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](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.

You're reading it!

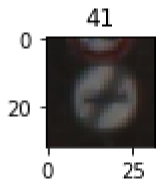
Data Set Summary & Exploration

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.

* 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.

I printed a copy of each of the 43 unique classes within the jupyter notebook. Here is an example:



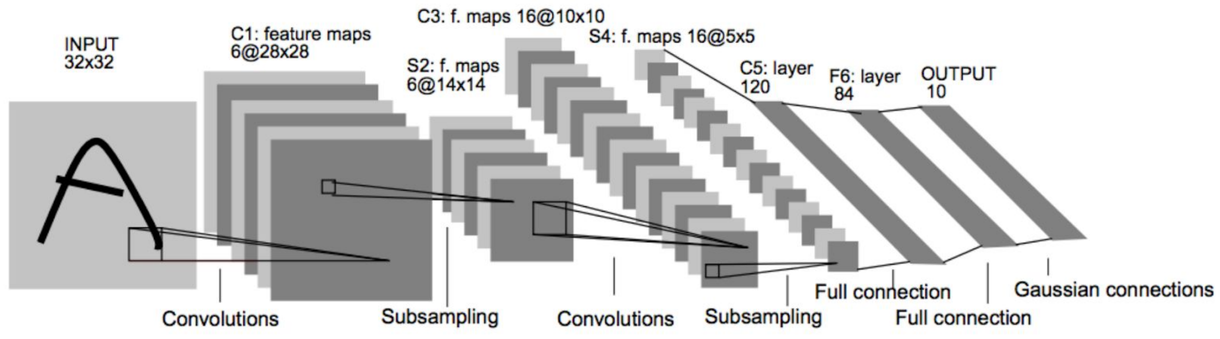
Design and Test a Model Architecture

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.

As a first step, I decided to convert the images to grayscale because it made the learning process easier when dealing in just one color channel.
As a last step, I standardized the image data to center the data around 0. This helps the learning process as well.

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 mimics the LeNet model from the lectures:



However, I also introduced three instances of dropout: once after the flattening from 5x5x16 to 400, once after the fully-connected layer of 120, and once after the fully-connected layer of 84.

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.

The training operation of my model was one of minimizing loss (average cross entropy). For optimization, I used Adam optimization. I used a batch size of 100 and 80 epochs. The learning rate was 0.001. The weights and biases were normalized with a mean of 0 and standard deviation of 0.01.

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.

LeNet's use of sparse convolutional layers and max pooling worked well for this model. As an iterative process, I tuned one parameter at a time while holding all other parameters constant. Additionally, I added dropout in various locations until the results were good. I calculated the accuracy of each image set using the evaluate() function (line 26-34 of "Train, Validate and Test the Model" cell).

My final model results were:
* training set accuracy of 1.000
* validation set accuracy of 0.947
* test set accuracy of 0.932

If an iterative approach was chosen:
* What was the first architecture that was tried and why was it chosen?
Lenet was chosen as the baseline because it was known that 89% accuracy was achievable with it. From there, with grayscaling added, parameters were tuned.

* What were some problems with the initial architecture?
Without dropout, the model was overfitting the training data.

* 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.

Since the model was overfitting, I added dropout to counteract the error and bring up the validation accuracy.

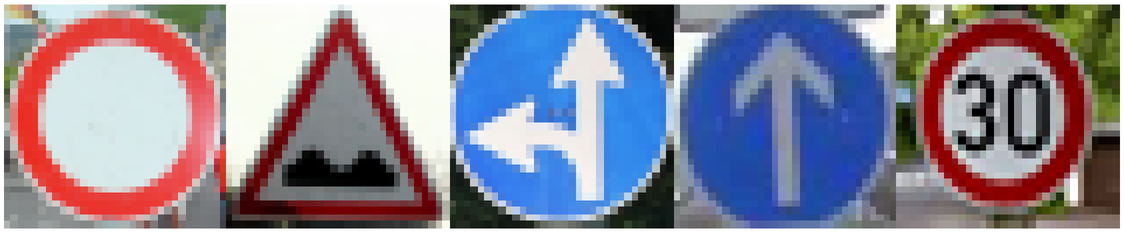
* Which parameters were tuned? How were they adjusted and why?
Number of epochs, batch size, learning rate, keep probability. They were adjusted incrementally; I held all other parameters constant and observed the accuracy as one of the parameters was increased or decreased slightly. This ensured no confounding variables when tuning the parameters. The point of tuning was to find the optimal combination for the highest accuracy.

