writeup_template

Traffic Sign Recognition

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! and here is a link to my project code

Data Set Summary & Exploration

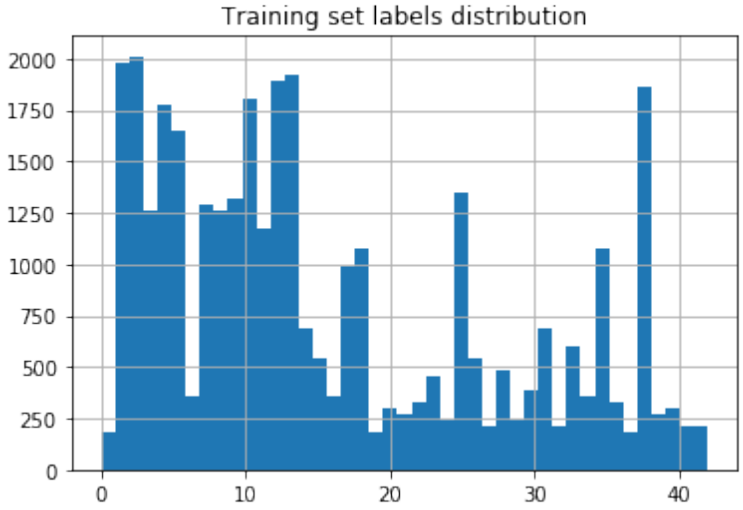
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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 32x32
- 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 distribution of the labels.



1. Comparison between the training, validation and test sets shows that the distribution of labels is identical across them.
2. Comparing number of images for each label, it is clear that some labels have significantly more samples than others. This would mean that the network is better at predicting those traffic signs.

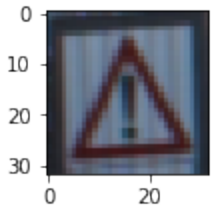
Design and Test a Model Architecture

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.)

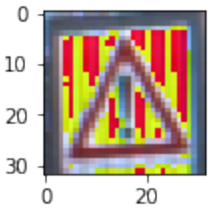
As a first step, I decided to normalize the data. This was done using the method prescribed in the lectures for RGB images - **(pixel - 128)/128** .Normalization results in images have zero mean and equal variance.

Here is an image before and after normalization:

Before :

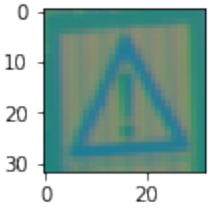


After :



As a next step, I converted the images to YUV images. This is similar to preprocessing described in a published baseline model for this problem.

Here is the above image after conversion to YUV:



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
Layer 1 : Convolution 5x5	1x1 stride, valid padding, output = 28x28x10
RELU	
Max pooling	2x2 stride, output = 14x14x10
Layer 2 : Convolution 5x5	1x1 stride, valid padding, output = 10x10x20
RELU	
Max pooling	2x2 stride, output = 5x5x20
Flatten	output = 500
Layer 3 : Fully connected	output = 500
Dropout	prob = 0.75
Layer 4 : Fully connected	output = 200
Dropout	prob = 0.75
Layer 5 : Fully connected(logits)	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 with batch size = 128.

I trained the model for 50 epochs using a learning rate of 1e-3.

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 99.6%
- validation set accuracy of 93.5%
- test set accuracy of 92.7%

I decided to go with an iterative approach to developnig the model architecture.

- What was the first architecture that was tried and why was it chosen?
    - LeNet-5 architecture was a great starting point
    - Decent performance out of the box @ 89% validation accuracy.
    - The model was not too complicated and gave me room to experiment with ideas from the lectures.
    - It was not as computationally intensive as a very large network. I was able to train quickly to validate ideas.
- What were some problems with the initial architecture?
    - Did not generalize to new images - Not performing as well as training data on validation data or new images from the web.
- 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.
    - Adjusted depth of both convolutional layers.
    - Adjusted size of the fully connected layers.
    - Added dropout
    - Added L2 regularization
- Which parameters were tuned? How were they adjusted and why?
    - Number of epochs was increased to ensure that model converges.
    - Regularization factor : tuned to 1e-6
    - Dropout probabilitiy : Set to 0.75

- 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?

    - Convolutional layers work well with this type of problem. Successive convolutional layers can capture increasingly complex details of the images.
    - Dropout layer helps reduce the extent of overfitting.
    - Penalizing the network based on L2 regularization also helps reduce overfitting.

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 some German traffic signs that I found on the web:

This image is interesting for 2 reasons. It has text as well as the traffic sign. Also, there are watermarks on the image which might confuse the learning algorithm. However, both are typical problems that a robust learning algorithm should be able to handle.

This image is interesting because it is a combination image. Theoretically, a given enough layers, a CNN should be able to capture sub-images like this example.


These 2 images are interesting because the image has been captured at an angle. The plane of the sign is not directly exactly parallel to that of the camera.

These are 3 normal images of traffic signs. There is nothing very unique about these and I expect the network to successfully classify these.

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 (Some comments are based on the visualization of softmax probabilities at the end of the notebook):

Image	Prediction	Comment
General Caution	No entry	The classification failed but 2 of the top 3 guesses also text below the signs. So, it did succeed in predicting *something* about the test image.
Wild animals crossing	Wild animals crossing	2nd best guess is a bit funny. The image of the animal and a "double curve" are a bit similar.
Right of way at next intersection	Right of way at next intersection	2nd guess is a bit funny in this case as well. It is an image of a pedestrian crossing a road. Blurred enough, those 2 traffic signs are a bit similar.
Road work	Road work	Succesfully classified!
Children crossing	Ahead only	Failed to classify this. The only thing common between the image and the guess appears to be the color - Blue
30 km/h	30 km/h	Succesfully classified!
Stop	Stop	Succesfully classified!

The model was able to correctly guess 5 of the 7 traffic signs, which gives an accuracy of 71.4%.

This is much lower than the test set accuracy of 92.7%. But I think that that is a bit like comparing apples and oranges. The network seemed to do well on "standard" images like the one in training set. The one it failed on are a bit complicated (as described in above sections).

That said, this hints at some future improvements -

- Train model to correctly predict images of "compound" traffic signs. By that, I mean traffic signs which have additional text or those that are actually 2 signs combined into 1.

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)

Top 5 softmax probabilities for all 7 test images :

Guess 1	Guess 2	Guess 3	Guess 4	Guess 5
0.9981835485	0.0018049686	0.0000111785	0.0000002769	0.0000000172
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0

1	0	0	0	0
0.9999991655	0.0000004886	0.0000003902	0	0
1	0	0	0	0

Towards the end of my ipython notebook, there is a matrix of images displaying original image, top 3 guesses and probability of each guess.

Here is a copy of that image matrix :


input
Guess 1:(99.81835%)
Guess 2:(0.18050%)
Guess 3:(0.00112%)
input
Guess 1:(100.00000%)
Guess 2:(0.00000%)
Guess 3:(0.00000%)
input
Guess 1:(100.00000%)
Guess 2:(0.00000%)
Guess 3:(0.00000%)
input
Guess 1:(100.00000%)
Guess 2:(0.00000%)
Guess 3:(0.00000%)
input
Guess 1:(100.00000%)
Guess 2:(0.00000%)
Guess 3:(0.00000%)
input
Guess 1:(99.99992%)
Guess 2:(0.00005%)
Guess 3:(0.00004%)
input
Guess 1:(100.00000%)
Guess 2:(0.00000%)
Guess 3:(0.00000%)