1 Data Set Summary & Exploration

1.1 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 first and second code cell of the Jupyter notebook.
I used the numpy library to calculate summary statistics of the traffic signs data set.
The following is the summary of the training, test and validation data set. As can be seen there are 43 unique classes.

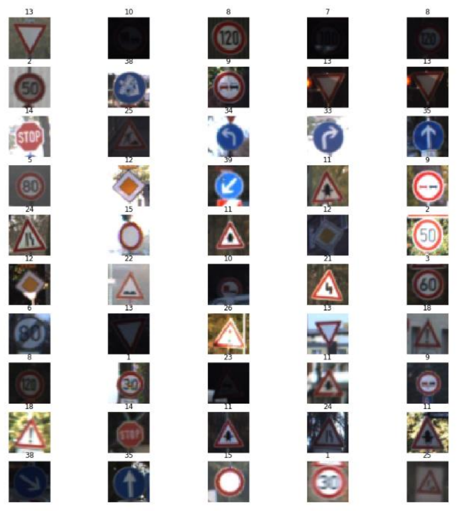
```
Training Features Attributes: (34799, 32, 32, 3)
Training Labels Attributes: (34799,)
validation Features Attributes: (4410, 32, 32, 3)
Validation Labels Attributes: (4410,)
Test Features Attributes: (12630, 32, 32, 3)
Test Labels Attributes: (12630,)

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

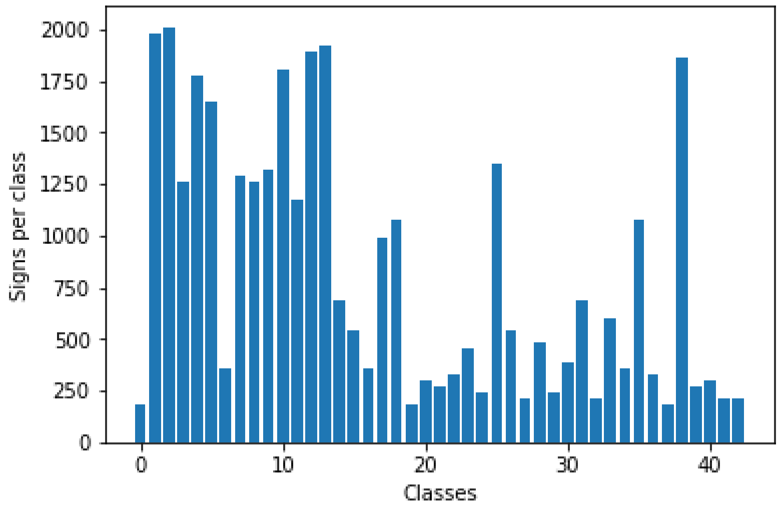
1.2 Include an exploratory visualization of the dataset and identify where the code is in your code file.

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

The below is the snapshot of the signs w.r.t. their labels



The below is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed w.r.t. the labels. This is shown in the 4th cell of the notebook. The bar chart was done using the matplotlib plots.



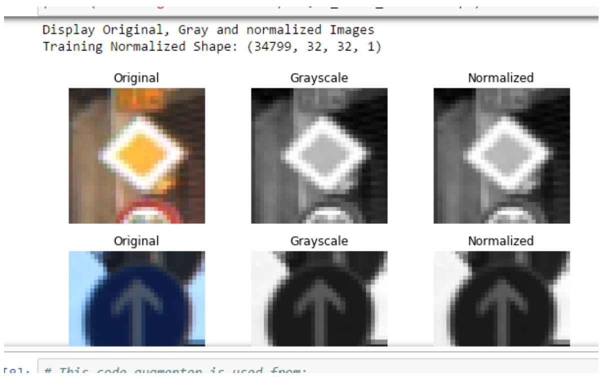
2 Design and Test a Model Architecture

2.1 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. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.

After getting the data I had done the grayscale conversion of the data and then I had also done the normalization of the data. Grayscale was done using the CV2 function cv2.cvtColor(). The grayscale was chosen as it increased the accuracy and as mainly the signs shape were need to be identified so the color information was not of much use. Moreover in the assignment it was **not** asked to differentiate the signs of different colors. For example:

Having the grayscale conversion done, would also save substantial memory and increase the speed.

