# TRAFFIC SIGN AND LANE RECOGNITION WITH DEEP LEARNING USING COMPUTER VISION IN AUTONOMOUS VEHICLES

*Report submitted to the Vellore Institute of Technology*
*as the requirement for the course*

**MINI PROJECT**

*Submitted by*

**P. Kundan Sai**
**(Reg. No.: 123003196, B.Tech-CSE)**

**K Ved Karthik Manikanta**
**(Reg. No.: 21BML0128, B.Tech- ECE BML)**

**V. Sai Jayanth**
**(Reg. No.: 123015105, B.Tech- IT)**

## List of Figures



## List of Tables

## ABBREVIATIONS

| | |
|---|---|
| CNN | - Convoluted Neural Networks |
| STN | - Spatial Transformer Networks |
| DL | - Deep Learning |
| GTSRB | - German Traffic Sign Recognition Benchmark |
| RELU | - Rectified Linear Activation Unit |

## NOTATIONS

No specific notation is used.

# ABSTRACT

We observe a huge increase in the number of vehicles and so are the road accidents, this is due to the poor enforcement of laws, and carelessness as people do not follow or recognize traffic signboards and lane rules. Automatic detection and recognition of traffic signs are very important. It could potentially be used for driver assistance to reduce the number of accidents and eventually in driverless automobiles. In this project, we are implementing a system that recognizes German traffic signs, signals, types of lanes, and speed breakers which alerts the driver to be cautious. The proposed system works in real-time detecting and recognizing traffic sign images with the use of CNN (Convolutional Neural Network) using Spatial Transformer Networks for the accuracy in the automobiles. This system consists of 2 parts, Detection of traffic signs and classifying them based on CNN. The data set includes images that are taken from different angles and include other parameters and conditions. The training and testing for the model will be done using the German Dataset. For detection of the traffic sign, a CNN with LeNet-5 architecture is implemented using Adam optimizer.



Keywords- LeNet-5 architecture, CNN, Traffic Sign Detection

# TABLE OF CONTENTS

# CHAPTER 1

# SUMMARY OF THE BASE PAPER

## 1.1. BASE PAPER DETAILS

**Title:** Deep learning using computer vision in self driving cars for lane and traffic sign detection
**Journal:** Springer Link (Original Article)

**Publisher:** Nitin Kanagaraj, David Hicks, Ayush Goyal, Sanju Tiwari & Ghanapriya Singh

**Published:** 14 May, 2021

**Web of science Core Collection:** Indexed in SCOPUS

## 1.2. INTRODUCTION

We observe a huge increase in the number of vehicles so are the road accidents. Be it a human driver driving a car or a self-driving car, following traffic signs on Indian roads is necessary . Major amount of accidents happen due to human error i.e, either the driver may not see the traffic sign clearly, or not notice the sign coming or not understand the meaning of a certain sign. For this reason, the proposed paper's major objective is to develop and increase the robustness and efficiency of traffic sign detection system. Automated driving and automated traffic sign detection might increase the demand for smart cars and smart driving.To implement this, a Convolution Neural Network with learnable module Spatial Transformer Networks (STN) is used to detect lane for autonomous self-driving cars . For detection of a traffic signal, CNN with LeNet-5 architecture is implemented, since machine learning and deep learning techniques are playing a significant role in object detection and data pre-processing. The paper presents the results of research efforts targeted at facilitating improvements in two areas that are very important for self-driving vehicles: lane detection and traffic sign detection. Adam optimizer is used for detecting traffic signal.The training and testing was done using the German Traffic Sign Dataset. The accuracy of the model was compared to the methodology of feed forward network.

## 1.3.METHODOLOGY

In recent years, many deep learning models combined with CNN has shown great results in the image processing area that includes image recognition and image detection. A CNN is a Deep Learning algorithm that takes an image as input, it identifies and gives importance to various aspects, and differentiates the image from other images. Neural networks generally have a higher performance rate than other image processing algorithms since they have little power of pre-processing.This neural network encompasses of convolutional layers,pooling layers,fully connected layers.

