Traffic Signs Recognition

By Adithya Kolla

Problem Statement

Given a various traffic signs along with images dataset , we are challenged to classify traffic signs present in the image into differentcategories.

Motivation and Introduction

In the world of **Artificial Intelligence** and advancement in technologies, many researchers and big companies are working on autonomous vehicles and self-driving cars.

So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly.

Citations and Links

Research Paper that I duplicated:

D. R. Bruno and F. S. Osorio, "Image classification system based on deep learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes," 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017, pp. 1-6, doi: 10.1109/SBR-LARS-R.2017.8215287.
Link: https://ieeexplore.ieee.org/document/8215287

Existing Approaches and methods I duplicated

TEAM	METHOD	TOTAL	SUBSET
[156] DeepKnowledge Seville	CNN with 3 Spatial Transformers	99.71%	99.71%
[3] IDSIA ⭐	Committee of CNNs	99.46%	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%
[1] INI-RTCV ⭐	Human Performance	98.84%	98.84%
[4] sermanet ⭐	Multi-Scale CNNs	98.31%	98.31%
[2] CAOR ⭐	Random Forests	96.14%	96.14%

Dataset

Contains different traffic signs.
classified into 43 different classes.

size

50,000 images

Source

From German INI
Benchmark Website

Tools

Data manipulation and Preprocessing ML libraries:

NumPy

scikit learn

Visualization libraries:

matplotlib

Deep learning:

