TRAFFIC SIGN DETECTION AND RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK (CNN)

A

MAJOR PROJECT REPORT

Submitted for the partial fulfilment of the requirement for the award of Degree

B.Tech.

COMPUTER SCIENCE & ENGINEERING

Submitted By:

Parul Taley - 0101CS191076

Sakshi Badole -0101CS191103

Guided By:

Dr. Sanjay Silakari

Dr. Shikha Agrawal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY INSTITUTE OF TECHNOLOGY

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA

BHOPAL-462033

SESSION 2019-2023

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that **Parul Taley, Sakshi Badole** of B.Tech. Fourth Year, Computer Science & Engineering have completed their Major Project entitled "**Traffic Sign Detection and Recognition using Convolutional Neural Network (CNN)**" during the year 2022-2023 under our guidance and supervision.

We approve the project for the submission for the partial fulfillment of the requirement for the award of degree of B.Tech. in Computer Science & Engineering.

Dr. Sanjay Silakari
Professor
Project Guide
DoCSE - UIT RGPV, Bhopal

Dr. Shikha Agrawal
Associate Professor
Project Guide
DoCSE - UIT RGPV, Bhopal

Prof. Uday Chourasia
HOD DoCSE
UIT RGPV, Bhopal

DECLARATION BY CANDIDATE

We, hereby declare that the work which is presented in the major project, entitled "**Traffic Sign Detection and Recognition using Convolutional Neural Network(CNN)**" submitted in partial fulfillment of the requirement for the award of Bachelor degree in Computer Science and Engineering has been carried out at University Institute of Technology RGPV , Bhopal and is an authentic record of our work carried out under the guidance of **Dr. Sanjay Silakari** (Project Guide) and **Dr. Shikha Agrawal** (Project Guide) Department of Computer Science and Engineering, UIT RGPV, Bhopal.

The matter in this project has not been submitted by us for the award of any other degree.

Parul Taley - 0101CS191076

Sakshi Badole - 0101CS191103

ACKNOWLEDGEMENT

We think if any of us honestly reflects on who we are, how we got here, what we think we might do well, and so forth, we discover a debt to others that spans written history. The work of some unknown person makes our lives easier every day. We believe it's appropriate to acknowledge all of these unknown persons; but it is also necessary to acknowledge those people we know have directly shaped our lives and our work.

First and foremost, we take this opportunity to express our deep regards and heartfelt gratitude to our project guide **Dr. Sanjay Silakari and Dr. Shikha Agrawal of Computer Science and Engineering Department, RGPV Bhopal** for their inspiring guidance and timely suggestions in carrying out our project successfully. They have also been a constant source of inspiration for us.

We are extremely thankful to **Mr. Uday Chourasia, Head, Computer Science and Engineering Department, RGPV Bhopal** for his cooperation and motivation during the project. We would also like to thank all the teachers of our department for providing invaluable support and motivation. We are also grateful to our friends and colleagues for their help and cooperation throughout this work.

Parul Taley - 0101CS191076

Sakshi Badole - 0101CS191103

Abstract

Since Traffic signs are an integral part of the road infrastructure. So, they play an important role in regulating and controlling the Traffic Management System. They provide critical and crucial information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behaviour and techniques in order to make sure they adhere with whatever road regulation currently enforced.

In adverse traffic conditions, the driver may not notice traffic signs, which may cause accidents. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs.

Among the other issues that also need to be addressed are partial obscuring, multiple traffic signs appearing at a single time, and blurring and fading of traffic signs, which can also create problem for the detection purpose

The data to be analysed here will be collected from the dataset of Traffic Signs with different visual appearance in order to recognize and detect that particular sign in any possible condition.

It is a challenge to categorize all kinds of images for a particular Traffic Sign that are entered and being present for the future analysis on the neural network. For this purpose, we used tkinter, python modules (pandas, numpy, matplotlib), tensorflow, keras, jupyter notebook.

TABLE OF CONTENTS

CHAPTER 4

CHAPTER 5

List of Figures

CHAPTER 1

INTRODUCTION

Project Title: Traffic Sign Detection and Recognition Using CNN

1.1 Project Definition:

It is a Deep Learning project in which we will build a model for the classification of the Traffic Signs detection and recognition using CNN (Convolutional Neural Network) and with the use of Keras library, Tensorflow library. The GUI of the model is based on the Tkinter GUI package. From the dataset of the Traffic signs, the particular traffic sign is detected and recognized in order to make
sure they adhere with whatever road regulation currently enforced.

Traffic sign boards have two major role to play during the travel of any vehicles, It assists the driver to the correct destination and indicates condition of the road. Traffic signs can be classified on the basis of their shape, size and color.

Traffic signs contain necessary messages about vehicle safety and they show the latest traffic conditions, define road rights, forbid and allow some behaviors and driving routes, cue dangerous messages and so on. They can also help drivers identify the condition of the road, so as to determine the driving routes.

Traffic sign detection system reminds and warns the driver about upcoming traffic signs coming in a way. These systems have the ability to detect traffic signs even in worst case scenarios, but still sometimes it is difficult to detect traffic signs in challenging conditions

1.2 Python

Python is a high-level, interpreted, general-purpose programming language. As well as its extensive collection of libraries, which are valuable for analytics and complex calculations.[2] Python's extensibility means that it has thousands of libraries dedicated to analytics. Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

Fig 1.1: Python logo[1]

1.3 NumPy

NumPy is a library for python programming language, adding support for large multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. In 2005, Travis Oliphnat created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications.

NumPy is open source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project. NumPy targets the CPython reference implementation of Python, which is a non optimizing bytecode interpreter. Mathematical algorithoms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

Fig 1.2: NumPy logo[2]

1.4 Pandas :

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language[7]. So in particular it offers the operations and data structures for manipulating the numerical data entries in the tables and time series.

It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. It offers features like reshaping and pivoting of data sets, data sets merging and joining,also provides data filtration, it provides slicing, indexing and subsetting of large data sets.

Fig 1.3: Pandas logo[3]

1.5 OpenCV:

OpenCV provides a real-time optimized Computer Vision library, tools, and hardware. It is an open source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems.

By using OpenCV, one can process images and videos in order to identify the objects, faces and even the handwriting of a human. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface.

Fig 1.4: OpenCV logo[4]

1.6 TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It is an end-to-end Machine learning platform. It offers tools that process and load data. It uses predefined ML models, if not, one can create the custom ones. It provides tools for deploying the models and also helps in the implementation of the models.

Fig 1.5: TensorFlow logo[5]

1.6.1 Keras:

Keras is the High-level API of the TensorFlow 2 as it acts as an interface for the TensorFlow library. It is an open-source software library that provides a Python interface for artificial neural networks. It is a deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. It reduces the cognitive load and offers consistent and simple API's.

Fig 1.6: Keras logo[6]

1.7 Matplotlib:

Matplotlib refers to Visualization with Python. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.
It offers features as:
Make interactive figures that can zoom, pan, update.
Customizes the visual style and layout.
Create publication quality plots.
Plant in JupyterLab and Graphical User Interfaces.

Fig 1.7: Matplotlib logo[7]

1.8 scikit-learn :

Scikit-learn is a machine learning library for the Python programming language. It is an open source software built on NumPy, SciPy, and matplotlib. It features various Classification, Regression, and Clustering algorithms that includes SVM (Support Vector Machines), random forest, k-Means, spectral clustering, mean-shift, nearest neighbors,

gradient boosting and many more such algorithms. These algorithms are used in the areas of Spam detection, Image recognition, Stock Prices, Customer segmentation, Grouping experiment outcomes etc.

