Traffic Sign Classifier

Jianxing Ke

December 27, 2018

The goals / steps of this project are the following:

- Load the data set (see below for links to the project data set)
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

1 Data Set Summary & Exploration

1.1 Summary of the data set

The data set used in this project was *German Traffic Sign Dataset*. After the data was loaded, we see that the size of training examples is 34799; the size of testing examples is 12630; the size of validation examples is 4410. The dataset contains 43 different traffic signs, and each image is 32x32x3.

To visualize the dataset, I randomly output 10 images, and counted the number of examples of each class and displayed it as a bar graph.

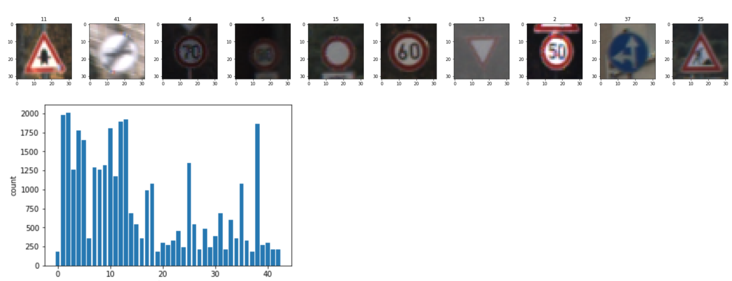

Figure 1: Image examples and bar graph

2 Model Architecture

2.1 Data Preprocessing

With the given dataset, I generated augmented data in the following ways: **Scale images**. For each image, I scaled it with factor 0.75 and 0.6 with respect to the center of the image, then resized it to 32x32x3.

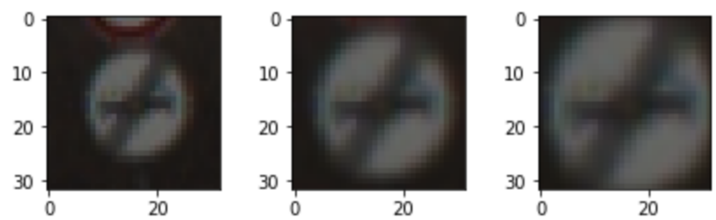

Figure 2: Scaled images 1

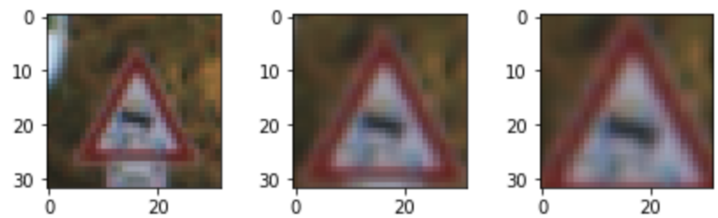

Figure 3: Scaled images 2

Rotate images. For each image, I rotated it 15 degrees clockwise and counterclockwise, without changing the size of the image. Here are two example outputs.

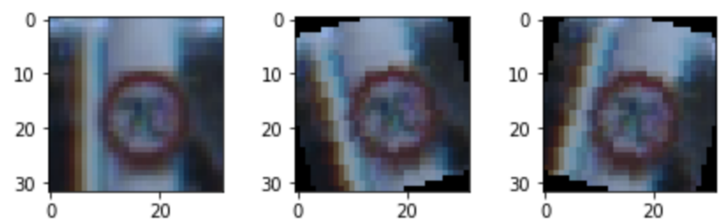

Figure 4: Rotated images 1

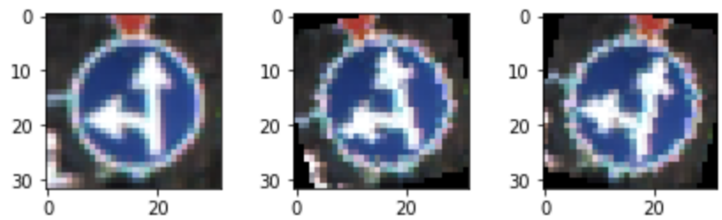

Figure 5: Rotated images 2

Add a value to images. For each image, I added a value of 3 and -3 to each pixels. The output images look similar to the original.

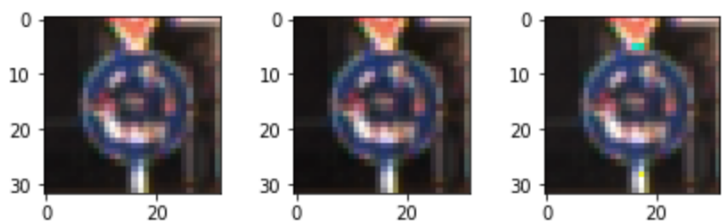

Figure 6: Images with added value

Normalize images. As suggested in the project instructions, for each image, I normalized it by subtracting 128 and divided by 128. By doing this the dataset would have 0 mean and equal variance.

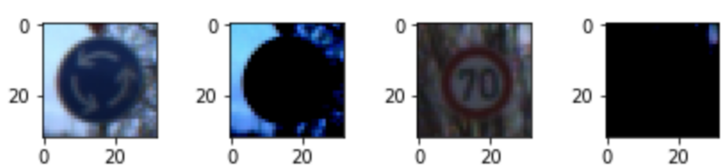

Figure 7: Normalized images

Turn images to grayscale. For our task of classifying the traffic signs, color is not really a useful information since we focus more on different shapes on a sign.

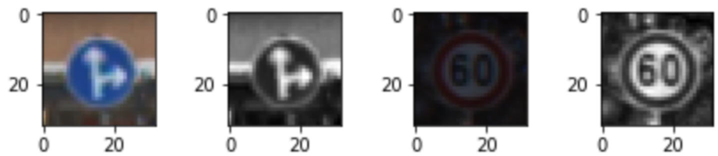

Figure 8: Grayscaled images

After all the preprocessing steps, I had a dataset of 243593 images, each has dimension 32x32x1.

