UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2513 ARTIFICIAL INTELLIGENCE

TITLE: HELP SIGN DETECTION SYSTEM

GROUP NAME: SIGNAL GUARD

MATRIC ID	NAME	SECTION
SD22059	LOW ANN GIE	01G
SD22061	ANG MEI YING	01G
SD22015	LIM JON VINCE	02G
SD22029	ALENA NG PEI YEEN	01G
SD22031	ANIS AQILAH BINTI MOHD ASRI	02G

Table of Contents

1.0 Executive Summary

1.1 Description of the selected project

The “Signal for Help” is a single-handed gesture that can be used by an individual to alert others that they feel threatened and need help over a threatening situation. The signal is performed by holding your hand up with your thumb tucked into your palm, then folding your fingers down, symbolically trapping your thumb in your fingers. The signal for help was first introduced in Canada by Canadian Women's Foundation in year 2020 to tackle the growing problem of domestic violence on that time. It provides an alternative way for people in danger to get help quickly and quietly, especially when they cannot ask for help aloud. As a result, it is important for everyone to know universal help sign language in our world today. Sign language is not just for disabled person, yet it is for everyone since we will never know when we might need to use it to keep ourselves safe.

Inspired by that situation, in this project we want to develop a system that can spot “Help Sign” hand gesture if students signal for help with their hand. We propose implementing real-time video analysis utilizing hand gestures recognition and artificial intelligence. This system's aim is to detect students who show they need some help with a “Help Sign” hand gesture. This way, we can quickly help the students when something bad happens to avoid a worse situation. However, it is important to understand that making the help sign does not guarantee your safety. Instead, it is like a tool to seek help from people nearby so they can act quickly.

Figure 1: Universal ”Help Sign” hand gestures

“Help sign detection” is an artificial intelligence analytics solution that employs algorithms and machine learning techniques to differentiate between normal hand movements and the specific gesture indicating a need for help. The technology evaluates the configuration of a person's hand

to swiftly identify instances where the help sign gesture is made, even amidst the noise and activity of a laboratory and student pantry environment. Behind the scenes, machine learning algorithms and reference models continuously refine the detection process for accuracy and reliability. This system can quickly alert laboratory assistants and fellow on duty for help.

Implementing helps sign hand gesture detection, makes action towards emergencies easier and faster.

Therefore, this approach aims to develop a method that not only accurately detects the “help sign" hand gestures in real-time video streams but also ensures scalability, automation, and reliability. Our group aims to minimize computational costs, enhance computational speed, and improve model accuracy to solve our problems.

1.2 Problem to be solved

The problems to be solved:

1) How can security measures on university campuses be improved to swiftly detect help signals in different situations, ensuring quick assistance during emergencies?

2) How can we reduce response time and improve the effectiveness of identifying and responding to emergencies?

3) What steps can be taken to design a detection system that is scalable and adaptable to university environments, integrating technological advancements to meet evolving needs?

1.3 Basic Description of the data selected

Our team has captured a collection of photos which were taken by our members themselves. We have three unique photographs in the photo collection. Figure 1 shows the hand gesture for sign B which is palm to camera and tuck thumb. Figure 2 shows the hand gesture for sign A where the thumb is trapped. Lastly, Figure 3 and 4 shows the "other" hand gesture such as "High-Five" and "Peace" which is not Sign A or Sign B.

In order to expand our capabilities, we developed a live streaming system that enables us to monitor help signs in real time. We employ the screen recording capability to capture the current situation as an illustration while using live streaming video, allowing us to identify and monitor "Help Sign" hand gesture in real-time, delivering crucial insights and recommending appropriate actions.

Figure 1: sign B

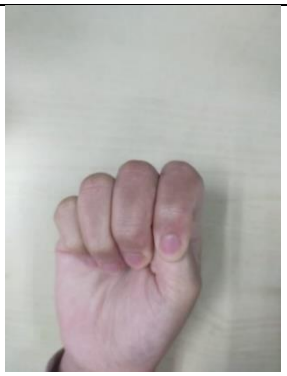


Figure 2: sign A

Figure 3: "other" hand gesture

Figure 4: "other" hand gesture

2.0 Summary of the Project Context and Objectives

2.1 Summary of the Project Context

In order to solve the problem of detecting help signs among people in busy public settings, the project focuses on developing an automated hand identification system that makes use of hand gestures recognition and artificial intelligence. The system analyses hand features and makes a distinction between people who are showing help signs and vice versa using algorithms and deep learning techniques. The solution is built to improve scalability and automation by automating the process and doing away with the requirement for manual identification.

This project aims to develop a comprehensive method that can accurately detect help sign in photos and real-time videos. The system should be able to identify different types of hand sign and demographic data while achieving high accuracy in distinguishing between help sign and normal sign. Real-time processing of video feeds requires efficient algorithms and optimizations to deliver timely results.

The project's dataset for data collection is made up of images that team members have taken. The images display individuals showing Sign A, Sign B and "other" hand sign to let the system detect the help sign. Sign A is typically referring to hand gesture where the thumb is trapped, Sign B is hand gesture, which is palm to camera and tuck thumb, while "other" sign refers to hand gestures that are other than sign A and sign B. For this "other" sign, it will not trigger any help sign detection.

Overall, the background of the project revolves around the urgent need for an automated help sign detection system that can accurately identify students who show help signs in real-time photos. By utilizing help signs recognition and artificial intelligence, the system aims to enhance the speed of rescue in real-time accidents that occur around the university.

2.2 Objectives

The objectives of the project are:

1. To enhance security measures

By implementing a reliable detection system to swiftly detect help signal in different circumstances, ensuring prompt assistance in emergency situations.

2. Improve response time

Utilize help-sign recognition and artificial intelligence techniques to reduce the time taken in identifying normal sign and help-sign and respond quickly in an emergency.

3. To ensure scalability and adaptability

Design a flexible detection system capable of accommodating the evolving needs of university environments by integrating with the advancements in technology.

3.0 Methodology

3.1 Data Collection and Preparation

To develop an accurate and reliable “Help Sign Detection” system, we collected and prepared a comprehensive images dataset. The team members actively captured a robust dataset for detecting hand gestures representing the sign “A”, sign “B” and “Other” hand gestures.

