PDIoT Final Report(2022-23)

Implementation of the human activity recognition system through an Android app

Group M
Xinchen Chen (s2269664)
Ran Yi (s1910268)
Dianze Li (s1973139)

19 JANUARY 2023

School of Informatics
University of Edinburgh

Content

Abstract

With the development of today's Internet of Things technology, real-time human activity recognition systems can play a major role in fields such as medical care, mainly through the use of sensors to continuously monitor physical activity. In this assignment, we build an Android APP that analyzes the current human behavior by continuously collecting data from two sensors, Respeck and Thingy, and importing it into a convolutional neural network model. In this article, we will first briefly describe the content, methodology and results of our project. Next, we will review the latest progress in the field of human activity recognition algorithms through some research papers. After that, we will introduce in detail the specific methods we used in realizing this system, including hardware, wireless communication, machine learning model, application programs, software structure, testing and other aspects. Then we will conduct a quantitative analysis of our system, mainly about the recognition accuracy of the application for real-time and offline. We will also test the performance of the system. Finally, we will summarize our project and discuss how this project can be improved and expanded in the future.

1. Introduction

1.1 Project aims

The purpose of this project is to design and implement an IoT system, which can be used for real-time human activity recognition, mainly through the wireless connection of two IMU sensors Respeck and Thingy and the application of some machine learning techniques. The system can identify a total of 14 different human activities. Users can pair two sensors with our Android APP, register and log in the APP interface to view his current activities, and each user can make the sensor more suitable for his/her own body data through the calibration function before using. At the same time, the APP will also record the user's activities, so that users can view their historical records conveniently. When it is recognized that the user is walking, running or having other movements, the number of steps will be recorded by the system. In addition, when the user is detected to be sitting for too long, the system will issue a warning to the user through the APP. We also provide a cloud mode, which allows users to obtain the latest model when they are connected to the Internet so as to get more accurate live recognition data. These functions are designed so that the system can help users maintain a healthy and healthy lifestyle while detecting user activities.

On the other hand, through the process of doing this project, we also got an opportunity to participate in the realization of a complete complex IoT system from various stages, understand and learn different aspects of knowledge and principles. We started from collecting data, using machine learning technology to process and train the data. At the same time, each of us has gained something in different aspects such as Android application development, software organization, interface design, performance testing and evaluation. Such project experience is undoubtedly very helpful for the comprehensiveness of our acquired knowledge and our future career development.

1.2 Brief description of the method adopted

The system is developed on the mobile Android platform with an intuitive UI, which enables users to use it conveniently. Users can register an account, utilize the recognition system on their smartphone in real time, and check previous activity recordings.

We tried several models to build the AI model for human activity recognition. The Convolutional Neural Network (CNN) model works the best among them all. It takes multimodal connection and translational invariance into account, both of which contribute to activity recognition. We also tested different combinations of parameters and reached an accuracy of 97%.

1.3 Physical activities used in the classification

In our project, we have a total of 14 different classifications of human activities, they are respectively:

- Sitting
- Sitting bent forward
- Sitting bent backward
- Standing
- Lying down on back
- Lying down left
- Lying down right
- Lying down on stomach
- Walking
- Running
- Climbing stairs
- Descending stairs
- Desk work
- Movement

For different actions, when Respeck and Thingy are activated at the same time, we will use the results of different models or use a specific proportion of the results of the two models to judge, so as to obtain more accurate judgment results.

1.4 Summary of results

In our application, we have two CNN (convolutional neural network) models, one for Respeck data and the other for Thingy data. We do not have a separate model for the hybrid mode of Respeck and Thingy, the results of this mode are obtained by linearly combining the results of the two models. We tested the models on 5-fold leave-one-subject-out cross-validation (LOSOXV) to measure the accuracy.

After constant adjustment of various parameters and the structures, our separate Respeck model can achieve an overall accuracy of more than 97% at a dropout rate of 50%. For the Thingy model alone, we can reach an 96% accuracy at a dropout rate of 40%. From the results, we can observe that each of these two single models has some actions that perform not well in recognition. So, in the hybrid mode, when these two models are used together to complement each other, we can achieve the highest final accuracy, which is about 99%. The methods and analysis of the tests will be introduced and discussed in detail in the subsequent sections 3.7 and 4.1.

2. Literature Survey

2.1 Background

Human activity recognition can be described as the art of collecting raw data on human activity through the use of artificial intelligence and multiple sources and then identifying and naming these activities. The devices used to collect the data are diverse and they include sensors (wearable sensors, smartphone inertial sensors), cameras (optical), frequency-emission recognition, etc [1].

Because HAR systems have the ability to capture and analyse data, they can help healthcare practitioners make quick decisions and diagnoses. The potential of HAR for elderly care and life improvement has led to a growing interest in HAR [2].

Because of their ability to collect and analyse data, HAR systems have gained popularity in recent years with the development of machine learning algorithms and neural networks in many areas: healthcare, improving human-machine interaction, security systems, industrial mechanisation, athlete training monitoring, robot monitoring, rehabilitation systems, etc [2]. In particular, HAR has the potential to help medical practitioners to make quick decisions and diagnoses in the medical field, as well as in elderly care and life improvement [3]. The rapid development of machine learning algorithms and neural networks in recent years has enabled HAR to be used in a variety of fields [1].

In general, a HAR system can be divided into several parts: perception, segmentation, feature extraction, classification and pre-processing. HAR can be divided into two categories based on sensing methods: vision-based and acceleration-based. By using vision-based HAR methods, the user is required to wear one or more cameras in order to collect data, and the accuracy of this method is greatly dependent on the lighting conditions at the time of data collection. Viewing angles and other physical external factors, but with this method, the user does not have to wear any equipment. Whereas using the acceleration-based method, although it requires the user to wear an accelerometer to collect the data, almost the data collected is not affected by external factors as it is with the vision-based method [4], and this method also protects user privacy and is less expensive to use with accelerometers [9].

2.2 Traditional ML methods for HAR

Many traditional machine learning algorithmic approaches, such as Random Forest (RF), K-Nearest Neighbours (KNNs), Support Vector Machines (SVM) and Hidden Markov Models (HMMs) have been applied to solve HAR problems, but as these methods rely on multiple pre-processing steps and the use of shallow features answer only unsupervised or incremental

learning with unsatisfactory performance [5].

2.3 Deep Learning

Compared to these traditional ML algorithms, deep learning automatically extracts abstract features, making the selection of the right features much less laborious, and can work alongside unsupervised [6][7] and reinforcement learning [8]. In addition to this, Deep learning algorithms are particularly adept and efficient in processing time-series signals to extract features and classification because to the advantages of local dependency and scaling invariance [11]. This is why deep learning-based methods: Artificial Neural Networks (ANNs), Convolutional Neural Network (CNNs), and Naïve Bayesians are being introduced in HAR. CNN feature extraction has several advantages over standard shallow learning algorithms in the HAR domain, including local dependency and scale invariance, as well as the ability to capture all conceivable complicated nonlinear interactions between features [12].
While there are many ML classifiers currently being used to classify HAR datasets, it is important to find the most appropriate classification for the dataset of interest in order to accurately identify human activity, and there is a major trend towards using neural networks for classification work [18].

