Traffic Sign Recognition - Project Report

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

Content

Code 2

1. Project code: link 2

Data Set Summary & Exploration 2

2. Provide a basic summary of the data set. 2
3. Include an exploratory visualization of the dataset 2

Design and Test a Model Architecture 3

4. Describe how you preprocessed the image data. 3
5. Describe what your final model architecture looks like. 3
6. Describe how you trained your model. 4
7. Describe the approach taken for finding a solution. 4
8. Choose five German traffic signs found on the web and provide them. 5
9. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. 5
10. Describe how certain the model is when predicting on each of the five new images 6

Code

1. Project code: link.

Data Set Summary & Exploration

2. Provide a basic summary of the data set.

The code for this step is contained in the second code cell of the IPython notebook.

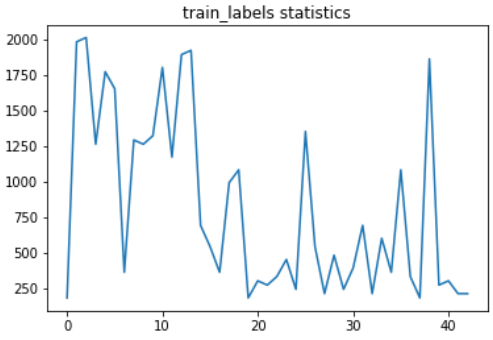
I used ordinary Python to calculate summary statistics of the traffic signs data set:

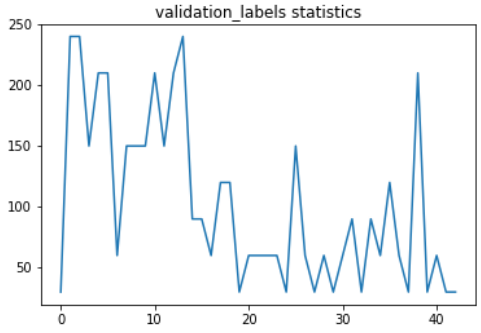
- The size of training set is 34799
- The size of test set is 12630
- The shape of a traffic sign image is [32, 32]
- The number of unique classes/labels in the data set is 43

3. Include an exploratory visualization of the dataset

The code for this step is contained in the third code cell of the IPython notebook.

Here is an exploratory visualization of the training and validation data set. The number of training and validation samples is not very balanced across the possible labels.





Design and Test a Model Architecture

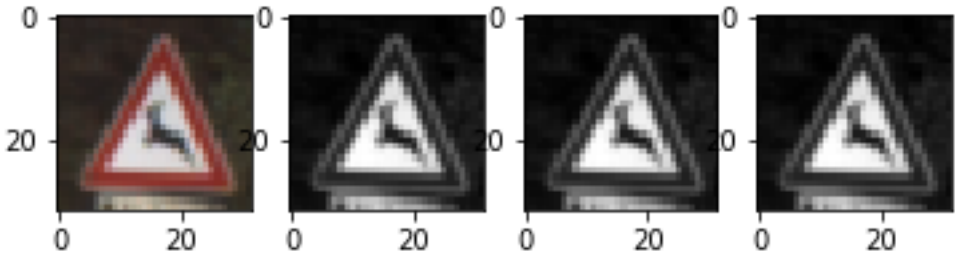
4. Describe how you preprocessed the image data.

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.

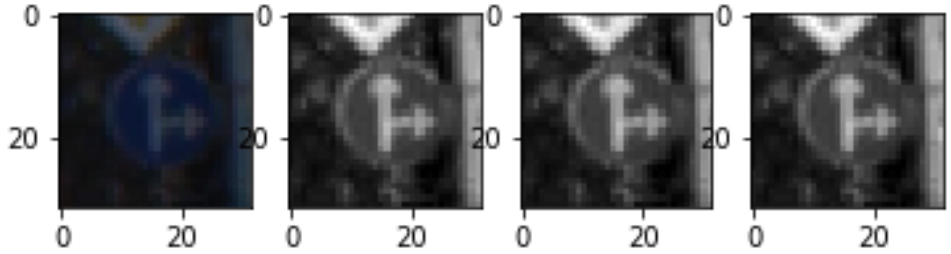
The code for this step is contained in the fourth code cell of the IPython notebook.

First, I converted the images to grayscale because grayscale (i.e. intensity) is usually sufficient to distinguish the edges and shapes needed for traffic sign detection. The leftmost image below is the original input. The second to left image is the image after grayscale conversion is done.

Then I shifted each image by its mean so that the mean of the image will become zero. This is done because some images might be brighter, while some are darker. Making the mean zero will reduce the bright/dark effect on the network parameters. The image after shifting is applied the shown at the second to right image below.