Adam optimizer was used in the detection of traffic signs as it is used for computer vision problems.Lenet-5 architecture has 7 layers. Firstly, convolution is done followed by subsampling. Finally, data is passed to the fully connected layer.

German Traffic Sign Dataset contains about 40,000 images which are classified into 43 different classes. Each class has about 30 images of traffic signs. The data set is divided into training data and testing data.During the training of the model, the activation function used is SoftMax and the loss function. The SoftMax function turns numbers into probabilities that sum to one. Respective graphs have been drawn and the accuracy is calculated. The accuracy obtained after the testing phase is 97%.

# CHAPTER 2

# MERITS AND DEMERITS OF THE BASE PAPER

## 2.1 MERITS OF THE BASE PAPER

The proposed architecture is aimed to deliver an accurate model in comparison with other methodologies such as the feed forward model. It generates the classification of images dynamically using the knowledge obtained during the training phase using the Adam optimizer, without consuming too much time or resources.It demonstrated the usefulness of CNN architecture by using it for building a practical model that can detect real-time Traffic Signs. The model can also detect objects that are in constant motion. Research in such fields can aim to deliver a fully automated self-driving vehicle.

## 2.2 DEMERITS OF THE BASE PAPER

The accuracy of the model with the classes highly depends on the images of the traffic signs. If the images given in the pre-processing are not up to the mark, classification during the testing might not show appropriate results, due to which accuracy will vary. Noise, Blurring , Orientation and many other metrics of the images have to be taken into consideration for better results.

# CHAPTER 3

# PROJECT PHASES AND CODE

## 3.1 Project Phases

### 3.1.1. Phase - I:

In this project, to efficiently predict the traffic sign we have to construct the CNN model which will efficiently detect it. Dataset was collected from **INI Benchmark** Website. The dataset is **German Traffic Sign Recognition Benchmark (GTSRB)**, a large multi-category classification benchmark with 43 classes.

```
classes = {1: 'Speed limit (20km/h)',
           2: 'Speed limit (30km/h)',
           3: 'Speed limit (50km/h)',
           4: 'Speed limit (60km/h)',
           5: 'Speed limit (70km/h)',
           6: 'Speed limit (80km/h)',
           7: 'End of speed limit (80km/h)',
           8: 'Speed limit (100km/h)',
           9: 'Speed limit (120km/h)',
           10: 'No passing',
           11: 'No passing veh over 3.5 tons',
           12: 'Right-of-way at intersection',
           13: 'Priority road',
           14: 'Yield',
           15: 'Stop',
           16: 'No vehicles',
           17: 'Veh > 3.5 tons prohibited',
           18: 'No entry',
           19: 'General caution',
           20: 'Dangerous curve left',
           21: 'Dangerous curve right',
           22: 'Double curve',
           23: 'Bumpy road',
           24: 'Slippery road',
           25: 'Road narrows on the right',
           26: 'Road work',
           27: 'Traffic signals',
```

```
28: 'Pedestrians',
29: 'Children crossing',
30: 'Bicycles crossing',
31: 'Beware of ice/snow',
32: 'Wild animals crossing',
33: 'End speed + passing limits',
34: 'Turn right ahead',
35: 'Turn left ahead',
36: 'Ahead only',
37: 'Go straight or right',
38: 'Go straight or left',
39: 'Keep right',
40: 'Keep left',
41: 'Roundabout mandatory',
42: 'End of no passing',
43: 'End no passing veh > 3.5 tons'}
```

### 3.1.2. Phase - II:

Then the collected dataset was pre-processed by carrying out basic augmentation techniques. In this operations carried out is listed below:

1. Rescaling (32 X 32)
2. Converting to Gray scale
3. Separating labels and images
4. Conveting images pixels to values between 0.0 and 1.0

### 3.1.3. Phase - III:

A model was created using Convolutional Neural Network and the model was trained.

