Traffic Signs Recognition

Adithya Kolla

ITCS-5156 Fall 2021- Lee

February 27, 2022

Abstract

This report is presented as a survey of a previous work [IC17]

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.

This report presents a system for classifying images based on Deep Learning and applied in the recognition of traffic signals aiming to increase road safety increased road safety using autonomous and semi-autonomous intelligent robotic vehicles. This Traffic Signs recognition is a system created to automate vehicles, but also to help the human drivers to increase safety and the respect of traffic rules while driving the car. The system must be able to classify several different traffic signs (e.g. maximum speed allowed, stop, slow down, turn ahead, pedestrian), thus helping to make navigation within the local traffic rules.

1 Introduction

1.1 Problem Statement

Structured road scenarios and urban street environments require basic and proper traffic laws, as well as a variety of information for drivers, so that a vehicle, whether autonomous or semi-autonomous, or even driven by a person, may safely navigate on them. Traffic signs (speed control, stop and direction), traffic lights (warning, attention, proceed, traffic semaphore), and traffic lanes are the most commonly used signs (regulation of flows and space). Traffic signs are used to aid drivers in their driving tasks while also alerting them of local traffic restrictions.

These signs are of great importance not only for the driver of a semi-autonomous/intelligent vehicle but also for a vehicle that travels autonomously and shares the same rules with other cars and humans. In this work, a Deep Learning based image classification method was applied for the recognition of traffic signs in benefit of road safety applied to intelligent robotic vehicles.

1.2 Motivation and Challenges

My motivation behind this is 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. So, I want to build a model that can take input of traffic signs and classify those images into different categories.

In real world, Challenges of Traffic Signs recognition are:

- Weather conditions: if the captured image is influenced by raining, snow or fog.
- Lightning conditions: if there are differences in acquiring the images by daylight and night. The shades of colors of image can be seen differently in different conditions.
- Traffic signs can be in different shapes and content also be different in different countries. The example is shown below.

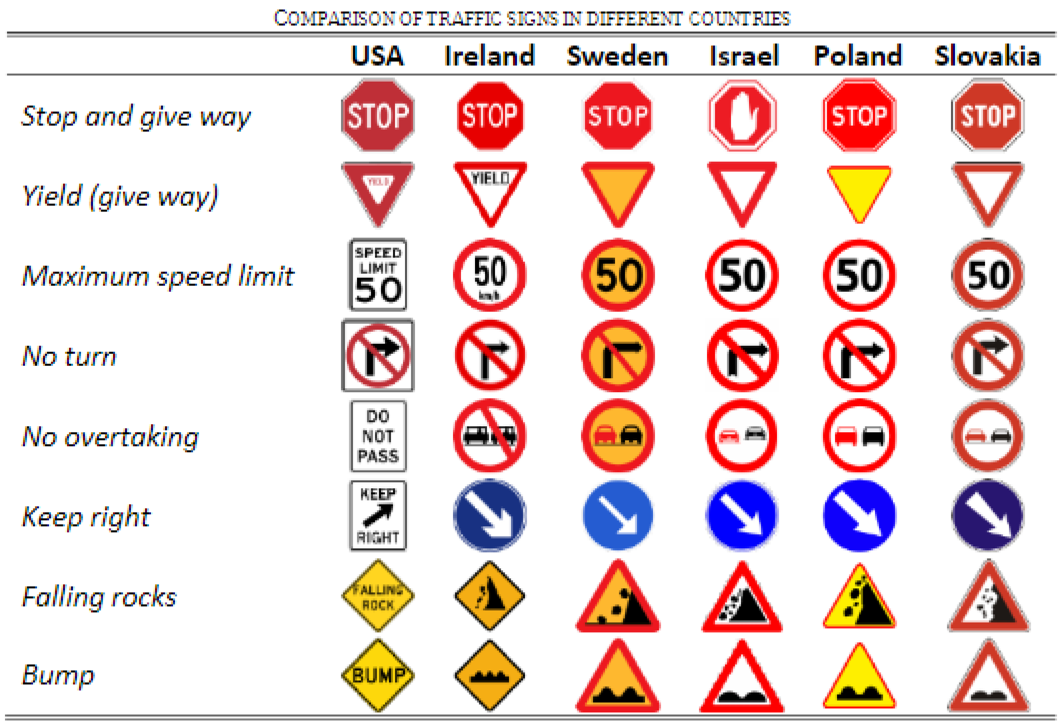

COMPARISON OF TRAFFIC SIGNS IN DIFFERENT COUNTRIES
USA
Ireland
Sweden
Israel
Poland
Slovakia
Stop and give way
Yield (give way)
Maximum speed limit
No turn
No overtaking
Keep right
Falling rocks
Bump
STOP
YIELD
SPEED LIMIT 50
50
DO NOT PASS
KEEP RIGHT
FALLING ROCK
BUMP

Figure1: Showing traffic images in different countries

1.3 Concise Summary

Summary of my approach is to built a Deep Convolution Neural Networks in order to classify the images. The images are passed to a Deep Learning model, adopting a DCNN Net (Deep Convolutional Neural Network/ConvNet) implemented using PyTorch. For processing of image data, one has to use Deep learning model and PyTorch is widely used today for image recognition applications

2 Related Works:

2.1 Convolutional Neural Networks

In 2017, at the Latin American Robotics Symposium D. R. Bruno and F. S. Osorio presented their work on image classification based on deep learning applied to traffic signs recognition [IC17].

D.R Bruno and his associates proposed a Deep Convolutional Neural Network to classify the images. In the Research paper [IC17] they did not mention the no of hidden layers or the architecture of the model they used. The no of hidden layers in the model is based on the data and type of problem to increase the accuracy of the model.

Here is a brief summary on the convolutional neural network.

Classification with artificial neural networks is a very popular approach to solve pattern recognition problems. A neural network is a mathematical model based on connected via each other neural units – artificial neurons – similarly to biological neural networks. Typically, neurons are organized in layers, and the connections are established between neurons from only adjacent layers. The input low-level feature vector is put into first layer and, moving from layer to layer, is transformed to the high-level features vector. The output layer neurons amount is equal to the number of classifying classes. Thus, the output vector is the vector of probabilities showing the possibility that the input vector belongs to a corresponding class.