Finally, I normalized the image data with min-max scaling to scale the values to [0.1, 0.9] because this will produce smaller standard deviations, which can suppress the effect of outliers.



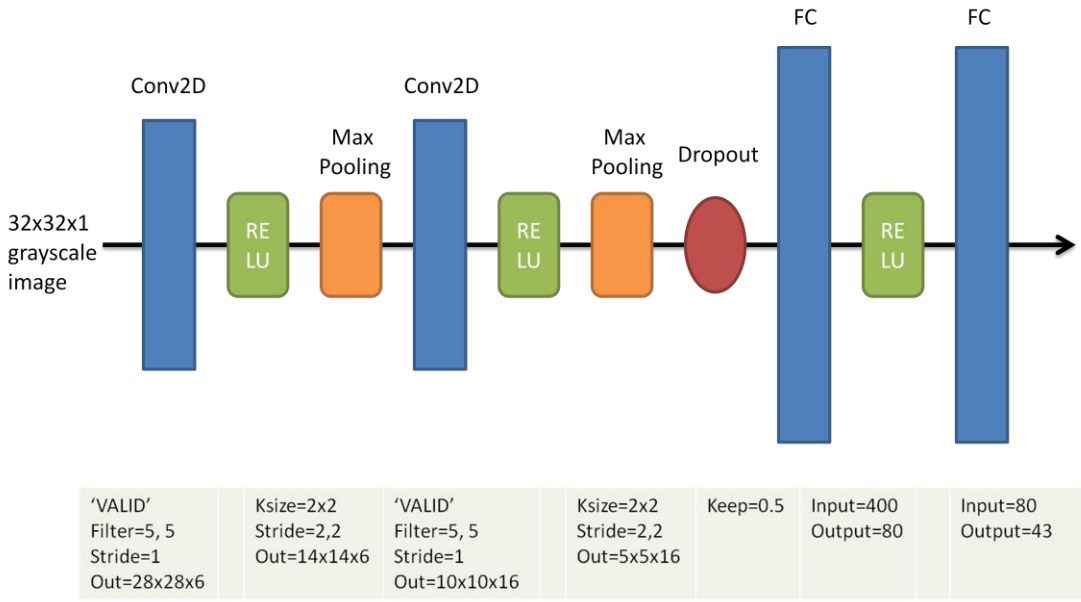
The normalizing effect can be seen more clearly with the second image sequence, where the original dark image is made more visible.

5. 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

The code for my final model is located in the 5th cell of the ipython notebook.

My final model consisted of the following layers:



6. Describe how you trained your model.

   The discussion can include the type of optimizer, the batch size, number of epochs and any hyper parameters such as learning rate.

   The code for training the model is located in the 5th and 6th cell of the ipython notebook.

   To train the model, I used an AdamOptimizer to minimize the loss.

   The final batch size is 128, with a learning rate of 0.0005, and epoch 91.

7. Describe the approach taken for finding a solution.

   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.

   The code for calculating the accuracy of the model is located in the 7th cell of the Ipython notebook.

   My final model results were:

training set accuracy	99.9%
validation set accuracy	96.2%
test set accuracy	94.2%

I tried first with the LeNet architecture, since it is was a tuned network architecture already known to work for some recognition work. I found that the training accuracy was very high, while the validation accuracy was not as high, indicating overfitting might have occurred.

Then I implemented two for loops to loop through possible values for learning rates and batch size. I also set the epoch to a larger value, and stored the maximum accuracy ever attained. This way it is easier to automatically go through these three parameters when something else, like the network architecture, have changed.

Since overfitting occurred, I tried to reduce the number of parameters by removing the last fully connected layer and by reducing the number of nodes in the hidden layers. Then I added a layer of dropout to make my network more robust. Also coloring seems not to be a critical factor for people to recognize the traffic signs, so I transformed the input images to grayscale to reduce unnecessary information been feed into the network. I also tried min-max scaling to scale the values to [0.1, 0.9] to suppress the effect of outliers.

Test a Model on New Images

8. Choose five German traffic signs found on the web and provide them.

   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 left most image might be difficult to classify because it is not shot from the front of the sign, so the circle rather looks like an oval in the image.

The second to left image seems to be shot during sunset. It might be difficult to classify since the supposedly white background looks yellow in the image.

The third to left image looks a bit difficult to classify because the sign is a bit dirty, there are multiple dark spots on the image obscuring parts of the sign.

The second to right image might be difficult to classify because there is a water reflection of the sign, which might trick the neural network into thinking there are two signs.

The right most image looks difficult to classify because the dirty water obscured one-third of the sign.

9. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set.

