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](https://review.udacity.com/#!/rubrics/481/view) 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.

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 len function, the shape method, and numpy's unique method 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

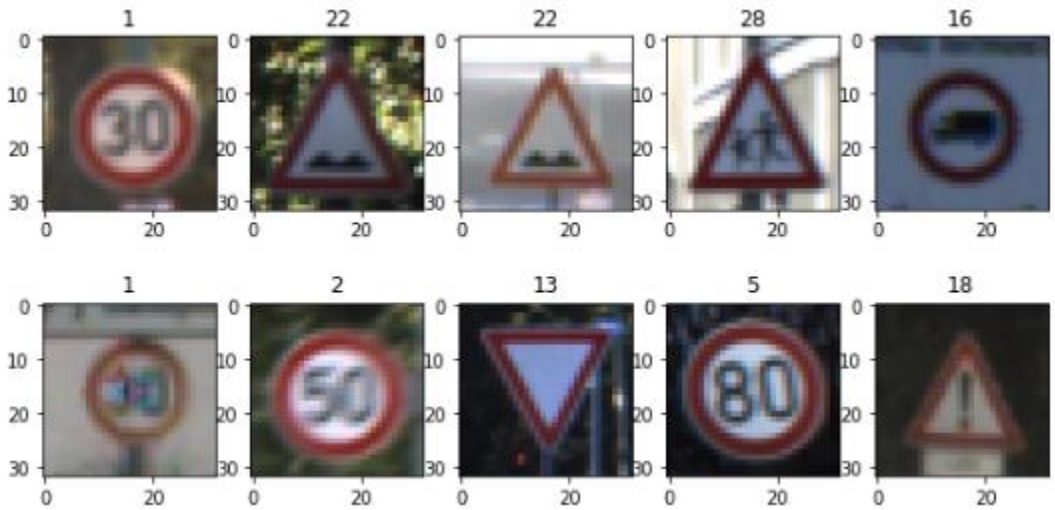
* The size of test set is ? 12630

* 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. Ten random pictures with labels are shown.

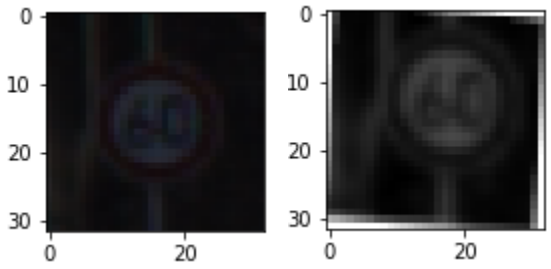


Design and Test a Model Architecture

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.)

I first used normalization as my only pre-processing step (see Traffic_Sign_Classifier-Copy1). Normalization is necessary for the sake of numerical stability. When adding very small number to a big number, there will be significant errors. By performing normalization, the mean will be zero, all other numbers will be between -1 and 1, hence error will be small. Also, having zero mean and equal variance will make it easier for optimization to proceed, less searching is required for the optimizer.

I then used pre-processing techniques such as grayscale, translation, scaling, and warping, in additional to the mandatory normalization step (see Traffic_Sign_Classifier-Copy2). Grayscale is used to eliminate the effect of color variations (as the same class of traffic signs may be different in colors) and to put more emphasis on the effect of shapes. Translation, scaling, and warping are all data augmentation techniques that can help the model generalize. Due to varying environmental conditions and camera conditions, pictures of the same type of traffic sign can appear in all sorts of positions, sizes, and orientations; hence translation, scaling, and warping should be able to help the model overcome those difficulties. However, I couldn't get the model to work very well with this heavily pre-processed data set, so I eventually reverted back to using normalization only (all following answers refer to Traffic_Sign_Classifier-Copy1 only). Below is an example of an original image and a pre-processed image.



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 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling 2x2	1x1 stride, outputs 27x27x6
Convolution 5x5	1x1 stride, valid padding, outputs 23x23x16
RELU	
Max pooling 2x2	1x1 stride, outputs 22x22x16
Flatten	outputs 7744
Fully connected	outputs 800
RELU	
Fully connected	outputs 120
RELU	
Fully connected	outputs 84
RELU	
Fully connected	outputs 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, a batch size of 70, 5 epochs, and a learning rate of 0.0007. I first determined the batch size maximizing the validation accuracy while keeping the number of epochs equal to 8 and learning rate equal to 0.001. Then I determined the learning rate in a similar manner by keeping batch size and number of epochs constant. Then I discovered that if I train too many epochs, the accuracy on the testing set would get very low due to overfitting. So I chose to only train 5 epochs.

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 ? 0.993

* validation set accuracy of ? 0.934

* test set accuracy of ? 0.922

If an iterative approach was chosen:

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

The first architecture used was the LeNet model given in the LeNet lab. It was chosen because it worked well with the MNIST dataset and achieved a test accuracy of 0.989. So I assumed it would work well with the traffic sign data too.

* What were some problems with the initial architecture?

No matter how I tuned the batch size, epochs, and learning rate, I could not get the validation accuracy to be above 0.93. So I had to adjust the model.

* 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.

The stride in the pooling layers was changed from 2 to 1. Also, an additional fully connected layer was added. It was discovered that the smaller stride and additional fully connected layer can improve validation set accuracy, hence prevents underfitting.

* Which parameters were tuned? How were they adjusted and why?

Please see my answer for quesiton 3 above.