Our dataset includes a diverse range of images, each depicting the hand gestures for the sign "A", sign “B” and “Other” in various movements and positions. These images were captured from different angles and positions to ensure the system can handle various scenarios and offer accurate detection capabilities.

Once the datasets were collected, we underwent a thorough preparation process. We checked every image to ensure clarity and that our hand was fully visible. It's crucial for our hand to appear in each picture. If there's any photo where our hand is missing due to the camera's rapid capture speed within 100 milliseconds, we’ll need to collect the data again. After collecting the data, each image was annotated and labeled to identify and mark the hand gestures for the letters "A” and “B”. The prepared dataset is used to train our Help Sign Detection algorithms. The annotations help the model accurately learn and distinguish between the hand gestures for sign "A", sign "B" and “Other” hand gestures.

By utilizing this well-prepared dataset, our project aims to develop a system that can reliably detect hand gestures representing the combination of sign "B", sign “A” in images, contributing to rapid action towards emergencies and hazard case.

3.2 Coding of Project without GUI

In this project, we are going to create a coding to detect help sign using PyCharm. Before we starting, there are a few libraries that need to be installed into PyCharm for later use. The list of libraries are as following:

- Os - This library provide functions for file and directory manipulation which help us to save the captured image during data collection.
- OpenCv Python - This library is used for video capture and image processing.
- Pickle Module – This library is used for serializing and de-serializing Python object structures. In this case, we used for serialized the collected data into a file called "AI.pickle" and deserialized the content during the traning stage.
- Mediapipe - This library is used to detect the landmarks of the hands for an image or video.
- Tkinter - This library is used to construct basic graphical user interface (GUI) applications.
- PLT - This library provides support for opening, manipulating, and saving many different image file.
- Numpy - This library is used to perform mathematical operations on arrays.
- Time - This library is used to set up the time of displaying the message for help sign detection and controlling alarm duration.
- Pygame - This library is used to play the alarm sound when a help sign is detected.
- Threading - This library allow same coding to run concurrently at different detection system in GUI.
- Scikit-learn - This library is used to create a Random Forest Classifier during training stage.

```
import os
import cv2
import pickle
import mediapipe as mp
import tkinter as tk
from PIL import ImageTk, Image
import numpy as np
import time
import pygame
import threading
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

The image shown that the libraries and packages are used in this project

3.2.1 Data collection

After importing all the necessary libraries, we start to collect the data. Before that, we need to create an empty file named AI image collection then copy its path and assigned it to the variable named “DATA_DIR”. The following coding uses “os.path.exists” function to check whether the file is existing or not:

```
DATA_DIR = "C:\\UMPSA_Y2S2\\BSD2513 AI\\AI Project\\AI image collection"
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)
```

Next, we defined two variables named “number_of classes” and “dataset_size”. The “Number_of classes = 3” means that the data is collected in 3 sets of datasets which is sign “A”, sign “B” and “others” sign while “dataset_size=100” represented that each dataset contains 100 samples.

```
number_of_classes = 3
dataset_size = 100
```

Besides, we also had set the width and height of the frames captured from camera.

```
frame_width = 640
frame_height = 480
```

Next, we are using OpenCV library to make a connection with the camera and prepare it for image capture.

```
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, frame_width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, frame_height)
```

Then, we use for loop to distribute the total number of 300 images separately at 3 different files for each class, with each class having 100 images. Started with file “0” which store the data collected from class “0”, followed by file “1” which store the data collected from class “1” and lastly file “2” which store the data collected from class “2”.

```
for j in range(number_of_classes):
    if not os.path.exists(os.path.join(DATA_DIR, str(j))):
        os.makedirs(os.path.join(DATA_DIR, str(j)))

    print('Collecting data for class {}'.format(j))

    done = False
```

The following coding show the process of capturing image from camera:

```
while True:
    ret, frame = cap.read()
    cv2.putText(frame, text: 'Ready? Press "q" ! :)', org: (100, 50), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1.3, color: (0, 255, 0),
                thickness: 3,
                cv2.LINE_AA)
    cv2.imshow( winname: 'frame', frame)
    if cv2.waitKey(25) == ord('q'):
        break
```

This coding will display a message which is "Ready?Press 'q' ! :)". When q is pressed, it will terminate from the looping.

We have initialise the counter equal to 0. It creates a loop that continuosly captures the images from the camera until the counter reaches the size of dataset (100). We are using "cv2.waitKey" function to capture an image within 1000 milliseconds (about 1 second). In the end, the images will stored in separate file for each class in a format of JPEG image.

```
counter = 0
while counter < dataset_size:
    ret, frame = cap.read()
    cv2.imshow( winname: 'frame', frame)
    cv2.waitKey(1000)
    cv2.imwrite(os.path.join(DATA_DIR, str(j), '{}.jpg'.format(counter)), frame)

    counter += 1
```

Lastly, we release the capture images and delete all OpenCV windows by using following commands:

```
cap.release()
cv2.destroyAllWindows()
```

Output for this data collection stage:

```
Collecting data for class 0
Collecting data for class 1
Collecting data for class 2
```

3.2.2 Create Database

After data collection is done, we proceed to the next step which is creating a database named "AI.pickle". This stage is important to store our hand landmark data in a structured format, makes it easy to be access and utilize for training machine learning models.

We begin this stage by calling the modules from MediaPipe. Firstly, "mp.solutions.hands" is used for hand tracking. Secondly, "mp.solutions.drawing_utils" is used to draw the hand landmarks at each joint and makes connections on image. Thirdly, "mp.solutions.drawing_styles" is used to modify the drawing styles like colors and line thickness to improve the visual performance.

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
```

Next, we assign "DATA_DIR" with the path to the files where the images are stored. We also create two empty lists and define them as "data" and "labels" to be used during for loop processing. The "data" will collect the hand landmarks data that extracted from the images while "labels" will hold the corresponding labels of class identifiers or categories for each image. After that, we use nested for loop to iterate over each files inside the "DATA_DIR" and the inner loop iterates over each image in the targeted file. Inside the looping, we define some variables as empty lists which are "data_aux", "x_" and "y_". The "data_aux" temporarily holds the hand landmark data that extracted from each image while "x_" and "y_" are used to collect the x and y coordinates of these landmarks.

```
DATA_DIR = "C:\\UMPSA_Y2S2\\BSD2513 AI\\AI Project\\AI image collection"

data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []

        x_ = []
        y_ = []
```

