Traffic Sign Recognition

Writeup

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

Writeup / README

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 python 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.

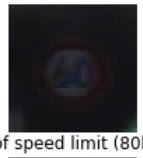
Here is an exploratory visualization of the data set. Shown below there is the list of classes and one of the images. Further below that is a bar chart displaying how many samples of data per sign contained in the data set.

Speed limit (20km/h)

Speed limit (30km/h)

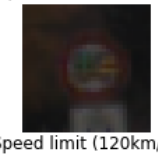
Speed limit (50km/h)

Speed limit (60km/h)

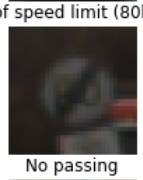


Speed limit (70km/h)

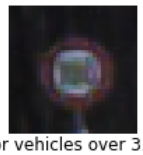
Speed limit (80km/h)



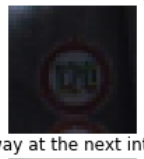
End of speed limit (80km/h)



Speed limit (100km/h)

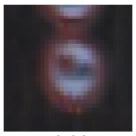


Speed limit (120km/h)



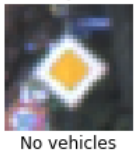
No passing

No passing for vehicles over 3.5 metric tons



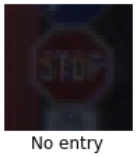
Right-of-way at the next intersection

Priority road



Yield

Stop

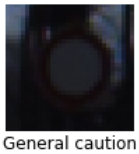


No vehicles

Vehicles over 3.5 metric tons prohibited

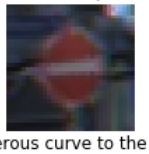
No entry

General caution



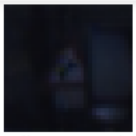
Dangerous curve to the left

Dangerous curve to the right



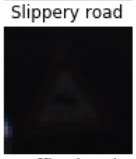

Double curve


Bumpy road


Slippery road


Road narrows on the right


Road work


Traffic signals


Pedestrians


Children crossing


Bicycles crossing

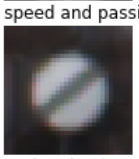

Beware of ice/snow

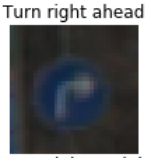

Wild animals crossing


End of all speed and passing limits


Turn right ahead


Turn left ahead


Ahead only

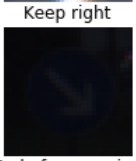

Go straight or right


Go straight or left

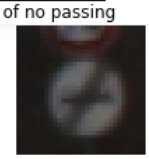

Keep right

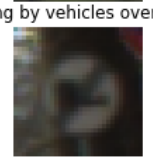

Keep left

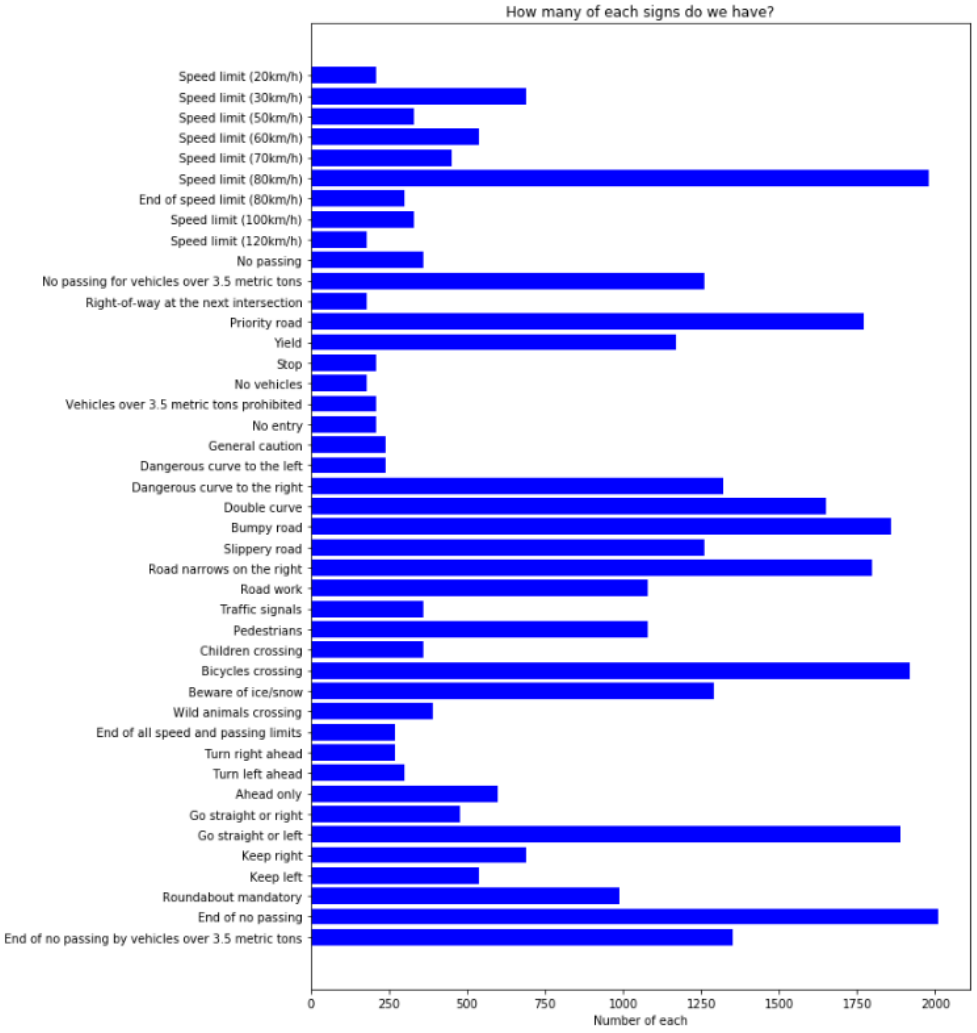

Roundabout mandatory


End of no passing


End of no passing by vehicles over 3.5 metric tons







