
###Data Set Summary & Exploration

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

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

I used the python numpy library 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 (32x32x3)
* The number of unique classes/labels in the data set is 43

####2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

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

Here is an exploratory visualization of the data set. I randomly select a image from the train dataset and see how it looks like. I also plot a a histogram to see the number of images per class.
![alt text][image1]

###Design and Test a Model Architecture

####1. Describe how, and identify where in your code, 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. Pr-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.
The input images are RGB images but the way camera perceives the color depends on external factor. Therefore converting the images to grayscale and then analyzing them is more logical approach.
I use opencv function to convert the RGB image to grayscale , the ouput size is (32x32). So to use it with LeNet Architecture i add a new channel to the images to make them (32x32x1).
I also Normalize the images in range 0.1-0.9 to have good numerical stability.

####2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The code for splitting the data into training and validation sets is contained in the fifth code cell of the IPython notebook.

To cross validate my model, But before splitting data into training and validation set I perform all the preprocessing steps on the training set images and then shuffle the images.
Afterwards i use sklearn's train_test_split function to randomly split the set into training and validation set.

My final training set had 31319 number of images. My validation set and test set had 3480 and 12630 number of images.

I never generated additional data.

####3. Describe, and identify where in your code, 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 seventh cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale image
Convolution1 5x5	1x1 stride, Valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution2 5x5	1x1 stride , Valid padding, output 10x10x16
RELU	
Max pooling.	2x2 stride , output 5x5x16
Fully connected1	input 400(after flatten input features as fector), output 120
RELU	
Dropout	
Fully Connected2	input 120 , output : 100
RELU	
Dropout	
Ouput Layer	input : 100, Output: 43

The Layer is mostly similar to LeNet lab Conv layer apart from minor changes which are
 1. I added dropout with a probability of 0.7 while training to take case of over fitting problem.
 2. Seconds i modified the final output layer input size to make the input more stable. In Machine learning its a good practice to have your matrix going into the classifier stable to have good numerical optimization.

####4. Describe how, and identify where in your code, 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 code for training the model is located in the ninth cell of the ipython notebook.

To train the model, I used an AdamOptimizer as it showed better results than normal Gradient descent optimizer. Before optimizing my loss function I also used L2 regularization with improved my validation accuracy by 3 percents from 95 to 98 approximately. I played around with different values of epochs and batch sizes. Increasing the batch size didn't have much effect on my training accuracy , but the training was much faster. But increasing the number of epoch improved my accuracy , i used 20 epoch finally which gave me the best result improving further didn't improve my accuracy any further. I never played with the learning rate as i know bigger the learning rate faster is training but classifier tends to over-fit the date.
Moreover 0.001 is a good staring point for learning Rate. I played a bit around with the values of beta the hyperparameter for L2 regularization and found 0.01 is a good staring point for it.

####5. 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. 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 calculating the accuracy of the model is located in the tenth cell of the Ipython notebook.
First step of my testing was just converting the images to gray scale normalized and using the same network architecture as LeNet Lab. i.e Learning Rate = 0.001, 2 conv layer , 3 fully connected layers including output, 10 epoch , and batch size of 128. My validation accuracy was 95% approximately.
Next i changed the number of epoch to 20 and reached validation accuracy close to 97%.
Next i changed epoch back to 10 and batch size to 200 and i ended up with validation accuracy of 95%. No improvement from the general case.
Next was introducing dropout and L2 with 0.7 and 0.01 and their hyper parameters, used 20 epoch and 128 as my batch size and i reached accuracy of about 97.3 percent. Changing the the output layer input size slightly improved my accuracy to 98% approx.

My final model results were:
* Training set accuracy of 98.8%
* validation set accuracy of 98%
* test set accuracy of 91.87%

###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 images were taken from the link

https://en.wikipedia.org/wiki/Road_signs_in_Germany.
The first can be misclassified because they resemble like concentric circles which is also present in many other traffic signals. The second image can be misclassified as it doesn't have any distinctive feature.
The third image has a truck and can be confused with similar signals like toll road for heavy lorises or No traffic allowed without indicated minimum distance between vehicles from German dataset, bus lane only etc. Arrows ate common with many traffic signal signs which can be a problem for 4th image.
the fifth image is distinctive but has lot of information of if model doesn't have proper architecture it might have hard time finding these minute details.

####2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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 code for making predictions on my final model is located in the fifteenth cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
Roundabout Mandatory	Roundabout Mandatory
No vehicles	Priority Road
no vehicle over 3.5 metric tons	No vehicle over 3.5 metric ton
Straight Ahead	Straight Ahead
Road work	Roadwork

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This is comparably lower than the test accuracy a few possible reasons can be images are not in proper resolution after being resized. The images in the training and test set can be visualized much better. One possible solution can be using more images in each class with different image quality. I'm not sure if that will overfit the model but can be tried.

####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 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. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The code for making predictions on my final model is located in the 15th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a Roundabout mandatory , and the image does contain Roundabout Mandatory. As it can been seen for the soft max probabilities above the model predicts Roundabout Mandatory with

the highest probability, the next for classes it predicts are 11 which is right of way at next intersection, 12 priority road, 7 speed limit 100, 27 pedestrians One thing common about rest of the classes are that they either have a consectric shape or circle. Though the highest correctly predicted probability is only 11%.
For the second Image , the model predicts the image as Priority Road , thought the actual image is no vehicles. But as seen for the soft max values the third highest prediction is for class 15 which is in fact no vehicles. One possible reason for the confusion maybe concentric circle and no vehicles and concentric diamond in Priory road, which in model sense could have been interpreted as concentric same curves. Next 2 is 32 which is end of all speeding and passing limits which makes sense as it has also concentric circles like no vehicle. Next 4th is class 38 keep right followed by 7 which is 100km .
For the third image, the Model predicted correctly , image was actually no vehicle above 3.5 metric tons with a probability of 35%. Next are 40 roundabout mandatory, 7 which is 100 km , 9 no passing , 41 keep left. They are all similar in regards that they all have a outer circle with information/ arrow in the center.
For the fourth image , the model predicts correctly as straight ahead as expected with a high probability of close to 50%. the Next highest probability classes are 13 yield, 25 road work, 9 no passing ,36 go straight and right. It makes sense it predicts it as class 36 which is straight or right which in same was is similar looking to straight only. it should have not confused it with yield which is altogether a different shape.
For the fifth image , the model predicts road work and its intact road work. The highest probability 35 % , next is 38 which is keep right with a probability of 34% . Followed by classes 20 dangerous curve on right , 36 go straight or right, 34 turn left ahead. One noticeable thing about all next 4 classes is all have some kind of arrows , which if look at image 5 above makes sense as the man look like a straight arrow and the working equipment looks like a right arrow.