Traffic Sign Recognition

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

Data Set Summary & Exploration

1. After importing the dataset files I mostly used the methods len() and shape() to calculate number of examples and the shape of the images. I also imported the names of the labels for the traffic signs using the CSV library.

- Number of training examples = 34799
- Number of validation examples = 4410
- Number of testing examples = 12630
- Image data shape = (32, 32, 3)
- Number of classes = 43

2. Include an exploratory visualization of the dataset.

Below are 10 traffic signs with the title as the labels.

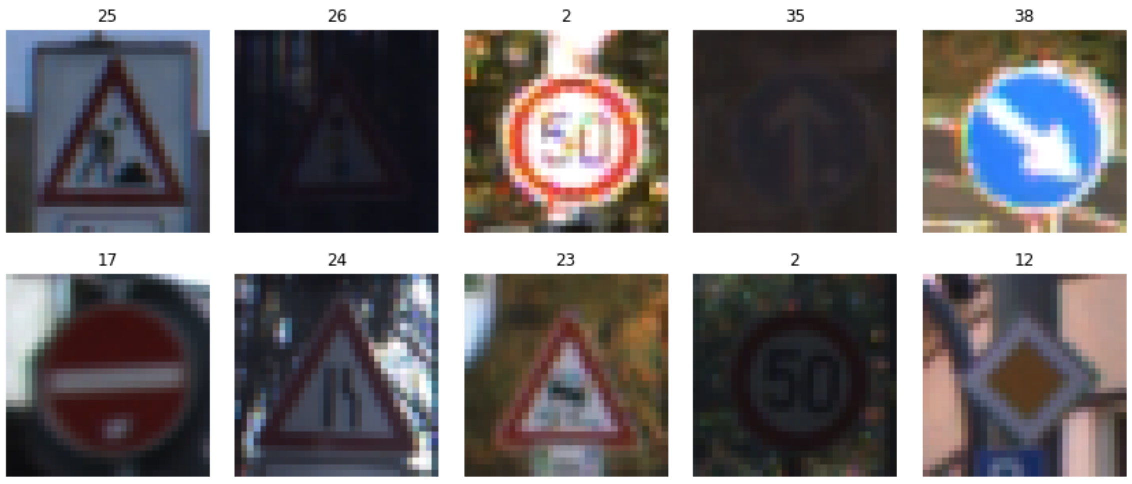


Figure 1: 10 Random Traffic Signs Images

Below is the traffic signs distribution in the training sets.

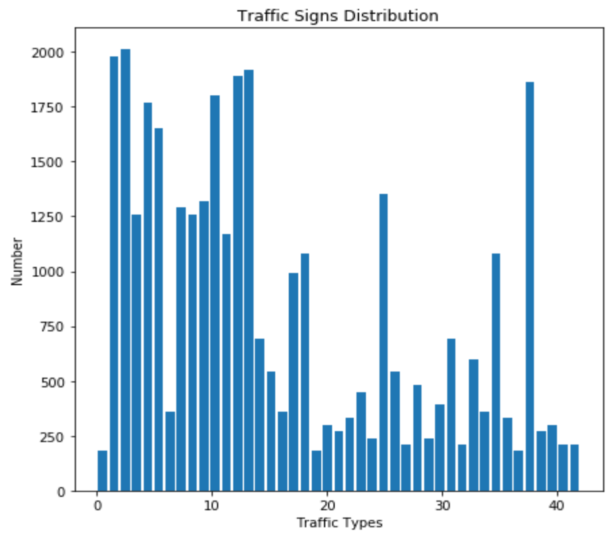


Figure 2: Traffic Signs Images Distribution

Design and Test a Model Architecture

1. **Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)**

   Each image is converted from RGB image to gray image and normalized to the range of [-1, 1] with 0 mean. The normalization can put all pixel values in similar range, which can facilitate the gradient descent step. The label of each image is expressed in terms of one-hot encoding, which could facilitate the calculation of loss function. For each training epoch, the training data is shuffled to prevent bias in dataset.