2.4 Convolutional Neural Network (CNN) Model

The authors of [13] suggest a new CNN design that employs a convolutional layer (1*9.1*14) with a small pooling size (1*2.1*3). The proposed approach was tested on a CNN with the temporal features of the original Fourier transform and FFT (Fast Fourier transform) signals.
In [14] authors proposed using a CNN to extract local features in addition to statistical features which including mean, variance, sum of absolute values, and histogram for each input in order to keep information about the original signal's global features. By setting the sliding window to 1 second, this body of work as well investigates the effect that signal length has on performance.
In [15], the authors argue that deep CNNs suffer from the computational power of wearable devices for wearable HAR tasks, and that the large amount of computation time required for deep learning is not suitable for real-time HAR on wearable devices, traditional machine learning methods cannot have good performance. The authors propose a new efficient solution: they have implemented real-time HAR on mobile and wearable devices by using CondConv (Convolution with conditional parameterization) instead of traditional convolution, and their proposed method can greatly improve the recognition accuracy of existing HARs using CNNs without compromising computational cost, making it ideal for HARs with strict latency constraints.

2.5 Long-Short Term Memory (LSTM) Model

The LSTM model is superior to other models when it comes to forecasting the initial sequence of time series signals. Additionally, this model classifies time series data by processing the complete data series via feedback connections.

Chung et al. built a testbed in their paper [16] that included eight wearable inertial measurement unit (IMU) sensors and an Android mobile smartphone that gathered data on activities. They also created a long short-term memory network architecture to enable the training of deep learning models on human activity data (obtained from a controlled real-world setting). This was done in order to facilitate the training of deep learning models on human activity data. They then experimentally demonstrate that classifier-level sensor fusion techniques can improve classification performance, and the authors hope to conduct further research into other sensing modalities such as environmental sensors (GPS, network location, background sound), as well as physiological sensors (ECG, skin electrical activity), in order to further implement sensor fusion techniques and adjust the weights of each sensor modality in order to improve recognition accuracy.

In point of fact, the present system that the authors have described is nothing more than a straightforward activity recognition testbed for a limited number of participants; it is not suited for use in applications on a broad scale.

The majority of the time, the HAR system has typically been addressed by making use of features that have been gathered via the use of heuristic approaches. Typical ML algorithms can easily be led astray towards finding a local minimum rather than a solution that is globally optimum, and the problem of low efficiency also exists.

In [17], The authors base their novel approach, which is a hybrid deep framework built on CNNs, LSTM recursive units, and ELM classifiers, on these challenges and suggest a new method as a solution to them. Their hybrid framework does not rely on expert knowledge in order to extract features and is better suited for categorising the features that have been extracted in a shorter amount of time. According to the results of the evaluation, their method is superior to deep non-recursive networks and cuts processing time by 38% when compared to neural networks based on BP algorithms.

3. Methodology

3.1 System Description and Implementation

The system is a real-time Human Activity Recognition (HAR) IoT system that utilises wireless

Inertial Motion Unit (IMU) sensors and machine learning techniques. The system allows users to pair a Respeck or a Thingy device and view their current activity.

The system integrates and covers all three categories of features.

Besides the basic requirements, the system offers desirable features such as classifying all activities, using both sensors for improved accuracy, and an intuitive user interface with user logins and the ability to view past data.

The system's advanced features include the ability for users to calibrate the sensor to their own body, live classification in the cloud, notifications for reminders to move, step counting and an accuracy of 97%. The system is designed to monitor human activities comprehensively and accurately in real time.

The fundamental logic consists of reading the Bluetooth data, running it through the model, and displaying and storing the result.

After registering and logging in, users were able to begin using the application.

Prediction using only Respeck or Thingy; after running the model and obtaining the result, the app will display the detected activity, save the result, and display live data on the screen.

For the purpose of recording history, a histogram is used to visualise the activities, and the user and date tally the number of steps.

Then, we utilised both sensors to improve the outcome. Combining the results of the two sensors linearly and adjusting the weight based on performance is the proper procedure.

The app can send a reminder when detecting sitting for a long time by increasing a variable when sitting and deskwork is detected; a notification will be sent if the variable reaches a certain value.

In order to improve accuracy further, a state machine is employed to prevent abrupt transitions, such as from lying flat to running.

In addition, we designed a function to determine whether the receptacle is in the correct position by requiring the user to perform two simple tasks and examine the data reading.

Finally, we utilised Firebase as one of the cloud services for machine learning. The application will retrieve the cloud-based model and perform classification. This feature enables us to update models easily. In addition, we created a Java server to enable live classification on LAN and simulate a server in the public cloud.

The following figure 3.1(a) shows the whole structure of our system.

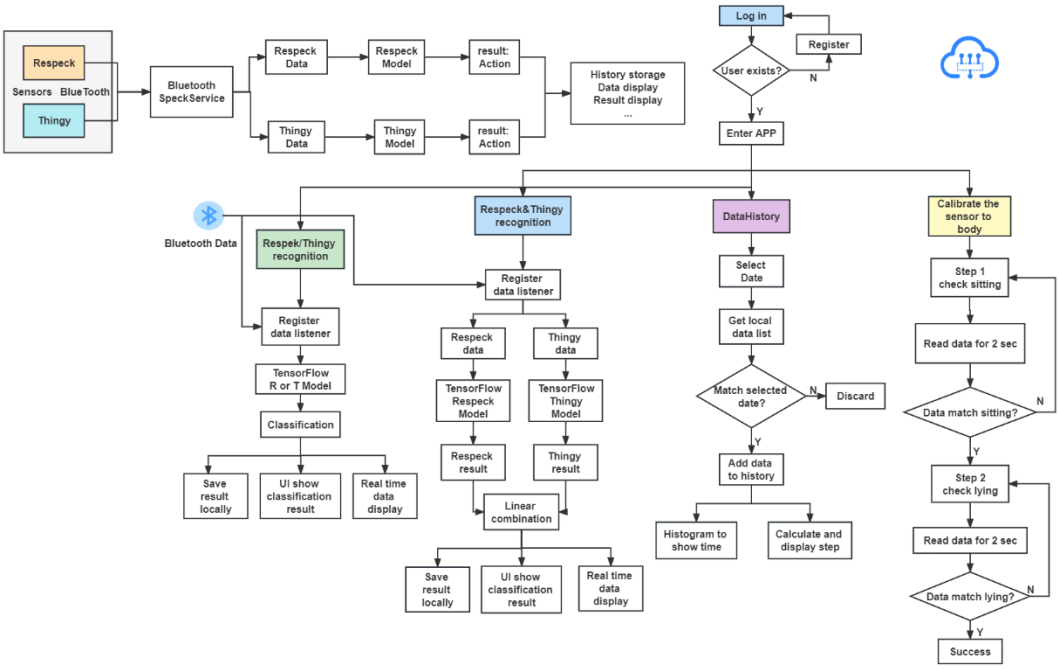


Figure 3.1(a): The architecture of the system implemented

3.2 Hardware and firmware

In this project, we mainly used two IMU (Inertial Motion Unit) hardware devices: Respeck and Thingy.

3.2.1 Respeck

Respeck is a wearable IoT device that incorporates a 3-axis accelerometer and a gyroscope sensor. When in use, it should be worn under the left rib cage with the MeFix tape and paired with the Android APP so that it can monitor physical activity by collecting data from the accelerometer and gyroscope and sending these packets to the APP through Bluetooth Low Energy (BLE) at a frequency of 25Hz. Figure 3.2.1(a) below shows the appearance of the Respeck device, and figure 3.2.1(b) shows the wearing position of the Respeck device and the directions of the three measurement axes.