The corresponding code is in the cell 5 and cell 6.



```
Display Original, Gray and normalized Images
Training Normalized Shape: (34799, 32, 32, 1)
```

Note: I had later removed the normalization as I had observed that after removing the normalization the accuracy was better. So the Above normalized image is same as grayscale image.

2.2 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)

As the training, validation and test data sets were separate there was no need to do any splitting.

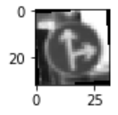
Augmentation of data set:
I had used the scipy.ndimage interpolation shift and rotate functions to distort the images in order to do the image augmentation. This was necessary as from the bar graph above it can be easily seen that the training data set was not balanced.

There were 2K samples from class 2 whereas there was merely 100 samples from the class 0. Because of this the network would be biased and would do the overfitting for the signs those represent the higher proportion. Precisely for this the data augmentation was done in the cell 9.

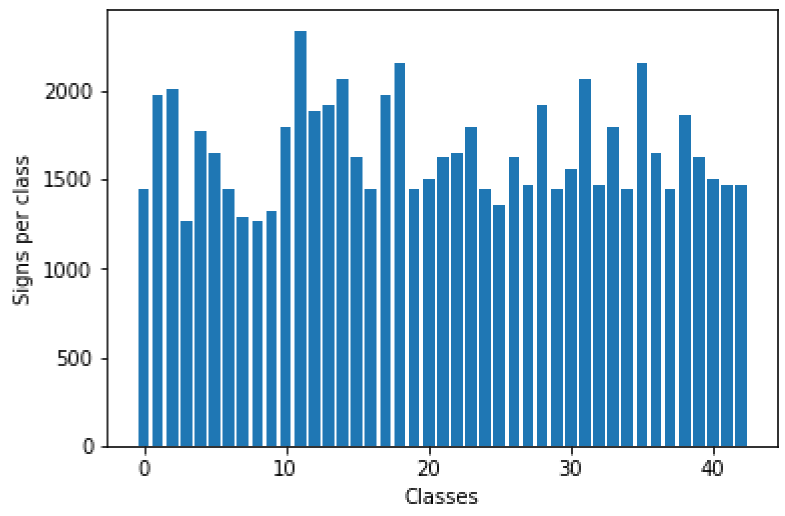
I had decided to augmented the underrepresented class to a specific number of images (1250/1000 based on the iteration) per class and below are the augmented number of images

```
Data Augmentation successfully done
Augmented Data Shape: (71397, 32, 32, 1)
Augmented Data Shape: (71397,)
```

The below image was also shown in cell 9:



The bar graph of the augmented data looks like below:



2.3 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 the CNN is in cell 11.

My model consisted of the following Layers:

Layer	**Description**
Input	32x32x1 Grayscale Image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Output 400
FC1	Output 120, mean = 0, stddev = 0.1
RELU	
FC2	Output 84, mean = 0, stddev = 0.1
RELU	

FC2	Output 43, mean = 0, stddev = 0.1

I had also tried the modified lenet based on my ideas, however the accuracy was poor it was very time consuming.

2.4 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.

Type of Optimizer: Adam
Learning rate (Final) = 0.001
EPOCHS = 28
BATCH_SIZE = 128

The training was done in the cells 12 to 16

The validation accuracy reached maximum limit of 94.5 (please see the attached sheet at the end of the page)

2.5 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 validation accuracy was measured in the cell: 16
The test accuracy was measured in the cell: 17

My final model results were: Please see the Iteration Details Table
The iterative approach was chosen: Please see the Iteration Details Table

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

- *LeNet as it's understood well and easy to begin with.*

What were some problems with the initial architecture?

- *Low accuracy for the Original Data converted to Gray Scale and the Normalization didn't improve the accuracy either.*

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

- *The EPOCHS, Learning Rate, Keep Probability, Batch Size. These parameters were tuned to find the maximum accuracy. And for my Model the increasing EPOCHS wasn't really helping. The* *learning rate for sure had an impact on the accuracy, so in the end I had chosen the 0.001 as the learning rate. The Batch size didn't have so much profound effect. The Keep probability was only relevant for the dropout that didn't work well*

How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing 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 architecture would be due to over fitting or under fitting. 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.