Fig 1.8: scikit-learn logo[8]

1.10 Pyttsx3 :

pyttsx3 is a text-to-speech conversion library in Python. It's not like other alternative libraries, it works offline and is compatible with both Python 2 and 3. It works without internet connection or delay. An application invokes the pyttsx3.init() factory function to get a reference to pyttsx3. It is a very easy to use tool which converts the entered text into speech. It includes Sapi5, nsss, and espeak. The pyttsx3 module supports two voices: the first is female and the second is male which is provided by "sapi5" for windows.

1.11 Tkinter :

Tkinter is a Python binding to the Tk GUI toolkit. "The tkinter package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems."[3]. It's a fast and easy-to-use Python GUI library, making it the go-to library for building a Python GUI application.

Fig 1.9: Tkinter logo[9]

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System Approach:

Traffic signs are an integral part of our road infrastructure. They provide critical information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behavior to make sure they adhere with whatever road regulation currently enforced. In adverse traffic conditions, the driver may not notice traffic signs, which may cause accidents. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs.

Among the other issues that also need to be addressed are partial obscuring, multiple traffic signs appearing at a single time, and blurring and fading of traffic signs, which can also create problem for the detection purpose.

In today's world, identification of traffic signs has become an important aspect of our lives. Looking at the increasing traffic, to ensure safety of all and for automatic driving in the future, traffic sign classification is utmost necessary. Considerable research has been done around recognition of traffic and road signs. In 1987, the first research on the topic "Traffic Sign Recognition" was done by Akatsuka and Imai , where they tried to build a fundamental system that could recognize traffic signs and alert the drivers and ensure his/her safety. But this was used to provide the automatic recognition for only some specific traffic signs.

Traffic sign recognition initially appeared in the form of only speed limit recognition in 2008. These symbols could only detect the circular speed limit signs. On the other hand, later, systems were designed that performed detection on overtaking signs. This technology was available in the Volkswagen Phaeton and in the 2012 in Volvo S80, V70 and many more. But the major drawback of these systems was that they could not detect the city limit signs as they were mostly in the form of direction signs. But nowadays,

such systems are expected to be present in the future cars to help

drivers while driving.

In [9], the authors used the colour processing system to reduce the effect of brightness and shadow on the images. This was the very first research done on this topic by the authors, Akatsuka and Imai. In [10], the authors have done a survey on traffic sign detection and recognition, where HOG (Histogram of Oriented Gradients) is used for classification purpose. In [11], a complete study of different traffic sign recognition algorithms has been done, where the highest accuracy (99.46%) was obtained by MCDNN (Multi-column Deep Neural Network). In [12], the authors have developed a model where they are converting the images to gray scale first and then filter those images using simplified Gabor wavelets. The gabor filters are used to extract features. These wavelets are important as they help to minimize the product of its standard deviation in both the time and frequency mapping. The authors extracted the regions of interest for recognition purpose and classified the signs using "Support Vector Machine" (SVM).

In [13], the authors have extracted the regions of interest in the detection stage and further examined the shapes of such regions. Here, in the classification system, they took the regions of interest and classified them into different classes.

In [14], the authors have created a module which consists of numerous convolutions. They combined the 1*3 kernel and 3*1 kernel and finally linked it with the 1*1 kernel to attain the 3*3 kernel. This was used to extract more features and thus reduce the number of parameters. In [15], the author has reviewed the traffic sign detection methods and divided them into 3 types of methods: color, shape and learning based methods. In [16], the author used the number of peaks algorithm to detect and recognize circular shaped traffic signs.

In [17], the authors have tried creating a classification model using the Enhanced LeNet-5 architecture, which consists of two consecutive convolution layers (before the MaxPooling layer) to extract high level features from the image. Also, they have used the data augmentation technique to make the dataset stable. In [18], the authors have used

the technique of colour segmentation and the RGB based detection which is used to identify the traffic signs on the road. The optimizer used was "Stochastic Gradient Descent" with Nesterov Momentum. The text to speech system was implemented to alert the driver about the traffic sign. Also, they utilized the GPU (graphical

processing unit) system, as part of hardware. In [19], the authors have tried generating a dataset for the Arabic Road signs and thus develop a CNN model for Arabic sign recognition.

2.2 Proposed Approach:

The existing system approach is useful only when the driver is clearly and accurately able to see the traffic signs in order to detect and recognize them to adhere with whatever road regulation currently enforced.

But there are times when the driver is unable to recognize and classify the sign either due to environmental conditions or due to his/her own problem. So, this will lead to accidents which in turn may cause a higher amount of cost of destruction of property, human and animal casualties. Our Proposed Approach for this system is to minimize or avoid such situations to minimize the overall destruction of property and human and animal life.

In our approach to building the traffic sign classification model we will design and execute models on the basis of following technologies, which consists of :

1) Logistic Regression
2) Convolutional Neural Network

These models designing and implementation have some stages, that are :

1) Explore the dataset (of traffic signs)
2) (i) Build a Logistic Regression Model
   (ii) Build a CNN Model
3) Train and validate the model

4) Test the model with test dataset

The designing and implementation of the above to models basically leads to the comparison of the working of these models on the basis of accuracy of execution of the model to a given particular input. These models evaluate the accuracy of recognising and classifying the Traffic Signs given as input in real time or from the pre-processed dataset.

Accuracy of the models will be computed on the basis of the Confusion Matrix, that is, say C is a two-dimensional matrix with i , j parameters, such that C[i,j] is equal to the number of observations known to be in group i and predicted to be in group j.

CHAPTER 3

PROBLEM STATEMENT

Since Traffic signs are an integral part of the road infrastructure. So, they play an important role in regulating and controlling the Traffic Management System. They provide critical and crucial information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behavior and techniques in order to make sure they adhere with whatever road regulation currently enforced.

In adverse traffic conditions, the driver may not notice traffic signs, which may cause accidents. In our modern age, around 1.3M people die on roads each year. This number would be much higher without our road signs.

Among the other issues that also need to be addressed are partial obscuring, multiple traffic signs appearing at a single time, and blurring and fading of traffic signs, which can also create problem for the detection purpose

The data to be analyzed here will be collected from the dataset of Traffic Signs with different visual appearance in order to recognize and detect that particular sign in any possible condition.

It is a challenge to categorize all kinds of images for a particular Traffic Sign that are entered and being present for the future analysis on the neural network. For this purpose, we used tkinter, python modules (pandas, numpy, matplotlib), tensorflow, keras, jupyter notebook.

CHAPTER 4
PROPOSED WORK

4.1 Overview:

The existing system approach is useful only when the driver is clearly and accurately able to see the traffic signs in order to detect and recognize them to adhere with whatever road regulation currently enforced.

But there are times when the driver is unable to recognize and classify the sign either due to environmental conditions or due to his/her own problem. So, this will lead to accidents which in turn may cause a higher amount of cost of destruction of property, human and animal casualties. Our Proposed Approach for this system is to minimize or avoid such situations to minimize the overall destruction of property and human and animal life.

In our approach to building the traffic sign classification model we will design and execute models on the basis of following technologies, which consists of :

1) Convolutional Neural Network
2) Logistic Regression

These models' designing and implementation have some stages, that are:

1) Explore the dataset (of traffic signs)
2) (i) Build a Logistic Regression Model
   (ii) Build a CNN Model
3) Train and validate the model
4) Test the model with test dataset

4.2 Planning

1) The Plan is to bring forward a traffic sign recognition technique on the strength of deep

learning, which mainly aims at the detection and classification of circular signs.
2) The detected road traffic signs are classified based on deep learning.

3) A traffic sign detection and identification method on account of the image processing is proposed, which is combined with convolutional neural network (CNN) and logistic regression to sort traffic signs.
4) TensorFlow is used to implement CNN and Logistic Regression.