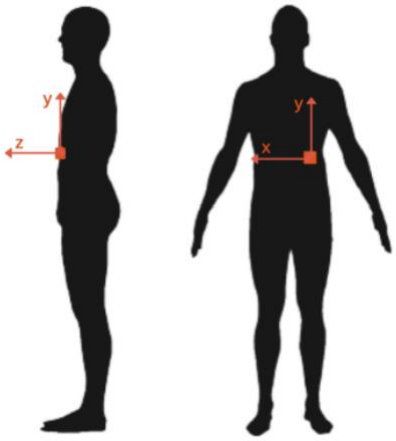


Figure 3.2.1(a): Appearance of Respeck Figure 3.2.1(b): Placement and Direction of axes of Respeck

The accelerometer in the device can be used to measure the acceleration to determine the motion state of the human body. The gyroscope is used to monitor the angular movement. Since the accelerometer cannot distinguish the direction and position of the human body's movement, using it together with the gyroscope can obtain more information about the human body's activities, thereby more accurately judging the human body's activities.

At the same time, the wearing position of the device Respeck is very close to the centre of gravity of the human body. Most of the human activities have clear movements in this position, and this position can also detect the breathing state of the human body, which is conducive to distinguishing the types of human activities.

3.2.2 Thingy

Thingy is an IMU prototyping platform developed by Nordic Semiconductor. It includes a 3-axis accelerometer and gyroscope like Respeck. In addition, it also includes a magnetometer sensor. This magnetometer sensor can measure the angle between the current device and the four directions of east, west, north and south, which is used to test the strength and direction of the magnetic field and locate the orientation of the device.

Unlike the Respeck, Thingy is worn in the right front pocket of the trousers, while the circle on the instrument should remain in the upper right corner. Such a wearing position makes Thingy more sensitive to leg movements and difficult to judge among actions with similar lower body movements such as different types of sitting and desk work. Therefore, in this project, the main function of Thingy is to use it with Respeck to improve the accuracy of recognition of some actions. It also needs to be paired with the Android APP and works at the same frequency as Respeck of 25 Hz. The following two figures show the appearance of Thingy and its wearing position.

Figure 3.2.2(a): Appearance of Thingy

Figure 3.2.2(b): Placement of Thingy

3.3 Wireless communication

In this project, the communication between our application and two sensors is implemented by BLE (Bluetooth Low Energy), a Bluetooth technology that can significantly reduce power consumption and cost while maintaining the original communication range. The technology is a novel application in healthcare, fitness and more, where it is often used to transfer small amounts of data between nearby devices or interact with proximity sensors. Using this technology, after the Respeck and Thingy sensors are paired with the app, they can send real-time data packets to the app at a frequency of 25Hz. In this project, we use the RXAndroidBLE library in Java to achieve these.

3.4 Machine Learning methods for activity recognition

3.4.1 Structure

In this project, in terms of machine learning, we use a sequential CNN (Convolutional Neural Networks) model, which consists of many neural network layers. This neutralization usually involves alternating convolutional and pooling layers. The depth of each filter in the network increases from left to right, and the last usually consists of one or more fully connected layers. The following Figure 3.4.1(a) shows the structure of our model for machine learning. For some of the parameter settings, the number of filters is 128, the kernel size is set to 3, the dropout rate is 0.5 for Respeck and 0.4 for Thingy and the pooling size is 2.

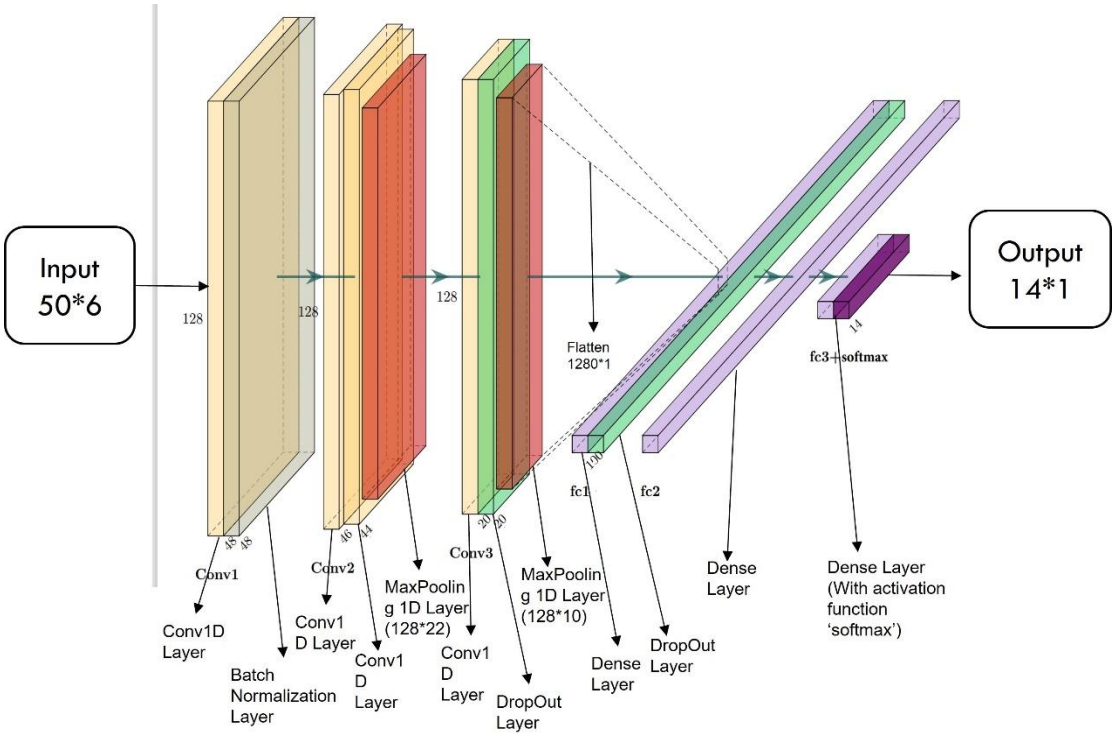


Figure 3.4.1(a): The architecture of our CNN model

Input Layer: In this layer, our input consists of a series of data with the dimension of 50*6, where 50 is the window size of the model and 6 is the number of data features. These six features are collected from the Respeck and Thingy sensors, which are the three-dimensional data of the accelerometer and the three-dimensional data of the gyroscope respectively.

Convolution 1D Layer: In our model, we set a total of 4 convolution 1D layers. This layer is used for text data and only performs convolution on the width, not on the height. Each convolutional layer is composed of several convolutional units, and the purpose of these layers is to extract feature information in the input data through convolutional operations. A single convolutional layer can only extract some low-level features, and we use multiple convolutional layers here to extract more complex features to improve the performance of the model.

Batch Normalization Layer: After the first Convolution 1D layer, we added a Batch Normalization layer, which is mainly to reduce overfitting. This layer implements a pre-processing operation in the middle of the neural network layers, which can correlate all samples in a mini-batch so that the network does not generate a certain result from a certain training sample. This effectively prevents gradient dispersion and accelerates network training.