An artificial neuron implements the weighted adder, which output is described as follows:

$$a_j^i = \sigma\left(\sum_k a_k^{i-1} w^i j_k\right),$$

where a^i_j is the j th neuron in the i th layer, w^{ij}_k stands for weight of a synapse, which connects the j th neuron in the i th layer with the k th neuron in the layer i-1. Widely used in regression, the logistic function is applied as an activation function. It is worth noting that the single artificial neuron performs the logistic regression function. The training process is to minimize the cost function with minimization methods based on the gradient decent also known as backpropagation. In classification problems, the most commonly used cost function is the cross entropy:

$$H(p,q) = -\sum_i Y(i)\log y(i).$$

Training networks with large number of layers, also called deep networks, with sigmoid activation is difficult due to vanishing gradient problem. To overcome this problem, the ELU function is used as an activation function.

$$ELU(x) = \begin{cases} \exp(x) - 1, x \le 0 \\ x, x > 0 \end{cases}.$$

Today, classifying with convolutional neural networks is the state of the art pattern recognition method in computer vision. Unlike traditional neural networks, which works with one-dimensional feature vectors, a convolutional neural network takes a two-dimensional image and consequentially processes it with convolutional layers. The number of layers can be optimized according to the problem in question.

Each convolutional layer consists of a set of trainable filters and computes dot productions between these filters and layer input to obtain an activation map. These filters are also known as kernels and allow detecting the same features in different locations. For example, Fig. 2 shows the result of applying convolution to an image with 4 kernels.

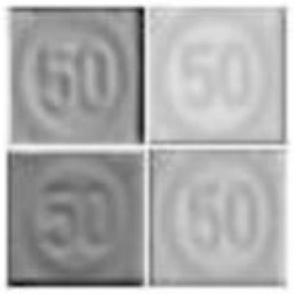


Fig. 2. Input image convolution.

2.2 A Method for Recognizing Traffic Sign Based on HOG-SVM

In 2021, International Conference on Computer Engineering and Application (ICCEA) Y. Liu, W. Zhong and his associates proposed a model [THS21] to classify traffic images using HOG-SVM. This method involves three steps

1) Image preprocessing

In this part, we mainly complete the traffic sign image size adjustment, color image graying, gray image Gamma correction and HOG feature extraction. Gamma correction is mainly used to process the brightness of the image and weaken the influence of light and shadow on the image.

2) HOG feature extraction

Extraction of HOG features, with the corresponding spatial, cell, block and sliding step size.

3) Classification and recognition of traffic signs based on SVM

In this part, firstly, based on the training data set, support vector machine is used to generate the initial classifier of traffic signs. The initial classifier is constructed by 42 binary classifiers. Then the initial classification result is used to get the final classification result.

The model architecture is shown in figure 3 below

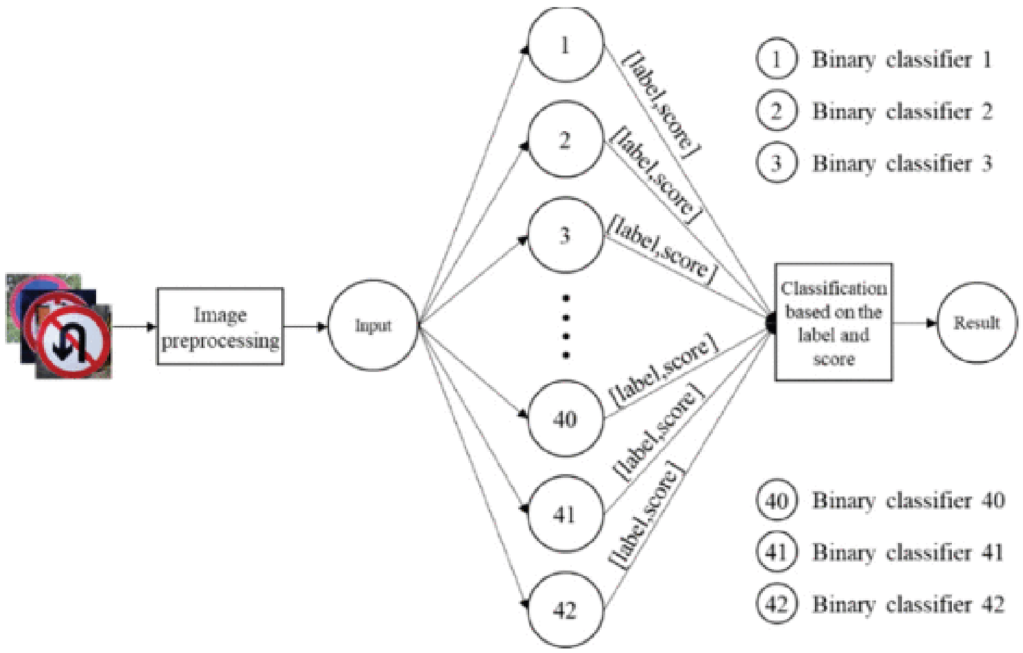


Figure 3: HOG-SVM based Traffic Sign recognition

2.3 Pros and Cons of the Approaches

One of the drawbacks of HOG is that its computation speed is tardy while detecting an object for large-scaled images as it uses a sliding window technique to extract features from every pixel of an image. Hence, the accuracy is not highly reliable compared to the current convolutional neural networks.

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. But the disadvantage is CNN do not encode the position and orientation of object. Lack of ability to be spatially invariant to the input data. Lots of training data is required.

If we compare results of Test Accuracy with Deep CNN outperformed HOG-CNN.

2.4 Relation to my Approach

Both these works demonstrate the idea of classifying the traffic sign images in different approaches, one research paper with the approach of using Deep CNN and the other with the HOG-SVM model. I replicated the approach [IC17] which is proposed by D. R. Bruno and F. S. Osorio, using a Deep Convolutional Neural Network to recognize traffic signs because the results are comparatively higher as compared to the other approach using HOG-SVM proposed by Y. Liu, W. Zhong.

