Q1.

The data collection method is an extension from 8.2C API wrapper. I have added some instance variables and functions to collect pictures alongside with Accelerometer axes. One thing to note is that instead of saving the data (picture and axes) at 10 seconds interval, I use 1 second interval because I can get more meaningful information about the pattern of distinct activities. Also, I used my phone camera instead of laptop's because my laptop's camera is broken. I won't go into details since the class has already been carefully explained in the 8.2C task, but here is what I have added in the API class.

```
# Initialize camera capture of camera_url is provided
        if self.camera_url:
            self.init_camera()
```

```
def init_camera(self) -> None:
        """Initialize the camera capture from the IP camera URL"""
        self.camera_cap = cv2.VideoCapture(self.camera_url)
        if not self.camera_cap.isOpened():
            print(f"Failed to open camera at {self.camera_url}")
```

```
def capture_image(self, sequence_number: int, timestamp: str) -> None:
        """Capture image from the camera, save with the sequence number and
timestamp"""
        try:
            if self.camera_cap and self.camera_cap.isOpened():
                ret, frame = self.camera_cap.read()
                if ret:
                    # Rotate and resize the image
                    rotated_frame = cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE)
                    resized_frame = cv2.resize(rotated_frame, (720, 1280))

                    # Ensure the image save directory exists
                    if not os.path.exists(self.image_save_dir):
                        os.makedirs(self.image_save_dir)

                    # Construct the file path with a proper file name and
extension (.jpg)
                    image_filename = os.path.join(
                        self.image_save_dir,
f"{sequence_number}_{timestamp.replace(':', '')}.jpg"
                    )

                    # Save the image and check if the operation is successful
                    if cv2.imwrite(image_filename, resized_frame):
```

```
                        print(f"Image saved successfully: {image_filename}")
                        self.latest_image_path = image_filename  # Update the
latest image path
                    else:
                        print(f"Failed to save image: {image_filename}")
                else:
                    print("Failed to capture image from the camera.")
            else:
                print("Camera is not initialized or not opened.")
        except Exception as e:
            print(f"Exception during image capture: {e}")
```

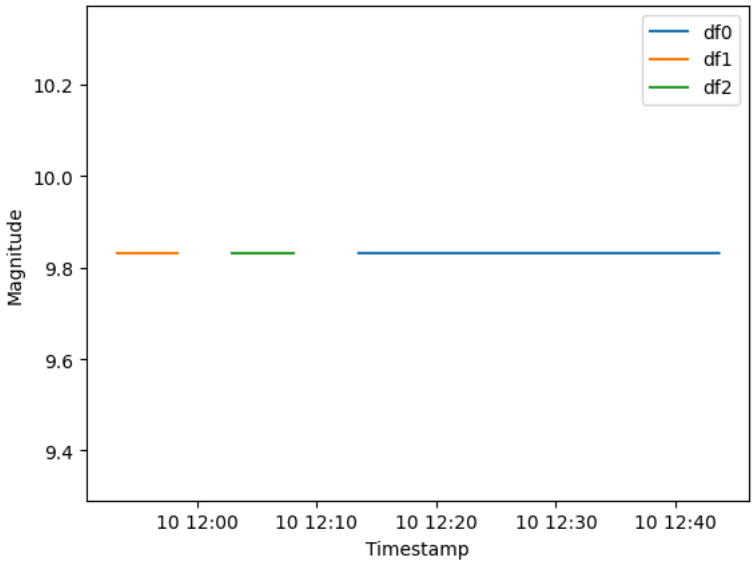
```
def create_dash_app(self) -> Dash:
        """Create the Dash app for live monitoring"""
        app = Dash(__name__)

        app.layout = html.Div([
            dcc.Graph(id='live-graph'),
            dcc.Interval(id='graph-update', interval=self.update_interval),
            html.Img(id='live-image', src='', style={'width': '25%', 'display':
'block', 'margin': 'auto'}),
        ])
# Update the image displayed on the dashboard
        @app.callback(
            Output('live-image', 'src'),
            Input('graph-update', 'n_intervals')
        )
        def update_image(n):
            if self.latest_image_path:
                encoded_image =
self.encode_image_to_base64(self.latest_image_path)
                return f"{encoded_image}"
            return ''

        return app
```

That is pretty much it, the comments have explained on the functionality of each part. I just added some code to save the picture and display the newest one on the dashboard. The visualization will be in the video.