- *As you can see in the **Traffic_Sign_Classifier_13.0.ipynb** I had tried to introduce a my own network based on LeNet where the network width was higher as there were several more parameters and there are also dropout and concat layers, however may be I went to the extreme end of the parameters and the accuracy was pretty low 3.4, moreover it was taking 15 min for 1 EPOCH so I left it like that.*

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?

- *The dependable architecture was the LeNet architecture and in fact I had done more experiments on the data management techniques such as*
- *Gray Scale Conversion*
- *Normalization*
- *Augmentation*
- *Feeding the original data with and without the above 3 techniques*

The effect of the above techniques on the accuracy was well described above and it can be also clearly seen from the ***Iteration Details Table*** *in the next page.*

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

- *As can be seen in the iteration Details Table in the next page, after the data augmentation maximum measured validation accuracy was 94.3% and the test accuracy was 91.7%. This definitely proves that the model has reasonable accuracy.*

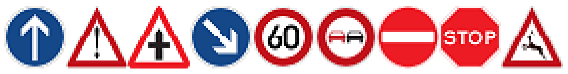
Iteration Details Table:

Sr. No	Name of the Notebook	Training Data		Validation Data		Test Data		EPOCHS	Batch Size	Aumented data Size	Learning Rate	mu	sigma	Network Architecture	Validation Accuracy		Test Accuracy	Accuracy for new Images	Comments
		X	y	X	y	X	y								Highest	Final			
1	Traffic_Sign_Classifier_1.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	10	128	71397	0.001	0	0.1	LeNet	78.5	78.5	77.7	Not Done	GrayScale and Normalization and Data Augmentation
2	Traffic_Sign_Classifier_2.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	10	128	71397	0.0009	0	0.1	LeNet	Not meaured	75.6	Not Measured	Not Done	Augmentation, length was wrongly fed, may need to repeat it
3	Traffic_Sign_Classifier_3.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	20	128	71397	0.0009	0	0.1	LeNet	94.1/93.6	94.1/93.6	91.3	Not Done	GraysScale Only, Normalization is Removed and Data Augmentation
4	Traffic_Sign_Classifier_4.0	X_train	y_train	X_valid	y_valid	X_test	y_test	20	128	34799	0.0009	0	0.1	LeNet	91.3	91.3	89.4	Not Done	Original Data, No Augmentation
5	Traffic_Sign_Classifier_5.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	50	100	71397	0.0009	0	0.1	LeNet	94.5	93.7	91.4	Not Done	GraysScale Only, Normalization is Removed and Data Augmentation
6	Traffic_Sign_Classifier_6.0	X_train_gray	y_train	X_valid_gray	y_valid	X_test_gray	y_test	20	128	34799	0.0009	0	0.1	LeNet	7.3	5.4	5.7	Not Done	Original Data only with gray scale, No Normaliuzation, No Augmentation
7	Traffic_Sign_Classifier_7.0	X_train_normal	y_train	X_valid_normal	y_valid	X_test_normal	y_test	20	128	34799	0.0009	0	0.1	LeNet	8.0	5.4	5.9	Not Done	Original Data With gray scale, Normalization and No Augmentation
8	Traffic_Sign_Classifier_8.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	20	128	44219	0.0009	0	0.1	LeNet	94.5	93.7	91.4	40, 5 Images	GraysScale Only, Normalization is Removed and Data Augmentation
9	Traffic_Sign_Classifier_9.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	20	128	71397	0.001	0	0.1	LeNet	93.7	92.2	89.9	88.9, 9 Images	GraysScale Only, Normalization is Removed and Data Augmentation
10	Traffic_Sign_Classifier_10.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	28	128	71397	0.001	0	0.1	LeNet	93.7	92.2	89.9	88.9, 9 Images	Same as 9, only the top 5 probability added
11	Traffic_Sign_Classifier_11.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	28	128	71397	0.001	0	0.1	LeNet	95.1	94.3	91.7	88.9, 9 Images	Same as 10, just an added iteration
12	Traffic_Sign_Classifier_12.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	28	128	71397	0.001	0	0.1	Lenet	Not Measured	Not Measured	Not Measured	Not Measured	Only the aumented image is shown.