3 Used Methods

The implementation of this work was found and slightly adapted from an article on Medium implemented by Barney Kim specifically the Traffic Signs Classification with CNN [TSB19]. I have used data augmentation additionally to extend the images for each class in the dataset which is not completely available in this article [TSB19]. Here is the link to my Project Repository

3.1 Data Loading and Preprocessing

Dataset Contains different traffic signs which are classified into 43 different classes. Total size of the dataset is around 50,000 images. The dataset is split into training images of 34799, validation images of 4410 and testing images of 12630.

The Figure4 below shows the distribution of no of images present in each class of the dataset.

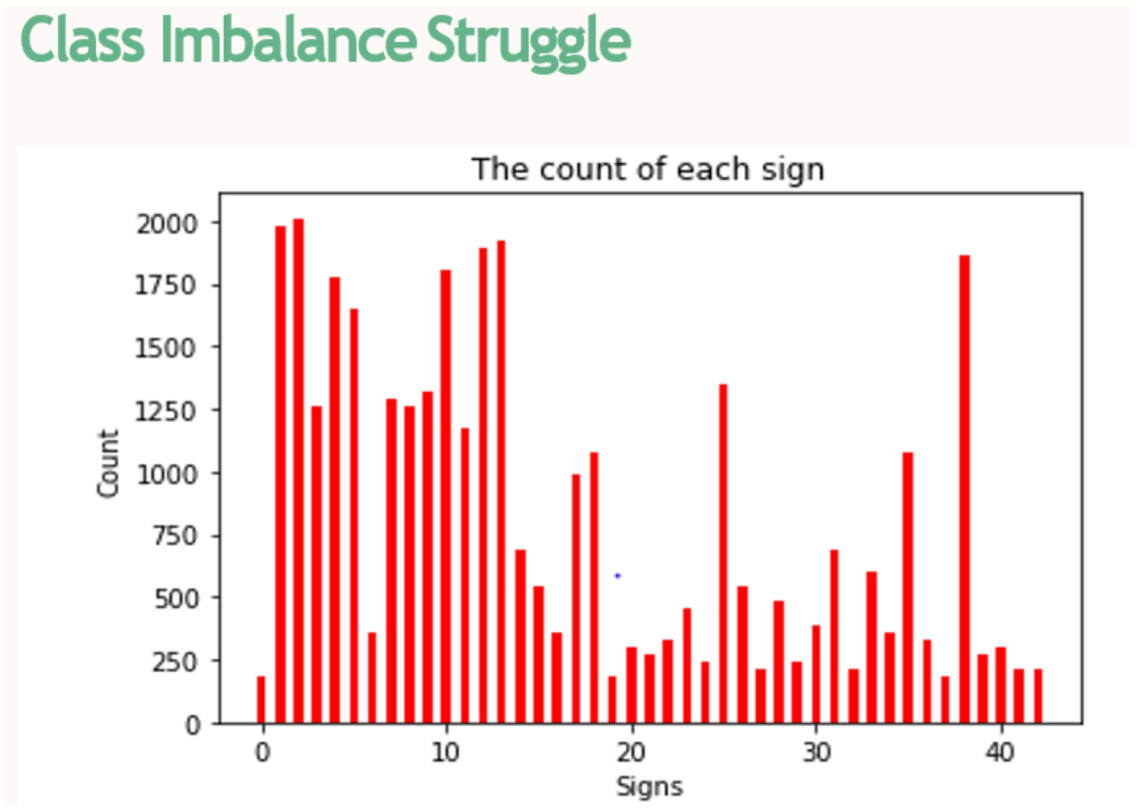
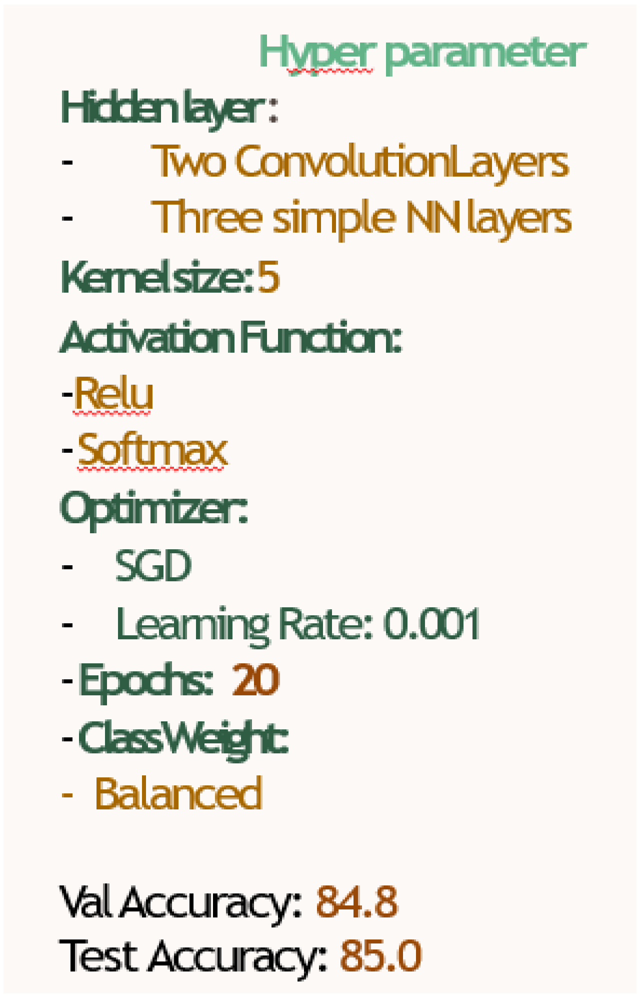


Figure 4: Count of images in each class

3.2 Construction of Baseline Model



Hyper parameter

Hidden layer :

- Two ConvolutionLayers
- Three simple NN layers

Kernelsize:5

Activation Function:

-Relu

-Softmax

Optimizer:

- SGD
- Learning Rate: 0.001

-Epochs: 20

-ClassWeight:

- Balanced

Val Accuracy: 84.8

Test Accuracy: 85.0

Figure 5: Baseline Model Hyper Parameters

After that I constructed a Baseline Model in Figure 5, which has two convolutional layers and three simple Neural Network Layers. The Kernel size is 5x5 and the activation function is Relu and the optimizer used is Stochastic Gradient Descent with a learning rate of 0.001 and the no of epochs or Iterations I performed is 20.