   The code for making predictions on my final model is located in the 10th cell of the Ipython notebook.

   Here are the results of the prediction:

Image index	Extra Image	Prediction	Predict correct?
1	50km/h	Roundabout mandatory	N
2	Right-of-way at next intersection	Right-of-way at the next intersection	Y

3	Bumpy road	Road work	N
4	Road work	Road work	Y
5	Stop Sign	Double curve	N

The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%.

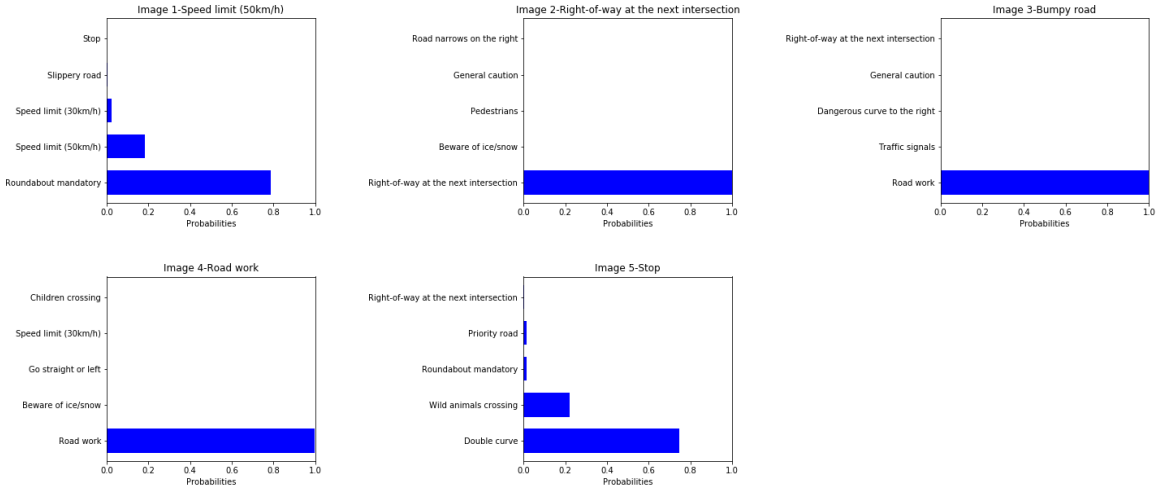
However, the model was able to attain an accuracy of 94.2% on the test set.

10. Describe how certain the model is when predicting on each of the five new images

By looking at the softmax probabilities for each prediction and identify where in your code softmax probabilities were outputted.

Provide the top 5 softmax probabilities for each image along with the sign type of each probability.

The code for outputting softmax probabilities on my final model is located in the 12th and 13th cell of the Ipython notebook.



For the first image, the model is relatively sure that this is a roundabout mandatory (probability of 0.78), but the image contains a speed limit sign. The top five soft max probabilities were

```
['Roundabout mandatory', 'Speed limit (50km/h)', 'Speed limit (30km/h)', 'Slippery road', 'Stop']

[ 0.78772819  0.18206777  0.02234828  0.00533143  0.00165933]
```

For the 2nd image, the model is very sure that this is a right-of-way at the next intersection (probability of 0.99), and the image indeed is such a sign. The top five soft max probabilities were

```
['Right-of-way at the next intersection', 'Beware of ice/snow', 'Pedestrians', 'General caution', 'Road narrows on the right']

[  9.99987125e-01   1.22189285e-05   5.61824663e-07   2.25352537e-09   1.22244737e-09]
```

For the 3rd image, the model is very sure that this is a road work sign (probability of 1.0), but the image is actually a bumpy road. The top five soft max probabilities were

```
['Road work', 'Traffic signals', 'Dangerous curve to the right', 'General caution', 'Right-of-way at the next intersection']

[  1.00000000e+00   3.04019885e-08   6.98623356e-14   4.72381432e-14   1.03068968e-14]
```

For the 4th image, the model is very sure that this is a road work sign (probability of 0.99), and the image indeed is such a sign. The top five soft max probabilities were

```
['Road work', 'Beware of ice/snow', 'Go straight or left', 'Speed limit (30km/h)', 'Children crossing']
```

```
[  9.97821569e-01   8.85612622e-04   4.99984308e-04   2.65171228e-04   2.62369722e-04]
```

For the last image, the model is relatively sure that this is a double curve (probability of 0.74), but the image is a cropped stop sign. The top five soft max probabilities were

```
['Double curve', 'Wild animals crossing', 'Roundabout mandatory', 'Priority road', 'Right-of-way at the next intersection']

[ 0.74563754  0.22216606  0.0154766   0.01369269  0.00174   ]
```