TRAFFIC SIGN CLASSIFICATION

A Special Assignment Report

Submitted in the
Open Elective Course

FUNDAMENTALS OF IMAGE AND

VIDEO PROCESSING

By

18BCE027 – SAHIL BHINGRADIYA
18BCE154 – BHOOMI PATEL
B.Tech Sem VI
COMPUTER SCIENCE & ENGINEERING

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY
AHMEDABAD-382481

APRIL 2021

ABSTRACT

The human visual system is the key to transferring the information between the environment and the vehicle by human perception and control. Along with the development of road infrastructure, any autonomous vehicle must be able to operate safely in the existing environment, it also needs human-level visual recognition of the environment in which it operates. Visual Classification is the most complex task in the domain of visual recognition. It is the key task of the autonomous vehicle in visual perception. That's why the Traffic Sign Classification is the most important integral part of the autonomous vehicle and Advanced Driver Assistance System - ADAS.[1]

Most of the time drivers miss the Traffic Signs on the roads due to some obstacles or lack of attentiveness. So the chances of an accident will increase. Automatic Traffic Sign Classification is very important because it classifies the Traffic Sign Images correctly so chances of the accident will decrease. To do this classification task everywhere Deep Learning Networks are used. Along with the rapid development of Deep Learning Algorithms, its structure, feasibility, and high-performance implementation with GPU, it gives the best result. [3]

In this paper, we proposed a deep neural network for traffic sign classification using Deep Learning techniques and the German Traffic Sign Recognition Benchmark - GTSRB dataset which performed very well in classification. To achieve our goal we use the Convolutional Neural Network using python. We use the google Tensorflow framework via python API. We Use the pickle files for Training, Validation, and Testing Examples. We did the preprocessing on the Images.

Traffic sign classification becomes a mature area with an increasing focus on autonomous driving research. Notable research work exists on the detection and classification of traffic signs for advanced driver assistance systems. Most of the works attempted to address the challenges involved in real-life problems due to scaling, rotation, blurring, etc. So we use the Different Data Augmentation techniques to Balance the given dataset. We use Zooming, Translation, Shearing, Flipping, Rotations, Sobel Edge, Inversion to augment the data. We can do this task in less number of parameters. If parameters are less so computations are also less and memory requirement is very low. We use the Normalization, Histogram Equalization for better-balanced pixel intensities of the images. After we train a CNN model using Training and Validation data, Softmax as the activation function. It gives us the best result for Traffic Sign Recognition. We get 95% Testing Accuracy. [7]

1. INTRODUCTION

Traffic Sign Classification is the most challenging problem for many real-world applications like Autonomous vehicles, ADAS, Sign monitoring for maintenance, etc. Some factors such as color fading of the traffic signs, weather conditions, undesirable backgrounds, poor visibility because of lightning, shadows, blurred images because of the speed of the vehicle make the task more challenging. For an autonomous vehicle, correct classification of the traffic signs must be required. An autonomous vehicle needs to detect the Traffic sign correctly and change its behavior according to it. [1]

1.1 Problem Statement

For the GTSRB challenge identify and classify the images of the traffic signs into 43 different classes using Deep Learning Convolutional Neural Network. Preprocess the data using normalization and data augmentation techniques to remove the unfavorable features of the images.

1.2 Objective

GTSRB dataset consists of the images extracted from the video under different conditions like illumination, weather, speed, day, night, etc, using the vehicle-mounted camera. The following objects are expected from the model. [3]

- Remove the effects of the illumination from the images
- Scaling and Normalization
- Data augmentation techniques like zoom, rotation, shear, brightness disturbance, Gaussian noise, color inversion, etc.
- Train the CNN model on the traffic sign images
- Classify the images into the appropriate class

2. LITERATURE REVIEW

There is a good amount of research in the area of traffic sign recognition. The paper (Ellahyani et al., 2016) introduces a new system that involves three stages. Segmentation of acquired images to extract ROI, a shape classifier is implemented to classify the extracted picture using HOG features. The extracted features are then provided to a Random Forest classifier to perform sign recognition. The paper utilizes a locally normalized HOG as feature descriptors for Human Detection. They use the HOG features and perform a classification using SVM (Support Vector Machine). This helps us with understanding the current state-of-art approaches to such a problem. Also, VGGNet is a very famous Deep learning NN for training the Traffic Sign Classifier. [2]
All the above works provide us an approach for the implementation of road sign recognition. For our implementation, we have used preprocessing methods like scaling, normalization, Histogram Equalization, and data Augmentation Techniques.

