TABLE OF CONTENTS

ABSTRACT

Traffic sign recognition system (TSRS) is a significant portion of intelligent transportation system (ITS). Being able to identify traffic signs accurately and effectively can improve the driving safety. This paper brings forward a traffic sign recognition technique on the strength of deep learning, which mainly aims at the detection and classification of circular signs. Firstly, an image is preprocessed to highlight important information. Secondly, Hough Transform is used for detecting and locating areas. Finally, the detected road traffic signs are classified based on deep learning. In this article, a traffic sign detection and identification method on account of the image processing is proposed, which is combined with convolutional neural network (CNN) to sort traffic signs. On account of its high recognition rate, CNN can be used to realize various computer vision tasks. TensorFlow is used to implement CNN. In the German data sets, we are able to identify the circular symbol with more than 98.2% accuracy.

Introduction:-

You must have heard about the self-driving cars in which the passenger can fully depend on the car for traveling. But to achieve level 5 autonomous, it is necessary for vehicles to understand and follow all traffic rules.

In the world of Artificial Intelligence and advancement in technologies, many researchers and big companies like Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi, etc are working on autonomous vehicles and self-driving cars. So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly.

A visual-based traffic sign recognition system can be implemented on the automobile with an aim of detecting and recognizing all emerging traffic signs. The same would be displayed to the driver with alarm-triggering features if the driver refuses to follow the traffic signs.

At eInfochips, we have made an attempt to help automotive companies detect and recognize traffic signs in video sequences recorded by on-board vehicle camera. Traffic Sign Recognition (TSR) is used to display the speed limit signs. Here, OpenCV is used for image processing. OpenCV is an Open source Computer Vision library designed for computational efficiency with a strong focus on real time applications.


Class 0-15
Class 1-110
Class 2-13
Class 3-15
Class 4-15
Class 5-11
Class 6-18
Class 7-157
Class 8-27
Class 9-18
Class 10-21
Class 11-7
Class 12-18
Class 13-90
Class 14-43
Class 15-9
Class 16-9
Class 17-79
Class 18-81
Class 19-231
Class 20-42
Class 21-43
Class 22-375
Class 23-15
Class 24-48
Class 25-42
Class 26-6
Class 27-18
Class 28-125
Class 29-33
Class 30-37
Class 31-63
Class 32-316
Class 33-12
Class 34-46
Class 35-60
Class 36-18
Class 37-98
Class 38-285
Class 39-196
Class 40-242
Class 41-148
Class 42-35
Class 43-30
Class 44-48
Class 45-74
Class 46-44
Class 47-147
Class 48-11
Class 49-12
Class 50-15
Class 51-27
Class 52-27
Class 53-199
Class 54-118
Class 55-12
Class 56-95
Class 57-78
Class 58-15
Class 59-42
Class 60-9
Class 61-282

- **<u>Technology Used:-</u>**

- Language:- PYTHON
- Tools – Kaggle for data sets

<u>Approach:-</u>

Our approach to building this traffic sign classification model is discussed in

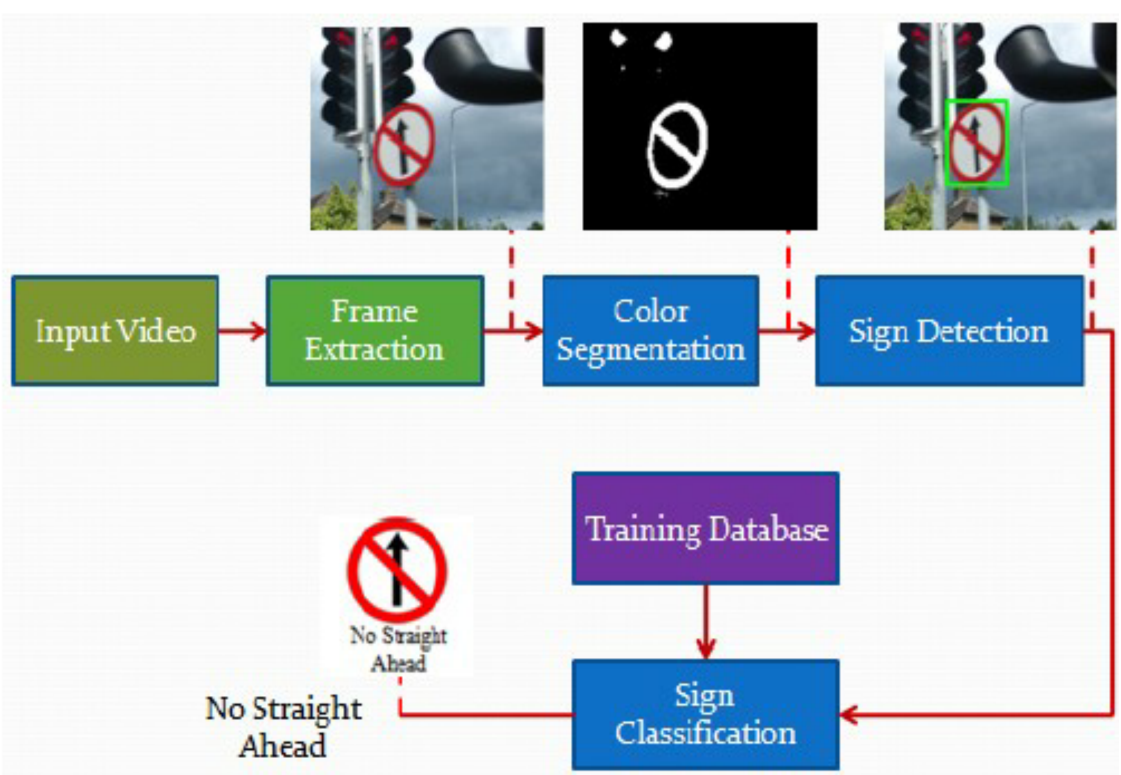
four steps:

- Explore the dataset
- Build a CNN model
- Train and validate the model

- Test the model with test dataset

OBJECTIVE OF THE PROJECT:-

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.