Activation Functions used:

Activation functions define the resulting output of the node for the given input (or set) on inputs. It will map the output of the Network with a certain range. Here, mainly 2 activation functions are used.

1. Rectified Linear Unit: This function will map to the positive part of the output.[0,1]

$$f(x) = x' = max(0, x)$$

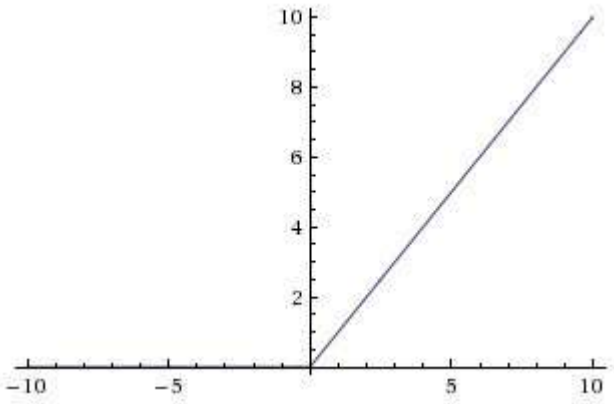


**Fig:1** Relu function

2. Sigmoid: This activation function will map the output to (0,1), but the curve will look like S-shaped. So, this function is used in the places where we have to predict the probability.