4.3 Methodology

The algorithm has three main stages:
1) Preprocessing: The fundamental function of the preprocessing stage here is to remove noise, enhance the input image.
2) Detection: The main task in this module is to extract the Region of Interests (ROI) from the images and to make them ready for the recognition stage.
3) Recognition and Classification: The classification stage takes the output from previous detection stage as its input. It then performs classification and gives the output with the sign type.

4.4 Classification using CNN Model

4.4.1 Dataset

Colours are made up of 3 primary colours; red, green, and blue. In computers, we define each colour value within a range of 0 to 255. So, there are approximately 16.5 million different ways to represent a colour. In our dataset, we need to map each colour's values with their corresponding names. Hence, we will be using a dataset that contains RGB values with their corresponding names. In our case we chose the GTSDB (German Traffic Sign Detection Benchmark) dataset for training and testing of the model.

4.4.2 Pre-processing of Dataset

Preprocessing of data refers to all the transformations on the raw data gathered before it is fed to the machine learning or any deep learning algorithm. It increases the speed of the training over training data. After preprocessing, the Normalisation process is used to adjust the values of features.

4.4.3 Splitting of Dataset

The dataset is being divided into the train and test samples according to the need. Common issues that are addressed by the splitting are overfitting and underfitting. In certain conditions creation of the extra set called validation set is also possible, it is basically required when apart from model performance we also need to choose among many models and evaluate which model performs better.

4.4.4 CNN Model Definition

In deep learning, a Convolutional Neural Network is a class of deep neural that specializes in processing data that has a grid-like topology, such as image. It consists of Convolutional layers, g layer, and fully connected layers. Convolutional layers are defined as the hidden layers in CNN model. CNN Model definition is followed by the Compilation of Model. Evaluation of the model also takes place.

4.4.5 Accuracy of Model

After the model definition, the model is run against the dataset for the recognition and classification of Traffic Signs and the accuracy score of the model is calculated. Graphs for the accuracy of the model are also plotted against each epoch. Graphs for data loss against each epoch are also implemented. The result of classification can also be visualized in the form of table layout that plots between all predicted and actual values of a classifier, known as Consufion Matrix.

4.5 Classification using Logistic Regression

4.5.1 Dataset

The process of choosing and defining the dataset is done, followed by the loading of the dataset. In our case we have chosen the GTSDB (German Traffic Sign Detection Benchmark) dataset for training and testing of the model.

Fig 4.1: Traffic signs classification containing 43 classes

4.5.2 Pre-processing of Dataset

Firstly this includes flattening and reshaping of the dataset. Flattening is done to convert the dataset the two-dimensional array or a multidimensional array into a contiguous flattened array. Preprocessing of the dataset takes place. Data is explored nd visualised. Standardization of Dataset is a common requirement for machine learning estimators because they might behave inappropriately if the individual features do not look like standard normally distributed data.

4.5.3 Splitting of Dataset

The Dataset is splitted into a training set and a testing set. In certain conditions a validation set is also introduced which compares the performance of the model among other defined models. The model is trained and tested against the training set and testing set respectively.

4.5.3 Logistic Regression Model Definition

The Logistic Regression Model can be defined with the help of several packages available in Python. In order to define a Logistic Regression Model, needed packages will be NumPy, scikit-learn, matplotlib, tensorflow and OpenCV packages. If one need a functionality that scikit-learn can’t offer, then StatsModel might be useful as it’s a powerful Python library for statistical analysis. Model fitting process will be initiated in which it will train the model by slicing the data into batches of a particular size, and repeatedly iterating over the entire dataset for a given number of epochs.

4.5.4 Accuracy of Model

The model is executed over the dataset to recognise and classify Traffic Signs, training dataset trains the model and test dataset tests the model. Accuracy score is calculated on the basis of the outcomes or result of the model. Confusion matrix is computed to evaluate the classification.