13	Traffic_Sign_Classifier_13.0	X_train_augmented,	y_train_augmented	X_valid_normal	y_valid	X_test_normal	y_test	28	128	71397	0.001	0	0.1	My_Lenet	Could Not Measure	Could Not Measure	Could Not Measure	Could Not Measure	The accuracy was very low and each epoch was taking a lot of time hence couldnt successfully finish it.

3 Test a Model on New Images

3.1 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:



The first sign of ahead only have several common features with Go straight or left and Go straight and right and hence difficult to distinguish.
For the 2nd and 3rd image the general caution and give way sign has the vertical line in common hence these are difficult to distinguish.
The 60 speed limit sign was is also difficult to classify as the there are several other speed limits signs as well such as 20, 30, 40, 50, 70 and 80. Especially the 50 and 60 signs are similar are lots of common features hence easy to be misclassified.
No overtaking sign has again lots of common features as No overtaking by lorries, also priority for oncoming vehicles sign.
In my understanding the stop and no entry signs are easy to identify as these are unique signs.
Wild Animals crossing sign is challenging as it has a lot of common features with the several other Warning signs enclosed on triangle.

3.2 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 21st and 22nd cell.

Here are the results of the prediction:

Image	Prediction
Ahead Only	Ahead Only
General Caution	General Caution
Right of way	Right of way

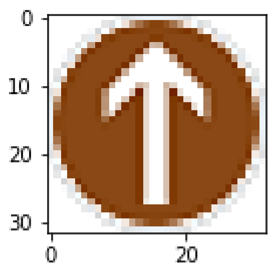
at the next intersection	at the next intersection
60 km/h	30 km/h
Keep Right	Keep Right
No Passing	No Passing
No Entry	No Entry
Stop	Stop
Wild Animals Crossing	Wild Animals Crossing

The model was able to correctly guess 8 of the 9 traffic signs, which gives an accuracy of 88.89%. This compares favorably to the accuracy on the test set of 89.9% to 91.7%.

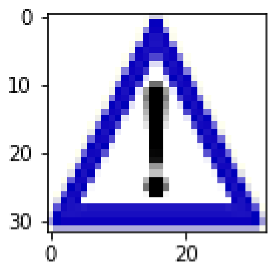
3.3 Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction and identify where in your code softmax probabilities were outputted. Provide the top 5 softmax probabilities for each image along with the sign type of each 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 22nd cell.

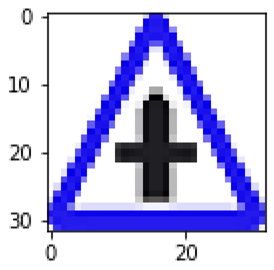
Below pictures clearly shows the softmax probabilities.



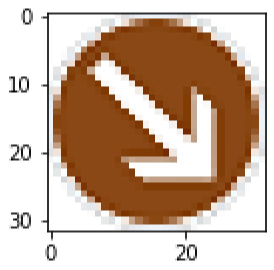
```
No 1 guess: Ahead only, Probability: 100.0
No 2 guess: Road work, Probability: 0.0
No 3 guess: Speed limit (20km/h), Probability: 0.0
No 4 guess: Speed limit (30km/h), Probability: 0.0
No 5 guess: Speed limit (50km/h), Probability: 0.0
```



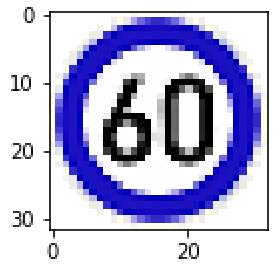
```
No 1 guess: General caution, Probability: 100.0
No 2 guess: Speed limit (20km/h), Probability: 0.0
No 3 guess: Speed limit (30km/h), Probability: 0.0
No 4 guess: Speed limit (50km/h), Probability: 0.0
No 5 guess: Speed limit (60km/h), Probability: 0.0
```



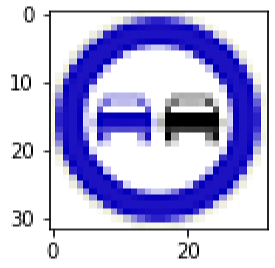
