

TRAFFIC SIGN RECOGNITION AND DETECTION

CONTENTS

S.NO.	Page No.
CHAPTER 1 INTRODUCTION	1
1.1 Overview of the project	5
1.2 Objective	6
1.3 Problem Statement	6
1.5 Literature survey	7
1.6 Tools and Platform Used	8
CHAPTER 2 CONVOLUTIONAL NEURAL NETWORK	
2.1 Introduction	9
2.2 Convolutional Neural Network	10-13
2.2.1 Kernal	
2.2.2 Pooling	
2.2.3 Padding	
2.2.4 CNN	
CHAPTER 3 METHODOLOGY	
3.1 Introduction	14
3.2 Block diagram	14-18
3.2.1 Training	
3.2.2 Testing	
3.3 Flow of Implementation	19-20
CHAPTER 4 IMPLEMENTATION AND RESULTS	
4.1 Code	21-36
4.1.1 Training code	
4.1.2 Testing code	
4.1.3 Webpage code	
4.2 Results	37-39
CHAPTER 5 CONCLUSIONAND FUTURE WORK	40
REFERENCES	

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

In the race of technology we are witnessing many researches where the control of steering wheel of motor vehicles will be given not to another person but to an electronic brain. For the driver to be replaced the electronic brain must take control of all commands as well as pay distributive attention on traffic signs, road, and also traffic situations. Traffic sign recognition is one of the essential part of this automation. They contain critical information that ensures the safety of all the people around us. Without traffic signs, all the drivers would be clueless about what might be ahead to them and roads can become a mess.

In this world of advancement in technologies, the self-driving cars in which the passenger can fully depend on the car for traveling to achieve level 5 automation, it is necessary for vehicles to understand and follow all traffic rules. So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly. There are different traffic signs like speed limits, turn indications like U turn, right, left turns etc. Through this project, our attempt is to build an Artificial neural network model which will help in recognition and detection of traffic sign present in the given image.

1.2 OBJECTIVE

- To design an artificial convolution neural network model for recognition of different traffic signs.
- To develop the designed model using a testing dataset of traffic signs.
- To increase the performance of the developed model by using different approaches.
- To store the model with best accuracy.
- To develop a website which is loaded with the developed model to test the model on the real world images.

1.3 PROBLEM STATEMENT

According to statistics, many road accidents take place due to lack of instant response to traffic events in the vehicles. The same problem will be forwarded in the autonomous vehicles as well if there is no proper software model embedded to monitor and interpret the traffic signs and recommend the actions accordingly in the system. It is also essential that the recognition and interpretation of the traffic signs should be done in a minimum time period so that the according action can be taken within the time to avoid the further violation consequences.

Not only in the self-driving vehicles this model for traffic sign recognition is also useful in the normal vehicles too to interpret and indicate the signs to the driver which will be helpful in the proper road safety engaging of the vehicles. So, we would like to develop a model which will give accurate assistance to vehicle.

1.4 MOTIVATION

By studying various papers published previously on this concept we came to identify certain limitations such as false positives due to improper training the model.

And also due to change in weather conditions or uncomfortable climatic conditions there will be situations when it will be difficult to interpret the signal on the sign board in real world situations for self- driving automation systems. With the help of our survey and knowledge we would like to build a model which overcome the identified limitations.

1.5 LITERATURE SURVEY

S.No	Title of Paper	Publication Details	Authors	Limitations/Remarks
1	An automatic Traffic Sign Detection and Recognition system based on colour Segmentation, shape Matching and	Published on 17 November 2015 in hindawi.com	Safat B.wali,Mahammad A.Hannan,Anni Hussain and Salina A.Samad	The model developed based on colour and shape segmentation fails to give accurate predictions in varying weather conditions.
2	Smart Traffic sign detection on autonomous car	published on November 2018 in researchgate.net	Maria-Adelina Sirbu, Andrei Baiasu, Razvan Bagdan,Mihaela Crisan-Vida	The model build using RasperryPi and OpenCV leads to false positive predictions due to lack of proper training and presence of obstacles which leads to improper functionality of vehicle.

1.6 TOOLS USED AND PLATFORM USED

We have used the dataset of traffic sign images of 43 different classes from “Kaggle” a global Machine learning Platform. For implementation of model the programming language used is python of version 3.6 and tensorflow is used to train the model.

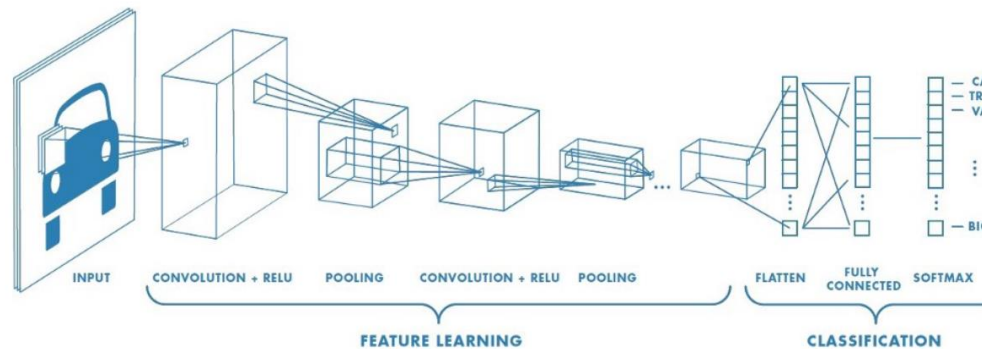
To develop the webpage for analysing the real world images html is used and visual studio code is used to develop the html , css and java script for the webpage to test the model. The platform used for developing the model using python language is “PYCHARM IDE”.

