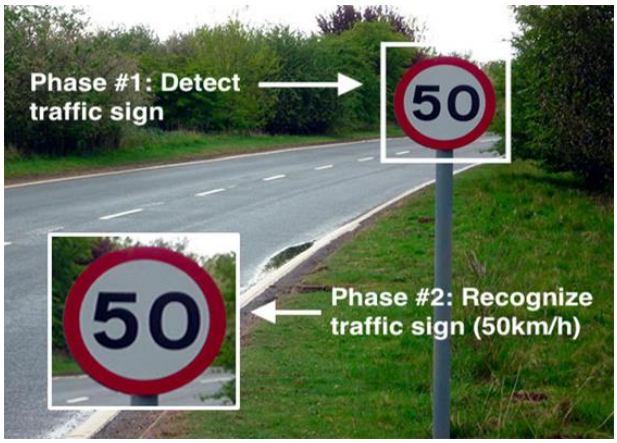
Symbiosis Skills and Professional University

Kiwale, Pune

PROJECT REPORT

On

"Traffic Signs Recognition"



Submitted by

Mr . Gaurang Vivek Sonkavde

(Registration Number : 2001207071)

DA-Batch-II

Under The Guidance of

Trainer's Name: Mr. Kushal Sharma

STUDENT DECLARATION AND ATTESTATION BY TRAINER

This is to declare that this report has been written by me. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I aver that if any part of the report is found to be plagiarized, I shall take full responsibility for it.

Signature of student

Name of student: Mr. Gaurang Vivek Sonkavde

Registration Number: 2001207071

Signature of trainer

Name of trainer: Mr. Kushal Sharma

CERTIFICATE

This is to certify that the report entitled, "**Traffic Signs Recognition**" submitted by "Mr.Gaurang Vivek Sonkavde" to Symbiosis Skills and Professional University, Pune, Maharashtra, India, is a record of bonafide Project work carried out by him under my supervision and guidance and is worthy of consideration for the completion of certificate course in 'Data Associate".

Signature of Trainer
Name of Trainer : Kushal Sharma

Date: / / 2021

_____ _____

Supervisor Supervisor

Date:

ACKNOWLEDGEMENTS

This is to acknowledge all those without whom this project would not have been reality. Firstly, I would wish to thank our Trainer Mr. Kushal Sharma sir who gave his immense support, dedicated his time towards it and made us understand how to make this project. Without his guidance, the project would not have been complete.

A project is a bridge between theoretical and practical learning and with this thinking I worked on the project and made it successful due to timely support and efforts of all who helped me.

Once again, I would like to put my gratitude and sincere thanks to DR. Shravan Kadvekar for giving me this opportunity.Then I would also like to thank my classmates and my friends also for their encouragement and help in designing and making my project creative.Only because of them I was able to create my project and make it a good and enjoyable experience.

INDEX (Sample)

2.

PURPOSE AND PROBLEM STATEMENT OF PROJECT

Purpose Of Project

Traffic signs are an essential part of our day to day lives. 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. The annual global roach crash statistics say that over 3,280 people die every day in a road accident. These numbers would be much higher in case if there were no traffic signs.

On the other hand, researchers and big companies are working extensively on proposing solutions to self-driving cars. Just to name a few these include Tesla, Uber, Google, Audi, BMW, Ford, Toyota, Mercedez, Volvo, Nissan, etc. These autonomous vehicles need to follow the traffic rules and for that, they have to understand the message conveyed through traffic signs.

Every country has some standards set for the design of different traffic signs like U-turn, Left-turn, Right-turn, No-entry, etc. Traffic sign recognition is the process of automatically identifying which of the following class the sign belongs to. The earlier Computer Vision techniques required lots of hard work in data processing and it took a lot of time to manually extract the features of the image. Now, deep learning techniques have come to the rescue and today we will see how to build a traffic recognition system for autonomous vehicles.

Problem Statement

- To propose a method to extract the image frames
- To propose a method to remove the effect of the different illuminations on
- extracted image frames.
- To propose an algorithm to identify the regions of interest
- To propose a method to extract the symbol of the traffic warning sign
- To propose a method to identify the symbol of the traffic warning sign.

3.

OBJECTIVE OF PROJECT

General Objective

Identification of the Region of interest

After capturing a single image frame from a video segment, the regions of interest can be identified. These regions are the candidate regions which may be a traffic warning signs. To identify these regions, first it should threshold the Normalized RGB image and then median filter should be applied to remove the noise. Figure 1.1 illustrates the Region of interest for a particular image frame.

Extracting the symbol of the warning sign

After extraction of the Regions of interest, each region is further analyzed to extract the symbol of the warning sign. Compliment the logical images, removing the regions attached with the image border are some of the techniques which were used for this purpose. illustrates the extracted symbol after analyzing a single candidate region.

Recognition of the Traffic warning sign

After extracting the symbol of the warning sign, there are several methods (that can be used) to identify the traffic warning signs. Template matching techniques and pattern recognition techniques using neural networks are some of the complex techniques which have been used for this at the previous related works. (But these methods are having lot of difficulties at the implementation process, and the target of this research work was to develop an algorithm which can be easily implemented). But (Therefore,) in this research work, simple mathematical techniques (matrix theory) were used to identify the traffic warning signs..