3. METHODOLOGY

3.1 Dataset :

The GERMAN TRAFFIC SIGN RECOGNITION BENCHMARK (GTSRB) dataset is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks 2011. It consists of images of the traffic signs taken from the german roads and a class label for each image. The images are taken from the video data of a moving car camera after removing all temporal information. [4]

- 43 different classes
- More than 50,000 Images in total
- consist of the class label (between 0 to 42) and sign name for each class
- We divide the whole dataset into Training, Validation, and Testing Images
- Each image is 32 x 32 pixels made up of 3 color channels formatted in RGB. Each pixel consists of an 8-bit integer that gives a total of 256 values between 0 to 255. [4]

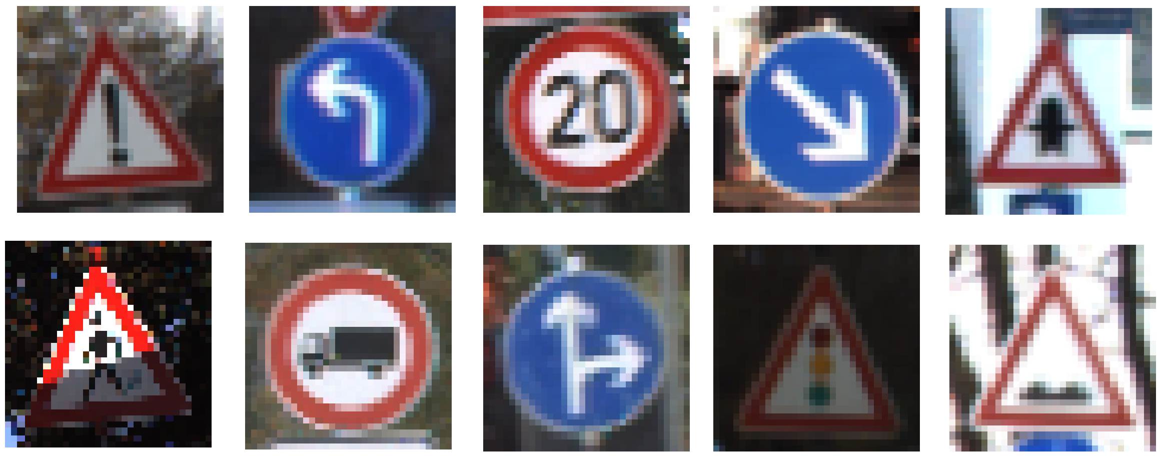


Figure 1: Random Images of the Dataset

3.2 DataSet summary and Exploration

We divide the whole dataset into Train, Validation, and Test. We use the pickle file for all 3 examples. A pickle file is mainly a dictionary with 4 key-value pairs.

- 'features' is a 4D array containing raw pixel data of the traffic sign images (num examples, width, height, channels).
- 'labels' is a 1D array containing the label/class id of the traffic sign. The file `signnames.csv` contains classes and their respective sign names.
- 'sizes' is a list containing tuples (width, height) representing the original width and height of the image.
- 'cords is a list containing tuples (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image.

