Traffic Sign Recognition

Writeup – Jesse R

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 https://github.com/JesseR2/CarND-Traffic-Sign-Classifier-Project

Data Set Summary & Exploration

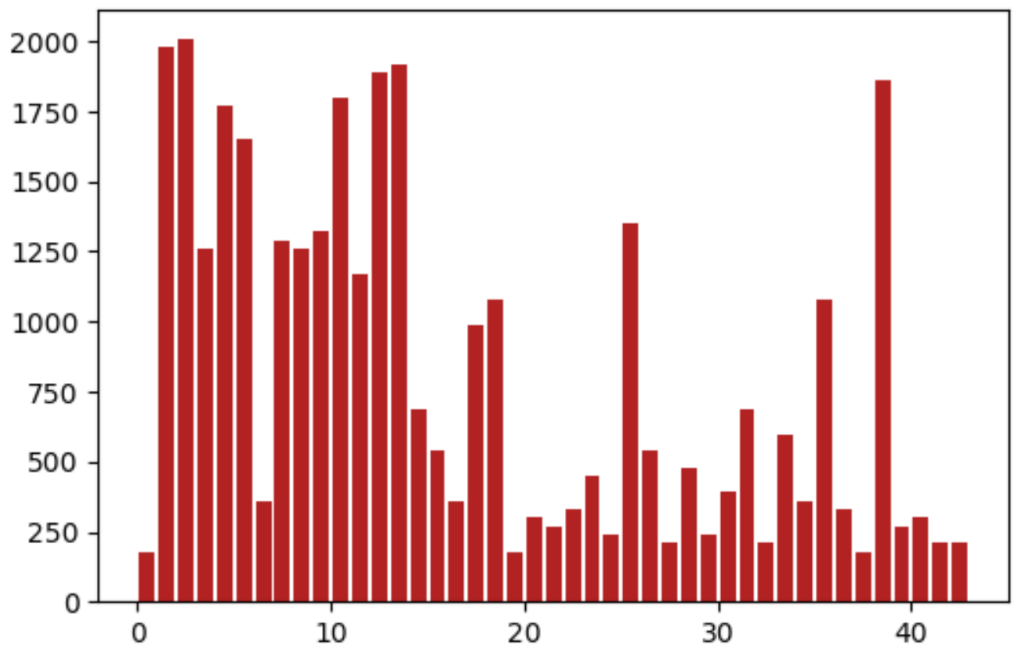
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

I used the numpy library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34,799 images (32x32x3)
- The size of the validation set is 4,410 images (32x32x3)
- The size of test set is 12,630 images (32,32,3)
- The shape of a traffic sign image is 32x32x3
- 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 many examples of each class are in the training set.



Here is a description of each of the 43 classes of signs. These were provided in a CSV file with the project

ClassId	SignName
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vehicles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vehicles
16	Vehicles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right
25	Road work
26	Traffic signals
27	Pedestrians
28	Children crossing
29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and passing limits
33	Turn right ahead
34	Turn left ahead
35	Ahead only
36	Go straight or right
37	Go straight or left
38	Keep right
39	Keep left
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vehicles over 3.5 metric tons

Here is a sample of each class from the training set prior to preprocessing.

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 tried two different methods of equalizing the images. First I converted the images to HSV and just equalized the values using 'exposure.equalize_hist' from skimage then converted them back to RGB. I also tried equalizing the images as RGB only and had good results. I chose not to use grayscale since the color seems to offer more insight into the sign class. The equalization was done after seeing that many of the images were very dark and other seemed to be overexposed. This same processing was also applied to validation and test images.

Here are the same images shown above but with the preprocessing applied.

As a last step, I scaled the images to a range of -1 to 1 for each color channel.

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 28x28x16
RELU	
Max pooling	2x2 stride, outputs 14x14x16

Layer	Description
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x32.
RELU	
Max pooling	2x2 stride, outputs 5x5x32
Fully Connected	512 nodes
RELU	
Dropout	Keep_prob .5
Fully Connected	128 nodes
RELU	
Dropout	Keep_prob .5
Fully Connected	43 classes

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.