With the Baseline model a got a Validation Accuracy of 84.8% and Test Accuracy of 85%. In Figure 6, graph shows the comparison between no of epochs and validation accuracy and validation loss.

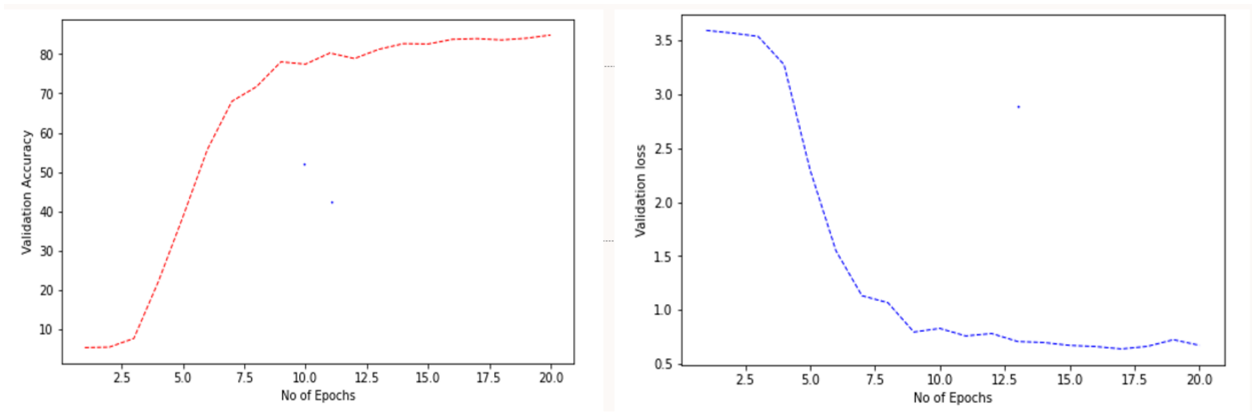


Figure 6: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

From the figure 6 we observe that as the no of epochs increases Validation Accuracy increases and Validation Loss decreases.

3.3 Gray Scale and Contrast limited adaptive histogram equalization (CLAHE)

After seeing the results of the Baseline model, I converted the images into gray scale and use CLAHE to improve the Contrast of images.

CLAHE: CLAHE is a variant of Adaptive histogram equalization (AHE) which takes care of over-amplification of the contrast. CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighboring tiles are then combined using bilinear interpolation to remove the artificial boundaries. This algorithm can be applied to improve the contrast of images.

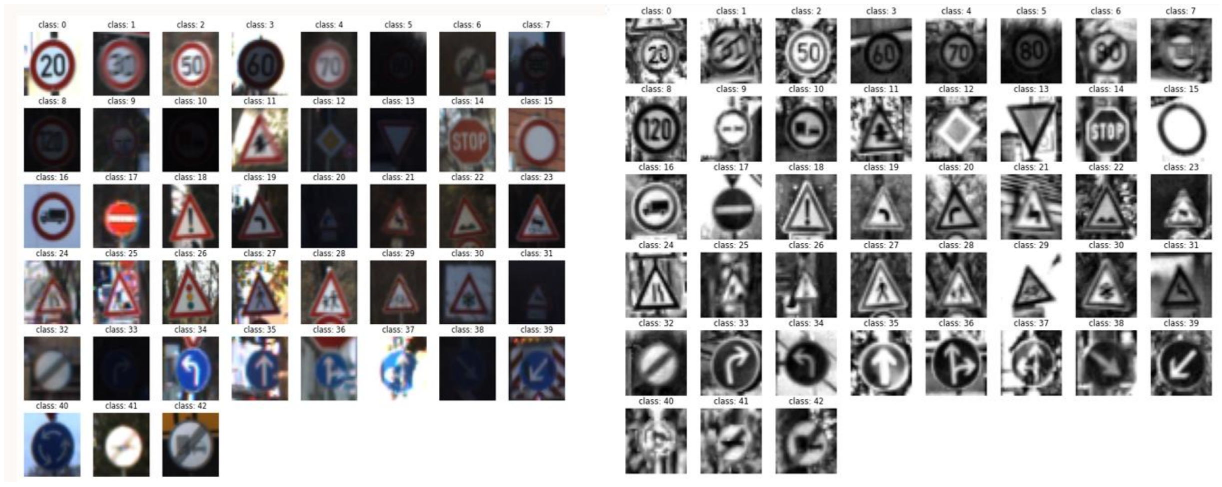


Figure 7: Original Images vs Applying Gray Scale and CLAHE

Again, I fit the training data after performing gray scale and CLAHE on the Baseline model to compare the accuracies. I got Validation Accuracy of 90.658 and Test Accuracy of 87.458.

In Figure 8, graph shows the comparison between no of epochs and validation accuracy and validation loss for Baseline model with images converted into gray scale and CLAHE.

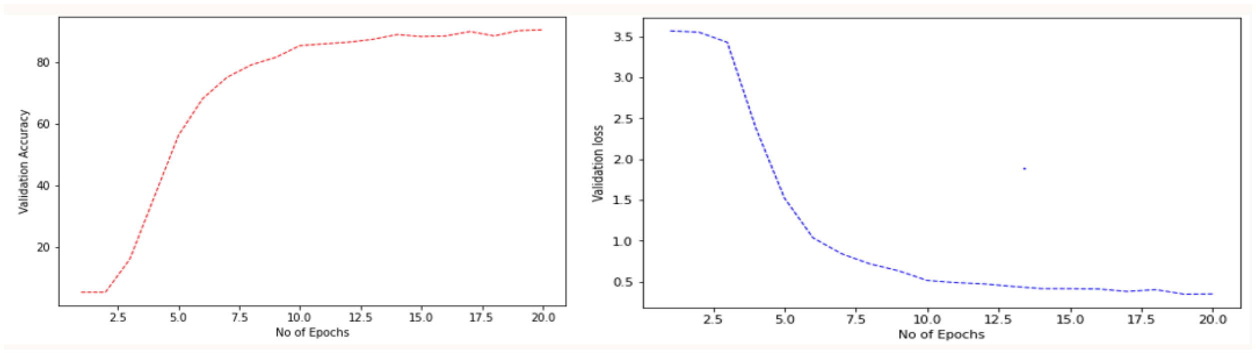


Figure 8: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

3.4 Data Augmentation

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. In Figure 9, Some of the data augmentation techniques are shown below.