workflow

01

Data Loading

02

Preprocessing

03

Baseline Model

04

Data Augmentation

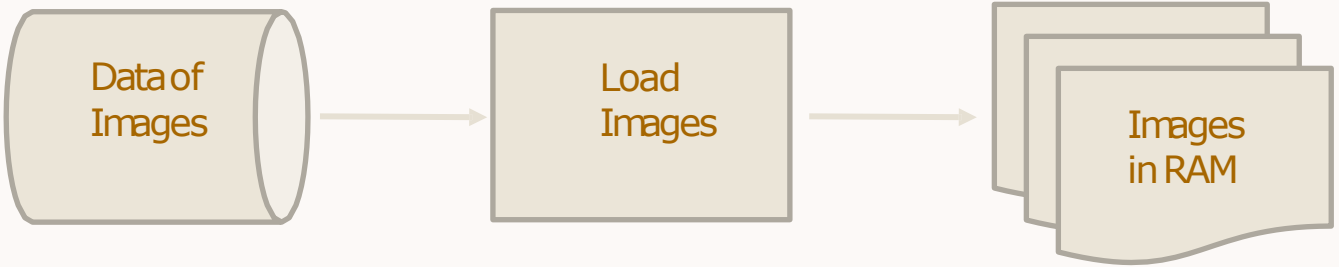
05

Build Deep Learning Model

06

Future work

01 Data Loading



02

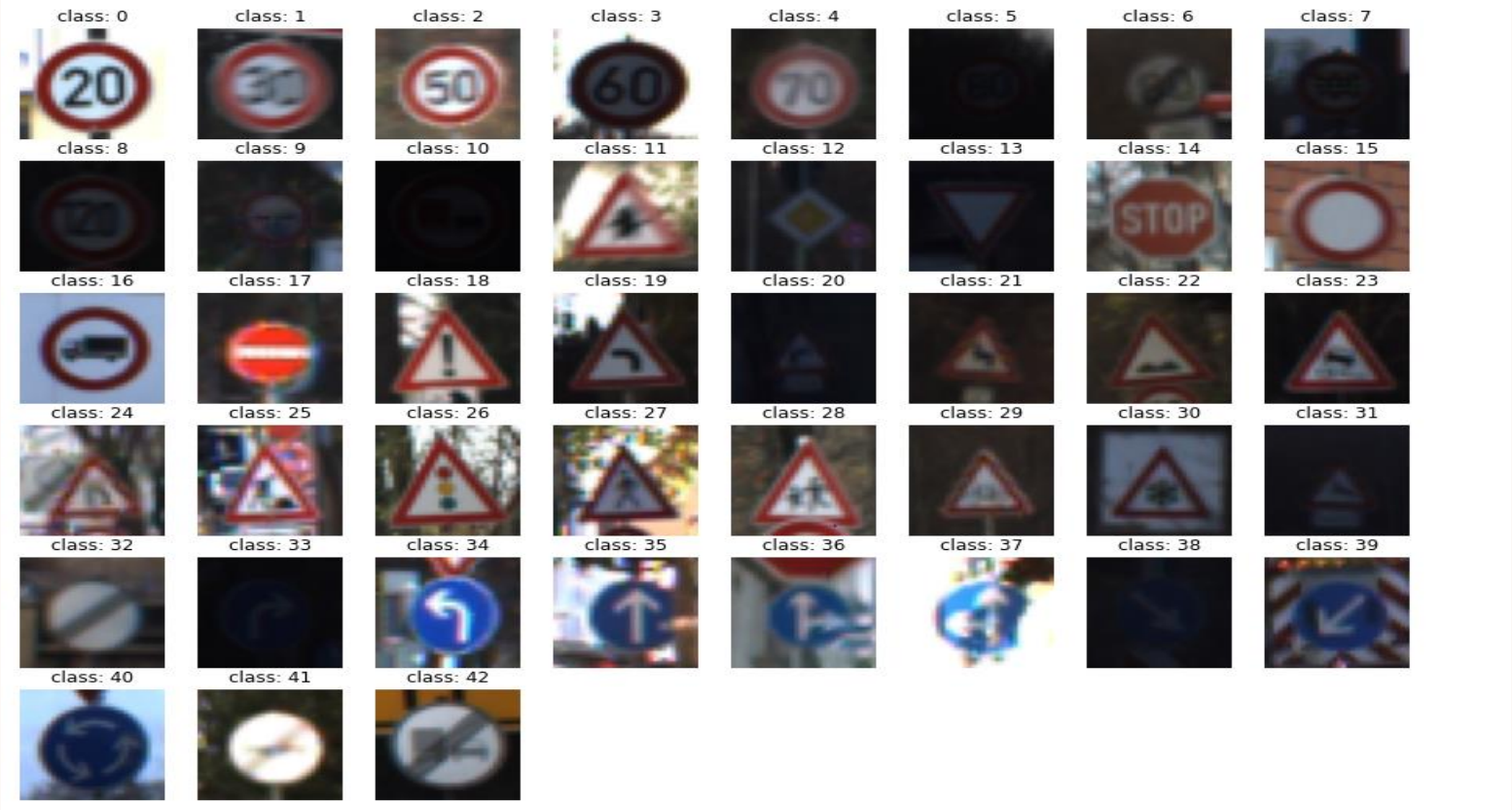
Image Preprocessing

- Set height and width to be 32x32
- Normalization, Divideby 255

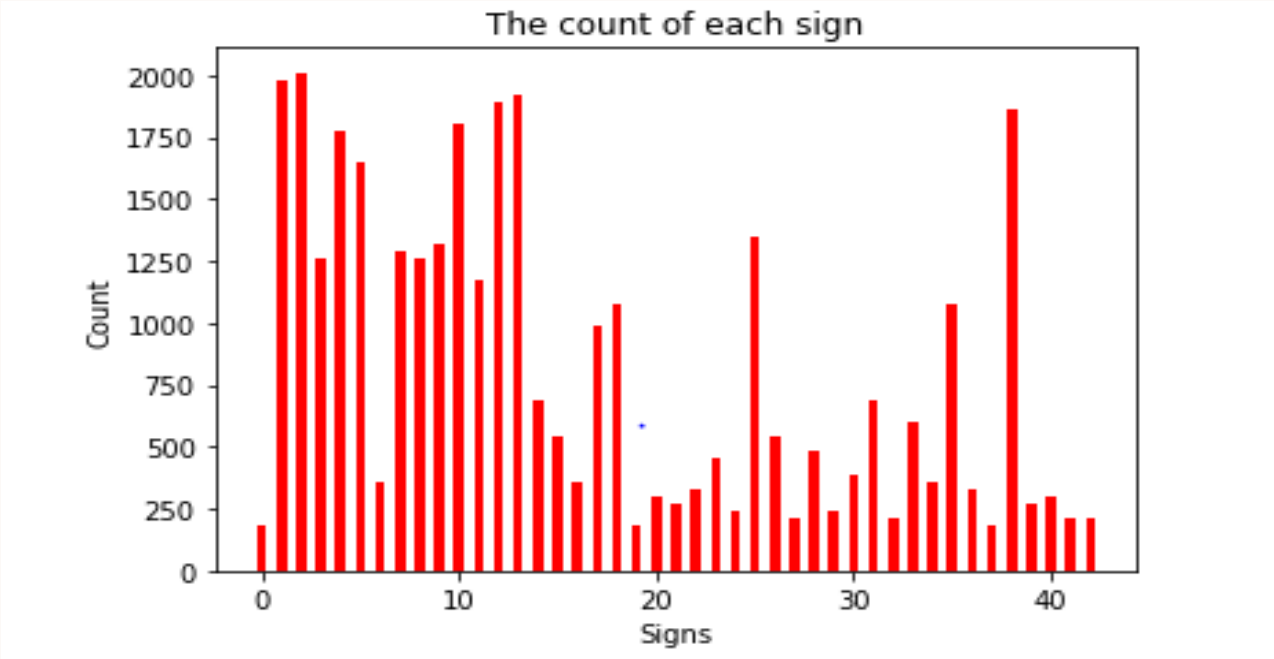
Model Preprocessing

- Splitting the Data
- Sets the labels and check the distribution.
- Labels Encoded

