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 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 .ipynb with jupyter notebook.

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.

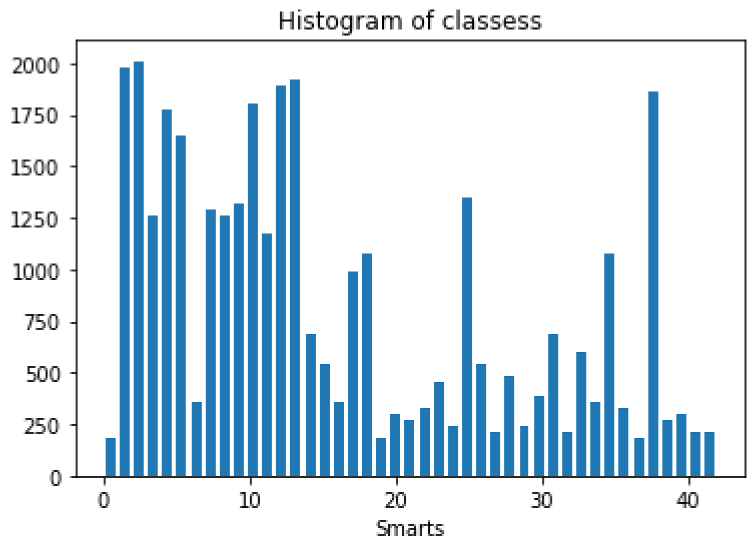
I used the pandas library to calculate summary statistics of the traffic signs data set:

- 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. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data distribute, and visualization of the dataset are shown following pictures.

Turn left ahead
Speed limit (50km/h)
Double curve
Speed limit (100km/h)
Right-of-way at the next intersection
End of no passing by vehicles over 3.5 metric tons
Vehicles over 3.5 metric tons prohibited
Keep right
Wild animals crossing

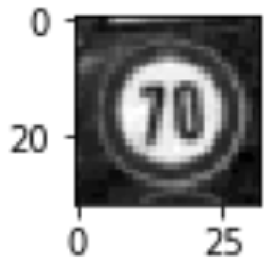


Design and Test a Model Architecture

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.)

- As a first step, I decided to convert the images of test and train to grayscale because lenet neutral network is needed by grayscale pics. After that, i tested grayscale pic from train pictures. Here is an example of a traffic sign image before and after grayscaling. As you can see in the next pictures.

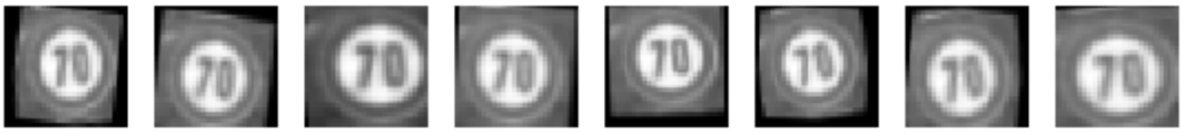
```
Here is classify (4: Speed limit (70km/h))
```



- Following step, I transfromed image with its rotating(`cv2.getRotationMatrix2D`), shearing, and translating(`cv2.getAffineTransform`) pictures for generating sample data. because it will increase robustness of the system. The difference between the original data set and the augmented data set is the following:

```
raw:
```

```
Generated images:
```



Next, To add more data to the the data set, i generated fake data to insreasing the accuracy of training, it shown some properties of pic sizes, feature nums, id and names for every classfied pictures, more details are shown in xxx.ipynb.

- As a last step, I normalized the image data to 0,1(`(xxx / 255.).astype(np.float32)`), and used `shuffle` module for disordered data. because if we do not disturb the order of data, then suppose we train a neural network classifier, then all small batches in each epoch will be identical, which will lead to slower convergence speed, and even lead to the tendency of neural network to the order of data.

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.

