Traffic Sign Recognition

Build a Traffic Sign Recognition Project My pipeline for this project consists of 5 steps.These steps are :

- step 0: Load the data set
- step 1: Explore, summarize and visualize the data set
- step 2: Design, train and test a model architecture
- step 3: Use the model to make predictions on new images
- step 4: Analyze the softmax probabilities of the new images
- Summarize the results with a written report

Step 0: Load the Data

First, I loaded the data set file and placed the them in train, valid, and test.

Step 1: Dataset Summary & Exploration

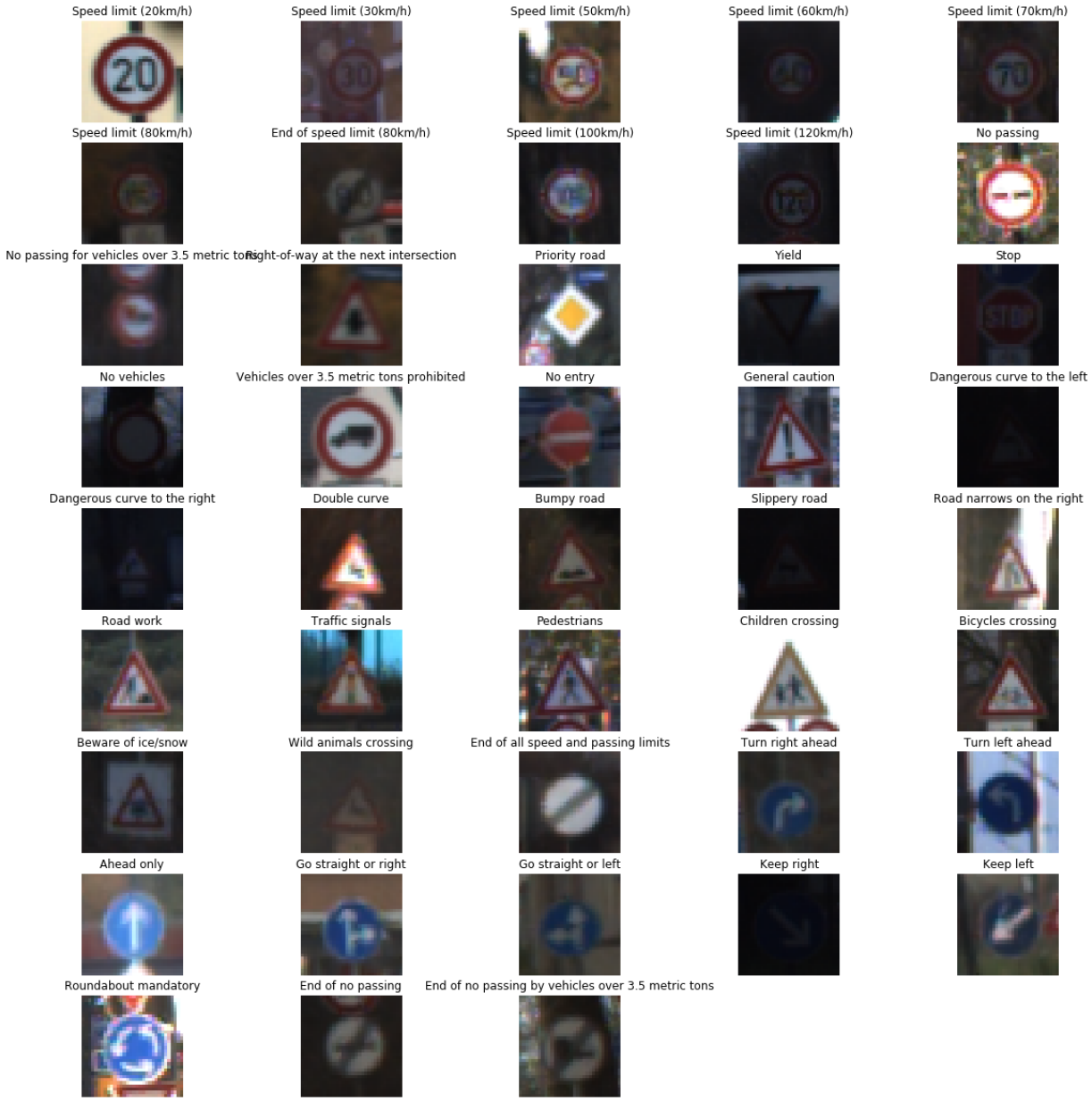
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