4.6 Workflow Diagram

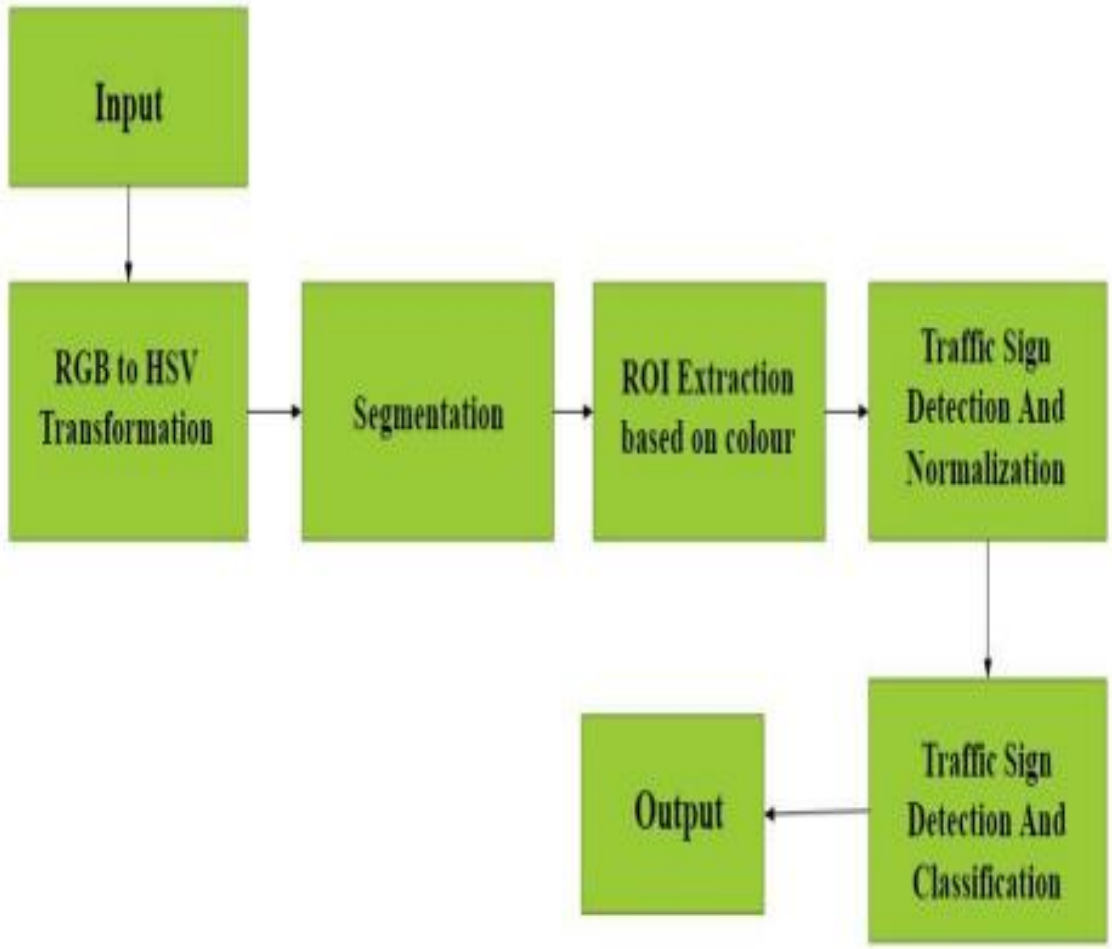

Fig 4.2: Workflow Diagram

4.7 Flowchart

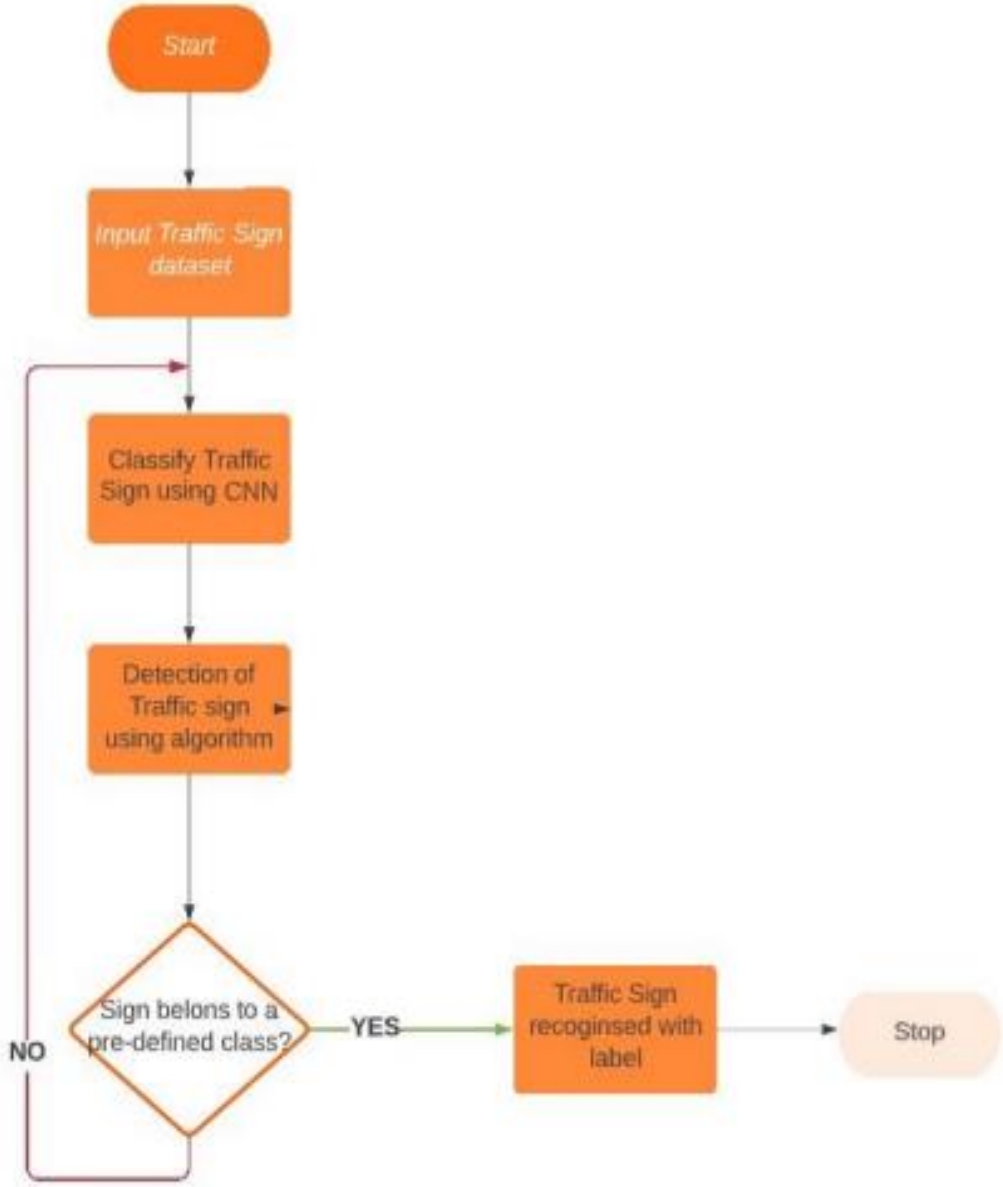

Fig 4.3: Flow Chart

4.8 Use Case Diagram

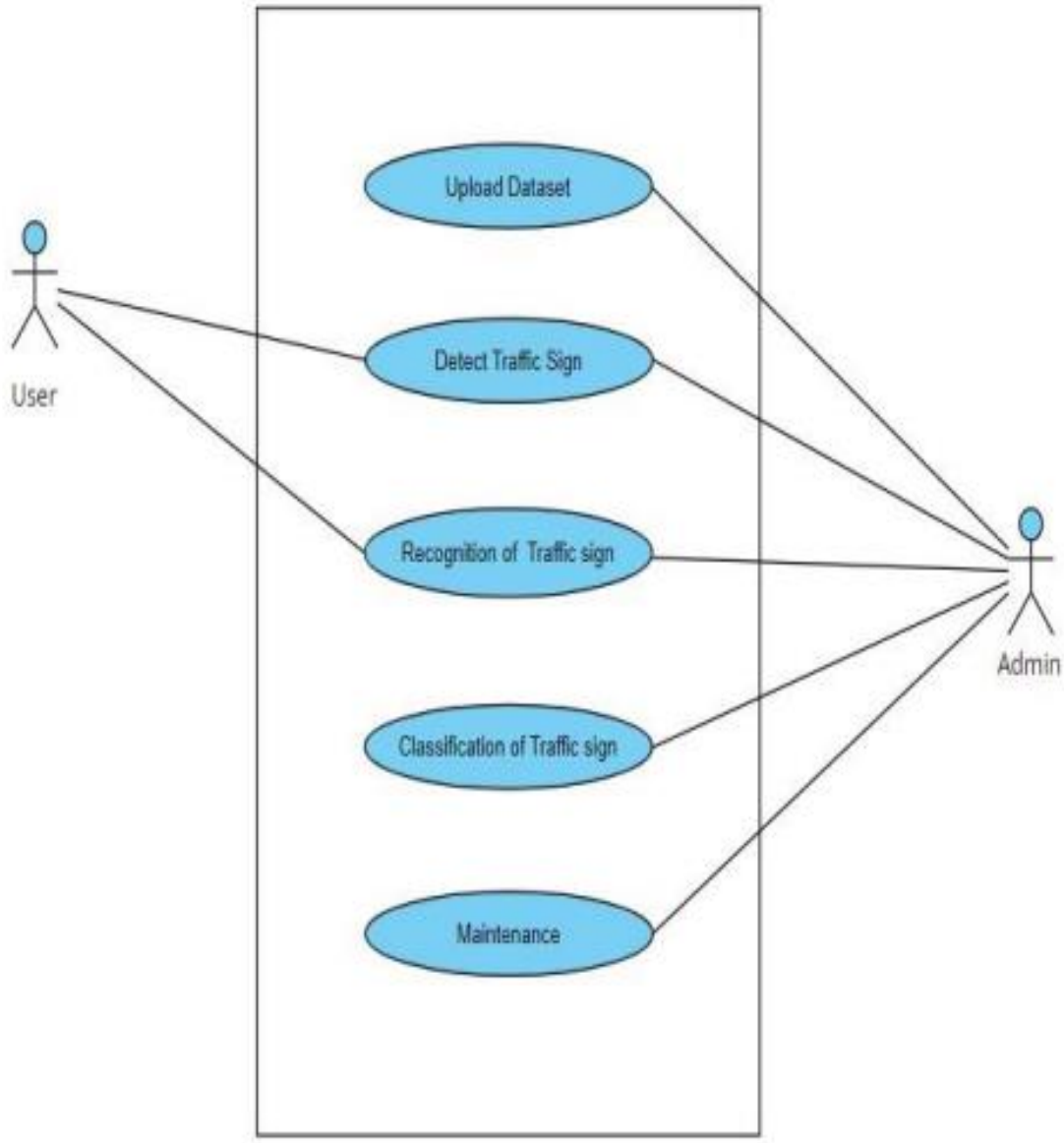

Fig 4.4: Use Case Diagram

CHAPTER 5

IMPLEMENTATION

5.1 Implementation of the CNN Model

The implementation of Convolutional Neural Network model for the Traffic Sign Recognition and Classification is stated in the following steps:

1) Dataset loading and reshaping

```
# loading dataset
data = []
labels = []
classes = 43
cur_path = os.getcwd()

for i in os.listdir(cur_path):
    dir = cur_path + '/' + i
    for j in os.listdir(dir):
        img_path = dir+'/'+j
        img = cv2.imread(img_path,-1)
        img = cv2.resize(img, (30,30), interpolation = cv2.INTER_NEAREST)
        data.append(img)
        labels.append(i)

data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