	A	B	C	D	E	F	G	H	I	J	K	L
1	SrNo	Date_of_week	Day_of_Week	Light_Conditions	Sex_Of_Driver	Vehicle_Type	Speed_limit	Pedestrian_Crossing	Road_Type	Special_Conditions_at_Site	Number_of_Pasengers	Accident_Severity
2	1	01-01-2021	Monday	Darkness	Female	1	20	0	5	0	3	Serious
3	2	02-01-2021	Tuesday	Darkness	Female	5	40	3	3	1	3	Slight
4	3	03-01-2021	Wednesday	Daylight	Female	0	40	0	2	1	3	Slight
5	4	04-01-2021	Thursday	Darkness	Male	1	20	1	3	0	5	Slight
6	5	05-01-2021	Friday	Daylight	Female	1	20	1	6	1	4	Slight
7	6	06-01-2021	Saturday	Daylight	Female	1	20	1	4	0	4	Serious
8	7	07-01-2021	Sunday	Darkness	Male	0	40	1	4	0	5	Slight
9	8	08-01-2021	Saturday	Daylight	Male	0	30	1	3	0	4	Serious
10	9	09-01-2021	Friday	Darkness	Male	1	30	1	1	0	3	Slight
11	10	10-01-2021	Friday	Daylight	Female	0	30	1	3	1	4	Slight
12	11	11-01-2021	Tuesday	Darkness	Male	0	40	0	2	1	3	Serious
13	12	12-01-2021	Friday	Darkness	Female	0	40	1	3	0	4	Serious
14	13	13-01-2021	Saturday	Darkness	Female	0	40	1	1	0	6	Slight
15	14	14-01-2021	Tuesday	Darkness	Male	0	40	1	2	0	3	Slight
16	15	15-01-2021	Friday	Darkness	Male	1	30	0	6	0	3	Slight
17	16	16-01-2021	Friday	Darkness	Female	1	30	0	6	1	2	Serious
18	17	17-01-2021	Wednesday	Darkness	Female	1	30	0	1	1	2	Serious
19	18	18-01-2021	Saturday	Daylight	Male	1	20	0	1	1	2	Serious
20	19	19-01-2021	Friday	Daylight	Female	0	40	1	6	1	1	Slight
21	20	20-01-2021	Monday	Daylight	Male	0	40	1	4	0	1	Slight
22	21	21-01-2021	Friday	Darkness	Male	1	30	0	3	1	2	Slight

4.

Steps and Stratergies Performed

Algorithm Used

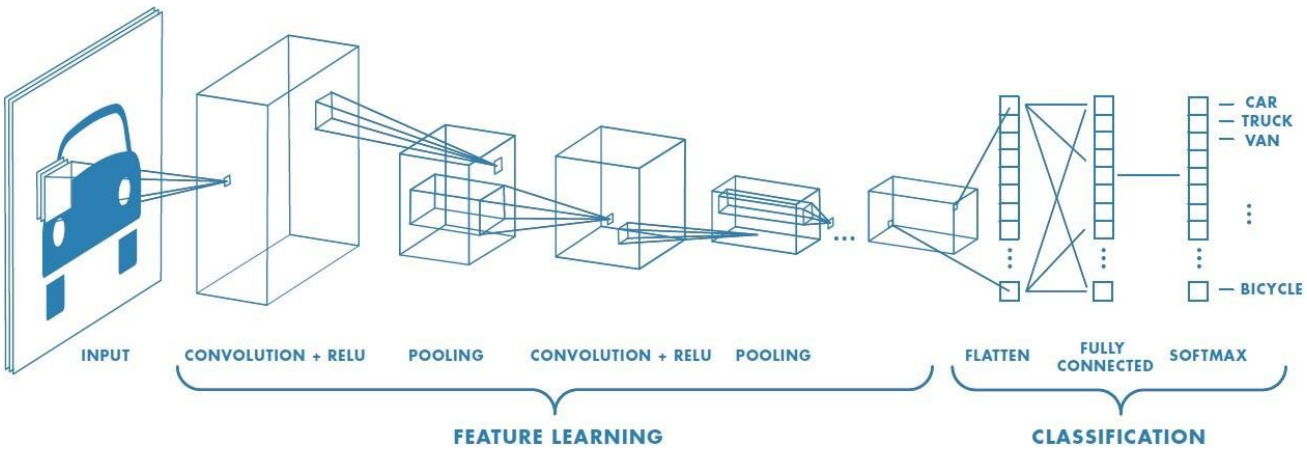
CNN (Convolutional Neural Network) Algorithm

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns.

CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification.

A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers, and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex.