The following coding is used to read the images from the files and convert them to RGB (Red, Green,Blue) color space:

```
img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Next, we need to process the RGB image and detect hand landmarks. Firstly, we use a loop process to iterate over each detected hand in the image if multiple hands are present in the image. Additionally, we also need another loop to iterate over each landmark for the detected hand. The following coding show the process of collecting all landmark coordinates:

```
results = hands.process(img_rgb)
if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        for i in range(len(hand_landmarks.landmark)):
            x = hand_landmarks.landmark[i].x
            y = hand_landmarks.landmark[i].y

            x_.append(x)
            y_.append(y)
```

Besides that, the following coding show the processes for each detected landmark point within a hand:

```
for i in range(len(hand_landmarks.landmark)):
    x = hand_landmarks.landmark[i].x
    y = hand_landmarks.landmark[i].y
    data_aux.append(x - min(x_))
    data_aux.append(y - min(y_))
```

From that, we had normalised the coordinates of each landmark point by substracting the minimum x and y coordinates respectively. Eventually, append the normalized x and y coordinates to the "data_aux" list.

Next, we append the “data_aux” to the “data” list and “dir_” to the “labels” list.

```
data.append(data_aux)
labels.append(dir_)
```

At the end of this stage, we save the processed “data” and “labels” into a database named “AI.pickle” and use it for further analysis.

```
f = open('AI.pickle', 'wb')
pickle.dump( obj: {'data': data, 'labels': labels}, f)
f.close()
```

Output for this stage:

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

3.2.3 Training Model

After creating a database, we proceed to the next step which is training model. In this stage, we are going to train a Random Forest classifier model to recognize hand gestures and classified them accurately based on extracted features from the images.

Firstly, we open the "AI.pickle" file in binary read mode and loads its content using following coding:

```
data_dict = pickle.load(open('./AI.pickle', 'rb'))
```

Next, we convert the data in lists "data" and "labels" into arrays using Numpy library as shown below:

```
data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])
```

After that, we split the arrays form "data" and "labels" into training and testing sets:

```
x_train, x_test, y_train, y_test = train_test_split( *arrays: data, labels, test_size=0.2, shuffle=True, stratify=labels)
```

As we see from this coding, "test_size=0.2" means that only 20% of data will be used for testing set and the remaining 80% will be used for traning. Additionally, we set "shuffle=True" to ensure that the data is randomly distributed while "stratify=labels" is used to ensure both training and testing sets have same proportion of each class label as the original dataset.

Once the splitting is done, we need to create a "RandomForestClassifier" and train it with the training data. This process is to make the classifier to look for the patterns in the data and learn from them. We create a Random Forest Classifier model using following command:

```
model = RandomForestClassifier()
```

Next, we fixed the model with training data:

```
model.fit(x_train, y_train)
```

This coding can help to train model by finding patterns in the training data.

Eventually, we used the trained model to make predictions on testing data. This coding will return the predicted class labels for the test data.

```
y_predict = model.predict(x_test)
```

Furthermore, we also evaluate the performance of model by calculating the accuracy of model. This is done by comparing the predicted result (y_predict) with testing data (y_test).

```
score = accuracy_score(y_predict, y_test)
```

Next, we print out the accuracy of the model in percentage.

```
print('{}% of samples were classified correctly !'.format(score * 100))
```

In the end of the training stage, we open a file named “RFmodel.p” and serializes the trained model into the file. Lastly, close the file using “f.close()” function.

```
f = open('RFmodel.p', 'wb')
pickle.dump( obj: {'model': model}, f)
f.close()
```

Output for this stage:

```
100.0% of samples were classified correctly !

Process finished with exit code 0
```

3.2.4 Inference Classifier

After training model, we proceed to the next stage which is inference classifier. In this stage, we will use the trained model to make predictions and classification on new data. In this project, we using trained Random Forest Classifier model to detect and classify the hand gesture in real-time video frames captured by a webcam.

Firstly, we reset the frame size with width 800 pixels and height 600 pixels using following command:

```
new_width = 800
new_height = 600
```

Next, we load the trained model from the file "RFmodel.p":

```
model_dict = pickle.load(open('./RFmodel.p', 'rb'))
model = model_dict['model']
```

Moreover, the following code block represented the importing the MediaPipe Hands module (hand landmark detection), drawing utilities module (draw connection and hand landmarks) and drawing styles from MediaPipe. We use static image processing rather than real-time video processing and set the minimum detection confidence and minimum tracking confidence as 0.8. In addition, maximum two hands are available for detection.

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.8, min_tracking_confidence=0.8, max_num_hands=2)
```

Next, we assign a name for the dataset "0" as "B", dataset "1" as "A" and dataset "2" as "other" sign languages:

```
labels_dict = {0: 'B', 1: 'A', 2: 'other'}
```

After that, we initialize the previous sign and help timer start as None, help detection as False, help display duration as 3 and alarm duration as 5:

```
prev_sign = None
help_detected = False
help_timer_start = None
help_display_duration = 3
alarm_duration = 5
```

Next, the following command show that alarm sound file is assigned to store mp3 file path:

```
alarm_sound_file = "C:\\UMPSA_Y2S2\\BSD2513 AI\\AI Project\\ALARM SOUND EFFECT.mp3"
```

Furthermore, we use “pygame.mixer.init()” function to initialise the audio mixer module. This module allows us to control and play the sound. Additionally, the alarm start time is assigned to None.

```
pygame.mixer.init()
alarm_start_time = None
```

Next, the following coding shows that when “initialize_camera()” function is called, it will return to camera to capture the object:

```
def initialize_camera():
    return cv2.VideoCapture(0)
```

We have declared the previous sign, help detected, help timer start and alarm start time as global variable:

```
def start_detection(camera):
    global prev_sign, help_detected, help_timer_start, alarm_start_time
```

Next, we intialised the “data_aux”, “x_” and “y_” with empty list to store hand landmarks data. The “camera.read()” captured a frame from camera and stored it in variable “frame” while the “ret” indicates whether the frame was captured successfully or not. We also used “cv2.resize” to resize the frame. Additionally, the “cv2.cvtColor” is used to convert the frame from BGR (Blue, Green, Red) to RGB (Red, Green, Blue). The order of the colours in OpenCV is BGR while in Pillow the order of colours is RGB. In addition, many machine learning models will expect the image in RGB colour space. Therefore, we need to convert it.

```
while True:
    data_aux = []
    x_ = []
    y_ = []

    ret, frame = camera.read()

    # Resize the frame
    frame = cv2.resize(frame, dsize: (new_width, new_height))

    H, W, _ = frame.shape

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

From the coding below, “if results.multi_hand_landmarks” command is used to check whether the hand landmarks are detected or not in the frame while “draw_landmarks” is used to draw the detected hand landmarks and its connections.