MaxPooling Layer: Between and after the Convolution 1D layer, we inserted a total of two MaxPooling layers also to try to reduce overfitting. Pooling is actually a downsampling operation in convolutional neural networks. The maximum pooling here is to divide the input into many rectangular areas and output the maximum value for each area. We use the maximum pooling layer here to reduce the size of the space occupied by the data, which can also reduce the number of parameters and calculations required.

Dropout Layer: In this model, we used two dropout layers. The role of this layer is to randomly

disable the weights of the set proportion of hidden nodes when training the model, so that these nodes are temporarily not considered in the network. But this does not mean deleting these nodes, because they may still be used when the next sample is input. This can avoid overfitting when the training samples are small.

Flatten Layer: This is used for flattening the tensor so as to make the structure fit with the form of the output.

Dense Layer: We used a total of 3 Dense layers in the model. After going through the previous layers such as Convolution 1D and MaxPooling, we have extracted many features of the data, and the role of this layer is to classify according to the combination of features and reduce the impact of feature positions on classification. In general, each feature node has a certain weight in determining which category the input belongs to, and the probability or weight of the category that the final input belongs to is determined by combining the weights of all features.

Activation Function: In neural networks, we need activation functions to increase the nonlinearity of the model. This is because if there is no activation function, each of our layers is just doing matrix multiplication, and the output is also a linear function, and it is impossible to approximate any function with nonlinearity. Therefore, the activation function can make the data better classified and make the neural network better solve complex problems. In this model, except for the last Dense layer, the activation functions we use are all ReLu. This is because the ReLu function uses threshold calculation, which is simpler and more efficient and effectively alleviates the problem of gradient disappearance so that it performs well in convolutional networks. In the last Dense layer, the activation function uses softmax. This is because we want to finally output the probability of an activity and the softmax function is to convert the output into a probability form.

Output Layer: For the final output, since we have a total of 14 different physical activities, the dimension of the classification result should be in the form of 14*1.

3.4.2 Model Training

Loss function: The machine learns through the loss function, which is used to measure the gap between the output of the model and the real value, and to point out the direction for the optimization of the model. The loss function is divided into classification loss and regression loss. In the classification task, we want to predict the output from the data set with limited category values, while the regression problem deals with the prediction problem of continuous values. In this project we are doing a classification task, so our loss function will be selected from the classification loss. And because we're dealing with multi-classification problems here, tasks in which an example can only belong to one of many possible classes, the model has to decide which class it belongs to. So, we finally chose the categorical crossentropy loss function. Formally, it aims to quantify the difference between two probability distributions. In order to

verify the results of this theory, we also tried some other loss functions in the model as tests, and the results did show that categorical crossentropy is the best performing loss function.
Optimizer: An optimizer in machine learning is used to tune the parameters of a neural network to minimize a loss function. Therefore, the choice of the optimizer can determine the quality of the training results. Here, the optimizer we choose is Adam, which absorbs the advantages of Adagrad (gradient descent algorithm with adaptive learning rate) and momentum gradient descent algorithm, which can not only adapt to sparse gradients, but also alleviate the problem of gradient oscillation. Not only is it currently the most used optimizer, it is also the best performer among adaptive optimizers in most cases. In practice, we also tried other different optimizers including SGD combined with our selected loss function for testing. We found that they all performed worse than Adam and the learning rate adjustment problem is a complex problem, so we finally settled on the Adam optimizer.
Pre-processing and training: In this project, we used the provided clean data for 2022 and 2021 to train the model. Among them, for the data of 2021, we excluded the data that included the action of falling because this is not within the functional scope of this project. Before the data enters the model, we first preprocess them, which requires us to use a sliding window algorithm. Here, we set the window size to 50 and the step size to 25 to achieve 50% overlap, which is based on the frequency of the two sensors being 25Hz and the actual test results. At the same time, during data training, we performed a 80%: 20% cut, 80% of the data was used for training, and the remaining 20% was used to test the training effect. Finally, we tested the model on 5-fold leave-one-subject-out cross-validation (LOSOXV) which is essentially pooling the means to measure the true validation accuracy of the model.

3.5 Mobile Application

The app was built using Android SDK Build Tools 30.0.1 and Android Gradle Plugin 7.0.2. The compiler warns that the app is incompatible with Android Gradle Plugin8.0. The minimum SDK version required is version 23, the target SDK is version 30.

3.5.1 Sign up & Sign in

For new users who are using the app for the first time, registration is required. Click on Sign up in the bottom left corner as shown in the Figure 3.5.1(a) to sign up. On the registration page, users will need to enter, respectively, Name, Email, Password and Retype Password to complete the registration. As the Figure 3.5.1(b) shows. The email address will be used to sign in.
After completing your registration, return to the start screen, enter the email and password, as Figure 3.5.1(a) shows, then click the Sign in button to sign in. When exiting the app and

returning, users will need to enter their information again like this to sign in.

Figure 3.5.1(a): Sign in page

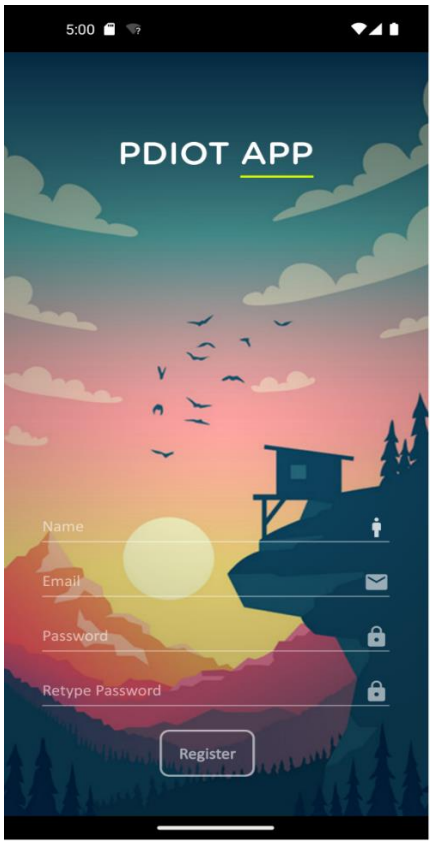


Figure 3.5.1(b): Sign up page

3.5.2 Home Menu

After completing the registration, after three short pages of instructions on how to use some features, the user can click on Skip to skip. Go to the Main screen as shown in Figure 3.5.2, there are 10 buttons as shown, we will introduce them in the next in order of use.

