Traffic Sign Classifier

Done by: Wael Farag

1. Training Data Pre-Processing

The following steps describes the implemented pre-processing steps in order of execution:

1. **Normalization (color)**: this is done for color images using the "normalize_color_image()" function. It is simply implementing a Min-Max scaling for color image data.
2. **Converting the color training image to gray**: this is done in function "Convert_Color2Gray" using the OpenCV function "ctvColor".
3. **Normalization (gray)**: this is done for gray images using the "normalize_grayscale ()" function. It is simply implementing a Min-Max scaling (between 0.1 and 0.9) for gray image data.
4. **Filtering the noise**: this is done in the "Filter_Blur()" function using the OpenCV function "GaussianBlur" which executes the Gaussian filter algorithm. A kernel size of 5 has been selected.
5. **Shuffling the training data**: this is done once each training epoch.

All the above steps as been tried. Pre-processing using color images has been tried. Also pre-processing using Filter_Blur() is also tried. We found out that using Gray images and "No Blur Filtering" give the best performance as per our many trial-and-error endeavors (which supports the results in [1]). However, I believe that converting images to gray makes the data loose vital information about the traffic signs. Therefore, more trials need to be carried out using color images with deeper and wider networks.

Finally, the actually used pre-processing pipeline is: Color-Image → Convert-to-Gray → Normalization → Shuffling.

2. The Model Architecture

The implemented neural network model is a modified version of LeNet architecture [2] and given the name WAF-LeNet. Table 1 below describes the architecture in details:

Table 1: WAF-LeNet Architecture.

#	Layer	Size/Output	Parameters	Comment
1	Input	32x32x1	----------	Gray scale
2	Convolutional #1	28x28x6	Kernel: 5x5, Strides: 1x1, Padding: Valid	With bias (6)
3	Activation #1	28x28x6 = 4704	RELU	

4	Pooling #1	14x14x6	Max, K: 2x2, Strides: 2x2, Padding: Valid	
5	Convolutional #2	10x10x16	Kernel: 5x5, Strides: 1x1, Padding: Valid	With bias (16)
6	Activation #2	10x10x16 = 1600	RELU	
7	Pooling #2	5x5x16	Average, K: 2x2, Strides: 2x2, Padding: Valid	
8	Flat	400	----------	Just flatten the input
9	Fully Connected #1	300		With bias (300)
10	Activation #3	300	RELU	
11	Drop-out	300		Keep Probability: 0.75
12	Fully Connected #2	150		With bias (150)
13	Activation #4	150	RELU	
14	Drop-out	150		Keep Probability: 0.75
15	Fully Connected #3	43	One-hot	The number of Traffic Sign Classes

2 Drop-out layers are added and the 1st and 2nd fully connected layers are widened. Also, average Pooling are used instead of the Max-Pooling is layer 7.

3. The Model Training

- The WAF-LeNet Model is trained using the parameters listed in Tables 2, 3 and 4.

Table 2: WAF-LeNet training parameters (Learning Rate).

Parameter	Value	Comment
Learning Rate	0.001	For epochs: 000 - 100
	0.0005	For epochs: 101 - 125
	0.0003	For epochs: 126 - 150
	0.0002	For epochs: 151 - 175
	0.0001	For epochs: 176 - 200

Table 3: WAF-LeNet training parameters (PKeep).

Parameter	Value	Comment
	0.75	For epochs: 000 - 100

Drop-out layers Keep Probability (PKeep)	0.75	For epochs: 101 - 125
	0.85	For epochs: 126 - 150
	0.9	For epochs: 151 - 175
	1.0	For epochs: 176 - 200

Table 4: WAF-LeNet training parameters (others).

Parameter	Value	Comment
Batch Size	128	For epochs: 000 - 200
Epochs	200	The whole training

- The training results are listed in Table 5 below:-

Table 5: WAF-LeNet training results.

Parameter	Value	Comment
Training-data Accuracy	100.0%	highest reached
Validation-data Accuracy	96.4%	highest reached
Testing-data Accuracy	94.5%	highest reached

- Attached with this report a file that contains the training progress in terms of accuracy improvement, as well as a folder containing the saved model parameters (weights) that resulted in the performance shown in Table 5, so that the model can be fully replicated.

4. Solution Approach

The training of the WAF-LeNet has been carried-out through several trials to the results shown in Table 5. The following observations has been collected during the training process:

1. I tried to train the network using the RGB images after being normalized, however, I didn't good enough results, and the validation accuracy didn't cross the 91% level.
2. I have then switched to using the normalized converted gray-image data with more successful results with the validation accuracy reached the 93% but didn't go much further beyond that.
3. Using Gaussian filtering for the both Gray and RGB images didn't improve the accuracy, in contrary, it worsen the accuracy much further. For this reason, this technique has been avoided.
4. In order to improve the performance further, two drop-out layers have been added as shown in Table 1 as well as increasing the breadth of the 1st and 2nd fully connected layers. This approach has significantly improved the results and the validation accuracy crossed the 95% mark.
5. For further improvement, the learning rate as well as the keep probability of the drop-out layers have been updated according to the profiles shown in Tables 2 & 3. This approach has significantly improved the validation accuracy to reach 96.4% as shown in Table 5.
6. The WAF-LeNet after training has been applied on the testing data resulting in 94.5% accuracy.

5. Testing on images from the web

10 new German traffic sign images have been downloaded from the web. Samples of these raw images are shown in Figure 1 below.

Figure 1 Raw traffic sign samples from the web.

The WAF-LeNet after being trained has been tested on these 10 raw images (after being resized to 32x32 using openCV resize function) with very low results achieved (only 50%). Then, I have implemented some processing on the images mainly cropping, centering and rotation as shown in Figure 2. After that the images have resized to 32x32 and fed to the network with 100% results in testing accuracy achieved.

Figure 2 Cropped and centered traffic sign samples from the web.

Compared to the results achieved on the original testing data set of 94.5% (Table 5), we can conclude the following:

a) If the tested images are well-centered and well-cropped (which means the traffic sign fills 75% of the image or more) as shown in Figure 2, the WAF-LeNet in its current state works very well (no over-fitting and no under-fitting).
b) However, if the images are not well-processed (raw) like the ones shown in Figure 1, the WAF-LeNet in its current state performs poorly with 50% performance or so (i.e. under fitting) which requires that much more training data is needed in order to significantly increase the accuracy.

6. Shortcoming of the implemented approaches

The following list summaries the identified shortcomings:

1. Needs to investigate using the RGB further. Using grayscale images deprive the network from precious information (like colors) which I believe is very important in detecting traffic signs.

2. More training data is needed especially for classes that have low number of samples like Class 0, Class 41 and Class 42.
3. There is a very big difference between classes in terms of the number of available samples in the training data which reflects in the tendency of recognizing the classes with larger number of samples better.

7. Suggested Improvements

The following list summaries the suggested improvements:

1. Increase the depth of the network and use RGB images instead of gray scale ones (going to work on it in the coming week).
2. Try to equalize the number of samples in each class by generating more data from the available data by the process of data augmentation (skewing, rotation, noise addition … etc.).
3. Apply some image improvement processing on the data like centering, cropping, filtering bright and dark spots … etc.

References

1. Pierre Sermanet and Yann LeCun**, "**Traffic Sign Recognition with Multi-Scale Convolutional Networks", The 2011 International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, DOI: 10.1109/IJCNN.2011.6033589
2. LeCun, Y, Bottou, L, Bengio, Y, and Haffner, P, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, 86(11):2278–2324, November 1998.