Problem statement

Objective

With the advent of AI, there is a wide range of applications. One of these applications is automatic cars. These cars can operate through artificial intelligence without any human interference. However, for the cars to run smoothly , they need to understand traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc.. Thus in this project, we will build a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles.

Aim

To create a system that can detect and recognise traffic signs in real time using Pytorch

About the dataset

The dataset used for training is German Traffic Sign Recognition Benchmark (GTSRB) containing 43 classes of traffic signs. The training set contains 39209 labeled images and the test set contains 12630 unlabelled images.

Solution

Methodology for Model creation and evaluation

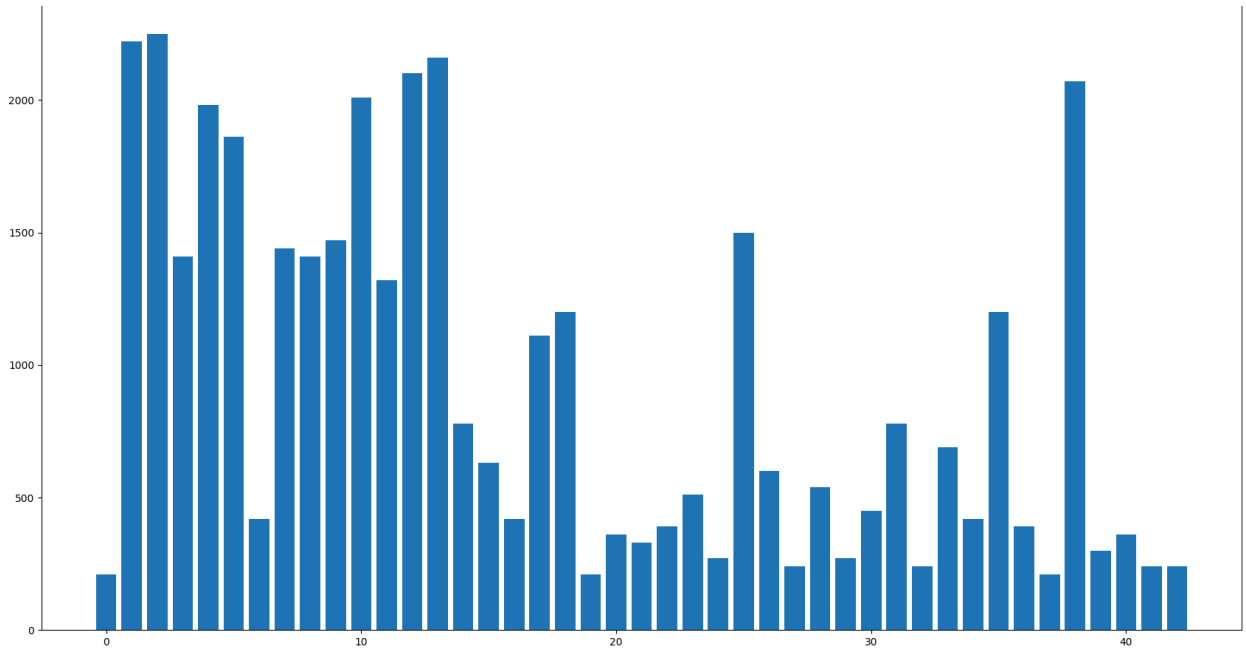
1. Loading relevant libraries, and alias where appropriate
    a. Libraries and alias used :
        i. Numpy as np
        ii. Torch : nn
        iii. Torchvision
        iv. Matplotlib as plt
        v. Seaborn as sns
        vi. PIL
        vii. Pandas as pd
        viii. os
        ix. google.colab
        x. Torchsummary
        xi. Prettytable
        xii. Time
        xiii. Datetime
        xiv. sklearn
2. Exploratory Data Analysis (EDA)
3. Creating Loaders
    a. Defining Transforms by performing various operations on images
    b. Defining class for dataset as GTRSBDataset
        i. Getting image path and bounding box coordinates
        ii. Cropping image based on Region of Interest (ROI)
    c. Creating training and testing datasets
    d. Creating validation dataset from training dataset
    e. Creating dataloaders from training , validation and testing datasets using batch size of 64