CHAPTER 2

CONVOLUTIONAL NEURAL NETWORK

2.1 INTRODUCTION

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a **Convolutional Neural Network**.



Artificial Intelligence has been witnessing a monumental growth in

2.2 WHAT IS CNN(Convolutional Neural Network)

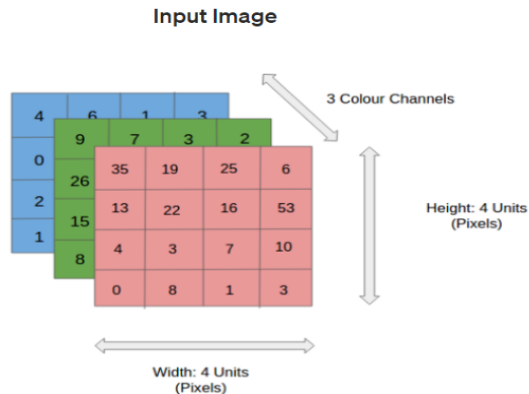
A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Traffic sign Recognition and Detection

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

INPUT IMAGE



In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

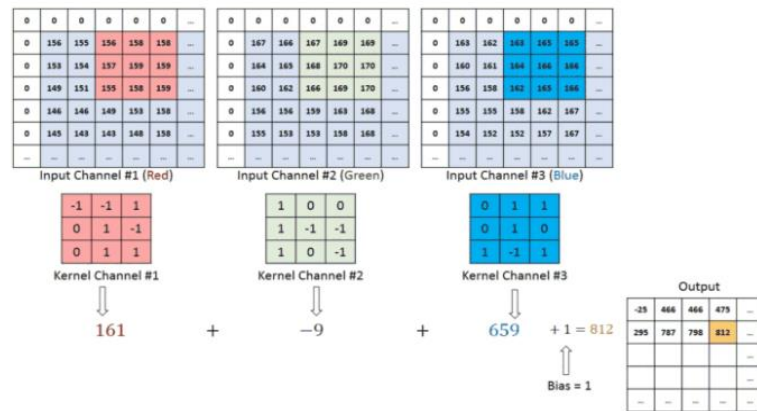
You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

KERNEL

The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter**,

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

Traffic sign Recognition and Detection



Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

STRIDE

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

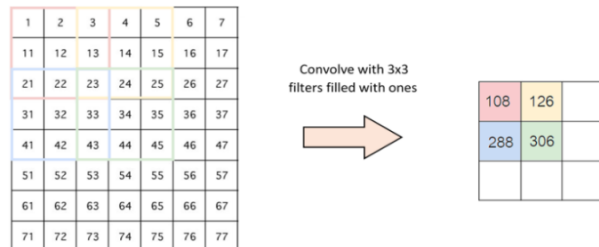


Figure 6 : Stride of 2 pixels

PADDING

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

RELU(NON-LINEARITY)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0, x)$. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

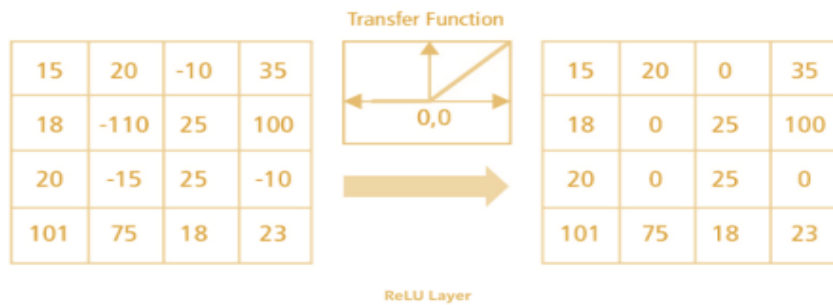


Figure 7 : ReLU operation

There are other non linear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two.

POOLING

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Traffic sign Recognition and Detection

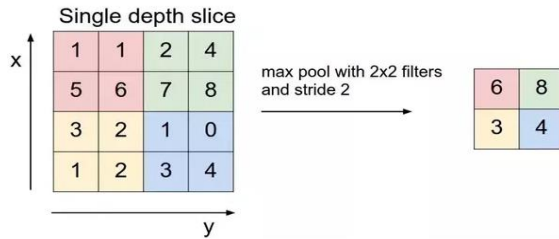


Figure 8 : Max Pooling

COMPLETE CNN ARCHITECTURE

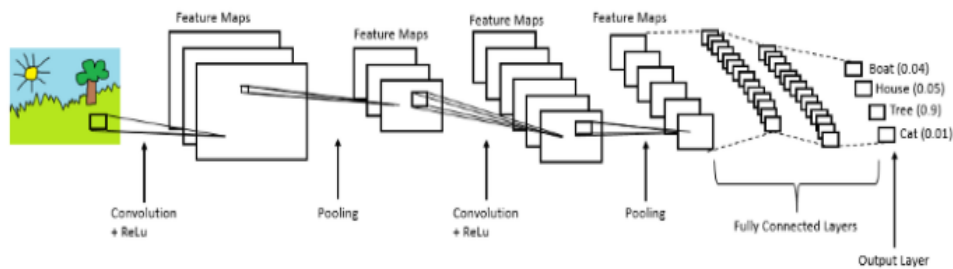


Figure 10 : Complete CNN architecture

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

CHAPTER-3

METHODOLOGY

3.1 INTRODUCTION

In this the methodology of the proposed system will be discussed.

