CS3237 Project
Final Report

Group 22
Lian Jiade (A0187884E)
Mao Yiru (A0244406L)
Qing Bowen (A0243489R)
Tsai Ming Chin (A0190770A)

Final Project Report

1. Introduction

1.1 Motivation

What is a similar problem youths from China and Singapore face these days?
No, it's not tiger mums or school pressure. It is obesity.

According to China's National Health Commission data, more than half of China's schoolchildren are short-sighted and nearly one in five between the ages of six and 17 is overweight or obese[i]. In Singapore, obesity among children and adolescents has increased tenfold over the past 40 years. In 2016, the obesity rate among school-going children in Singapore stood at 12% and is still continuing to rise[ii].

With rising obesity rates in China and Singapore, the governments from both countries have been pushing for youths to exercise more and develop a habit for outdoor activities to lead a more balanced lifestyle. President Xi, for example, has said he wants the next generation of youths to spend more time outdoors to "civilise the spirit and toughen the body" and has enacted policy changes such as giving fitness exams significantly more weightage in the "zhongkao (中考)" exam taken by Chinese students.

On the other hand, in Singapore, the IPPT fitness test, which comprises 3 stations: Push-Ups, Sit-Ups, and 2.4km Run is conducted annually for secondary and JC students. Males continue to undergo this fitness test annually even after their schooling years as mandated by law to maintain their physical fitness level and have to attend remedial training conducted by the military if they fail to meet the minimum passing score.

Furthemore, with the COVID-19 pandemic continuing, indoor gyms and fitness centers tend to be risky for COVID-19 transmission because of physical and surface contact. Hence, people have chosen to exercise alone at home. However, there are some drawbacks to this. For example, whether their exercise postures are correct or not seems ambiguous for them without any professional or peer guidance. A nonstandard posture not only means improper training for their fitness exam, it may also put lots of stress on some body parts, which can lead to chronic injuries.

Hence, considering people's various needs for exercising such as to stay healthy, reduce obesity and prepare for their fitness exam as illustrated above, we would like to propose a product called FitTrack that can conveniently record users' physical activity data for health tracking purposes and also to help specific users such as students prepare for their annual fitness exams.

2.1 Problem Statement

- Lack of a specific fitness product for students in Singapore and China to train and prepare for their annual fitness exam resulting in students not performing to their best ability
- There are few fitness products that can track the fitness activity of the user in an automated way. Hence, users usually resort to manually tracking their exercise repetitions which is not only distracting while exercising. They may also lose count of the actual exercise reps they perform.
- People who exercise alone lack guidance about whether the posture is standard or not. Training with an improper exercise posture can lead to deduction in marks during their actual fitness exam. In addition, a non-standard posture can also lead to chronic injuries. Hence, there is a need for a fitness product that can detect and alert users about their current exercise posture

2.2 Solution Description

To address the above problems, our group has decided to propose a comprehensive IOT fitness product called FitTrack. This product has 3 main features:
Activity Recognition, Posture Detection and Heat Injury Warning.

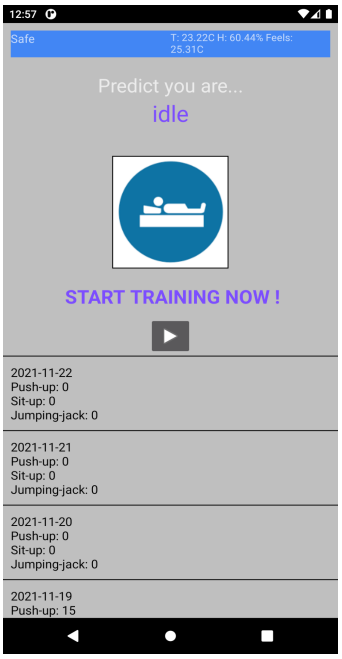


Fig 1: FitTrack's Home User Interface

Fig 2: FitTrack's Pose Detection Interface

Fig 3: Sensor Setup for the user

2.3 Walk Through of Key Features

1) Users will wear two TI SensorTags on two specific positions (one on the wrist and another on the waist). These sensor tags collect motion, humidity and temperature data (Fig 3).
2) Sensors will sent the data to a laptop for data preprocessing and machine learning models will recognise their current activity in real time, count their exercise repetition and classify heat injury risks level
3) Users can then simply refer to FitTrack's Android app to track their fitness activity in real time and the current heat injury risk level
4) All fitness activity data is also stored in the cloud and displayed on the app for users to monitor their long-term fitness activity
5) If users are interested in their exercise posture for push-up, they can enable FitTrack's Posture detection functionality which will capture a live video and display a counter for the number of incorrect push-ups they performed

Activity Recognition

This feature aims to address Problem Statement 1 and 2.
FitTrack will be able to recognise the current physical activity the user is engaged in using a machine learning model based on the sensor motion data and display it in real-time on an android app. FitTrack can recognise Six Activities: Push-up, Sit-up, Idle, Walking, Running and Jumping Jacks. If users are engaged in jumping-jacks, push-ups and sit-ups, Fitrack will also be able to count the repetitions done and display the records on the app in real time. In addition, all exercise data is sent to the cloud for storage so that users are able to keep track of their fitness activity levels on a long-term basis for training or health reasons.