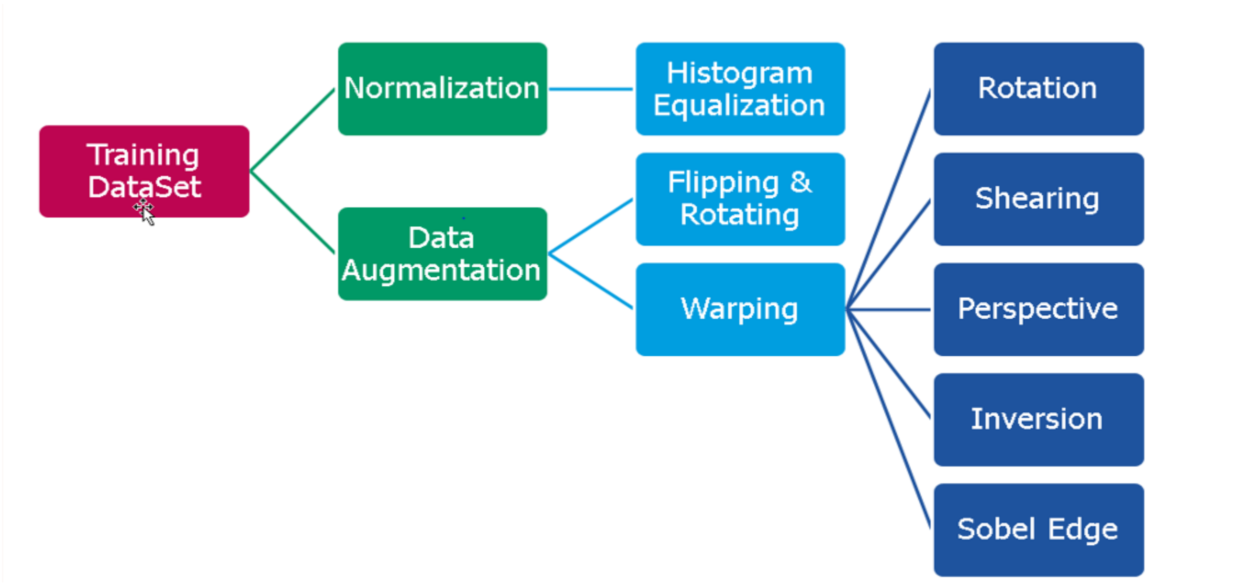


Figure 9: Data Augmentation Techniques

I performed Rotation, Shearing, flipping for the given dataset in order to generate more images for each label and overcome the class imbalance. It also helps in increasing the no of Training samples for the model which in turn increases the accuracy of the model.

In Figure 10, After performing Data Augmentation here are the sample images shown below