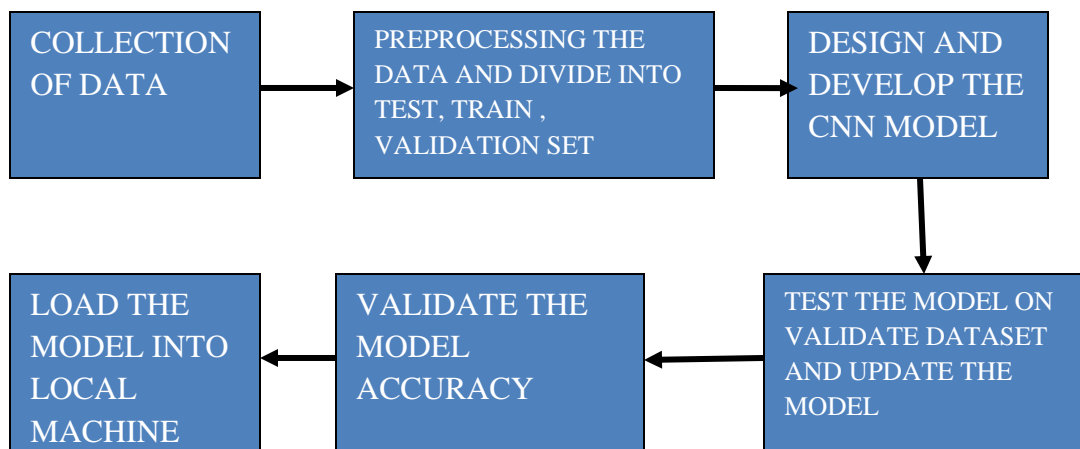
To build the model there are certain pre requisites. They are getting a proper dataset as per the requirement and then prepossessing the images of traffic signs in the dataset so that the model will be able to give the better results. The prepossessing includes forming the images so same dimensions and converting to grayscale and soon. After prepossing the contents of dataset, an appropriate convolution neural network is to be choosen. Now the model is implemented using python and tested for the accuracy using the validation set.

Now a testing code is written which takes some random traffic sign image and predict the sign. For testing the page we have designed a webpage in which we upload the image and get the result of the prediction with atmost accuracy.

3.2 BLOCK DIAGRAM

In any of the Artificial neural network system a model is trained initially using the dataset and the trained model is stored in binary format and this model is used to test or deploy in real world applications.

TRAINING A MODEL



Traffic sign Recognition and Detection

COLLECTION OF DATA

Data set consisting of appropriate images of traffic signs is to be collected and we have collected a data set which consists of images of 43 different classes of traffic signs

The 43 classes of traffic signs present in dataset are:

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 for vehicles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vehicles
16	Vehicles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right
25	Road work
26	Traffic signals

Traffic sign Recognition and Detection

27	Pedestrians
28	Children crossing
29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and 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 of no passing by vehicles over 3.5 metric tons

PREPROCESSING OF DATA

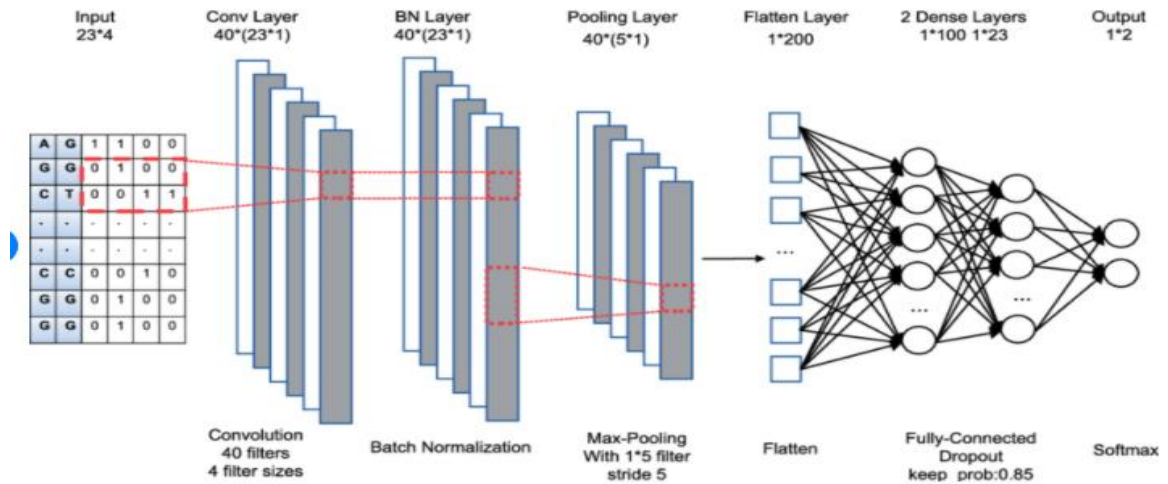
The collected dataset images are preprocessed i.e each of the image in the dataset is verified for compatibility of it with the convolutional model to be built, and also the images are converted to grayscale for better accuracy .

And also all the compatible images are splitted randomly into training, testing and validating sets.

DESIGNING AND IMPLEMENTING THE CNN MODEL

An appropriate convolution model with required number of hidden layers is designed and is trained with the training dataset that is splitted from the actual dataset of traffic signs. The weights of the convolution layers will be updated based on the training input images. The convolution layer will be as follows.

Traffic sign Recognition and Detection



Certain activation functions will be used at the end of every layer. In this model we used ReLU activation functions for every hidden layer output and for the output layer the activation function used is Softmax. Max-pooling, padding techniques are used for compatibility of one layer with the other.

TEST THE MODEL FOR VALIDATING DATASET

For every epoch test the model for the Validation test and alter the weights of the layers of convolution neural network and iterate the model with the updated weights again for the Training dataset. This process is continued until the weights of the layers of CNN model works for both Validation and Training dataset, this means the model is giving better accurate results.