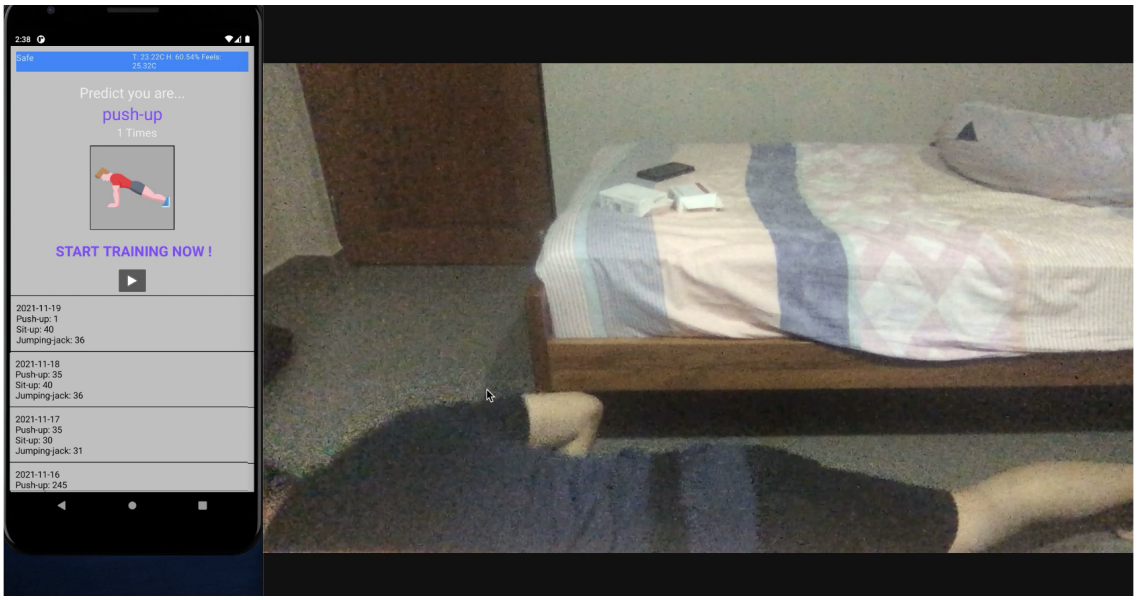


Fig 4: Example of a user engaged in Push-up activity using FitTrack

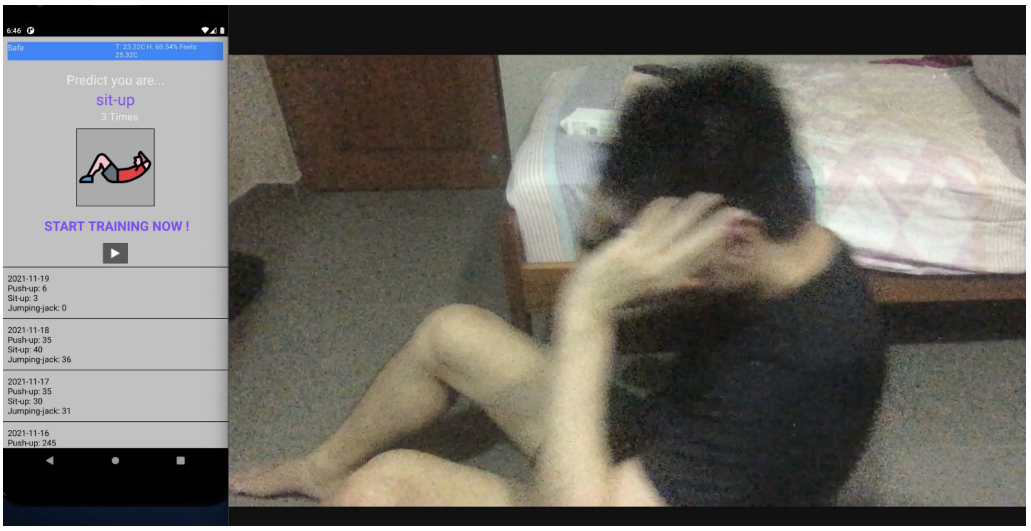


Fig 5: Example of a user engaged in Sit-up activity using FitTrack

Posture Detection

This feature aims to address Problem Statement 3.

FitTrack comes with an android app and users can utilise the android app to capture a live video of themselves engaged in a physical activity such as push-up.

The android app comes with built-in SDKs for keypoint detection of key human body parts such as shoulder, elbow, face and knees. The positions of these keypoints will then be used by a machine learning model within the app to determine if the user's posture is currently standard or not standard. If FitTrack determines an improper push-up being done, the counter on the screen will be incremented (Fig 7).

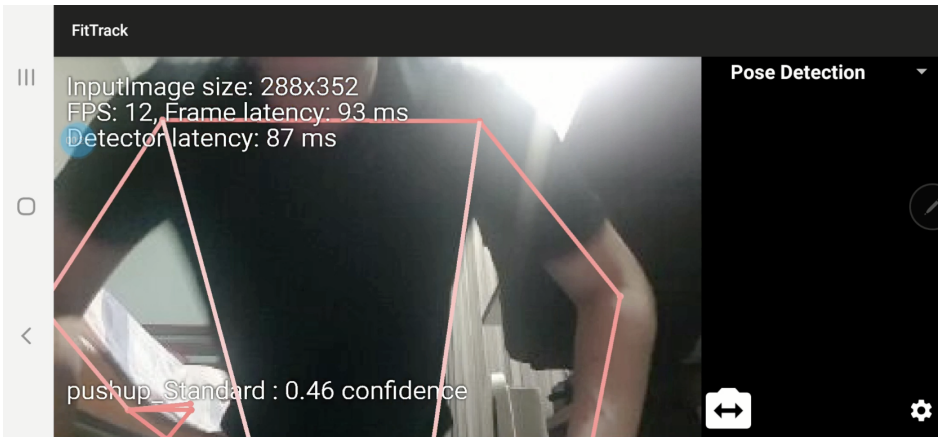


Fig 6: Example of user engaged in Push-up using FitTrack's pose Detection

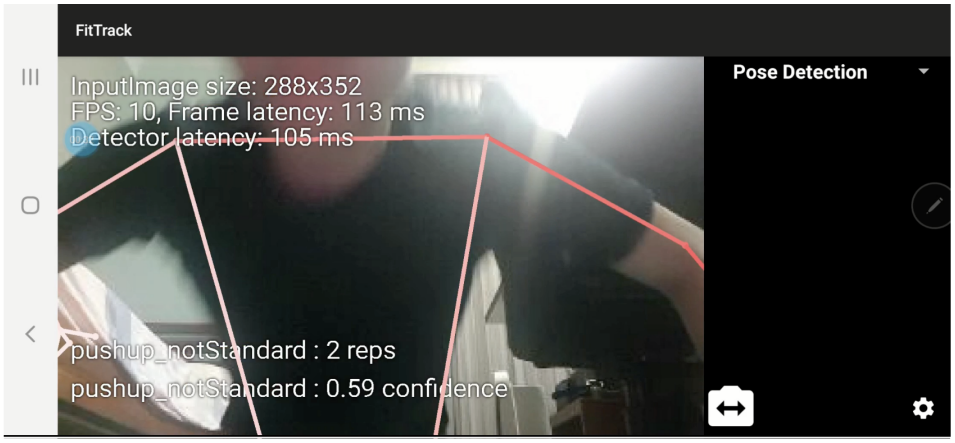


Fig 7: Example of user engaged in improper wide-arm Push-up

Heat Injury Warning

This feature is a complementary feature to address Problem Statement 1.
FitTrack will be able to predict the heat index, which is a human-perceived equivalent temperature known also as “Feels Like Temperature” based on temperature and humidity data and display different heat injury risk levels on the android app to ensure that as users train and exercise hard, they also train safely.

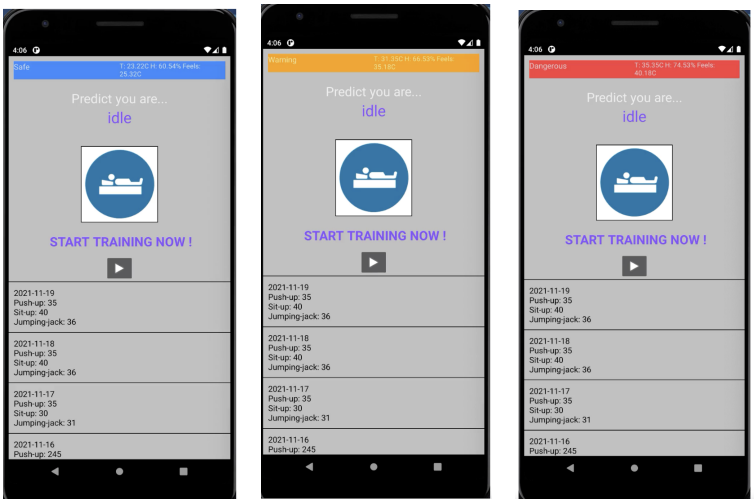


Fig 8: Change in text color as heat injury risk level increases from safe to dangerous