3.3 Data Visualization

- Sort the data
- Distribute the data using NumPy
- CSV file read using the pandas' Data Frame
- Add the additional Columns into the Data Frame
- Plot the Data Frame as the Histogram using Matplotlib

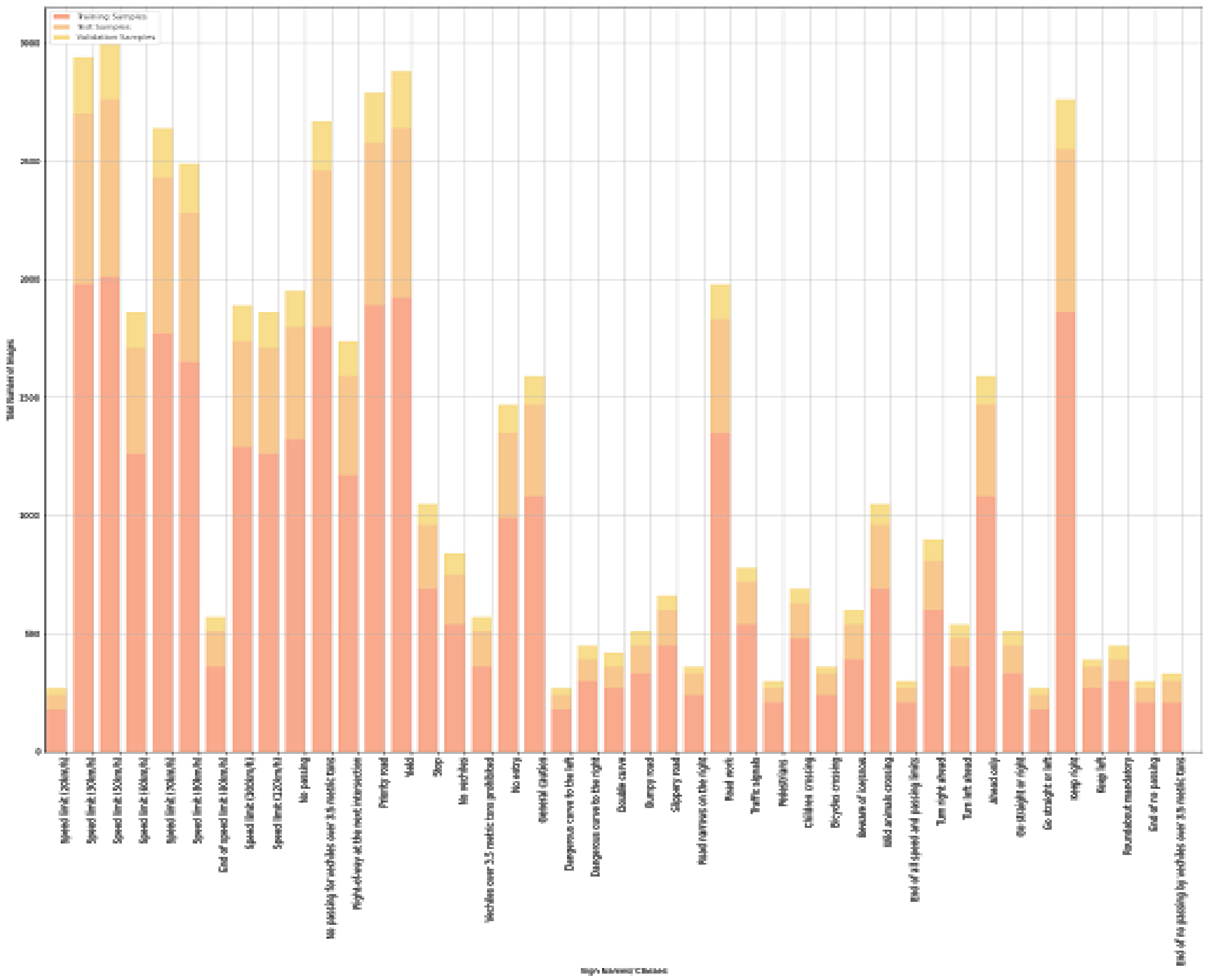


Figure 2: Visualization of the dataset
(Red - Training Data, Orange - Validation Data, Yellow - Testing Data)

3.4 Data Augmentation

In figure 2 we can see that the dataset is very unbalanced. Some classes are significantly better than others. In figure 3 we plot 10 random images of the same class. The images are significantly different in terms of contrast and brightness, so we need to do the histogram equalization, so while training we can extract good features.

