Write-up Report of Project 2

Traffic Sign Classification jiyang_wang@yahoo.com

1. Goals

- Develop deep neural networks and convolutional neural networks to classify traffic signs. Specifically, train a model to classify traffic signs from the **German Traffic Sign Dataset**.
- Reach and go beyond the minimum requirement for validation accuracy rate of 0.93. Try various techniques to increase the out-of-sample test accuracy and the accuracy of identifying the real-world traffic signs.
- Implement all the required functions listed in Jupyter Notebook cells. In addition to Traffic_Sign_Classifier.ipynb notebook file, this report provides more explanations of the functions implemented.

2. Files Submitted

The following files are submitted:

1. The Traffic_Sign_Classifier.ipynb notebook file with all questions answered and all code cells executed and displaying output.
2. An HTML export of the project notebook with the name report.html.
3. Any additional datasets or images used for the project that are not from the German Traffic Sign Dataset.
4. The write-up report

3. Dataset Exploration

Data Summary

Image Shape**: (Width, Height, Channels) = (32, 32, 3)**

Training Set: **34799** samples

Validation Set: **4410** samples

Test Set: **12630** samples

Number of classes = **43**

Average number of training examples over all classes = **809**

Visualization of dataset

From the histogram below we can see the example distribution over the classes.

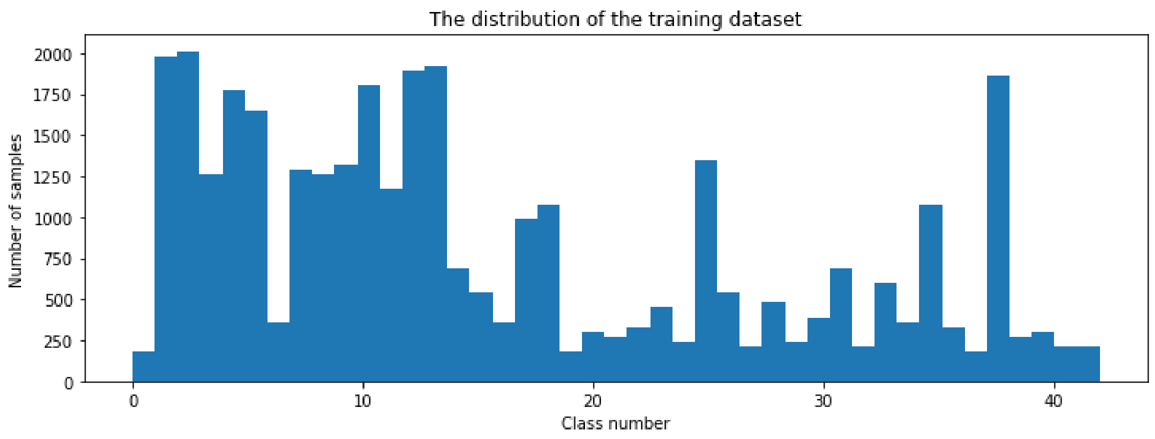


Figure 1. Distribution of Examples of Training Dataset over Classes

This distribution of training examples is not friendly to the classifier since some of the traffic signs do not have enough examples in the training dataset for the model to be well trained. We'll address this issue later on when we augment the dataset.

The two diagrams below demonstrate the sample distribution of the training, validation and test dataset.