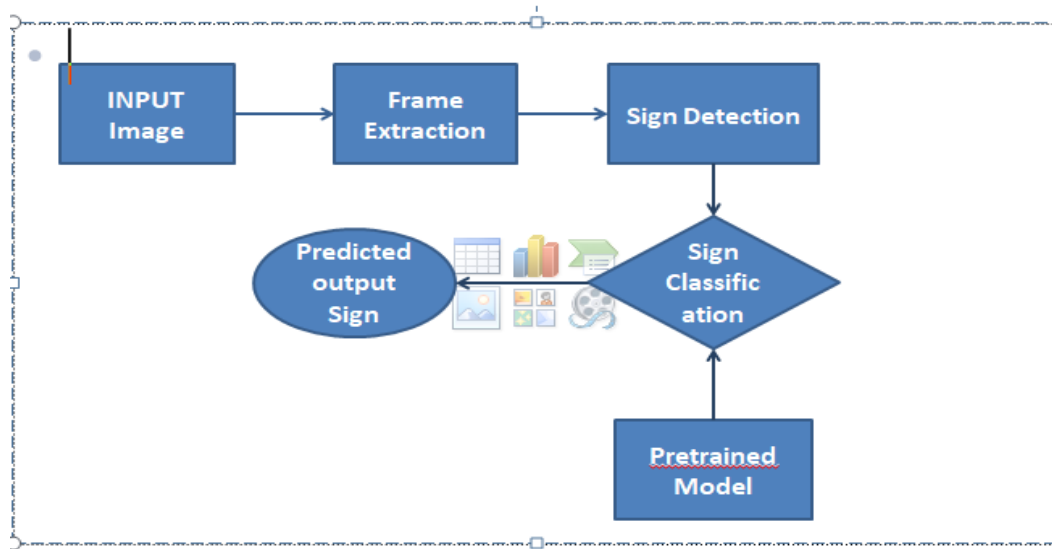
VALIDATE THE MODEL ACCURACY

The final Traffic sign recognition model is validated on the testing dataset which is splitted from the actual dataset for estimating the accuracy and loss of the model .

LOAD THE MODEL

Finally the model built is converted into binary format and stored for testing and using it in real world applications.

TRAINING THE MODEL



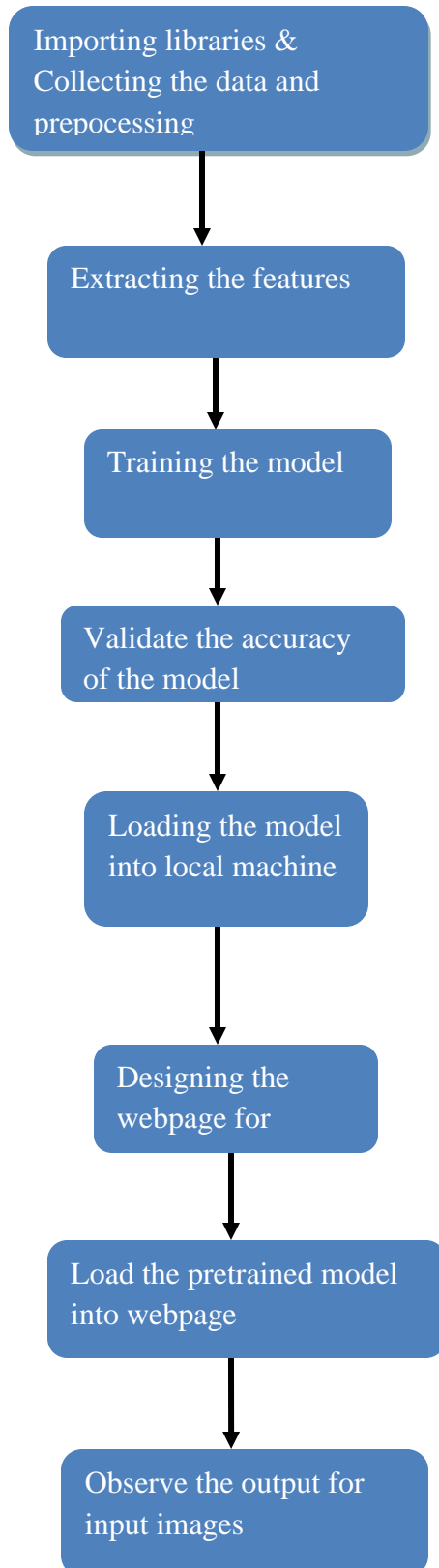
FRAME EXTRACTION

In testing algorithm when an image of traffic sign is given, the frame or range of pixels where the image of traffic sign is present is extracted.

SIGN DETECTION and CLASSIFICATION

From the extracted frame traffic sign is detected and it is classified into specific sign based on the CNN model pretrained using the dataset and finally the output is given along with the accuracy percentage of the predicted output.

3.3 FLOW OF IMPLEMENTATION



Traffic sign Recognition and Detection

STEP 1: Initially the dataset of traffic sign images is collected and it is preprocessed .The preprocessing involves changing the dimensions of the image, converting to gray scale, equalizing and then rotating shifting the images in the dataset so that there will be higher chances for extracting different types of features from the images and having better accuracy when using real world images.

STEP 2: In this step different features of all the images used in training the model are extracted . This is done by altering the imaging in the training dataset to the maximum possible ways so that the images are similar so that of images captured in alternating weather conditions.

STEP3: In this step a convolutional neural network with sufficient hidden layers is designed and developed using the training dataset i.e the weights of the model are defined based on the training images. The model is validated for every epoch with validation set and the process continues for certain number of epochs about 30 to get the better accuracy.

STEP 4: After completion of training the model it's accuracy and loss are evaluated based on the testing dataset.