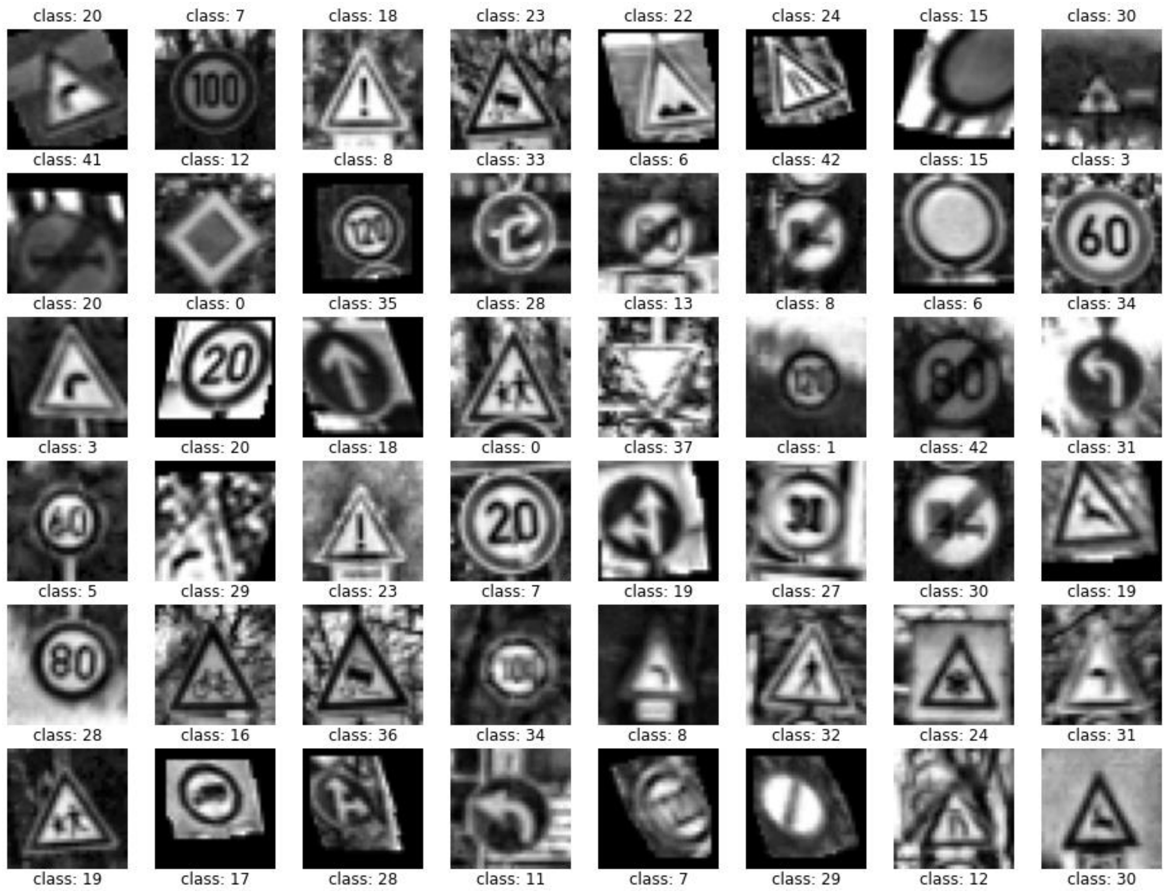


Figure 10: Sample images after Data Augmentation

With the help of data generated from Data Augmentation I again trained the Baseline model and performed evaluation on test set, I got a Validation Accuracy of **95.578** and Test Accuracy of **93.793.** Validation accuracy and Test Accuracy increased after data augmentation compared to previous results.

In figure 11 below contains the graph showing comparison between no of epochs and validation accuracy and validation loss for Baseline model with images converted into gray scale and CLAHE and also implementation of data augmentation.

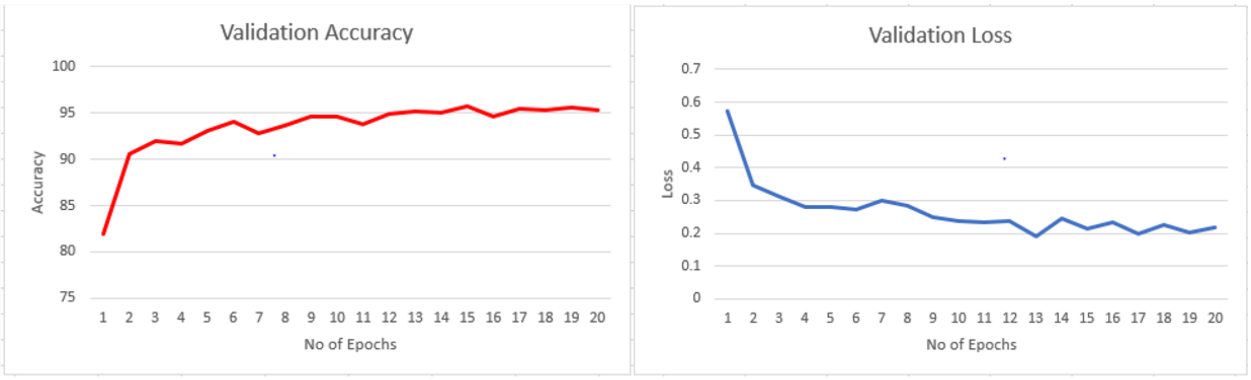


Figure 11: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

3.5 Final Model of Deep Convolutional Neural Network

Model Architecture:

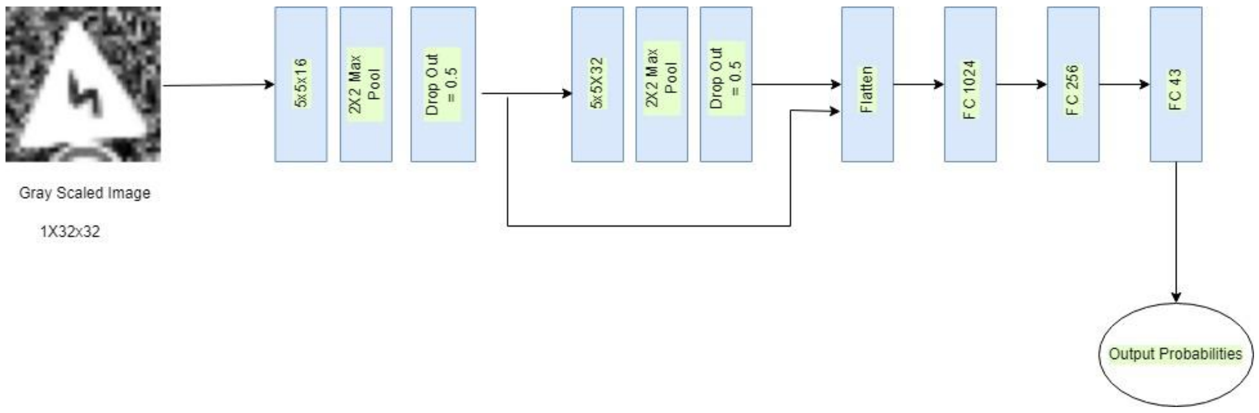


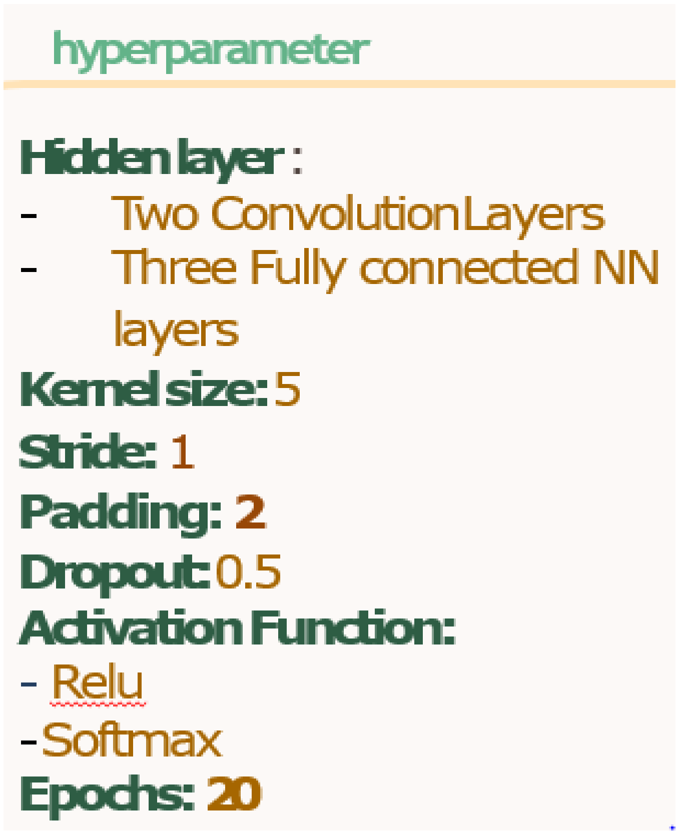
Figure 12: Model Architecture

Figure 12 represents the model architecture of final Deep CNN; Model takes input of the image which is in gray scale of size 1x32x32, contains two convolutional layers, Max pooling of size 2x2 in each convolutional layer and three fully connected layers. Final layer consists of output size of 43 which matches with the no of classes in the dataset.

