Traffic Sign Classification Project Writeup

1.Dataset Exploration

a.load the training, validation and test dataset, and show the dimension and size of each dataset.
each input data is 32*32 pixel, and with 3 RGB colors.
b. visualize training dataset, and print out the actual image and labels.
c. draw bar charts to visualize the distribution of training dataset, and validation dataset. I can see that certain labels have very low counts.

2.Design and Test a Model Architecture

The architecture I used is similar to LeNet-5 model, as what the course materials presented. As suggested, I first convert image to graysacle, and then normalize the image pixel value to be between -1 and 1.

But as I tried more experiments, I realized some key changes have to be made.

The major differences are :
1. I used 3 convolutional layers instead of 2 to improve the validation accuracy.
2. increase the depth of each convolutional layers to capture more features in this dataset
3. changed the 1st and 2nd fully-connected layer to 512 and 1024, as the num of categories increased from 10 to 43, so the original size of 80 is not big enough.
4. add dropout to fully connected layers, except for the last one.(very important)
5. Even with smaller filter size 3*3, it can out perform the original LeNet-5.

Here are the actual parameters I used in the final report

layer	parameters and details
input	32*32 grayscale image
initialization parameters	mu=0, sigma =0.1, rate=0.001, EPOCHS=100, BATCH_SIZE=64
Conv layer 3*3, depth=12	1x1 stride, valid padding, outputs 28x28x12
RELU	
Max pooling	2x2 kernel size, 2x2 stride, outputs 14x14x12
Conv layer 3*3, depth=32	1x1 stride, valid padding, outputs 12x12x32
RELU	
Max pooling	2x2 kernel size, 2x2 stride, outputs 6x6x32
Conv layer 5*5, depth=64	1x1 stride, valid padding, outputs 2x2x64

RELU	
Flattern	inputs 2x2x64, outputs 256
Fully connected	inputs 256, outputs 512
RELU	
dropout	drop out prob=0.6
Fully connected	inputs 512, outputs 1024
RELU	
drop out	drop out prob=0.6
Fully connected	inputs 1024, outputs 43

All the experiment details are documented in the LOG sections in the python notebook.
(I just copy that section over.)

1.started with same architecture and similar parameters used in the lecture about LeNet. (but forgot to normalize the validation dataset), the validation accuracy was just 0.82
Also tried various parameters of LeNet, such as batch_size, dropout, but still no improvements.

2.realized the mistake, and changed to use normalized validation dataset and standard LeNet params,
(learning rate =0.001,mu=0, sigma=0.1, 1st conv layer: depth_1=6, filter size =5*5, 2nd conv layer: depth_2=16, filter size =5*5, validation accuracy: 0.91~0.92 after epoch 40)

3.tried to use larger filter size for conv layer.
(learning rate =0.001,mu=0, sigma=0.1, 1st conv layer: depth_1=6, filter size =7*7, 2nd conv layer: depth_2=16, filter size =5*5, validation accuracy: 0.925~0.937 after epoch 40, but accuracy had some flucuation)

4.tried to increase the depth of each convolutional layer. (depth_1=64, depth_2=32, slightly better, validation accuracy: 0.929~0.942 after epoch 40)

5.tried to make the depth of 1st convolutional layer smaller than 2nd layer. (depth_1=32, depth_2=64, nothing changed; similarly, tried depth_1 =64 and depth_2 =64, nothing changed).

6.tried to reduce the batch size (BATCH_SIZE=32, validation accuracy decreased.)

7.kept the same depth=64 for both layers, but increase 1st conv layer: filter size =9*9, 2nd conv layer: filter size =7*7, validation accuracy still looks similar to Experiment 4, validation accuracy: 0.925~0.943 after epoch 60)

8. 1st conv layer: depth_1=16, filter size =9*9, 2nd conv layer: depth_2=32, filter size =9*9, validation accuracy: 0.928~0.941 after epoch 60, more stable

9.tried to adjust learning rate with 0.0009 and 0.0008, and increase epoch=100, nothing changed.

10.no drop out after fc layer, no changes to other params. saw more fluctuations in validation accuracy. No good. Decided that I still need to keep the drop out layer to avoid overfitting.

11.Didn't want to supply more augmented data as training. So decided to go and add another convolutional layer, also reduced the filter size of each layer. layer1=3*3, layer2=3*3, layer3=5*5 (see function LeNet3). and the performance suddenly improved a lot. validation accuracy never drops below 0.931 after epoch 80, and it is kinda stable at 0.94~0.96

12. I can try with larger reception fields or more layers, but I think for this project, the accuracy is good enough. Final test data accuracy is 0.939

Results:

```
Accuracy on testing data= = 0.937
```

3.Test a Model on New Images

I downloaded 8 images from internet.
a. Most of the new images have correct predictions.
But in some repeated runs, I noticed that there is one new images is kinda similar to a “weight limit” sign but sometimes was predicted to be a “Speed limit 20km/h” sign. This is mostly due to this new image was in a category which doesn’t have enough variation of training data. and this specific image might not be very similar to other variation of images in this category.

b. also calculated the top5 softmax probabilities
c. Also tried the optional tasks to visualize each tensor’s state.

4. future suggestions

a. could try with larger reception fields.
b. supply more training data by using data augmentation, especially for those categories with less input training data.
c. try with l2 regularization to convolutional layer and fully-connected layers if the model is overfitting.
d. use more convolutional layers, by changing the padding style from “valid” to “SAME”; this will keep the input and output has same dimensions.
e. try to use different model architecture