Traffic Sign Recognition Project

The goals/steps of this project are the following :-

- Load the 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

Explore, summarize and visualize the model

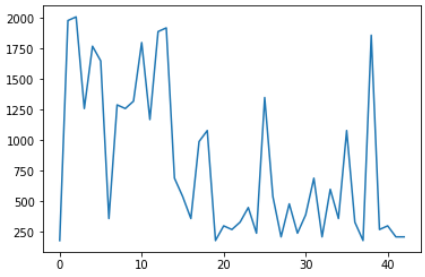
The code to Explore the dataset is in the second cell.
I used python and numpy methods for knowing the summary statistics.
The size of the training set is 34799
The size of the validation test set is 4410
The size of the test set is 12630
Shape of the traffic sign image is (32,32,3)
Number of unique classes in the dataset is 43

Visualization

The code to visualize the dataset is in the 3rd and 4th cells. In the below graph the traffic sign labels (for training data) are shown in the x-axis and the count of photos are shown in the Y-axis. As can be seen from the graph, dataset is not balanced.

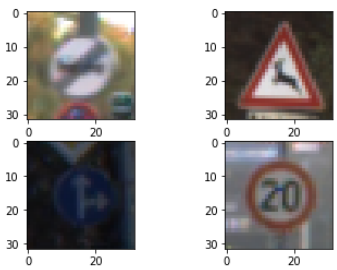
```
# Visualizations will be shown in the notebook.
%matplotlib inline
```

```
dict_items([(0, 180), (1, 1980), (2, 2010), (3, 1260), (4, 1770), (5, 1650), (6, 360), (7, 1290), (8, 1260), (9, 132
0), (10, 1800), (11, 1170), (12, 1890), (13, 1920), (14, 690), (15, 540), (16, 360), (17, 990), (18, 1080), (19, 18
0), (20, 300), (21, 270), (22, 330), (23, 450), (24, 240), (25, 1350), (26, 540), (27, 210), (28, 480), (29, 240), (3
0, 390), (31, 690), (32, 210), (33, 599), (34, 360), (35, 1080), (36, 330), (37, 180), (38, 1860), (39, 270), (40, 30
0), (41, 210), (42, 210)])
```



Visualisation of some random images:-

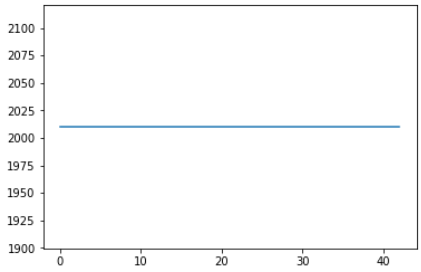
```
<matplotlib.image.AxesImage at 0x19839fd0e
```



Data Preprocessing:-

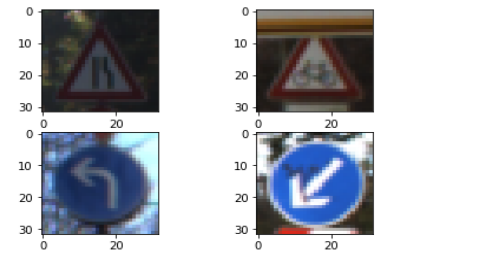
As machine learning algorithms work well on large balanced datasets, I did oversampling of underrepresented class labels. The function used is RandomOverSampler from the package imblearn. After data augmentation by oversampling the count of all images went up to 2010 for all the class labels. The graph after data augmentation is as follows:-

```
dict_items([(0, 2010), (1, 2010), (2, 2010), (3, 2010), (4, 2010), (5, 2010), (6, 2010), (7, 2010), (8, 2010), (9, 20
10), (10, 2010), (11, 2010), (12, 2010), (13, 2010), (14, 2010), (15, 2010), (16, 2010), (17, 2010), (18, 2010), (19,
2010), (20, 2010), (21, 2010), (22, 2010), (23, 2010), (24, 2010), (25, 2010), (26, 2010), (27, 2010), (28, 2010), (2
9, 2010), (30, 2010), (31, 2010), (32, 2010), (33, 2010), (34, 2010), (35, 2010), (36, 2010), (37, 2010), (38, 2010),
(39, 2010), (40, 2010), (41, 2010), (42, 2010)])
```

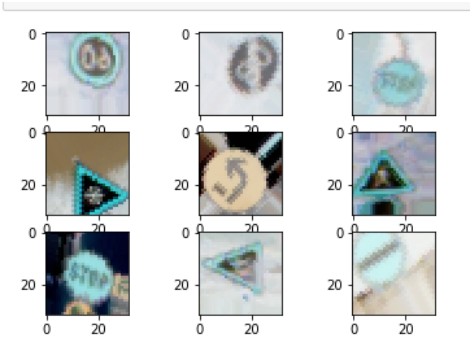


Visuvalisation of some images after data Augmentation:-

```
<matplotlib.image.AxesImage at 0x1983cae3320>
```



Images are further processed with zca_whitening and they are randomly rotated, flipped, horizontally and vertically shifted and zoomed to improve the accuracy of neural network training. The sample output of images after all these data preprocessing steps are as follows:-



As a final step the pixel values are normalized within a range between 0 and 1.

The code for splitting the dataset is in the first cell of IPython notebook. To avoid leaking of information when training and validation is carried out test set is also created. Finally,after data augmentation the count of train, validate and test images are as follows:-

Training images are 86430.
Validation images are 4410.
Test images are 12630.