$$f(x) = \frac{1}{1+ e^{-x}}$$

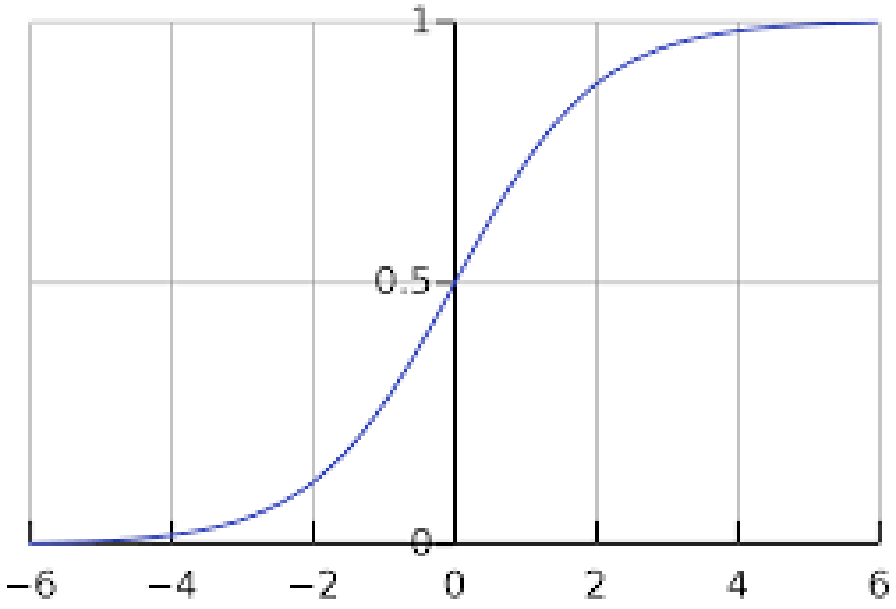


**Fig:2** Sigmoid function

### 3.1.4. Phase - IV:

Understanding the CNN Lenet-5 Architecture

1. 3 Convolutional layers
2. 2 Subsampling layers
3. 2 Fully connected layers

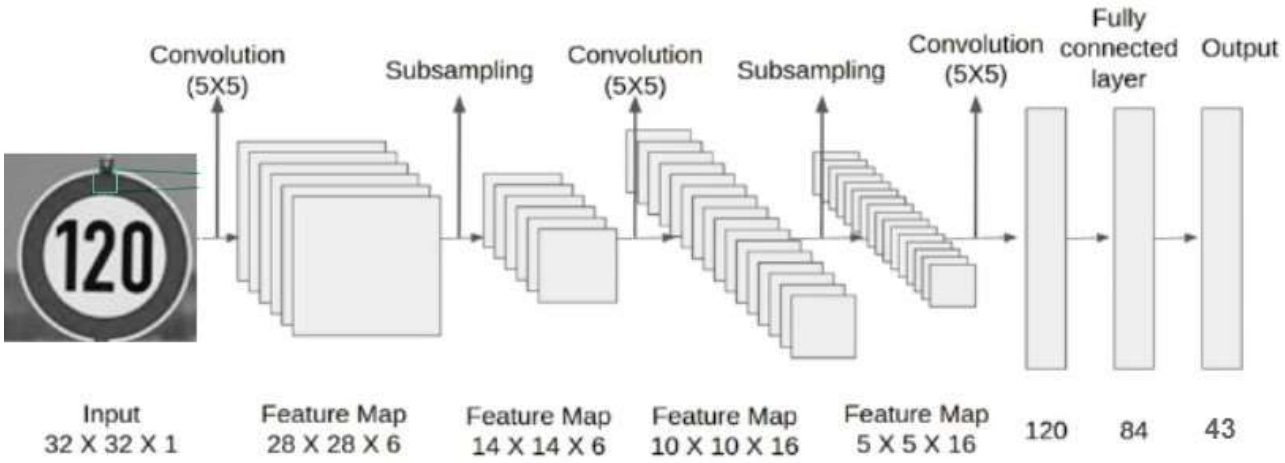


**Fig:3** Structure of Lenet-5 model architecture

### 3.1.5. Phase -V:

Performance Improvement:The Model which performed well was selected and to improve it, tuning has been done to improve the accuracy of model as well as its performance.

In tuning,

1. Drop Out was carried out to reduce Overfitting.
2. Batch Normalization was carried out to normalize the set of input data.
3. 50 epochs

### 3.1.6. Phase - VI:

Testing and Validation: Finally the tuned model was tested and accuracy was compared.

### 3.1.7. Phase - VII:

A GUI using python is created to test the model.Library used are

1. Numpy
2. Keras
3. Tkinter
4. PIL(Python Imaging Library)


**Fig:4** Uploading the image in the GUI to classify


**Fig:5** Correct model prediction of Traffic sign of class 6

# Source Code:

## Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

## Importing Libraries

```
import numpy as np
import random
import os
import cv2 as cv
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.models import Sequential, load_model
from keras.layers import Conv2D, Dense, Flatten, Rescaling, AveragePooling2D,
Dropout
```

## Reading and Pre-processing Images

```
images = []
labels = []
classes = 43
current_path = '/content/gdrive/My Drive/GTSRB/Final_Training/Images/'
for i in range(classes):
    path = os.path.join(current_path, str(str(i).zfill(5)))
    img_folder = os.listdir(path)
    for j in img_folder:
        try:
            image = cv.imread(str(path+'/'+j))
            image = cv.resize(image, (32, 32))
            image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
            image = np.array(image)
            images.append(image)
            label = np.zeros(classes)
            label[i] = 1.0
            labels.append(label)
        except:
```

```
            pass

images = np.array(images)
images = images/255
labels = np.array(labels)
print('Images shape:', images.shape)
print('Labels shape:', labels.shape)
```

## Splitting the Dataset into Train,Test and Validation subsets

```
X = images.astype(np.float32)
y = labels.astype(np.float32)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=123)
X_val=X_train[:5500]
y_val=y_train[:5500]
print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)
print('X_test shape:', X_test.shape)
print('y_test shape:', y_test.shape)
```