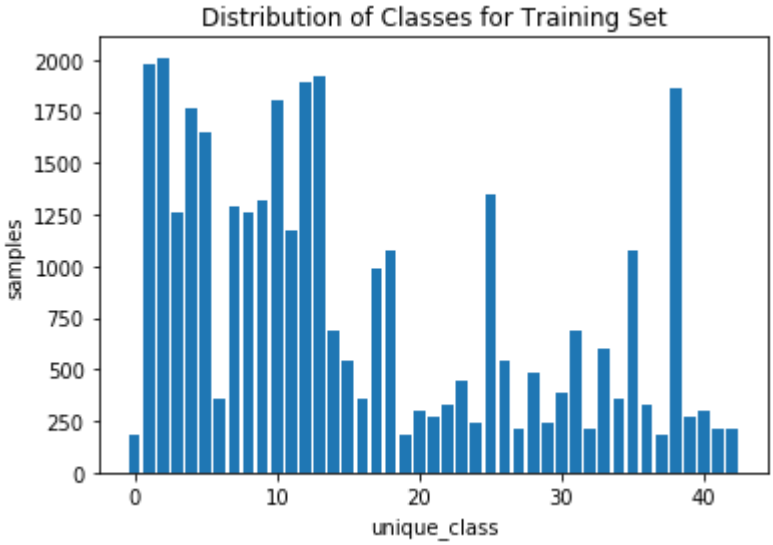
- The size of training set is ? 34799
- The size of the validation set is ? 4410
- The size of test set is ? 12630
- The shape of a traffic sign image is ? (32, 32, 3)
- The number of unique classes/labels in the data set is ? 43

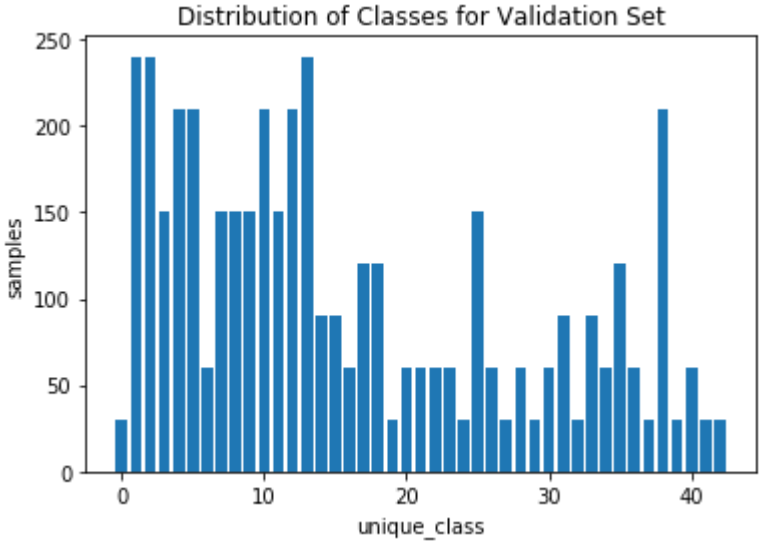
2. Include an exploratory visualization of the dataset.

I visualized one images in each class (43 classes in totall).



After that I showed the distribution of classes in the training, validation and test set. The distribution is not the same and some classes has more samples than the others. You can see the bar charts below.



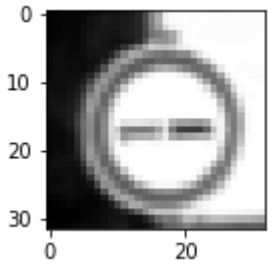


Step 2: Design and Test a Model Architecture

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.

First of all, we need to pre-process the data set. I converted the images to grayscale because it is really good for Sermanet and LeCun as it mentioned in traffic sign classification article. Also, it reduces the training time, especially when Gpu is not avaiable. Then I normalized the image data because I wanted to my data has a mean close to the zero and equal variance. I used this code "normal_image= (gray_image - 128) / 128" to normalize my data, and its easy to use it.



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:

Layer	Description
Input	32x32x1 gray_scale image
Convolution_1 5x5	1x1 stride, VALID padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution_2 5x5	1x1 stride, VALID padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Fully connected_0	Output = 400
Dropout	keep_prob = 0.5
Fully connected_1	Output = 120
RELU	
Dropout	keep_prob = 0.5
Fully connected_2	Output = 84
RELU	
Dropout	keep_prob = 0.5
Fully connected_3	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.