Sample Images from each Label



Class Imbalance Struggle



Convolution Neural Network

03

Baseline Model

Hyper parameter

Hidden layer :
- Two ConvolutionLayers
- Three simple NN layers

Kernel size: 5

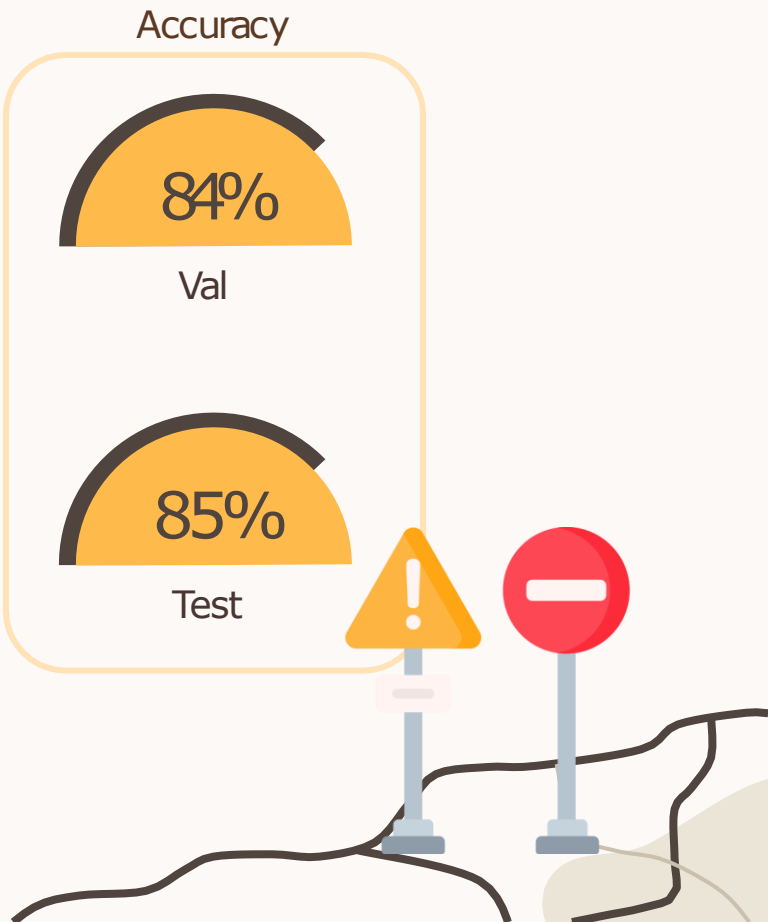
Activation Function:
- Relu
- Softmax

Optimizer:
- SGD
- Learning Rate: 0.001

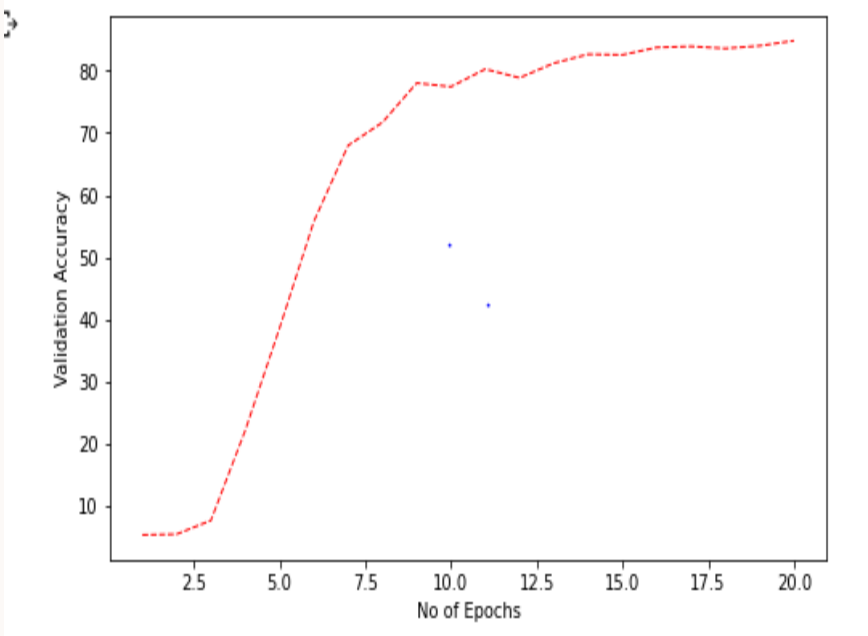