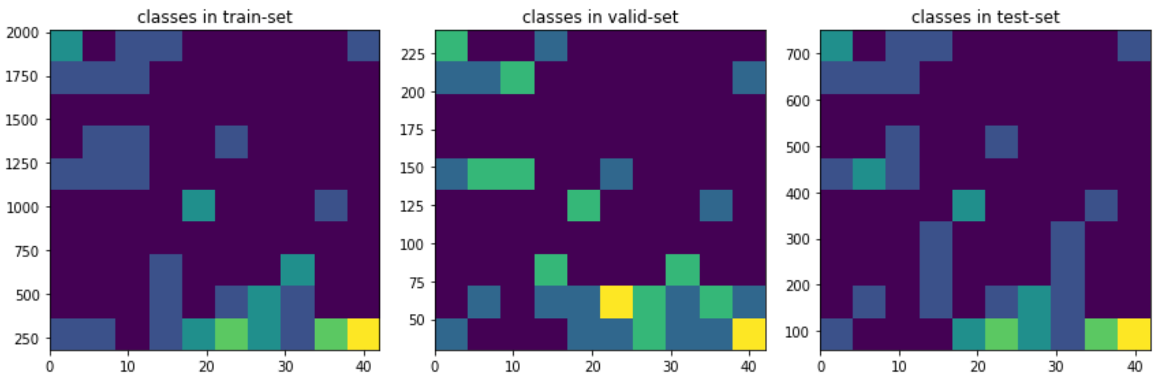


Figure 2. Example Distribution Pattern of Training, Validation and Test Dataset

We can see that the example distribution of the validation and, more importantly, the test dataset is largely in line with that of the training dataset even though not exactly the same, giving us reasonable credibility of the validation and testing of the learning results.

4. Dataset Preprocessing

Grayscale and normalization are the two techniques used in this project. The input images in the dataset are color ones, but the way the model learns how to identify traffic signs mainly lies on the structures rather than the colors of the signs. Plus, contrast, lightness and darkness, etc. of colorful images may have negative effects that disturb the model. I ran the test on learning with un-processed color images and with pre-processed images and the results show that gray scaled and normalized images help improve the test accuracy a little bit (0.915 vs. 0.914 on average) over the initial LeNet-5 model, and the learning time was significantly reduced because the number of channels of the sample images is reduced from 3 to 1 so that the computation work of convolution layer 1 is reduced by 2/3.

Thus, all the follow-on tests in this project only use gray scaled and normalized images as input.

5. Model Architecture

The initial model architecture is copied directly from LeNet as shown below, except that the output layer has 43 instead of 10 outputs. Gaussian connections in the diagram become Softmax classifiers.

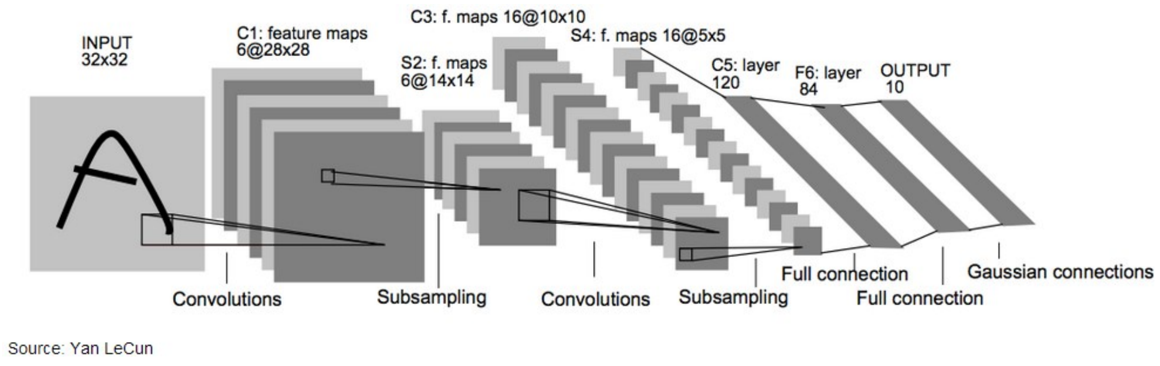


Figure 3. LeNet Model

6. Model Training

Model Learning Parameters and Functions

The model uses Adam optimizer and cross-entropy cost function to learn from data. Activation function at convolution and full-connection hidden layers is simply ReLU and the final layer uses Softmax as the final classifier.