STEP 5: This is the final step of training . In this step the trained model is converted to binary format and stored in local machines for using in applications.

STEP 6: For testing of the built CNN model of traffic sign recognition a webpage is created which takes the image uploaded and predicts the output.

STEP 7: The trained model is loaded into the testing algorithm which is interfaced with the webpage.

STEP 8: Finally the webpage created acts like an interface between user and the trained model. When we give an image of traffic sign to the webpage the model predicts the output.

CHAPTER – 4

IMPLEMENTATION

4.1 SOURCE CODE

4.1.1 TRAINING

Importing necessary Packages

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.models import model_from_json
import cv2
from sklearn.model_selection import train_test_split
import os
import pandas as pd
import random
from keras.preprocessing.image import ImageDataGenerator
```

Loading the dataset and label the images with their respective class names

```
count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Total Classes Detected:", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
for x in range(0, len(myList)):
    myPicList = os.listdir(path + "/" + str(count))    //path="MyData"
```

Traffic sign Recognition and Detection

```
for y in myPicList:
    curImg = cv2.imread(path + "/" + str(count) + "/" + y)
    images.append(curImg)
    classNo.append(count)
print(count, end=" ")
count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

CONSOLE OUTPUT

Total Classes Detected: 43

Importing Classes.....

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42

Splitting the dataset into test, Train, Validation images

```
X_train, X_test, y_train, y_test = train_test_split(images, classNo, test_size=testRatio)
// testRatio=0.2
```

```
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train,
test_size=validationRatio)
```

```
//validationRatio=0.2
```

- ```
print("Data Shapes")
print("Train", end="");
print(X_train.shape, y_train.shape)
print("Validation", end="");
print(X_validation.shape, y_validation.shape)
print("Test", end="");
print(X_test.shape, y_test.shape)
```

## Traffic sign Recognition and Detection

---

CONSOLE OUTPUT:

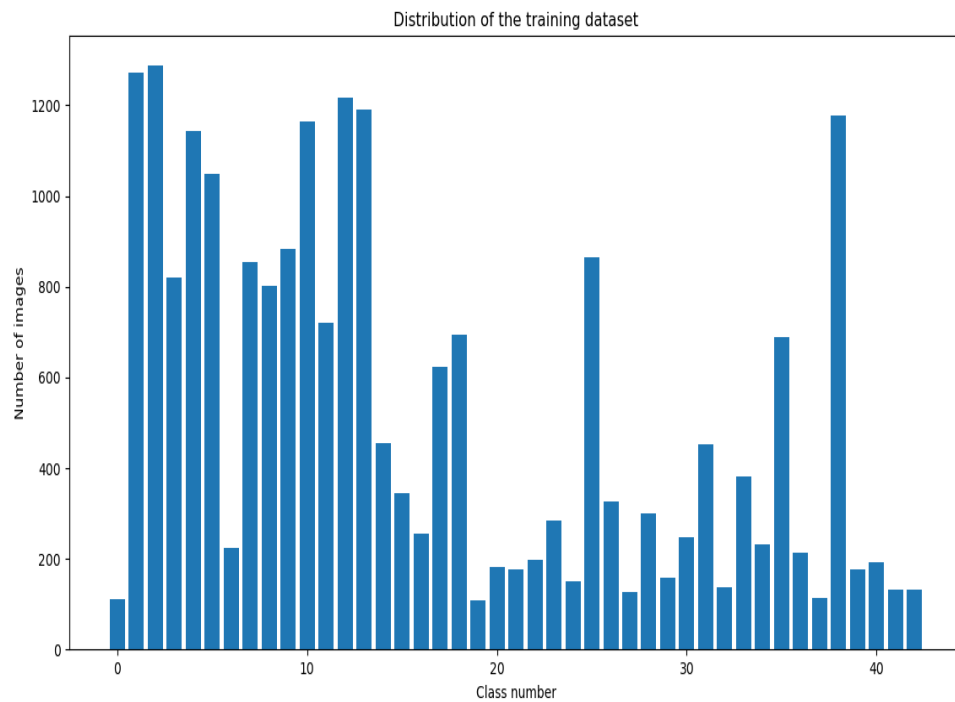
Data Shapes

Train(22271, 32, 32, 3) (22271,)

Validation(5568, 32, 32, 3) (5568,)

Test(6960, 32, 32, 3) (6960,)

BAR PLOT OF DATASET



### Data Preprocessing

```
def grayscale(img):
```

```
 img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
 return img
```

```
def equalize(img):
```

```
 img = cv2.equalizeHist(img)
```

```
 return img
```

```
def preprocessing(img):
```

```
 img = grayscale(img)
```

## Traffic sign Recognition and Detection

---

```
img = equalize(img)
img = img / 255
return img
```

```
X_train = np.array(list(map(preprocessing, X_train)))
X_validation = np.array(list(map(preprocessing, X_validation)))
X_test = np.array(list(map(preprocessing, X_test)))
cv2.imshow("GrayScale Images",
 X_train[random.randint(0, len(X_train) - 1)])
• X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2],
 1)
 X_validation = X_validation.reshape(X_validation.shape[0],
 X_validation.shape[1], X_validation.shape[2], 1)
 X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)

dataGen = ImageDataGenerator(width_shift_range=0.1,
 height_shift_range=0.1,
 zoom_range=0.2,
 shear_range=0.1,
 rotation_range=10)
dataGen.fit(X_train)
batches = dataGen.flow(X_train, y_train,
 batch_size=20)
X_batch, y_batch = next(batches)

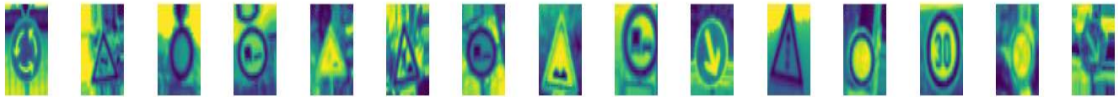
fig, axs = plt.subplots(1, 15, figsize=(20, 5))
fig.tight_layout()
• for i in range(15):
 axs[i].imshow(X_batch[i].reshape(imageDimesions[0], imageDimesions[1]))
```

