Data Set Summary & Exploration

1. Basic summary of the data set

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

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

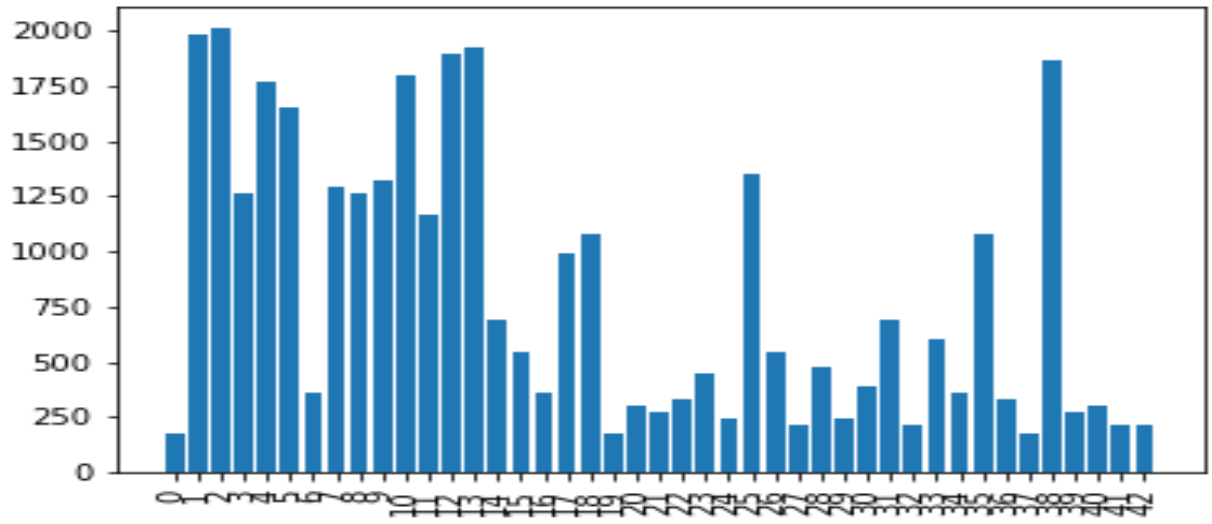
- The original size of training set is 34799
- The size of test set is 12630
- The shape of a traffic sign image is 32x32x3
- The number of unique classes/labels in the data set is 43

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.

The data set counts for each class is visualized with a bar chart showing how the data is distributed between each unique class. The count values for each class are:

```
dict_values([180, 1980, 2010, 1260, 1770, 1650, 360, 1290, 1260, 1320, 180
0, 1170, 1890, 1920, 690, 540, 360, 990, 1080, 180, 300, 270, 330, 450, 24
0, 1350, 540, 210, 480, 240, 390, 690, 210, 599, 360, 1080, 330, 180, 1860
, 270, 300, 210, 210])
```



For visulisation purposes, 20 random images where displayed to get an idea of the data that was to be classified. The code is implemented in cell 3.

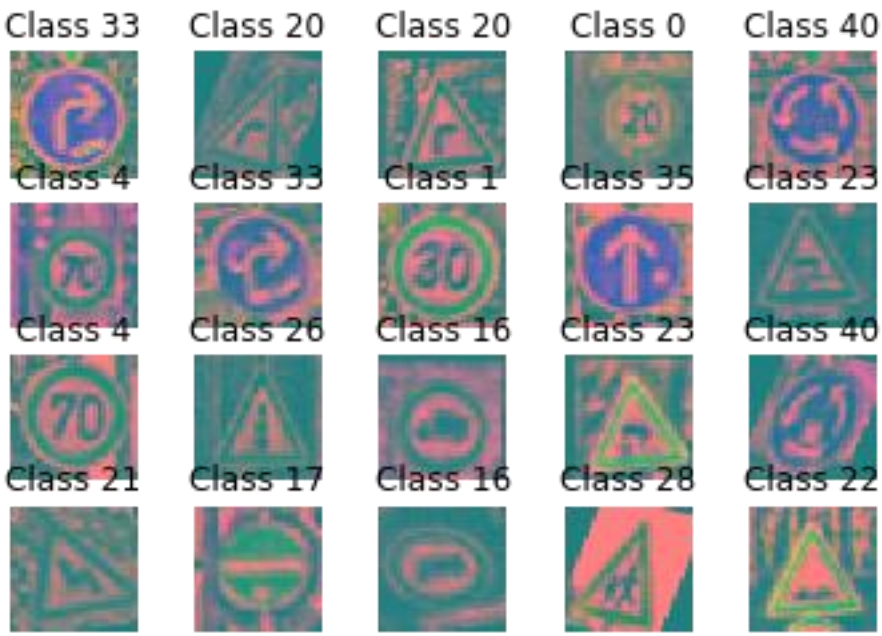
Design and Test a Model Architecture

1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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 seventh and eighth code cells of the IPython notebook.

