Traffic Sign Recognition

Data Set Summary & Exploration

1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

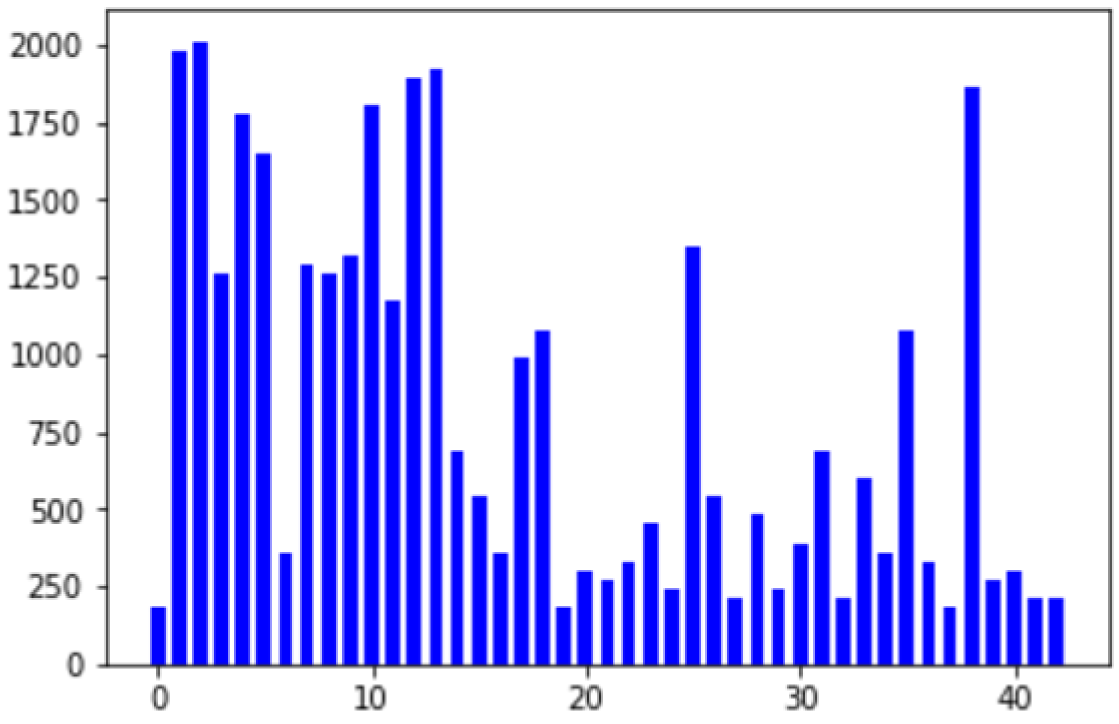
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

Here's a bar chart showing the distribution of different class of traffic signs from the training data. The x-axis shows the traffic sign type index, and the y-axis shows the count of training images for that class.

We can see that it's not distributed evenly. Some traffic signs appear much more often than others in this training data set.

