CarND-P2-Report Traffic Sign Classification

1. Dataset Summary and Exploration

1.1. Basic Summary of Dataset

Number of training examples = 34799

Number of validation examples = 4410

Number of testing examples = 12630

Image data shape = (32, 32)

Number of classes = 43

1.2. Explorative Visualization of Dataset

In cell 120 of Jupyter notebook, we can visualize the frequency by Traffic sign ID in training dataset.

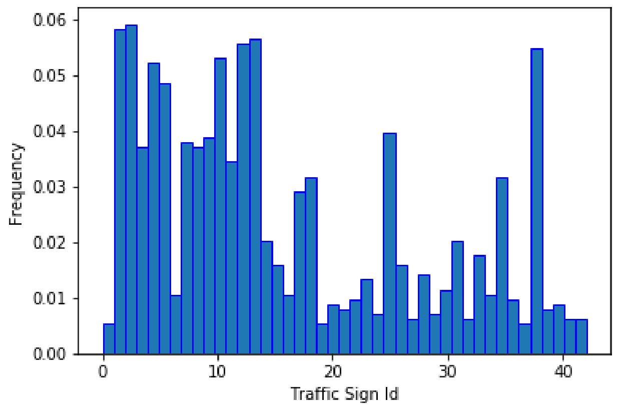


Figure 1. Frequency (Percentage) by Traffic Sign ID in Training Dataset

In cell 121-122, I imported the “signnames.csv” and found one example for each traffic sign ID.

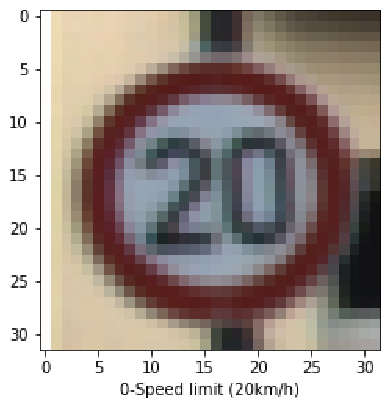


Figure 2. Visualization of Traffic Sign 0

2. Design and Test a Model Architecture

2.1 Data Preprocessing

In cell 123-124, I converted the images from RGB to grayscale using package cv2. This is because the color information may not be useful in this task. Using 1 channel instead of 3 channels may also reduce the computational cost in deep learning neural network.

The following is an example of "End of no passing" before and after grayscaling:

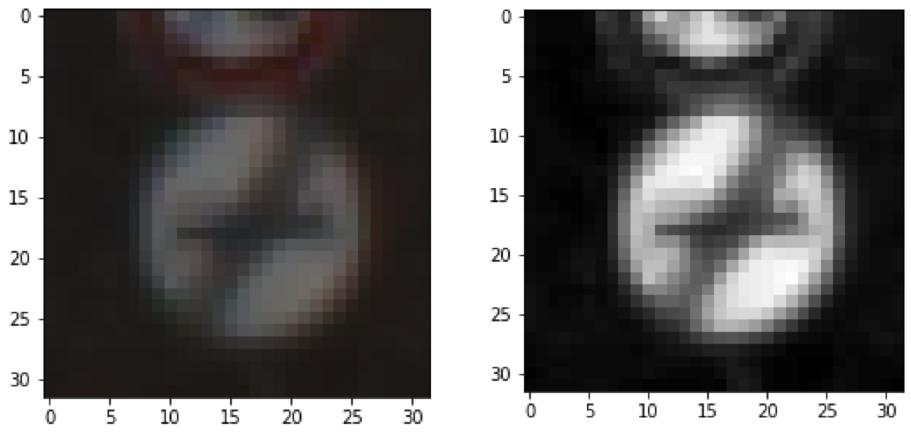


Figure 3. "End of no passing" before and after Grayscaling

2.2 Model Architecture

The basic architecture follows the architecture of LeNet, which has been shown to be powerful. The details are as follows:

Layer 1: Convolutional. Input 32x32x1, the filter 5x5x1, stride = [1,1,1,1], the output 28x28x6.

Activation: RELU

Pooling: The output 14x14x6.

Layer 2: Convolutional. Input 14x14x6, the filter 5x5x1, stride = [1,1,1,1], the output 10x10x16.

Activation: RELU

Pooling: The output 5x5x16.

Flatten using tf.contrib.layers.flatten

Layer 3: Fully Connected. Input 400, the output 120.

Activation: RELU.

Layer 4: Fully Connected. Input 120, output 84.

Activation: RELU

Layer 5: Fully Connected (Logits). This should have 43 outputs.

The cell 126 implemented this architecture.

2.3 Model Training and Results

The training data set includes 34,799 samples,

The validation data set includes 4,410 samples,

The testing data set includes 12,630 samples.

