**Model Building**

import numpy as np

import pandas as pd

import matplotlib.pyplot as pl t#to plot accuracy

import cv2

import tensorflow as tf

from PIL import Image

import os

from sklearn.model\_selection import train\_test\_split #to split training and testing data from tensorflow.keras.utils import to\_categorical

#to convert the labels present in y\_train and t\_test into one-hot encoding

from keras.models import Sequential, load\_model

from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout#to create CNN data = []

labels = []

classes = 43

cur\_path = os.getcwd()

#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:

print("Error loading image") #Converting lists into numpy arrays data = np.array(data) labels = np.array(labels) print(data.shape, labels.shape)

#Splitting training and testing dataset

X\_t1, X\_t2, y\_t1, y\_t2 = train\_test\_split(data, labels, test\_size=0.2, random\_state=42) print(X\_t1.shape, X\_t2.shape, y\_t1.shape, y\_t2.shape) #Converting the labels into one hot encoding y\_t1 = to\_categorical(y\_t1, 43) y\_t2 = to\_categorical(y\_t2, 43) #Building the model model = Sequential()

model.add(Conv2D(filters=32,kernel\_size=(5,5),activation='relu',input\_shape=X\_t1.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'))

#Compilation of the model model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']) epochs = 5 history = model.fit(X\_t1, y\_t1, batch\_size=32, epochs=eps, validation\_data=(X\_t2, y\_t2)) model.save("my\_model.h5") #plotting graphs for 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() 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 accuracy on test dataset from sklearn.metrics import accuracy\_score 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) predict\_x=model.predict(X\_test) classes\_x=np.argmax(predict\_x,axis=1) #Accuracy with the test data from sklearn.metrics import accuracy\_score print(accuracy\_score(labels, classes\_x)) model.save("traffic\_classifier.h5") **#User interface** import numpy

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

#load the trained model to classify sign from keras.models import load\_model model = load\_model('traffic\_classifier.h5') #dictionary to label all traffic signs class. classes = { 1:'Speed limit (20km/h)',

2:'Speed limit (30km/h)',

3:'Speed limit (50km/h)',

4:'Speed limit (60km/h)',

5:'Speed limit (70km/h)',

6:'Speed limit (80km/h)',

7:'End of speed limit (80km/h)',

8:'Speed limit (100km/h)',

9:'Speed limit (120km/h)',

10:'No passing',

11:'No passing veh over 3.5 tons',

12:'Right-of-way at intersection',

13:'Priority road',

14:'Yield',

15:'Stop',

16:'No vehicles',

17:'Veh > 3.5 tons prohibited',

18:'No entry',

19:'General caution',

20:'Dangerous curve left',

21:'Dangerous curve right',

22:'Double curve',

23:'Bumpy road',

24:'Slippery road',

25:'Road narrows on the right',

26:'Road work',

27:'Traffic signals',

28:'Pedestrians',

29:'Children crossing',

30:'Bicycles crossing',

31:'Beware of ice/snow',

32:'Wild animals crossing', 33:'End speed + passing limits',

34:'Turn right ahead',

35:'Turn left ahead',

36:'Ahead only',

37:'Go straight or right',

38:'Go straight or left',

39:'Keep right',

40:'Keep left',

41:'Roundabout mandatory',

42:'End of no passing',

43:'End no passing vehicle with a weight greater than 3.5 tons' }

#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 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) predict\_x=model.predict([image])[0] classes\_x=numpy.argmax(predict\_x,axis=0) print(predict\_x) classes\_x=numpy.argmax(predict\_x,axis=0) sign = classes[classes\_x+1] print(sign) label.configure(foreground='#011638', text=sign) 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) 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="check traffic sign",pady=20, font=('arial',20,'bold')) heading.configure(background='#CDCDCD',foreground='#364156') heading.pack() top.mainloop()