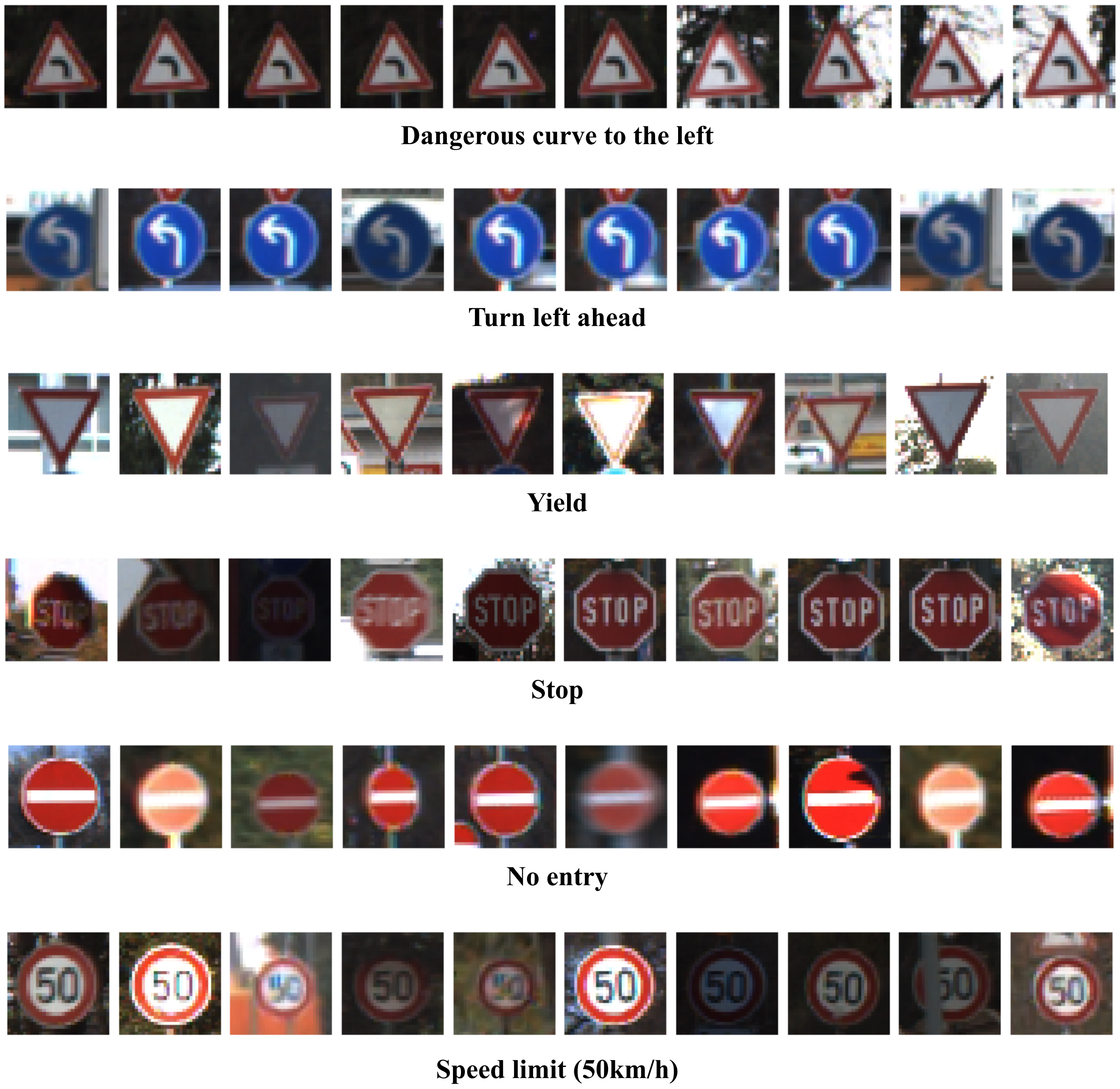


Figure 3: Random images of the same class

Flipping: We can extend the dataset using flipping. You can be noticed that some traffic signs are invariant to horizontal like **Speed limit (30km/h)** or vertically flipped like **Right-of-way at the next intersection**. We can flip some signs in any way horizontally or vertically like **Priority road** or **No entry**. Some signs are invariant to 180 degrees flipped like **End of all speed and passing limits**. After all these operations the resultant image will be also classified as the same class. Some of the signs that we can flip and we will get some other sign of some other class like **Turn left** become **Turn right**. All these operations are of no cost and no computation. [6]

Rotation: A Rotation is a circular movement of an object around a center of rotation. We do a random rotation between -10 to 10 degrees.