The optimizer we used is “tf.train.AdamOptimizer”

2.3.1 Hyper Parameter Tuning

Through experiments, the optimal hyper parameters are summarized as follows:

The learning rate is set as 0.001.

Batch size is set as 16.

The epoch number is set as 100.

The “keep_prob” is set as 0.5.

2.3.2 Results

Previously, the “keep_prob” dropout layer was not added, the result is

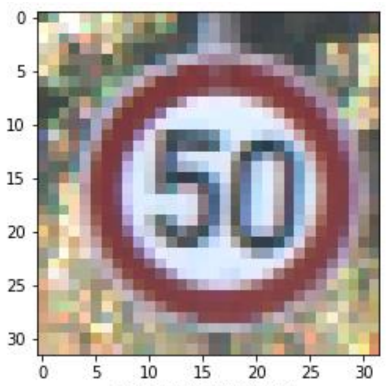
The best validation accuracy is 0.945, the test accuracy using the best model is 0.901.

As suggested by the Udacity project reviewer, a dropout layer with “keep_prob” equal to 0.5 was added during training. The new result is

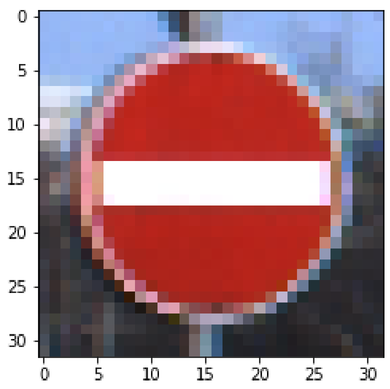
in cell 28, the best validation accuracy is 0.942; in cell 29, the test accuracy is 0.917.

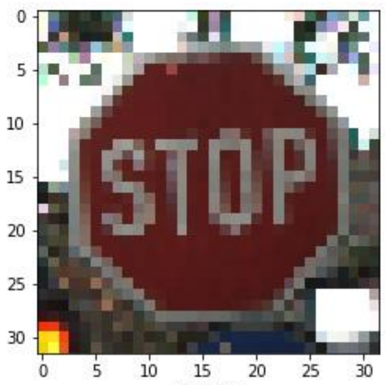
3. Test the Model on New Images

3.1 Visualization of Seven New Images

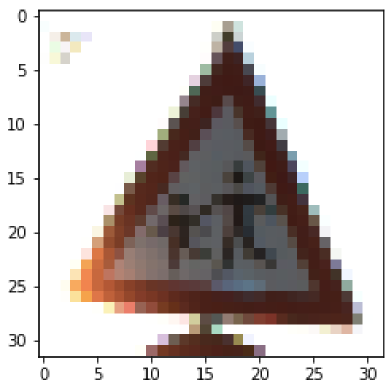


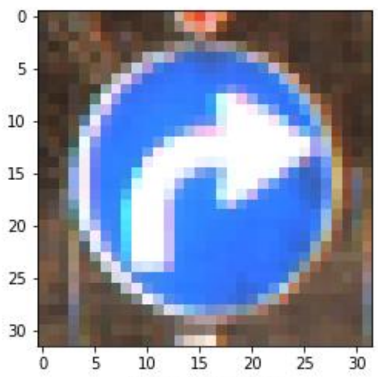
2-Speed limit (50km/h)

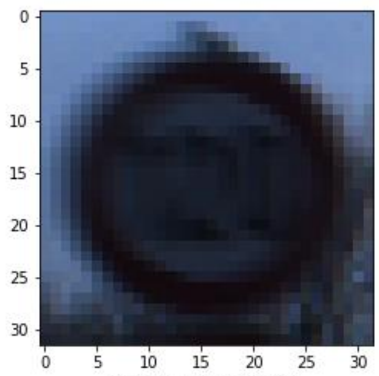

17-No entry



14-Stop


28-Children crossing


33-Turn right ahead


8-Speed limit (120km/h)

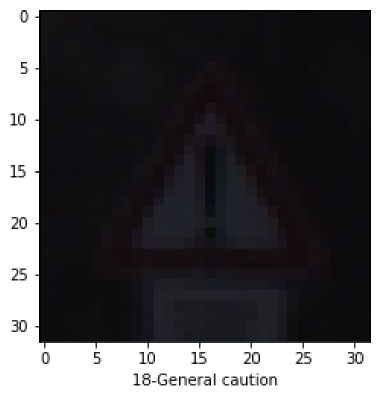


Figure 4. Seven New Images on Website.

These seven images are randomly selected. The original image is not 32x32, "cv2.resize()" function is applied to change the image size.