```
test_file = os.getcwd()
test_data = []
test_labels = []
classes = 43
for i in os.listdir(test_file):
    print(i)
    img_path = test_file+'/'+i
    print(img_path)
    img = cv2.imread(img_path,-1)
    img = cv2.resize(img, (30,30), interpolation = cv2.INTER_NEAREST)
    test_data.append(img)
    test_labels.append(i)

test_data = np.array(test_data)
test_labels = np.array(test_labels)
```

Fig 5.1: 43 classes of the traffic signs.

2) Splitting dataset into Training set and Test set , Splitting training set into validation set

```
    X_train, y_train = data, labels
    X_test, y_test = test_data, test_labels

    print("Image Shape: {}".format(X_train[0].shape))
    print(y_train.shape)
    print(y_test.shape)
    print("Training Set:   {} samples".format(len(X_train)))
    print("Test Set:       {} samples".format(len(X_test)))
```

```
Image Shape: (30, 30, 3)
(39209,)
(12630,)
Training Set:   39209 samples
Test Set:       12630 samples
```

```
print(X_train.shape, labels.shape)
#Splitting training and validation dataset

X_train, X_valid, y_train, y_valid = train_test_split(X_train, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_valid.shape, y_train.shape, y_valid.shape)
```

```
(39209, 30, 30, 1) (39209,)
(31367, 30, 30, 1) (7842, 30, 30, 1) (31367,) (7842,)
```

3) RGB to GrayScale Transformation - GaryScaling benefits in Dimension reduction, reduces model complexity, and for other algorithms to work as many algorithms are customized to work only on grayscale images.

```
from numpy import newaxis

def gray_maker(data):
    gray_images = []
    for image in data:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        gray_images.append(gray)
    return np.array(gray_images)

def preprocess(data,data_name,verbose):
    if verbose:
        print('Preprocessing '+ data_name + '...')

    # Iterate through grayscale
    data = gray_maker(data)
    data = data[..., newaxis]

    #Normalizes the data between 0.1 and 0.9 instead of 0 to 255
    data = data / 255 * 0.8 + 0.1

    if verbose:
        print('Finished preprocessing '+ data_name + '...')

    # Double-check that the image is changed to depth of 1
    image_shape = data.shape

    if verbose:
        print('Processed '+ data_name + ' shape =', image_shape)
        print(" ")
    return data
```

4) Preprocessing of data

```
X_train = preprocess(X_train,'train_data',verbose = True)

X_test = preprocess(X_test,'test_data',verbose = True)
```

```
Preprocessing train_data...
Finished preprocessing train_data...
Processed train_data shape = (39209, 30, 30, 1)

Preprocessing test_data...
Finished preprocessing test_data...
Processed test_data shape = (12630, 30, 30, 1)
```

5) Converting the labels into one hot encoding

```
#Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_valid = to_categorical(y_valid, 43)
```

```
y_train.shape, y_valid.shape
```

```
((31367, 43), (7842, 43))
```

6) CNN Model is initialized using two convolutional layers along with Pooling layers (we have used Max Pooling) and Dropout layers. The Fully connected layer consists of Flatten layer and Dense layer along with dense layers. Activation function is used to compute non-linear elements of the input, in this model we have used 'relu' activation function (Rectified Linear Function). It is mathematically represented as :

$$f(x) = \{ 0 , x < 0 ; x , x >=0 \}$$

OR

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

```
#CNN model
model = Sequential()


model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))


model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))


model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

7) CNN Model Summary

This shows the various hyperparameters and parameters of the model after constructing the model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        832

 conv2d_1 (Conv2D)           (None, 22, 22, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 11, 11, 32)       0
 )

 dropout (Dropout)           (None, 11, 11, 32)        0

 conv2d_2 (Conv2D)           (None, 9, 9, 64)          18496

 conv2d_3 (Conv2D)           (None, 7, 7, 64)          36928

 max_pooling2d_1 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 3, 3, 64)          0

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 256)               147712
...
Total params: 240,651
Trainable params: 240,651
Non-trainable params: 0
```

8) Compilation of Model for the execution

The model runs 20 epochs on the training data X_train and y_train, and the compiled model is saved in the file for further actions.

```
#Compilation of the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 20
with tf.device('/GPU:0'):
    history = model.fit(X_train, y_train, batch_size=160, epochs=epochs, validation_data=(X_valid, y_valid))
model.save("../../../MajorProject/traffic_classifier.h5")
```

9) Calculation of Accuracy of the Model

```
# Calculate the Accuracy
from sklearn.metrics import accuracy_score
score=accuracy_score(y_pred_class,y_valid_class)
score
```

```
0.9913287426676868
```

10) Confusion Matrix of CNN Model

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_valid_class, y_pred_class)
import seaborn as sns
plt.figure(figsize = (25,15))
sns.heatmap(cm,annot=True)
# plt.savefig('h1.png')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(283.22222222222223, 0.5, 'Truth')
```

11) Graphical User Interface to use CNN Model is based on the Tkinter

```
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')

label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)

def gray_maker(data):···

def preprocess(data):···

def classify(file_path):···

def show_classify_button(file_path):···

def realtime():···

def upload_image():···
```

12) Using Tkinter based GUI, users can upload the image in real-time and using pyttsx3, the output will be in both speech and text form on the screen. First, the image will be processed by the CNN Model if it is a traffic sign lying in any of the classified classes then it will give the traffic sign name, in both form, speech as well as text, on the screen.

```
#dictionary to label all traffic signs class.
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Veh > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End no passing veh > 3.5 tons' }
```

5.1.1 Working of the CNN Model

This image shows the GUI of the CNN model where we can either upload an image or run the real time implementation of the model, where it captures the image from the camera and processes it and produces the result same way the upload an image option does.

Fig 5.2: GUI page created using Tkinter, Showing two options to test the CNN model to recognize a traffic sign image.

After an image is uploaded or captured in real time it is displayed on the screen and the option of classify image comes in the upload an image option and it directly proceed to process it in the real time option.

Fig 5.3: GUI page showing the traffic sign captured/uploaded.

The uploaded/captured traffic sign is then processed when clicked on the classify image button and its class is predicted and the result is displayed on the screen with the audio dictating the result.

Fig 5.4: GUI page predicting the traffic sign classification.

5.2 Implementation of the Logistic Regression Model

Logistic Regression is a Machine Learning classification algorithm, which is used to predict the probability of categorical dependent variables; unlike Linear Regression it predicts a continuous outcome. The implementation of Convolutional Neural Network model for the Traffic Sign Recognition and Classification is stated in the following steps:

1) Dataset loading

```
# Loading dataset
data = []
labels = []
classes = 43
cur_path = os.getcwd()

for i in os.listdir(cur_path):
    dir = cur_path + '/' + i
    for j in os.listdir(dir):
        img_path = dir+'/'+j
        img = cv2.imread(img_path,-1)
        img = cv2.resize(img, (30,30), interpolation = cv2.INTER_NEAREST)
        data.append(img)
        labels.append(i)

data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

```
array([[[ 80,  78,  75],
        [ 80,  78,  75],
        [ 78,  76,  74],
        ...,
        [ 75,  75,  68],
        [ 68,  69,  65],
        [ 66,  67,  66]],

       [[ 86,  84,  83],
        [ 86,  84,  83],
        [ 82,  80,  80],
        ...,
        [ 78,  77,  73],
        [ 75,  78,  76],
        [ 78,  80,  80]],

       [[ 80,  78,  78],
        [ 80,  78,  78],
        [ 86,  85,  86],
        ...,
        [ 72,  74,  72],
        [ 69,  74,  73],
        [ 74,  78,  78]],

       ...,
...
        [107, 101,  95],
        ...,
        [ 95, 102, 102],
        [ 90, 102,  99],
        [ 89,  97,  90]]], dtype=uint8)
```

2) Flattening and reshaping of dataset