2. Approaches

3.1 IoT Architecture

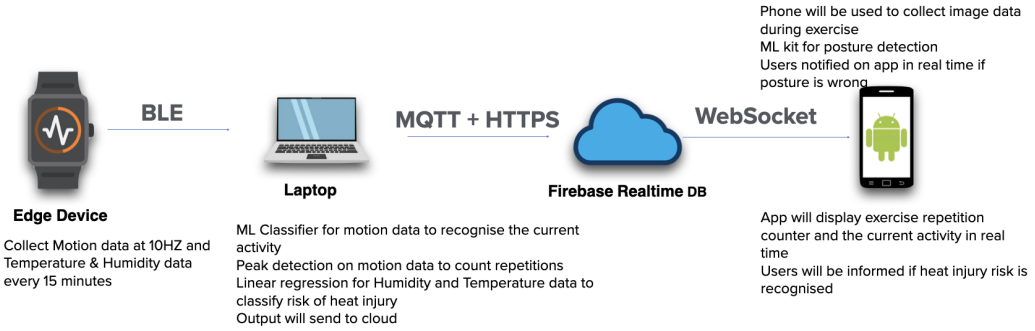


Fig 9: High Level System Design of FitTrack

The above diagram represents our overall architecture.
Key Hardware: 2 TI CC2650 Sensor Tags, Android Phone and laptop that supports BLE
Key Technologies: MQTT, Android Studio, Google ML Kit and Firebase real time

Sequence Flow for Sensor Data

1) Sensors collect the relevant motion (Gyrometer and Accelerometer Data), Temperature and Humidity data
2) Bluetooth low energy is used to transmit the sensor data to the laptop where there are 2 machine learning models ready to utilise the incoming data for prediction
3) The laptop receives the motion data and will perform data pre-processing. Then data will be sent via MQTT to a subscriber every 5 seconds with a machine learning model loaded to predict the current activity the user is engaged in (Fig 10).
4) In addition, a peak detection algorithm will be used to count the number of peaks for the current window of motion data to determine the number of exercise repetitions (Fig 10)
5) The laptop also receives the environmental data and will send the data via MQTT every 15 minutes to another subscriber with a machine learning model loaded to predict the current "Feels like Temperature" and the current environmental Temperature and Humidity (Fig 11)
6) Once the machine learning models predict an output, it will send the output and other relevant data such as exercise repetition via MQTT to another intermediate subscriber
7) The subscriber will listen for incoming data and upon receiving it, publish it to Firebase realtime database (Firebase RTDB) (Fig 12)
8) The android app which is connected to Firebase RTDB via websockets will be notified when new data is sent into the database and will render an updated UI to reflect the changes (Fig 13)

```python
def setup(hostname):
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(hostname)
    client.loop_start()
    return client
def main():
    #set up the MQTT publisher
    global loaded_rf
    global scaler
    loaded_rf = joblib.load("onevsonelinearsvc.joblib")
    scaler = joblib.load('scaler.joblib')
    setup("localhost")
    mqtt_pub.run()
    mqttPeak_pub.run()
    while True:
        pass

if __name__ == "__main__":
    main()
```

```python
def on_message(client, userdata, msg):
    global count
    global arr
    recv_dict = json.loads(msg.payload)
    sensor = recv_dict['sensor']
    if sensor == int(1):
        value1 = recv_dict['payload']
        arr = arr + value1
        count = count + sensor
        peak_data = recv_dict['peak']
        features = arr
        transformed_data = scaler.transform([features])
        value = loaded_rf.predict(transformed_data)
        activity = thisdict[str(value[0])]
        json_msg = {
            'type':'activity',
            'result': thisdict[str(value[0])]
        }
        mqtt_pub.publish(json.dumps(json_msg))

        if activity == 'push-up':
            threshold = 0.98 * max(peak_data)
            peaks, _ = find_peaks(peak_data, height=threshold)
            json_msg = {
            'type':'peak_pushup',
            'peak_count': len(peaks)
            }
            mqttPeak_pub.publish(json.dumps(json_msg))

        if activity == 'sit-up':
            threshold = 0.94 * max(peak_data)
            peaks, _ = find_peaks(peak_data, height=threshold)
            json_msg = {
            'type':'peak_situp',
            'peak_count': len(peaks)
            }
            mqttPeak_pub.publish(json.dumps(json_msg))

        if activity == 'jumping jack':
            threshold = 0.99 * max(peak_data)
            peaks, _ = find_peaks(peak_data, height=threshold)
            json_msg = {
            'type':'peak_jump',
            'peak_count': len(peaks)
            }
```

Fig 10: Sample code of Machine Learning for Activity Recognition and Repetition Counting

```python
mqtt_pub = Publisher("sensor/data", "localhost", 1883)
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Successfully connected to broker")
        client.subscribe("sensor2/data")
    else:
        print("Connection failed with code %d." %rc)


def on_message(client, userdata, msg):

    recv_dict = json.loads(msg.payload)
    temp = np.array(recv_dict['payload'][0]).reshape(-1, 1)
    RH = np.array(recv_dict['payload'][1]).reshape(-1, 1)

    data = np.concatenate((temp, RH), axis = 1)
    env_humi = loaded_RH.predict(data)
    env_temp = loaded_Temp.predict(data)
    tempF= env_temp[0][0]*1.8+32
    data2 = np.concatenate(
        (
        np.array(tempF).reshape(-1,1),
        np.array(env_humi[0][0]).reshape(-1,1),
        )
        ,axis = 1)

    feelikeF = loaded_HI.predict(data2)
    feellikeC = (feelikeF[0][0]-32)/1.8
    json_msg = {
            'type': 'heat_injury',
            'env_temp':env_temp[0][0],
            'env_humi':env_humi[0][0],
            "FL_temp": feellikeC
    }
    mqtt_pub.publish(json.dumps(json_msg))
```

```python
def setup(hostname):
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(hostname)
    client.loop_start()
    return client

def main():
    global loaded_RH
    global loaded_Temp
    global loaded_HI
    loaded_RH = joblib.load("./Predict_env_humi.pkl")
    loaded_Temp = joblib.load("./Predict_env_temp.pkl")
    loaded_HI = joblib.load('./Predict_Heat_Injury.pkl')
    setup("localhost")
    mqtt_pub.run()
    while True:
        pass
if __name__ == "__main__":
    main()
```

