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
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](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb)

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 basic python commands 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.

Below is an exploratory visualization of the data set. It is a bar chart showing how the data is separated into the various classes.

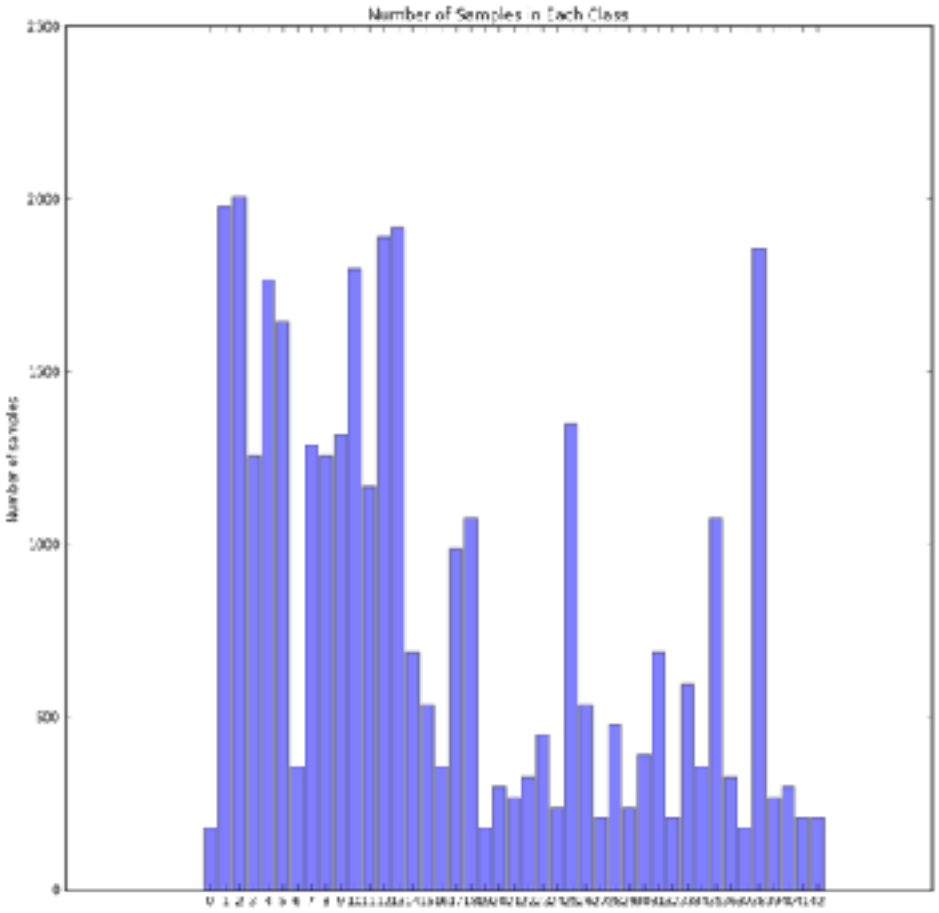

Figure 1: Bar Chart of Classes

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.)

Before preprocessing the data at all, I ran it through several different convolutional neural network architectures. I found that I could achieve about 90% validation accuracy without any preprocessing using a "beefed-up" version of LeNet-5. Therefore I decided to try only a couple relatively simple preprocessing techniques in hopes of gaining an extra 3%-5%. I implemented two techniques.

Normalization- First I normalized the data by passing all training, validation, and test images through a function that scales all pixel values into the range [0,1]. Normalization can have several beneficial effects. It can help to avoid rounding errors, it can improve training speed, and it can help to avoid local minima and maxima in the error-space that would stall training. It is easy to do, so it was worth trying.

After training the network I read that it is generally better to have the inputs centered around zero than to in the range from zero to one. Therefore in future attempts I would suggest trying a slightly modified algorithm that centers the data in the range [-1,1].

Augmentation- Next (but prior in the actual notebook code) I implemented a vertical flip of the images. I then concatenated the new images onto the training set and doubled the labels array. This gave double the amount of training examples. In hindsight this may not have been the best way to augment that data because flipping some of the signs might have changed their meanings (this is definitely true with a horizontal flip, which is why I use a vertical flip). If I were to try for further improvements to this model, I would implement class-specific augmentation. I would flip only the classes that are under-represented in the total number of training examples, as show in Figure 1.

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:

Model Architecture
Below is a description of my model's architecture. I started from the LeNet-5 example and added complexity from there.

Layer 1:
- Convolutional 5x5. Stride = 1. Depth = 20.
- Activation Layer: Relu.
- Maxpool Layer 2x2. Stride = 2.

Layer 2:
- Convolutional 5x5. Stride = 1. Depth = 20.
- Activation Layer: Relu.
- Maxpool Layer 2x2. Stride = 2.