2.2 Model Architecture

My current model is very similar to the LeNet model. I think my model underfitted for this project, even though it achieved pretty decent accuracy with the augmented dataset.

The model had the following layers:

Layer	**Description**
Input	32x32x1 gray image
Convolution	16 5x5 filters, stride 1x1, valid padding, outputs 28x28x16
RELU	
Max pooling	2x2 filter, stride 2x2, valid padding, outputs 14x14x16
Convolution	16 5x5 filters, stride 1x1, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 filter, stride 2x2, valid padding, outputs 5x5x16
Fully Connected	input 5x5x16, outputs 120
RELU	
Fully Connected	input 120, outputs 84
RELU	
Output	input 84, outputs 43

2.3 Model Training

To train the model, I used Adam optimizer with learning rate 0.001, and the loss function was cross entropy. During each epoch, the dataset was randomly shuffled, then a batch was fed into the classifier at a time. With the number of epochs 10 and batch size 256, my model had an validation accuracy of 94.9%. Without the augmented data, the model only achieved about 90% validation accuracy. Compared to other well-known architectures that achieve 99% accuracy, more layers can be added to my model, and parameters can be better tuned to achieve better results.

3 Test on New Images

To test my model, I picked 5 extra traffic signs online.

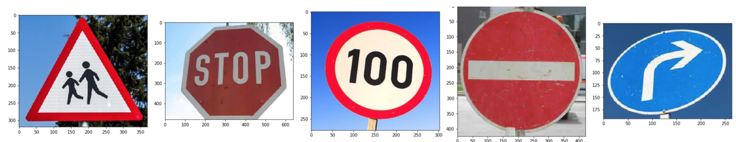

Figure 9: test images

These images were in different sizes, so I needed to preprocess them before feeding them to the model. Below are images that were actually fed to the model.

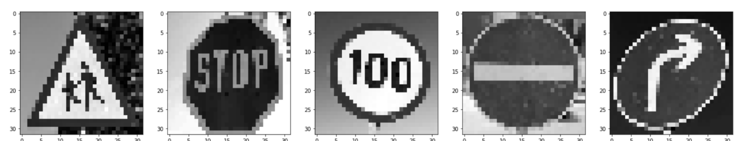

Figure 10: Preprocessed test images

The model achieved 80% accuracy on the 5 test images. To find out which image the model predicted wrong, I calculated the top 5 softmax probabilities for test images.

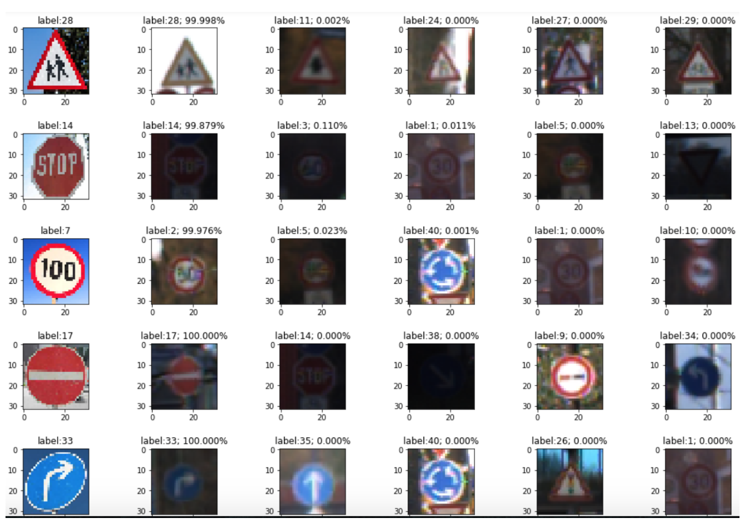

Figure 11: Top 5 softmax probabilities

From this, we see that the model predicted 4 traffic signs correctly, with high confidence. But it also predicted 100km/h speed limit sign incorrectly with high confidence.

4 Visualize the Network's State

With the given outputFeatureMap function, we can easily visualize a model's feature layers. I plot the outputs of my model's 2 convolution layers for one of the test images here.

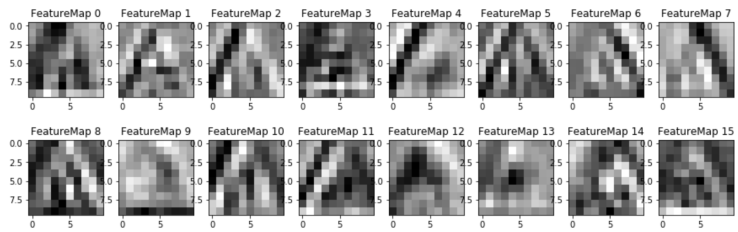

Figure 12: Conv layer 1

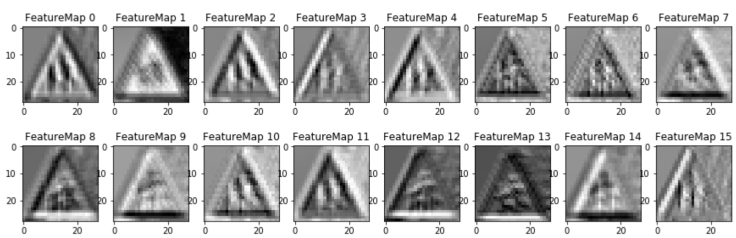

Figure 13: Conv layer 2

The feature maps captured the shape (triangle) of the sign very well, then it also captured the contents of the sign, in this case two human figures.