Fig 11: Sample code of Machine Learning for Heat Injury

```python
def on_message(client, userdata, msg):
    recv_dict = json.loads(msg.payload)
    if recv_dict['type'] == 'activity':
        value = recv_dict['result']
        ref = db.reference("date"+ "/" + today + "/" +'message')
        current_value = ref.get()
        if current_value != value:
            print("changed")
            ref.set(value)

    if recv_dict['type'] == 'peak_pushup':
        value = recv_dict['peak_count']
        ref = db.reference("date"+ "/" + today + "/" + 'pushup_count')
        current_value = ref.get()
        value = int(current_value) + int(value)
        print(value)
        ref.set(value)

    if recv_dict['type'] == 'peak_situp':
        value = recv_dict['peak_count']
        ref = db.reference("date"+ "/" + today + "/" + 'situp_count')
        current_value = ref.get()
        value = int(current_value) + int(value)
        print(value)
        ref.set(value)

    if recv_dict['type'] == 'peak_jump':
        value = recv_dict['peak_count']
        ref = db.reference("date" + "/" + today + "/" + 'jump_count')
        current_value = ref.get()
        value = int(current_value) + int(value)
        print(value)
        ref.set(value)

    if recv_dict['type'] == 'heat_injury':
        value1 = recv_dict['FL_temp']
        value1 = round(value1, 2)
        value2 = recv_dict['env_temp']
        value2 = round(value2, 2)
        value3 = recv_dict['env_humi']
        value3 = round(value3, 2)
        temp_text = "T: " + str(value2) + "C " + "H: " + str(value3)+ "% " + "Feels: " + str(value1) + "C"
        ref = db.reference("date" + "/" + today + "/" + 'temp')
        ref.set(temp_text)
```

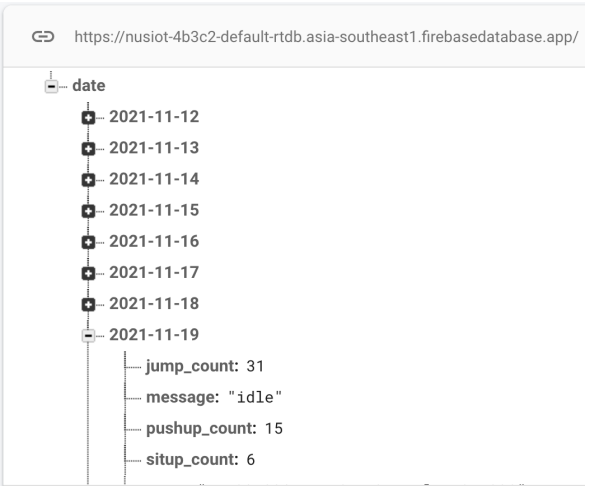


Fig 12: Sample code of intermediary subscriber that sends data to Firebase DB

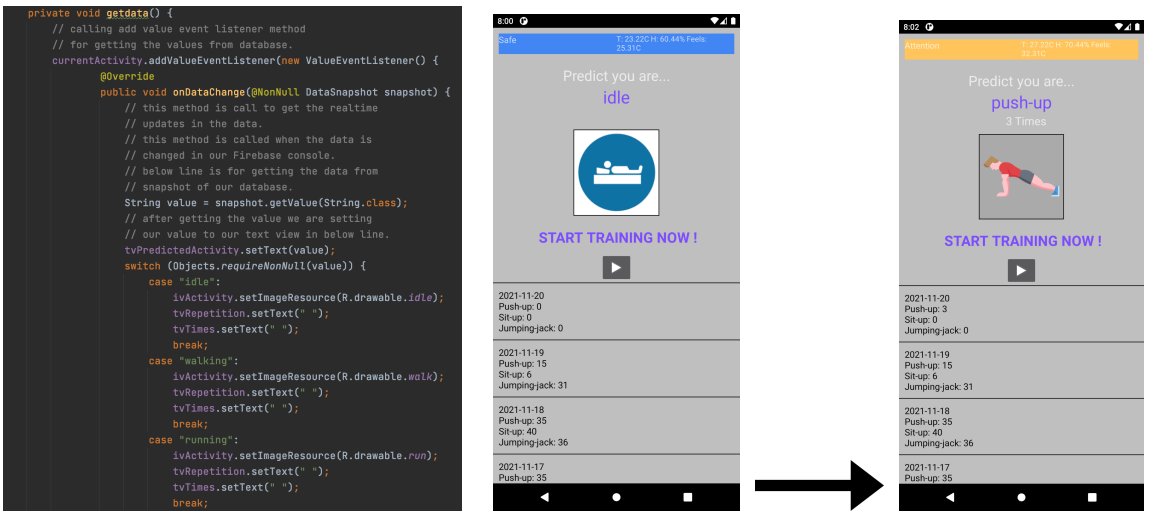


Fig 13: Event listener to render new UI on android app when data is sent to Firebase

Sequence Flow for Image Data

1) User presses the "Start training button" on the android app which will activate their phone camera to capture a live video (Fig 14)
2) Users then go to settings and run classification to start push-up pose detection (Fig 14)
3) The android app will splice the live video to image frames and run Google ML Kit pose detection API to recognise 33 human body keypoints [iii]
4) App will show the human body keypoints in real-time by rendering a graphical overlay on the image (Fig 15)
5) The human body keypoints are then sent to a trained KNN classifier embedded in the app for low-latency reasons to determine if the push-up is "Standard or "Not Standard"
6) If the push-up is determined to be "Not Standard" with a confidence score greater than 0.7, the application will increment the "Not Standard" pushup counter displayed on the app so that users can obtain feedback in real-time that their current push-up posture is "Not Standard" (Fig 15)