Translation: Translation is used to improve the visualization of an image. We do random translations between -10 to 10 in all directions.

Zoom: When we zoom an image, pixels are interested in the image in order to expand the size of the image, we use random zoom between 1 and 1.3. [5]

Histogram Equalization: This will spread out the most frequent intensity of the image, it will enhance the images with the low contrast. We deal with real-life data so it is very useful.

Normalization: Change the range of the pixel intensity values.

$$x_i = \frac{(x_i - \mu_B)}{\sqrt{\sigma_B^2 + \epsilon}}$$

Shuffling: Shuffle is a permutation of n elements. In every Shuffle it produces a new permutation and at some point, it would return the original order. It is used to increase the randomness and variety in the dataset.

Shearing: Transformation that slants the shape of an object is called shearing Transformation. We do random shearing between -25 to 25 degrees.

Invert: Inversion is a technique of image processing where light areas are mapped into dark and vice versa. An inverted black and white **image** can be thought of as a digital negative of the original **image**. randomly invert 1/4th of the images in each batch.

Sobel Edge: Sobel filter is used to detect a Sobel edge. It works by calculating each pixel's intensity within the image. randomly apply a Sobel edge detector to 1/4th of the images in each batch. [5]

Projection Transformation: The projection transformation converts the viewing frustum into a cuboid shape. It does shearing and scaling as we randomly position image corners in a [-delta, delta] range.

Delta = 30 * Pixel intensity

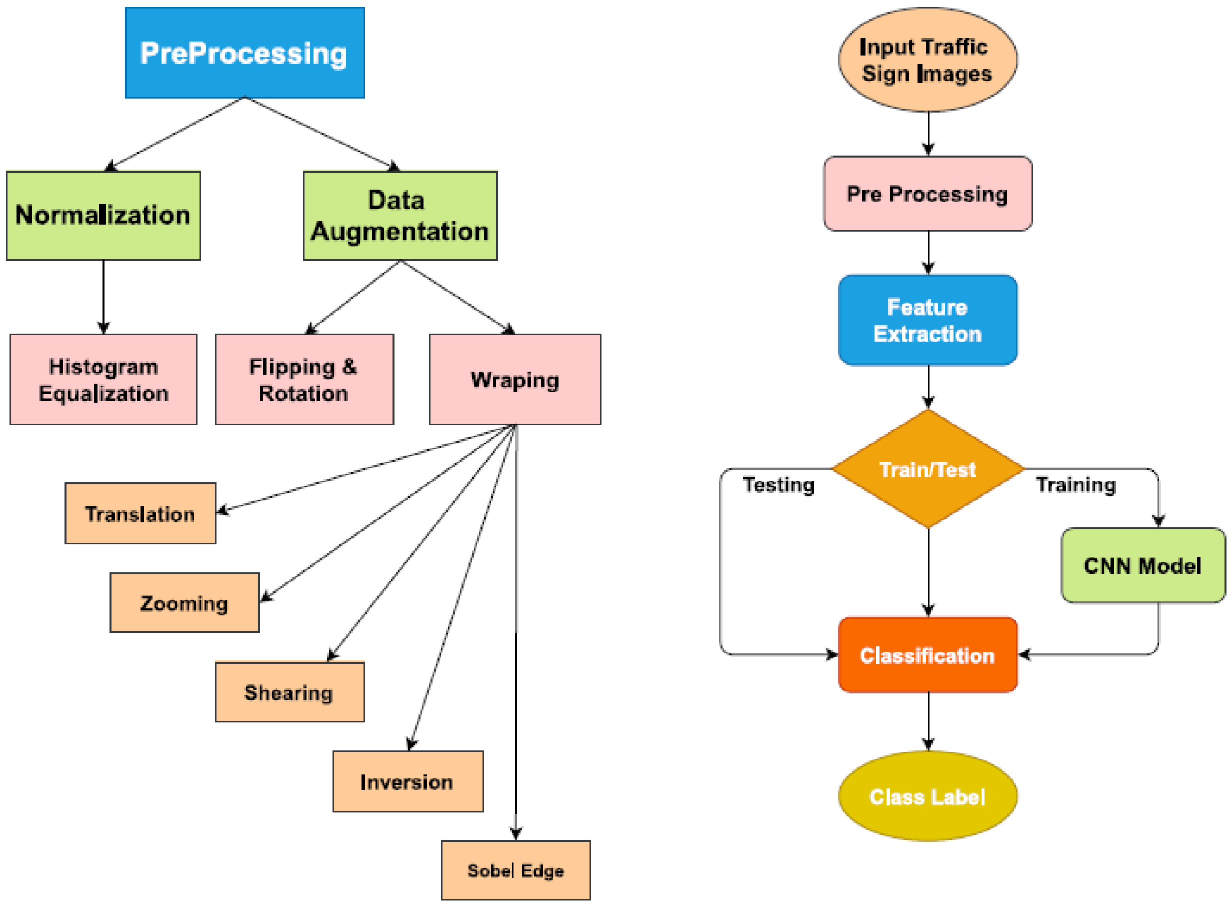
Gaussian Noise:

Gaussian Noise is a statistical Noise having a probability density function equal to a normal distribution, also known as Gaussian Distribution. Gaussian noise is also known as Electronic noise because it arises in detectors or amplifiers.

The formula for Gaussian noise,

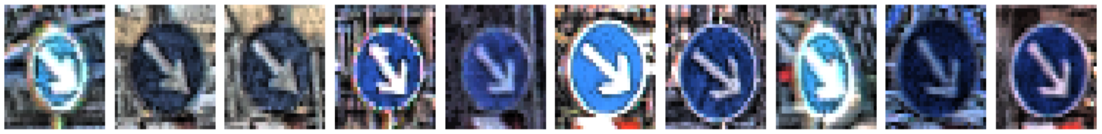
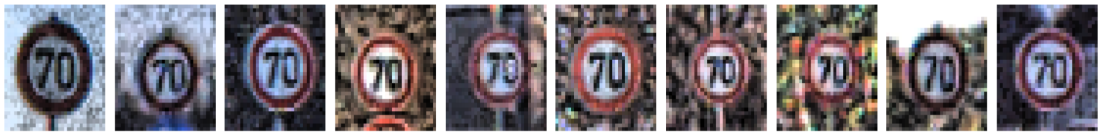
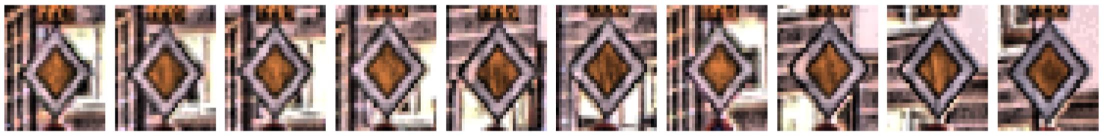
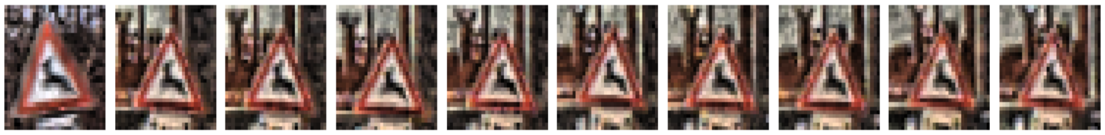
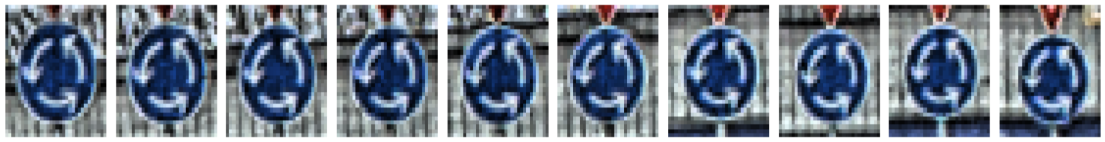
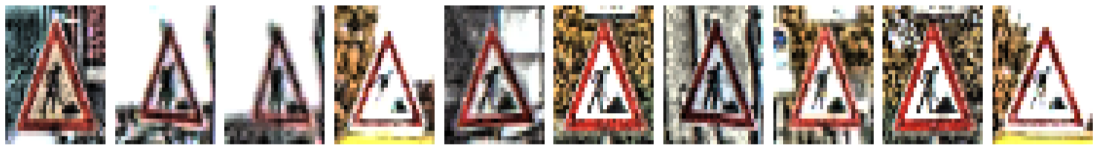
$$P_G(z) = \frac{1}{\sigma\sqrt[2]{2\pi}} \frac{e^{-(z-\mu)^2}}{2\sigma^2}$$