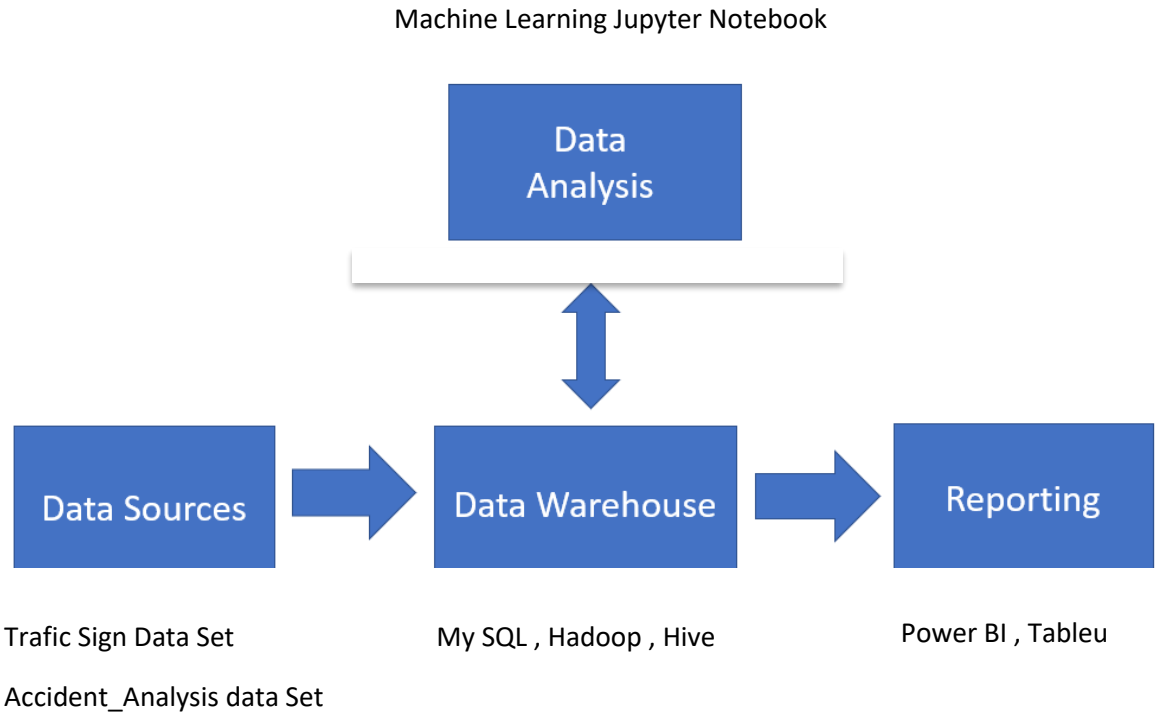
The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.



4.1 Architecture:

The Dataflow is as follows:

1. We made a Project where we used Deep Learning also we used Keras and made an Project which will Identify Traffic signal with help of an Machine learning Algorithm called as CNN (Convolution Neural Network) also after that we made a GUI(Graphic User Interface) where I we need to upload Images and with the help of Dataset Traffic Sign is identified.
2. After that we download Data from Kaggle which we will use for Accident Analysis of India which is an CSV File and will have all the features.
3. These csv files will be loaded to MySQL Database for Data Querying.
4. Later this data is exported to Apache Hadoop (Hadoop Distributed File Storage) using Sqoop (ETL tool).
5. Again using Sqoop command we store or import that Accident Analysis CSV File to Hive which is Data Warehouse.
6. Data is then used for Reporting and after that we analyze the data set and built an Dashboard on Power BI and Tableau and after that Report it.



4.2 Implementation:

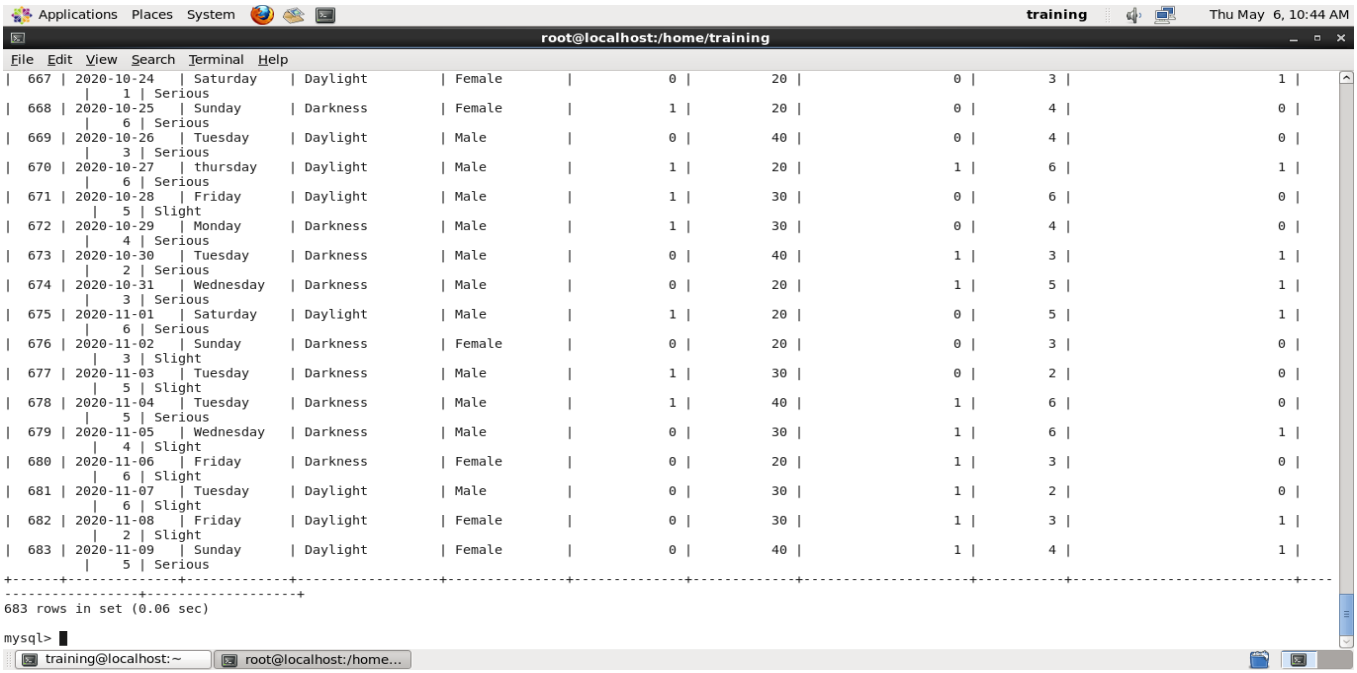
The implementation details are shared according to the dataflow mentioned in section 3.1.