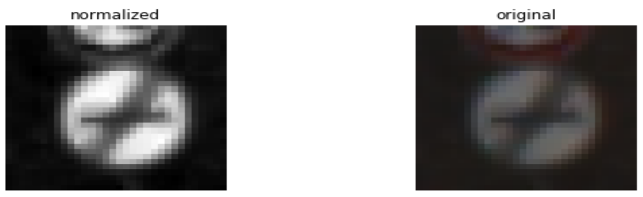


Figure 3: Comparison between Original RGB Image and Normalized Gray Image

2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.

Below is the network, which is the Lenet architecture design. The initial weights of each layer are determined by truncated normal distribution with standard deviation as 0.1 and mean value as 0.

Layer	Description
Input	32x32x1 gray images
Convolution	Weights Shape: (5, 5, 1, 6); Stride: (1, 1, 1, 1); Padding: Valid; Output Shape: (28, 28, 6)
Relu	
Max Pooling	Pooling Shape: (1, 2, 2, 1); Stride: (1, 2, 2, 1); Padding: Valid; Output Shape: (14, 14, 6)
Convolution	Weights Shape: (5, 5, 6, 16); Stride: (1, 2, 2, 1); Padding: Valid; Output Shape: (5, 5, 16)
Relu	
Max Pooling	Pooling Shape: (1, 2, 2, 1); Stride: (1, 2, 2, 1); Padding: Valid; Output Shape: (5, 5, 16)
Flatten	Output Shape: 400
Fully Connected	Weights Shape: (400, 120); Output Shape: 120
Relu	
Dropout	Probability to keep: 0.6

Fully Connected	Weights Shape: (120, 84); Output Shape: 84
Relu	
Fully Connected	Weights Shape: (84, 43); Output Shape: 43

3. 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.

The learning rate is set to 0.001, the batch size is 128 and the total number of epochs is 60. The loss function is based on Softmax Cross Entropy and the optimizer is Adam Optimizer. The final accuracy is 0.950.

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

My final model results were:

Validation Set Accuracy: 0.950

Test Set Accuracy: 0.939

Initially, the neural network tends to over fit itself since the validation accuracy increases to a peak and decrease. In this case, I add a dropout layer right after the first fully connected layer. Below is the figure that depicts the training progress. Even the model is equipped with dropout layer, the model still tends to over fit itself a little bit when the validation accuracy decreases and the training accuracy stays constant.

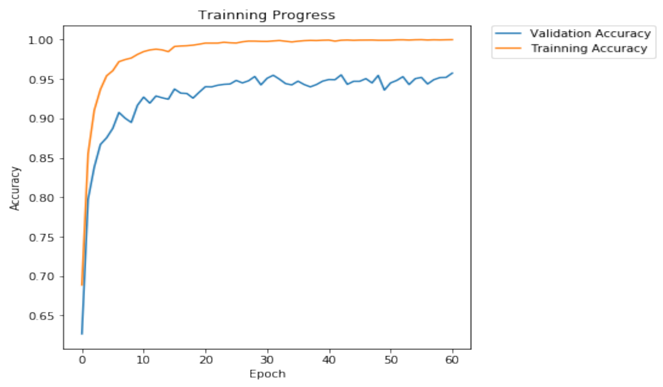
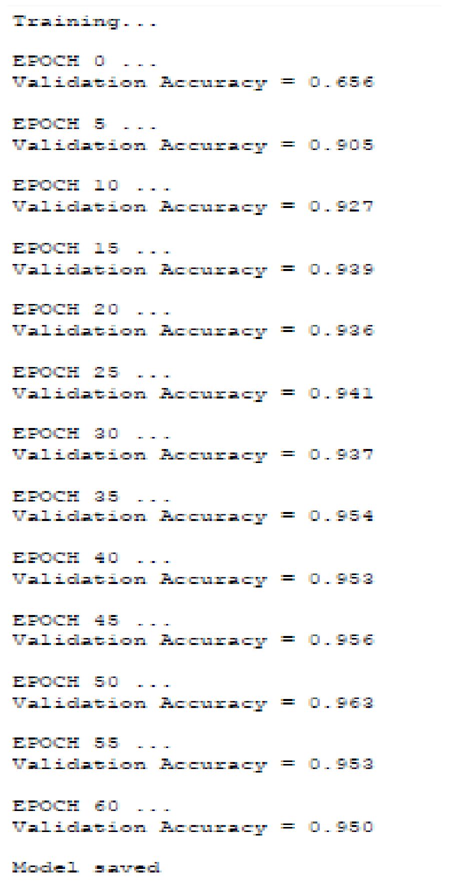


