

Traffic Signs Recognition

Avinash Kamatham

ITCS-5156 Fall 2022- Lee

October 12, 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.

Using autonomous and semi-autonomous intelligent robotic vehicles, this research describes a system for classifying photographs that is based on deep learning and employed in the detection of traffic signals. This traffic sign recognition technology was developed to automate automobiles, but it also assists human drivers in upholding traffic laws and increasing safety while operating the vehicle. The system must be able to categorize a variety of traffic signs (such as the maximum speed limit, stop, slow down, turn ahead, and pedestrian) to make it easier to navigate according to local traffic laws.

1 Introduction

1.1 Problem Statement

In order for a vehicle, whether autonomous, semi-autonomous, or even operated by a person, to properly navigate on them, structured road scenarios and urban street environments require fundamental and correct traffic laws as well as a variety of information for drivers. The three types of traffic signs that are most frequently used are traffic lanes, traffic lights, and traffic signs (for speed control, stop and direction) (regulation of flows and space). Traffic signs are used to both inform drivers of any local traffic limitations and to assist drivers in their driving chores.

These warnings are crucial for both the driver of a semi-autonomous or intelligent vehicle and a vehicle that travels autonomously and abides by the same traffic laws as other vehicles and people. This research used a Deep Learning-based image classification system to help intelligent autonomous cars recognize traffic signs for the benefit of road safety.

1.2 Motivation and Challenges

My rationale for doing this is that many researchers and major corporations are working on autonomous vehicles and self-driving cars in the field of artificial intelligence and technological growth.

Therefore, for this technology to be accurate, the vehicles must be able to read traffic signs and act accordingly. Therefore, I want to create a model that can classify photos of traffic signs into several groups using input from users.

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						

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.

Artificial neural network classification is a highly well-liked method for addressing pattern recognition issues. A neural network is a mathematical representation of a biological brain network made of artificial neurons connected to one another. Neurons are typically arranged in layers, and connections are only made between neurons from adjacent layers. The first layer contains the input low-level feature vector, which is then turned into the high-level feature vector as it moves through the layers. The number of categorizing classes is the same as the number of output layer neurons. As a result, the probability vector representing the likelihood that the input vector belongs to the specified class makes up the output vector.

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

$$a_j^i = \sigma(\sum_k a_k^{i-1} w_{kj}^i),$$

where a_j^i is the j^{th} neuron in the i^{th} layer, w_{kj}^i 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.

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

Convolutional neural networks are currently the most advanced pattern recognition technique in computer vision for classifying images. Convolutional neural networks interpret two-dimensional images using convolutional layers, in contrast to standard neural networks, which operate on one-dimensional feature vectors. The number of layers can be optimized for the particular task.

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

This section focuses on adjusting the size of the traffic sign images, color image graying, and gray image. HOG feature extraction and gamma correction. Gamma correction is mostly used to adjust the image's brightness and lessen the impact of light and shadow on the picture.

2) HOG feature extraction

Feature extraction from HOG, with related spatial, cell, block, and sliding step sizes.

3) Classification and recognition of traffic signs based on SVM

In the first step of this section, a support vector machine is utilized to create the initial classifier for traffic signs based on the training data set. 42 binary classifiers are used to build the initial classifier. The subsequent step is to obtain the final classification result using the original classification result.