Q2. The findings

I have done 2 things to try to find pattern in my distinct activities. I need to explain the distinct activities first. The first one ‘0’ is just me working, not showing any hand gestures; the second ‘1’ is when I raise a thump up; the third '2’ is when I raise a peace sign. Each picture below corresponds to 0,1,2 label, and each picture is taken at different angles.

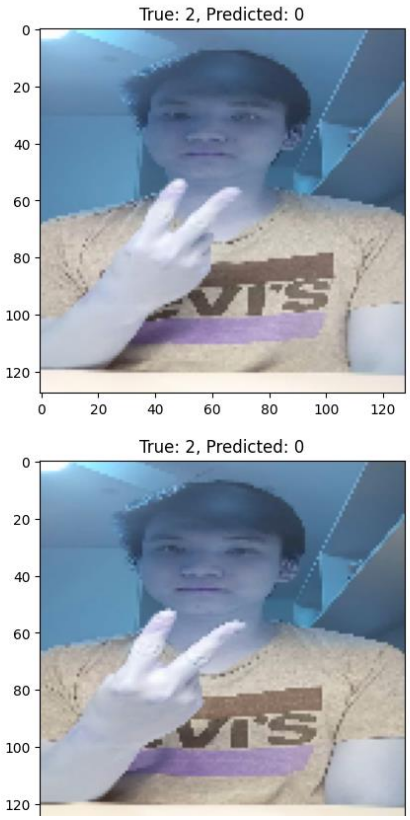
First is through the representation of linear Accelerometer Linear or the magnitude of the axes. This technique is similar to my 8.2C task when I combine the 3 axes into 1 line. Here is the time series plot.



The only pattern I can see is that different activity is recorded at different time (as the lines are disconnected). This makes me realize I need a more powerful technique to recognize the patterns of my activities, so I decided to use CNN (Convolutional Neural Network). I will explain how in section 3 but results the model gives are quite good. Using image matrices and its respective labels, I got 100% accuracy on the train set, and 91% on the test set.

```
1/5 ━━━━━━━━━━━━━━━━━━━━ 4s 1s/step - accuracy: 1.0000 - loss: 0.0282
I0000 00:00:1725962434.075708    9472 device_compiler.h:188] Compiled clus
once for the lifetime of the process.
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 133ms/step - accuracy: 0.9640 - loss: 0.0989
Test Accuracy: 0.9155844449996948
```

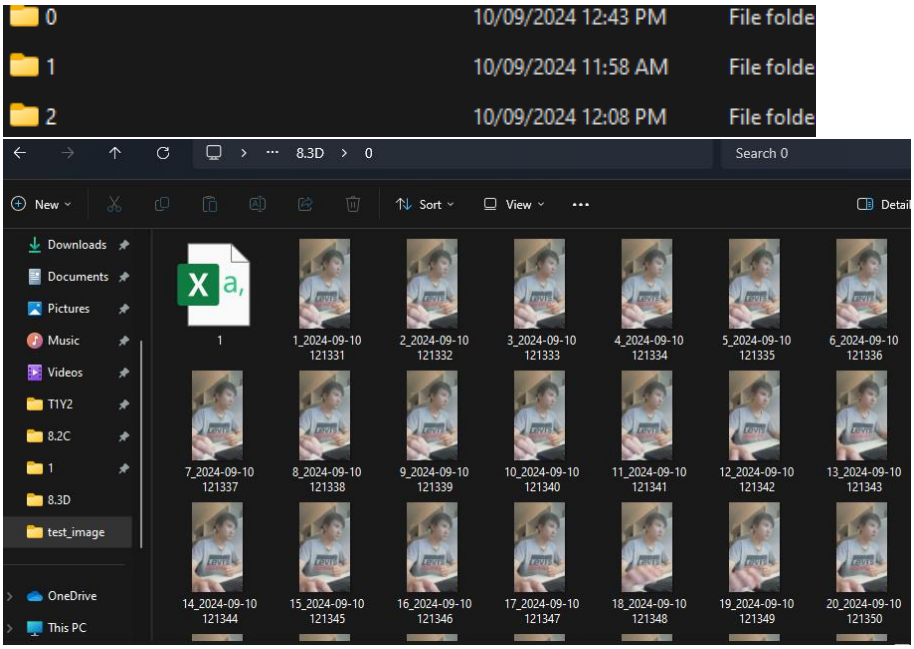
```
Number of misclassified images: 13
```