The hyper-parameters are as follows (throughout all the tests):

Learning rate: **0.002**

Epochs: **10**

Batch size: **128**

Dropout Keep Probability: **0.8**

Initial Performance

This model, with the above parameters and original dataset (gray scaled and normalized), achieves:

Validation Accuracy: **0.933**

Test Accuracy: **0.914**

This result has satisfied the minimum requirement of 0.93 of validation accuracy, and it gets reasonable rate of accuracy (4 out of 6) on the real-world traffic signs as shown on next page. However, the test accuracy is not high enough and the model has big room for improvement.

While tuning hyper-parameters is trivial, I focused on the following two solution categories: **enhancing the model**, and **augmenting the dataset**, to increase the accuracy of test and of real-world traffic sign identification.

Solution Approach

To increase the test accuracy and the performance of predicting the real-world traffic signs, I did the following:

Enhancing the Model

1. Connecting both layer S2 and S4 to layer C5 (denoted as S2+S4 => C5)

In traditional ConvNet, the output of the last convolution layer is fed to the

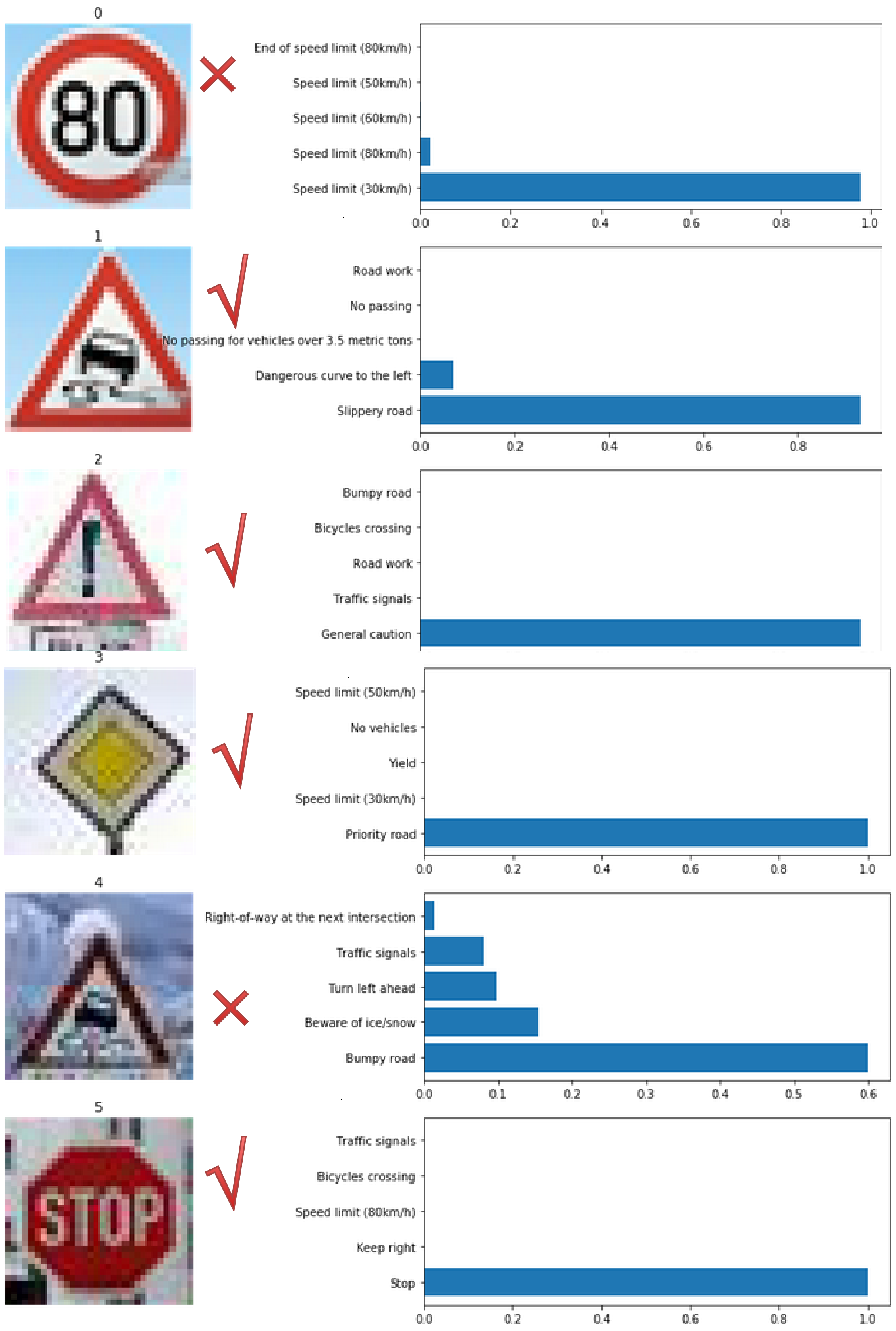


Figure 4. Test Result of LeNet on New Images