- Epochs: 20

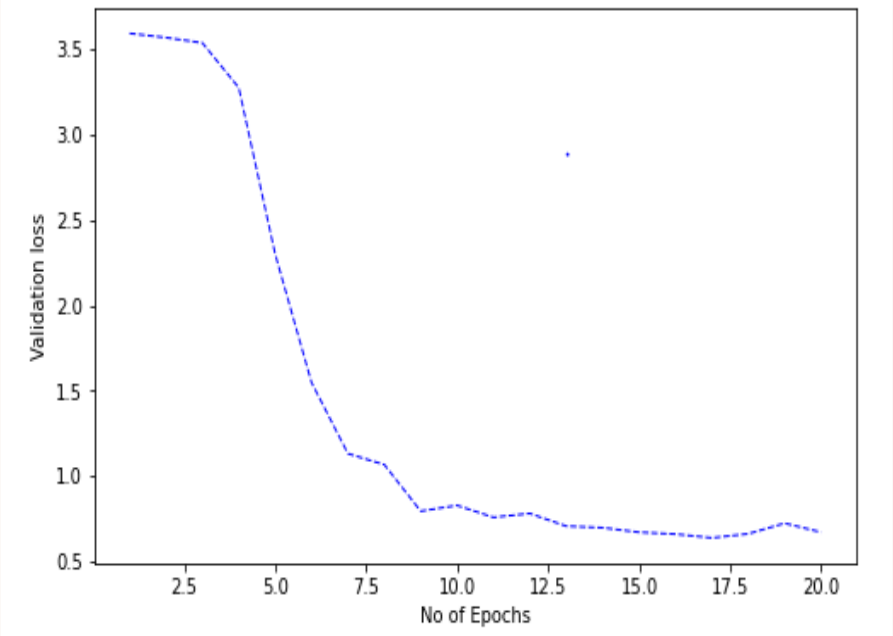
- Class Weight:
- Balanced

Val Accuracy: 84.8
Test Accuracy: 85.0

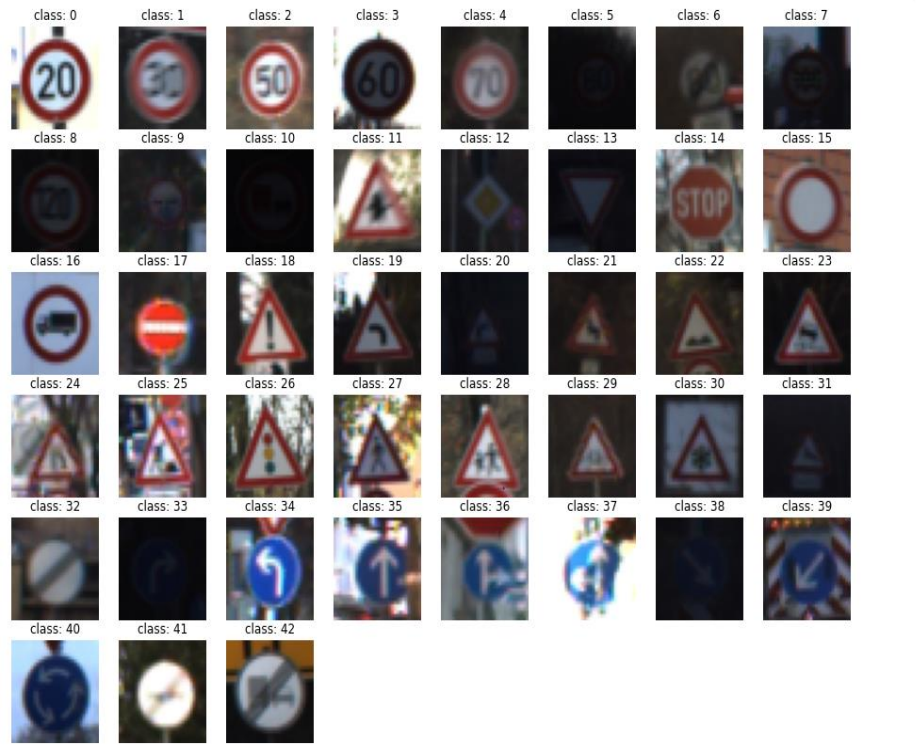


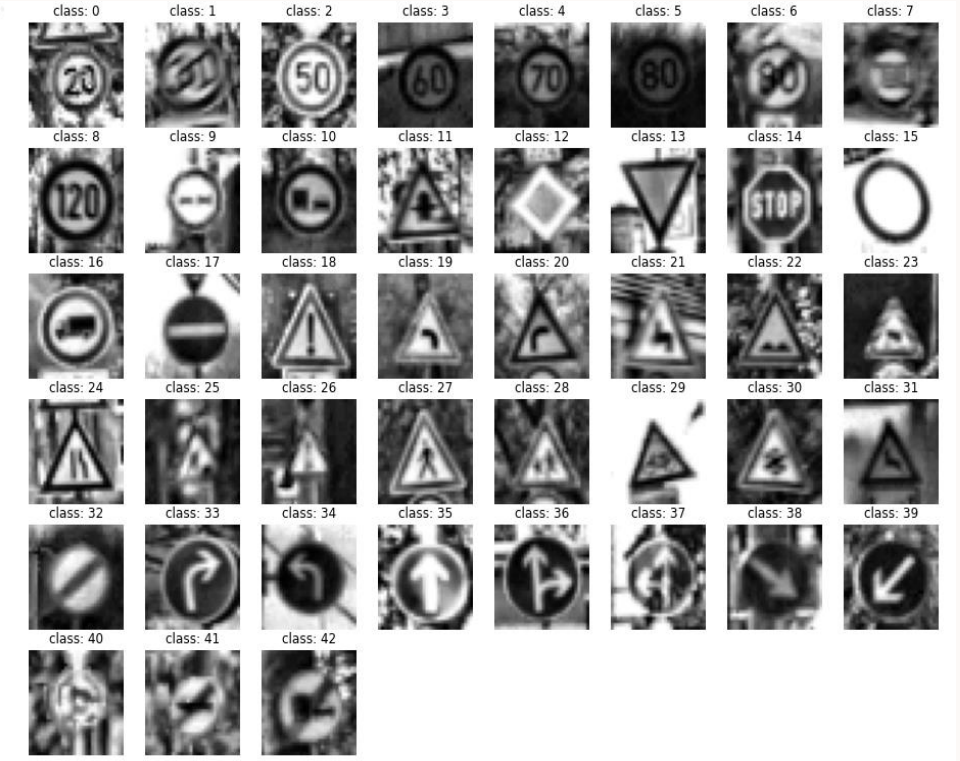
Validation Loss and Accuracy





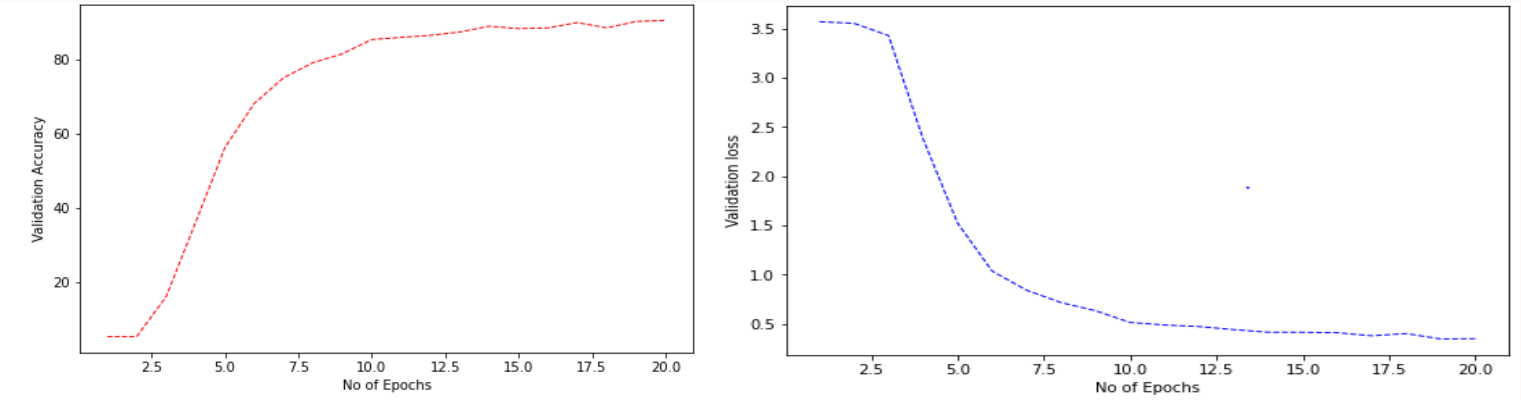
CLAHE and Gray Scale





Baseline Model(CLAHE and Gray Scale)