The model architecture is shown in figure 3 below

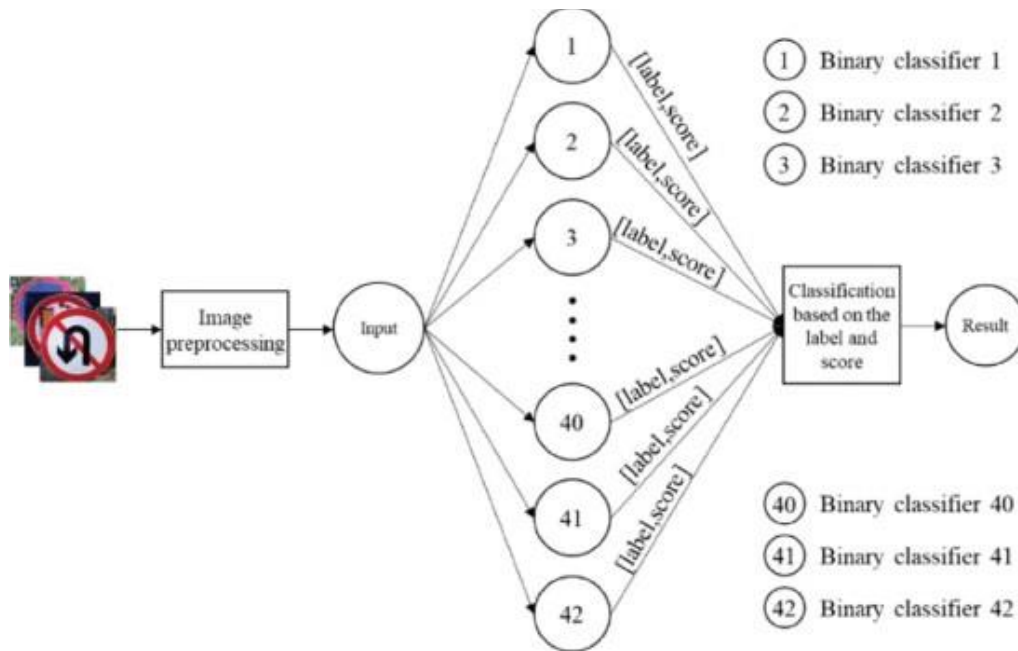


Figure 3: HOG-SVM based Traffic Sign recognition

2.3 Pros and Cons of the Approaches

One of HOG's shortcomings is that, because it uses a sliding window technique to extract characteristics from each pixel of an image, its computation time is slow when recognizing an object for large-scaled images. In comparison to the present convolutional neural networks, the accuracy is therefore not very trustworthy.

CNN's fundamental advantage over its forerunners is that it uses machine learning to identify key elements without human intervention. However, CNN does not encode the position or orientation of the object, which is a drawback. lack of spatial invariance with respect to the supplied data A large amount of training data is needed.

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 consists of several traffic indicators divided into 43 different classifications. The dataset has about 50,000 photos in total. The dataset is divided into 34799 training photos, 4410 validation images, and 12630 testing images.

