Traffic Sign Classifier Project

1. **Data Set Summary & Exploration (Cell 2, 3, 4 on Ipython Notebook)**
   - I used the pandas library to visualize the display the summary of the statistics of the traffic sign data

   a) The size of the training set is **34799**

   b) The size of the validation set **is 4410**

   c) The size of the test is **12630**

   d) The shape of the traffic sign image is **(32,32,3)**

   e) The number of unique classes/labels in the data set is **43**

The figure below shows the distribution of the data

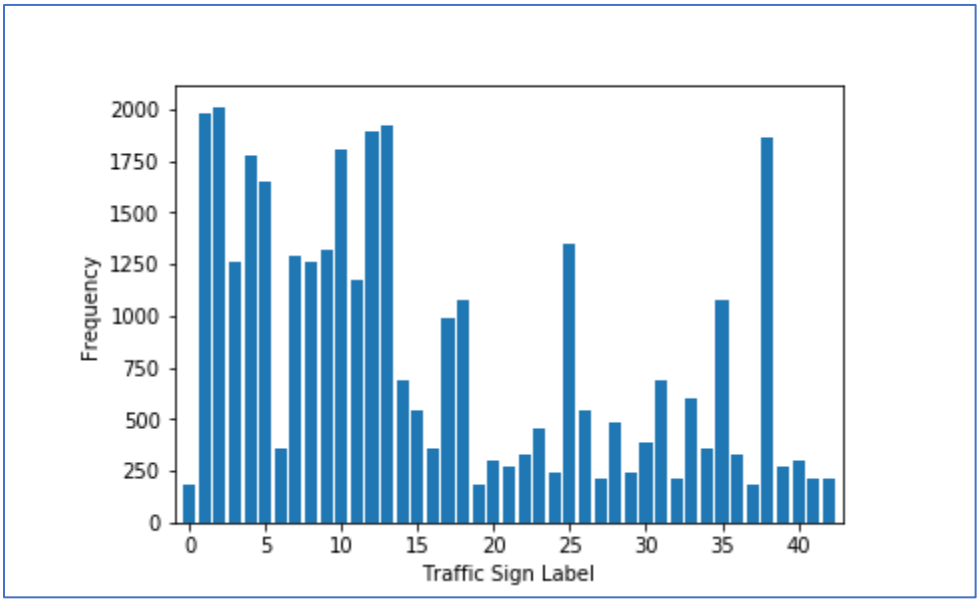


Figure 1 Data Summary

The figure below shows the sample visualization of the data. The full visualization is shown in the Ipython notebook file

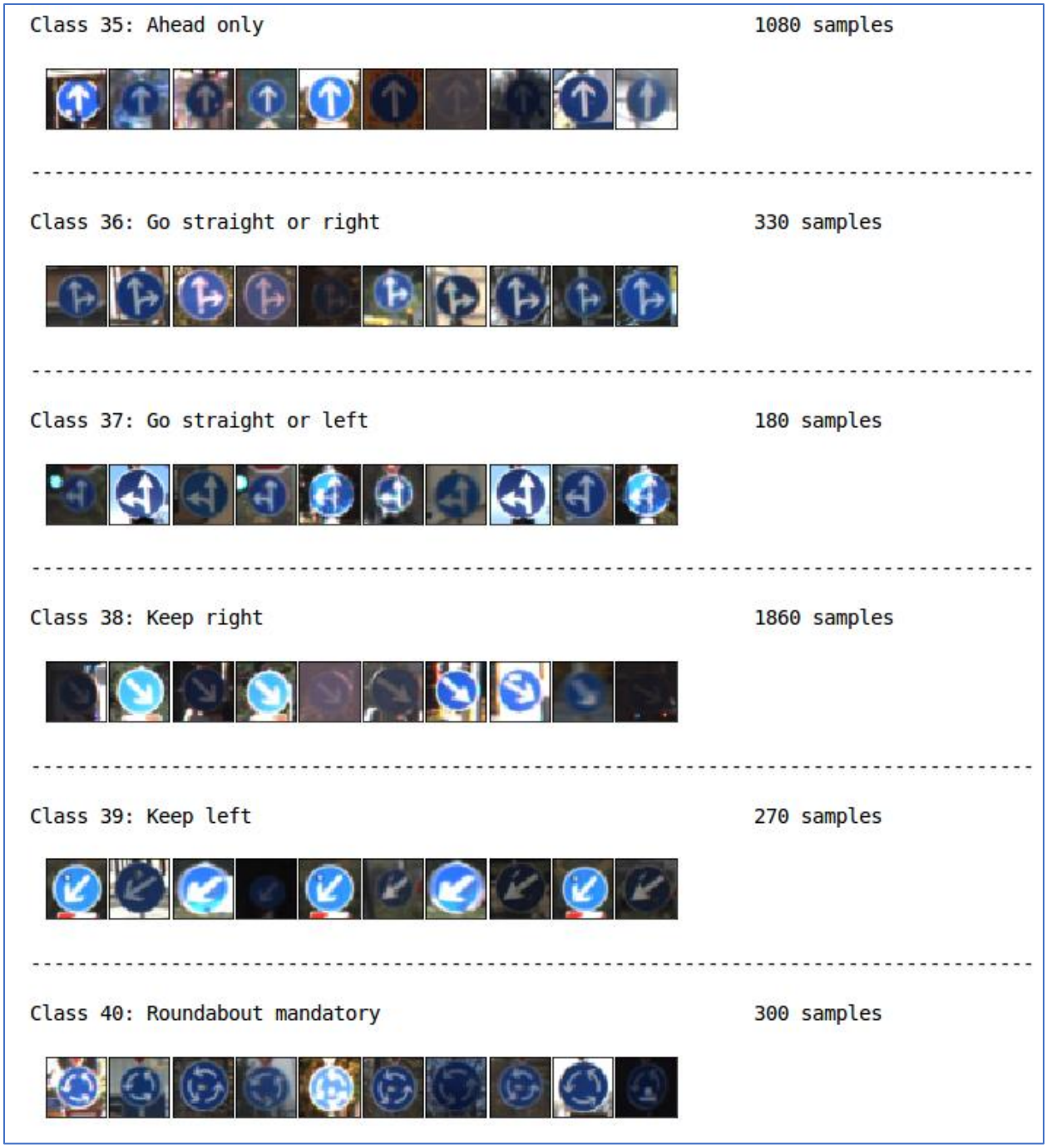


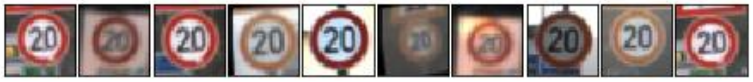
Figure 2 Data Visualization

2. Data Augmentation (Cell 5)

The data augmentation technique used is rotation and translation. At first shear was also chosen as a technique but was dropped due to extreme distortion of the test images. Brightness augmentation is also used. The image is first converted to HSV, then brightness augmentation is done and then converted back to RGB.

The augmentation techniques are applied randomly to the training images before training so that the augmentation is not biased and unpredictable. The figure below shows the comparison of the original image and the augmented image. **Note: The image augmentation is done before normalizing the images**

```
Class 0: Speed limit (20km/h)                                    180 samples
```



```python
def augment_brightness_camera_images(image):
    outimageHSV = np.uint8(image)
    image1 = cv2.cvtColor(outimageHSV,cv2.COLOR_RGB2HSV)
    image1 = np.array(image1, dtype = np.float64)
    random_bright = .5+np.random.uniform()
    image1[:,:,2] = image1[:,:,2]*random_bright
    image1[:,:,2][image1[:,:,2]>255] = 255
    image1 = np.array(image1, dtype = np.uint8)
    image1 = cv2.cvtColor(image1,cv2.COLOR_HSV2RGB)
    return image1

def transform_image(image,ang_range,shear_range,trans_range):

# Rotation

    ang_rot = np.random.uniform(ang_range)-ang_range/2
    rows,cols,ch = image.shape
    Rot_M = cv2.getRotationMatrix2D((cols/2,rows/2),ang_rot,1)

# Translation
    tr_x = trans_range*np.random.uniform()-trans_range/2
    tr_y = trans_range*np.random.uniform()-trans_range/2
    Trans_M = np.float32([[1,0,tr_x],[0,1,tr_y]])

    image = cv2.warpAffine(image,Rot_M,(cols,rows))
    image = cv2.warpAffine(image,Trans_M,(cols,rows))

    #Brightness augmentation
    image = augment_brightness_camera_images(image)

    return image

# Random Data Augmentation
a=0
while a<=len(X_train):

    for i in range(1):
        #ax1 = plt.subplot(gs1[i])
        #ax1.set_xticklabels([])
        #ax1.set_yticklabels([])
        #ax1.set_aspect('equal')
        X_train[a]= transform_image(X_train[a],20,10,5)
    a+=random.randint(1,7)
    if a> 34799:
        print('Done')
        break
```

3. Data preprocessing (Cell 6)

The image normalization technique is used to preprocess the dataset. The pixel values after normalization lies between -1 and 1:

New Pixel Value = (Old Pixel value/122.5) -1

Normalization is used to convert the data into a well-conditioned dataset with zero mean and equal variance. If the dataset is badly conditioned, then the optimizer has to do a lot a searching to best find the best solution.