Requirement:-

This project requires prior knowledge of Keras, Matplotlib, Scikit-learn, Pandas, PIL and image classification.

For this project, we are using the public dataset available at Kaggle:

The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around 300 MB. The dataset has a train folder which contains images inside each class and a test folder which we will use for testing our model.

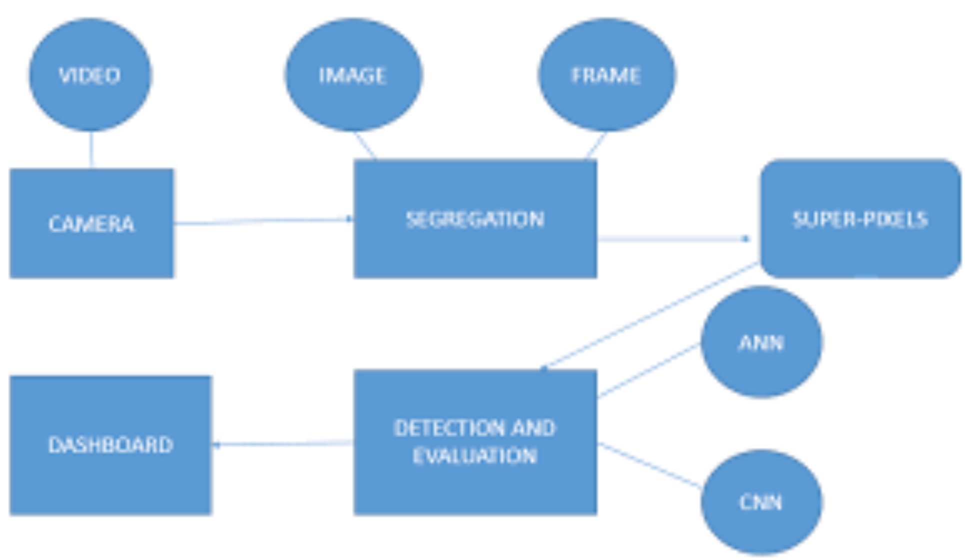
System Requirement:-

Operating System: Windows

RAM: 1GB or more memory

Hard-disk drive: 250 GB

Hard-disk drive: 500 GB



Conclusion:-

we have successfully classified the traffic signs classifier with 95% accuracy and also visualized how our accuracy and loss changes with time, which is pretty good from a simple CNN model.

SOURCE CODE :-

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
import os
os.chdir('D:/Traffic_Sign_Recognition')
```

```
Using TensorFlow backend.
```

In [2]:
```
data =[]
labels = []
classes =43
cur_path = os.getcwd()
```

In [3]:
```
cur_path
```

Out[3]:
```
'D:\\Traffic_Sign_Recognition'
```

In [4]:
```
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path +'\\'+ a)
            image = image.resize((30,30))
            # Resizing all images into 30*30
            image =np.array(image)
            data.append(image)
            labels.append(i)
        except Exceptionas as e:
            print(e)
```

In [5]:
```
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
```

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

In [6]:
```
X_train, X_test, y_train, y_test =train_test_split(data, labels, test_size=0.2, random_state=0)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

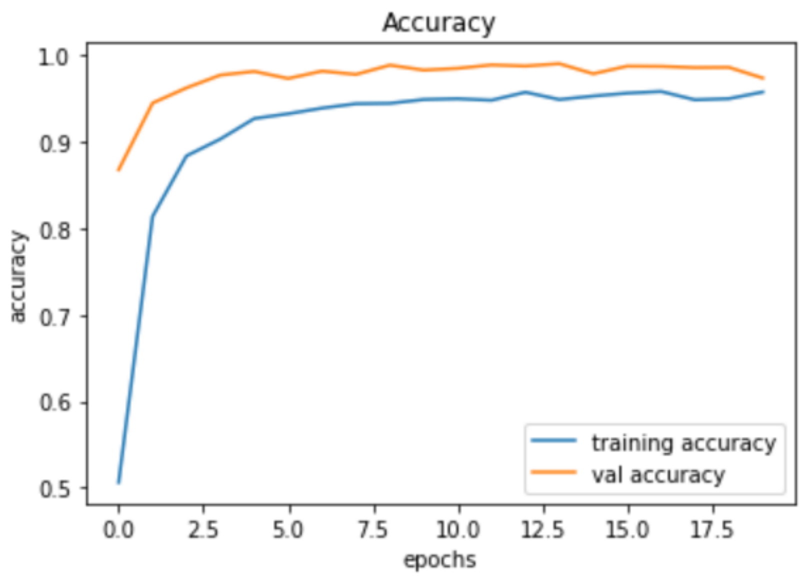
```
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

In [7]:
```
y_train = to_categorical(y_train,43)
y_test = to_categorical(y_test,43)
```

In [8]:
```
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))
# We have 43 classes that's why we have defined 43in the dense
model.add(Dense(43, activation='softmax'))
```

In [10]:

```python
#accuracy
plt.figure(0)
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()
```




```
Epoch 12/20
31367/31367 [==============================] - 127s 4ms/step - loss: 0.2055 - accuracy: 0.9482 - val_loss: 0.0472 - val_accurac
y: 0.9890
```

In [11]:
```
#Loss
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()
```