Design and Test a Model Architecture

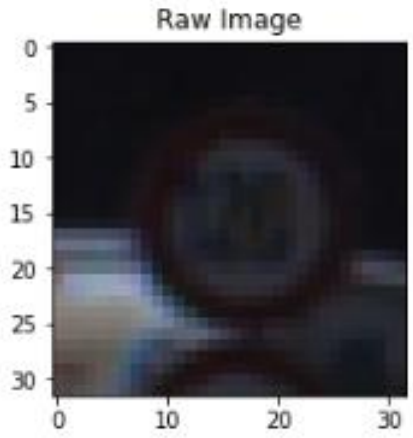
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described 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.)

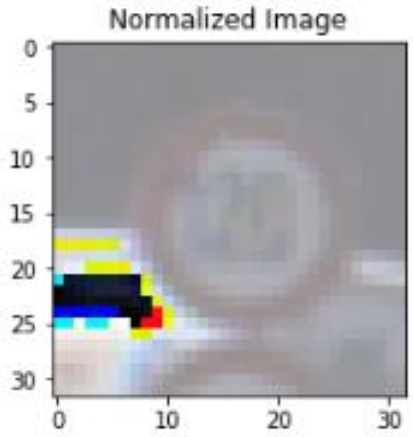
As a first step, I decided to normalize the image data. I normalized the existing data from values ranging from 0-255 to -0.5 to 0.5. This was based on the normalization algorithm found on Wikipedia. It translates the existing range to a new range. Normalization is important because gradient descent and other classifiers calculate the distance between two points. If a feature has a wide range then the classifier will be controlled disproportionately by this feature.

Here is the numpy mean of training, validation, and test set before and after normalization.

```
Training Set Mean = 82.656170745
Validation Set Mean = 83.5564273756
Test Set Mean = 82.1484603612

Training Set Mean Normalized = -0.175858153941
Validation Set Mean Normalized = -0.172327735782
Test Set Mean Normalized = -0.177849175054
```





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.

The final model was only a minor modification of the existing LeNet architecture.

Shown below is the LeNet architecture shown in the course:

Layer	Description
Input	32x32x1 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	400
Fully connected	400 to 120, mean = 0, stddev = sigma
RELU	
Fully connected	120 to 84, mean = 0, stddev = sigma
RELU	
Fully connected	84 to 10, mean = 0, stddev = sigma

Layer	Description
RELU	
Softmax Cross Entropy	

For the Final model I changed the input, first convolution layer shape, last fully connected layer, and added a dropout before classification. The input changed from one to the three-layer depth from the input image. It could have been possible to pre-process this but I pursued the RGB route. The first convolution layer was changed from 5x5 to 7x7. The last fully connected layer output was increased to 43 to account for the additional classes. An additional dropout was added after the fully connected layer to help generalize the results.