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

Class Imbalance Struggle

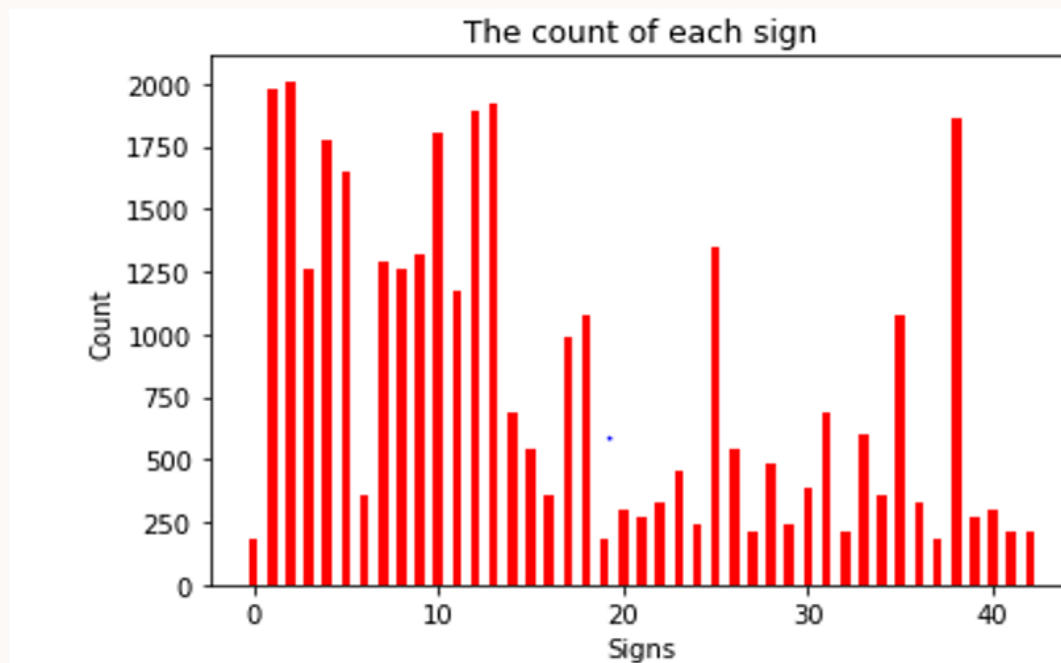


Figure 4: Count of images in each class

3.2 Construction of Baseline Model

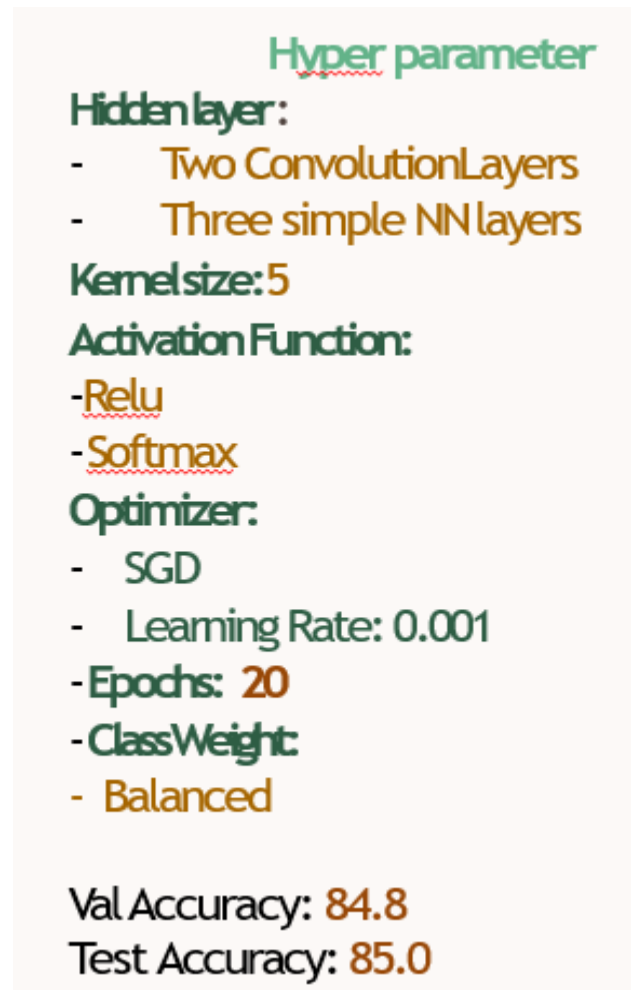


Figure 5: Baseline Model Hyper Parameters

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.