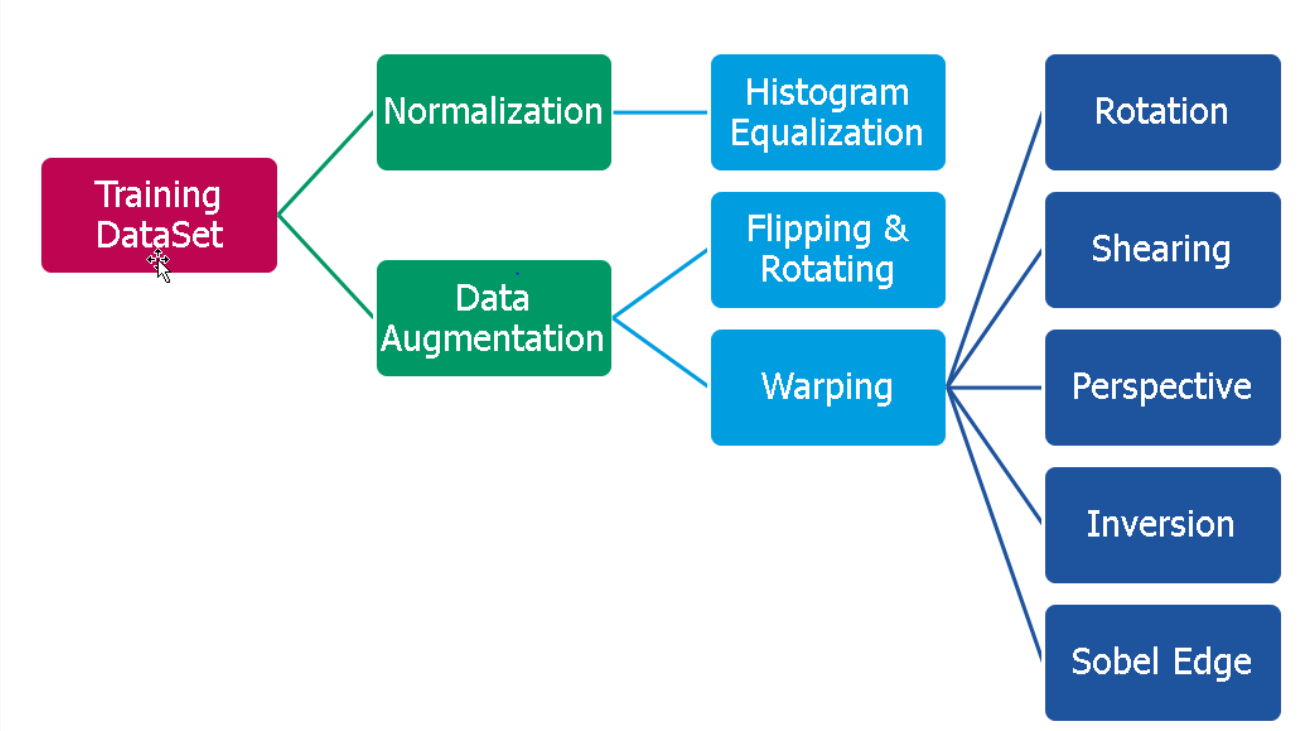
Val Accuracy: 90.658
Test Accuracy: 87.458