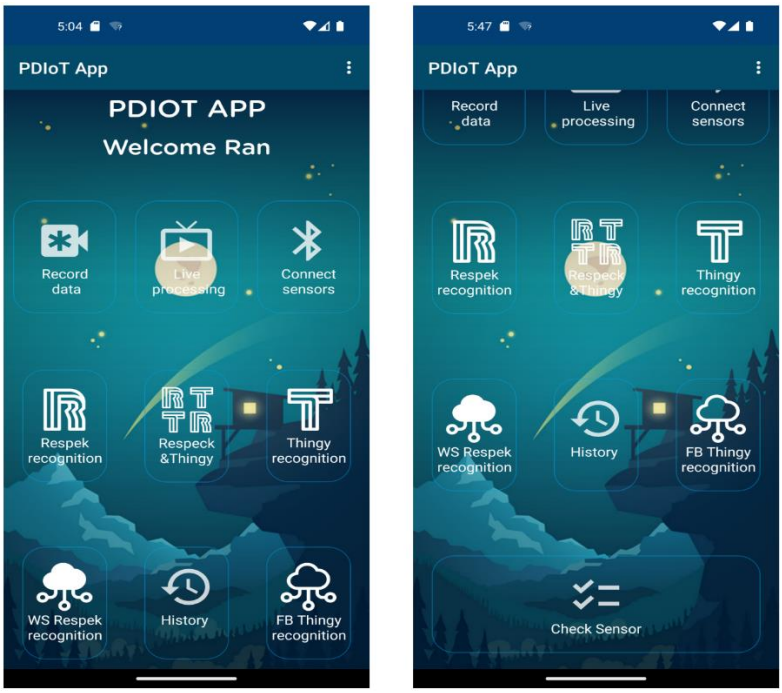


Figure 3.5.2: Home Menu page

3.5.3 Connect Sensor and Check Sensor

In fact, we kept the three basic functions in the first line, which are still available to users as they were in the original version provided. When the users first start the application, they will need to connect it to the Respeck and the Thingy. (The function to view historical data does not require connect to the sensor, which will be described later.) Click on the third button Connect sensors in the first row as is displayed in Figure 3.5.3(a) to connect. Make sure your phone's Bluetooth and sensors are on before connecting the sensors. Respeck should flash green light and Thingy should flash blue light when they are on. Like the original version, there are still three ways to connect:

1. If the user's phone supports NFC, NFC pairing can be used to connect.
2. Users can connect by scanning the QR code on the sensor.
3. Users can connect sensors by manually entering the ID on the device.

The correct way to wear the two sensors can be found in the previous 3.2 hardware section.

If the sensor is correctly connected and worn, the user can check that the sensor is being worn in the correct position by clicking on the button Check Sensor in the Figure 3.5.2, and the Figure 3.5.3(b) will be displayed. Once the sensor has been placed in the correct position, stand up straight and click START in Step one to start checking. As shown in the figure, when the detection is in process, it will show Checking.... Once the detection is complete, if it is worn in the correct position then Passed will be displayed and the user can proceed to the second step of detection. If Failed is displayed then the user will need to adjust the position of the Respeck

and then check again. After completing the first step of the test, user should lie down flat for the second step of the check. Similarly, Passed will be displayed when the test is passed. Also, if the test is not passed, Failed will be displayed and Step one and two should be done again after adjusting the position of the Respeck.

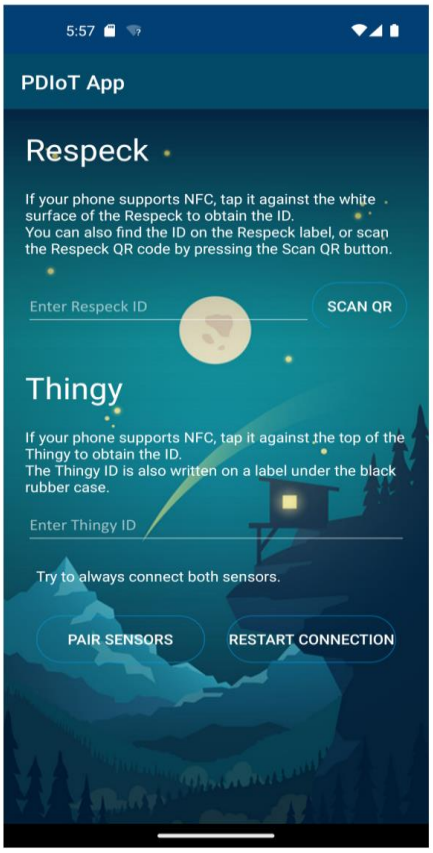


Figure 3.5.3(a): Connect sensor page

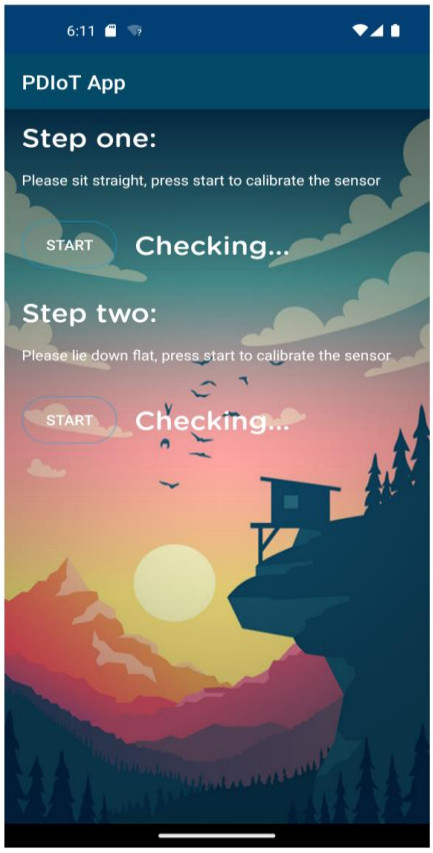


Figure 3.5.3(b): Check sensor page

3.5.4 Live processing & Record data

If both sensors are connected correctly, the sensors' live data are visible in the "Live processing" feature. Two real-time graphs of the accelerometer data from the Respeck (top) and Thingy are displayed (bottom) as shown in Figure 3.5.4(a).

The recording data part is the same as before, the user can record the sensor data through this part. Figure 3.5.4(b) shows the page for this feature. Specific instructions for using this feature can be found in https://github.com/specknet/pdiot-practical/blob/master/Labs/Week%201%20Lab.md part 8.

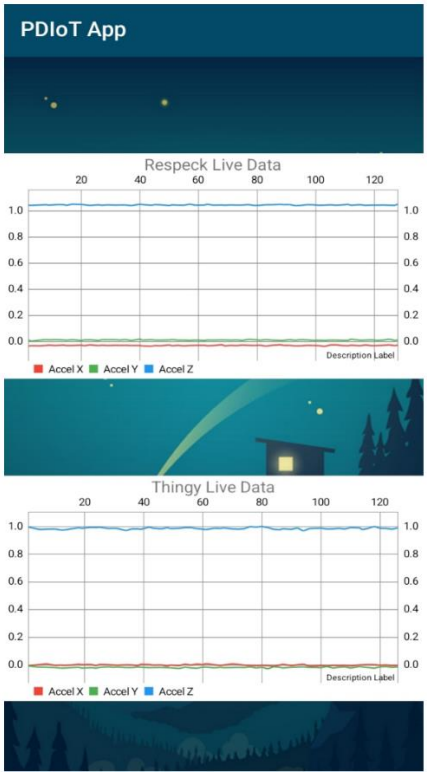


Figure 3.5.4(a): Live processing page

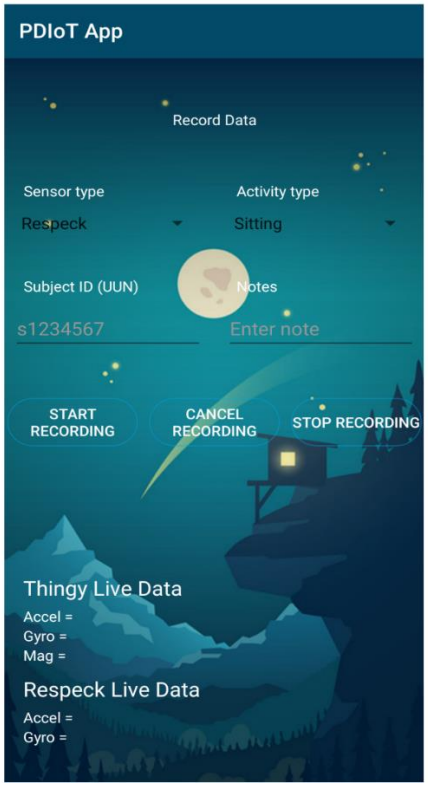


Figure 3.5.4(b): Record data page

3.5.5 Activity recognitions

In our program, the user can choose to use either Respeck or Thingy alone, or use a combination of both for activity prediction, and there are the three buttons from left to right in the second row of the main menu page shown in Figure 3.5.2. As it takes time to read the data and calculation, there is usually a 2-seconds delay for all three functions. When the recognition function is activated, the system will also detect how long the user spent sitting. If the user remains sitting for a long time, a reminder will pop up and suggest the user to take a break.