As suggested by the Udacity project reviewer, two more images are added (Traffic sign 8 and 18). They are included because the images look dark and blurry, which make the new testing dataset more challenging.

The images are converted to grayscale images using the same function.

3.2 Performance on the New Images

In cell 32, the best model was applied on the five new images. The test accuracy is 85.7%.

3.3 Model Certainty – Softmax Probabilities

Cell 34 and 39 calculated the softmax probabilities and printed out the top five probabilities for each image.

Probabilities (each row represents the top five softmax probabilities for each image)

```
[ 1.00000000e+00,   0.00000000e+00,   0.00000000e+00,0.00000000e+00,   0.00000000e+00],
[ 1.00000000e+00,   5.33214433e-08,   1.26626812e-12, 1.29910053e-13,   1.28394762e-14],
[ 1.00000000e+00,   0.00000000e+00,   0.00000000e+00, 0.00000000e+00,   0.00000000e+00],
[ 1.00000000e+00,   0.00000000e+00,   0.00000000e+00, 0.00000000e+00,   0.00000000e+00],
 [ 1.00000000e+00,   3.91283390e-22,   2.32067449e-30, 6.07944456e-36,   3.25670014e-36],
 [ 9.96532798e-01,   3.46712838e-03,   2.40412579e-10, 1.75710470e-11,   8.78924104e-12],
 [ 1.00000000e+00,   0.00000000e+00,   0.00000000e+00, 0.00000000e+00,   0.00000000e+00]
```

Corresponding Traffic Sign id:

```
[ 2,  0,  1,  3,  4],
[ 8,  7,  5, 12,  4],
[14,  0,  1,  2,  3],
[17,  0,  1,  2,  3],
[18, 26, 11, 27, 25],
[30, 23, 29, 11, 21],
[33,  0,  1,  2,  3]
```

Combining with Figure 4, we can see that the deep neural network labels the correct traffic IDs for the six images, except for traffic sign “28 Children crossing”. It is falsely labeled as “30 be aware of ice/snow”. This may be because we don’t have enough samples for traffic sign “28 Children crossing” as shown in Figure 1.

3.4 Visualization of Softmax Probabilities

As suggested by Udacity Project Reviewer, cell 45 shows the figures of the softmax probability for each test image, for example:

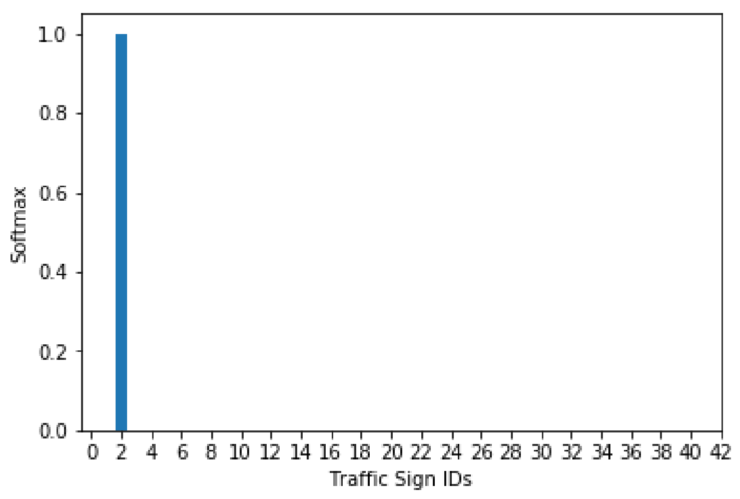
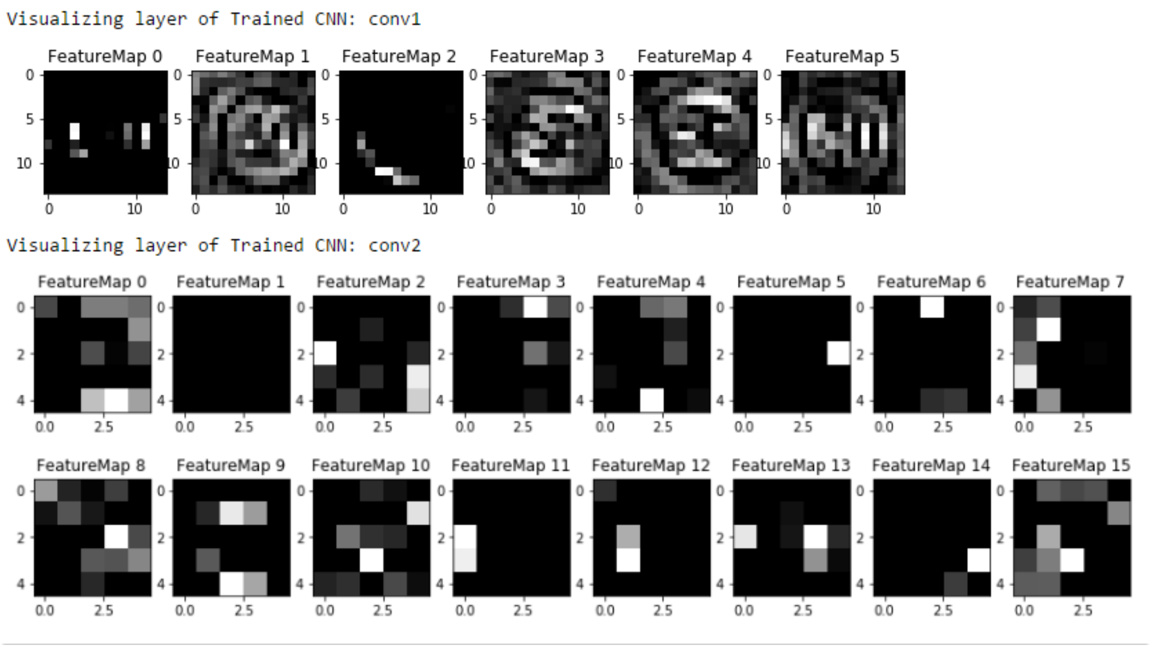


