Traffic Sign Recognition

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

Data Set Summary & Exploration

1. A basic summary of the data set I used the numpy library 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. An exploratory visualization of the dataset. Here is an exploratory visualization of the data set. It is a bar chart showing how the data distributed. It's clear that the data set are unbalanced. ![alt text][image1] ### Design and Test a Model Architecture

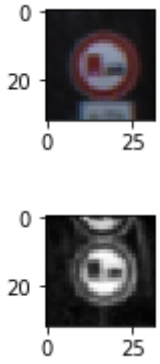
1. Preprocessed the image data

As a first step, I decided to convert the images to grayscale because using color channels won't improve things a lot.

Then, I devide the data with 255 to normalize the image data in order to make searching faster.

I also apply histogram equalization to extract more feature from the image.

Here is an example of a traffic sign image before and after pre processing.



2. Model architecture

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 grayscale image
Convolution1 5x5x6	1x1 stride, valid padding, outputs 28x28x6

Layer	Description
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution2 5x5x16	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Fully connected1	inputs 400, outputs 120
RELU	
Droupout	keep probility 50%
Fully connected2	inputs 120, outputs 84
RELU	
Fully connected3	inputs 84, outputs 43
Softmax	softmax cross entropy with logits

3. Modele Training

To train the model, I used training dataset with TensorFlow AdamOptimizer and learning rate set to 0.001. I use batch size 256 and 100 epochs,however, It normally stops improving after ~60 epochs and achieve ~96% accuracy on the validdation data set.

4. Discuss

At first, I try to use the same architecture from LeNet lab, but the maximum accuracy I can achieve is around 87%. Then I tried to normalize the data to make the searching fater. After implement the histogram equalization, 92% of accuracy was achieved. The last measure I took is add dropout to the fully connected layer to avoid overfitting and I tune the final model with a lower learning rate. Then it is able to achieve 96 % of accuracy in the validation data set and 94.2% in the test data set.

My final model results were:

- validation set accuracy of 96.3%
- test set accuracy of 94.2%

Test a Model on New Images

1. New Image

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

The first image might be difficult to classify because there is some dirt on it.