## Traffic sign Recognition and Detection

---

```
 axs[i].axis('off')
plt.show()
```

### CONSOLE OUTPUT



```
y_train = to_categorical(y_train, noOfClasses)
y_validation = to_categorical(y_validation, noOfClasses)
y_test = to_categorical(y_test, noOfClasses)
```

### Defining and building the model

```
• def myModel():
 no_Of_Filters = 60
 size_of_Filter = (5, 5)

 size_of_Filter2 = (3, 3)
 size_of_pool = (2, 2)
 no_Of_Nodes = 500
 model = Sequential()
 model.add((Conv2D(no_Of_Filters, size_of_Filter,
input_shape=(imageDimesions[0], imageDimesions[1], 1),
activation='relu'))))
model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu'))))
model.add(MaxPooling2D(pool_size=size_of_pool))

model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu'))))
model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2, activation='relu'))))
model.add(MaxPooling2D(pool_size=size_of_pool))
```

## **Traffic sign Recognition and Detection**

---

```
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(no_Of_Nodes, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(noOfClasses, activation='softmax'))

model.compile(Adam(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])
return model
model = myModel()
print(model.summary())
history = model.fit(dataGen.flow(X_train, y_train, batch_size=batch_size_val),
 steps_per_epoch=steps_per_epoch_val/batch_size_val,
epochs=epochs_val,
 validation_data=(X_validation, y_validation), shuffle=1)
```

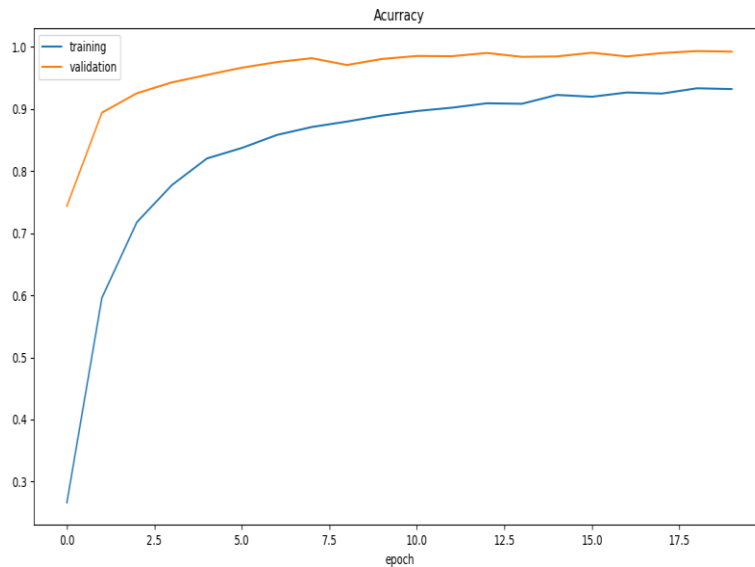
### **Plotting the accuracy and time plots**

- ```
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'])
plt.title('Accuracy')
plt.xlabel('epoch')
```

Traffic sign Recognition and Detection

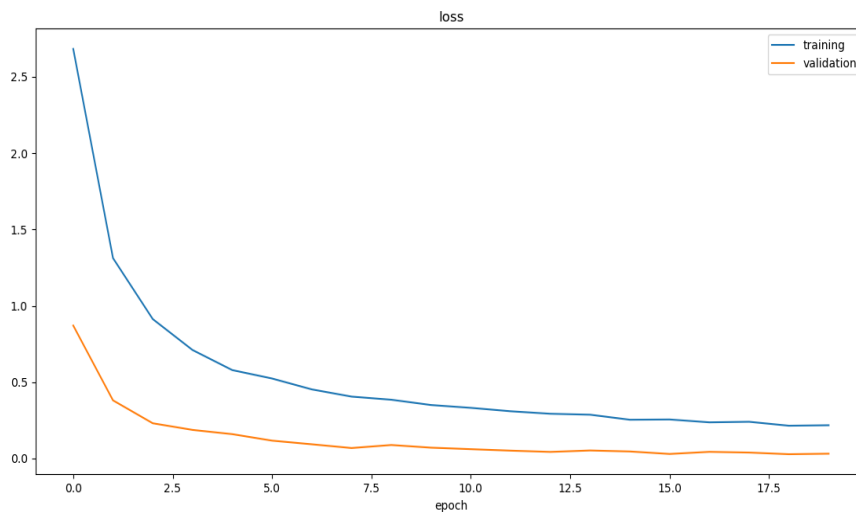
```
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Score:', score[0])
print('Test Accuracy:', score[1])
```

OUTPUT PLOTS OF MODEL



- Test Accuracy: 0.9926724433898926
- Saved model to disk
- Loaded model from disk
- accuracy: 99.27%

Accuracy and epoch plot



Loss vs epoch plot

Traffic sign Recognition and Detection

Load the model into binary file

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)

model.save_weights("model.h5")
print("Saved model to disk")
```

4.1.2 TESTING

Importing the necessary packages