$$P_G(z) = Probability\ Density\ Function$$
$$\sigma = Standard\ Daviation$$
$$z = gray\ level\ Imge$$
$$\mu = Mean$$



(a) Steps of PreProcessing **(b)Proposed CNN Model**

Figure 4

Original	Augmented Intensity = 0.75
	
	
	
	
	
	

Tabel 1: Augmented Images related to the Original Image

3.5 Model

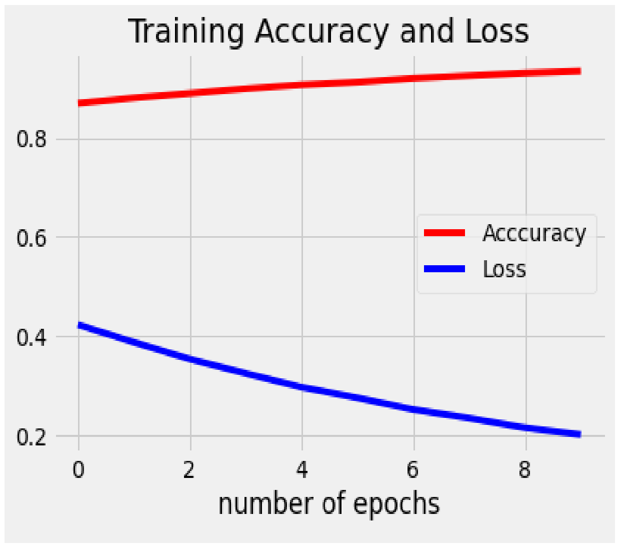
We use the Deep Neural Network Classifier as our Training Model. After Data augmentation, we get 129100 training examples, 12630 testing images, 4410 validation images. We rescale all the images between [0,1] range. We use four con2D layers, two MaxPooling2D layers, two dense layers, dropout, and BatchNormalization, Flatten. We use **Activation = Softmax** as the activation function in the output layer, and relu as all hidden layers. We also use **Learning Rate = 0.001**, **Optimizer = Adam,** we train the model 1st for batch size = 100 and for 10 epochs and finally **Batch size = 250 and Epochs = 10.**

Layer	Filters	Kernal_size	Activation	Output
Convolution_1	16	3 x 3	Relu	30 x 30 x 16
Convolution_2	32	3 x 3	Relu	28 x 28 x 32
MaxPooling_1	-	2 x 2	-	14 x 14 x 32
BatchNormalization_1	-	-	-	14 x 14 x 32
Convolution_3	64	3 x 3	Relu	12 x 12 x 64
Convolution_4	128	3 x 3	Relu	10 x 10 x 128
MaxPooling_2		2 x 2	-	5 x 5 x 128
BatchNormalization_2	-	-	-	5 x 5 x 128
Flatten	-	-	-	3200
Dense_1	512	-	Relu	512
BatchNormalization_3	-	-	-	512
Dropout(0.5)	-	-	-	512
Dense_2	43	-	Softmax	43