classifier, which provides high-level, global features with little precise details. However, the earlier convolution layers draw from the input the low-level features, which tend to be more local, less invariant, and more accurately encode local motifs of the input images. Intuitively, feeding both high-level and low-level features to the classifier should achieve higher accuracy than otherwise. This new model architecture is shown below (source: Traffic Sign Recognition with Multi-Scale Convolutional Networks, by Pierre Sermanet and Yann LeCun).

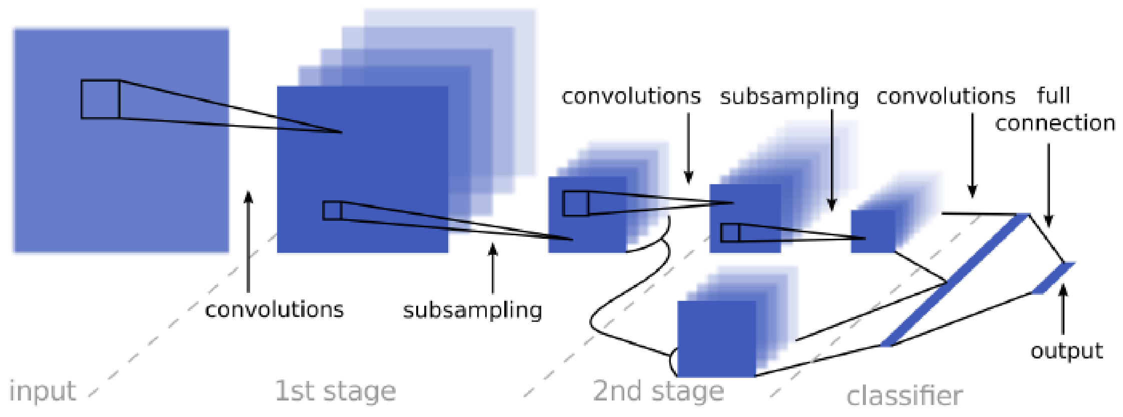


Figure 5. New ConvNet Architecture

The new model increased testing accuracy from 0.914 to 0.923 (average).

2. Increasing the number of filters on layer C1 and C2

Literally, the more features fed to classifier, the higher the accuracy. As such, we can add more filters to convolution layers. The results are (accuracies averaged over 2 to 3 rounds of learning):

No. of filters	LeNet	New Model-1	New Model-2	New Model-3	New Model-4	New Model-5
1st Conv.	6	6	16	32	16	32
2nd Conv.	6	6	6	6	32	32
FC Input	S4 => C5	S2+S4=> C5	S2+S4=> C5	S2+S4=> C5	S2+S4=> C5	S2+S4=> C5
Validation	0.939	0.937	0.946	0.941	0.968	0.955
Test	**0.915**	**0.923**	**0.932**	**0.923**	**0.945**	**0.938**
New Images	4/6	4/6	4/6	4/6	4/6	4/6

In general, test accuracy increases with the number of filters on each convolution layer. However, it seems that having much more filters on the 1st convolution layer than on the 2nd does not improve the performance.