Now ,to train the model, I used the Lenet Model used in the course material with some changes to increase the accuracy of model. So the approaches was adding droupout in fully connected layers with keep probablities of 0.5. Also I increased the number of echos to have more accurate optimization. And the type of optimizer was ADAM optimizer.

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.

The code for this part is in the last section of " Step 2: Design and Test a Model Architecture".

To improve the accuarcy, I worked on hypermaraters such as learning rate, batch size, number of epchos, and keeo probabilities of droupout.

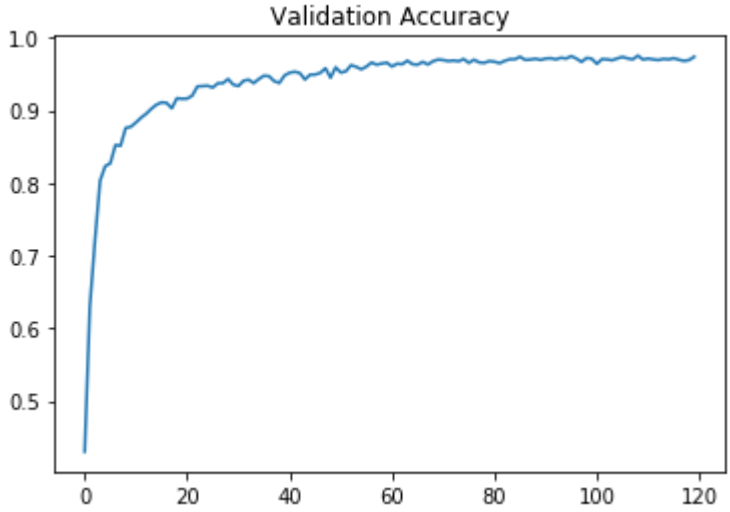
And this is the final parameters which I used in my code:

- Epoch = 120
- batch Size = 128
- Learning Rate = 0.0009
- keep_prob = 0.5

My final model results were:

- training set accuracy of = 0.995
- validation set accuracy of = 0.974
- test set accuracy of = 0.952

This is the plot for validation accuracy:



If an iterative approach was chosen:

What was the first architecture that was tried and why was it chosen?
I used the LeNet architecture very similar to the one that was given in the course material and this is due to the accuracy of the prediction was good.
What were some problems with the initial architecture?

One problem was lack of data for some images. Also the accuracy is not high enough beacuse image data was not pre processed.

How was the architecture adjusted and why was it adjusted?

To adjust the Lenet architecture, I added droupout in fully_connected layers to increase the accuracy.

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

Epoch, learning rate, batch size, and drop out probability were all parameters tuned.

- Epoch: First I used a big number and then I decreased the number when I assured that it reached to the maximume accuracy.
- Learning Rate: Higher leraning rate trains model faster but the accuarcy is low. So I used lower learning rate that trained the model slower but the acheived loss is low.
- Batch Size: We use it beacause using the whole data set needs high computation power. So batch gives us the oppurtunity to split the model in batches. I didnt changed it , I used batch size of 128.

What are some of the important design choices and why were they chosen?

One of the most important thing is, to have more uniform dataset along with enough convolutions.

Step 3: 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 nine German traffic signs that I found on the web:

I changed size of all images to 32*32. and then used normalization and converted to gray scales.

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.

Here are the results of the prediction:

Image	Prediction
Bumpy road	Bumpy road

Image	Prediction
Speed limit (60km/h)	Speed limit (60km/h)
Children crossing	Children crossing
No entry	No entry
No passing	No passing
Turn left ahead	Turn left ahead
Road work	Road work
General caution	General caution
Right-of-way at the next intersection	Right-of-way at the next intersection