I decided to convert the images to YUV scale because I could remove brightness variations between images, creating a more normalized data set for the model to learn. The Images Y scale was clipped to either darken or brighten up the image.

Here is an example of 20 random traffic signs after YUV scaling.



I normalized the image data because the model originally tested had low accuracy and a lot of the data appeared to the human eye to be black.

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)

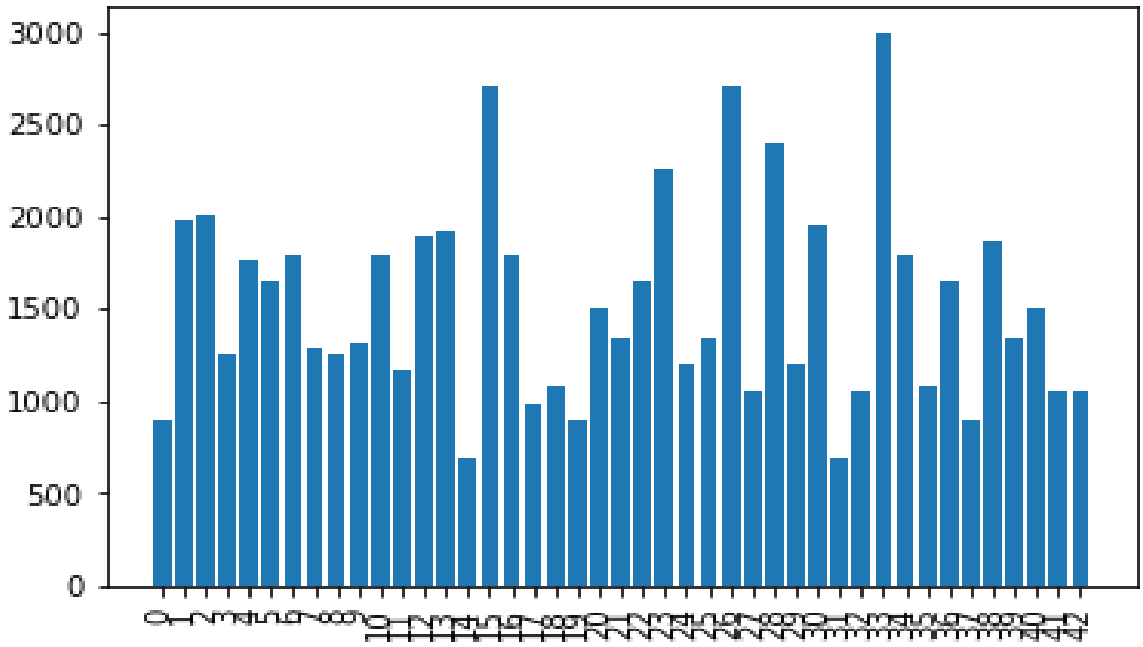
My final training set had 65755 number of images. My validation set and test set had 4410 and 12630 number of images.

Due to low data in certain classes had low test counts. It was deemed that the low valued data should be increased to help the model train. The code for this is in cells 4 and 5 and implements random translation and augmentation of the images to add extra data.

