Project 2 Traffic sign classifier

October 12, 2017

1 RUBIC POINT

Project 2 and corresponding course talked about use Tensorflow to implement traffic signs classification use LeNet architecture.

1. Provide a basic summary of the data set.

   The size of training set is $39200 \times 32 \times 32 \times 3$, the validation size is $39200 \times 32 \times 32 \times 3$ and the test size is $39200 \times 32 \times 32 \times 3$. The shape of each traffic sign image is $39200 \times 32 \times 32 \times 3$. The unique classes in the data set is 43. The training images with each class are illustrated in Fig 1.1

2. Include an exploratory visualization of the dataset. From Fig 1.2, we can tell the distribution of each class is not equal. This may bring the problem of bias for the network work.

3. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques?

   Generally speaking, I mainly consider normalize or scale the image data. More specifically, I used two different normalization methods, min-max scaling and zero-score scaling. I think the color information is useful, therefore, the grayscale is not employed in my implementation. I tried directly using the original image, using min-max scaling and zero-score scaling. The results can be significant different. Directly using original training data achieved less than 80% training accuracy. However, using zero-score I achieved 99% training accuracy and 93% testing accuracy. The improvement is significant.

Figure 1.1: Illustration of the original training images.

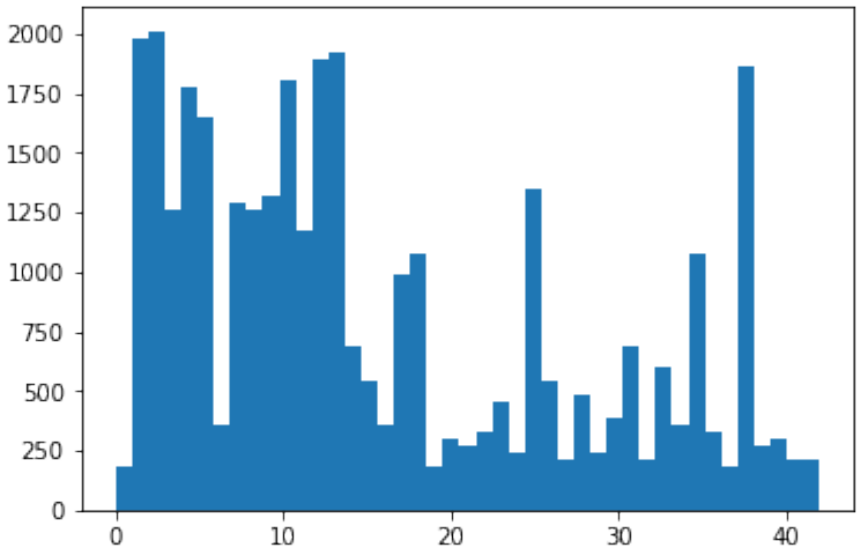

Figure 1.2: Histogram figure of training data

Figure 1.3: Normalize the training image

4. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.

   My final model is almost exactly the same with classic LeNet-5 layers. More specifically, I have the following layers:

Layer	Description
Input	$32 \times 32 \times 3$
Convolution 3×3	1x1 stride, same padding, filter size: $5 \times 5 \times 3 \times 6$
Activation layer	ReLu layer
Pooling Layer	Max pooling layer 2x2 stride
Dropout Layer	Prob = 0.5
Input	$32 \times 32 \times 3$
Convolution 3×3	x1 stride, same padding, filter size: $5 \times 5 \times 6 \times 16$
Activation layer	ReLu layer
Pooling Layer	Max pooling layer 2x2 stride
Dropout Layer	Prob = 0.5
Fully connected (dense) layer	input size 400, output size 120
Fully connected (dense) layer	input size 120, output size 84
Fully connected (dense) layer	input size 84, output size 43

5. 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.

   To train the model, I used an Adam optimizer, and set epoch size as 20, batch size as 128, the initial learning rate as 0.001.

6. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

   My final model results with each epoch are: EPOCH 1: Training Accuracy = 0.742, Validation Accuracy = 0.702
   EPOCH 2: Training Accuracy = 0.879, Validation Accuracy = 0.831
   EPOCH 3: Training Accuracy = 0.922, Validation Accuracy = 0.876
   EPOCH 4: Training Accuracy = 0.948, Validation Accuracy = 0.902
   EPOCH 5: Training Accuracy = 0.959, Validation Accuracy = 0.923
   EPOCH 6: Training Accuracy = 0.962, Validation Accuracy = 0.912
   EPOCH 7: Training Accuracy = 0.975, Validation Accuracy = 0.926
   EPOCH 8: Training Accuracy = 0.981, Validation Accuracy = 0.929
   EPOCH 9: Training Accuracy = 0.986, Validation Accuracy = 0.947
   EPOCH 10: Training Accuracy = 0.980, Validation Accuracy = 0.932
   EPOCH 11: Training Accuracy = 0.990, Validation Accuracy = 0.955
   EPOCH 12: Training Accuracy = 0.990, Validation Accuracy = 0.950
   EPOCH 13: Training Accuracy = 0.992, Validation Accuracy = 0.956
   EPOCH 14: Training Accuracy = 0.993, Validation Accuracy = 0.956
   EPOCH 15: Training Accuracy = 0.992, Validation Accuracy = 0.955

EPOCH 16: Training Accuracy = 0.994, Validation Accuracy = 0.957
EPOCH 17: Training Accuracy = 0.994, Validation Accuracy = 0.957
EPOCH 18: Training Accuracy = 0.995, Validation Accuracy = 0.957
EPOCH 19: Training Accuracy = 0.995, Validation Accuracy = 0.955
EPOCH 20: Training Accuracy = 0.994, Validation Accuracy = 0.956

I only test one final testing accuracy, and the result accuracy is: 93.73713382632602%

- What was the first architecture that was tried and why was it chosen?

  I tried the regular LeNet without drouput. I choose it as the most successful architecture in digits classification.

- How was the architecture adjusted and why was it adjusted?

  From my experience, the most significant effect is using dropout layer. Without dropout layer, my testing accuracy is around 90%. In other words, the testing accuracy is improved higher than 93% with appending two dropout layers. I did not change the activation function since ReLu has been proved as the most effective activation function.

- Which parameters were tuned? How were they adjusted and why?

  I tuned epoch numbers, with the increase of epoch numbers the accuracy can be improved. Another parameter is learning rate. It is important because we have to trade-off between the converge speed and training loss. The rule of thumb is 0.001.

- 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?

  In my opinion, Convolutional layer works since it has much less parameters than dense layer or regular fully layer. In this way we can overcome overfitting problem. Dropout layer help the training because it works like an ensemble filter to prevent training highly depends on some particular parameters.

7. If a well known architecture was chosen
   - What architecture was chosen? I employed LeNet architecture, which constructed by 2 convolutional layers and 3 fully connected layers.
   - Why did you believe it would be relevant to the traffic sign application? Traffic sign classification is very similar with digits classification. And convolutional layers are good for solve the image classification problem.

8. Test a Model on New Images
   - 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. The randomly selected figures are illustrated in Fig 1.4 and the detected results are shown in Fig 1.5.

Figure 1.4: Illustration of 5 Germain traffic signes.

Figure 1.5: Illustration of 5 Germain traffic signs and detected results.

Image	Prediction
No passing	No assing
Speed limit (60km/h)	Speed limit (60km/h)
Go straight or right	Go straight or right
No passing	No passing
Dangerous curve to the right	Dangerous curve to the right

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

Probability	Prediction
1	No passing
0	Vehicles over 3.5 metric tons prohibited
0	Speed limit (50km/h)
0	No passing for vehicles over 3.5 metric tons
0	No vehicles

Probability	Prediction
0.999	Speed limit (60km/h)
0.001	Speed limit (80km/h)
0	Speed limit (50km/h)
0	Ahead only
0	No passing

Probability	Prediction
0.837	Go straight or right
0.154	Keep right
0.009	Turn left ahead
0	End of all speed and passing limits
0	End of no passing

Probability	Prediction
1	No passing
0	No passing for vehicles over 3.5 metric tons
0	Speed limit (60km/h)
0	Vehicles over 3.5 metric tons prohibited
0	Slippery road

Probability	Prediction
1	Dangerous curve to the right
0	Children crossing
0	Slippery road
0	Road work
0	No passing

2 POSSIBLE IMPROVEMENTS

There is no difference between deep learning and more conventional "neural network". I would like to try deeper layers (with more than 10 convolutional layers) or new architecture introduced in recent years such as ResNet to improve the classification rate.