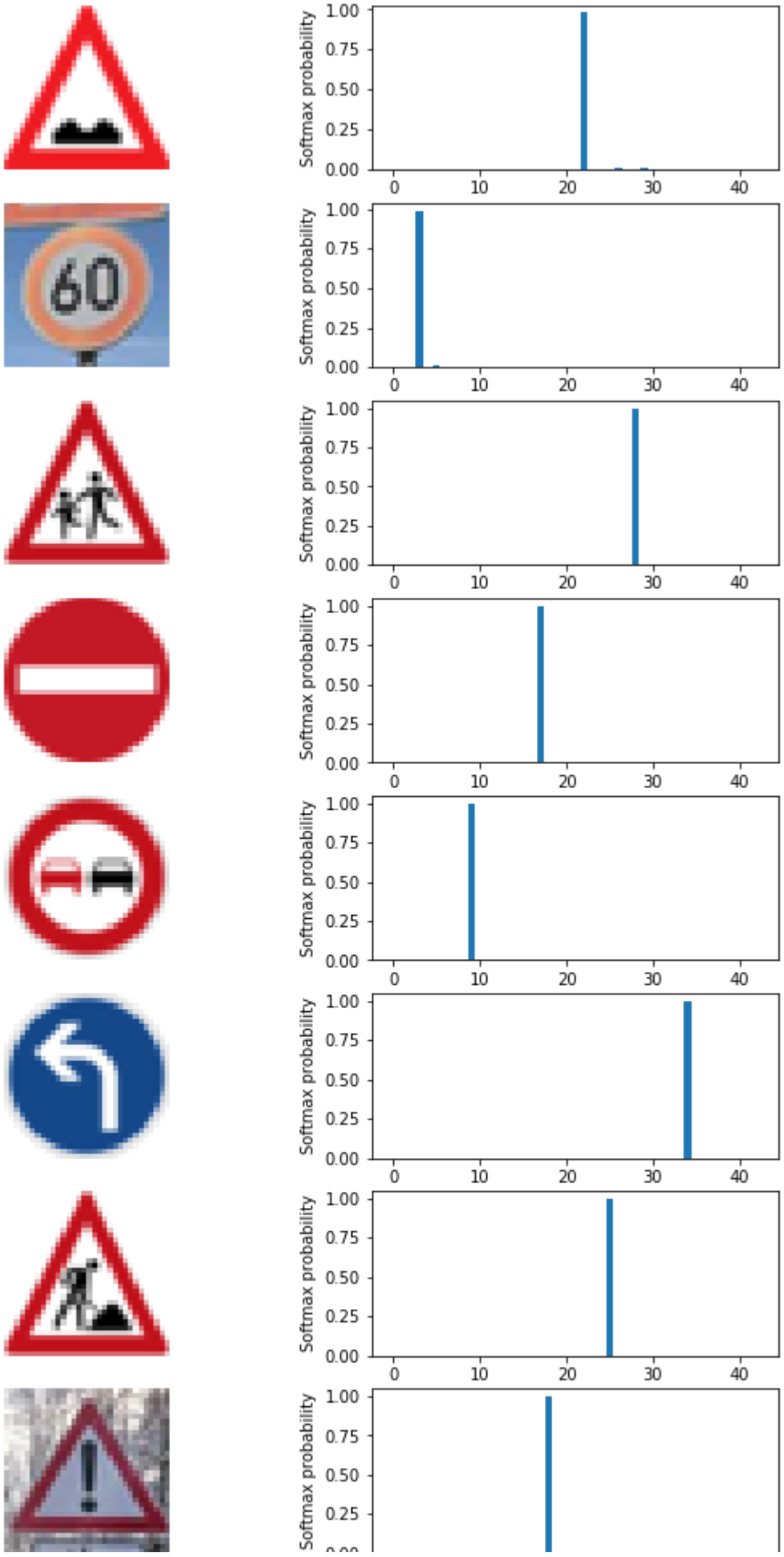
The model was able to correctly guess 9 of the 9 traffic signs, which gives an accuracy of nearly 100%.

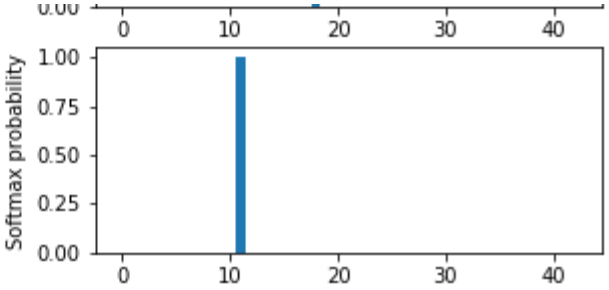
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability.

The code for making predictions on my final model is located in the final cell of "Step 3: Test a Model on New Images".

Probability	Prediction
%98	Bumpy road
%99	Speed limit (60km/h)
%100	Children crossing
%100	No entry
%100	No passing
%100	Turn left ahead
%100	Road work
%100	General caution
%100	Right-of-way at the next intersection

Figure showing sofmax probability.


Softmax probability
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40


0.00
0
10
20
30
40
1.00
0.75
0.50
0.25
0.00
Softmax probability
0
10
20
30
40

.