```
    data_flatten = data.reshape(data.shape[0],-1)
```

```
    data_flatten[0]
```

```
array([80, 78, 75, ..., 89, 97, 90], dtype=uint8)
```

```
    labels.shape
```

```
(39209,)
```

```
    labels
```

```
array(['0', '0', '0', ..., '9', '9', '9'], dtype='<U2')
```

3) Splitting of dataset into training set and test set

```
    X_train, X_test, y_train, y_test = train_test_split(data_flatten,labels, test_size=0.2)
    print(y_train.shape)
    print(y_test.shape)
```

```
(31367,)
(7842,)
```

```
    print(y_train.shape)
    print(y_test.shape)
    print(X_train.shape)
    print(X_test.shape)
```

```
(31367,)
(7842,)
(31367, 2700)
(7842, 2700)
```

4) Preprocessing of data - Includes standardization of data using StandardScalar()

```
from sklearn import preprocessing
scaler =  preprocessing.StandardScaler().fit(X_train)
scaler_test =  preprocessing.StandardScaler().fit(X_test)
print(scaler)
```

```
StandardScaler()
```

```
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler_test.transform(X_test)
print(X_train_scaled)
print(X_test_scaled)
```

```
[[-0.69394539 -0.70830029 -0.74446502 ... -0.54106846 -0.53624697
  -0.53163433]
 [-0.83469851 -0.90290558 -0.95212913 ... -0.85214066 -0.89186084
  -0.89807647]
 [-0.22050305 -0.55261606 -0.79638105 ... -0.69660456 -0.72178464
  -0.74539225]
 ...
 [-0.89867721 -0.967774   -0.99106615 ... -0.52551485 -0.64447728
  -0.73012383]
 [-0.7963113  -0.82506346 -0.86127608 ... -0.77437261 -0.83001495
  -0.83700278]
 [-0.75792408 -0.39693183 -0.03061965 ...  2.32079571  2.55604756
   2.76634494]]
[[-0.59319282 -0.62083596 -0.70996104 ... -0.44733213 -0.4765204
  -0.56616306]
 [-0.15912833 -0.16773621 -0.30894673 ...  2.40320127  2.89756141
   2.8243767 ]
 [-0.31232756 -0.19362763 -0.29601079 ... -0.08906837  0.0342351
  -0.0316185 ]
 ...
 [ 0.38983559  0.0523408  -0.14077944 ... -0.63425235 -0.69320456
  -0.71889008]
 [ 1.71756228  1.17861731  0.85528836 ...  1.78013386  1.14861075
   0.73201658]
 [-0.59319282 -0.62083596 -0.5547297  ... -0.26041191 -0.13601673
   0.0141996 ]]
```

5) Logistic Regression Model is initialized using sklearn.linear_model and model

fit process is executed to execute the model

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='lbfgs', multi_class='multinomial', max_iter=1000, verbose=True)


model.fit(X_train_scaled, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial', verbose=True)
```

6) Logistic Regression Model score calculation

```
model.score(X_test_scaled, y_test)
```

```
0.9305024228513135
```

```
model.predict(data_flatten[0:5])
```

```
array(['0', '0', '0', '0', '0'], dtype='<U2')
```

```
y_predicted = model.predict(X_test_scaled)
```

7) Confusion Matrix for the Logistic Regression Model

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
cm
```

```
array([[ 32,   3,   0, ...,   0,   0,   0],
       [  4, 378,   0, ...,   2,   5,   0],
       [  0,   0, 381, ...,   0,   1,   1],
       ...,
       [  0,   2,   1, ..., 271,   8,   0],
       [  0,   1,   0, ...,   9, 256,   0],
       [  0,   1,   9, ...,   1,   0, 285]], dtype=int64)
```

```
import seaborn as sn
plt.figure(figsize = (25,15))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(283.22222222222223, 0.5, 'Truth')
```

CHAPTER 6

RESULT & ANALYSIS

6.1 Result of Convolutional Neural Network (CNN) Model

The CNN model successfully predicts the classes of the traffic signs being fed to it. This can be either via uploading and image or by the real time capture of the image from the camera. It dictates the name of the sign and produces the output on the screen.

Fig 6.1: GUI page predicting the traffic sign classification.

The comparison between the accuracy of the two models, i.e., Convolutional Neural Network and Logistic Regression, have been made where the results are as follows :

6.1.1 Accuracy of CNN Model

On implementation of the CNN Model on dataset which consists of train_set, test_set, and valid_set:

```
    # Score
    score = model.evaluate(X_valid, y_valid, verbose=0)
    print('Test Loss', score[0])
    print('Test accuracy', score[1])
```

```
Test Loss 0.027854114770889282
Test accuracy 0.9913287162780762
```

```
    y_pred = model.predict(X_valid)
    y_valid_class = np.argmax(y_valid,axis=1)
    y_pred_class = np.argmax(y_pred,axis=1)
```

```
246/246 [==============================] - 3s 11ms/step
```

```
    # Calculate the Accuracy
    from sklearn.metrics import accuracy_score
    score=accuracy_score(y_pred_class,y_valid_class)
    score
```

```
0.9913287426676868
```

(a) Plotting Accuracy graph using matplotlib

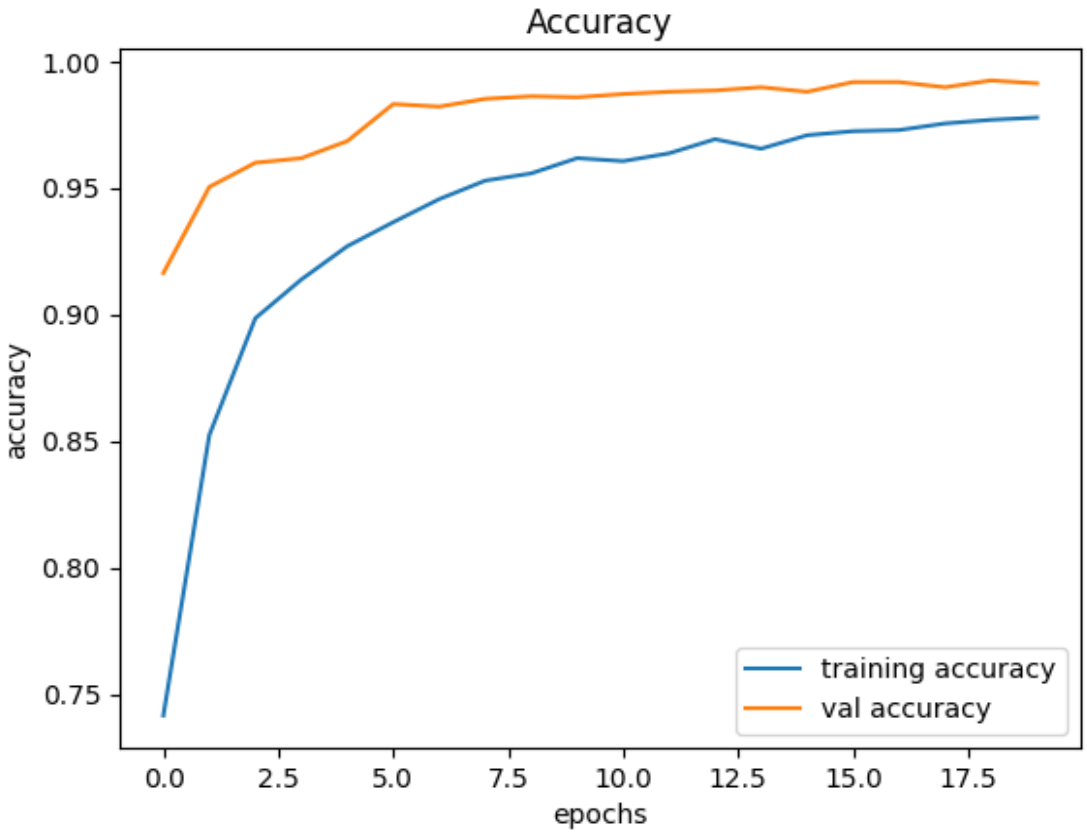

Fig 6.2: Graph showing the accuracy vs epochs in training and validation dataset.