```
results = hands.process(frame_rgb)
if results.multi_hand_landmarks:
    for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            frame,  # image to draw
            hand_landmarks,  # model output
            mp_hands.HAND_CONNECTIONS,  # hand connections
            mp_drawing_styles.get_default_hand_landmarks_style(),
            mp_drawing_styles.get_default_hand_connections_style())
```

Next, “x” and “y” are assigned to identify the coordinate of each hand landmarks and store them in “x_” and “y_”:

```
for hand_landmarks in results.multi_hand_landmarks:
    for i in range(len(hand_landmarks.landmark)):
        x = hand_landmarks.landmark[i].x
        y = hand_landmarks.landmark[i].y

        x_.append(x)
        y_.append(y)
```

The following coding show that we used “x – min(x_)” and “y – min(y_)” to calculate the relative positions of hand landmarks and eventually store them in “data_aux”:

```
for i in range(len(hand_landmarks.landmark)):
    x = hand_landmarks.landmark[i].x
    y = hand_landmarks.landmark[i].y
    data_aux.append(x - min(x_))
    data_aux.append(y - min(y_))
```

Next, the following code of block draw the bounding boxes around each detected hand:

```
for hand_landmarks in results.multi_hand_landmarks:
    hand_landmarks_list = []
    for landmark in hand_landmarks.landmark:
        hand_landmarks_list.append((int(landmark.x * W), int(landmark.y * H)))

    x1 = min(hand_landmarks_list, key=lambda x: x[0])[0]
    y1 = min(hand_landmarks_list, key=lambda x: x[1])[1]
    x2 = max(hand_landmarks_list, key=lambda x: x[0])[0]
    y2 = max(hand_landmarks_list, key=lambda x: x[1])[1]
```

After that, we make prediction the hand sign using trained model:

```
prediction = model.predict([np.asarray(data_aux)])
```

This code of block is used to check the “A” sign and “B” sign while other hand sign is assigned as other category:

```
if int(prediction[0]) in labels_dict:
    detected_sign = labels_dict[int(prediction[0])]
else:
    detected_sign = labels_dict[2]  # Assign "Other" category for other letters
```

Next, the percentage of confidence score is calculated and assigned as “confidence_score” variable. The “cv2.putText” function is used to display the confidence score percentage.

```
confidence_score = 100*results.multi_handedness[0].classification[0].score

# Overlay confidence score on the frame
cv2.putText(frame, text: f"{confidence_score:.2f}%", org: (x1, y1 - 35), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.7,
            color: (255, 255, 255), thickness: 2)
```

This code of block checking whether prev_sign equal to “B” and the detected_sign equal to “A”. If achieve this requirement, it will set the “help_detected” equal to true and “help_timer_start” equal to current time in seconds. Besides, the “help_timer_start” is used to check whether it has assigned by other value and “time.time() – help_timer_start >= help_display_duration” check whether the time has passed. Since the prev_sign is set as None, it will be None during the first iteration. After the first iteration, “prev_sign” will equal to the previously detected sign.

```
if prev_sign == 'B' and detected_sign == 'A':
    help_detected = True
    help_timer_start = time.time()
elif help_timer_start is not None and time.time() - help_timer_start >= help_display_duration:
    help_detected = False
    help_timer_start = None

# Update previous detected sign
prev_sign = detected_sign
```

If help sign is detected, a rectangle with “Help sign detected!” will display on the frame. If the alarm is not already playing, “pygame.mixer.music.load(alarm_sound_file)” is used to load the alarm file and playing the alarm in infinite looping with “pygame.mixer.music.play(-1)” command. If the help sign is not detected, it will displace a rectangle with a word or letter that corresponds to hand signs.

```
if help_detected:
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 4)
    cv2.putText(frame, text: "Help sign detected!", org: (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1, color: (0, 0, 255),
                thickness: 2)
    if not pygame.mixer.music.get_busy():  # Check if alarm is not already playing
        pygame.mixer.music.load(alarm_sound_file)
        pygame.mixer.music.play(-1)
    alarm_start_time = time.time()
else:
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 4)
    cv2.putText(frame, detected_sign, org: (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1, color: (0, 255, 0), thickness: 2)
    pygame.mixer.music.stop()
```

If the help sign is not detected or the help sign detected but the duration of alarm play more than “alarm_duration”, then the alarm will be stopped playing:

```
if (help_detected and time.time() - alarm_start_time >= alarm_duration) or not help_detected:
    pygame.mixer.music.stop()
```

If “q” key is pressed, the camera frame will be terminated:

```
cv2.imshow( winname: 'frame', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

The camera is released and close all OpenCV windows:

```
camera.release()
cv2.destroyAllWindows()
```

The code of block is used to ensure the same coding accessed by multiple threads and avoid concurrency issues occurs:

```
def start_detection_thread(btn_num):
    # Start detection in a separate thread
    camera = initialize_camera()
    detection_thread = threading.Thread(target=start_detection, args=(camera,))
    detection_thread.start()
```

At the end of this stage, we used the following function to exit from the GUI application:

```
def exit():
    # Exit the GUI application
    os._exit(0)
```

3.3 Adding the GUI to Coding

At the end of this project, we have to implement a graphical user interface (GUI) for help sign detection. We have to create a Tkinter window to provide a user-friendly interface:

```
helpDetect = tk.Tk()
helpDetect.title('Help Signal Detection GUI')
helpDetect.configure(bg='#19115D')
helpDetect.geometry('600x760')
helpDetect.resizable( width: False,  height: False)
```

This window is customised by setting a size, background colour and defining with a title "Help Signal Detection GUI". We also had set the Tkinter window to be non-resizable.