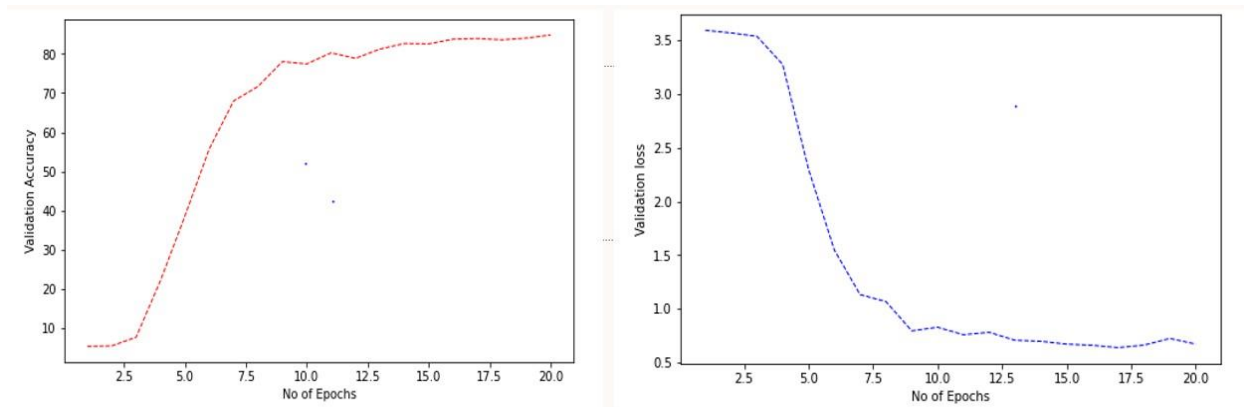


Figure 6: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

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.

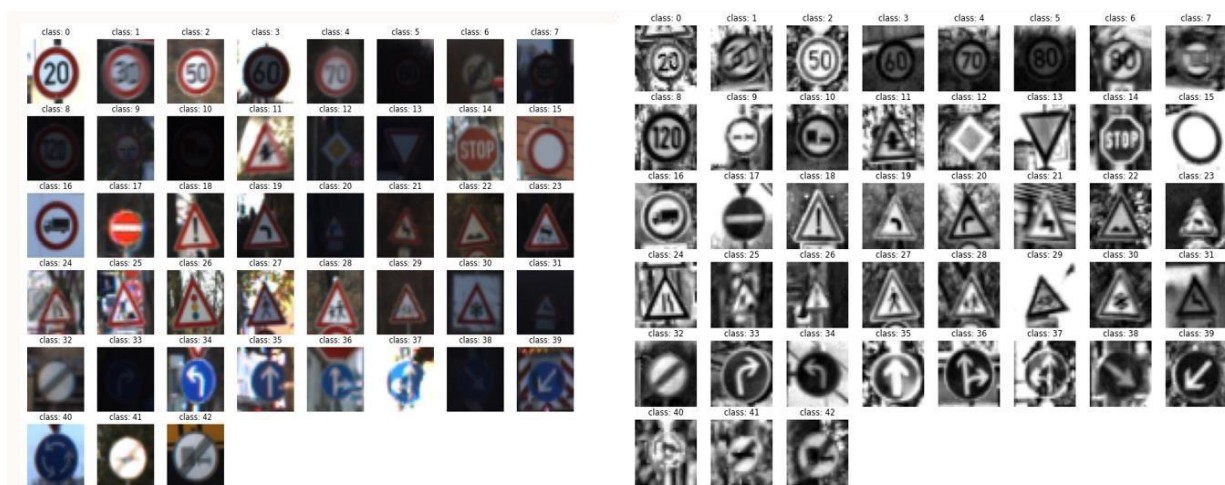


Figure 7: Original Images vs Applying Gray Scale and CLAHE

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.