The resulting new dataset counts as implemented in cell 6 is 65755 and the data spread is shown in the dictionary counts and histogram below.

```
dict_values([900, 1980, 2010, 1260, 1770, 1650, 1800, 1290, 1260, 1320, 18
00, 1170, 1890, 1920, 690, 2700, 1800, 990, 1080, 900, 1500, 1350, 1650, 2
```

```
250, 1200, 1350, 2700, 1050, 2400, 1200, 1950, 690, 1050, 2995, 1800, 1080
, 1650, 900, 1860, 1350, 1500, 1050, 1050])
```



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 10th cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 YUV image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x64
Batch Norm	Batch normalized with data mean and variance
RELU	
Dropout	Train dropout of 0.6, test and valid with 1

Layer	Description
Max pooling	2x2 stride, outputs 14x14x64
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x64
Batch Norm	Batch normalized with data mean and variance
RELU	
Dropout	Train dropout of 0.6, test and valid with 1
Max pooling	2x2 stride, outputs 5x5x64
Fully connected	Flatten convolution data to 1600
Hidden 1	Input features = 1600, output hidden layer = 1200
RELU	
Dropout	Train dropout of 0.6, test and valid with 1
Hidden 2	Input features = 1200, output hidden layer = 600
RELU	
Dropout	Train dropout of 0.6, test and valid with 1
Hidden 3	Input features = 600, output hidden layer = 43
Softmax	Softmax with cross entropy

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 cells 11, 12, 13 and 14 of the ipython notebook.

To train the model, I used an the loss of the system from the output of the softmax optimized with the Adam gradient decent method. The learning rate implemented was a random uniform log value (cell 14) as this helped both speed up, slow down the learning rate minimizing the potential for the model to get stuck in local minima.

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 cell 13 of the Ipython notebook.

My final model results were:

- validation set accuracy of 0.965
- test set accuracy of 0.975

If an iterative approach was chosen:

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

  The first architecture was the one implemented in the model presented. Ather models were tested, from no max pooling and several convolutional layers, different sized filters, smaller and larger final connected layers. As well as the attempt to implement inception layers as outlined in the GoogleNet model.

- What were some problems with the initial architecture?

  The later architectures produced either worse results or were extremely computationally in efficient. The original was chosen as it provided a happy median and a good accuracy

- How was the architecture adjusted and why was it adjusted?

  Due to my inexperience in Deep learning and fine tuning. A lot of the hyper parameters and models were implemented to get a better understanding of "black box" and what it can do. Through this experimentation and reading articles, the data and hyperparameters were finely tuned to what created the best results. Batch norm was observed to increment the accuracy by 10% and the dropout was added when overfitting was observed between the loss and accuracy.

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

  Drop out, learning rate, layer depth, hidden unit counts. Mostly through trial and error based of what I have read, heard and seen others do.

- 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?

  Convolutional layers are great at images as they can find shapes lines or colours of the same object anywhere with in the image.

If a well known architecture was chosen:

- What architecture was chosen?

  LeNets

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

  It was good on classifying digits and was easy to expand and implement

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:

Image 1: This image was chosen as it is the least augmented image. The sign is clear, centered and the image has very little background noise. The model should be able to verify this sign, however considering the noisy training data, a clear image may be miss interpreted if the model has classified noise as a part of the sign.

Image 2: Clear, moderately centered, low background noise. Should have no issue classifying unless noise has been considered relevant to the image in the training data.

Image 3: Clear, slight augmentation. Another sign underneath providing extra image noise which could cause problems with classification.

Image 4: Sign slightly larger than image which could cause problems if that signs classification requires a nice clear edge. The image is also large which could potentially cause the model to be unsure of correct classification between other round and blue signs.

Image 5: Off centered, blurred, small and noisy background image. These will challenge the models classification of this sign type for all the reason outlined. Classification of this image will heavily depend on the quality of the data supplied for training.

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 tenth cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
Class 27	27
2	2
18	18
40	40
17	12

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares reasonably well for only 5 images.

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.

The code for making predictions on my final model is located in the cells, 16, 17, 18 and 19 of the Ipython notebook.

For the second image class 27

Probability (%)	Prediction
38.61	27
15.344	18
14.26	25
10.025	11
8.898	28

For the second image class 2

Probability (%)	Prediction
96.62	2
3.37	1
4.17424785e-03	5
3.30335824e-05	8
1.45174399e-05	3

For the Third image class 18

Probability(%)	Prediction
100	18
5.95074511e-12	26

Probability(%)	Prediction
8.22699554e-15	21
8.60741448e-17	15
1.54510340e-17	11

For the Fourth image class 40

Probability(%)	Prediction
99.62	40
3.71629536e-01	16
2.15807324e-03	39
1.29696040e-03	33
7.94464431e-04	36

For the Fifth image 17

Probability(%)	Prediction
94.13	12
3.051	34
1.296	30
0.858	40

Probability(%)	Prediction
0.539	32