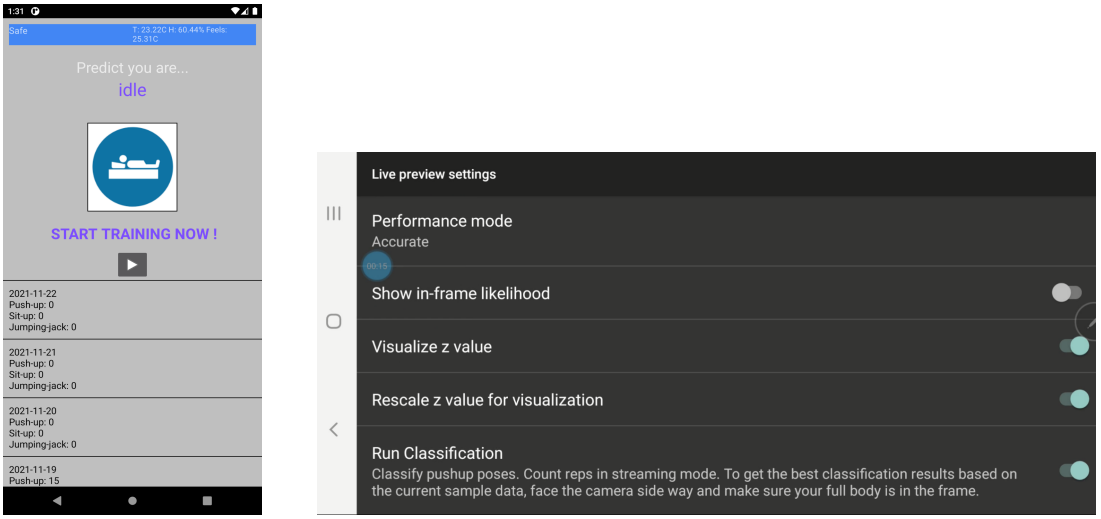


Fig 14: App User Interface to start Pose Detection

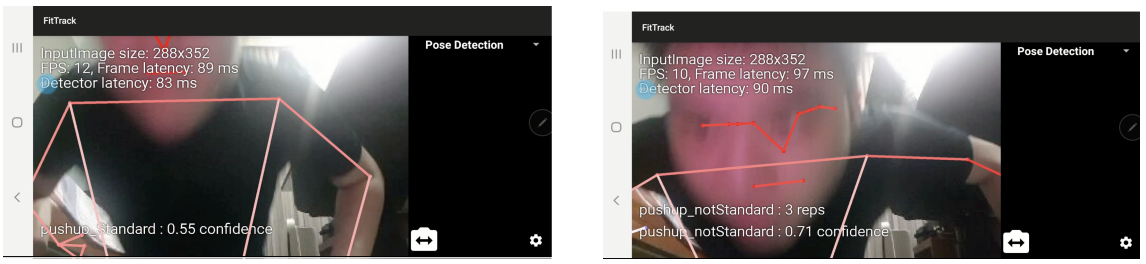


Fig 15: Human Body Keypoint Graphical Overlay and Non-Standard Push-up Counter

3.2 ML Model for Activity Recognition

For our activity recognition, we make use of the wrist and waist accelerometer and gyroscope sensor values in 3-axis as inputs and attempt to classify 6 different activities (Fig 16). From our observations, different activities would have varying magnitudes of movement in different axes. For example, for push-ups, there is a lot of movement for the waist and very little movement for the wrist, while for sit-ups, there is a lot of movement for the wrist and not much movement for the waist. Support Vector Machine (SVM) should be suitable for this classification as the data points should be separable.

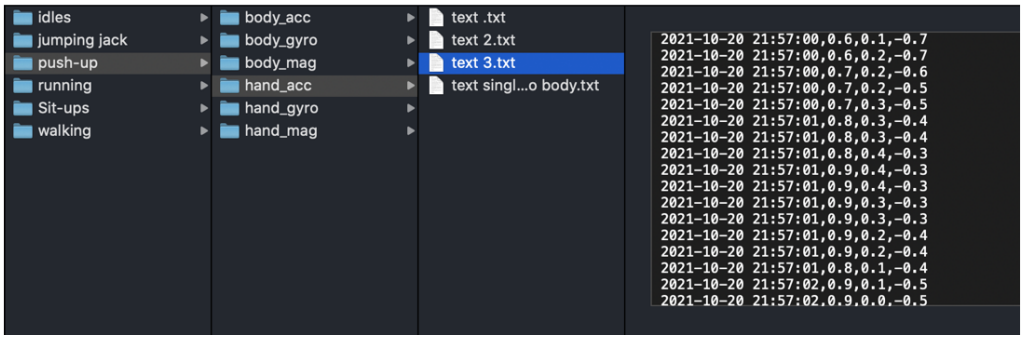


Fig 16: Data collection of physical activity data using TI CC2650 Sensor Tag

However, creating a model for this classification problem wasn't so straightforward. TI Sensor records sensor values at a polling rate of 1 Hz by default which makes it impossible to classify activities. Increasing the polling rate to 10 Hz made it possible to classify activities with a SVM by taking the mean sensor values in 3 second intervals

Referring to the confusion matrix below, at first glance, our SVM model seemed to have very high testing accuracy at classifying activities. However, when testing the model in reality, the model did not work very well as it had difficulty predicting many activities correctly. Some potential reasons are the SVM model overfitting or differences between different people doing activities as we had different group members test out the model.

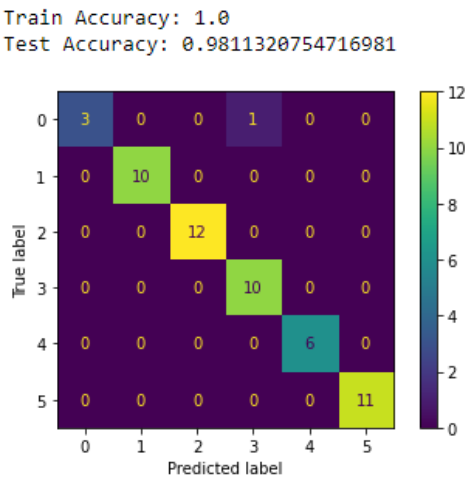


Fig 16: Confusion Matrix of SVM Model without feature selection

We realized that feature selection was required for this problem. More features were extracted from the simple data by calculating values such as the max, min, median, std, var and percentile values.

Recursive feature elimination selects features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from the current set of features recursively. [iv] Recursive feature elimination was used with a logistic regression model to rank the features and the top 20 ranking features were used for our model.

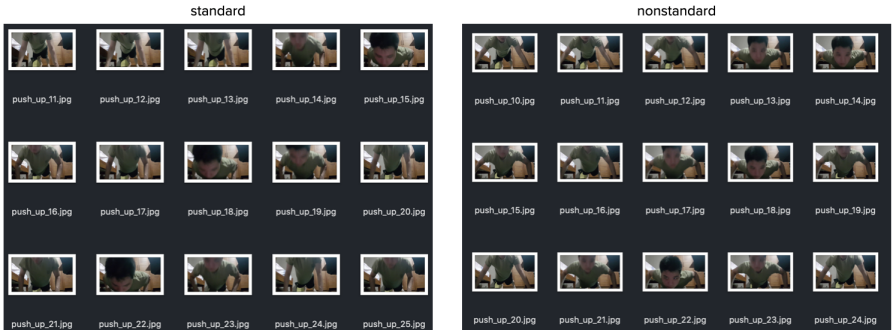
More features can be extracted by using Fourier transforms to obtain a frequency series, as these exercises are repetitive and somewhat harmonic. However, it wasn't necessary as the time domain features gave us sufficient accuracy for the model. After feature selection, we could obtain 100% testing accuracy and the model worked well with our app during testing.

3.3 ML Model for Pose Detection

For our pose detection, The Google ML Kit Pose Detection API was used. The API detects skeletal landmark points which enable us to determine the exact coordinates of key human body parts.

Here are the steps taken for our pose detection.

1. We recorded videos of us doing push-ups and created image samples.
2. The pose detection API was used on the image samples to create a set of pose landmarks (Fig 17)
3. Lastly, the K-Nearest Neighbors (k-NN) algorithm was used to train a model to count repetitions and determine if the pose was correct



