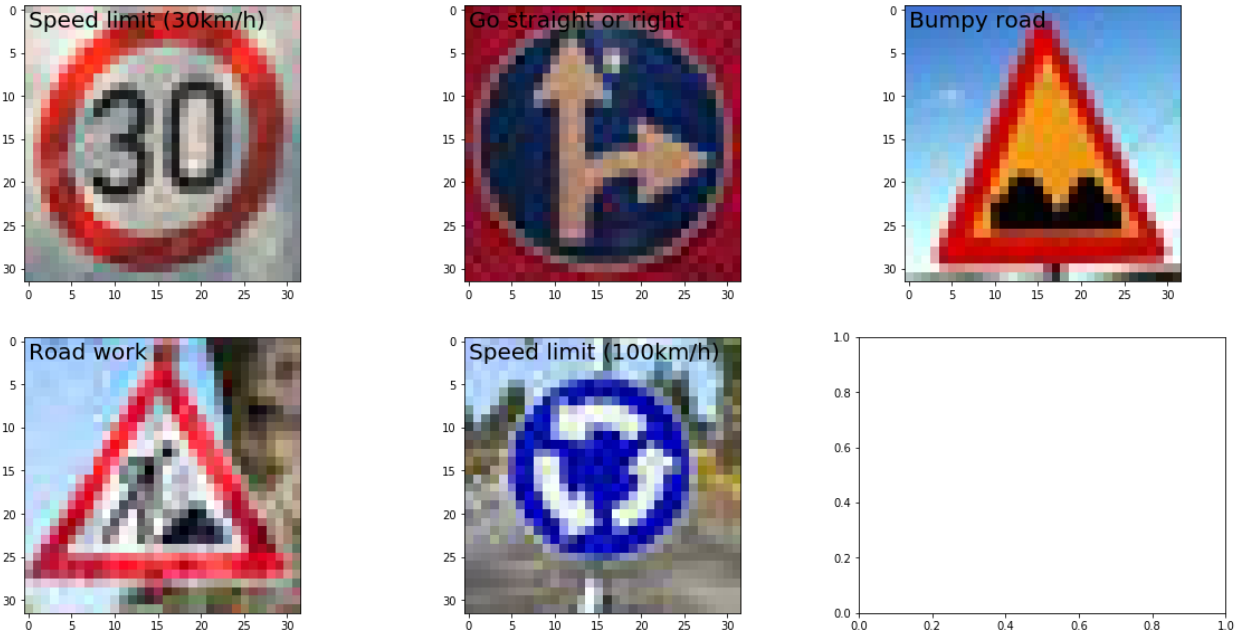
The third image might be difficult to classify because it is slightly stretched.

The fourth image might be difficult to classify because its background is complicated and there are shades on the sign.

The fifth image should be relatively easy to classify.

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 (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:



Accuracy of model on new images is 80%
(To my surprise, the "easy" fifth image is the only one to be wrongly classified)

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. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

Looking at the top 5 softmax probabilities for each image:
Values:

```
[[ 1.00000000e+00   6.95386413e-36   0.00000000e+00   0.00000000e+00
   0.00000000e+00]
 [ 1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00]
 [ 1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00]
 [ 1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00]
 [ 1.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
   0.00000000e+00]]
```

Indices:

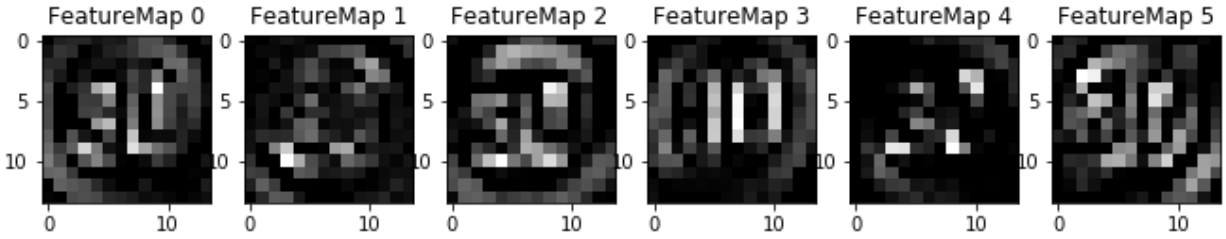
```
[[ 1  0  2  3  4]
 [36  0  1  2  3]
 [22  0  1  2  3]
 [25  0  1  2  3]
 [ 7  0  1  2  3]]
```

The model is very certain on its predictions. Each softmax probability for its winning class is almost 100% sure.

Visualizing the Neural Network

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

This is the conv1 layer of the network. It can be seen that this layer extracts the simple line and shape features of the images



This is the conv2 layer of the network. It can be seen that this layer extracts more advanced features such as the overall arrangement of the shapes. But they are indecipherable to me.