To creating TF architecture, i set paramters of DL

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 RGB image
Convolution_1 3x3	5x5 stride, valid padding, 6 convolution kernels, outputs 28x28x6
RELU	
Max pooling	2x2 stride, valid padding, outputs 14x14x6
Convolution_2 3x3	5x5 stride, valid padding, 16 convolution kernels, outputs 10x10x16
RELU	

Layer	Description
Max pooling	2x2 stride, valid padding, outputs 5x5x16
FLATTEN	Input 5x5x16, Output 400
Fully connected_1	Input 400, Output 120
RELU	
Fully connected_2	Input 120, Output 84
RELU	
Fully connected_3	Input 84, Output 43
DropOut	
Softmax	43

More details are shown in the function of `LeNet(x, keep_prob)`.

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.

```
EPOCHS = 150
BATCH_SIZE = 128
LEARNING_RATE = 0.0001
```

To train the model, I used one_hot, cross_entropy and optimizer.

- one_hot [`tf.one_hot(tf.placeholder(tf.int32, (None)), 43)`]
- cross_entropy [`tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=one_hot_y)`]
- optimizer [`tf.train.AdamOptimizer(learning_rate = LEARNING_RATE)`]

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 0.994
- validation set accuracy of 0.952
- test set accuracy of 0.930 you can find out with title of `To validate data`

```
valid_accuracy = evaluate(X_valid, y_valid)
test_accuracy = evaluate(X_test, y_test)
print("Valid Accuracy = {:.3f}".format(valid_accuracy))
print("Test Accuracy = {:.3f}".format(test_accuracy))
```

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen? I used the LetNet Architecture at first, LeNet-5 can tell apart two characters even when they strongly overlap.

- What were some problems with the initial architecture? I spent a lot of time on the train DL. because i need to adjust the parameters of batch size and leaning rate.

- 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 architecture are shown in the top table. According to the requirement, I grayscalize and nomarlize the original data, when I use formulate x-128/128. but i found that the accuracy is low 93, then i set `(xxx / 255.).astype(np.float32)`. I change the value of batchsize,learning rate,epoch, at last I get accuracy of 95.2% in validation set and accuracy of 93.0% in test set.

- Which parameters were tuned? How were they adjusted and why? I set `BATCH_SIZE = 256`, because it can accelerate neural network results. After that the accuracy can not meet standard. then i set lower num of 128, the final results of test is 0.93.

- 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? I trained the model using an Adam optimizer , learning rate of 1e-4 , epochs of 150 and batch size of 128. These parameters determine the speed and precision of deep learning training.

If a well known architecture was chosen:

- What architecture was chosen? LeNet-5
- Why did you believe it would be relevant to the traffic sign application? When using shift invariant feature detectors on multilayer and constrained networks, and traffic classify can met this characteristics, so the model can perform very well in this task
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well? I added a test set later to verify that the results meet the acceptance criteria.

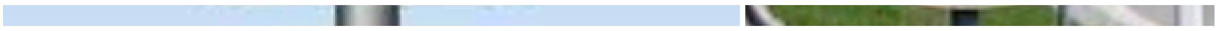
Test a Model on New Images

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:

80

STOP



The mistake this modle had made in new image set is Keep right sign which had been predict as Speed limit. because this image has a black background which could be difficult to distinguish in gray level

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:

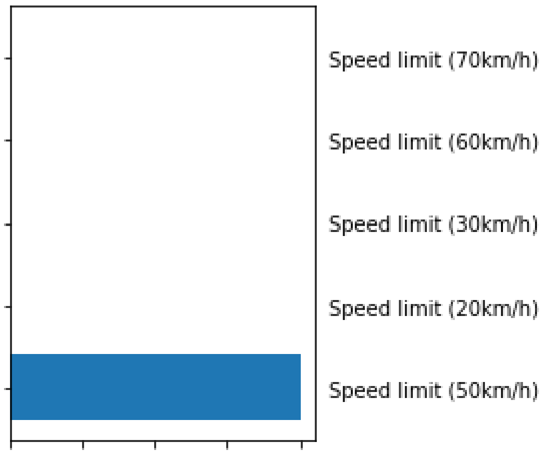
Image	Prediction
Speed limit (80km/h)	Speed limit (80km/h)
Stop	Stop
No entry	No entry
General caution	General caution
Turn right ahead	Speed limit (80km/h)

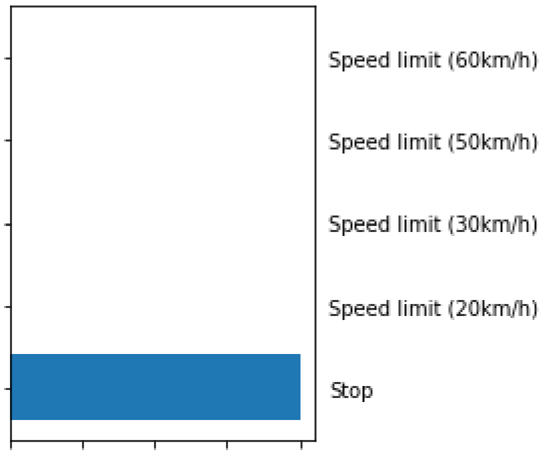
The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%.

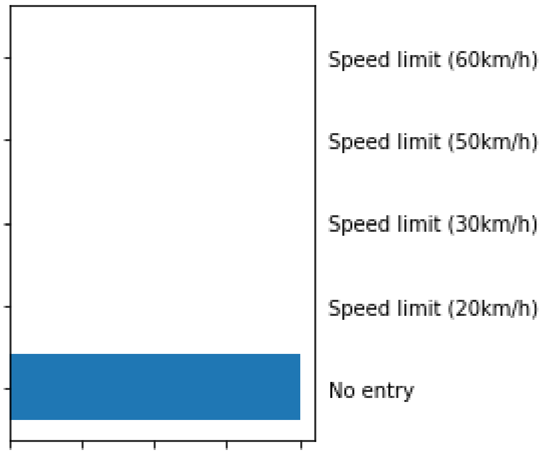
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

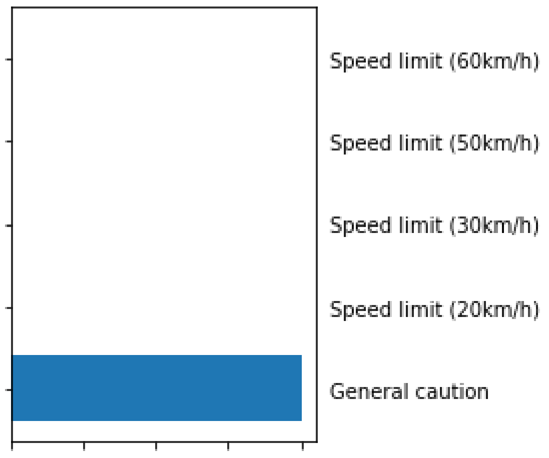
The code for making predictions on my final model is located in the 11th cell of the Ipython notebook.

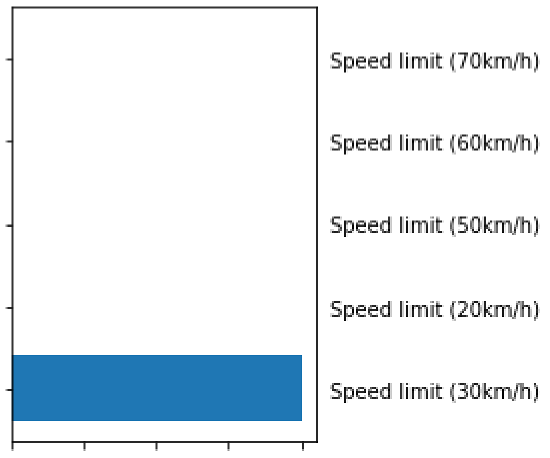
For the first image, the model is relatively sure that this is a stop sign (probability of 0.6), and the image does contain a stop sign. The top five soft max probabilities were shown in the following pictures.

- Speed limit(80km/h) 

- Stop 

- No Entry 

- General Caution 

- Turn Right Ahead 
Speed limit (70km/h)
Speed limit (60km/h)
Speed limit (50km/h)
Speed limit (20km/h)
Speed limit (30km/h)