Respeck Recognition:

By clicking on the first button in the second row in the main menu page, the user will be able to view the results predicted by Respeck's data. In Figure 3.5.5(a), the top part is the predicted result(activity with the highest likelihood of occurrence), and the outer progress circle represents the probability of the predicted activity, with the whole circle being blue when the probability is 100%, and three-quarters of the circle being blue when the probability is 75%. The bottom half shows the live processing, which is the same image as in the living processing function.

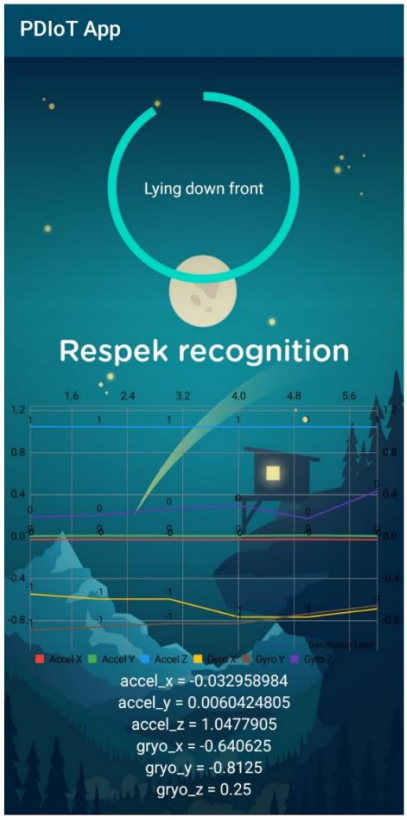
Thingy Recognition:

For Thingy, we have used the same design. By clicking on the first button in the second row in the main menu page, the user will be able to view the results predicted by Thingy's data.

However, in this project, using Thingy alone is not effective for most activity recognition, so we do not recommend using this feature alone. Thingy is more used in conjunction with Respeck to improve the accuracy of recognition. The interface of this feature is almost the same as that of Respeck recognition in Figure 3.5.5(a), so we will not show the figure of this page separately here.

Respeck Thingy Recognition:

By clicking the middle button Respeck & Thingy in the main menu page, users can get a prediction result which combines the result of the two sensors. Actually, this result will be more reliable than that of an individual sensor. Similarly, the activities with the highest likelihood of occurrence are given inside the progress circles, along with the possibility represented by the progress circles. The following sections separately give the likelihood of each other activity which may occur, with the activity with a higher possibility occurring in the upper section. The page for this mode is shown in the Figure 3.5.5(b).



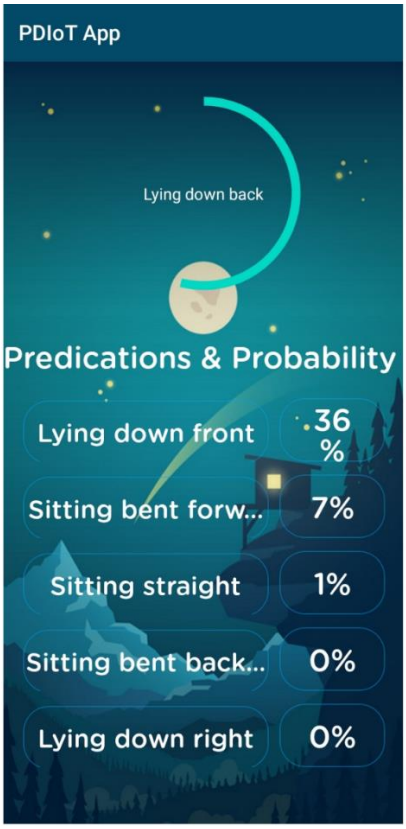


Figure 3.5.5(a): Page for R recognition Figure3.5.5(b): Page for R&T recognition

3.5.6 Cloud mode: WS Respeck recognition & FB Thingy recognition

The first and the third buttons in the third row of Figure 3.5.2 are WS Respeck recognition and FB Thingy recognition. Optionally, users can choose to use the cloud mode to get the activity predication by using two sensors individually.

When the app is installed for the first time and the device is connected to WI-FI, clicking on

the button WS Respeck recognition or FB Thingy recognition will automatically download the latest model, and when the model is downloaded, this function will work like normal mode.(UI are same as Figure 3.5.5(a)) and the advantage is that if there is a change in the model, it can be update in our application easily.

However, using this model relies on the network, and when the network breaks, the prediction stops and resumes when the network is reconnected again.

3.5.7 Historical data & Step counter

The button in the middle of the third row in Figure 3.5.2 can be used to view the history of activity, where the number of steps is also displayed.

Select the date and the time for each activity during that day will be counted in the bottom half of the bar chart, with the horizontal axis representing the activity name and the vertical axis showing the hours counted.

The Step in the bottom shows the number of steps of the user, this number will be added up when running and walking as well as climbing stairs and descending stairs are detected. The following Figure 3.5.7 shows the page for these features.