## Random 30 images from Dataset

```
plt.figure(figsize=(20, 20))
start_index = 0
for i in range(30):
    plt.subplot(6, 5, i+1)
    plt.grid(True)
    plt.axis('on')
    plt.xticks([])
    plt.yticks([])
    label = np.argmax(y_train[start_index+i])
    plt.xlabel('CLASS={}'.format(label))
    plt.imshow(X_train[start_index+i])
plt.show()
```

## Building the model

```
model = Sequential([
    Rescaling(1, input_shape=(32, 32, 1)),
    Conv2D(filters=6, kernel_size=(5, 5), activation='relu'),
```

```
    AveragePooling2D(pool_size=(2, 2)),
    Conv2D(filters=16, kernel_size=(5, 5), activation='relu'),
    AveragePooling2D(pool_size=(2, 2)),
    Conv2D(filters=120, kernel_size=(5, 5), activation='relu'),
    Dropout(0.2),
    Flatten(),
    Dense(units=120, activation='relu'),
    Dense(units=43, activation='softmax')
])
```

## Model Compilation

```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

## Model Architecture

```
model.summary()
history = model.fit(X_train, y_train, epochs=50,validation_data=(X_val, y_val))
```

## Validation Accuracy and loss

```
val_loss, val_acc = model.evaluate(X_train[:5500], y_train[:5500], verbose=2)
print('\nValidation accuracy:', val_acc)
print('\nValidation loss:', val_loss)
plt.figure(0)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.8, 1])
plt.legend(loc='lower right')
plt.figure(1)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.ylim([0, 0.2])
plt.legend(loc='lower right')
```

## Testing Accuracy and loss

```
test_loss, test_acc = model.evaluate(X_test,y_test,verbose=2)
print('\n Test accuracy:',test_acc)
print('\n Test loss:', test_loss)
```

### Saving the model

```
model.save('/content/gdrive/My Drive/keras_model/Trafic_signs_model.h5')
```

# GUI Code :

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

import numpy

from keras.models import load_model
import warnings

model = load_model('traffic_signs_model.h5')
warnings.filterwarnings("ignore", category=DeprecationWarning)

#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
           10:'No passing',
           11:'No passing veh over 3.5 tons',
           12:'Right-of-way at intersection',
           13:'Priority road',
           14:'Yield',
```

```
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
            36:'Ahead only',
            37:'Go straight or right',
            38:'Go straight or left',
            39:'Keep right',
            40:'Keep left',
            41:'Roundabout mandatory',
            42:'End of no passing',
            43:'End no passing veh > 3.5 tons' }
window = tk.Tk()
window.geometry('600x500')
window.title('Traffic sign classifier')

window.configure(background='#1e3e64')

heading = Label(window, text="Traffic Sign Classifier",padx=220,
font=('Verdana',20,'bold'))
heading.configure(background='#143953',foreground='white')
heading.pack()

sign = Label(window)
```

```
sign.configure(background='#1e3e64')

value = Label(window,font=('Helvetica',15,'bold'))
value.configure(background='#1e3e64')

def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    print(image.shape)
    pred = model.predict([image])[0]
    cnt=numpy.argmax(pred)
    sign = classes[cnt+1]
    print(sign)
    value.configure(foreground='#ffffff', text=sign)

def show_cb(file_path):
    classify_b=Button(window,text="Classify Image",command=lambda:
classify(file_path),padx=20,pady=5)
    classify_b.configure(background='#147a81',
foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.6,rely=0.80)

def uploader():
    try:
        file_path = filedialog.askopenfilename()
        uploaded = Image.open(file_path)

uploaded.thumbnail(((window.winfo_width()/2.25),(window.winfo_height()/2.25)))
        im = ImageTk.PhotoImage(uploaded)

        sign.configure(image=im)
        sign.image=im
        value.configure(text='')
        show_cb(file_path)
    except:
        pass

upload = Button(window,text="Upload an image",command=uploader,padx=10,pady=5)
```

```
upload.configure(background='#e8d08e',
foreground='#143953',font=('arial',10,'bold'))
upload.pack()
upload.place(x=100, y=400)

sign.pack()
sign.place(x=230,y=100)
value.pack()
value.place(x=240,y=300)

window.mainloop()
```