Figure 4: Model Training Progress

Also, the initial training epoch is set to 20, which is far less enough to fully train the model, so I increase the training epoch up to 60. Beyond the number of epoch, the batch size was set to around 400 initially. The larger the batch size for stochastic gradient descent, the more accurate the gradient moves towards to the target. However, it will consume more memory. So I choose the batch size of 128 at the end. Below is the training accuracy of the last model. We can notice that the model accuracy stops increasing at around 0.96. The potential reasons for it could be either model over fitting or not enough training data to feed in.



```
Training...

EPOCH 0 ...
Validation Accuracy = 0.656

EPOCH 5 ...
Validation Accuracy = 0.905

EPOCH 10 ...
Validation Accuracy = 0.927

EPOCH 15 ...
Validation Accuracy = 0.939

EPOCH 20 ...
Validation Accuracy = 0.936

EPOCH 25 ...
Validation Accuracy = 0.941

EPOCH 30 ...
Validation Accuracy = 0.937

EPOCH 35 ...
Validation Accuracy = 0.954

EPOCH 40 ...
Validation Accuracy = 0.953

EPOCH 45 ...
Validation Accuracy = 0.956

EPOCH 50 ...
Validation Accuracy = 0.963

EPOCH 55 ...
Validation Accuracy = 0.953

EPOCH 60 ...
Validation Accuracy = 0.950

Model saved
```

Figure 5: Model Training Process

Test a Model on New Images

1. Choose six 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.

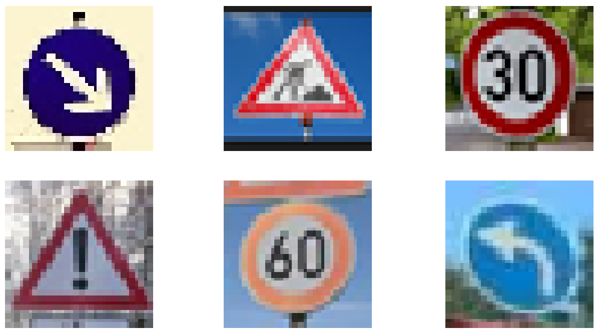


Figure 6: Six New German Traffic Signs

The images I choose are all front view without any rotation angle and background objects, so it would be easier for the algorithm to classify. However, compared these six images with the dataset, they also seems to be brighter and sharper, which could potentially confuse the model in the classification.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

The classification of the six new German traffic sings has 0.833accuracy. The classification accuracy for test data is around 0.933, which indicates the six new traffic sign images cannot fully evaluate the trained model.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

From the below image, there are six different German traffic sign images as input in the left most columns. The deep neural network classifies their types starting from the second column and the title of each image starting from the second column indicates the classification probability and the labels.

As you can see from the below prediction, the model is 100% certain of the classified images except for the second and fourth images. Even the model is not quite certain in the classification of the second image, it still gives 3% probability to an incorrect traffic sign. The model actually gives an incorrect label to the fourth image since the 60 Km/h sign and 50 Km/h are very similar to each other.