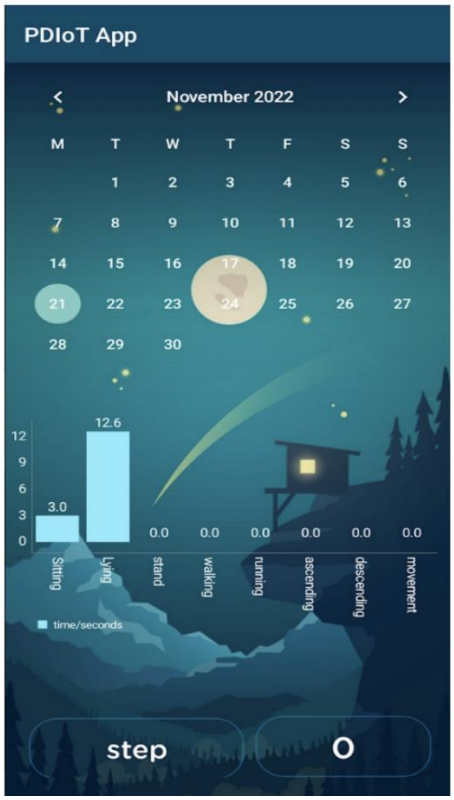


Figure 3.5.7: Page for Historical data & Step counter

3.6 Software organisation

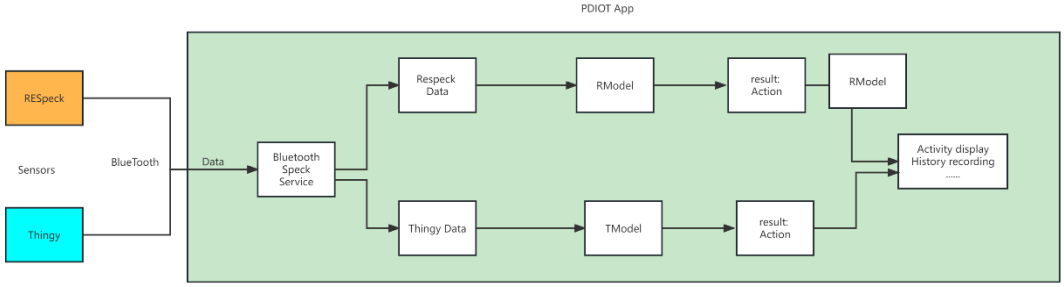
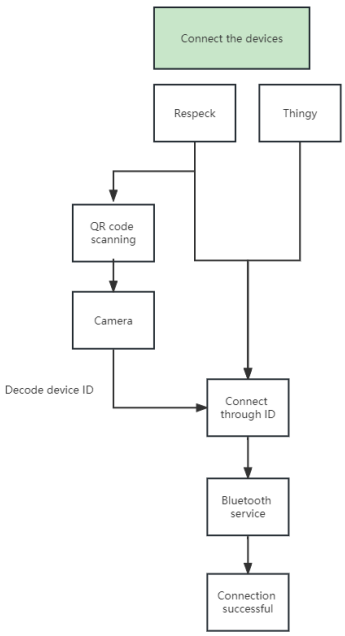


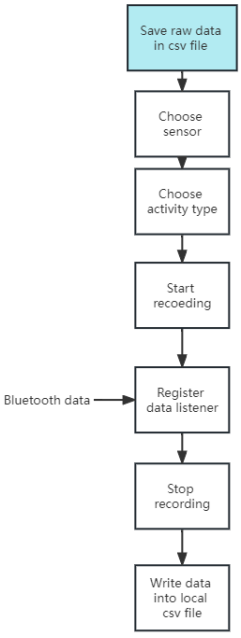
Figure 3.6(a): Organisation of the software APP

The figure above shows the overall software organization of the app. Next, we will split it into 9 small sections to introduce the composition in detail.

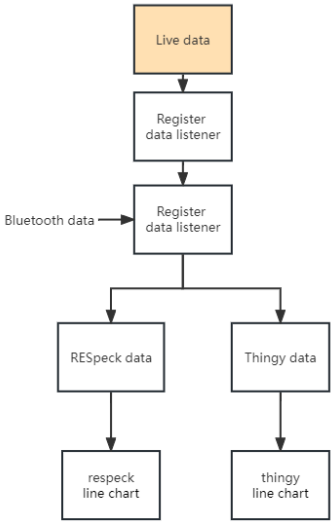
3.6.1 Sensor connection



3.6.2 Saving raw data



3.6.3 Live data display



For the connection function, saving raw data function and the live data display function, they are the same as the original PDIoT application. They are kept helping to connect the sensors and confirm the connection. It can also collect new data for training models.

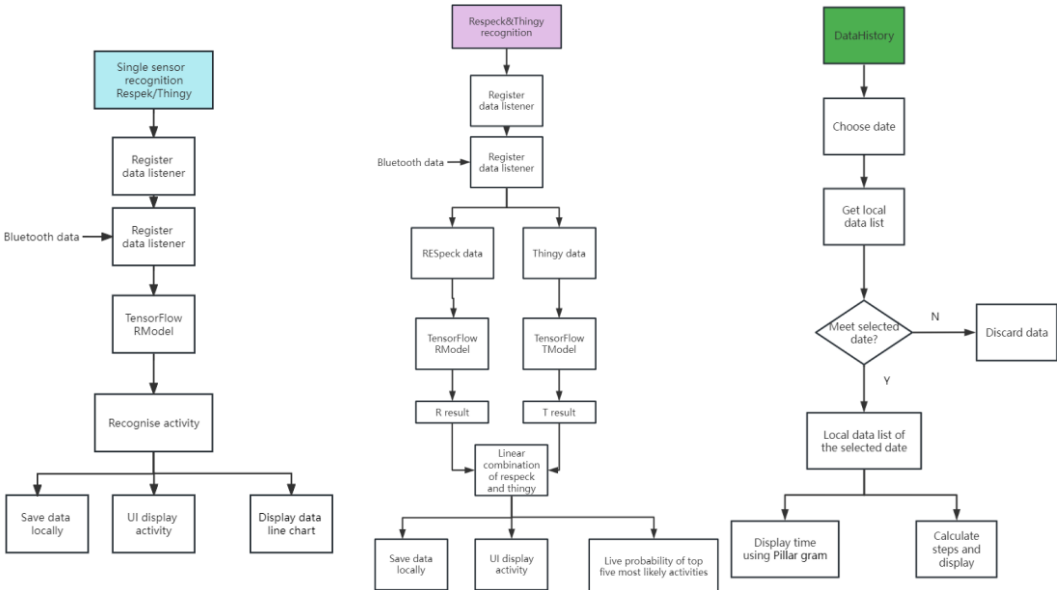
Before users start to use the functionalities of the app, they can register and login to protect their privacy.

The code for login and register uses the AppCompatActivity class and overrides the onCreate() method.

The layout for this activity is set in the setContentView method. The activity has an ImageView and TextView element, the ImageView has a touch listener that changes the image when the user swipes left or right. The activity also has EditText fields for email and password, and two buttons, one for signing up and one for logging in. The activity also uses the SharedPreferences class to check if the entered email and password match with the registered email and password of any user. If a match is found, the user is logged in, and if not, a toast message is displayed telling the user to enter valid email and password.

For register, it initializes the EditText and Button fields. The class uses the onClickListener to listen to the button clicks. When the user clicks on the register button, it calls the registerUserInfo() method which validates the user input and saves the user information in the shared preferences if the validation is successful. The class uses the Gson library to convert the objects to json strings and save them in the shared preferences. The class also uses regular expressions to check the email input by the user. The class contains methods like checkEmail() which validates the email input by the user and saveUserInfo() which saves the user information in the shared preferences.

3.6.4 Single sensor recognition 3.6.5 Respeck&Thingy recognition 3.6.6 Data history



After the Bluetooth data is read into the app, if the Respeck recognition or Thingy recognition is activated, the data will run through the model and generate the result. Then the app will display the detected activity and display the live raw data; a ring is added to show the probability of the current activity. It will also save the result locally.

The function of utilising both sensors can improve the outcome. In this function, data will go through both the Respeck model and Thingy model and generate two probability arrays. The final result combines the results of the two sensors linearly and adjusting the weight based on performance of each sensor, for example, Respeck performs better on the lying down activities and Thingy performs better on distinguishing standing and sitting straight.

The Respeck or Thingy Recognition Activity extends the AppCompatActivity class. This class creates a queue to update the graph smoothly and initializes elements in the UI such as a progress bar and text view for displaying predictions. The class can load a TFLite machine learning model and a list of labels for the 14 possible activity classifications. The class uses them to make predictions on the activity being performed. The class also listens for data updates from a RecognitionDataManager and uses that data to update the graph and make predictions on the activity being performed. The code also includes a timer that periodically updates the predicted activity and keeps track of the number of times each activity is recognized. The code also has a list of labels for the 14 possible activity classifications.