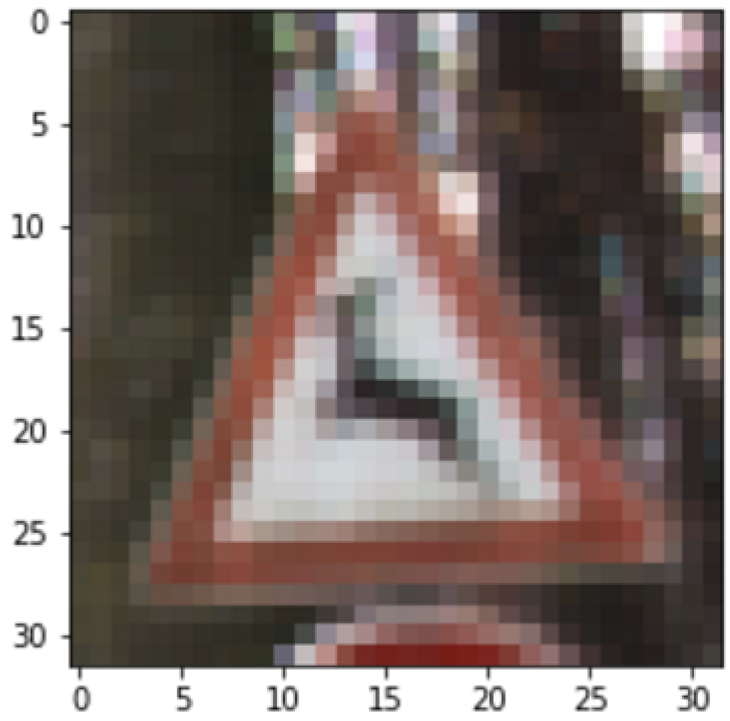


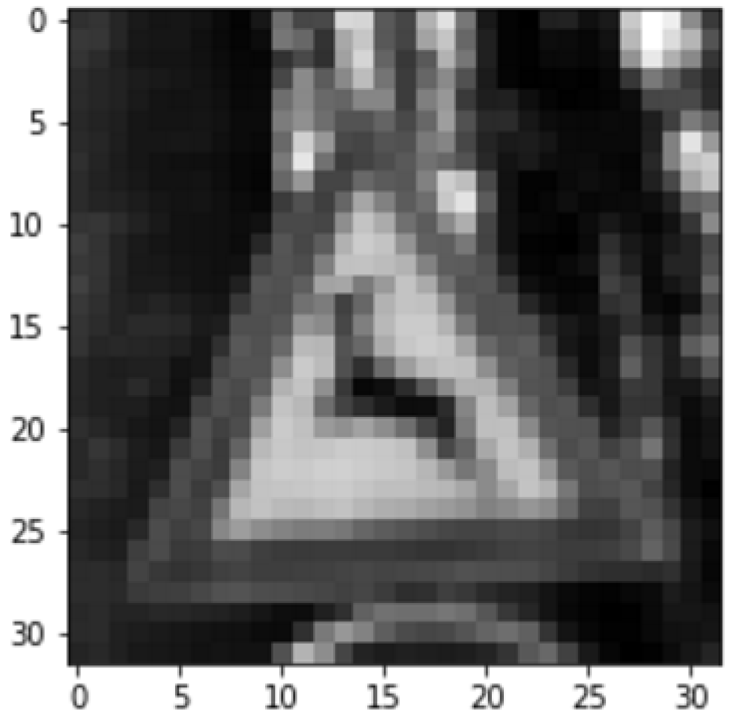
Design and Test a Model Architecture

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.)

I did two steps for preprocessing the images.

First I turn image into grayscale. Color is not that useful (compared with shape or contrast) when identifying traffic sign. Converting from RGB color to grayscale greatly reduces the data size and makes our training more efficient.





Then I normalize the picture. We learned that putting all input data within the range of (-1,1) with a mean of 0 will makes our training more efficient.

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.

My final model consisted of the following layers:

- Layer1: input 32x32x1, 5x5 convolution with stride (1,1), 10 output channels, then relu, then 2x2 maxpool, final output 14x14x10
- Layer2: input 14x14x10, 5x5 convolution with stride (1,1), 50 output channels, then relu, then 2x2 maxpool, then flatten, final output 1250
- Layer3: input 1250, fully connected to 500, then relu, output 500
- Layer4: input 500, fully connected to 200, then relu, output 200
- Layer5: input 200, fully connected to 43, output 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.

Here's the component that I prepared for training the model:
- Cost function: use softmax cross entropy between onehot encodeded prediction and label
- Optimizer: Use AdamOptimizer
- Batch size: 128
- Number of epochs: 30
- Learning rate: 0.0005

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:

- training set accuracy of 1.000

- validation set accuracy of 0.926

- test set accuracy of 0.928

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?

In the begining, my choice of architecture was very similar to the final one, except that the output channels and fully connected nodes are much much smaller, like 5~10x smaller.

- What were some problems with the initial architecture?

I found that my training and validation accuracy couldn't raise more than 85%. I tried with different number of batch size, number of epochs, and learning rate, none of that really helped.

- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

I thought my model might not be complicated enough, thus it might be under fitting. So I increase the size of the convolution output, so is the fully connected layer size. But it was not very easy to find a good number. When I set the network size to be too big, the validation accuracy actually decreases. After several trial, I tried to lower the learning rate from 0.001 to 0.0005, and it does increase the validation accuracy and the accuracy over later epochs are much more stable.

- Which parameters were tuned? How were they adjusted and why?

Batch size and number of epochs. The latter was easier to tune because a higher number of epochs combined with moderate learning rate will certainlly always makes the result better. Batch size is a bit weird and I don't fully understand how it affected the result. I tried to make it bigger or smaller and finally set it to 128.

- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model?

Convolution is certainly helpful because the traffic sign is a typical example of pixel patterns being features. Convolution combines low-level features (pixels) into higher-level features (patterns) and these higher-level features are bascially able to match the traffic sign's design.

I didn't use dropout layer. I tried to add it but unfortunately I didn't find it generates much difference for my case.

If a well known architecture was chosen:

- What architecture was chosen?

I chose the well-known LeNet architecture.

- Why did you believe it would be relevant to the traffic sign application?

LeNet was designed to recognize written numbers. I think traffic sign has a lot of similarities when compared with written numbers because both features= line patterns (curve, straight line, dots) and none of then features colors or lighting conditions.

- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

The training accuracy is 100%, and the validation and test accuracy are around 93%. I think the result looks good. It might be a little overfitted, but overall still generalizes well.

Test a Model on New Images

1. Choose five 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.

Here are five German traffic signs that I found on the web:

The first image might be difficult to classify because …

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).

Here are the results of the prediction:

Image	Prediction
Speed limit (120km/h)	Speed limt (120km/h)
Yield	Yield
Road work	Road work
Bicycles crossing	Bicycles crossing
End of no passing	End of no passing

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%. This is even better than the test accuracy in previous training process.

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)

- First image: Speed limit (120km/h). The prediction results with probabilities:
  class: 8, probability: 1.0, signname: Speed limit (120km/h)
  class: 7, probability: 3.873366904993958e-11, signname: Speed limit (100km/h)
  class: 40, probability: 6.261631431457929e-14, signname: Roundabout mandatory
  class: 0, probability: 5.136434261929679e-17, signname: Speed limit (20km/h)
  class: 14, probability: 2.312363239864558e-19, signname: Stop

- Second image: Yield. The prediction results with probabilities:
  class: 13, probability: 1.0, signname: Yield
  class: 35, probability: 1.638729776876069e-11, signname: Ahead only
  class: 1, probability: 1.946183266025373e-16, signname: Speed limit (30km/h)
  class: 15, probability: 7.69797505273448e-17, signname: No vehicles
  class: 9, probability: 6.696285800712366e-17, signname: No passing

- Third image: Road work. The prediction results with probabilities:
  class: 25, probability: 1.0, signname: Road work
  class: 19, probability: 1.1825431211782023e-15, signname: Dangerous curve to the left
  class: 22, probability: 1.6098153653632577e-16, signname: Bumpy road
  class: 29, probability: 1.6576936049217865e-17, signname: Bicycles crossing
  class: 23, probability: 3.4634509781871587e-19, signname: Slippery road

- Forth image: Bicycles crossing. The prediction results with probabilities:
  class: 29, probability: 1.0, signname: Bicycles crossing
  class: 28, probability: 3.4673536264406835e-10, signname: Children crossing
  class: 24, probability: 4.831855943999985e-13, signname: Road narrows on the right
  class: 22, probability: 1.8851250855461688e-13, signname: Bumpy road
  class: 35, probability: 1.779948428503198e-14, signname: Ahead only

- Fifth image: End of no passing. The prediction results with probabilities:
  class: 41, probability: 1.0, signname: End of no passing
  class: 42, probability: 5.07171145358587e-11, signname: End of no passing by vehicles over 3.5 metric …
  class: 32, probability: 8.970013967712909e-12, signname: End of all speed and passing limits
  class: 6, probability: 2.978466811295183e-13, signname: End of speed limit (80km/h)
  class: 12, probability: 1.3348034344360304e-15, signname: Priority road

(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?