4. Defining model
    a. Defining a class for the model by the name of TrafficSignalNet containing three convolutional blocks and a fully connected block
    b. Creating model and moving to device (gpu/cpu)
5. Training the model
    a. Initializing tracking variables
    b. Calculating total batches for progress tracking on the basis of batch size
    c. Training phase
        i. Forward phase
        ii. Backward phase
        iii. Calculating ETA
    d. Calculate average training loss
    e. Validation phase
    f. Early Stopping Check
    g. Printing results
    h. Plotting the loss graph
6. Testing and evaluating the model
    a. Testing the model
    b. Printing accuracy , F1score , Precision and Recall
    c. Printing Classification Report
    d. Visualizing confusion matrix
    e. Plotting the confusion matrix
7. Saving the model
    a. Save the model in the form of a ‘.pth’ extension

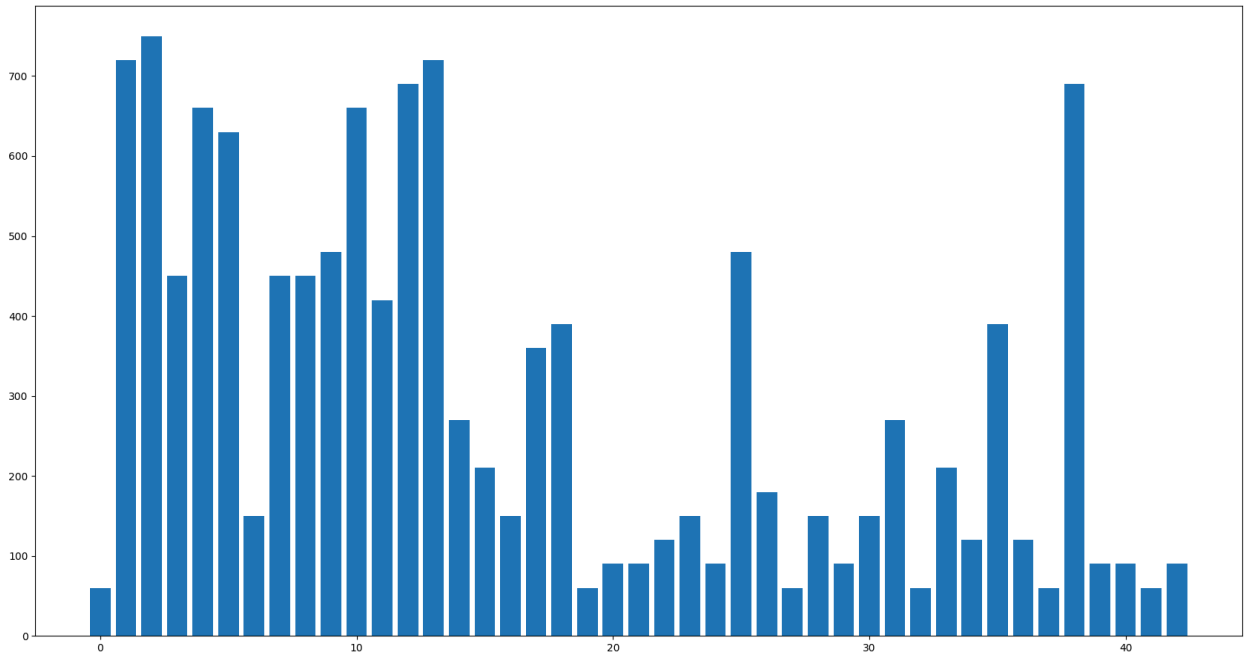
Methodology for Real Time interference

1. Loading relevant libraries, and alias where appropriate
    a. Numpy as np
    b. Cv2
    c. Torch : nn
    d. Torchvision
2. Initializing Video capture
3. Initializing FPS counter
4. Setting model to evaluation mode
5. Displaying results on frame
6. Displaying frame in Jupyter notebook
7. Defining transform
8. Calling the function with the model