I have used similar camera angle on both the training and testing set because we only have limited amount of data.

Q3. Annotation file and how did I recognize patterns in images

I have recorded and saved the csv file for the Accelerometer axes and pictures for each label in folder like this.



I transformed those images into matrices, scaled them and added them to a dataframe using this code:

```
dir = r"/mnt/c/Users/tomde/OneDrive/Documents/Deakin/Deakin-Data-
Science/T1Y2/SIT225 - Data Capture Technologies/Week 8 - Using smartphone to
capture sensor data/8.3D/2"

# Define the image size for the model
image_size = (128,128)

# Function to extract year '2024' from picture names
def contain_2024(filename):
    return re.search(r'2024', filename) is not None

# List all files in the directory
files = os.listdir(dir)

# Filter for image files containing '2024' in the filename
image_filenames_2024 = [file for file in files if file.endswith(('.jpg', '.png'))
and contain_2024(file)]
image_filenames_2024

# List to store image matrix
```

```
image_data = []

# Loop through the filtered filenames and convert each image to a matrix
for filename in image_filenames_2024:
    img_path = os.path.join(dir, filename)

    # Read the image using OpenCV
    image = cv2.imread(img_path)

    # Resize the image to a uniform size (128x128)
    image_resized = cv2.resize(image, image_size)

    # Normalize the image (values between 0 and 1)
    image_resized = image_resized / 255.0

    # Convert the image to a CuPy array (move to GPU)
    image_matrix = np.array(image_resized)

    # Append the matrix along with the filename
    image_data.append({
        'Filename': filename,
        'ImageMatrix': image_matrix
    })

df_images = pd.DataFrame(image_data)

# Combine with df0
df2 = pd.read_csv('2/1.csv')
df2['ImageMatrix'] = df_images['ImageMatrix']
df2['Label'] = 2
df2.head()
```