2.Model Prediction

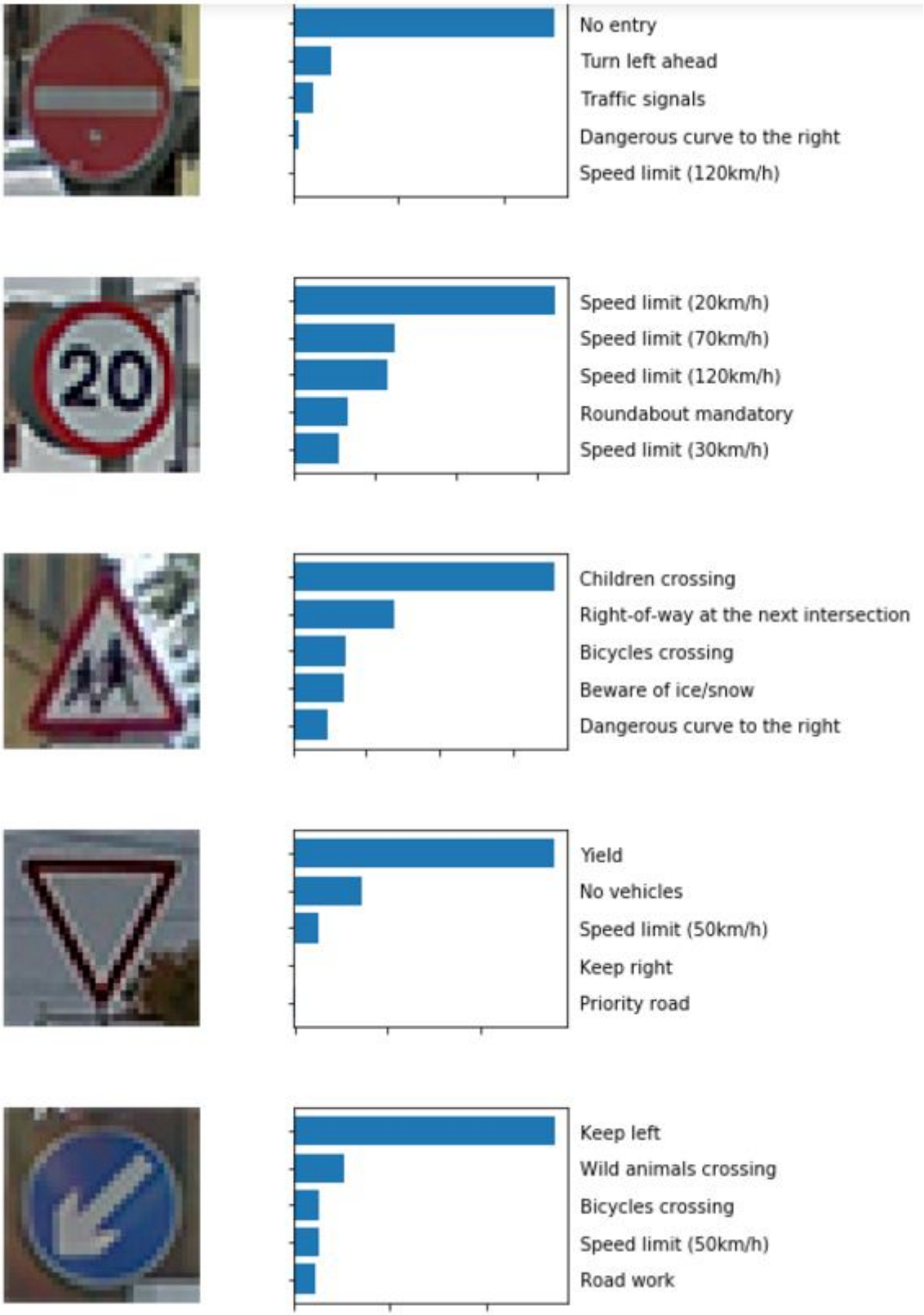
Here are the results of the prediction:

Image	Prediction
No entry	No entry
Speed limit(20km/h)	Speed limit(20km/h)
Children crpssing	Children crpssing
Yield	Yield
Keep left	Keep left

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This is excellent result, but maybe it's just because the new image are not challenging enough.

3. Softmax Probability

The code for making predictions on my final model is located in the 14th cell of the Ipython notebook.



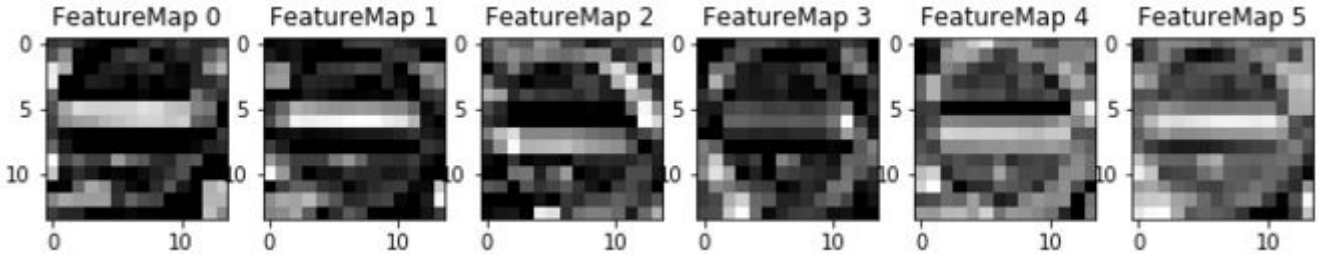
For the first image, the model is relatively sure that this is a stop sign (probability of 49%), and the image does contain a no entry sign. The top five soft max probabilities were

For the second image, the model is not so sure about different speed limit. It seems the model is reletively not good at number regonization than graph regonization.

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

1. Feature maps

The visualization of conv1 layer is shown here, which contains 6 5x5 filters. As expect, the first layer is able to detect low level pixel patterns, like the "-" sign in the no entry traffi sign.



In [ ]: