P2 - Traffic Sign Recognition – Writeup

Ilsun Choi

1 Data Set Summary & Exploration

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.

1.1 Q1

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

```
### Replace each question mark with the appropriate value.
### Use python, pandas or numpy methods rather than hard coding the results

# TODO: Number of training examples
n_train = len(X_train)

# TODO: Number of validation examples
n_validation = len(X_valid)

# TODO: Number of testing examples.
n_test = len(X_test)

# TODO: What's the shape of an traffic sign image?
image_shape = X_train[0].shape

# TODO: How many unique classes/labels there are in the dataset.
n_classes = len(set(y_train))

print("Number of training examples =", n_train)
print("Number of testing examples =", n_test)
print("Image data shape =", image_shape)
print("Number of classes =", n_classes)
```

```
Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 43
```

Figure 1 Dataset's information

2017-10-15

1.2 Q2

Here is an exploratory visualization of the data set.

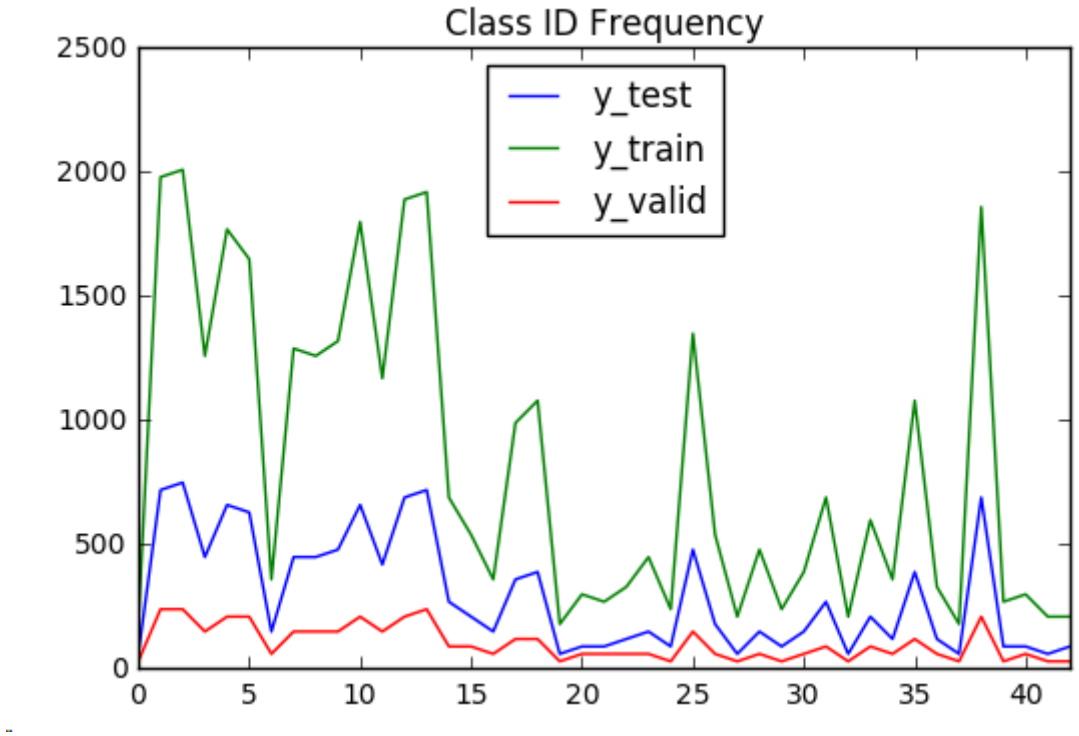

Figure 2 Dataset's graph

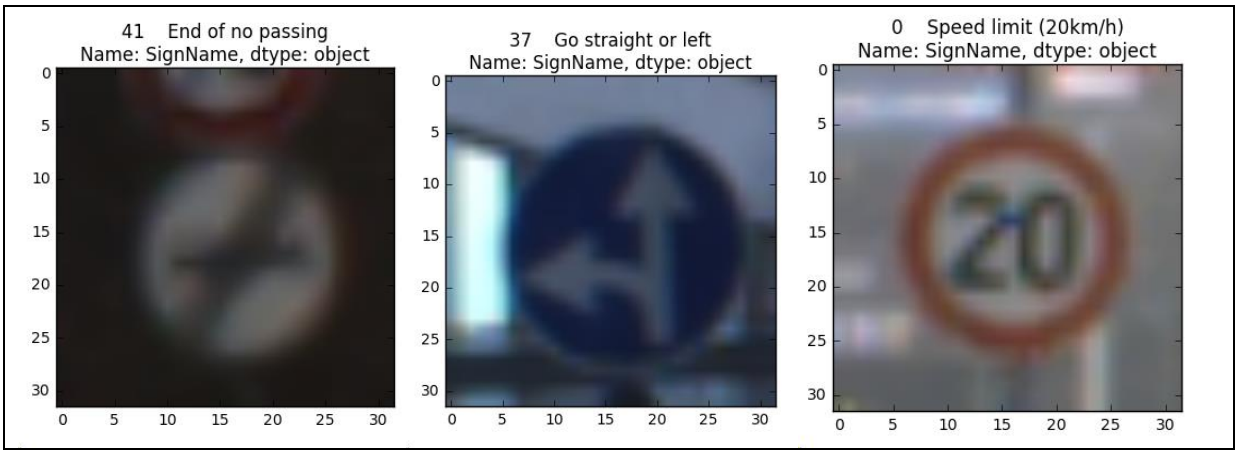

Figure 3 Dataset sample images

2 Design and Test a Model Architecture

2.1 Q1

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.)

step1. I define the preprocess_data() function. The function change the data for learning.

```
## grayscale
import cv2  #bringing in OpenCV libraries

print(X_train[index])
def preprocess_data(X):
    X = [cv2.cvtColor(x, cv2.COLOR_RGB2GRAY) for x in X]
    X = np.array(X)
    X = np.divide(np.subtract(X,128.),128)
    #X = 0.114 * X[...,0] + 0.587 * X[...,1] + 0.299 * X[...,2] # BGR->Gray
    return X

X_train = preprocess_data(X_train)
X_valid = preprocess_data(X_valid)
X_test = preprocess_data(X_test)

print(X_train[0].shape)

plt.imshow(X_train[index], cmap='gray')
print(X_train[index])
```

Figure 4 preprocess_data() function

First the function convert color images to grayscale images. And I adapt the nomalizing fomula x = (x-128)/128(when divide the data, a number must be float. if not, doesn't work well).

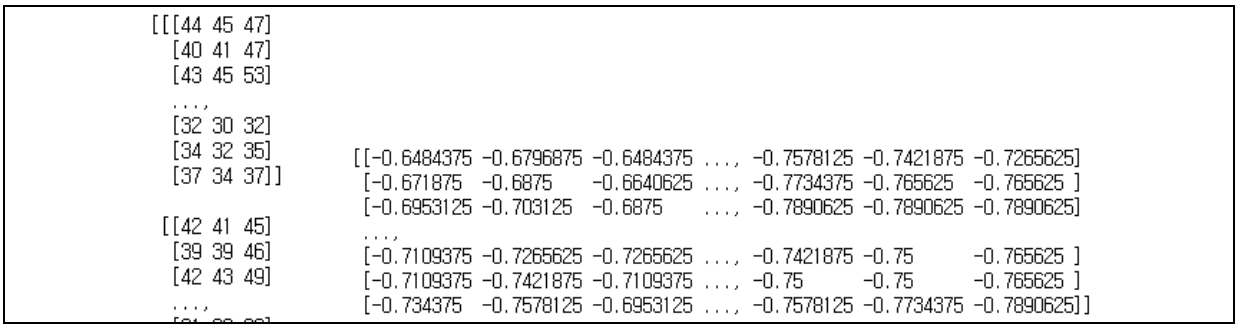

Figure 5 data normalization(left to right)

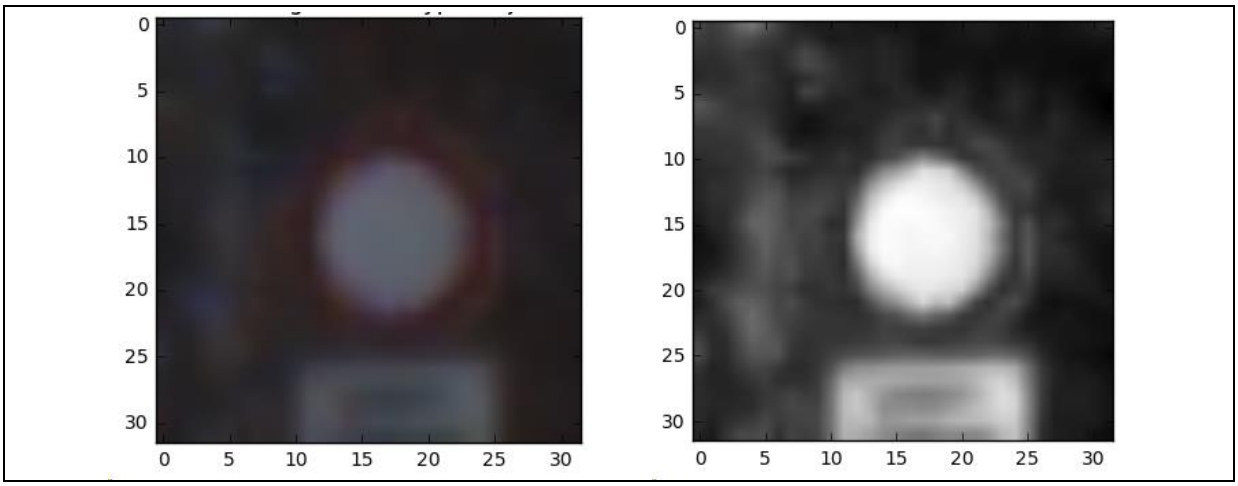

Figure 6 (left) original image, (right) data normalized image

Which are the benefits of grey scaling?

We do not recognize images by color. We do not have to worry though the color of the image is a little different. It does not affect the results. We can remove noise from the color when converting to a grayscale image.

Which are the benefits of implementing normalization?

Accurate calculations are possible when using normalized data. The data has big number, sometimes it give us inaccurate result. So we need to make the data smaller.

while we are training the network, the weights can converge very well to the optimal value. If an ellipse is plotted on the data graph, it can be difficult to converge from the optimized values when trying to train.

2.2 Q2

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	32x32x1 grayscale images
Convolution1 5x5x1x12	1x1 stride, VALID padding, outputs 28x28x12
RELU	
Max pooling	2x2 stride, outputs 14x14x12
Convolution2 5x5x12x32	1x1 stride, VALID padding, outputs 14x14x32
RELU	
Max pooling	2x2 stride, outputs 5x5x32
Fully connected 0	flat to outputs 800
RELU	
dropout	
Fully connected 1	outputs 512
RELU	
dropout	
Fully connected 2	outputs 256
logits	outputs 43

2.3 Q3

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.

type of optimizer use AdamOptimizer algorithm

```
# Loss and Optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y_onehot))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

Figure 7 AdamOptimizer algorithm

the batch size is 128 for memorys allocation. number of epochs is 20 or less. learning rate is 0.01. dropout is 0.5 when it train.

2.4 Q4

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:

```
### Train your model here.
### Calculate and report the accuracy on the training and validation set.
### Once a final model architecture is selected,
### the accuracy on the test set should be calculated and reported as well.
### Feel free to use as many code cells as needed.

#This is test set
print_stats(sess, X_train, y_train, cost, accuracy, epoch = 'Train data')
print_stats(sess, X_valid, y_valid, cost, accuracy, epoch = 'Valid data')
print_stats(sess, X_test, y_test, cost, accuracy, epoch = 'Test data')
```