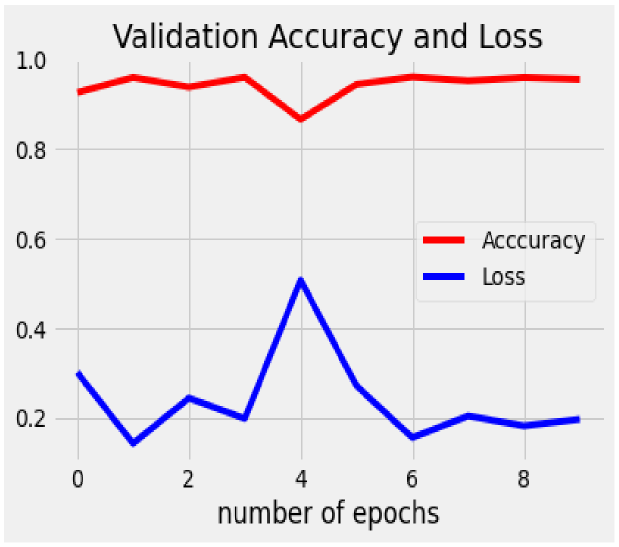
Table 2: Layers of the CNN Model

4 IMPLEMENTATION RESULTS

After all the pre-processing of the images, we get the good quality of the image as compared to the original image. After Data augmentation, we get 129100 training examples instead of 34799, 12630 testing images, 4410 validation images. We get the Balanced Training Dataset to train the model. After training the model we get Training accuracy as 93.55%, Validation accuracy as 95.65%, and Testing accuracy as 94.50%.



(a) Training Accuracy and Loss



(b) Validation Accuracy and Loss

Figure 5: Plot of Results

5 CONCLUSION

This paper proposes the Scaling, Normalization techniques, and Data Augmentation techniques like zoom, rotation, shear, Gaussian noise, color inversion, etc. It will almost remove all the unfavorable conditions like color, weather conditions, undesirable backgrounds, poor visibility because of lightning, shadows, blurring, and Improve the image quality. This paper proposes a deep convolutional neural network with a fewer number of parameters and memory requirements in comparison to existing methods. The presented network uses the Convolution Layer, Max Pooling Layer, Batch Normalization, Dropout to avoid overfitting due to the large dataset, some fully connected Layers. We get the Training accuracy as 93.55%, Validation accuracy as 95.65%, and Testing accuracy as 94.50%. This will give us the best result

6 FUTURE SCOPE

In this approach, we use the Simple Convolutional Neural Network to train our model and we use scaling, normalization, and data augmentation techniques to improve the quality of the images. We can try different sharpening techniques also. We also use the RGB to Gray, thresholding, Morphology, erosions, and dilations. We can try with the different filters like high pass filters, low pass filters, Ideal and Butterworth filters, laplacian filters, adaptive mean filters, etc. We can try with the different Deep Learning Architectures like VGGNet, GoogleNet, Google Inception, Transfer Learning, and Domain Adaptation Techniques to Classify the image and achieve better accuracy. We can also try real-time traffic sign detection and recognition using a webcam or video.

REFERENCES

[1]. “Traffic Sign Classification Using Deep Inception Based Convolutional Networks” by Mrinal Haloi from IIT Guwahati, July 2016

[2]. Prasanna Mahesh Gunawardana Monash University “Automatic detection and recognition of traffic sign” Australia March 2016

[3]. Pratiksha Manjare Prof. S. M. Hambarde Electronics-Telecommunication Department JSCOE, “Image Segmentation and Shape Analysis for Road Sign Detection” Pune India December 2014

[4]. GTSRB - German Traffic Sign Recognition Benchmark Dataset from Kaggle

[5]. Florian Muellerklein blog on Machine learning and statistics Feb 2016

[6]. Alex Staravoitau “Traffic sign Classification with CNN” January 2017

[7]. Danyah A.Alghmghama Ghazanfar Latif Jaafar Alghazo Loay Alzubaidi “Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN” January 2020