Next, we also loaded a background image for the window to make it look more attractive and eye catchy.

```
background_main = ImageTk.PhotoImage(Image.open("C:\\UMPSA_Y2S2\\BSD2513 AI\\AI Project\\HELP SIGN GUI.png").resize((600, 760)))
help_label_main = tk.Label(helpDetect, image=background_main)
```

Additionally, we created two buttons in the GUI window and each button stands for different types of detections that can be carry out in UMPSA which are "CHEMICAL LABORATORY DETECTION" and "STUDENT PANTRY DETECTION". These buttons are customized by background colors, foreground colours, width and height.

```
btnDetect1 = tk.Button(helpDetect, text="CHEMICAL LABORATORY DETECTION", width=64, height=2, fg='white', bg='#C768D1',
                       command=lambda: start_detection_thread(1))
btnDetect1.place(x=75, y=345)
btnDetect2 = tk.Button(helpDetect, text="STUDENT PANTRY DETECTION", width=64, height=2, fg='white', bg='#C768D1',
                       command=lambda: start_detection_thread(2))
btnDetect2.place(x=75, y=405)
```

To facilitate user interaction, a QUIT button is included to allow users to exit the GUI window.

```
btnQuit = tk.Button(helpDetect, text="QUIT", width=20, height=2, fg='white', bg='#C768D1', command=exit)
btnQuit.place(x=225, y=670)
help_label_main.pack(side=tk.TOP)
```

Lastly, the following coding is used to run the Tkinter window and respond to any user actions like clicking buttons and updates the GUI accordingly:

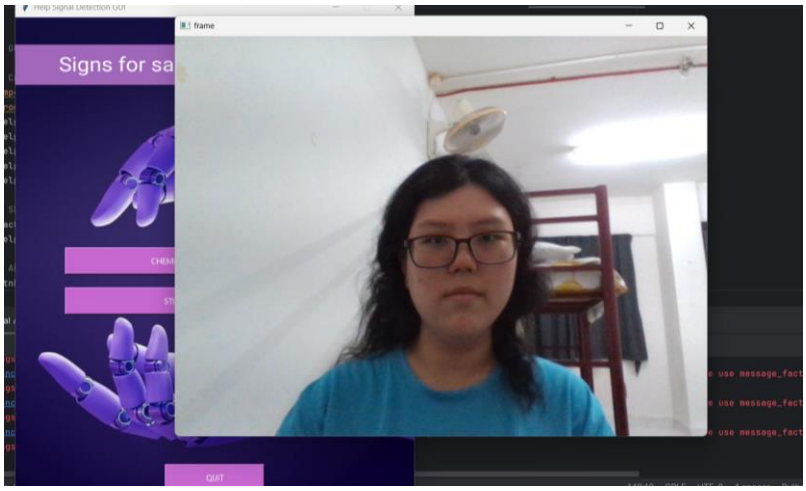
```
helpDetect.mainloop()
```

4.0 Results and Discussion

In this section, we present our GUI dashboard. The header title of our GUI dashboard is “Signs for safety, signals for care”, where it represents our mission to ensure a secure environment.

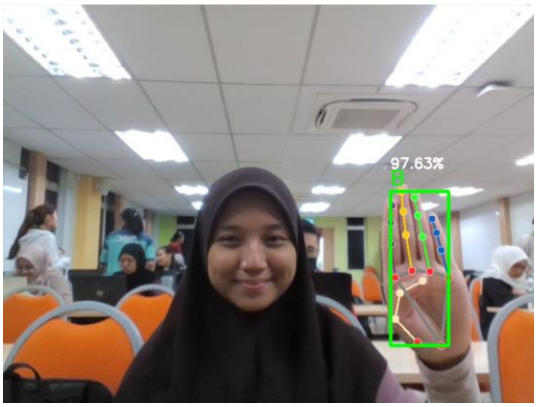
Our main menu features three interactive buttons which are the "Chemical Laboratory Detection" button, "Student Pantry Detection" button, and "Quit" button. By clicking on either of the detection buttons, users can access real-time camera feeds from the respective locations, allowing them to monitor any emergency case in real-time. Lastly, user can also click the "QUIT" button to exit the application at any time.

Since our help sign detection is only for real-time video, therefore when user click on any of the detection buttons, it will pop up a frame to capture live video from a camera and provides real time output.



User can place their hand in front of the camera, and the help sign detection system will start to scan the environment and identify specific help signs.

The following image shows the hand gesture for sign B which is palm to camera and tuck thumb:

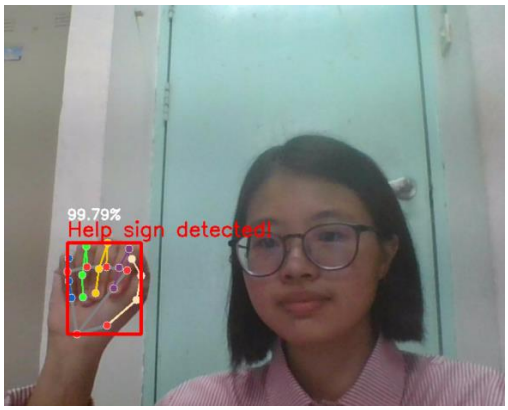


As we can see here, the detection system accurately identify sign B with the percentage of 97.63%.

Besides that, the following image shows the hand gesture for sign A where the thumb is trapped:

As shown, the detection system accurately identify sign A with the percentage of 99.76% which is almost 100%.

In this case, the help sign only can be detected when an individual performs hand gesture for sign B follow by the hand gesture for sign A. The results are shown as following:



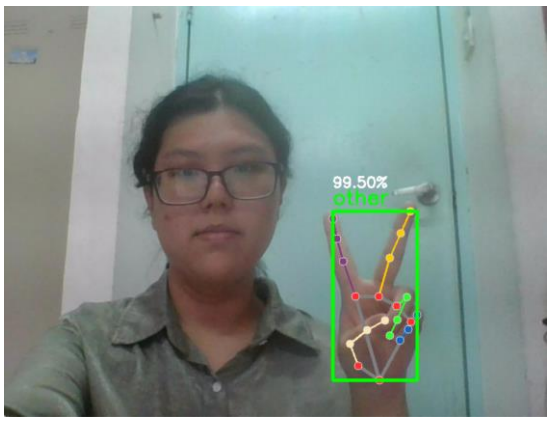
The image shows that the percentage of help sign achieved 99.79%.