1. Data Sources:

   With the help of Kaggle we downloaded data in CSV File for accident Analysis and Image Data set of Traffic Signs to Jupyter Notebook .The CSV files are then uploaded to MySQL database.

2. MySQL database Description:

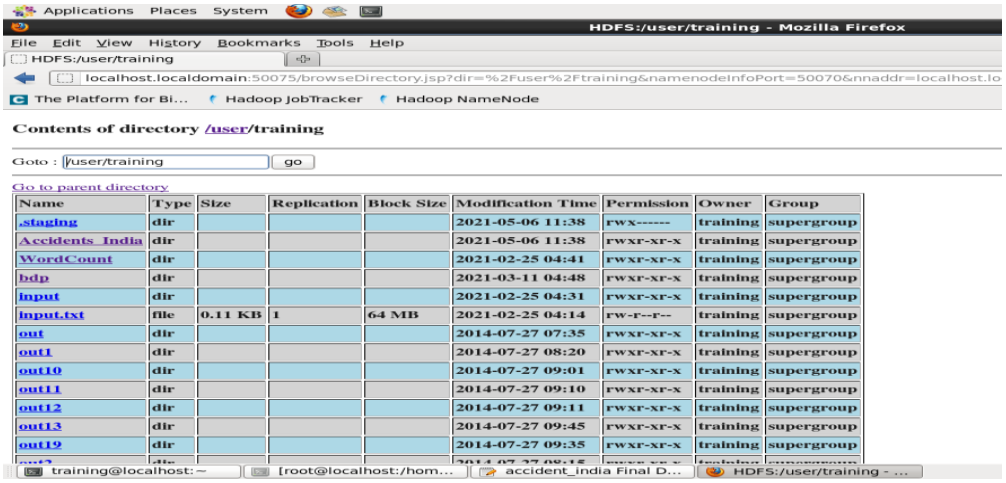
We have created a database named “Accident Analysis”, which contains table “Accidents_India” which hasAccident_India data mentioned in section The data is loaded with the CSV file mentioned in the MySQL Source file (.sql). also later we uploaded that data in HDFS and Hive.