```
#plotting graphs for accuracy
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

(b) Plotting Loss graph using matplotlib

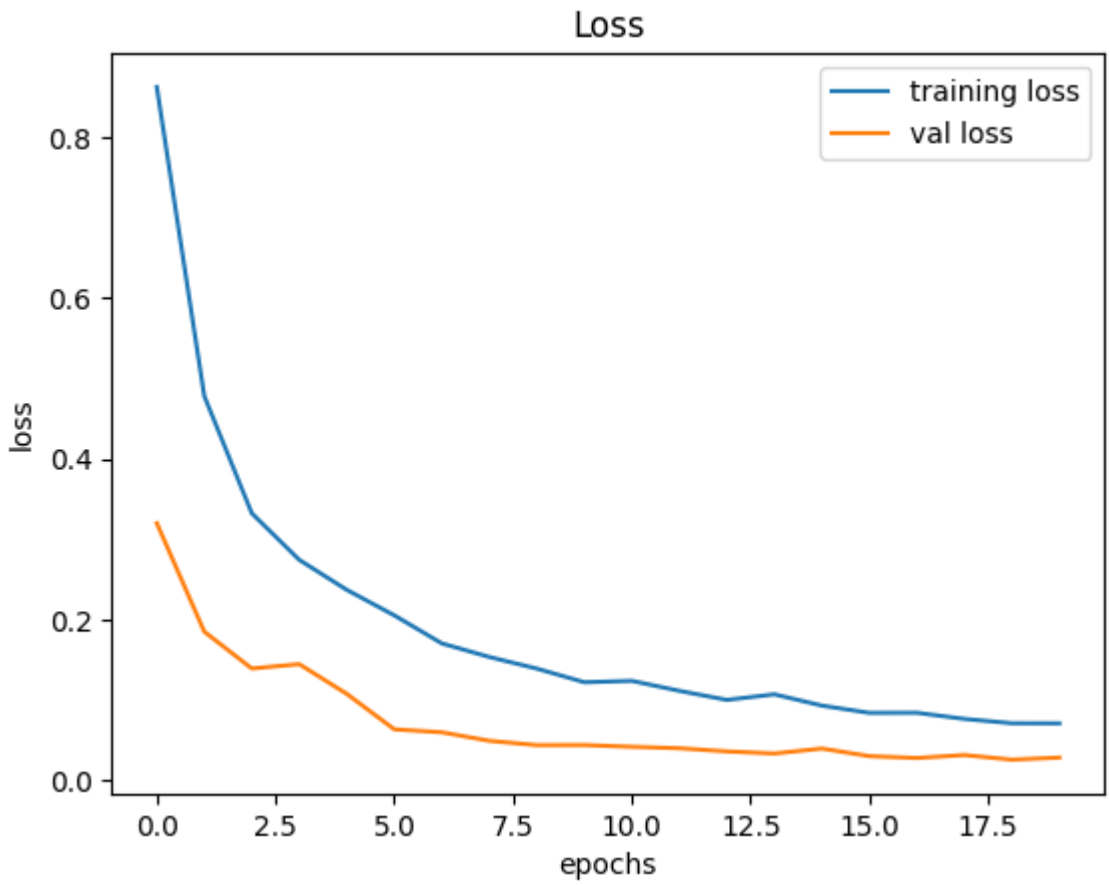

Fig 6.3: Graph showing the loss vs epochs in training and validation dataset.

```
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

6.1.2 Confusion Matrix graph of CNN Model

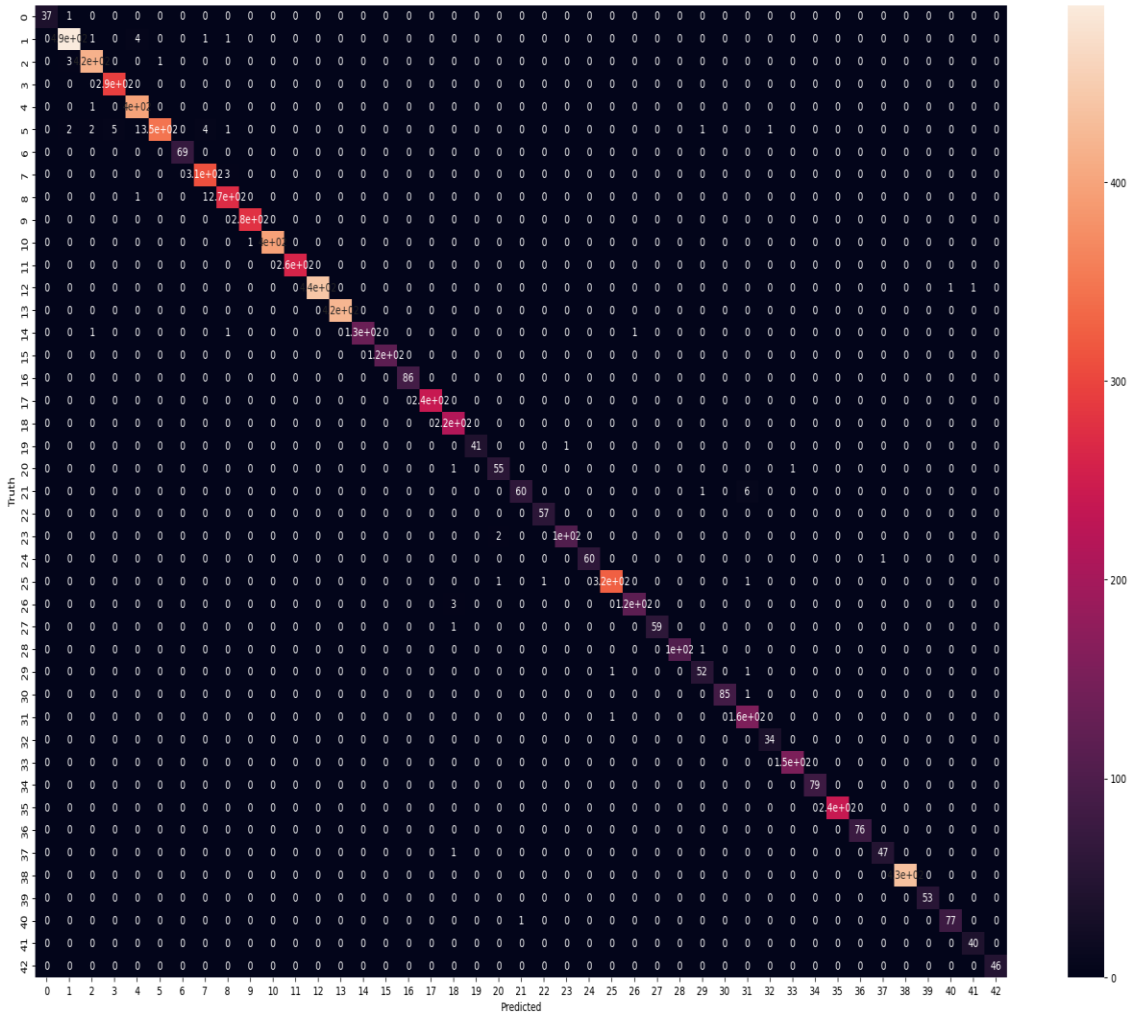

Fig 6.4: Confusion matrix for CNN model.

6.2 Result of Logistic Regression Model

After the implementation of the Logistic Regression Model, results obtained are:

6.2.1 Accuracy of Logistic Regression Model

On implementation of the Logistic Regression Model on dataset which consists of train_set, test_set, and valid_set:

```
model.score(X_test_scaled, y_test)
```

```
0.9305024228513135
```

```
model.predict(data_flatten[0:5])
```

```
array(['0', '0', '0', '0', '0'], dtype='<U2')
```

```
y_predicted = model.predict(X_test_scaled)
```

6.2.2 Confusion Matrix graph of Logistic Regression Model

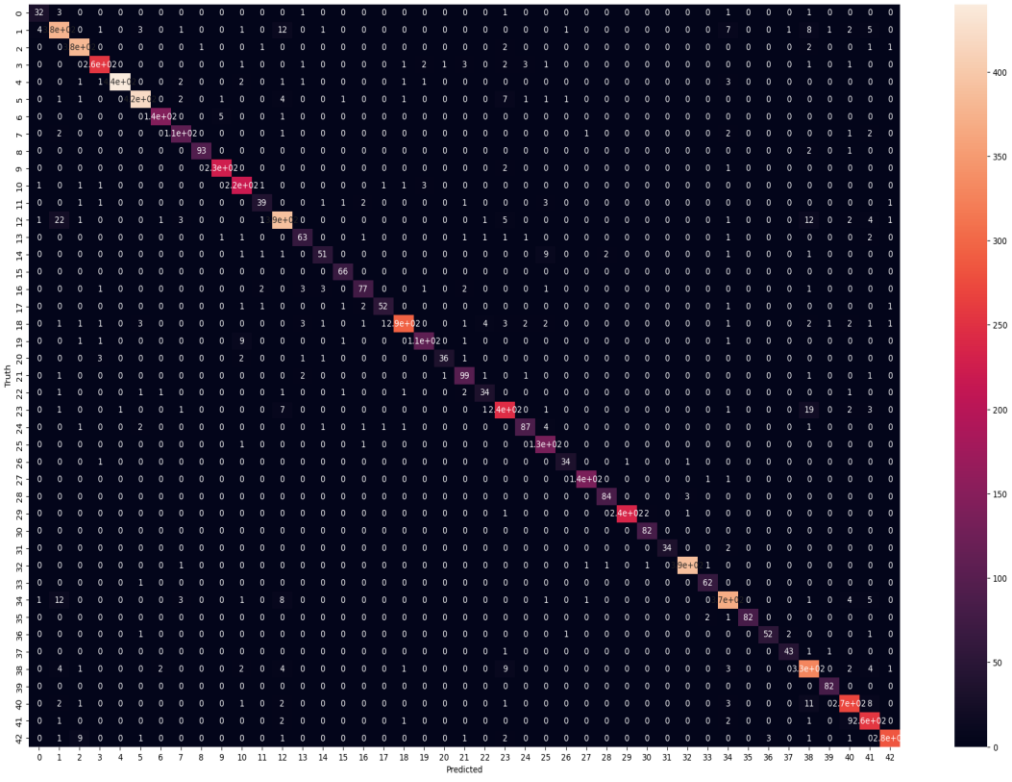

Fig 6.5: Confusion matrix for logistic regression model.

6.3 Analysis of Models

On analyzing the accuracy of both, Convolutional Neural Network (CNN) and Logistic Regression Models, it concludes that the Accuracy of Convolutional Neural Network (CNN) Model is better than that of Logistic Regression Model.

```
# Calculate the Accuracy of CNN Model
from sklearn.metrics import accuracy_score
score=accuracy_score(y_pred_class,y_valid_class)
score
```

```
0.9913287426676868
```

```
#Calculate Accuracy of Logistic Regresssion Model
model.score(X_test_scaled, y_test)
```

```
0.9305024228513135
```

Accuracy of CNN Model = 0.9913287426676868

Accuracy of Logistic Regression Model = 0.9305024228513135

Accuracy (CNN) > Accuracy (Logistic Regression)

From this analysis we can conclude that the Convolutional Neural Network (CNN) Model is more efficient than the Logistic Regression Model to recognise and classify traffic signs.

CHAPTER 7

CONCLUSION & FUTURE WORK

7.1 Conclusion

Traffic Sign Detection and Recognition Using Convolutional Neural Network detects the occluded sign boards and classifying them depending upon their structure, size, shapes and color and could play a vital role in the area of automotive industries as it becomes necessary in the future to assist the driver in identifying the occluded sign boards and warn them using the audio prediction of the traffic sign.

We have implemented the Traffic Sign Recognition and Classification

Classification of signals on the basis of structure, shape, size can be possible.

If the signals are not properly visible to driver due to some obstacles, then it will also use to identify blur and partially visible signal.

7.2 Future Scope

The future scope for this model is to be able to recognize the traffic sign in a matter of few seconds in real time traffic and warn the driver accordingly. For real time prediction it is necessary for training it on a dataset having the surroundings so that it will select the filters to gather the correct features of the traffic sign only on the feature map and then processes the image and predict its class.

With the development of Auto Focus Convolutional Neural Network it may be possible to capture the traffic sign from the environment and nullify all other objects. It is supposed to be implementable in vehicles for which driver requires special training to operate other functions. For example, Bulldozer.

CHAPTER 8

REFERENCES

[1] Python Software Foundation. https://www.python.org/community/logos/

[2] Numpy https://numpy.org/

[3] Pandas https://pandas.pydata.org/about/citing.html

[4] OpenCV https://opencv.org/

[5] TensorFlow https://www.tensorflow.org/

[6] Keras: the python deep learning API https://www.keras.io/

[7] Matplotlib https://www.matplotlib.org/stable/gallery/misc/logos2.html

[8] scikit-learn https://scikit-learn.org/stable/

[9] Tkinter https://www.docs.python.org/3/library/tkinter.html

[10] Akatsuka, H., & Imai, S. (1987). Road signposts recognition system (No. 870239). SAE Technical Paper.

[11] Albert Keerimolel, Sharifa Galsulkar, Brandon Gowray, "A SURVEY ON TRAFFIC SIGN RECOGNITION AND DETECTION", Xavier Institute of Engineering, Mumbai, India, International Journal of Trendy Research in Engineering and Technology Volume 7 Issue 2 April 2021.

[112 Aditya, A.M., & Moharir, S. (2016). Study of Traffic Sign Detection and Recognition Algorithms.

[13] Shao, F., Wang, X., Meng, F., Rui, T., Wang, D., & Tang, J. (2018). Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs. Sensors, 18(10), 3192.

[14] SADAT, S. O., PAL, V. K., & JASSAL, K. RECOGNIZATION OF TRAFFIC SIGN.

[15] Li W., Li D., & Zeng S. (2019, November). Traffic Sign Recognition with a small convolutional neural network. In IOP conference series: Materials science and engineering (Vol. 688, No. 4, p. 044034). IOP Publishing.

[16] Brkic, K. (2010). An overview of traffic sign detection methods. Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska, 3, 10000.

[17] Almustafa, K. M. (2014). Circular traffic signs recognition using the number of peaks algorithm. Int J Image Process (IJIP), 8(6), 514.

[18] Zaibi A., Ladgham A., & Sakly A. (2021). A Lightweight Model for Traffic Sign Classification

Based on Enhanced LeNet-5 Network. Journal of Sensors, 2021.

[19] G. Bharath Kumar, N. Anupama Rani, "TRAFFIC SIGN DETECTION USING CONVOLUTION NEURAL NETWORK A NOVEL DEEP LEARNING APPROACH", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Vol.8, Issue 5, May 2020.

[20] Alghmgham, D. A., Latif, G., Alghazo, J., & Alzubaidi, L. (2019). Autonomous traffic sign (ATSR) detection and recognition using deep CNN. Procedia Computer Science, 163, 266-274.