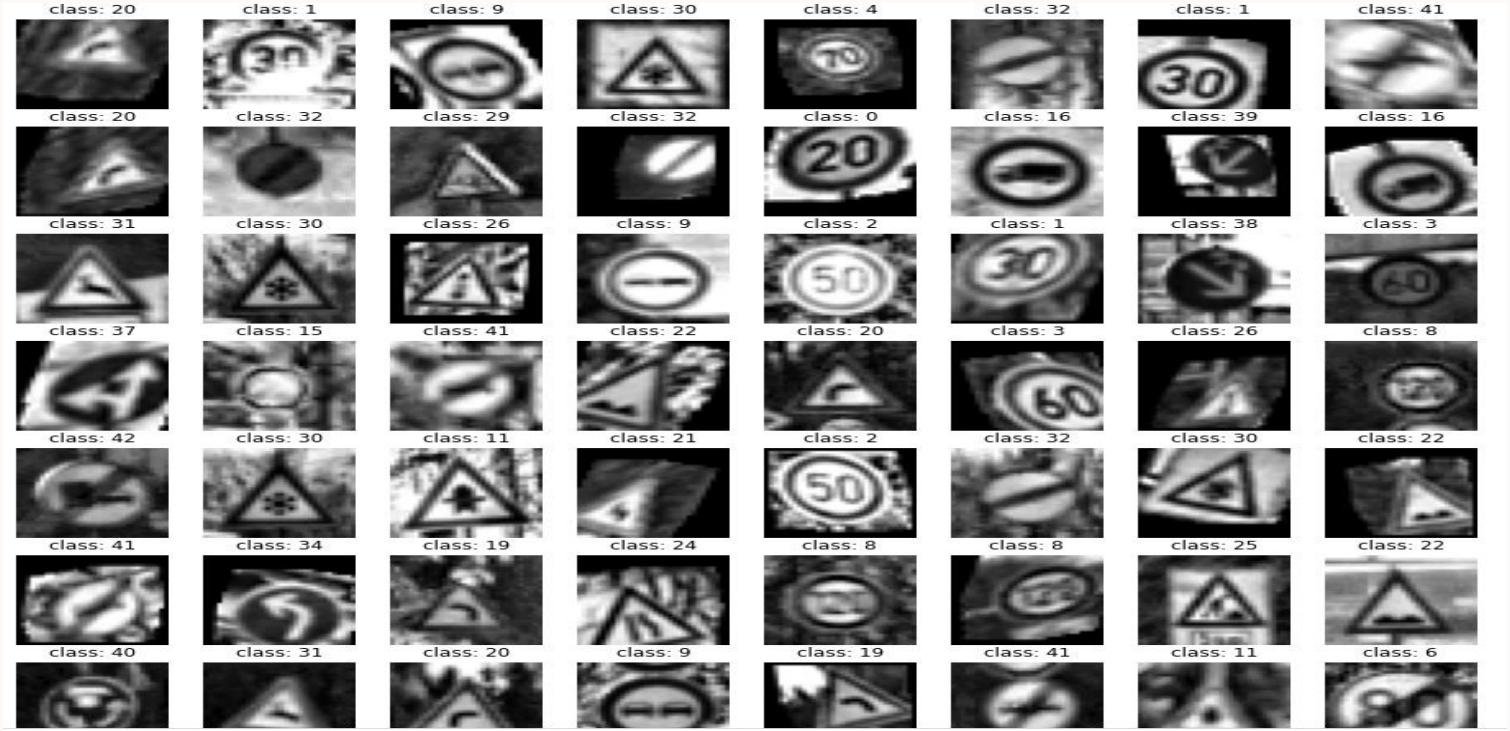
04

Data Augmentation

Data Augmentation Techniques

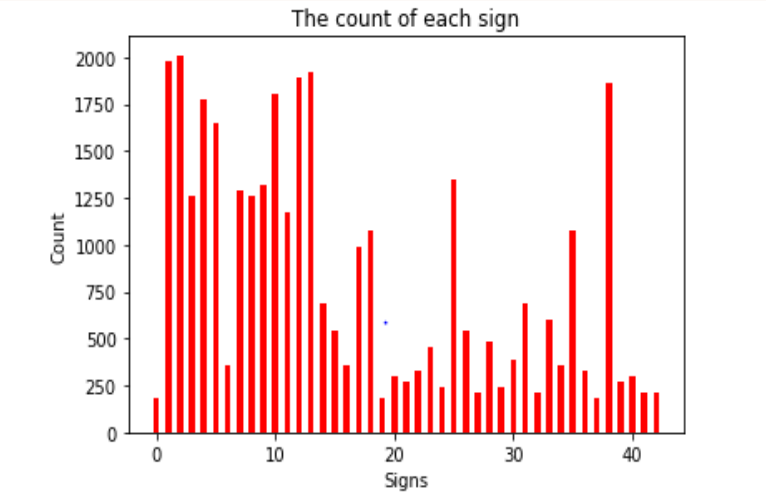


Sample Images after Data Augmentation

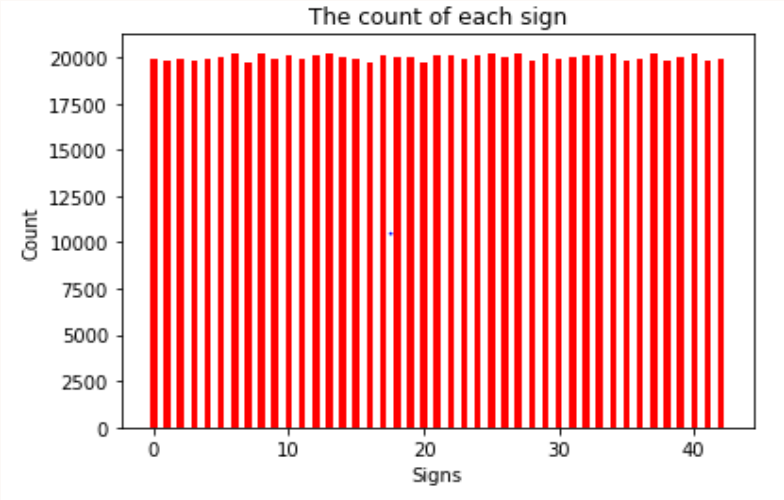


Comparison of Image count

Before Data Augmentation

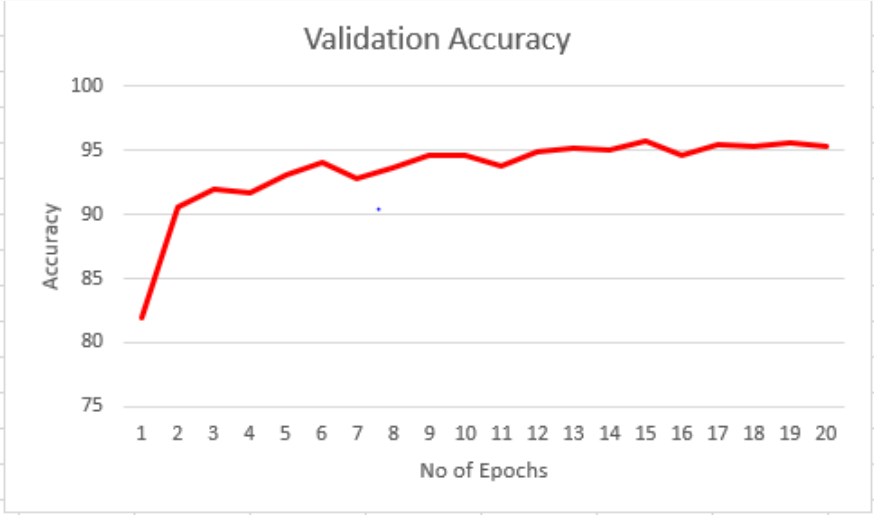


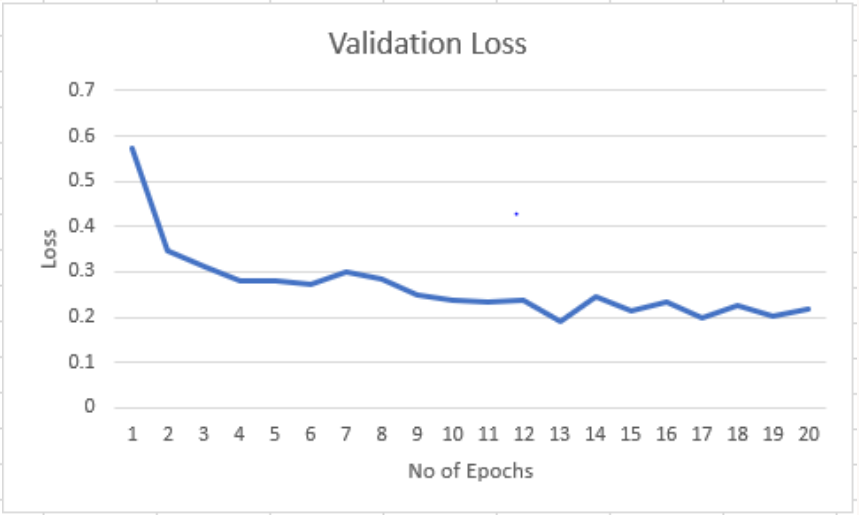
After Data Augmentation



Baseline Model results(After Data Augmentation)