Layer 3:
- Convolutional 5x5. Stride = 1. Depth = 20.
- Activation Layer: Relu.
- Maxpool Layer 2x2. Stride = 2.

- Flattening.

Layer 4:
- Fully-Connected. Neurons = 120.
- Activation Layer: Relu.

Layer 5:
- Fully-Connected. Neurons = 120.
- Activation Layer: Relu.

Layer 5:
- Output, Fully-Connected. Neurons = 43.

The weights have been initialized with a truncated normal distribution around zero, with sigma left as a hyper-parameter.

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyper-parameters such as learning rate.

To train the model, I used an Adam Optimizer. This was chosen because it had already been proven effective in LeNet-5.

Hyperparameters:
EPOCHS = 100 #Train
BATCH_SIZE = 128 #Batch size
rate = 0.001 #Learning rate
sigma = 0.001 #Weights initialization value

It was found that for such a small data set, it is better to use a small batch size. This allows for the weights to be updated more frequently than with a larger batch size.

The learning rate was modified many times during training. Eventually 0.001 was chosen as the highest rate that still achieved a convergence above 93%.

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:
* validation set accuracy of 93%
* test set accuracy of 91%
* new image accuracy of 60%

If an iterative approach was chosen:
What was the first architecture that was tried and why was it chosen?
LeNet-5 was chosen as a starting point because of the example notebook that was given.

What were some problems with the initial architecture?
The initial architecture actually worked quite well on the original dataset. The best validation accuracy that I got with this architecture was about 92%. However in the end I expanded that size by increasing the number of neurons in each layer. This was done after I added in the flipped images. It seems that the original architecture was too small to handle the expanded data set. Later I added a third convolutional layer, hoping to increase the complexity of features that could be detected. I hoped that this would help the model generalize to new pictures beyond the original data set.

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.
Discussed above. Also I used pooling layers to avoid overfitting. I considered using dropout as well, but I never saw any signs of overfitting, so it seemed unnecessary to add further regularization.

Which parameters were tuned? How were they adjusted and why?
After settling on my expanded architecture, I repeatedly adjusted the batch size, learning rate, and to a lesser extent the sigma value of weight initialization.

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?

A convolutional network approach was chosen because of the nature of the image classification problem. In image data, it is most important to look at how an image varies in a small region than to try to compare all of the pixel data streams separately. This is because pixels located close to one another are much more likely to be meaningfully related than pixels that are far apart. The CNN approach also allows for a high degree of translational invariance, which is crucial for image classification. In this example, it makes sense because we can expect the same sign to appear in different locations in various images.

Neural networks are highly prone to overfitting. This became of great concern once the size of the network was increased by adding more neurons and an extra layer. Therefore it was important to consider regularization techniques. In this case pooling was used as the only regularization technique. If the model had shown signs of overfitting, I would have considered implementing dropout on the fully-connected layers.

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:

Figure 2: Five New Test Images Found On Google Images

These images were chose to be easily identifiable but not trivially so. For instance, note the distinct shape of the tree behind the stop sign, the text below the yield sign, and the black bar at the bottom of the priority sign image. All of these shapes could cause confusion in the network. The images were also chose to be a variety of shapes and sizes. The images were then resized to be (32,32,3) arrays.

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

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. At first this seems not to be a very good accuracy, especially when compared to the test accuracy of 91%. However it shows that the network has definitely developed at least some ability to generalize to totally new information that comes in a different form. I would consider this a successful starting point that could be improved upon in the future.

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.

Interestingly the model appears to be completely certain in its selection choices. See Table 1 below. I am not certain why it exhibits such certainty. I would suggest further investigation in the future.

Table 1: Softmax Scores

Classification: Stop
Top Five Indices: [[14 17 3 11 13]]
Softmax Scores:
[[1.00000000e+00 6.06958209e-18 1.67870463e-19 2.37640141e-26
4.13896158e-29]]

Classification: Speed limit (50km/h)
Top Five Indices: [[2 6 21 7 5]]
Softmax Scores:
[[1.00000000e+00 4.19613210e-13 5.40708310e-15 1.84899181e-20
1.51613692e-20]]

Classification: No entry
Top Five Indices: [[17 0 1 2 3]]
Softmax Scores:
[[1. 0. 0. 0. 0.]]

Classification: Priority road
Top Five Indices: [[12 10 5 0 20]]
Softmax Scores:
[[1.00000000e+00 3.20878324e-11 1.98658115e-18 9.35559988e-19
2.30131934e-19]]

Classification: Road work
Top Five Indices: [[25 26 20 11 0]]
Softmax Scores:
[[9.99892831e-01 1.07204512e-04 2.09057571e-08 1.53296015e-10
6.28544306e-17]]