# CHAPTER 4

# SNAPSHOTS

## 30 Random images from the dataset

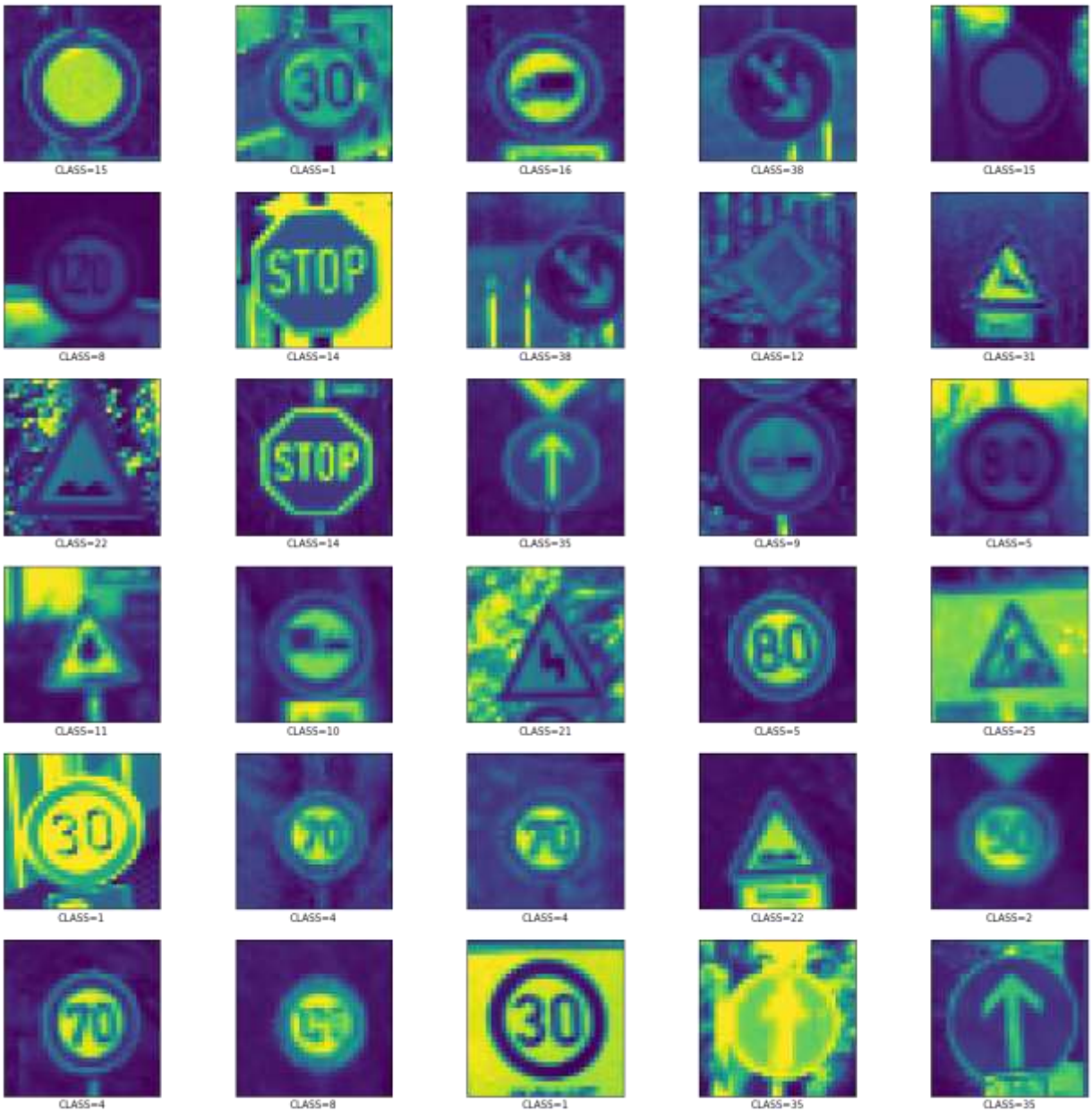


**Fig:6** Random 30 traffic sign images after gray scaling

## Model Summary:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 32, 32, 1)         0

 conv2d_3 (Conv2D)           (None, 28, 28, 6)         156

 average_pooling2d_2 (Averag  (None, 14, 14, 6)        0
 ePooling2D)

 conv2d_4 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_3 (Averag  (None, 5, 5, 16)         0
 ePooling2D)

 conv2d_5 (Conv2D)           (None, 1, 1, 120)         48120

 dropout_1 (Dropout)         (None, 1, 1, 120)         0

 flatten_1 (Flatten)         (None, 120)               0

 dense_2 (Dense)             (None, 120)               14520

 dense_3 (Dense)             (None, 43)                5203

=================================================================
Total params: 70,415
Trainable params: 70,415
Non-trainable params: 0
_________________________________________________________________
```

**Table 1:** Model Summary with total params

## Validation Accuracy and Loss:

```
172/172 - 2s - loss: 0.0047 - accuracy: 0.9991 - 2s/epoch - 10ms/step

Validation accuracy: 0.9990909099578857

Validation loss: 0.004732626024633646
```

## Testing Accuracy and Loss:

```
368/368 - 4s - loss: 0.0633 - accuracy: 0.9884 - 4s/epoch - 10ms/step

 Test accuracy: 0.9884412884712219

 Test loss: 0.06330263614654541
```

## Accuracy V/S Epoch:

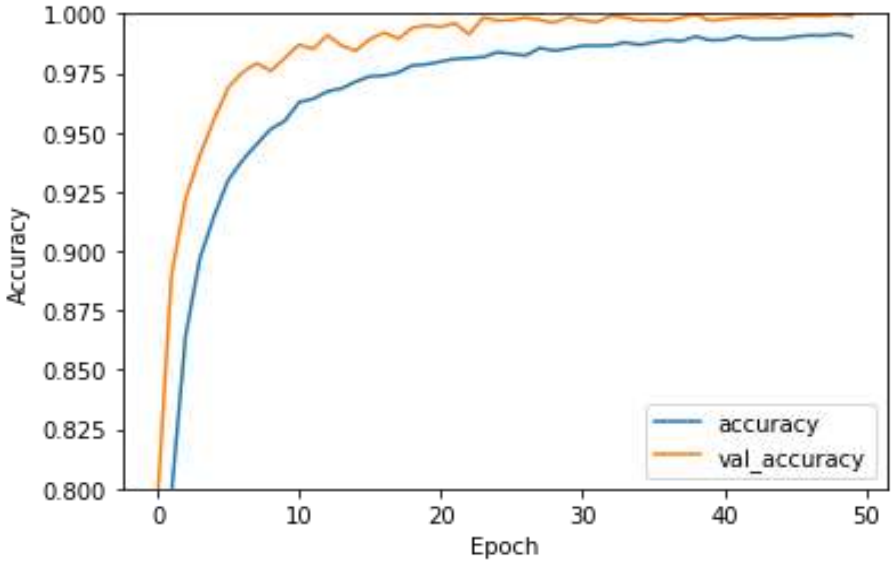


**Fig:7** The accuracy vs epoch of model for validation data

## Loss V/S Epoch:

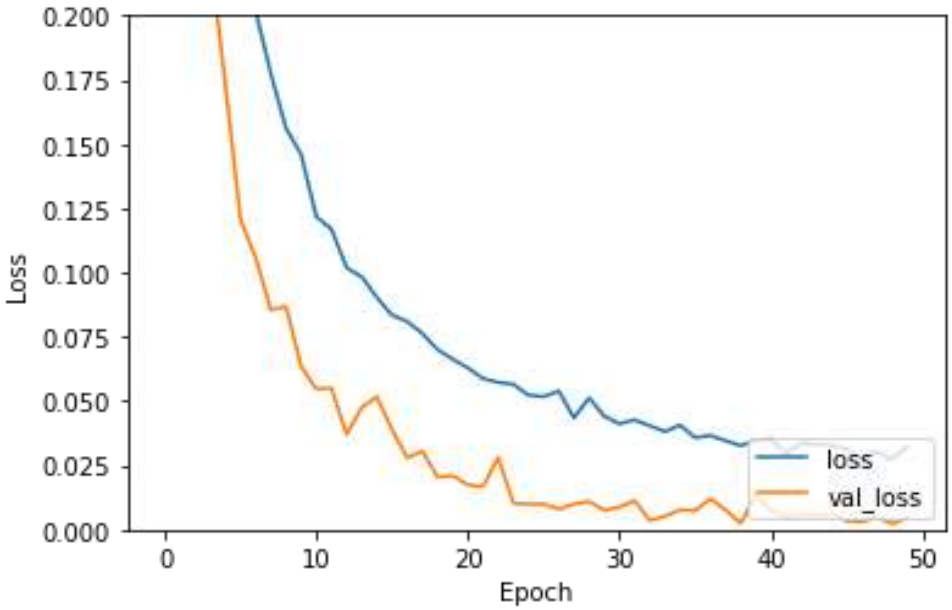


**Fig:8** The loss value vs epoch of model for validation data

# CHAPTER 5

# CONCLUSION AND FUTURE PLANS

## 5.1 Conclusion

In this project, we have created a Convolution Neural Networks (CNN) model which identifies the given traffic signs and classifies them accordingly. This is developed and tested on an open source German Traffic Sign dataset. We observed the accuracy and loss changes. The GUI of this model makes it easy to understand how signs are classified into several classes. The image is uploaded by clicking on upload image in gui and is classified into its category with 98% accuracy. Thus CNN can provide good accuracy in image classifications.

## 5.2 Future Plans

In future, in order to improve making potential contributions to fully automated self-driving vehicles research in these areas can be of a greater benefit. By that India can also start the era of automated self-driving cars, which can prevent human mistakes in driving in turn scaling down the number of accidents.