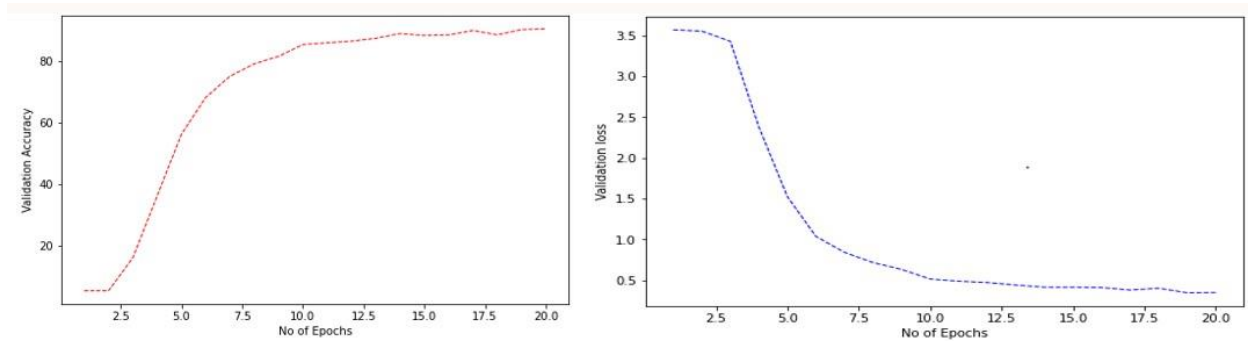


Figure 8: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

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.

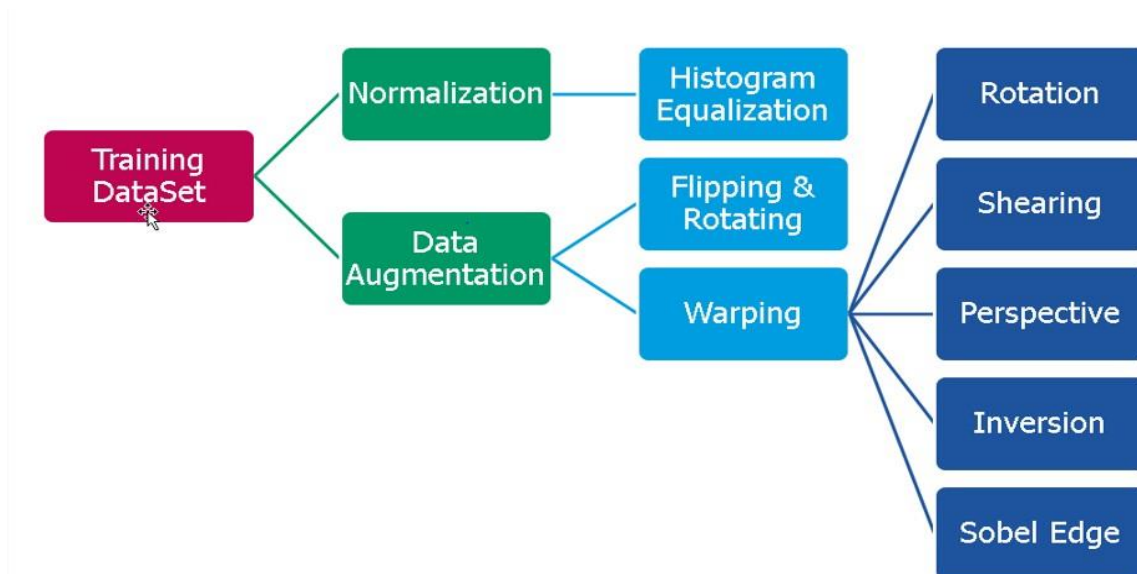


Figure 9: Data Augmentation Techniques

In order to generate additional images for each label and correct the class imbalance, I rotated, sheared, and flipped the given dataset. Additionally, it aids in raising the number of training samples for the model, which raises the model's accuracy.

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.