Figure 5. Visualization of Top 5 Softmax Probability for Traffic Sign “2- Speed Limit (50 km/h”

4. Visualization of Hidden Layers

In order to verify that the CNN really can capture some important features in images, it is interesting to visualize the hidden layers of the CNN.

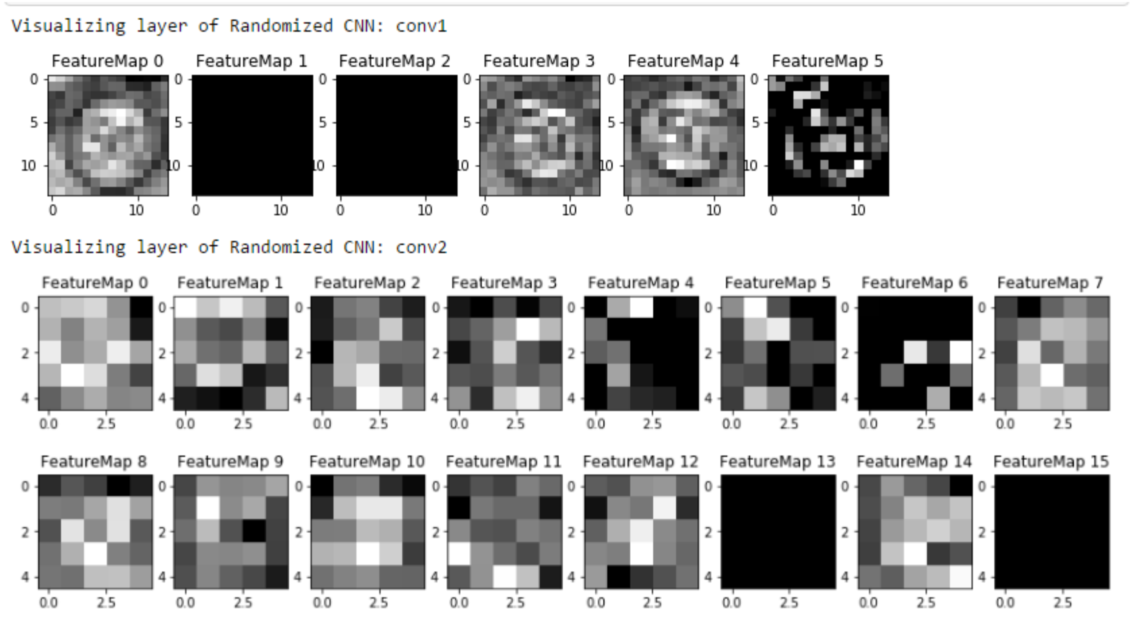
Recall that in the model architecture of this project, the output of the first convolutional layer is 14x14x6 and the output of the second convolution layer is 5x5x16. For the best model we found, the visualization of first and second convolution layers for the “Speed Limit (50 km/h)” in Figure 4 are as follows:



```
Visualizing layer of Trained CNN: conv1
Visualizing layer of Trained CNN: conv2
```

Figure 6. Visualization of Hidden Layers in the Learned Best Model

Similarly, we can visualize the hidden layers of a CNN with random weights for the same input image:



```
Visualizing layer of Randomized CNN: conv1
Visualizing layer of Randomized CNN: conv2
```

Figure 7. Visualization of Hidden Layers in a CNN with Random Weights

It is obvious that a well-trained CNN model really captures some features comparing with a randomized CNN model.