This is the final model architecture:

Layer	Description
Input	32x32x3 RGB image
Convolution 7x7	1x1 stride, valid padding, outputs 26x26x6
RELU	
Max pooling	2x2 stride, outputs 13x13x6
Convolution 5x5	1x1 stride, valid padding, outputs 9x9x16
RELU	
Max pooling	2x2 stride, outputs 4x4x16
Flatten	256

Layer	Description
Fully connected	256 to 120, mean = 0, stddev = sigma
RELU	
Fully connected	120 to 84, mean = 0, stddev = sigma
RELU	
Fully connected	84 to 43, mean = 0, stddev = sigma
RELU	
Dropout	0.5 percentage
Softmax Cross Entropy	

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 instead of a stochastic gradient descent. There are two advantages detailed in this article. First, the algorithm maintains a per-parameter learning rate in contrast to a single learning rate. Second, the algorithm updates these learning rates based on the recent gradients for the weights.

Epochs – 20 (increase from 10)

The epoch value was increased because testing revealed validation test set accuracy below 90. Increasing the epoch would allow more time to train increasing accuracy but at the cost of added time to train.

Batch Size – 128

Learning Rate – 0.001

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.

Overall, this showed that the algorithm required more training to reach the required accuracy. A more meaningful improvement would have bee

My final model results were:

- training set accuracy of 0.98
- validation set accuracy of 0.947
- test set accuracy of 0.907

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?

  The first architecture was Lenet-5. It was chosen because it was provided earlier.

- What were some problems with the initial architecture?

  The first architecture was not able to achieve a validation set accuracy of >0.93.

- 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. Which parameters were tuned? How were they adjusted and why?

  To get from the original LeNet-5 to the final adjusted model a lot of tests and adjustments were made. The original LeNet-5 only achieved about 87% validation accuracy. I notice that the LeNet-5 would get a good test accuracy but

stabilize at around the high 80s for validation accuracy. I thought that adding a dropout might help prevent over-fitting and it did but only to the low 90s. Further reading stated dropout benefits from larger datasets its impact was not as high as I was hoping for. I then tweaked the initial convolution layer to take a larger volume from 5x5 to 7x7. This helped push the validation accuracy above the 0.93. An alterative would be to put two convolution layers instead of a larger input volume.
The epochs were increased from 10 to 20.

- 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?

  Convolution layers work well on this problem because they capture spatial features at each given input volume. A dropout layer will help prevent overfitting and make it more robust by randomly removing a percentage of inputs. This forces the network to learn redundant representations. However, it is more effective with larger data sets

If a well known architecture was chosen:

- What architecture was chosen?

  LeNet-5

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

  I used it because it was shown as an example. It seems to be an earlier CNN implementation from 1998. Other CNN implementations used much deeper networks like GoogleNet. I was trying to do a Zf net or a alexnet but the input image was much smaller than what was used for the ILSVRC competition.

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

  The final model’s accuracy is decent but it decreases from training to validation to test at about ~0.40. This is probably due to some overfitting. I would like any advice on how to get the difference between the different data sets down.

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 of lighting.

The second image might be difficult because of the green background and random text.

The third image might be difficult because of the angle of the sign.

The fourth image might be difficult because of the bottom sign appended to it.

The fifth sign might be difficult because of the warm sunny glow and slight upward angle. There are also faint pole lines in the back.

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
60 km/h	60 km/h
Ahead Only	Ahead Only
No Entry	No Entry
General Caution	General Caution
Right of way at the next intersection	Right of way at the next intersection

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy of the test, validation, and training sets. My guess would have been 80% or 100% given the small number of images tested.

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)

Probability	Prediction
.86	60 km/h
1.00	Ahead Only
1.00	No Entry
1.00	General Caution

Probability	Prediction
1.00	Right of way at the next intersection

It concerns me that 4/5 of the probabilities are practically at 1.00. This leads me to believe that there may be some overfitting if a high probability is an indication of overfitting.

(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?