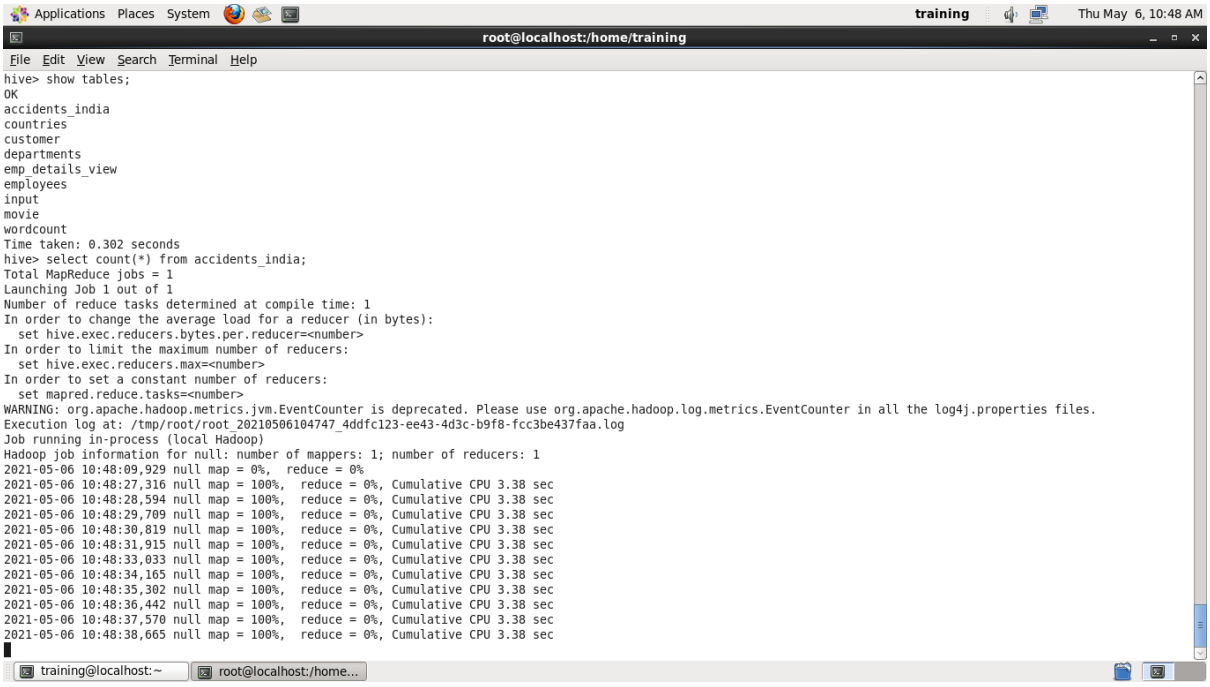
Screenshot: SQL

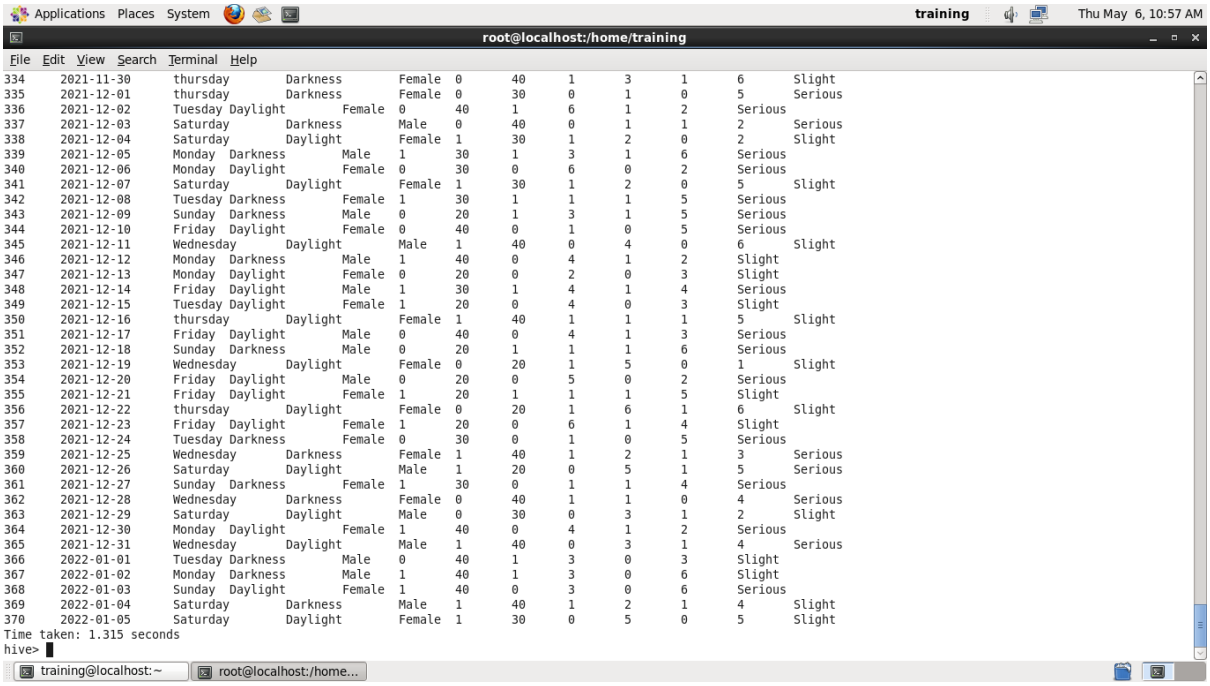
3. MySQL to HDFS using Sqoop ETL tool:

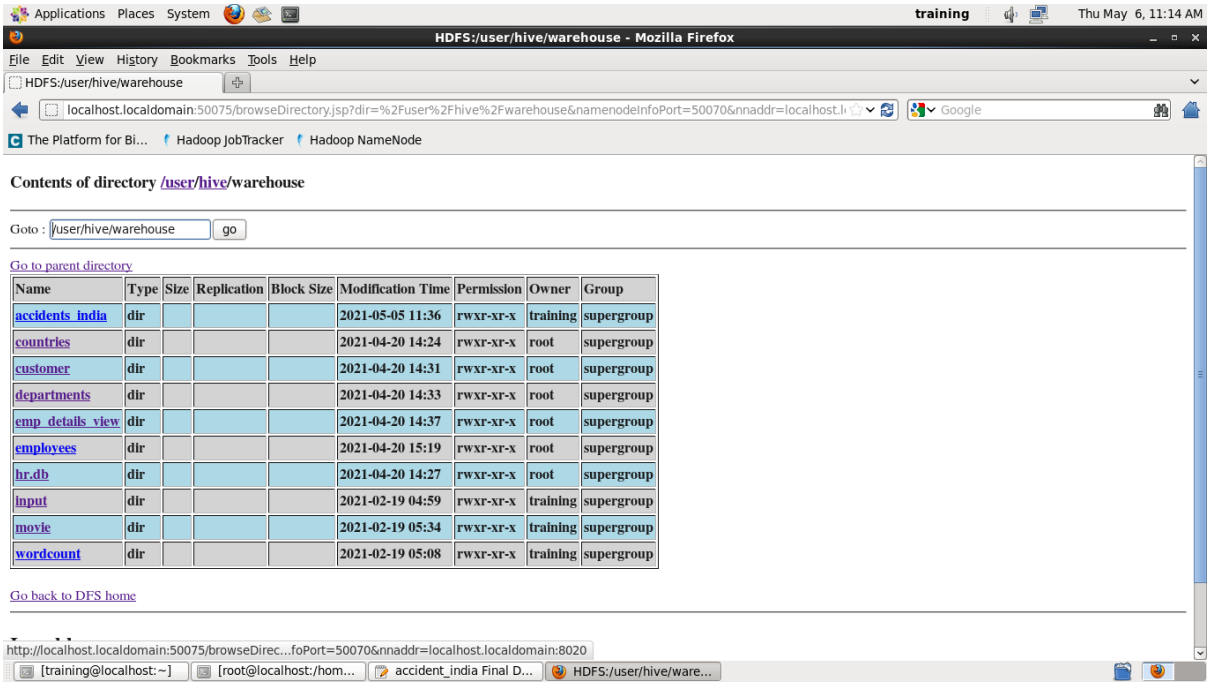
   With the use of Sqoop, I have transferred the cleveland table from MySQL to HDFS at /HeartA/.



Screenshot : HDFS







Screenshot: HIVE

4. Machine Learning:

With Jupyter Notebook ML library, we created a Classification Neural Network Model using KERAS and Deep Learning. By splitting the dataset in 80-20 ratio, we trained our Model with 80% of Train Dataset and the rest was used for testing our model. Later we made an GUI and by using that we can classify Traffic Signs.

```
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

In [2]: import os

In [3]: os.chdir(r'C:\Users\gaura\OneDrive\Desktop\traffic')

In [4]: 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
```

Store Data Labels in the List

```
In [5]: data = []
        labels = []
        classes = 43

In [6]: cur_path = os.getcwd()

In [7]: cur_path
Out[7]: 'C:\\Users\\gaura\\OneDrive\\Desktop\\traffic'
```

Preprocess the Image

```
In [8]: #Retrieving the images and their Labels
        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))
                    image = np.array(image)
                    #sim = Image.fromarray(image)
                    data.append(image)
                    labels.append(i)
                except Exception as e:
                    print(e)
```

Convert List into NumPy Arrays

```
In [9]: data = np.array(data)
        labels =np.array(labels)
```

Save Labels and data for Future Use

```
In [10]: #os.mkdir('training')

         np.save('./training/data',data)
         np.save('./training/target',labels)
```

Load Data and Labels

```
In [11]: data=np.load('./training/data.npy')
         labels=np.load('./training/target.npy')

In [12]: print(data.shape, labels.shape)
         (39209, 30, 30, 3) (39209,)
```

```
In [13]: X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=0)

In [14]: print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
         (31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

Convert labels to One Hot Encoding

```
In [15]: y_train = to_categorical(y_train, 43)
         y_test = to_categorical(y_test, 43)
```

Build the Model

```
In [16]: 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'))
```

Compile the Model

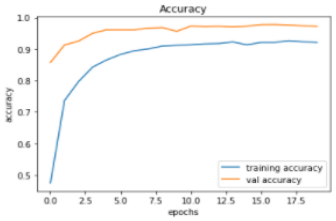
```
In [17]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

In [18]: epochs = 20
         history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
         Epoch 1/20
         981/981 [==============================] - 105s 105ms/step - loss: 3.5325 - accuracy: 0.2974 - val_loss: 0.6073 - val_accuracy:
         0.8576
```

In [19]:
```
model.save("my_model.h5")
```

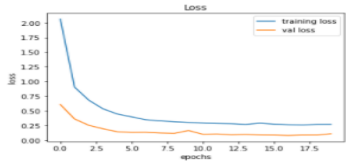
Accuracy Plot

In [20]:
```
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()
```



Loss Plot

In [21]:
```
plt.figure(1)
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()
```



Testing on Test Data

In [22]:
```
from sklearn.metrics import accuracy_score
def testing(testcsv):
    y_test = pd.read_csv('Test.csv')
    labels = y_test["ClassId"].values
    imgs = y_test["Path"].values
    data=[]
    for img in imgs:
        image = Image.open(img)
        image = image.resize((30,30))
        data.append(np.array(image))
    X_test=np.array(data)
```

In [23]:
```
X_test=np.array(data)
```

In [24]:
```
Y_pred = model.predict_classes(X_test)
Y_pred
```