The image shows that the percentage of help sign achieved 99.56%.

In practice, if a help sign is detected, an alarm will sound to notify the chemical laboratory assistant or any person in charge, ensuring immediate attention to any emergency.

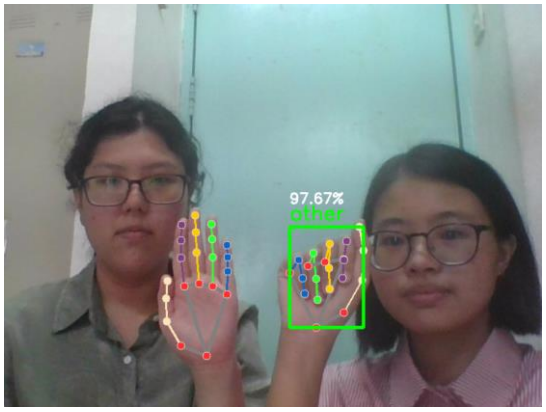
The system will categories any hand gestures that are not sign A or sign B as “other” category. For this “other” sign, it will not trigger any help sign detection.

The image shows that the percentage of other sign achieved 95.81%.



The image shows that the percentage of other sign achieved 99.50%.

The system can detect a maximum of two hand gestures. However, it can only draw the rectangle box on one hand only.



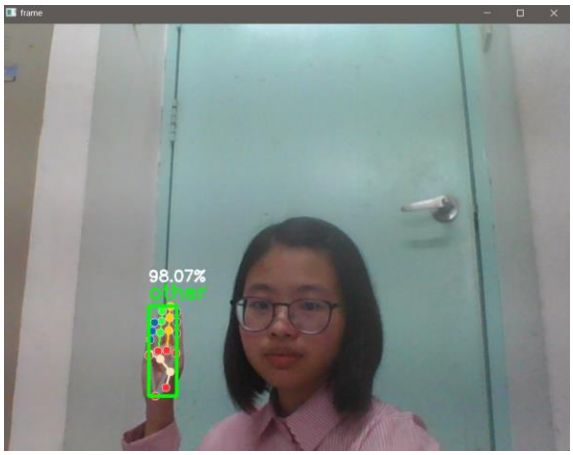
The image shows that the percentage of other sign achieved 97.67%.

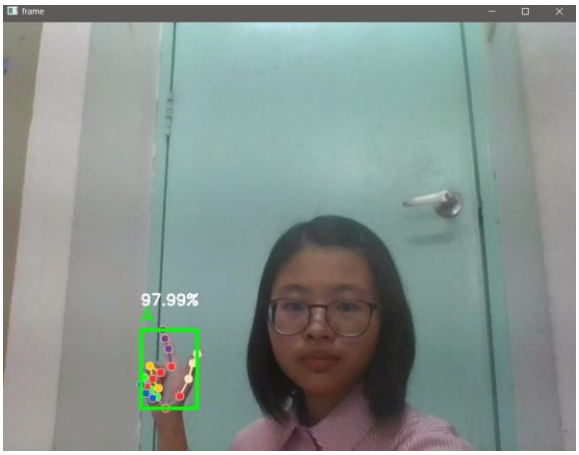
5.0 Limitation

Despite the effectiveness of our AI “Help Sign Detection” there are certain limitations that we addressed in our projects:

1) Limited Detection Accuracy caused by Environmental Factor

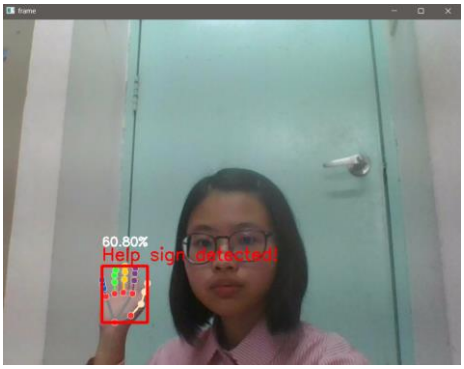
   Our system may struggle to accurately identify “Help Sign” hand gesture in certain environmental conditions, such as low-light environment, complex background, and obstructed views. The system faces dif ficulty to detect relevant hand gestures from mentioned environment where leading to misclassification.





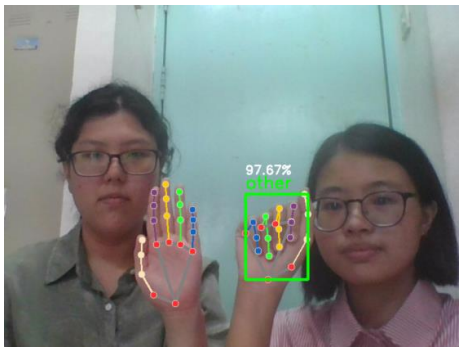
2) Limited Detection Accuracy caused by Distance Factor

   Our system may struggle to accurately detect the “Help Sign” hand gestures at varying distances between the camera and the subject. When we stand little further away from the camera and display ‘others’ sign hand gestures, the system will recognize it as help sign and trigger the alarm.



3) Limited Number of Hand Gesture Detection and Limited Rectangle Box Drawing

   Our system face limitation when it only can detect maximum up to 2 hand gestures in a single frame, but only can draw 1 rectangle box for detection on one hand at a time. In this case, the system may miss or misinterpret hand gestures in scenario involving multiple hand or complex hand movements, leading to inaccurate detection of help signs since the rectangle box for detection only captured one hand at a time.



4) Continuous Alarm Activation During Emergency Detection.

   Our system is facing limitation where the alarm will continue to ring constantly when the hand detection system is triggered during emergencies. It can lead to confusion, alarm fatigue or user annoyance. The system cannot recognize and deactivate the alarm even after the emergency has been addressed.

In summary, our “Help Sign Detection” systems show promising capabilities in identifying help signs, but we still face limitations in a few areas. Addressing these issues is crucial for enhancing the system's effectiveness and reliability in future use.