```
No 1 guess: Right-of-way at the next intersection, Probability: 99.47283864
No 2 guess: Traffic signals, Probability: 0.52715829
No 3 guess: Speed limit (30km/h), Probability: 0.0
No 4 guess: Double curve, Probability: 0.0
No 5 guess: Beware of ice/snow, Probability: 0.0
```



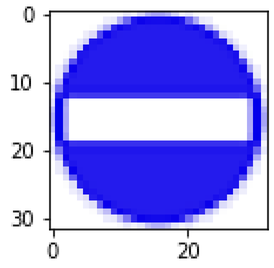
```
No 1 guess: Keep right, Probability: 100.0
No 2 guess: Stop, Probability: 0.0
No 3 guess: Speed limit (70km/h), Probability: 0.0
No 4 guess: Speed limit (20km/h), Probability: 0.0
No 5 guess: Speed limit (30km/h), Probability: 0.0
```



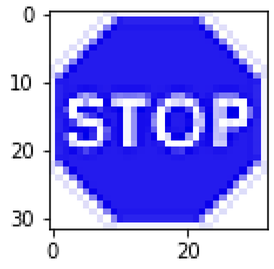
```
No 1 guess: Speed limit (30km/h), Probability: 100.0
No 2 guess: Speed limit (20km/h), Probability: 0.0
No 3 guess: Speed limit (60km/h), Probability: 0.0
No 4 guess: Speed limit (50km/h), Probability: 0.0
No 5 guess: Speed limit (70km/h), Probability: 0.0
```



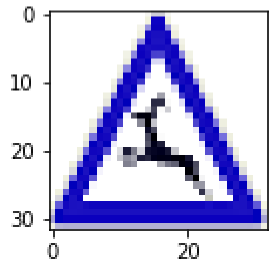
```
No 1 guess: No passing, Probability: 98.62852097
No 2 guess: End of no passing, Probability: 1.371432
No 3 guess: No entry, Probability: 5.113e-05
No 4 guess: Slippery road, Probability: 4.8e-07
No 5 guess: Vehicles over 3.5 metric tons prohibited, Probability: 0.0
```



```
No 1 guess: No entry, Probability: 100.0
No 2 guess: Speed limit (20km/h), Probability: 0.0
No 3 guess: Speed limit (30km/h), Probability: 0.0
No 4 guess: Speed limit (50km/h), Probability: 0.0
No 5 guess: Speed limit (60km/h), Probability: 0.0
```



```
No 1 guess: Stop, Probability: 100.0
No 2 guess: General caution, Probability: 0.0
No 3 guess: Speed limit (20km/h), Probability: 0.0
No 4 guess: Speed limit (70km/h), Probability: 0.0
No 5 guess: End of speed limit (80km/h), Probability: 0.0
```



```
No 1 guess: Wild animals crossing, Probability: 100.0
No 2 guess: Slippery road, Probability: 0.0
No 3 guess: Dangerous curve to the left, Probability: 0.0
No 4 guess: Speed limit (20km/h), Probability: 0.0
No 5 guess: Speed limit (30km/h), Probability: 0.0
```

4 References

https://hackernoon.com/traffic-signs-classification-with-deep-learning-b0cb03e23efb#.p709ztxxf https://github.com/MehdiSv/TrafficSignsRecognition/blob/master/final_Traffic_Signs_Recognition.ipynb

https://medium.com/@vivek.yadav/dealing-with-unbalanced-data-generating-additional-data-by-jittering-the-original-image-7497fe2119c3#.s684vz343

https://github.com/vxy10/ImageAugmentation

https://medium.com/@ckyrkou/udacity-sdc-nanodegree-term-1-project-2-traffic-sign-classifier-8d8a30e735b8#.wmfotylet

https://github.com/ckyrkou/Traffic_Sign_Classifier/blob/master/Traffic_Sign_Classifier.ipynb

https://github.com/hello2all/CarND-Traffic-Sign-Classifier-Project

https://github.com/paramaggarwal/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

https://github.com/jeremy-shannon/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

https://en.wikipedia.org/wiki/Road_signs_in_Germany

https://medium.freecodecamp.com/recognizing-traffic-lights-with-deep-learning-23dae23287cc#.iglnjs532