```
C:\Users\gaura\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[24]:
```
array([ 0,  0,  0, ..., 42, 42, 42], dtype=int64)
```

Accuracy with Test Data

In [25]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, Y_pred))
```

```
0.9804891734040654
```

Save Model

In [26]:
```
model.save('./training/TSR.h5')
```

Make Graphic User Interface

In [27]:
```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
```

Load the trained model to Classify Sign

In [28]:
```
from keras.models import load_model
model = load_model('./training/TSR.h5')
```

Dictionary to label all traffic signs class

In [29]:
```
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
            10:'No passing',
            11:'No passing veh over 3.5 tons',
            12:'Right-of-way at intersection',
            13:'Priority road',
            14:'Yield',
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
```

Initialize Graphic User Interface (GUI)

In [30]:
```
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)
```

In [31]:
```
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict_classes([image])[0]
    sign = classes[pred+1]
    print(sign)
    label.configure(foreground='#011638', text=sign)
```

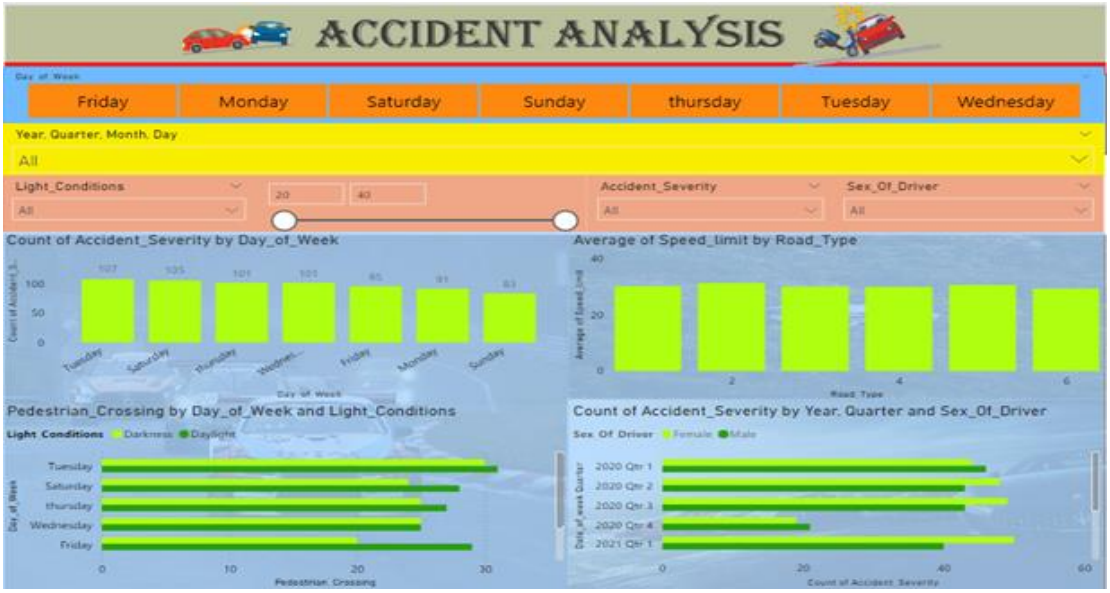
In [32]:
```
def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda: classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)
```

In [ ]:
```
def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Traffic Sign",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

```
No vehicles
```

5. <u>Reporting:</u>

   With the help of Business Intelligence Tools like Tableau and Power BI, we created interactive dashboards which help us visualize the data and draw insights from it.



