ДОДАТОК А

Вихідний код програми

traffic_sign.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, Dense, Flatten, Dropout,
MaxPool2D

data = []
labels = []
classes = 43
cur_path = os.getcwd()

print("Retrieving")
#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((32,32))
            image = np.array(image)
```

```
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")

print("Converting lists into numpy arrays")
#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

print(data.shape, labels.shape)
print("Splitting training and testing dataset")
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data,
labels, test_size=0.1, random_state=42)

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

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


print("Building the model")
#Building the model
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(5,5),
activation='relu', input_shape=(32, 32, 3), padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
Dropout(0.2)
```

```
model.add(Conv2D(filters=64, kernel_size=(3,3),
activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
Dropout(0.2)

model.add(Conv2D(filters=128, kernel_size=(3,3),
activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
Dropout(0.2)

model.add(Flatten())
model.add(Dense(128, activation='relu'))
Dropout(0.3)
model.add(Dense(classes, activation='softmax'))

#Compilation of the model
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])


epochs = 15
history = model.fit(X_train, y_train, batch_size=16,
epochs=15, validation_data=(X_test, y_test))
model.summary()
model.save("my_classifier.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((32,32))
    data.append(np.array(image))

X_test=np.array(data)

pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
print('Accuracy score: ')
print(accuracy_score(labels, pred))
```

detection.py

```
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import cv2 as cv
PATH_TO_MODEL = 'frozen_inference_graph.pb'

def detection(image_np, detection_graph, sess):
    image_np_expanded = np.expand_dims(image_np, axis=0)
    #Gets the detection graph tensors
    image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
    boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
    scores =
detection_graph.get_tensor_by_name('detection_scores:0')
    classes =
detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

    #Executes the detection
    (boxes, scores, classes, num_detections) =
sess.run([boxes, scores, classes, num_detections],

feed_dict={image_tensor: image_np_expanded})

    return boxes, scores

#Filters detections with less than 20% probability
def filter_detections(detected, boxes, scores, w, h):
    count = 0 #Amount of valid detections
```

```
    for box, score in zip(np.squeeze(boxes),
np.squeeze(scores)): #Bounding box and score from each
detection
        if score*100 > 20: #>20%
            box[0], box[1], box[2], box[3] = box[0]*h,
box[1]*w, box[2]*h, box[3]*w #Normalized bounding boxes
coordinates
            detected.append(box) #Append to list with filtered
detections
            count += 1
    return detected, count


def detect(model, image):
    detection_graph = tf.Graph()

    config = tf.compat.v1.ConfigProto()
    config.gpu_options.allow_growth = True

    with detection_graph.as_default(): #Loads the detection
model (frozen graph)
        od_graph_def = tf.compat.v1.GraphDef()
        fid = tf.io.gfile.GFile(PATH_TO_MODEL, 'rb')
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

        sess = tf.compat.v1.Session(graph=detection_graph,
config=config) #Initializes the session
        w, h = image.size
        detected = []

        image_np = np.array(image.getdata()).reshape((h, w,
3)).astype(np.uint8) #Image to numpy array
```

```
        #Detect and filter detections for the normal image
        boxes, scores = detection(image_np, detection_graph,
sess)
        detected, num_detections = filter_detections(detected,
boxes, scores, w, h)

        #Predicts which traffic sign each detection is
        pred = predict(image_np, model, detected,
num_detections)

    sign = classes[pred+1]
    return sign

#Function for predictions.
#Parameters: image, the recognition model and a list of
detection boxes.
def predict(img, learn_inf, boxes, num_detections):
    image = cv.cvtColor(img, cv.COLOR_RGB2BGR)#Transforming
image from RGB to BGR because of OpenCV
    for box in boxes: #Each detection box in the list
        yi, yf, xi, xf = int(box[0]), int(box[2]),
int(box[1]), int(box[3])#bounding box coordinates in the image
        sign = img[yi:yf, xi:xf]#bounding box content to
variable
        sign = cv.resize(sign, (32,32))
        pred = learn_inf.predict_classes([sign])[0]
```

gui.py

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import detection as dt
import numpy
```

```
#load the trained model to classify sign
from keras.models import load_model
model = load_model('my_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 veh > 3.5 tons' }

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

label=Label(top, font=('roboto',16,'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    detection(model, image)
    print(sign)
    label.configure(foreground='#011638', text=sign)


def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Sign",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="Know Your Traffic Sign",pady=20,
font=('arial',20,'bold'))
heading.configure()
heading.pack()
top.mainloop()
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	82 с.
	Інші документи	
2.	Презентаційні матеріали	плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		КоноваловО.Ю.			Розробка підсистеми розпізнавання дорожніх знаків автомобільного автопілота
Перевірив		Вітько О.В.			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІм-19-2	122
ХНУРЕ кафедра ШІ	