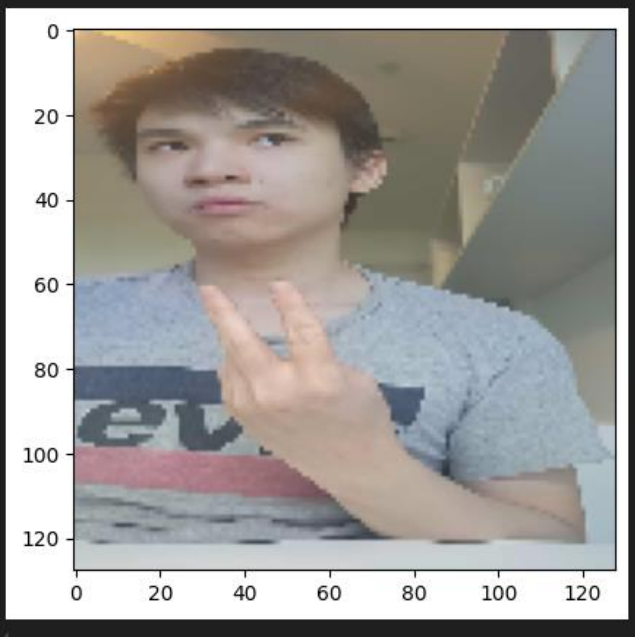
The output is

	Timestamp	Accelerometer_X	Accelerometer_Y	Accelerometer_Z	ImageMatrix	Label
0	2024-09-10 12:02:56	0.15705	0.105	9.82995	[[[0.8431372549019608, 0.9686274509803922, 1.0...	2
1	2024-09-10 12:02:57	0.15705	0.105	9.82995	[[[0.8627450980392157, 0.9725490196078431, 1.0...	2
2	2024-09-10 12:02:58	0.15705	0.105	9.82995	[[[0.8627450980392157, 0.9764705882352941, 1.0...	2
3	2024-09-10 12:02:59	0.15705	0.105	9.82995	[[[0.8627450980392157, 0.9764705882352941, 1.0...	2
4	2024-09-10 12:03:00	0.15705	0.105	9.82995	[[[0.8627450980392157, 0.9764705882352941, 1.0...	2

```
image_uint8 = (df2.ImageMatrix[0] * 255).astype(np.uint8)
plt.imshow(cv2.cvtColor(image_uint8, cv2.COLOR_BGR2RGB))
```

Python

```
<matplotlib.image.AxesImage at 0x7efb77f2d990>
```



The above example is just for Label 2, but I have done the same for the other two labels. Next, I combine the three dataframes into one. Then, I splitted the data into X and y, and in the correct format for CNN.

```
# Split data
X = cp.array([cp.asarray(image) for image in df['ImageMatrix'].values])
y = cp.array(df['Label'].values)
```

```
# # Oversample X and y
# from imblearn.over_sampling import RandomOverSampler
# ros = RandomOverSampler(sampling_strategy='auto', random_state=42)
# X_resampled, y_resampled = ros.fit_resample(X.get(), y.get())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_train = to_categorical(y_train.get(), num_classes=3)
y_test = to_categorical(y_test.get(), num_classes=3)
```

```
X_test.shape, y_train.shape
```

```
((469, 128, 128, 3), (1877, 3))
```

Then I train the CNN model.

```python
# Split data
X = cp.array([cp.asarray(image) for image in df['ImageMatrix'].values])
y = cp.array(df['Label'].values)
```

```python
# # Oversample X and y
# from imblearn.over_sampling import RandomOverSampler
# ros = RandomOverSampler(sampling_strategy='auto', random_state=42)
# X_resampled, y_resampled = ros.fit_resample(X.get(), y.get())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
y_train = to_categorical(y_train.get(), num_classes=3)
y_test = to_categorical(y_test.get(), num_classes=3)
```

+ Code + Markdown

```python
X_test.shape, y_train.shape
```

```
((469, 128, 128, 3), (1877, 3))
```

```python
# Value counts
pd.Series(y_test.argmax(axis=1)).value_counts()
```

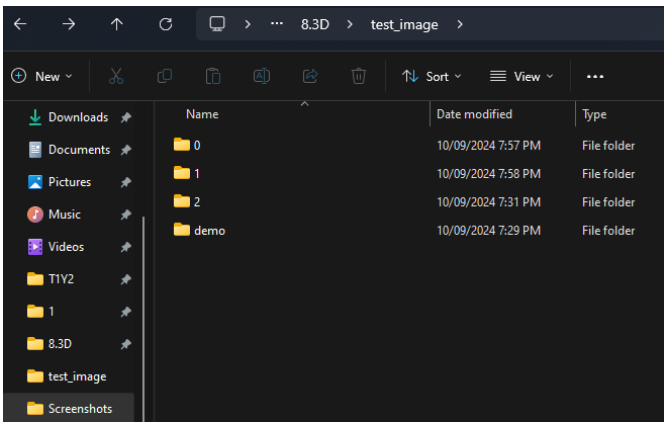
```
0    357
1     59
2     53
Name: count, dtype: int64
```

```python
# Train the model
model.fit(X_train.get(), y_train, epochs=10, batch_size=32, validation_data=(X_test.get(), y_test))
```