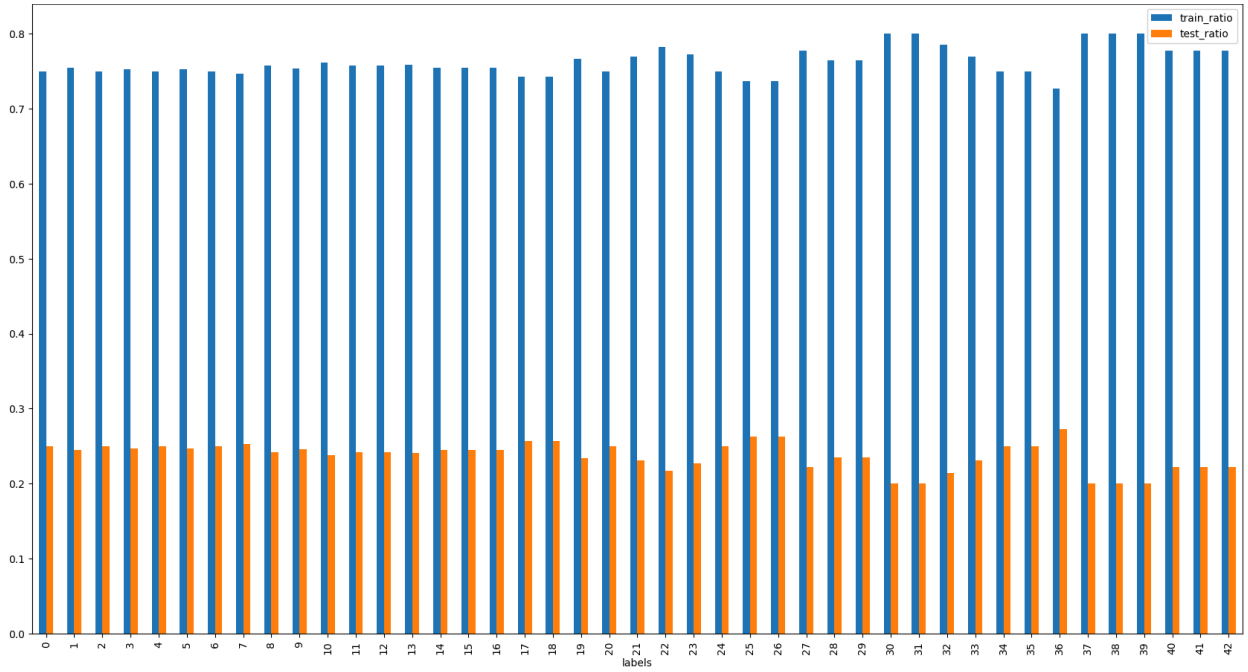
Results



Class distribution of training dataset



Class distribution of Testing dataset



Train test ratio for each class

As the ratio between training and testing dataset remains same , there is no need for data augmentation

```
TrafficSignNet(
  (conv1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(128, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=8192, out_features=4096, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=2048, bias=True)
    (5): ReLU()
    (6): Dropout(p=0.5, inplace=False)
    (7): Linear(in_features=2048, out_features=43, bias=True)
  )
)
```

Structure of the model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           9,472
       BatchNorm2d-2          [-1, 64, 32, 32]             128
              ReLU-3          [-1, 64, 32, 32]               0
         MaxPool2d-4          [-1, 64, 16, 16]               0
            Conv2d-5         [-1, 128, 16, 16]         204,928
       BatchNorm2d-6         [-1, 128, 16, 16]             256
              ReLU-7         [-1, 128, 16, 16]               0
            Conv2d-8         [-1, 128, 16, 16]         409,728
       BatchNorm2d-9         [-1, 128, 16, 16]             256
             ReLU-10         [-1, 128, 16, 16]               0
        MaxPool2d-11           [-1, 128, 8, 8]               0
           Conv2d-12           [-1, 256, 8, 8]         295,168
      BatchNorm2d-13           [-1, 256, 8, 8]             512
             ReLU-14           [-1, 256, 8, 8]               0
           Conv2d-15           [-1, 512, 8, 8]       1,180,160
      BatchNorm2d-16           [-1, 512, 8, 8]           1,024
             ReLU-17           [-1, 512, 8, 8]               0
        MaxPool2d-18           [-1, 512, 4, 4]               0
          Flatten-19                [-1, 8192]               0
           Linear-20                [-1, 4096]      33,558,528
             ReLU-21                [-1, 4096]               0
          Dropout-22                [-1, 4096]               0
           Linear-23                [-1, 2048]       8,390,656
             ReLU-24                [-1, 2048]               0
          Dropout-25                [-1, 2048]               0
           Linear-26                  [-1, 43]          88,107
================================================================
Total params: 44,138,923
Trainable params: 44,138,923
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 4.58
Params size (MB): 168.38
Estimated Total Size (MB): 172.97
----------------------------------------------------------------
```

Summary of the Pytorch model

```
Number of parameters: 44138923
```

Number of parameters in the model

Accuracy : 94 %

Real time working of the model