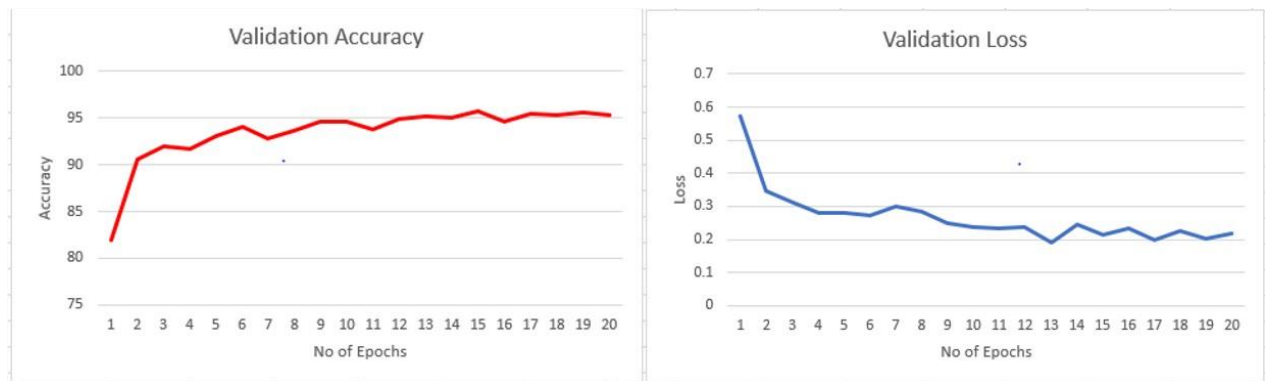


Figure 11: No of Epochs vs Validation Accuracy, No of Epochs vs Validation Loss

3.5 Final Model of Deep Convolutional Neural Network

Model Architecture:

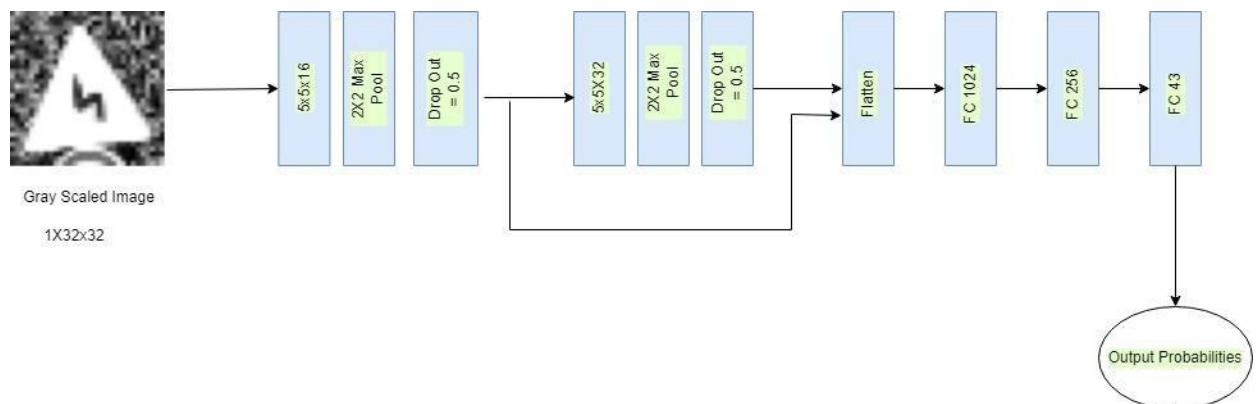


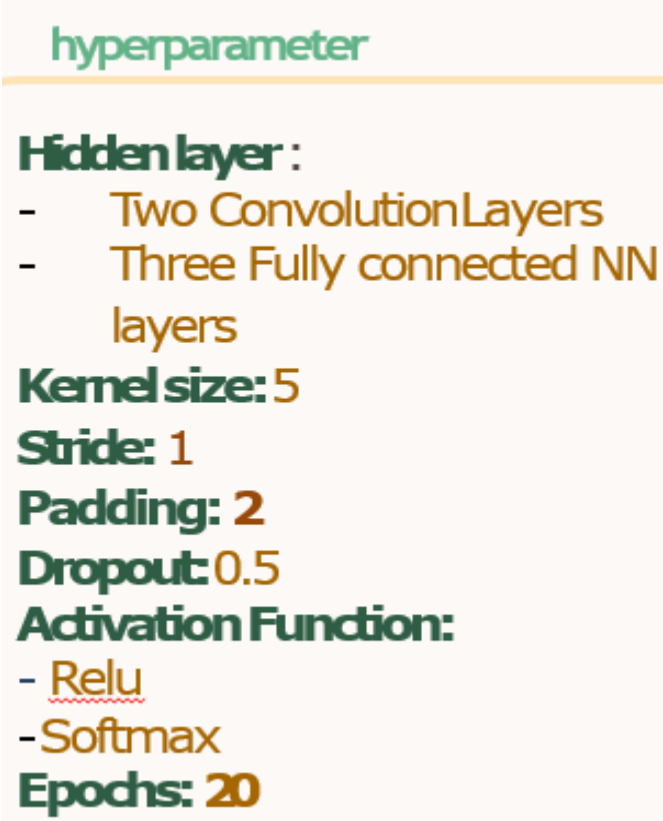
Figure 12: Model Architecture

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

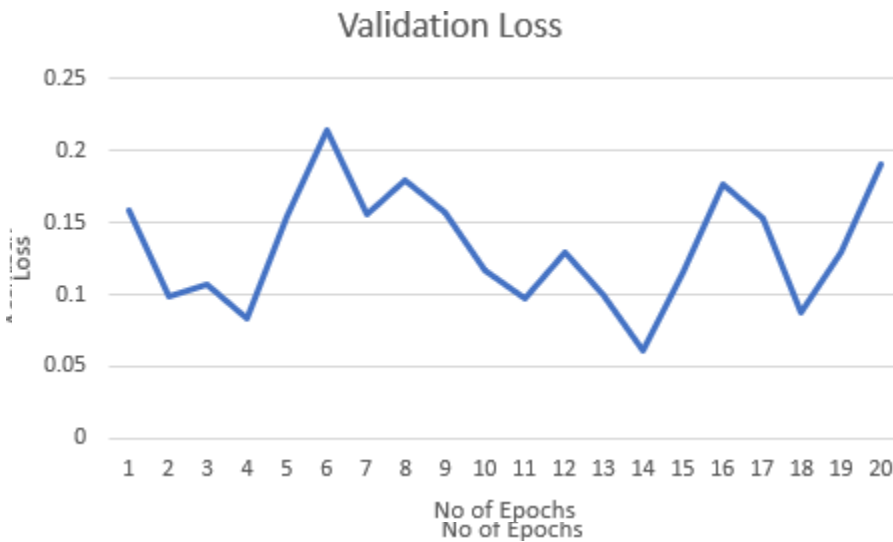
Epochs: 20

Figure 13: hyper parameters 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:

I was able to create the final model after conducting extensive research, reading numerous articles in Medium, and working with PyTorch to incorporate works on Data Augmentation and CNN construction.

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 finished the data augmentation by shearing, rotating, and flipping the photos, which creates an extended dataset from the original images and provides extra data to train the model with. Test accuracy has increased to 93.793% after data augmentation, which is significantly better than it was before. The model was trained on a GPU using Google Collab in little under two hours and thirty minutes.

After a great deal of experimenting with the number of convolutional layers and thinking about the number of fully connected hidden layers, as well as with various learning rates, Max pool sizes, strides, and padding, I finally came up with the Deep Neural Network that is described in Section 3.5. Using Google Collab and a GPU, it took over three hours and fifteen minutes to train the model. The model's Test Accuracy is 96.698%, which is a little less accurate than the model presented by D. R. Bruno and F. S. Osorio, which has a Test Accuracy of 97.85%.

Although I can duplicate the experiment described in the study, the outcomes are not the same. My test accuracy was 96.698%, less than 1% of the study published by D. R. Bruno and F. S. Osorio's test accuracy.

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.