Augmenting the Dataset

Augmenting the training set might help improve model performance. Common data augmentation techniques include rotation, translation, zoom, flips, and/or color perturbation. These techniques can be used individually or combined.

In this project, scaling images randomly with factors of 0.9 to 1.1, combined with random rotation of images in range of [-15, 15] degrees in any directions, is implemented for data augmentation.

To compensate the classes that have much less examples than the average, their images are reproduced a few times depending on how far they are from average. The result of the new dataset distribution is shown below. Apparently, it looks more reasonable than before.

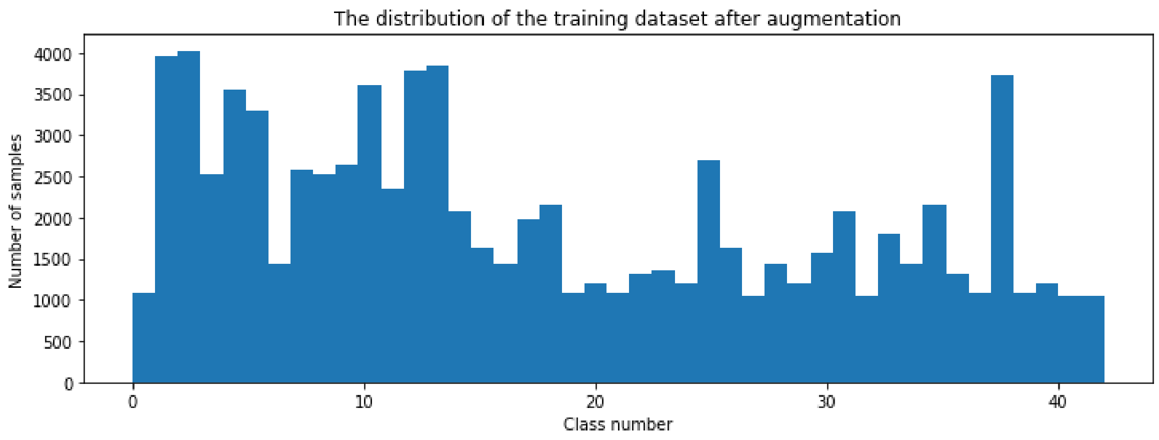


Figure 6. Example distributions of the new augmented dataset

Result: test accuracy is increased remarkably to **0.952**, and validation to **0.973**.

7. Test a Model on New Images

Acquiring New Images

Six images of German traffic sign were copied from Web, shown below after being scaled to 32x32:

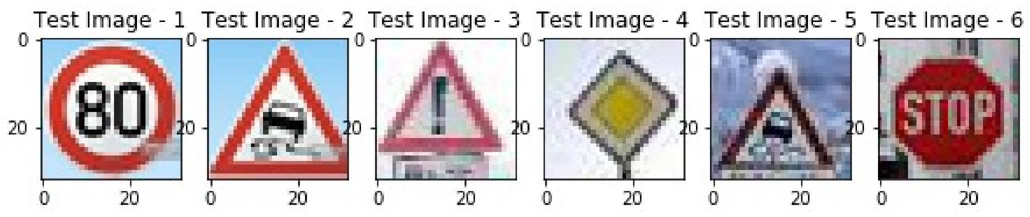


Figure 7. Six new images of German traffic signs copied from Web

Test Image-5 has very poor quality and should be the most difficult one for the model to identify correctly.

Performance on New Images

The model trained by the augmented dataset correctly identified 5 signs out of 6 (83%), which is an improvement (before it was 4 out of 6). The one identified wrongly is the image of "slippery road" that has very poor image quality.