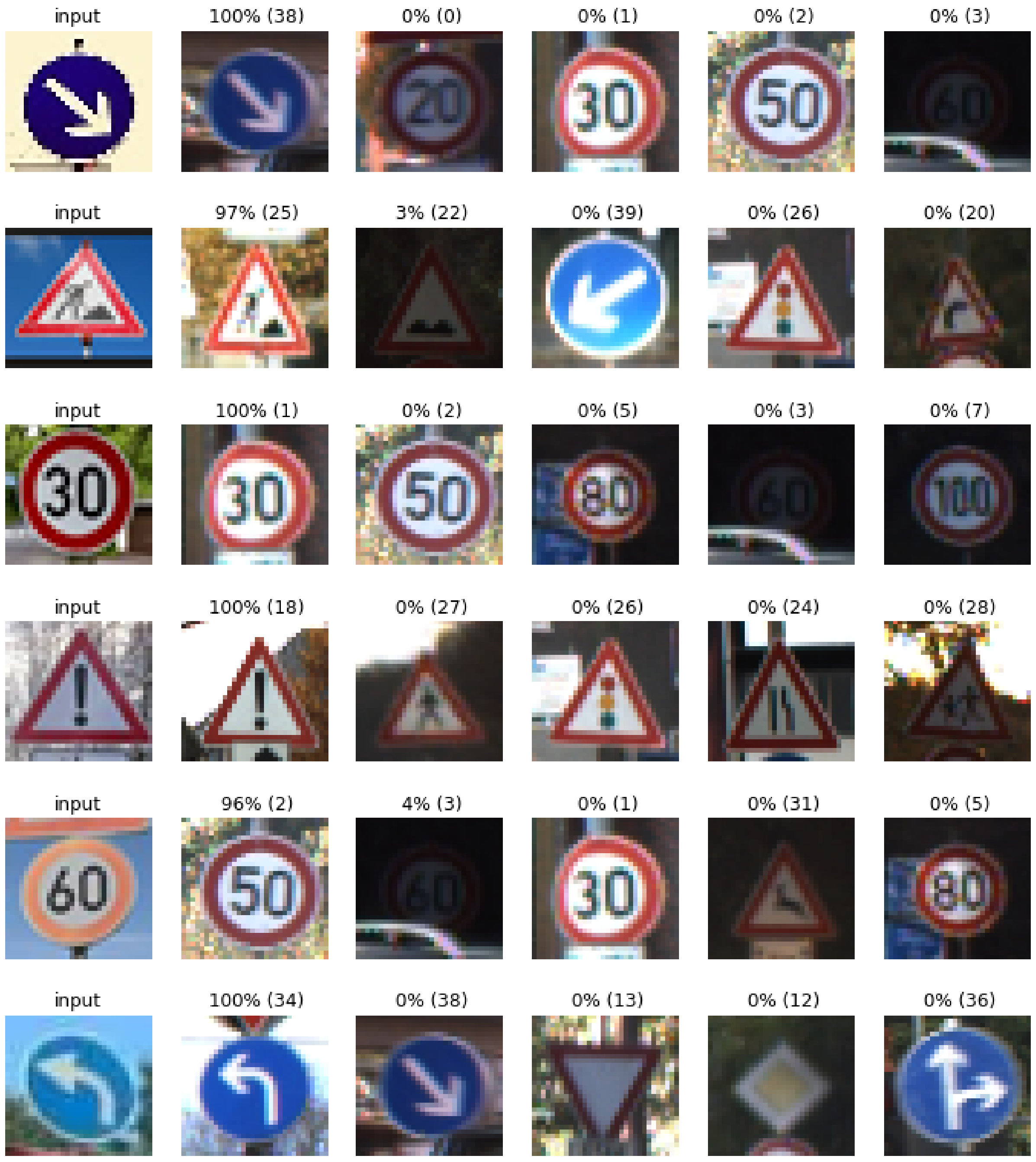


Figure 7: Classification of the Six German Traffic Signs

(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

1. Discuss the visual output of your trained network's feature maps. What characteristics did the neural network use to make classifications?

Below is the feature map of the first convolution layer. We can notice that the six features map can distinguish the lines and circle of a traffic sign, which can be used as one of the characteristics to classify the traffic sign.

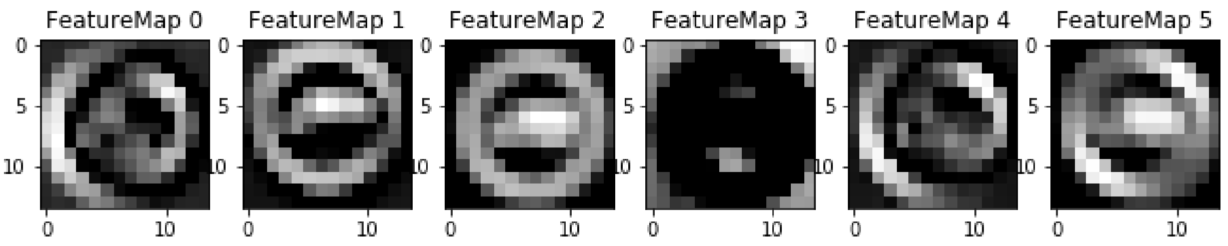


Figure 8: Feature Map of First Convolutional Layer