After testing several sets of hyperparameters and models, I chose to use 20 Epochs with Adam optimizer to reduce the mean cross entropy at a learning rate of .0006 with batch size of 128. I set both dropout layers at 50% keep.

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 100.0%
- validation set accuracy of 95.9%
- test set accuracy of 94.4%

If an iterative approach was chosen:

- *What was the first architecture that was tried and why was it chosen?* I first used the standard LeNet model and parameters from the previous project.
- *What were some problems with the initial architecture?* This provided a validation set accuracy of 90.3% which is below the target for the project.
- *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 first increased the number of filters in the convolutional layers to 16 and 32 in an attempt to reduce the underfitting. I also increased the fully connected layers to 256 and 128. This improved the accuracy about 2%. I then increased the first dense layer to 512 nodes and gain a bit more accuracy. I added dropout to the last layer and gained about 1% and finally added it to the first fully connected layer as well gaining almost one more percent in accuracy.
- *Which parameters were tuned? How were they adjusted and why?* I tried several combinations of dropout (increasing as overfitting was observed), batch size (as memory seemed to be an issues) and learning rate. Lowering it to .0006 seemed to give a good balance of quick learning without causing a lot of noise from epoch to epoch in the later epochs. I tried rates from .001 to .0003.
- *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?* As with most image data models, CNNs are used to allow spatial invariance since classifying the sign is the goal regardless of its location in the image. The LeNet architecture was good start since it was successful with images of similar size. It was unable to get accurate results even with preprocessing and hyperparameter tuning so widening the model as a

obvious choice. This didn't impact training time too much and gave good results. As the model grew it tended to overfill after several epochs. Early stopping may have helped this, but I chose to add dropout to reduce it.

If a well known architecture was chosen:

- *What architecture was chosen?* I was training on a laptop for this project so I tried to keep the model small. This was able to run at about 120 seconds per epoch. I did do some brief testing on and EC2 p2.2xlarge instance and found about a 20x speed up.
- *Why did you believe it would be relevant to the traffic sign application?* As stated above, this model is proven to be successful at classifying small color images.
- *How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?* The network achieved 100% on the training data indicating that it is still overfitting but was able to achieve 95.9% on validation and 94.4% on the test set.

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 ten German traffic signs that I found on the web, I cropped and resized the images to match the model input:

The images are all similar to those in the training data and are likely to be classified accurately by the model.

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
Keep right	Keep right
Straight or Left	Straight or Left
Turn Right Ahead	Turn Right Ahead
Straight or Right	Straight or Right
Speed Limit 120	Speed Limit 120
Speed Limit 50	Speed Limit 50
Speed Limit 80	Speed Limit 80
Stop	Stop
Right of way – Next Intersection	Right of way – Next Intersection
No Passing	No Passing

The model was able to correctly predict 10 of the 10 traffic signs, which gives an accuracy of 100%. This compares similarly to the accuracy on the test set of 94.5%

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 21th cell of the Jupyter notebook.

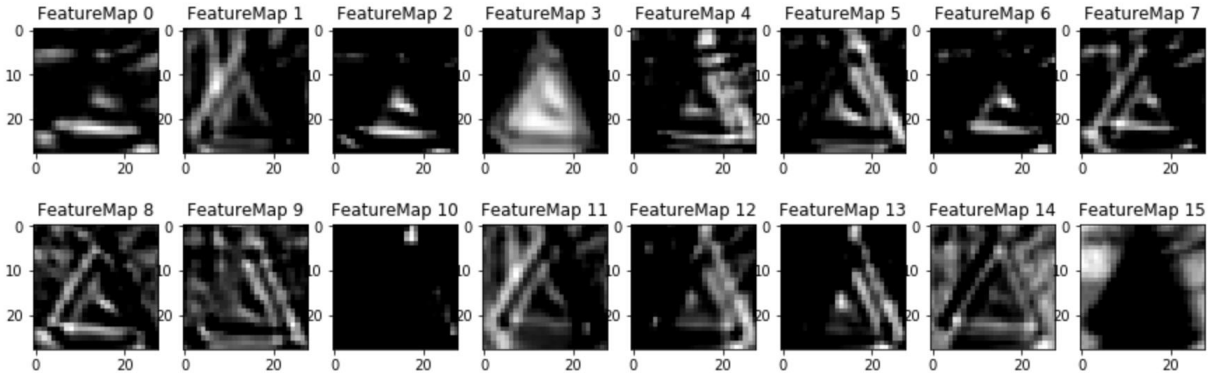
Each of the predictions had a softmax output of at least 99.999%. The top 5 results for each can be found in 23rd cell if the notebook. This was unexpected, so I ran this without softmax and found that the logits were all above 20 or 30 for the predicted sign and in the 10 to 20 range or less for the others. Since softmax takes uses exp(logit) and averages it over the sum of the exp(logits) it is reasonable that the predictions were so close to 100%. I suspect that if the use of relu rather than activations like sigmoid or tanh allows the outputs of the model to get much larger than the expected range of -1 to 1.