Snippet for summary of the model:

```
TrafficSignNet(
  (conv1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (batch1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (batch2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=2048, out_features=1024, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=1024, out_features=256, bias=True)
  (dropout1): Dropout(p=0.5, inplace=False)
  (fc3): Linear(in_features=256, out_features=43, bias=True)
)
```

Hyper Parameters of Final Model:



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

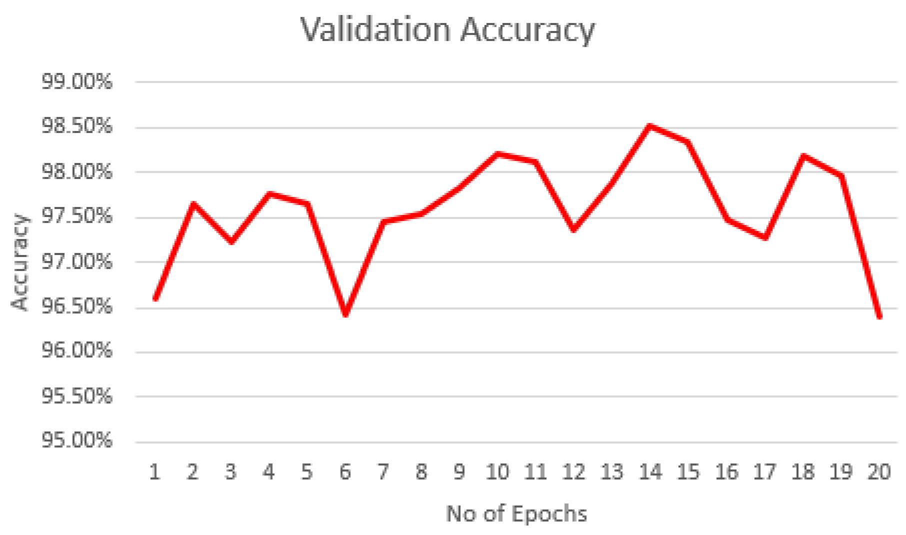
Figure 13: hyperparameters of the final model

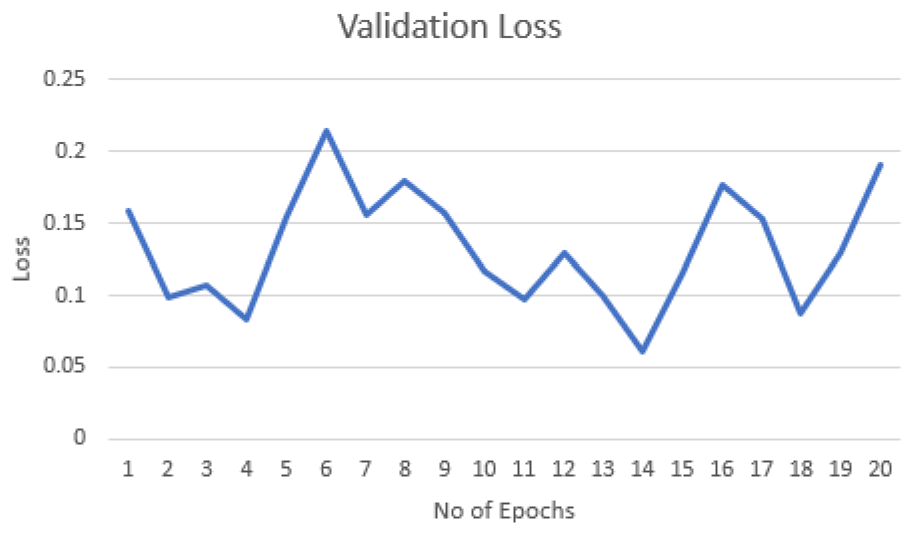
Here the kernel size is 5x5 and stride is 1, padding is 2, drop out is 0.5, activation function is Relu and 20 Epochs for training model. For Baseline model I used SGD as optimizer but for the final model I used Adam optimizer. Learning rate is 0.001.

Results of Final Model:

The Final model has a **Validation Accuracy is 97.785** and **Test Accuracy is 96.698.**

Graphs:





4. Experiments:

After a lot of research and reading different articles in medium and related works on Data Augmentation and Construction of CNN and implementation using PyTorch I am able to construct the final model.

Initially I Constructed a Baseline Model and trained the model using train dataset. But the Test accuracy is pretty low which is around 85% which is very low as compared to accuracy of the model presented by D. R. Bruno and F. S. Osorio [IC17] having an accuracy of 97.85%.

After that I converted all the images into gray scale and used CLAHE to adjust the contrast in the images which helped in the slight improvement of the Test accuracy to 87.45% with baseline model.

I completed the data augmentation using Shearing, Rotation, flipping which helps in creation of extended dataset from the existing images and has more data to train to increase the accuracy of the model. After data augmentation test accuracy is improved to 93.793% which is comparatively better than previous accuracy. It took nearly 2 hour 30 minutes to train the model on GPU using Google Collab.

After lot of tinkering with the no of convolutional layers and contemplating with no of hidden layers that are fully connected, and with different values of learning rates, Max pool size, stride and padding I ended up with a final Deep neural network already explained in Section 3.5. To Train the model it took nearly 3 hour 15 minutes on GPU using Google Collab. The model has a Test Accuracy of 96.698% which is slightly less than the accuracy of the model that is 97.85% presented by D. R. Bruno and F. S. Osorio.

