Traffic Sign Recognition

Writeup

Files submitted:

- **Traffic_Sign_Classifier.html**
- **Traffic_Sign_Classifier.ipynb**
- **Writeup.md**
- **Writeup.pdf**

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.

You’re reading it!

Data Set Summary & Exploration

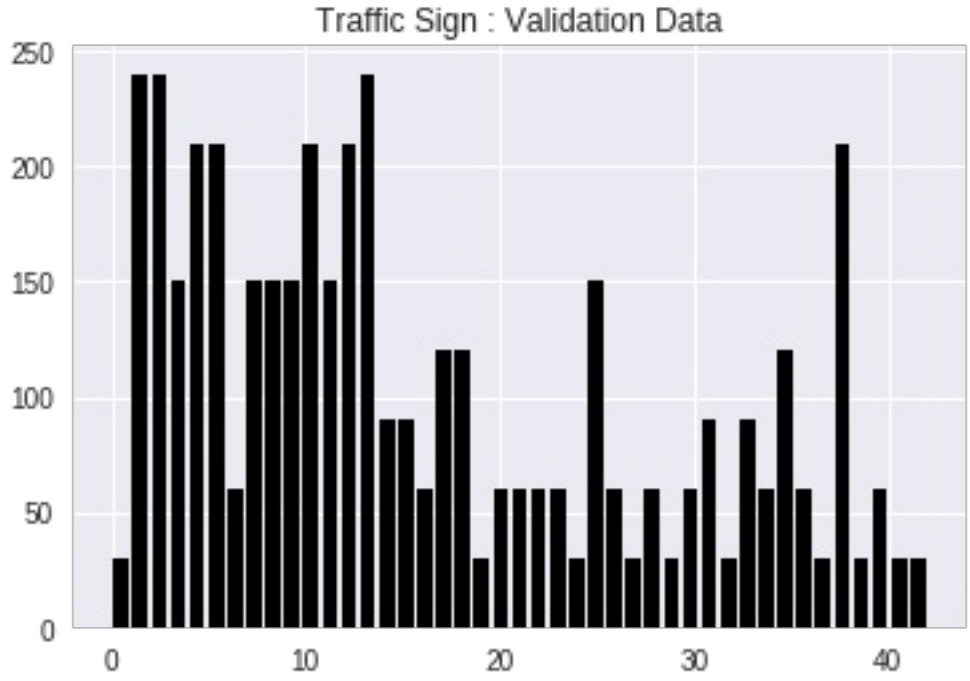
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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 distributed for various datasets.



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.)

As a part of pre-procesing the image dataset, I normalized the images in the method **pre_process(img)**.

It takes individual images from the whole dataset and normalizes it in order to attain **0 mean** in the dataset.

The function is as follows:

```
def pre_process(img):
  return (img - 128.)/128.
```

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	32x32x3 RGB image
Convolution	1x1 stride, valid padding, outputs 28x28x6
RELU	Activation function
Dropout	Helps in reducing the over-fitting
Max Pooling	Input = 28x28x6,same padding output = 14x14x6
Dropout	Helps in reducing the over-fitting
Convolution	Input = 14x14x6, vlid paading, output = 10x10x16
RELU	Activation function
Dropout	Helps in reducing the over-fitting
Max Pooling	Input = 10x10x16,same padding, output = 5x5x16
Dropout	Helps in reducing the over-fitting
Flatten	Input = 5x5x16, output = 400
Fully Connected Layer	Input = 400, output = 120
RELU	Activation function
Dropout	Helps in reducing the over-fitting

Fully Connected Layer	Input = 120, output = 84
RELU	Activation function
Dropout	Helps in reducing the over-fitting
Fully Connected Layer	Input = 84, output = 43

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**

Furthermore, various other hyper-parameters are as follows:

- **Learning Rate** = 0.001
- **Batch Size** = 128
- **Number of Epochs** = 50

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 **1.000**
- validation set accuracy of **0.954**
- test set accuracy of **0.949**

The following steps were taken while implementing the solution:

- First, I used the **LeNet** architecture that was taught to me in the previous lessons of the nanodegree program.
    - It had **10 EPOCHS** and a **batch size** of **128**.
    - I was able to attain an accuracy of around **85%** after training was completed.
- Then, I moved onto feeding in the pre_processed image data that had 0 mean value.
    - The accuracy increased but not by much.
    - But, I was seeing an upward trend in the validation accuracy. So, I moved onto the next step.