<u>Power BI dashboard</u>

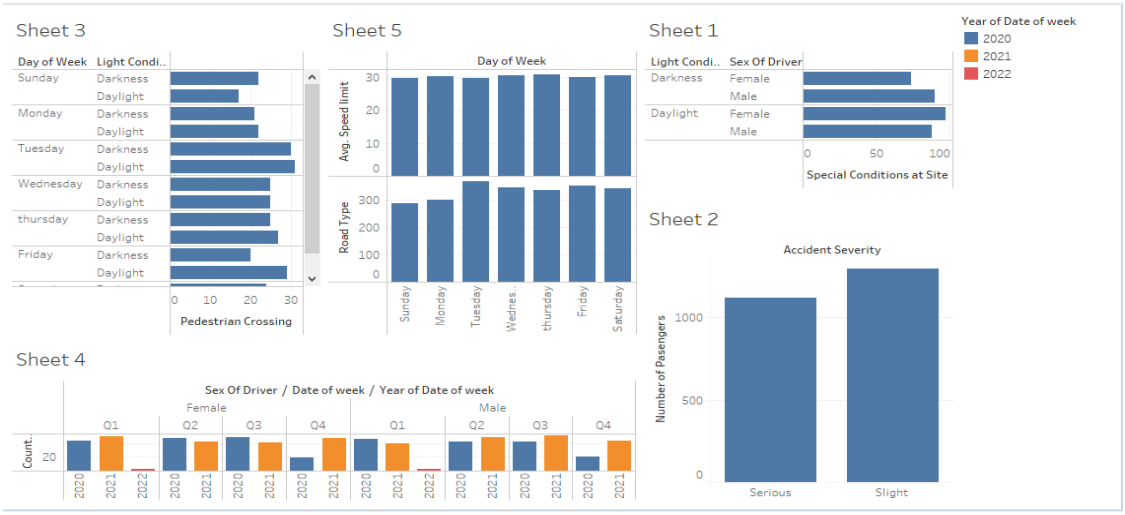
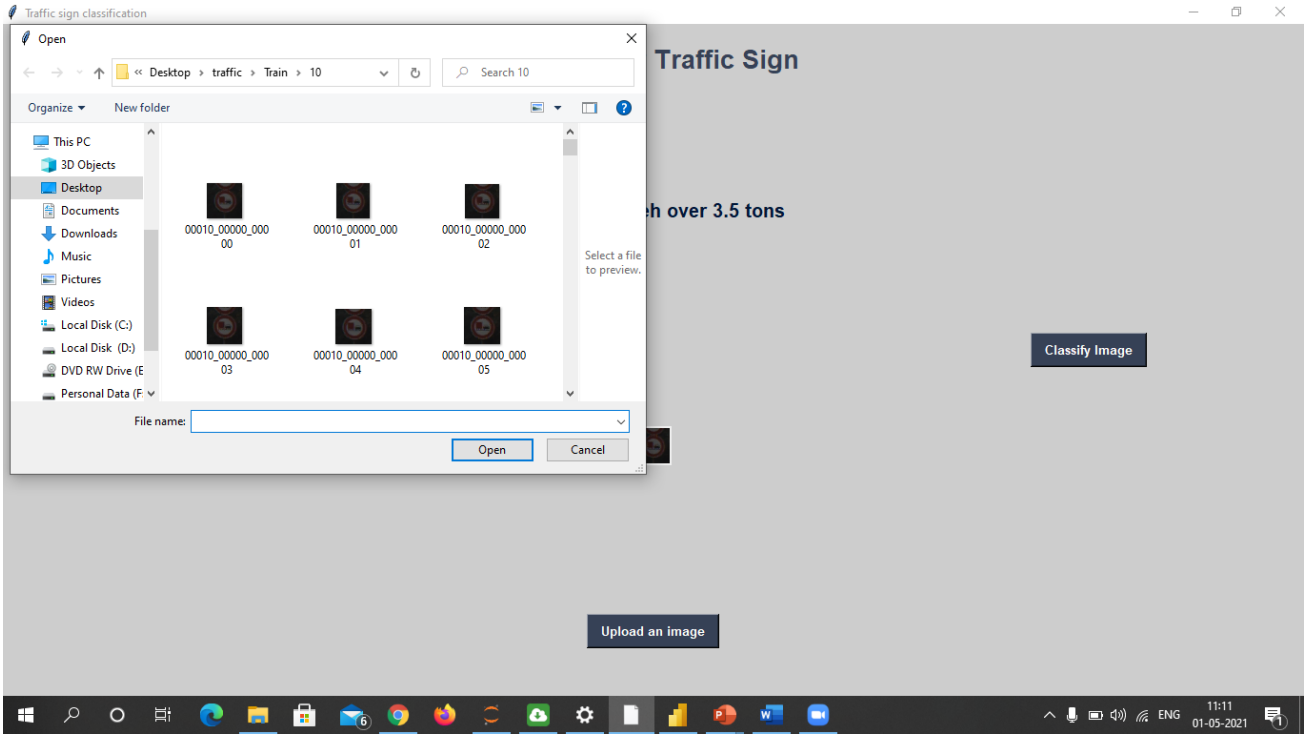


Tableau Public Dashboard

5.

OUTPUT

GUI Representation




Traffic sign classification
Know Your Traffic Sign
No passing veh over 3.5 tons
Classify Image
Upload an image
ENG
11:07
01-05-2021

6.

Future Scope And Conclusion

Future Scope

This system can be improved to identify the traffic signs other than warning signs. In this research work, we have used the Normalized RGB color space to identify the traffic warning signs. By using Normalized RGB color space; detection of the Red color is also possible. Therefore this color space can be used to identify the traffic signs with the Red color outline also.

The techniques introduced in this report can be used as a basis for developing general purpose, advanced intelligent traffic surveillance systems. By combining with character pattern recognition process, our method can be extended to recognize the vehicle license plate number, which has recently become an active research area. Extraction of the characters can be done by changing the threshold values.

Conclusion

This paper considers an implementation of the classification algorithm for the traffic signs recognition task. Combined with preprocessing and localization steps from previous works, the proposed method for traffic signs classification shows very good results: 99.94 % of correctly classified images. The proposed classification solution is implemented using the TensorFlow framework. The use of our TSR algorithms allows processing of video streams in real-time with high resolution, and therefore at greater distances and with better quality than similar TSR systems have. FullHD resolution makes it possible to detect and recognize a traffic sign at a distance up to 50 m.