Figure 3 Image Before Normalization

Figure 4 Image After Normalization

4. Final Model Architecture (Cell 7, 8, 9, 10,12)

Initially, the Lenet architecture was chosen for training. Iterations were carried out by tuning the hyper- parameters but the validation accuracy was low and the training accuracy was high which proved that there was over-fitting. So, the architecture was changed by following the steps below:

a) Increased the depth of the filters from 6 to 20 in the first convolutional layer
b) Increased the depth of the filters from 8 to 40 in the second convolutional layer
c) Max pooling
d) Added One extra fully- connected layer
e) L2-regularisation on the convolutional layer
f) Dropouts of different magnitude in the fully connected layers.

The best combination was arrived at after iterations and the model summary is shown in the table below

During the training, the Adam Optimizer is used

Sno	Layer	Description
1	Input	RGB Image 32x32x3
2	Convolution 5x5	Padding= VALID, Strides = 1x1 Output - 28x28x20
3	RELU	-
4	Max Pooling	2x2 Stride , Output= 14x14x20
5	Convolution 5x5	Padding= VALID, Strides = 1x1 Output - 10x10x40
6	RELU	-
7	Max Pooling	2x2 Stride , Output= 5x5x40
8	Flatten	1000 Outputs
9	Fully-connected 1	Dropout = 0.9 , 300 Outputs
10	Fully-connected 2	Dropout = 0.7 , 120 Outputs
11	Fully-connected 3	Dropout = 0.7 , 84 Outputs
12	Logits Layer	Dropout = 0 , 43 Outputs

- Hyperparameters:

  EPOCHS – 15
  BATCH SIZE - 128

The results of the training are shown below:

- Training Set Accuracy – 99.38%
- Validation Set Accuracy – 96.1%
- Training Set Accuracy – 94.17%

The accuracy and loss plots were shown below:

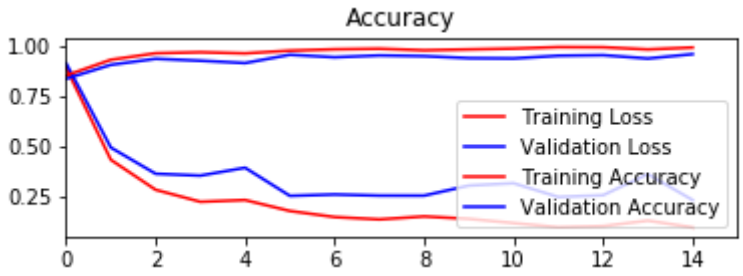


Figure 5 Accuracy and Loss curve

5. Model Performance on New Image Prediction (17,18,43,30) –

The new images selected and the corresponding labels are shown below

Sno	Image Label	Sign name
1	14	Stop Sign
2	5	Speed Limit - 80
3	39	Keep Left
4	11	Right-of-way at the next intersection
5	17	No entry

The images were resized to 32x32 and are shown below:

The model was able to correctly guess the all the 5 signs correctly which gives an accuracy of 100%.

The predictions and the top 5 softmax probabilities are shown below:

Prediction for Stop Sign		
Probability	**Prediction**	**Label**
0.83	Stop Sign	14
0.057	Speed Limit (20km/hr)	0
0.053	Speed Limit (30km/hr)	1
0.023	Speed Limit (70km/hr)	4
0.023	Bicycles Crossing	29

Prediction for Speed Limit (80km/hr)		
Probability	**Prediction**	**Label**
0.998	Speed Limit (80km/hr)	5
0.0016	Speed Limit (80km/hr)	3
0.0003	Speed Limit (30km/hr)	1
9.01E-07	Speed Limit (50km/hr)	2
3.59E-08	End of Speed Limit (80km/hr)	6

Prediction for Keep Left		
Probability	**Prediction**	**Label**
1	Keep Left	39
7.48E-12	Roundabout mandatory	40
9.90E-13	Turn right ahead	33
4.40E-14	Go straight or left	37
3.19E-16	Keep right	38

Prediction for Right of way at the next intersection		
Probability	**Prediction**	**Label**
1	Right of way at the next intersection	11
3.45E-11	Beware of ice/snow	30
7.09E-15	Pedestrians	27
2.21E-15	Double curve	21
2.20E-16	Dangerous curver to the left	19

Prediction for No entry		
Probability	**Prediction**	**Label**
1	No Entry	17
8.10E-30	Speed Limit (20km/hr)	0
6.02E-31	No Passing	9
2.96E-31	Stop	14
1.87E-31	No Passing for vehicles over 3.5 metric tons	10

As seen in the table above the model is absolutely sure about the signs