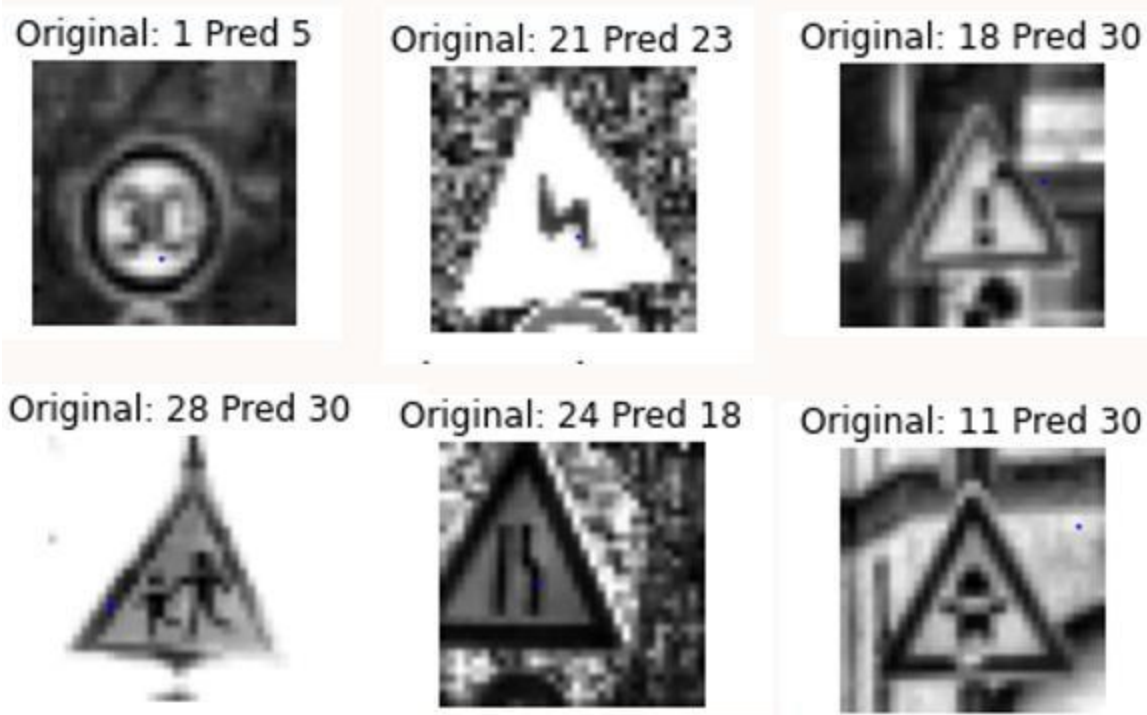


Figure 14: Example showing the wrongly classified images

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.

With a Test accuracy of 96.698, the model has performed well in my opinion. However, the accuracy of the model can still be increased by experimenting with different hyper parameters, such as the number of convolutional layers, the kernel size, the stride, the padding, or even the learning rate. Spatial transformer networks can also aid to boost accuracy up to 99 or 100%.

5. Conclusion and Future Work

Due to the limited availability and the length of time required to train a model on GPU using Google Collab for 20 epochs, I was able to test various hyper parameters to increase the test accuracy of the model, including the number of convolution layers, kernel size, stride, padding, and even the learning rate.

I am new to these Deep Learning model concepts, but I have learned a lot about them while putting the project together, including how to use convolutions, max pooling, activation functions, and cost functions of neural networks. I'm pleased that the Convolutional Neural Network I was able to build for image processing produced some good outcomes.

I can understand the principles and get through these difficulties with the aid of articles from Medium and YouTube videos on convolutional neural networks.

6. My Contributions:

In order to finish the work, I consulted various articles on Medium and Kaggle. I used this Medium article, which Barney Kim has already implemented [TSB 19], for data loading, image processing, and data augmentation. I therefore utilized this already-existing work for data augmentation and image processing.

My contribution entails building a baseline model and training it at several phases following grayscale picture conversion, CLAHE, and data augmentation. With two convolutional layers and three fully linked layers, I also built my final Deep Convolutional Neural Network model and trained it. In order to display the photos where the model failed to correctly categorize the given image, I also created various plots for comparison of Number of Epochs against Validation Accuracy and Number of Epochs versus Validation Loss, Construction of Confusion matrix. I was able to complete this project [CNN20] by learning all the fundamentals of CNN from this post.

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