Model Architecture

The code for the model architecture is in the 12th cell of Ipython notebook. The model architecture is as follows:-

Layer	Description
Input	32x32x3 RGB Image
Convolution 5x5	1x1 stride, valid padding, output is 28x28x6
Activation	RELU
Max pool	Size 2x2, stride 2x2, output is 14x14x6
Convolution 5x5	1x1 stride, valid padding, output is 10x10x16
Activation	RELU
Max pool	2x2 size, 2x2 stride, output is 5x5x16
Flatten	Output is 400
Activation	RELU
Fully Connected	Output is 120
Activation	RELU
Fully connected	Output=84
Activation	RELU
Fully connected	Output = 43
Activation	Softmax cross entropy with logits

The code for training the model is located in the 14th cell of IPython notebook. The optimizer used is AdamOptimizer, 40 Epochs with a batch size of 256. The learning rate used is 0.001. Dropout of 0.7 along with l2 regularization of 0.15 was used.

The function for calculating the accuracy of the model is located in the 15th cell and the accuracy of the model is calculated in the 16th and 17th cell.

The final model results are as follows:-
Training Set accuracy is 99.3%
Validation Set accuracy is 93%
Test set accuracy is 94.2%

The final model is based on fine tuning the ImageNet architecture. For fine tuning the architecture iterative approach was followed. As the original dataset was imbalanced, it was decided to augment, preprocess the dataset and to use transfer learning for classifying the traffic signal images. ImageNet was giving 89% accuracy without any modification. If it is finetuned and changed according to the problem at hand, it will give better accuracy in a short time frame. In addition, as the architecture of ImageNet is not very dense it is suitable for implementing on a local cpu. So it was chosen for this particular dataset.

My initial model faced problem with overfitting. The challenge from starting was to overcome overfitting. To avoid overfitting dropout was used. Dropout reduced both the training and validation accuracy. So different values for dropout was tested and finally a value of 0.7 was found to be suitable. To further reduce the overfitting l2 regularization was introduced and tested with several values.

To further increase the accuracy of the model additional convolution layers were introduced in the architecture. But results were not good. The accuracy of the model dropped with the addition of convolution layers. Different filter sizes were also tried.

Other parameters which were tuned are Epoch and Batch size. Different values of Epoch and Batch size was tested. For this model, a batch size of 256 and Epoch of 25 was found suitable. Due to the limited computational capability available very large values of Epoch could not be tested.

Additionally, learning rate was increased. The accuracy of training and validation was found to be fluctuating. So, the learning rate was again reduced to a value where the test and validation accuracy was found to be stable.

If a model is able to generalize well on data which it has not seen before than it is considered good. The accuracy on test data which the model dint see before is 94.2%. As the accuracy of test data is very high, it can be concluded that the model is doing very well.

Test model on new images:-

The five german signs I found on the web are as follows:-



Model should predict Image 1 correctly, as there isn't any noise in it.
For image 2 the prediction should be slightly hard as there is "Getty Images" caption written on the image. But it should not be very difficult.
For image 3 the prediction should be challenging as there are other boards on the image.
For images 4 & 5 the model should predict it correctly, as they are very straightforward.

The code for predicting the images is in the 22nd cell of the ipython notebook.

The results of the prediction are as follows: -

Image	**Prediction**
Road work	Road Work
Roundabout compulsory	Roundabout compulsory
Speed limit 70 (km\hr)	Bicycle crossing
Bumpy Road	Bumpy Road
Road Narrows on the Right	Road Narrows on the Right

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of 94.2%.

The code for making predictions on my final model is located is in the 24th cell of the Ipython notebook.

For the **first image**, the model is relatively sure that this is a stop sign (probability of 1.0), and the image does contain a Road work sign. The top five soft max probabilities were

Probability	**Prediction**
1.0	Road Work
0	Wild Animal crossing
0	Bumpy Road
0	Bicycle crossing
0	Traffic signals

For the **second image**, the model is relatively sure that this is a Roundabout Mandatory sign (probability of 1.0), and the image does contain a Roundabout Mandatory sign. The top five soft max probabilities were

Probability	**Prediction**
1.0	Roundabout Mandatory
0	General Caution
0	Go Straight or right
0	Go Straight or left
0	Keep left

For the **third image**, the model is not sure that this is a speed limit 70 sign (probability of 0.0), and the image does contain a speed limit 70 sign. The top five soft max probabilities were

Probability	**Prediction**
0.4	Bicycle Crossing
0.2	Traffic signals
0.15	Children crossing
0.09	Speed limit 20
0.05	Speed limit 120 km\hr

For the **fourth image**, the model is relatively sure that this is a Bumpy road sign (probability of 1.0), and the image does contain a Bumpy road sign. The top five soft max probabilities were

Probability	**Prediction**
1.0	Bumpy road
0	Road work
0	Traffic signal
0	Dangerous curve to the right
0	General caution

For the **fifth image**, the model is relatively sure that this is a road narrows to the right sign (probability of 1.0), and the image does contain a road narrows to the right sign. The top five soft max probabilities were

Probability	**Prediction**
1.0	Road narrows to the right
0	Traffic signals
0	Pedestrians
0	Children crossing
0	Vehicles over 3.5 Tons prohibited