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

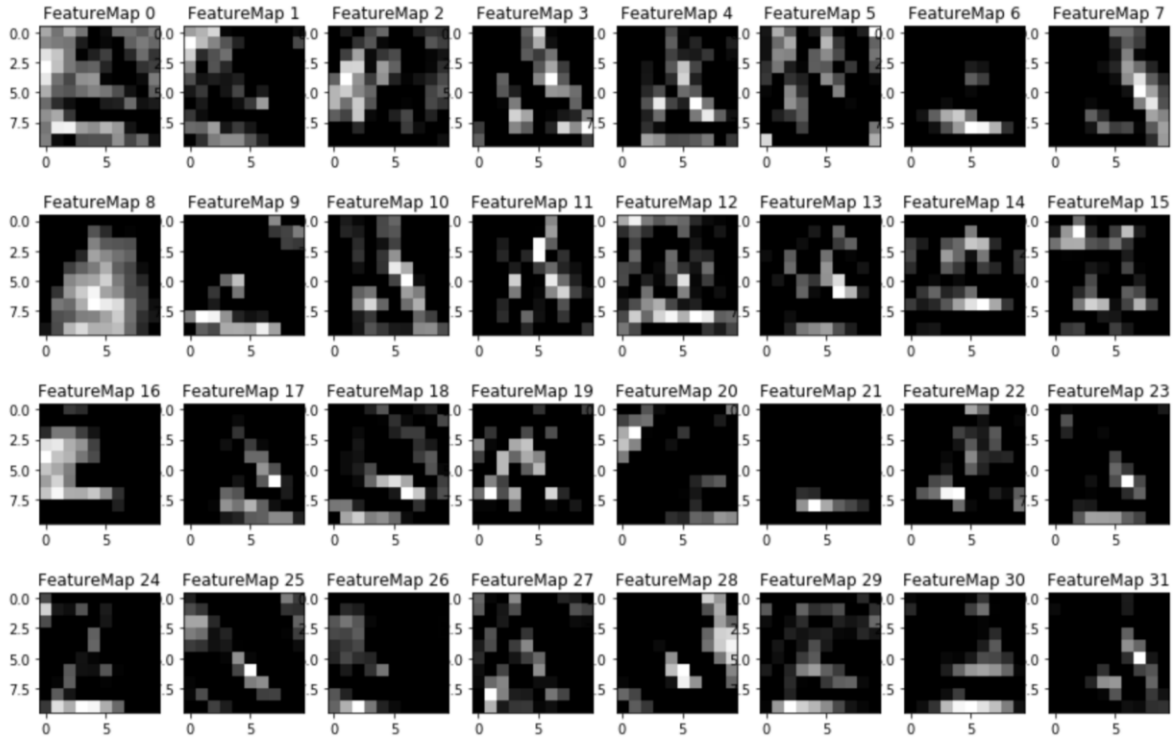
1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

The 16 feature activations of the first layer were visualized using the following image as an input.

Here is what they look like.



I also generated a visualization of the 32 features of the second convolutional layer.



The first layer was primarily finding edges but feature 3 looks like its activating on white areas and feature 15 is activating on dark areas. None of these appear to be activating on red so perhaps using a grayscale image would have produced similar results with a smaller and easier to train model.

The features in the second layer are difficult to identify the activation cause but it clear they are using higher level features than the previous layer.