push_up_10.jpg	pushup_notStandard	2063.2258	421.34238	54.54825	2098.58	386.27496	74.06054	2098.1892	378.98785	74.50135
push_up_11.jpg	pushup_notStandard	2063.4678	432.2417	60.23303	2095.1401	389.6478	86.81709	2092.864	382.87042	87.30694
push_up_12.jpg	pushup_notStandard	2055.7317	425.65625	22.688732	2090.936	393.50317	41.867237	2090.2947	386.0227	42.271774
push_up_13.jpg	pushup_notStandard	2059.783	429.9131	42.403618	2096.4487	394.93558	57.596428	2096.426	386.42322	58.032764
push_up_14.jpg	pushup_notStandard	2056.0046	419.003	95.51567	2088.9746	378.60364	115.10623	2088.5032	369.84464	115.50451
push_up_15.jpg	pushup_notStandard	2061.4844	414.22504	106.236015	2094.7004	373.272	121.31361	2094.1362	364.138	121.671425
push_up_16.jpg	pushup_notStandard	2079.6616	412.67157	95.93653	2108.0232	368.6398	116.36164	2106.4993	359.39545	116.83487
push_up_17.jpg	pushup_notStandard	2074.6982	403.43347	48.53549	2106.1677	363.98434	63.49245	2104.126	356.482	63.98985
push_up_18.jpg	pushup_notStandard	2091.1487	397.17456	5.4319305	2121.5166	364.20172	24.770569	2119.1785	358.77734	25.431927
push_up_19.jpg	pushup_notStandard	2085.4282	388.27924	69.8566	2113.6292	344.76154	92.05189	2110.826	336.4538	92.59062
push_up_2.jpg	pushup_notStandard	2093.331	398.7181	87.246475	2129.7854	361.11255	103.471375	2127.7437	352.26498	103.88265
push_up_20.jpg	pushup_notStandard	2105.3523	389.22546	-16.458775	2134.6897	352.08887	0.4776717	2132.6724	345.6121	1.0542713
push_up_2_1.jpg	pushup_notStandard	678.3413	301.2346	-1086.445	693.6191	269.82846	-1087.5236	702.9406	267.06186	-1087.6382
push_up_2_12.jpg	pushup_notStandard	592.25775	455.20462	-435.06757	586.8121	456.75223	-439.04492	584.96857	456.82666	-438.6275
push_up_2_13.jpg	pushup_notStandard	612.35205	517.28754	-421.2462	610.7245	516.0661	-446.9485	610.6926	514.9694	-446.80887
push_up_2_14.jpg	pushup_notStandard	640.6082	579.8262	-797.5897	651.1432	558.85986	-840.86993	660.1313	554.0503	-840.74603
push_up_2_15.jpg	pushup_notStandard	490.78702	476.58044	22.30533	480.21063	471.1521	28.075686	480.78488	467.84824	28.284283
push_up_2_16.jpg	pushup_notStandard	534.53436	521.766	-193.13206	531.48694	514.3406	-209.34174	534.19324	511.93713	-208.85774
push_up_2_17.jpg	pushup_notStandard	498.08987	536.789	-318.7094	497.49255	531.12115	-339.89913	502.4786	529.8243	-339.4152
push_up_2_18.jpg	pushup_notStandard	570.3564	496.65262	-457.7636	569.5223	491.1278	-467.35077	571.4444	488.00183	-467.1299
push_up_2_19.jpg	pushup_notStandard	579.67017	479.20193	-518.47784	574.90985	477.59793	-529.9529	576.1512	476.52805	-529.7893
push_up_2_2.jpg	pushup_notStandard	676.30316	301.01874	-1059.4442	691.99414	269.1598	-1061.6512	701.3654	266.40988	-1061.7456
push_up_2_22.jpg	pushup_notStandard	690.57025	293.08102	-889.0519	703.7026	266.808	-892.251	712.06256	264.65347	-892.2805
push_up_2_28.jpg	pushup_notStandard	670.892	318.85504	-357.4047	668.4292	322.45474	-365.47046	666.34656	322.52628	-365.34793
push_up_2_3.jpg	pushup_notStandard	674.65454	298.76477	-1030.2369	690.3084	266.41785	-1032.5319	699.6848	263.73758	-1032.6377
push_up_2_30.jpg	pushup_notStandard	407.01593	446.257	-242.08325	395.27283	433.75876	-248.76619	399.91473	430.198	-248.67961
push_up_2_31.jpg	pushup_notStandard	550.5741	487.48584	-9.068157	545.306	476.1119	-4.942879	547.1675	471.95715	-4.5333586

Fig 17: Image samples of push-up and CSV file containing the pose landmarks

The k-NN algorithm is suitable as we create feature vectors for the different skeletal landmark points and then compute the distance between vectors. For example, for the sample videos on the right, repetition counting of push-ups can be easily done by computing a downward movement of the feature vectors.

Also, depending on the activity, it can be possible to determine if the pose is correct. For example, for push-ups, it is inappropriate to adopt a posture where the arms are spread out too widely. By training a model with both correct and wrong sample data, the k-NN algorithm can easily detect if the posture is wrong by computing the distance between the arms.

3.4 ML Model for Heat Injury Risk Detection

In addition to activity classification and pose detection, FitTrack also has a third feature: heat injury risk monitoring. As the TI Sensors are placed close to our body, the recorded temperature and humidity sensor values are affected by our body temperature and perspiration. As such, we use a ML model to predict the ambient temperature and humidity based on the sensor values. Then, a ML model is used to predict the heat index, which is a human-perceived equivalent temperature. We will then use a threshold to determine if the user is at risk of heat injury based on the heat index. In this section we use two machine learning models and a threshold classifier, which will be elaborated further.

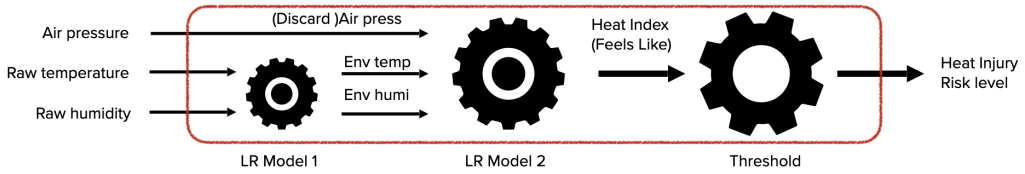


Fig 18: Process for Heat Injury Risk Detection

The first model is a linear regression model, which uses the temperature and humidity data collected by the sensor as input to predict the temperature and humidity of the user's environment. As we found in the actual experiment, the raw temperature collected by the sensor is influenced by the body temperature and does not represent the ambient temperature of the user well, which is generally a little lower than the raw temperature. So we used two sensors, one on the wrist of the user doing exercise to represent the raw data, and one on the desktop away from the user to represent the ambient data, as the input and output of the model, respectively, and after training have been able to correctly convert the relationship between the two. We used this model to design a converter, and the original transformed environmental data was used for the next step of heat index prediction. The following figure shows our example training data:

raw_env_data

Raw temp	Raw humi	Env temp	Env humi
28.4	54	27.1	51
28.7	53.4	27.1	51
28.8	54	27.1	51
25.3	61.3	24.8	56
26.3	57.8	25.0	56
26.6	56.7	25.2	55
26.4	56.5	25.4	55
26.3	56.6	25.6	56
27.1	58.3	26.0	57

Fig 19: Sample of Sensor Training Data

The second model is a linear regression model, which we use to convert the input ambient temperature and ambient humidity into a heat index. The ambient temperature and ambient humidity are converted from the previous linear regression model, while the heat index information is collected from the weather website (www.worldweatheronline.com). It contains 110,000 climate data including hourly temperature humidity barometric heat index for 13 years from 2008 to 2021. We used this large dataset of temperature, humidity and barometric pressure as input and heat index as output to train the relationship between the two and to predict the current heat index when we input the ambient temperature and humidity.