Val Accuracy: 95.578
Test Accuracy: 93.793

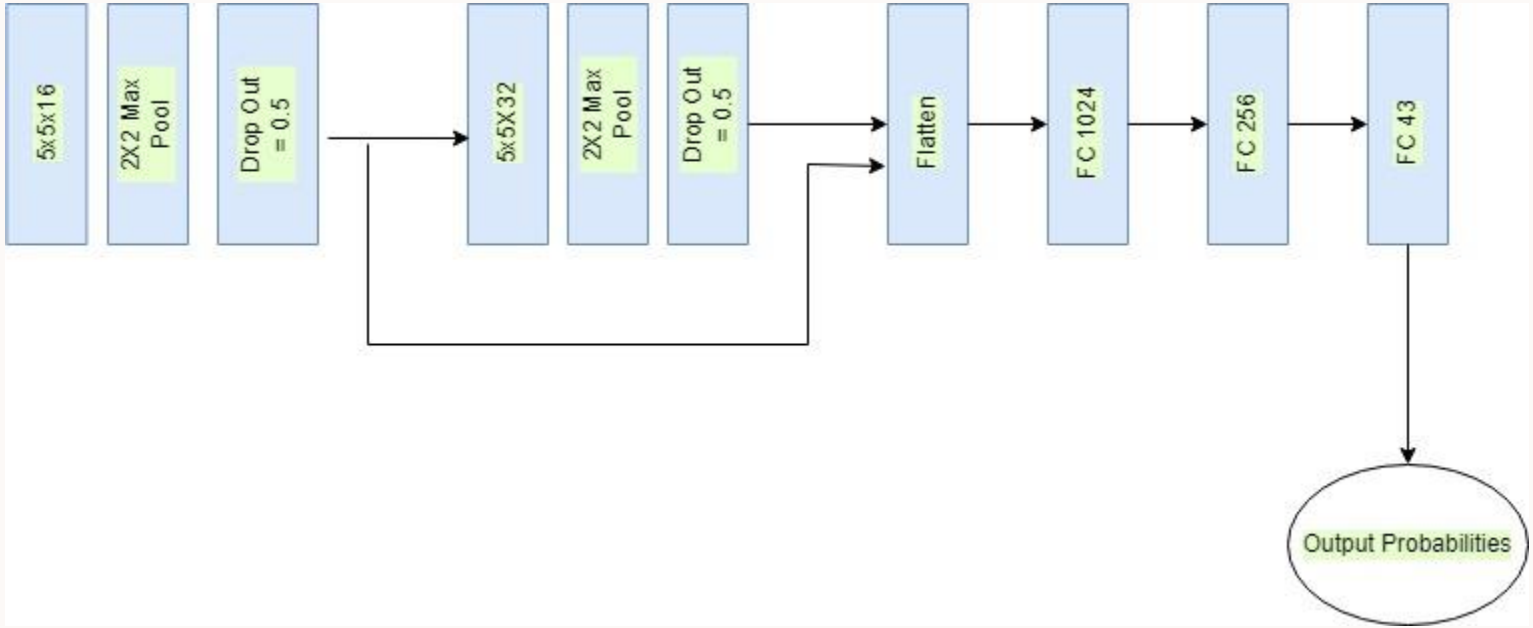




05

Final Deep Learning Model

Model Architecture



Snippet of CNN model

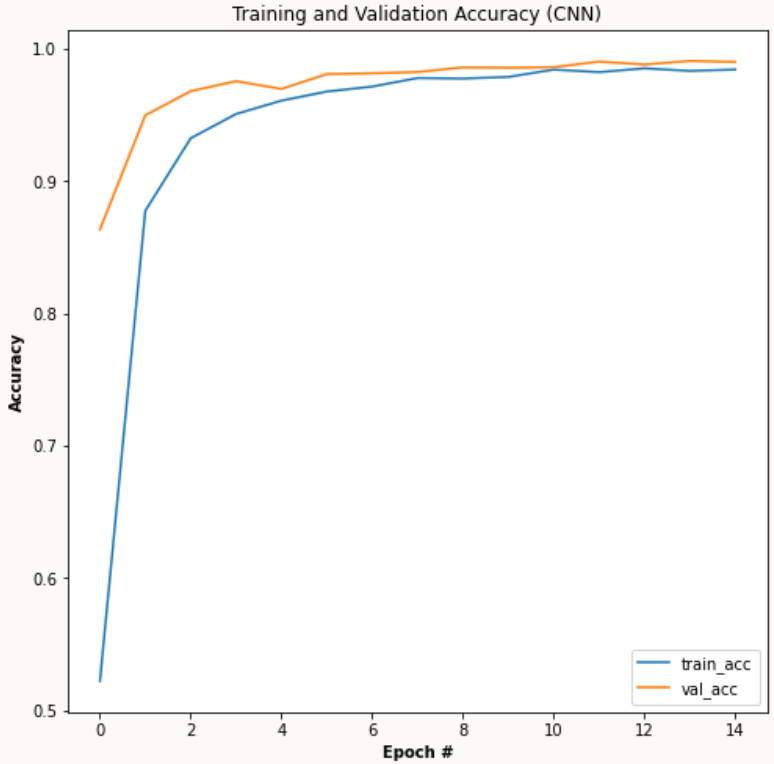
```
class TrafficSignNet(nn.Module):
    def __init__(self):
        super(TrafficSignNet, self).__init__()
        # Constraints for layer 1
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=16, kernel_size=5, stride = 1, padding=2)
        self.batch1 = nn.BatchNorm2d(16)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2) #default stride is equivalent to the kernel_size
        # Constraints for layer 2
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32, kernel_size=5, stride = 1, padding=2)
        self.batch2 = nn.BatchNorm2d(32)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2)
        # Defining the Linear layer
        self.fc1 = nn.Linear(2048,1024)
        self.dropout = nn.Dropout(p=0.5)
        # Defining the Linear layer