6.0 Conclusion

In conclusion, we successfully developed a hand gestures recognition system with Python coding to detect the Universal “Help Sign” hand gestures in real-time video analysis. We collected 100 captured images for each of the hand gestures and created a database for our training model to ensure the accuracy and consistency of the recognition system to the “Help Sign”. When the user puts their hand into the camera's line of sight, the hand gestures recognition system enables to detect hand gestures is one of the “Help Sign” patterns or not, and also if both of the “Help Sign” hand gestures detected in the short time, the system will display that is the Universal “Help Sign” hand gestures and play an alarm sound to alert the user.

Then, we also designed a GUI dashboard for this system to ensure user friendly of the application. The user can easily open the recognition system by just clicking the buttons and put their hand in real-time capture and the user can also exit the application easily by clicking the “QUIT” button. This hand gestures recognition system may have some improvement need to do in the future time, but we hope it will help the companies which have security requirement to the hand gestures recognition system for timely rescue when the CCTV detect the Universal “Help Sign” in the real-time streaming.

7.0 Reference

Canadian Women Foundation. (2024, June 6). *Signal for help: Use Signal to ask for help*. Become a Signal For Help Responder. https://canadianwomen.org/signal-for-help/

Womens's Funding Network. (2021, December 13). *Signal for help*. Signal for Help. https://www.womensfundingnetwork.org/signalforhelp/

The World Bank. (2022, July 18). *Signal for help / violence at home signal for help*. Health and Safety Directorate. https://www.worldbank.org/en/work-with-us/hsd/home/signal_for_help#:~:text=The%20signal%20is%20performed%20by,could%20be%20made%20easily%20visible.

8.0 Appendix

Dataset for hand sign images collection:

https://drive.google.com/drive/folders/1sjETGXpcE9_77Mf17P68QICOyNHNSsai?usp=sharing

Link for Help Sign Detection Coding:

https://docs.google.com/document/d/1NtTMZpp0YkA6QGRKSPIVpkvgXD-lLsep9KxI1-JD9dY/edit?usp=sharing

BSD2513 ARTIFICIAL INTELLIGENCE - GROUP PROJECT MARKING SCHEME

Rubric for CO2

CO2: Demonstrate critical thinking ideas of artificial intelligence knowledge in problem-solving situation.								
CRITERIA	**LEVEL OF ACHIEVEMENT**						**WEIGHTAGE**	**SCORE A**
	0 Grossly Inadequate	**1 Inadequate**	**2 Emerging**	**3 Developing**	**4 Good**	**5 Excellent**		
Description and explanation of the selected project and problem to be solved.	No description or explanation about the project selected and the problem to be solved in the report.	Barely describe and explain the project selected and the problem to be solved in the report.	Partly describe and explain the project selected and the problem to be solved in the report.	Clearly describe and explain the project selected and the problem to be solved in the report.	Very clearly describe and explain the project selected and the problem to be solved in the report.	Excellently described and explained the project selected and the problem to be solved in the report.	0.5	
Explanation regarding summary of the project context and objectives.	Failed to explain the summary of the project context and objectives.	Barely explain the summary of the project context and objectives.	Partly explain the summary of the project context and objectives.	Clearly to explain the summary of the project context and objectives.	Very clearly to explain the summary of the project context and objectives.	Excellently explain the summary of the project context and objectives.	1	
Explanation of the project methodology.	Failed to explain the project methodology.	Barely explained the project methodology.	Partly explained the project methodology.	Clearly explained the project methodology.	Very clearly explained the project methodology.	Excellently explained the project methodology.	1	
Analysis explanations and interpretations of the results and findings	Analysis is insufficiently explained. The interpretation of the results is not clearly discussed.	The analysis is haphazardly explained. The interpretation of the results is not clearly discussed.	The analysis is lack sufficient explanation. The interpretation of the results is sufficiently discussed.	The analysis is sufficiently and logically explained. The interpretation of the results is sufficiently discussed.	The analysis is clearly and logically explained. The interpretation of the results is clearly discussed.	The analysis is excellently and logically explained. The interpretation of the results is well discussed.	1	
Concluding remarks.	No concluding remarks were provided.	Barely concluding remarks provided and need to be more accurate.	Concluding remarks were provided but unclear and inaccurate.	Concluding remarks were provided but partly inaccurate.	Clear and good concluding remarks were provided.	Very clear and excellent concluding remarks were provided.	0.5	
						TOTAL		**20**

Rubric for CO3

CO3: Develop an artificial intelligence system prototype using appropriate software.								
CRITERIA	**LEVEL OF ACHIEVEMENT**						**WEIGHTAGE**	**SCORE**
	0 Grossly Inadequate	**1 Inadequate**	**2 Emerging**	**3 Developing**	**4 Good**	**5 Excellent**		
Ability to extract the datasets from sources very well.	Unable to extract the datasets.	Barely able to extract the datasets from the sources.	Partly able to extract the datasets from the sources.	Able to extract the datasets from the sources in successful results.	Very good to extract the datasets from the sources.	Excellently extract the datasets from the sources excellently.	0.5	
Ability to plan and construct the project design.	Unable to plan and construct the project design.	Barely able to plan and construct the project design.	Partly to plan and construct the project design.	Able to plan and construct the project design.	Very good to plan and construct the project design.	Excellently plan and construct the project design.	1	
Ability to construct and simulate artificial intelligence techniques.	Unable to construct and simulate artificial intelligence techniques.	Barely able to construct and simulate artificial intelligence techniques.	Partly able to construct and simulate artificial intelligence techniques.	Able to construct and simulate artificial intelligence techniques with successful results.	Very good to construct and simulate artificial intelligence techniques.	Excellently construct and simulate artificial intelligence techniques.	2	
Ability to modify or extended the programming codes extracted.	Unable to modify or extended the code extracted.	Barely able to modify or extended the code extracted.	Partly able to modify or extended the code extracted.	Able to modify or extended the code extracted.	Very good to modify or extend the code extracted.	Excellently modify or extend the code extracted.	2	
Ability to visualise (table graph, GUI, web-apps and etc) the programming codes.	Unable to visualise (table graph, GUI, web-apps and etc) the programming codes.	Barely able to visualise (table graph, GUI, web-apps and etc) the programming codes.	Partly to visualise (table graph, GUI, web-apps and etc) the programming codes.	Ability to visualise (table graph, GUI, web-apps and etc) the programming codes in successful results.	Very good to visualise (table graph, GUI, web-apps and etc) the programming codes.	Excellently visualise (table graph, GUI, web-apps and etc) the programming codes.	1	
The code can be executed and easy to understand the codes constructed.	No code is constructed.	Only few codes can be executed and difficult to follow the structure and flow of the codes.	Some of the codes can be executed, and fairly difficult to follow the structure and flow of the codes.	The code can be executed, and fairly easy to follow the structure and flow of the codes.	The code can be executed easily to follow the structure and flow of the codes.	The code can be executed and well easy to follow the structure and flow of the codes.	0.5	
Level of programming difficulty and effort.	Codes developed/ modified is simple, no ambitious and effort.	Codes developed/ modified is not challenging, less ambitious and takes few efforts.	Codes developed/ modified is not challenging, less ambitious and takes few efforts.	Codes developed/ modified are less challenging, moderately ambitious and take some effort.	Codes developed/ modified are quite challenging, moderately ambitious and take effort.	Codes developed/ modified are very challenging, and ambitious and takes extra effort.	1	
						TOTAL		**40**