The following are our example training data:

weather_data_1hr

tempC	tempF	humidity	pressureMB	FeelsLikeC	FeelsLikeF
27	80	91	1011	31	87
26	80	90	1011	30	87
26	79	90	1011	30	86
26	79	89	1011	29	85
26	79	89	1011	30	86
26	80	88	1011	30	86
27	80	87	1011	30	87
27	81	82	1012	31	88
28	83	76	1012	32	90

Fig 20: Sample of Weather Training Data

In the third part, we used a threshold classifier to classify the previously obtained thermal indices into risk classes. The standard we refer to is the NOAA thermal risk class reference table, which classifies five thermal risk classes as normal, cautious, especially cautious, dangerous, and especially dangerous, respectively, using thermal indices of 77, 91, 104, and 127 as thresholds. The reference table is shown in the following figure:

	80	82	84	86	88	90	92	94	96	98	100	102	104	106	108	110	112	114	116	118	120	122	124	126	128	130
5	77	79	80	81	83	84	86	87	89	91	93	94	96	98	100	101	104	105	107	108	110	111	113	114	116	117
10	78	79	81	82	84	85	87	89	90	92	94	96	98	100	102	104	107	109	111	113	116	118	120	123	125	128
15	78	80	81	83	84	86	88	90	92	94	96	98	100	103	105	108	111	113	116	119	122	125	129	132	135	
20	79	80	81	83	85	86	88	90	93	95	97	100	103	106	109	112	115	119	122	126	130	134				
25	79	80	82	83	85	87	89	91	94	97	100	103	106	109	113	117	121	125	129	134						
30	79	80	82	84	86	88	90	93	96	99	102	106	110	114	118	122	127	132	137							
35	80	81	83	85	87	89	92	95	98	102	106	110	114	119	123	129	134									
40	80	81	83	85	88	91	94	97	101	105	109	114	119	124	130	136										
45	80	82	84	87	89	92	96	100	104	109	114	119	124	130	137											
50	81	83	85	88	91	95	99	103	108	113	118	124	131	137												
55	81	84	86	89	93	97	101	106	112	117	124	130	137													
60	82	84	88	91	95	100	105	110	116	123	129	137														
65	82	85	89	93	98	103	108	114	121	128	136															
70	83	86	90	95	100	106	112	119	126	134																
75	84	88	92	97	103	109	116	124	132																	
80	84	89	94	100	106	113	121	129																		
85	85	90	96	102	110	117	126	135																		
90	86	92	98	105	113	122	131																			
95	88	94	101	109	117	127	137																			
100	89	96	104	112	121	132																				

Likelihood of Heat Disorders with Prolonged Exposure or Strenuous Activity

Caution Extreme Caution Danger Extreme Danger

Fig 21: NOAA thermal risk class reference table

3. Implementations

4.1 Sensor Sampling Frequency

We perform repetition counting by analysing the wave signals generated by our activities in the 3 axes. After determining which axes are suitable for which activities, we do peak counting to count repetitions.However, The original 1 Hz adoption rate of the accelerometer and gyroscope was not enough to observe valid wave information or capture faster action detail information for machine learning to classify actions, so we analyzed the sensor tag BLE SDK and found the relevant profile for the accelerometer sensor. The sampling frequency of the accelerometer was changed to 10Hz which provided us with enough information for machine learning and counting.

```
#define MOVEMENT_INACT_CYCLES   (MOVEMENT_INACT_TIMEOUT * \
                                 (10000/sensorPeriod) / 10)

/*********************************************************
 * CONSTANTS and MACROS
 */
// How often to perform sensor reads (milliseconds)
#define SENSOR_DEFAULT_PERIOD     100

// Length of the data for this sensor
#define SENSOR_DATA_LEN           MOVEMENT_DATA_LEN

// Event flag for this sensor
#define SENSOR_EVT                ST_GYROSCOPE_SENSOR_EVT
```

Fig 22: Relevant code to change polling rate

4.2 Power Management

Meanwhile, in order to save power, we adjusted the sensors that do not need real-time information, such as temperature and humidity sensors, to be turned on every fifteen minutes to obtain data and send it to the server. To save power, we opted to turn off the sensors that are not in use such as the light sensor. This can be done by simply not initializing the sensors on the BLE SDK.

```
/* Kick off application - Priority 1 */
SensorTag_createTask();
SensorTagTmp_createTask();
SensorTagHum_createTask();
SensorTagBar_createTask();
```

Fig 23: Relevant code to initialize sensors

4.3 Communication Protocol

BlueTooth low energy transfer

For sending data from our TI CC2650 sensor tags to the laptop, BLE transfer was used as it offers very low power consumption and hence battery life of sensors can be extended. Furthemore, the 1 Mbps/2 Mbps data transfer rates are suitable for our project since the sensors are transmitting only small amounts of data. Moreover, the advertising channels for BLE are different from wifi channel frequencies to avoid interference between wifi.

MQTT

For sending the sensor data received by the laptop for machine learning, MQTT was used as it is a lightweight publish/subscribe messaging protocol designed for low-bandwidth and low latency environments making it an excellent option for sending the high volumes of sensor data received by the laptop for pre-processing and machine learning. Furthermore, since we are running more than one machine learning model, the publish/subscribe protocol allows us to easily organize and route the motion and environmental data to their respective machine learning models for prediction

HTTPS

For sending data from laptop to the cloud, HTTPS was used as it allows for data to be securely sent across the network to Firebase Realtime DB. In addition Firebase only serves traffic over SSL. We also selected our data centre hosting our data to be in Singapore to reduce latency

WebSockets

For server and android app communication, websockets were used.
The rationale for this was low latency and speed since in websockets data frames can be sent back and forth between the client and the server in full-duplex mode and connection is maintained once established. This allows users to be able to get real-time instant feedback on their app whenever data in the server changes

4. Experiment Evaluation

5.1 ML Accuracy

Activity Recognition

For our activity recognition SVM model, we were able to obtain 100% testing accuracy as seen in the confusion matrix below. During real life testing, the activity recognition works perfectly most of the time. However, during transitions between activities, the activity recognition may temporarily classify the activity incorrectly. For example, when standing up from a sit-up position, due to the motion of standing up, the ML model may incorrectly classify the activity as a jumping jack. However, apart from transitions between activities, our model works well.