```
epoch: Train data, loss : 0.004849292803555727, validation accuracy: 0.998878538608551
epoch: Valid data, loss : 0.2272847592830658, validation accuracy: 0.95555579662323
epoch: Test data, loss : 0.3202013671398163, validation accuracy: 0.9428350329399109
```

Figure 8 My final model results

If an iterative approach was chosen:

1) **What was the first architecture that was tried and why was it chosen?**

I tried the network that you gave me and added some drop out layers. Also used batches

2) **What were some problems with the initial architecture?**

Data is inserted into the network in the class hierachy.

3) **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 adopted a random mixed order.

4) **Which parameters were tuned? How were they adjusted and why?**

learning rate was too big! so I reduce it to 0.01

5) **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?**

When I try to train the network without dropout, it get overfitting. Because it train for training data perfectly. So if we want to use it for other datas. We need to add dropout

3 Test a Model on New Images

3.1 Q1

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 9 my test set images

3.2 Q2

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
Stop Sign	Stop sign
Turn left ahead	Turn left ahead
Yield	Yield
Turn right ahead	Turn right ahead
Bumpy road	Bumpy road

Figure 10 my test set prediction result

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%.

My five images can be divided into those with complex background images and those without. The images 1, 2, and 4 with a lot of edges in the background measure the possibility of other classes as follows. Of course, we have recorded a very high probability of accuracy, but we can see that there are more possibilities for the other classes than there is no background.

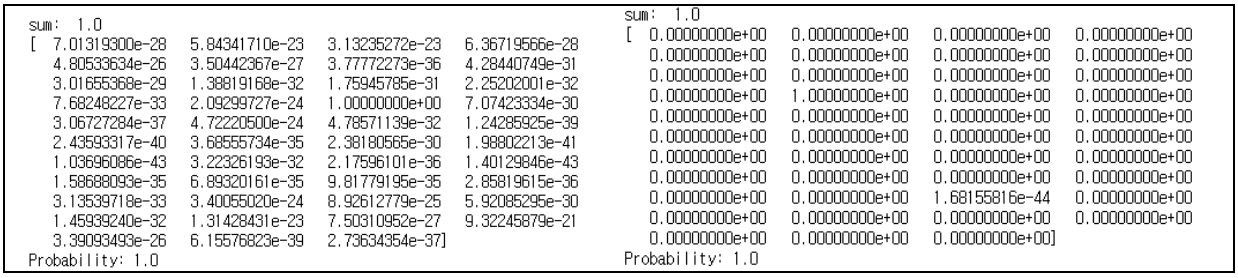

Figure 11 Predictions for images with backgrounds(right) and predictions for images without backgrounds(left)

compare the accuracy to the accuracy on the test set

The images I got were collected with no noise and looking at the front. Because of this, the accuracy is very high. I also tested it with other images. In 32x32 images, the letters on the speed limit sign tend to be invisible and did not detect it properly. It also showed patterns that could not be properly detected even when there was a lot of noise. This tells us that the test set we use is a very clean dataset.

3.3 Q3

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)

I use a softmax function using numpy. It work very well.

```
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()

pred_classid = sess.run(logits, feed_dict={x:imgs, keep_prob:1})

for pred, __y in zip(pred_classid, _y):
    print('sum: ',softmax(pred).sum())
    print(softmax(pred))
    print('Probability:',softmax(pred)[__y])
```

Figure 12 probabilities calculation code

Probability	Prediction
1.0	Stop sign

0.999995	U-turn
1.0	Yield
.0.999986	Bumpy Road
1.0	Slippery Road

Figure 13 probabilities calculation result

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

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

This is my result using the function. These pictures are variations from the same picture. Twelve 14x14 images through the first convolutional network. It take features form the traffic sign image. Because the network is trained!

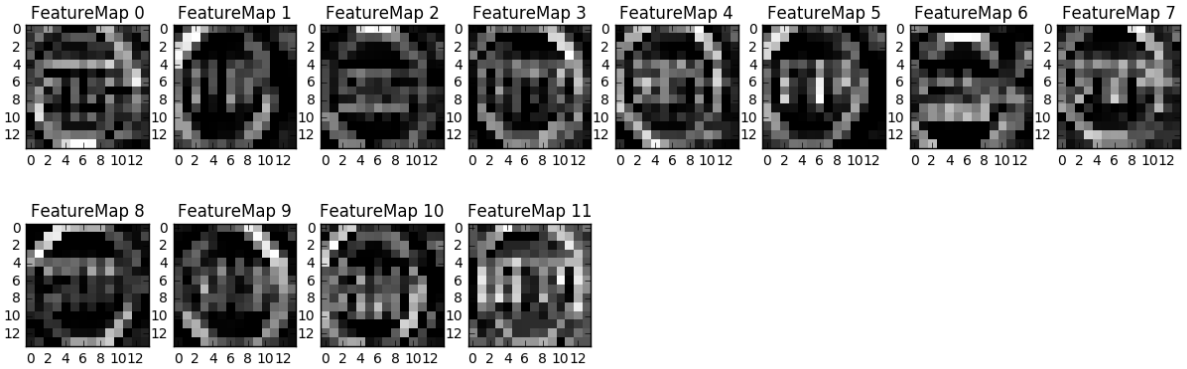

Figure 14 feature maps of conv1