* 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 for this image identification task, because they act as filters to detect specific features in the image. As mentioned previously, dropout layers help with preventing overfitting; if the model becomes too reliant on the neurons which are available from the training set, then it will not perform as well when some neurons are missing.

Test a Model on New Images

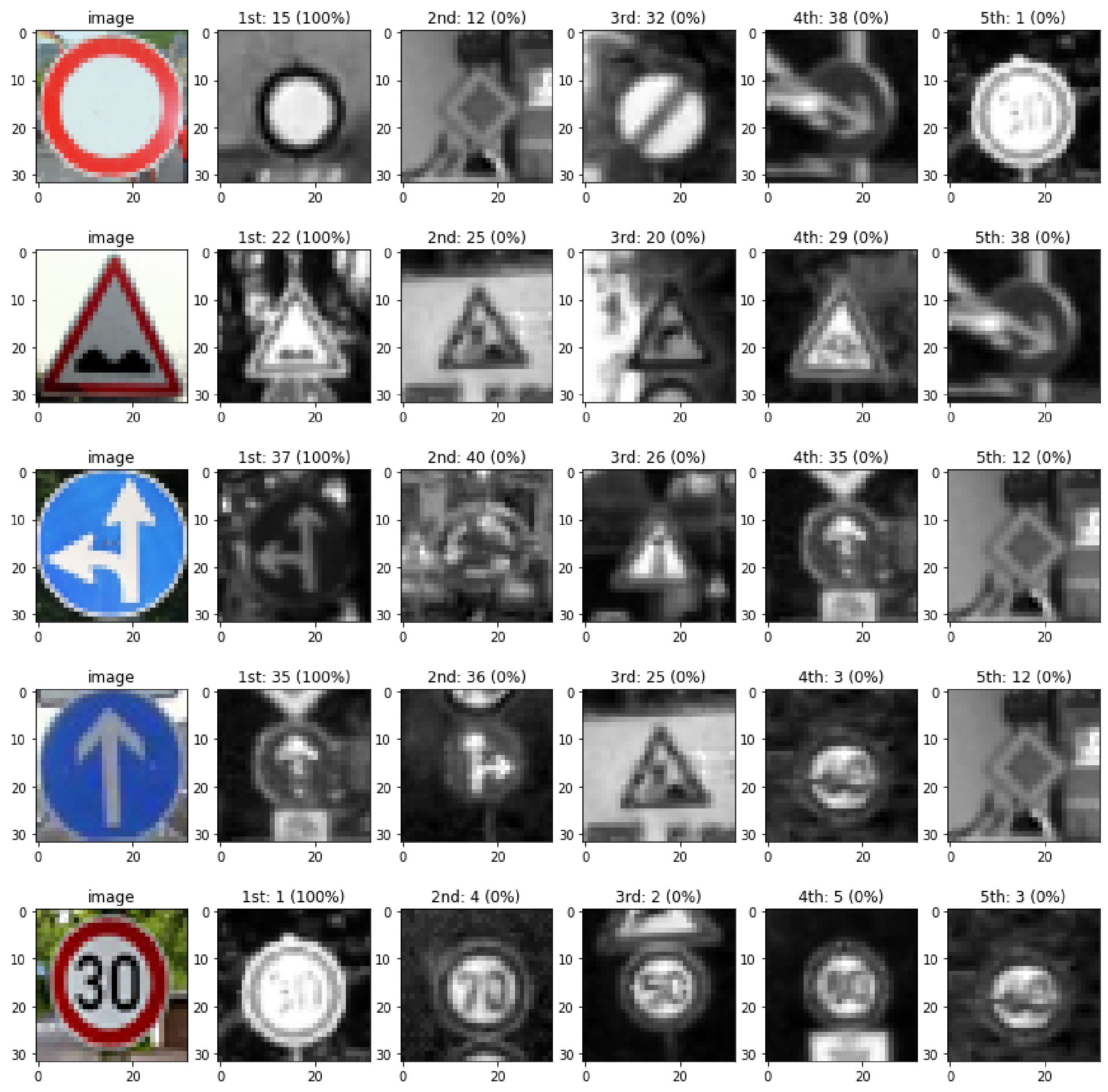
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

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



All of these should be relatively easy to classify. The images are clear, unobstructed and untransformed.

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 5 of the 5 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of 93.2%.

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.

The softmax probabilities are depicted in the image above. The model was ~100% sure of each of its 1st guesses. Probability for other guesses were negligibly low.