```
from flask import *
import os
from werkzeug.utils import secure_filename
from keras.models import load_model
import numpy as np
from PIL import Image

app = Flask(__name__)
```

Classes of traffic signs

```
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',
```

Traffic sign Recognition and Detection

- 11: 'Right-of-way at intersection',
- 12: 'Priority road',
- 13: 'Yield',
- 14: 'Stop',
- 15: 'No vehicles',
- 16: 'Vehicle > 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',

Traffic sign Recognition and Detection

42: 'End no passing vehicle > 3.5 tons'}


Test Image processing

```
def image_processing(img):  
    model = load_model('./TSR.h5')  
    data = []  
    image = Image.open(img)  
    image = image.resize((30, 30))  
    data.append(np.array(image))  
    X_test = np.array(data)  
    Y_pred = model.predict_classes(X_test)  
    return Y_pred
```

Move to webpage

```
@app.route('/')  
def index():  
    return render_template('index.html')
```

Take the uploaded image and predict the output

```
@app.route('/predict', methods=['GET', 'POST'])  
def upload():  
    if request.method == 'POST':  
  
        f = request.files['file']  
        file_path = secure_filename(f.filename)  
        f.save(file_path)  
  
        result = image_processing(file_path)  
        s = [str(i) for i in result]  
        a = int("".join(s))  
        result = "Predicted Traffic  Sign is: " + classes[a]
```

Traffic sign Recognition and Detection

```
os.remove(file_path)

return result

return None
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

4.1.3 WEBPAGE CODES

HTML

BASE.HTML

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
<title>Traffic Signs 🚦 Classification</title>
```

```
<link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet">
```

```
<script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
```

```
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
```

```
<script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
```

```
<link href="{ { url_for('static', filename='css/main.css') } }" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<nav class="navbar navbar-dark bg-dark">
```

```
<div class="container">
```

```
<a class="navbar-brand" href="#"> <h3>Traffic 🚦 Signs Classifier</h3></a>
```

Traffic sign Recognition and Detection

```
</div>

</nav>

<div class="container">
  <div id="content" style="margin-top:2em">{% block content %}{% endblock
%}</div>
</div>
</body>

<footer>

  <script src="{ { url_for('static', filename='js/main.js') } }"
type="text/javascript"></script>
</footer>
</html>
```

INDEX.HTML

```
{% extends "base.html" %} {% block content %}
```

```
<h2>Upload Traffic Signs 🚦 </h2>
```

```
<style> .footer {
  position: fixed;
  left: 0;
  bottom: 0;
  width: 100%;
  background-color: #2dcba7;
  color: black;
  text-align: center;
  font-size: 25px;
  font-weight: bold;
}</style>
```

Traffic sign Recognition and Detection

```
<div>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <form id="upload-file" method="post" enctype="multipart/form-data">
    <label for="imageUpload" class="upload-label">
      Upload...
    </label>
    <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
  </form>

  <div class="image-section" style="display:none;">
    <div class="img-preview">
      <div id="imagePreview">
      </div>
    </div>
    <div>
      <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict
Traffic 🚦 Signs</button>
    </div>
  </div>

  <div class="loader" style="display:none;"></div>

  <h3 id="result">
    <span> </span>
  </h3>

</div>

{ % endblock % }
```

Traffic sign Recognition and Detection

CSS CODE

```
.img-preview {  
    width: 256px;  
    height: 256px;  
    position: relative;  
    border: 5px solid #F8F8F8;  
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);  
    margin-top: 1em;  
    margin-bottom: 1em;  
}
```

```
.img-preview>div {  
    width: 100%;  
    height: 100%;  
    background-size: 256px 256px;  
    background-repeat: no-repeat;  
    background-position: center;  
}
```

```
input[type="file"] {  
    display: none;  
}
```

```
.upload-label{  
    display: inline-block;  
    padding: 12px 30px;  
    background: #39D2B4;  
    color: #fff;  
    font-size: 1em;  
    transition: all .4s;  
    cursor: pointer;
```

Traffic sign Recognition and Detection

```
}
```

```
.upload-label:hover{  
    background: #34495E;  
    color: #39D2B4;  
}
```

```
.loader {  
    border: 8px solid #f3f3f3; /* Light grey */  
    border-top: 8px solid #3498db; /* Blue */  
    border-radius: 50%;  
    width: 50px;  
    height: 50px;  
    animation: spin 1s linear infinite;  
}
```

```
@keyframes spin {  
    0% { transform: rotate(0deg); }  
    100% { transform: rotate(360deg); }  
}
```

JAVA SCRIPT CODE

```
$(document).ready(function () {  
    // Init  
    $('.image-section').hide();  
    $('.loader').hide();  
    $('#result').hide();  
  
    // Upload Preview  
    function readURL(input) {  
        if (input.files && input.files[0]) {
```


Traffic sign Recognition and Detection

```
var reader = new FileReader();
reader.onload = function (e) {
    $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
    $('#imagePreview').hide();
    $('#imagePreview').fadeIn(650);
}
reader.readAsDataURL(input.files[0]);
}
}

$("#imageUpload").change(function () {
    $('.image-section').show();
    $('#btn-predict').show();
    $('#result').text("");
    $('#result').hide();
    readURL(this);
});

// Predict
$('#btn-predict').click(function () {
    var form_data = new FormData($('#upload-file')[0]);

    // Show loading animation
    $(this).hide();
    $('.loader').show();

