**SWS3009 Summer Workshop**

**Lab 7 – Answer Book**

**Submission Deadline: Saturday 6 July 2024 2359**

Name 1: YANG RUNKANG

Name 2: WEN SIJIE

Name 3:

**QUESTION 1.**

The stray “b” in the output stands for bytes. In Python 3, MQTT messages are received as byte strings, hence the “b” prefix indicating that the message is in bytes format. This is because the payload of MQTT messages is treated as a byte array by the MQTT library.

**QUESTION 2.**

import paho.mqtt.client as mqtt

def on\_connect(client, userdata, flags, rc):

    print("Connected with result code " + str(rc))

    client.subscribe("hello/#")

def on\_message(client, userdata, msg):

    print(msg.topic + " " + msg.payload.decode('utf-8'))

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

print("Connecting")

client.connect("localhost", 1883, 60)

client.loop\_forever()

**QUESTION 3.**

To ensure that the model is loaded only once and reused for each classification, the model is loaded globally at the start of the main() function and stored in a global variable model. This avoids repeated loading delays and ensures efficient handling of classification tasks. By doing this, the model is loaded into memory when the program starts and remains there, ready to classify any incoming data. The global keyword is used to ensure that the model variable accessed inside the main() function refers to the global variable model declared at the top of the script, not to a local one.

**QUESTION 4.**

def classify\_flower(filename, data):

    global model

    print("Start classifying.")

    result = model.predict(data)

    win = int(np.argmax(result))

    score = float(result[0][win])

    print("Done.")

    return {"filename": filename, "prediction": classes[win], "score": score, "index": win}

Global Model Reference: The function starts by referencing the global model variable to ensure it uses the model loaded in main().

Model Prediction: It uses the model.predict(data) method to get the prediction probabilities for the input data. The data should be pre-processed in the same way it was when training the model.

Finding the Best Prediction: np.argmax(result) is used to find the index of the highest probability output, which corresponds to the most likely class. This index is used to fetch the class name from the predefined classes list.

Extracting Score: It also extracts the probability score of the winning class to return how confident the model is about its prediction.

Return Structure: Finally, it constructs and returns a dictionary containing the filename, the predicted class, the confidence score, and the class index.

**QUESTION 5.**

from os import listdir

from os.path import join

PATH = "./samples"

def main():

    client = setup("127.0.0.1")

    print("Sending data.")

    # send\_image(client, "tulip.jpg")

    # print("Done. Waiting for results")

    for file in listdir(PATH):

        filename = join(PATH, file)

        send\_image(client, filename)

    while True:

        pass

Directory Listing: We can use os.listdir() to retrieve all entries in the "samples" directory. It constructs the full path for each file using os.path.join(), ensuring the path is correctly formed regardless of the operating system.

Sending Images: Within the loop, each file path is passed to the send\_image function, which was previously defined to handle image preprocessing and sending via MQTT.

**QUESTION 6.**

The output (accuracy) of my classifier:

Received message from server.

Prediction: tulips, Score: 0.9803

Received message from server.

Prediction: roses, Score: 0.9842

Received message from server.

Prediction: sunflowers, Score: 0.5109

Received message from server.

Prediction: tulips, Score: 0.9180

Received message from server.

Prediction: roses, Score: 0.9946

Received message from server.

Prediction: dandelion, Score: 0.8113

Received message from server.

Prediction: sunflowers, Score: 0.9920

Received message from server.

Prediction: tulips, Score: 0.9977

Received message from server.

Prediction: tulips, Score: 0.9803

Received message from server.

Prediction: tulips, Score: 0.9943

Received message from server.

Prediction: sunflowers, Score: 0.9774

Received message from server.

Prediction: daisy, Score: 0.4470

Received message from server.

Prediction: daisy, Score: 0.6328

Received message from server.

Prediction: roses, Score: 0.9918

Received message from server.

Prediction: sunflowers, Score: 0.9993

Received message from server.

Prediction: dandelion, Score: 0.6052