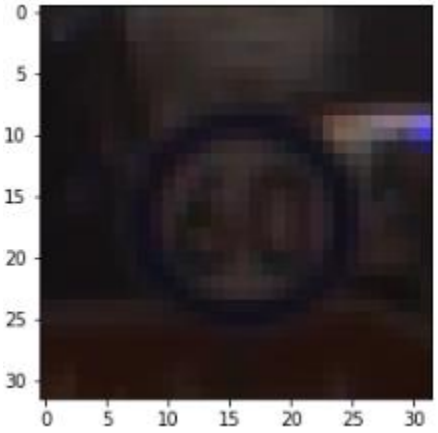
* What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

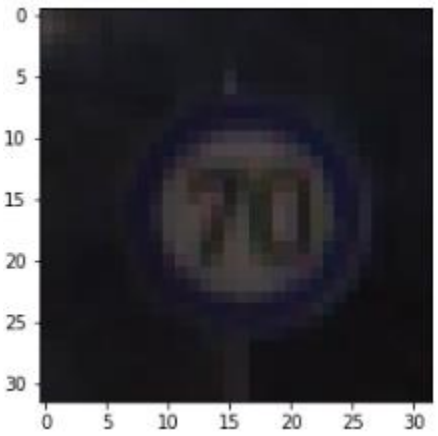
Some major design choices include the kernel size of the convolution layer, stride of the pooling layers (explained above), and the decision to not use dropout layers. I experimented with a kernel size of 3x3, but the model didn't perform better than when the kernel size is 5x5, so I stick with 3x3. My reasoning is that 3x3 convolutions are too small to capture important features in the image, whereas 5x5 convolutions can. I also experimented with dropouts, thinking that can prevent overfitting. But after trying various dropout rates, the validation accuracy turned out to be worse than when I did not use dropout. So I chose to not use it.

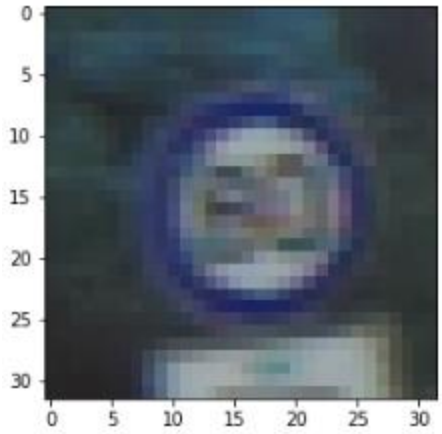
Test a Model on New Images

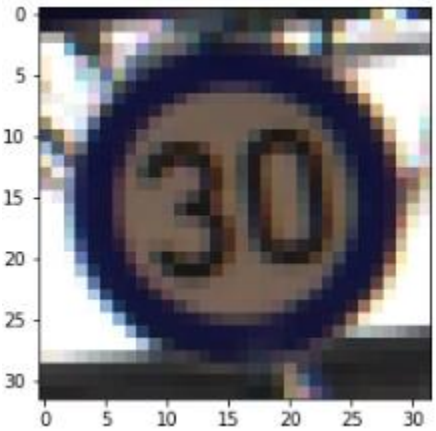
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

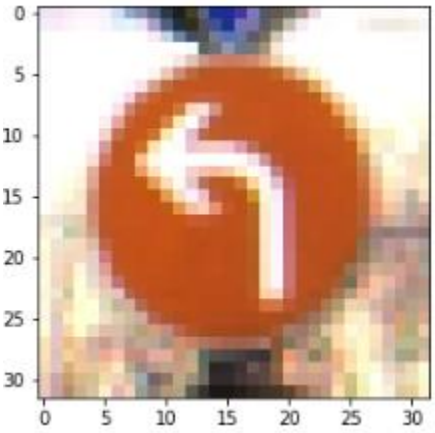
Here are five German traffic signs that I found on the web:











The first image is dark, so it might be hard to identify. The second and fourth image are clear and should be easy to identify. The third image is challenging because it is blurry. The last image is also tricky because the turn left head sign is usually blue, however, this one is orange.

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
Speed limit (60km/h)	Speed limit (60km/h)
Speed limit (70km/h)	Speed limit (70km/h)
Speed limit (80km/h)	Speed limit (80km/h)
Speed limit (30km/h)	Speed limit (30km/h)
Turn left ahead	Stop

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This is worse than the accuracy on the test set, which is 0.922.

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 top 5 softmax probabilities for each image are presented below.

Image 1:

Probability	Prediction
.99990	Speed limit (60km/h)
.7145e-04	Speed limit (80km/h)
.1593e-04	Yield
.1361e-04	Speed limit (50km/h)
.2614e-05	Keep right

Image 2:

Probability	Prediction
.99995	Speed limit (70km/h)
.3617e-04	Speed limit (120km/h)
.3411e-05	Speed limit (80km/h)
.2934e-05	Speed limit (20km/h)
.1725e-05	Turn right ahead

Image 3:

Probability	Prediction
.99978	Speed limit (80km/h)
.1705e-03	Speed limit (100km/h)
.3922e-04	Speed limit (50km/h)
.2919e-05	Speed limit (60km/h)
.1105e-05	No passing for vehicles over 3.5 metric tons

Image 4:

Probability	Prediction
.99943	Speed limit (30km/h)
.5587e-03	Speed limit (20km/h)
.6465e-04	Speed limit (70km/h)
.3274e-06	End of speed limit (80km/h)
.3136e-06	Speed limit (100km/h)

Image 5:

Probability	Prediction
.99981	Stop sign
.1407e-03	No passing
.4064e-04	Yield
.3884e-05	Priority road
.2082e-06	Speed limit (80km/h)

(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?