I am able to reproduce the paper's experiment but the results are not identical. I got a test accuracy of 96.698% which is nearly 1% lesser than the Test accuracy of the paper presented by D. R. Bruno and F. S. Osorio.

Examples where the model is not able to classify

In Figure 14 here of the images where the model is not able to classify the correct one on the test dataset.

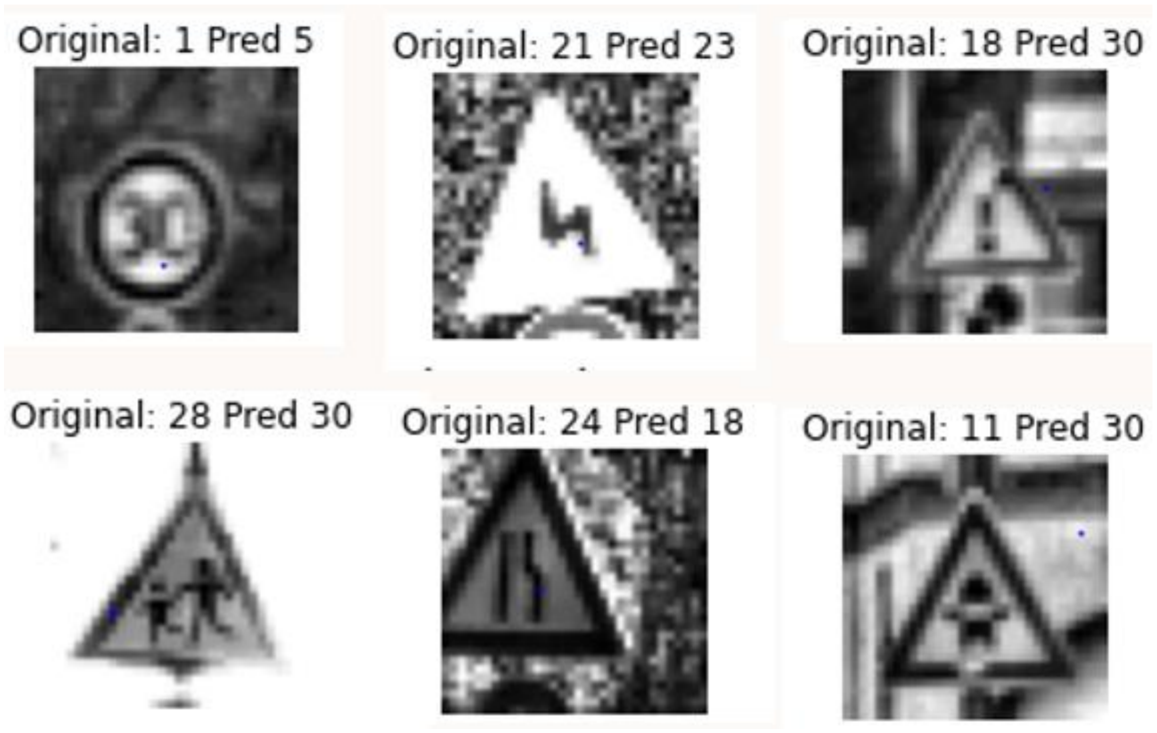


Figure 14: Example showing the wrongly classified images

From the figure above the image displayed is the original image and label as Original on top of it, but the model is classifying it to a different label.

My thoughts about the model are that the model has performed well with a Test accuracy of 96.698 but the accuracy of the model can still further improve with the trying different hyper parameters of the model such as number of convolution layers, kernel size, stride, padding or even the learning rate of the model. The accuracy can also be increased with the help of spatial transformer networks up to 99 or 100%.

5. Conclusion and Future Work

Because of the limited availability and also the time taken to train the Neural Network which taking nearly 3.50 hours to train a model on GPU using Google Collab for 20 epochs, I could test different hyper parameters such as number of convolution layers, kernel size, stride, padding or even the learning rate of the model to improve the test accuracy of the model.

I am new to these concepts of Deep Learning models, I have learnt a lot of concepts during the implementation of the project about Convolutional Neural networks such as usage of convolutions, Max pooling, Activation functions and the Cost functions of the Neural networks. I am happy that I can create a Convolutional Neural Network for image processing and yields some good results.

With the help of articles from medium and watching YouTube videos on Convolutional Neural Networks I am able to learn the concepts and overcome these challenges.

I would like to test the model on USA traffic images and see the performance of the model on new data of USA traffic signs. I also want to improve the accuracy of the model using Spatial Transformer networks which can improve the accuracy of the model.

6. My Contributions:

I referred to several articles on medium and Kaggle in order to complete the work. For Data loading, Image processing and Data augmentation I used this article from Medium which are already implemented by Barney Kim [TSB 19]. So, I used this existing work for Image Processing and Data Augmentation.

My contribution is construction of a baseline Model and training the baseline model at various stages after Conversion of image to gray scale and CLAHE, after Data Augmentation. I also constructed my Final model of Deep Convolutional Neural Network with 2 convolutional layers and three fully connected layers and training the model. I also visualized different plots for comparison of Number of Epochs vs Validation Accuracy and Number of Epochs vs Validation Loss, Construction of Confusion matrix, To Display the images where the model failed to correctly classify the given image. This article helped me learn all the necessary basics of CNN and helped me in completion of this project [CNN20]

7. References:

[IC17] 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.

[THS21] Y. Liu, W. Zhong, W. Wang, Q. Cao and K. Luo, "A Method for Recognizing Prohibition Traffic Sign Based on HOG-SVM," 2021 International Conference on Computer Engineering and Application (ICCEA), 2021, pp. 486-490, doi: 10.1109/ICCEA53728.2021.00101.

[RT19] Alexander Shustanov, Pavel Yakimov, CNN Design for Real-Time Traffic Sign Recognition, Procedia Engineering, Volume 201, 2017, Pages 718-725, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2017.09.594. (https://www.sciencedirect.com/science/article/pii/S1877705817341231)

[TSB19] Traffic Sign Recognition by Barney Kim, Mar 2019

[CNN20] Convolutional Neural networks Explained, Mayank Mishra Aug 2020