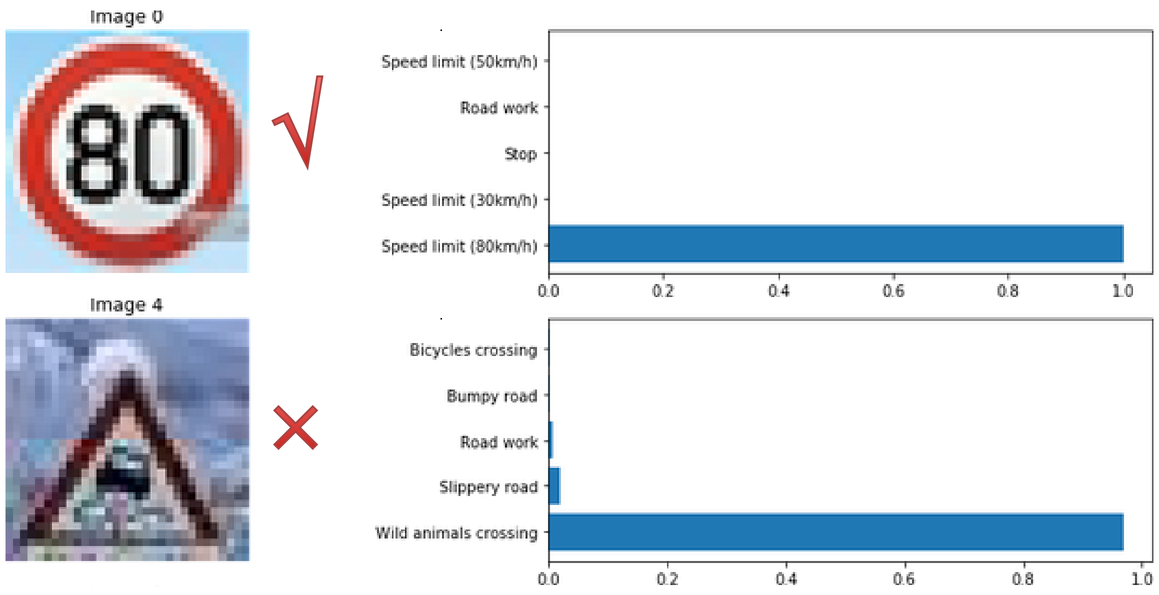


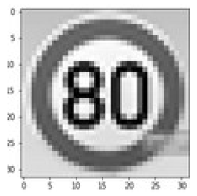
Figure 8. Prediction of 2 new images of German traffic signs

The certainty of all the predictions except that for the 5th image is as high as close to probability of 1.

8. Visualize the layers of the Neural Network

By plotting their feature maps after successfully training, we can see what its feature maps look like and better understand what characteristics of an image the network finds interesting.

This is the image used as stimuli:



The feature maps of the 1st convolution layer:

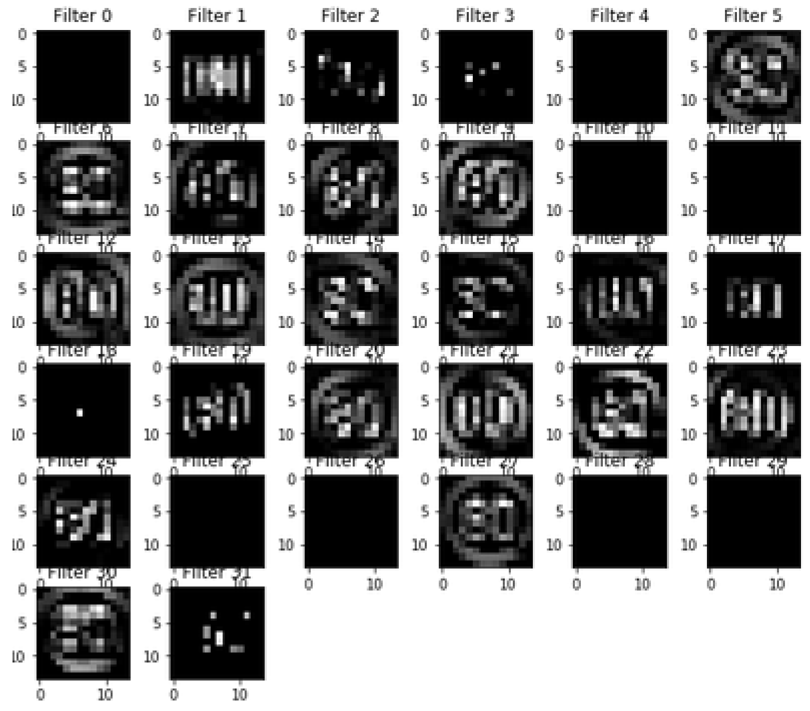


Figure 9. Feature maps of the 1st Convolution Layer

The feature maps of the 2nd convolution layer:

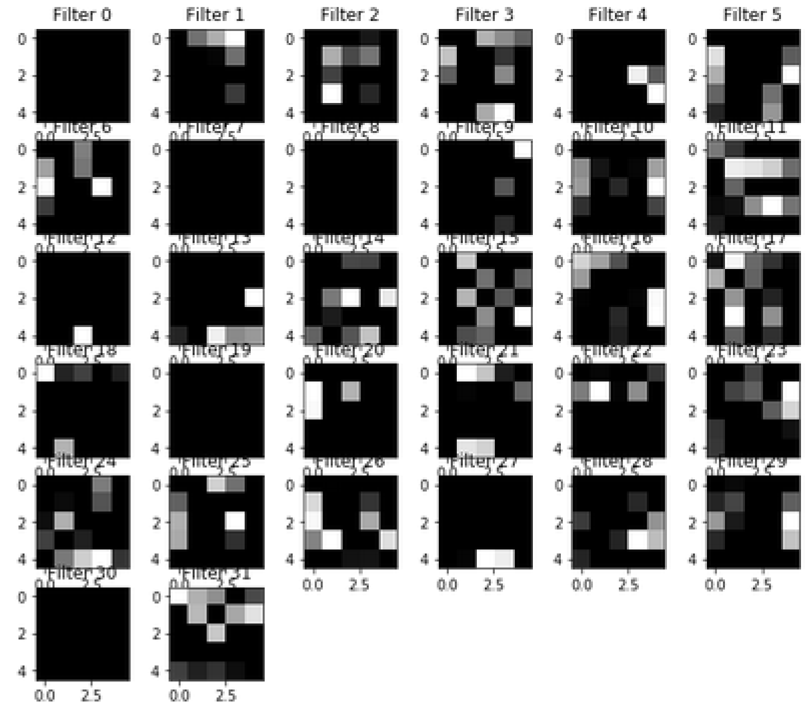
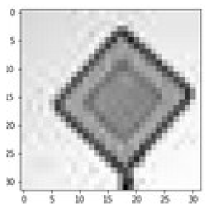
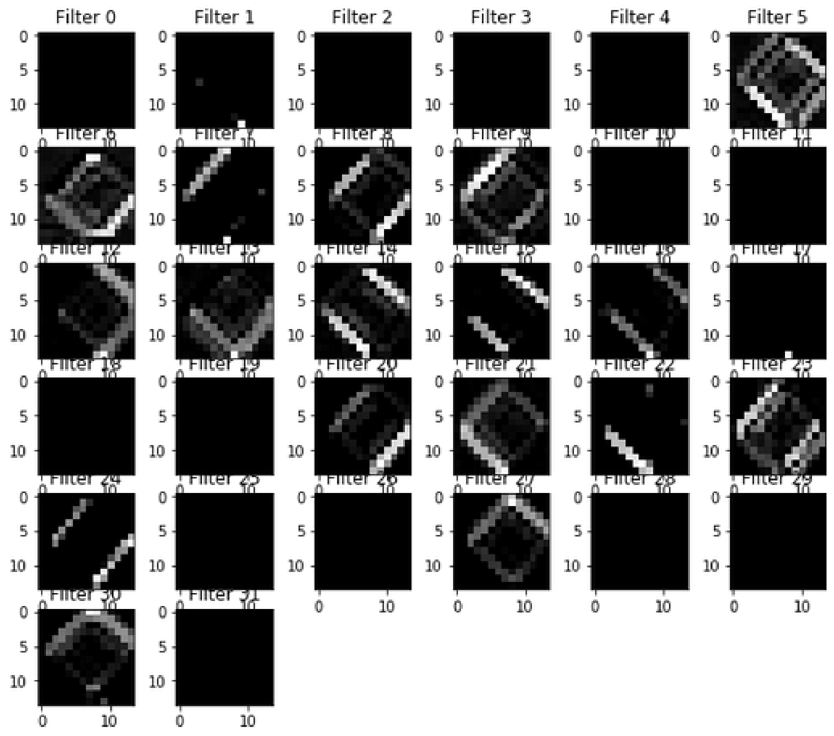
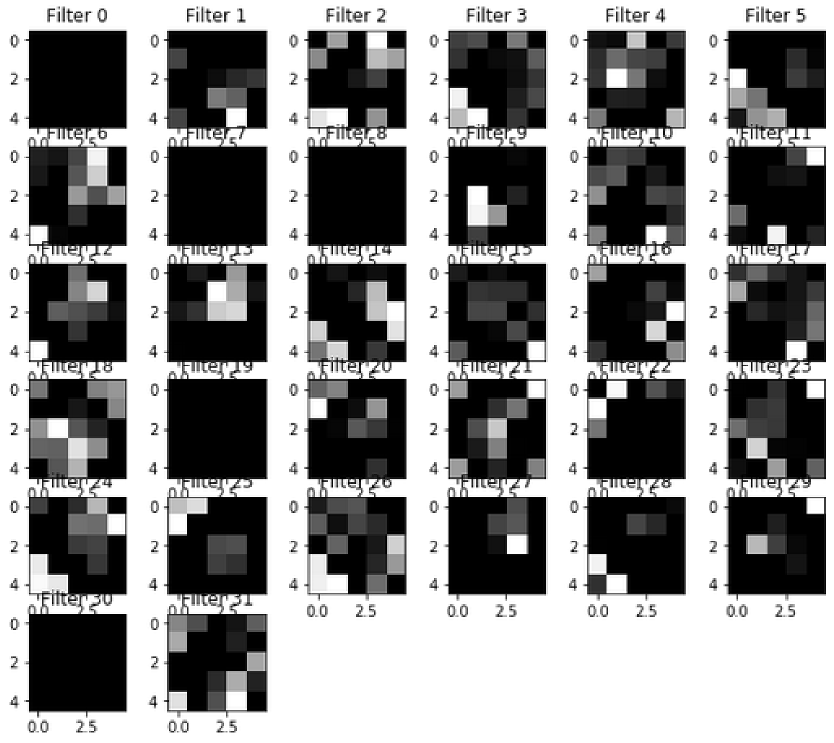


Figure 10. Feature maps of the 2nd Convolution Layer



Its feature maps:





For a sign, maybe the feature maps of the inner network react with high activation to the sign's boundary outline or to the contrast in the sign's painted symbol.

From the feature map visualization of two traffic sign images we can observe that S2 layer (the outcome of the 1st convolution) seems to be interested in the edges and contours of the traffic signs, and S4 layer (2nd convolution) not only in higher level of the latent structures of the traffic signs but also in further details such as corners (for the 2nd image, and not for the 1st image which does not have corners).

This observation suggests that connecting S2 to the input of the full-connection classifier, in addition to S4, may improve the test accuracy as S2 does provide informative hints (e.g., edges and contours) to the classifier even though this will add significant amount of computation work.

It should be pointed out that S2 and S4 layer produce a few full-black feature maps from both traffic sign images and black feature maps contribute little to classification. I'm not sure how this happens but it's worth of digging into this phenomenon if time permits. (hint: applying dropout to convolution layers).

Test results demonstrate that more filters on convolution layers can improve the accuracy.