        self.fc2 = nn.Linear(1024,256)
        self.dropout1 = nn.Dropout(p=0.5)
        # Defining the Linear layer
        self.fc3 = nn.Linear(256,43)
```

Convolutional Neural Networks

hyperparameter

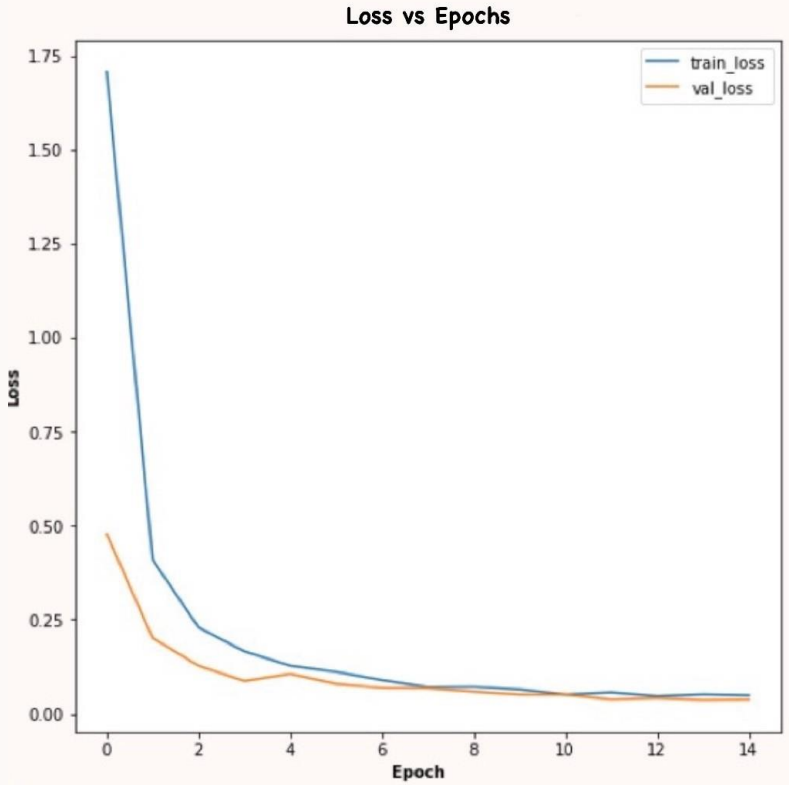
Hidden layer :
- Two Convolution Layers
- Three Fully connected NN layers

Kernel size: 5
Stride: 1
Padding: 2
Dropout: 0.5
Activation Function:
- Relu
-Softmax
Epochs: 20
Class Weight:
- Balanced



Val Accuracy: 97.785
Test Accuracy: 96.698

Convolutional Neural Networks



Loss: Categorical Crossentropy
Optimizer: Adam
Metrics: Accuracy

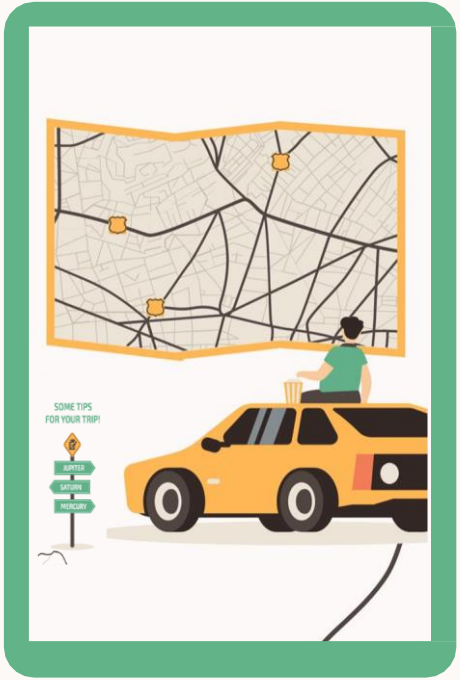
Predicting Some images from Test Dataset (Wrongly Classified)

Future Work

- Increase the accuracy of the model further to 99 to 100% percent with the help of Spatial transform networks which can yield high accuracy on test data

- Fine tuning with Hyper parameters of the model to increase the accuracy

- Deploy the model as a web application using Flask



Thank you..