import matplotlib.pyplot as plt

import datetime

import numpy as np

from keras.models import load\_model

from PIL import Image, ImageOps

import pytz # Import pytz module to work with time zones

def load\_and\_preprocess\_image(image\_path):

# Load the image

image = Image.open(image\_path).convert("RGB")

# Resize and crop the image to 224x224

size = (224, 224)

image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# Convert the image to a numpy array

image\_array = np.asarray(image)

# Normalize the image

normalized\_image\_array = (image\_array.astype(np.float32) / 127.5) - 1

# Create the array of the right shape to feed into the keras model

data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

data[0] = normalized\_image\_array

return data

def predict\_image(image\_data, model\_path, labels\_path):

# Load the model

model = load\_model(model\_path, compile=False)

# Load the labels

class\_names = open(labels\_path, "r").readlines()

# Predict using the model

prediction = model.predict(image\_data)

index = np.argmax(prediction)

class\_name = class\_names[index]

confidence\_score = prediction[0][index]

return class\_name, confidence\_score

def record\_and\_analyze\_detections(image\_data, class\_name, model\_path, labels\_path):

class\_name, confidence\_score = predict\_image(image\_data, model\_path, labels\_path)

# Print prediction and confidence score

print("Class:", class\_name[2:])

print("Confidence Score:", confidence\_score)

def record\_detection\_times(class\_name):

detection\_times = []

while True:

user\_input = input(f"Enter 'detected' for {class\_name[2:]}, 'stop' to end, or 'image' to change the image: ").strip().lower()

if user\_input == 'stop':

return detection\_times

elif user\_input == 'image':

return None

elif user\_input == 'detected':

# Get current time in Indian time zone and append it to detection\_times

tz = pytz.timezone('Asia/Kolkata') # Indian Standard Time

current\_time = datetime.datetime.now(tz).replace(tzinfo=None) # Remove timezone information

detection\_times.append((class\_name[2:], current\_time.astimezone(pytz.utc)))

print(f"Detection recorded for {class\_name[2:]} at:", current\_time)

def analyze\_detection\_times(detection\_times):

half\_hour\_counts = {}

start\_time = datetime.datetime.combine(datetime.date.today(), datetime.time(0, 0)).replace(tzinfo=pytz.utc)

half\_hour = datetime.timedelta(minutes=30)

for i in range(48): # 24 hours \* 2 intervals per hour

end\_time = start\_time + half\_hour

half\_hour\_counts[start\_time.strftime('%H:%M')] = sum(1 for dt in detection\_times if start\_time <= dt[1] < end\_time)

start\_time = end\_time

max\_half\_hour = max(half\_hour\_counts, key=half\_hour\_counts.get)

return half\_hour\_counts, max\_half\_hour

def plot\_detection\_analysis(half\_hour\_counts, max\_half\_hour):

plt.bar(half\_hour\_counts.keys(), half\_hour\_counts.values(), alpha=0.5)

plt.xlabel('Half-hour Intervals')

plt.ylabel('Number of Detections')

plt.title('Object Detection Analysis')

plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability

plt.axvline(x=max\_half\_hour, color='r', linestyle='--', label=f'Most Detected Interval ({max\_half\_hour})')

plt.legend()

plt.tight\_layout() # Adjust layout to prevent clipping of labels

plt.show()

while True:

detection\_times = record\_detection\_times(class\_name)

if detection\_times is None:

break

if detection\_times:

half\_hour\_counts, max\_half\_hour = analyze\_detection\_times(detection\_times)

print("Most detected half-hour interval:", max\_half\_hour)

plot\_detection\_analysis(half\_hour\_counts, max\_half\_hour)

else:

print("No detection times recorded.")

break

if \_name\_ == "\_main\_":

# Example usage:

# Replace placeholders with actual paths to the model and labels

model\_path = r"/content/keras\_model.h5"

labels\_path = r"/content/labels.txt"

while True:

image\_path = input("Enter the path to the image file: ")

image\_data = load\_and\_preprocess\_image(image\_path)

class\_name, \_ = predict\_image(image\_data, model\_path, labels\_path)

record\_and\_analyze\_detections(image\_data, class\_name, model\_path, labels\_path)