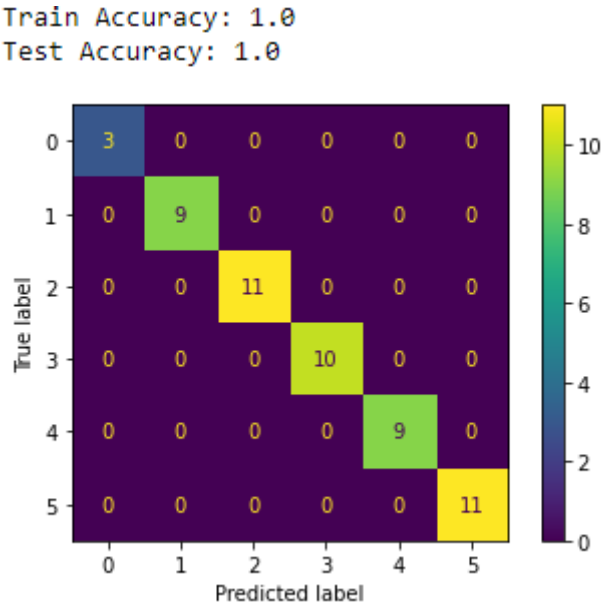


Fig 24: Confusion Matrix of Activity Recognition Model

Heat Injury

In the heat injury risk detection section, we originally planned to use three inputs of temperature, humidity and barometric pressure, but when we tested the model with three different inputs, we found that humidity significantly helped the accuracy of heat index prediction, but barometric pressure contributed little to the prediction accuracy, so we removed the barometric pressure input. The accuracy of the model with the three inputs is shown in the following figure.

```
Model using temp:
Train accuracy: 0.939949754769459
Test accuracy: 0.9395040100036047

Model using temp, humi:
Train accuracy: 0.9702990252122644
Test accuracy: 0.969845854848203

Model using temp, humi, press:
Train accuracy: 0.9704791272955113
Test accuracy: 0.9700106395535958
```

Fig 25: Accuracies of Heat Injury Models

After using temperature and humidity as inputs, we can predict the heat index with an accuracy of 97% on both the training and test sets.

Posture Detection Accuracy

Initially, when testing the KNN classifier model on a sample of test images for non-standard and standard push-up, the accuracy was very poor (<0.5) as the first model relied heavily on Push-up images collected online.

Hence, we decided to collect a large sample size of our own push-up images taken at various camera distances to re-train the KNN classifier. This improved the testing accuracy significantly and we managed to obtained ~0.9 accuracy

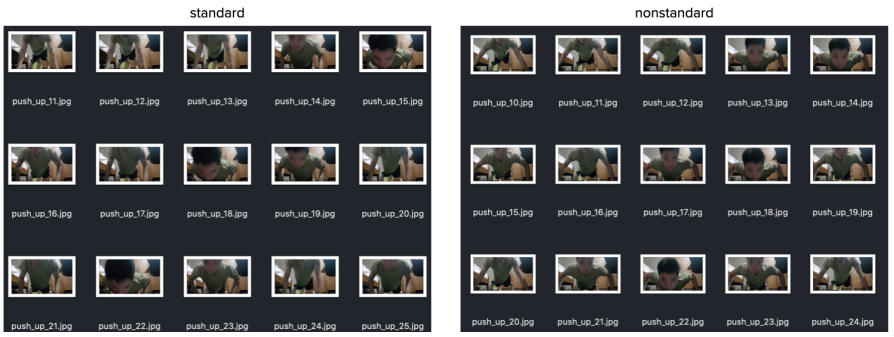


Fig 26: Image Samples for Push-ups

Fig 27: Confusion Matrix of Pose Detection Model

5.2 Battery Life Estimation

The following picture shows the power consumption after modifying the BLE SDK. Fig 28 below shows the power consumption when all the sensors are turned on and the accelerometer is working at 1Hz by default. It consumes roughly 7 percent of battery in half an hour.

Fig 28: Battery Consumption with default BLE SDK

To save power, we turned off the unused sensors and drastically reduced the polling rate for the temperature and humidity sensors to once every 15 minutes. However, we increased the polling rate of the accelerometer and gyroscope from 1 Hz to 10 Hz. Despite this, we managed to reduce power consumption. As you can see in Fig 29, only 4 percent of battery was consumed in half an hour.

Fig 29: Battery Consumption with optimized BLE SDK

5. Summary

6.1 Challenge

Model Training

For challenges one issue would be model training. In order to train our models to attain a high accuracy, we had to collect thousands of acceleration and gyrometer datasets for the respective activities. However, the accuracy was still poor and hence we had to adjust the sensor frequency and re-collect the data again which was very time consuming. In addition,we had to video push-ups from different angles and distances to ensure the posture recognition feature was able to recognise standard and non-standard push-ups from different camera distance

Developing Low-Latency Application

Also, for our application to function optimally we had to pay special consideration to latency. Hence, we had to explore various low latency technologies that we are not very familiar with such as firebase real time database, Tensorflow Lite, Google ML KIT for edge devices to ensure our system can provide that real-time feedback functionality that is critical for IOT sports products

6.2 Limitation

Unable to distinguish human activities that are similar in motion

For activity recognition, there are scenarios where the feature may recognise the wrong fitness activity since it is reliant on the gyrometer and accelerometer. For example, if the user is lying down and suddenly wakes up, our ML model at times detects him as doing 1 sit-up. Also if the user reaches down to the ground in quick fashion, it may get detected as push-up. This is caused by the sensor orientation and hence, our product will still need more data collection for the various human activities for it to become more robust and accurate

Posture Recognition Constraints

In addition, while our posture detection is able to recognise wrong push-ups, there are certain constraints. For example distance. If the user is too far/too close from the app camera, then this feature may be unable to utilise the key points optimally to determine his posture as our image data bank is not extensive enough to cover a range of distance. Also, since we are utilising Google ML kit for human keypoint detection, this feature may have performance issues when more than 2 people are in the image frame as the ML kit works best when only one person is in it [vi].

6.3 Future direction

Remove intermediate hardware

Our system design, although functional, is not the most ideal since there is a laptop as an intermediary. Hence, one path forward is to revamp the system to remove the laptop and utilise BLE for direct data transfer between the sensor and the android app. This will make our system much more lean and it also facilitates the use cases of our product for outdoor physical activities.

Increase the fitness activity recognised and display health metrics

We do hope to expand the activity recognition feature such that the app can also record other data such as squats, planking and cobra stretch and number of steps taken in a given day. In addition, we do believe that an IOT device can only be complete if there are health metrics such as calories burnt, blood pressure, water intake, sleep time. Hence, all these could be added to build a fully holistic and comprehensive IOT fitness product.

Commercialisation

Given how health and wellness is a trendy investment trend these days, as more people are health conscious and want to live a healthy and balanced lifestyle, We do believe that IOT fitness products have tremendous market potential. Hence, to get active users flowing to our product, we could reach out to officials in the Ministry of Education and Ministry of Defence to get them to use FitTrack on a trial-basis for students and servicemen to train for their fitness exam.

References

[i] South China Morning Post. Available at: https://www.scmp.com/news/china/society/article/2098042/china-has-largest-number-obese-children-world-says-study [Accessed November 21, 2021].

[ii] HealthExchange.sg. Available at: https://www.healthxchange.sg/children/parenting-tips/childhood-obesity-tips-parents [Accessed November 21, 2021].

[iii] Real-time Body Pose Tracking with MediaPipe BlazePose Available at: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.ht

[iv] Anon, RFE. scikit-learn. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html [Accessed November 21, 2021].

[v] Singapore Historical Weather. Available at: https://www.worldweatheronline.com/singapore-weather-history/sg.aspx [Accessed November 21, 2021].

[vi] Pose Detection. Available at: https://developers.google.com/ml-kit/vision/pose-detection [Accessed November 21, 2021].