Future work will include increasing the size of the dataset and publishing it so that it can be used by other researchers for benchmarking purposes. Work will also continue on developing more robust and computationally low cost recognition systems.

# CHAPTER 6

## REFERENCES

[1] Nitin Kanagaraj, David Hicks, Ayush Goyal, Sanju Tiwari
https://link.springer.com/article/10.1007/s13198-021-01127-6

[2] Y. Aoyagi and T. Asakura, “A study on traffic sign recognition in scene image using genetic algorithms and neural networks,” in Proc. IEEE IECON 22nd Int. Industrial Electronics, Control, and Instrumentation Conf. , vol. 3, 1996, pp. 1838–1843.

[3] T. Warsop and S. Singh, “Distance-invariant sign detection in high- definition video,” in Proc. IEEE 9 th Int. Cybernetic Intelligent Systems (CIS) Conference , 2010, pp. 1–6.

[4] G. Piccioli, E. de Micheli, P. Parodi, and M. Campani, “Robust method for road sign detection and recognition,” Image and Vision Computing , vol. 14, no. 3, pp. 209–223.

[5] P. Viola and M. J. Jones, “Robust real-time face detection,” International Journal of Computer Vision , vol. 57, no. 2, pp. 137–154, 2004.

[6] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision , vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. van Gool, “Surf: Speeded up robust Features,” Computer Vision and Image Understanding , vol. 110, no. 3, pp. 346–359, 20