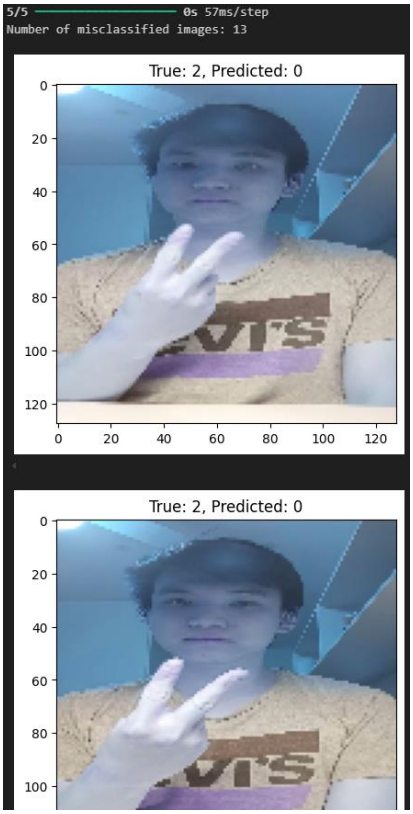
Python

```
Epoch 1/10
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1725958983.533093    1497 service.cc:148] XLA service 0x7ef95c00e2c0 initialized for platform CUD
I0000 00:00:1725958983.534014    1497 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 3090, C
2024-09-10 19:03:03.585781: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR
I0000 00:00:1725958983.718621    1497 cuda_dnn.cc:530] Loaded cuDNN version 90400
I0000 00:00:1725958985.358375    1497 device_compiler.h:188] Compiled cluster using XLA!  This line is logged
59/59 ━━━━━━━━━━━━━━━━━━━━ 6s 47ms/step - accuracy: 0.9183 - loss: 0.5491 - val_accuracy: 1.0000 - val_loss:
Epoch 2/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 1.0000 - loss: 2.3296e-07 - val_accuracy: 1.0000 - val_los
Epoch 3/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 1.0000 - loss: 1.8333e-07 - val_accuracy: 1.0000 - val_los
Epoch 4/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 1.0000 - loss: 1.5334e-07 - val_accuracy: 1.0000 - val_los
Epoch 5/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 1.0000 - loss: 1.2769e-07 - val_accuracy: 1.0000 - val_los
Epoch 6/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 1.0000 - loss: 1.5095e-07 - val_accuracy: 1.0000 - val_los
Epoch 7/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 1.0000 - loss: 1.0820e-07 - val_accuracy: 1.0000 - val_los
Epoch 8/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 1.0000 - loss: 1.0895e-07 - val_accuracy: 1.0000 - val_los
Epoch 9/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 1.0000 - loss: 9.0233e-08 - val_accuracy: 1.0000 - val_los
Epoch 10/10
59/59 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 1.0000 - loss: 6.9258e-08 - val_accuracy: 1.0000 - val_los

<keras.src.callbacks.history.History at 0x7ef9e8383af0>
```

Then I created a new folder containing the test set. The test set photos are taken in roughly the same angle as the train ones.



There are a total of 154 test samples, and I scaled those down and fit into the model.



The model got 13 wrong out of 154. From this, I conclude that for complex pattern recognition like this (those the deals with pictures), we need cutting-edge machine learning models. Although, I do not know which neuron of the model captures which pixel, or how can it recognize patterns in complex image matrices since deep learning is a black box, I do know that this proves complex pattern analysis requires machine learning.

Q4.

https://www.youtube.com/watch?v=nITgq3rVxk8

Q5.

https://github.com/tomadonna1/SIT225_2024T2/tree/main/Annotate%20smartphone%20accelerometer%20data%20by%20capturing%20activity%20images