- I increased the number of epochs to **50**.
    - The accuracy did increase but it plateaued at around **90%**.
- Then, I increased the number of EPOCHS even more.
    - Number of EPOCHS = 150
    - The accuracy remained same i.e., 90%.
    - The model resulted in overfitting and thus the validtion accuracy didn't increase.
- After this, I tried lowering the learning rate so as to achieve higher accuracy.
    - But, the results didn't vary much.
- Finally, I introduced dropouts into the model in order to prevent the over-fitting.
    - I first introduced dropout after the first **convolutional layer**.
    - Kkeeping the numebr of EPOCHS as same the accuracy for the validation set increased as the training accuracy took longer to plateau this time.
- So, I introduced more dropouts to the model.
    - Appartenly, the model achieves the accuracy of **0.93** for the validation set at around 30 EPOCHS.
    - But, I wanted to attain maximum accuracy for the validation set hence, I've kept the EPOCHS as 50 and BATCH SIZE as 128 for the final model architecture.
- The final model was quite different from LeNet architecture also in terms of input and output.
    - The LeNet architecture had input images with 1 channel

while traffic sign images are 3 channel.
- Also, the output of the LeNet architecture had 9 nodes, whereas the current model has 43 output nodes.

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:

This above image might be difficult to classify because the traffic sign represented above has a triangular boundary which makes it similar to other traffic signs of the same shape. Such as the **General Caution** sign below.

The image has a higher resolution and size than what my model accpets, this image needs to be re-shaped to be fit into the classification model.

This above image might be difficult to classify because the traffic sign represented above has a **square boundary** which makes it similar to other traffic signs of the same shape.

This above image might be difficult to classify because the circle is a bit skewed in the image.

This above image might be difficult to classify because the traffic sign represented above has a **triangular boundary** which makes it similar to other traffic signs of the same shape. Such as the **Right of way** sign above.
Furthermore, the sign is a bit bent to the sideways making its classification difficult. Also, it has a black patch in the traffic sign boundary, which might result in false classification.

This above image might be difficult to classify because the traffic sign represented above has a **circular boundary** which makes it similar to other traffic signs of the same shape. Such as the **No Vehicles** sign above.
Furthermore, the sign has smudges on it which might result in a non-accurate classification.

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:

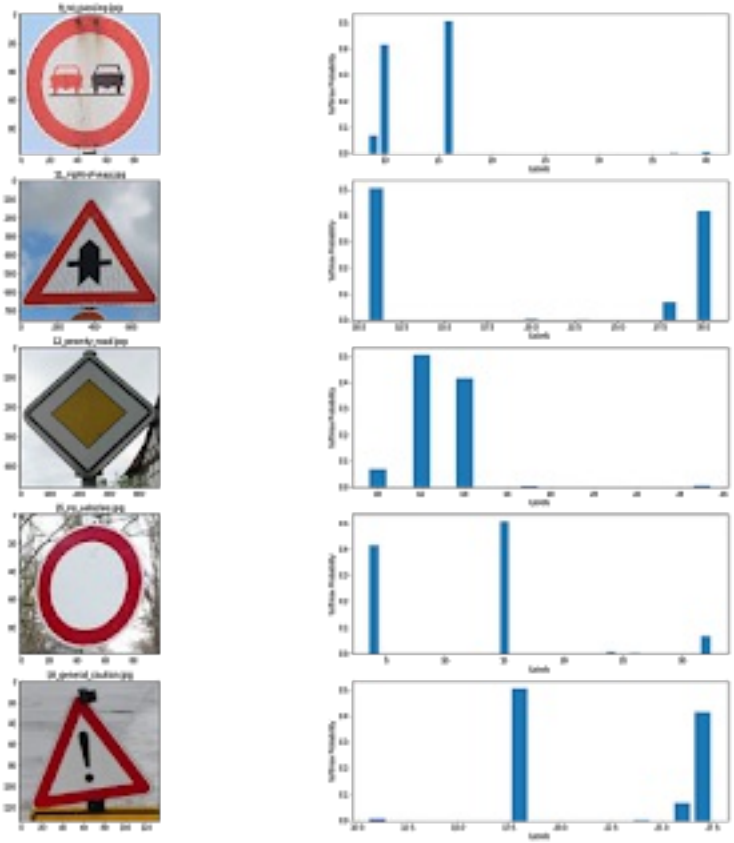
Image	Prediction
No passing	Stop
Right of way	Right of way
Priority road	Priority road
No vehicles	No vehicles
General Caution	General Caution

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of **0.949**

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)

The code for making predictions on my final model is located in the 23rd cell of the Ipython notebook.



(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

1. Discuss the visual output of your trained network’s feature maps. What characteristics did the neural network

use to make classifications?