Rubric for CO4

CO4: Demonstrate verbal and written communication skills.								
CRITERIA	**LEVEL OF ACHIEVEMENT**						**WEIGHTAGE**	**SCORE**
	0 Grossly Inadequate	**1 Inadequate**	**2 Emerging**	**3 Developing**	**4 Good**	**5 Excellent**		
Verbal: Able to communicate in team and with lecturer.	Do not commit to any discussion session presentation.	Poorly able to communicate/ discuss the artificial intelligence knowledge in presentation.	Fairly able to communicate/ discuss regarding artificial intelligence knowledge in presentation.	Able to communicate/ discuss regarding artificial intelligence knowledge in presentation.	Able to communicate/ discuss in a good way regarding artificial intelligence knowledge in presentation.	Able to communicate/ discuss excellently regarding artificial intelligence knowledge in presentation.	1	
Verbal: Able to provide general and background information which corresponds with the purpose. Show understanding of the topic.	Unable to provide general and background information which correspond with the purpose. Do not show understanding of the topic.	General and background information are inadequate and do not correspond with the purpose. Show poor understanding of the topic.	General and background information are slightly inadequate and fairly correspond with the purpose. Show fair understanding of the topic.	General and background information are adequate but moderately correspond with the topic. Show average understanding of the topic.	General and background information are adequate and correspond with the topic. Show a good understanding of the topic.	General and background information are adequate and correspond with the topic. Show an excellent understanding of the topic.	1	
Written: Able to demonstrate report in a clear manner, coherent and systematic.	Do not submit group project report.	Poorly able to demonstrate report in clear manner, coherent and ssystematic.	Fairly able to demonstrate report in a clear manner, coherent and systematic.	Able to demonstrate report in a clear manner, coherent and systematic.	Demonstrate a good report in a clear manner, coherent and systematic.	Demonstrate an excellent report in a clear manner, coherent and systematic.	1	
Written: Able to provide adequate, relevant and significant information on the chapters. Provide links between ideas.	Unable to provide adequate, relevant and significant information of the chapters. Do not provide links between ideas	Information is inadequate; The relevance and significance of information is poorly made; Links and connections between ideas are poorly made.	Information is present, but supporting details are inadequate; The relevance and significance of information is fairly made; Links and connections between ideas are fairly made.	Information is adequate; The relevance and significance of information is moderately made; Links and connections between ideas are moderately made.	Information is adequate; The relevance and significance of information is good; Links and connections between ideas are good.	Information is adequate; The relevance and significance of information is excellent; Links and connections between ideas are excellent.	1	
						TOTAL		**20**

Rubric for CO5

CO5: Integrate artificial intelligence knowledges to the project and future problems.								
CRITERIA	**LEVEL OF ACHIEVEMENT**						**WEIGHTAGE**	**SCORE**
	0 Grossly Inadequate	**1 Inadequate**	**2 Emerging**	**3 Developing**	**4 Good**	**5 Excellent**		
Knowledge application or transfer.	No reference from previous learning and do not apply the knowledge in methodology, result and discussion.	Makes vague reference to previous learning but does not apply the artificial intelligence knowledge to performance in methodology, result and discussion.	Makes defined reference to previously gained knowledge but does not apply to methodology, result and discussion.	Makes defined reference to gathered previous knowledge, and demonstrates limited capacity to utilise in methodology, result and discussion.	Demonstrates reference to previously gained knowledge and demonstrates strong application to methodology, result and discussion.	Makes explicit reference to previous knowledge and applies these skills to methodology, result and discussion. Creative methods/ manners.	1	
Able to pursue artificial intelligence knowledge beyond expectation in completing the project.	Unable to pursue artificial intelligence knowledge beyond expectation in completing the project.	Rarely pursue artificial intelligence knowledge beyond expectation in completing the project.	Fairly pursue artificial intelligence knowledge beyond expectation in completing the project.	Sometimes pursue artificial intelligence knowledge beyond expectation in completing the project.	Often pursue artificial intelligence knowledge beyond expectation in completing the project.	Always pursue artificial intelligence knowledge beyond expectation in completing the project.	1	
Look for relevant information independently.	Too depend on others for relevant and reliable information in the project.	Always needs help from others to look for relevant and reliable information in doing the project.	Continuously needs help from others to look for relevant and reliable information in doing the project.	Often needs help from others to look for relevant and reliable information in doing the project.	Sometimes needs help from others to look for relevant and reliable information in doing the project.	Rarely needs help from others to look for relevant and reliable information in doing the project.	1	
Reflection from prior learning towards life experience.	Do not review any prior learning at any level.	Reviews prior learning at a surface level without clarifying meaning or indicating a broader perspective about educational or life events.	Reviews prior learning with limited capacity, giving minor clarification or broad perspective.	Reviews prior learning and shows clarification or broad perspective about educational or life events.	Reviews prior learning in some depth, revealing clear meaning and indicating broad perspectives related to educational events.	Review prior learning in depth to reveal significantly changed perspectives about education and life experiences.	1	
						TOTAL		**20**