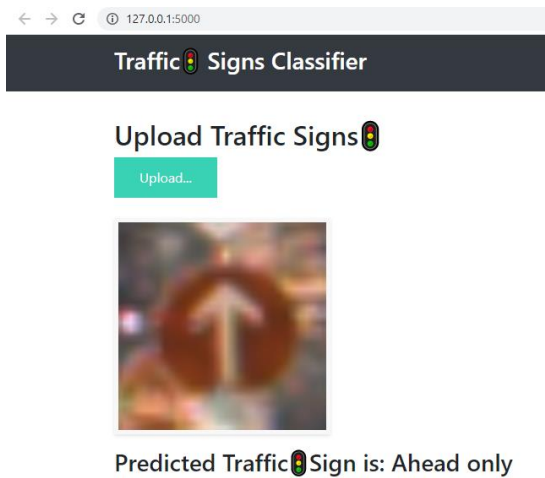
    // Make prediction by calling api /predict
    $.ajax({
        type: 'POST',
        url: '/predict',
        data: form_data,
        contentType: false,
```

Traffic sign Recognition and Detection

```
cache: false,  
processData: false,  
async: true,  
success: function (data) {  
    // Get and display the result  
    $('#loader').hide();  
    $('#result').fadeIn(600);  
    $('#result').text(data);  
    console.log('Success!');  
},  
});  
});  
  
});
```

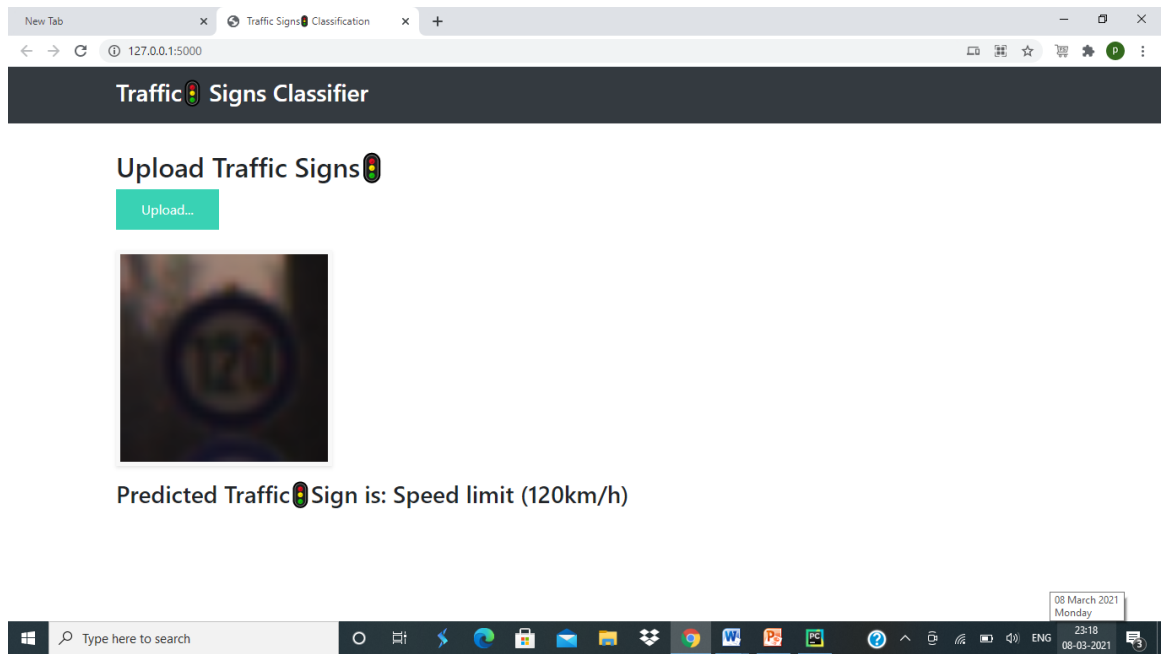
4.2 RESULTS

OUTPUT1:



Traffic sign Recognition and Detection

OUTPUT2:



OUTPUT 3:



Traffic sign Recognition and Detection

OUTPUT 4

New Tab x Traffic Signs Classification x +

127.0.0.1:5000

Traffic Signs Classifier

Upload Traffic Signs

Upload...



Predicted Traffic Sign is: Children crossing

Type here to search

Task View Edge File Explorer Mail Calendar Photos OneDrive Google Chrome Word PowerPoint Excel ZTE-SM55Sg Internet access ENG 23:19 08-03-2021

CHAPTER -5

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The developed Convolutional neural network model of Traffic sign Recognition system produced the desired results for the random input images of Traffic sign boards. The images tested includes images which are taken in snow weather conditions i.e the images which have low quality or resolution. The model developed even gave good results for the those images. And also the accuracy of the model is very good . Therefore the developed model gives very good results for different sign images of trained classes within less time.

6.2 FUTURE SCOPE

The model can be further developed to extract and detect the multiple traffic signs in the same frame of image by adding the feature of searching for the different frames in the given image for multiple traffic signs.

REFERENCES

- [1] A. de la Escalera; L.E. Moreno; M.A. Salichs; J.M. Armingol **“Road traffic sign detection and classification”** IEEE JOURNAL 2015.

- [2] Safat B.wali,Mahammad A.Hannan,Anni Hussain and Salina A.Samad **“An automatic Traffic Sign Detection and Recognition system based on colour Segmentation, Shape Matching, and SVM”** Published on 17 November 2015 in hindawi.com

- [3] Maria-Adelina Sirbu, Andrei Baiasu, Razvan Bagdan,Mihaela Crisan-Vida **“Smart Traffic sign detection on autonomous car”** published on November 2018 in researchgate.net

- [4] Safat B.wali,Mahammad A.Hannan,Anni Hussain and Salina A.Samad,Pin J.Ker and Muhamad Bin Mansor **“Vision-Based Traffic sign Detection and Recognition System”** published on 6th May 2019 in ncdi.nlm.nih.gov

- [5] *Zhongyu Wang ,Hui Guo “Research on traffic sign detection based on convolutional neural network”* published in the proceedings of the 12th International Symposium on visual communication and interaction September 2019.