For Respeck and Thingy recognition, a slightly different UI is used, the top 5 probable activities and their probabilities will be displayed.

The app can remind user not to sit for too long. After detecting sitting or deskwork continuously for a long time, a reminder will pop up to remind the user to take a rest.

In order to improve accuracy furthermore, a state machine is used to prevent sudden changes, such as from lying flat to running.

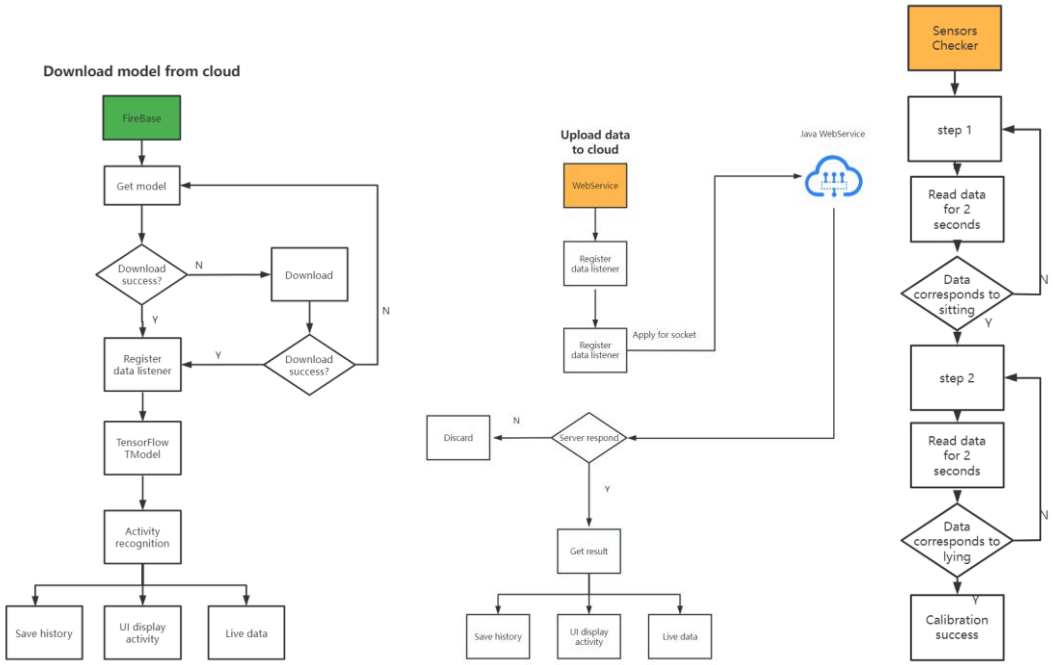
We used a local data structure to store historical data, which includes variables such as activity type and time stamp. When reading historical data, the app first locates the time stamp based on the selected date and then displays the history data in the form of a bar chart.

The class for displaying historical data extends the AppCompatActivity class and overrides the onCreate() method. Inside the onCreate method, the class utilizes the CalendarView and BarChart libraries to allow the user to select a date and display the data of that date. It calls several methods such as initCalenderView(), initBarChartStyle(), and initBarChartData(startTime). The initBarChartStyle method is responsible for initializing the x-axis of the bar chart, setting the text size, text color, and rotation angle of the labels. It also sets the values of the x-axis to be displayed as the labels, for a better user understanding. The initBarChartData method filters the data to be displayed on the bar chart based on the selected date, it's a crucial method for the functionality of the historical data view.

Overall, the class allows the user to interact with the historical data stored in the local data structure, it provides a user-friendly interface to select a date, and it displays the data of that

date in the form of a bar chart.

3.6.7 Download model from cloud 3.6.8 Upload data to cloud 3.6.9 Sensor check



We used 2 ways to realize the cloud service of the HAR app.

Firstly, we used Firebase's machine learning function. Firebase allow user to store the ML model online and when used, the app will download the model from google's Firebase cloud. After downloading the model to local storage, the app perform classification using the model. This feature enables us to update models easily.

Moreover, a Java is created to store the ML model. The class also creates a Retrofit object to connect to a server to obtain prediction results through LAN and send raw data to the server. The server will run machine learning on cloud and then send the classification result to the app through LAN. The app and server are connected using IPV4 address.

The CheckSensorsActivity is a feature of the app that allows users to calibrate the sensor to their own body. This is done by performing two simple tasks: sitting and lying down and examining the data readings each time. The app will then confirm if the device is worn in the correct position. The two buttons, "Step One Start" and "Step Two Start", check the sensor position, and provide feedback to the user. The RecognitionDataManager class is used to receive live data from the sensors, and calculations are performed on the data to determine the sensor position. The results of the check are displayed on TextViews, stepOneResult and stepTwoResult, which will indicate if the sensor is in the correct position, or if adjustments need to be made. The RESpeckLiveData class is also used to store the sensor data during the check.

3.7 Testing

The following sections outline the testing procedures for the PDIoT app, which includes the major functionality of the mobile application.

Test Objective:

To ensure the functionality of the "RespeckAndThingyRecognitionActivity" class in the PDIoT app, including the prediction of human activity, the display of prediction results, and the correct implementation of machine learning models.

Test Scope:

- The testing will focus on the RespeckAndThingyRecognitionActivity and its related classes and methods.
- The testing will include functional testing, performance testing, and usability testing.

Test Environment:

- The testing will be conducted on a device running Android 9 or higher.
- The app will be tested with both Respeck and Thingy sensors connected to the device.

Test Cases:

1 Verify that the prediction progress bar and prediction text are displayed correctly.

2 Verify that the TFLiteModel class is correctly implemented and able to load the machine learning model files.

3 Verify that the prediction results are displayed correctly and match the expected output.

- Input: Perform a set of physical activities while connected to both Respeck and Thingy sensors.
- Expected Output: The prediction results displayed on the activity should match the expected output for the performed activities, with a high degree of accuracy.

4 Verify that the prediction results are updated in real-time as new data is received from the sensors.

- Input: Perform a set of physical activities while connected to both Respeck and Thingy sensors.
- Expected Output: The prediction results displayed on the activity should update in real-time as new data is received from the sensors.

5 Verify that the app does not crash or cause any unexpected behaviour during usage.

- Input: Use the app for a period of time and perform various activities
- Expected Output: The app should not crash or cause any unexpected behaviour during usage.

Acceptance Criteria:

- All test cases should pass with no errors or issues.
- The prediction should be accurate.
- The app should be easy to use and have a good usability.

- The app should not crash or cause any unexpected behaviour during usage.

Through passing all the above tests, we found that our APP successfully met all the criteria. This shows that all the functions we want to achieve have been completed correctly and the app is ready for users to use without significant bugs and errors.

4. Results

4.1 Critical analysis of the implementation using quantitative methods for accuracy of the classification both offline and in real-time

Offline accuracy:

For the offline accuracy check, first we split the overall data into two parts, 80% and 20%, when training the model. 80% of the data is used for training and the remaining 20% is used to verify the training effect. According to the verification results of the 20% data, we continuously improved and optimized the model, and finally determined the structure and parameters of the model based on the optimal results. The validation accuracy line chart of the final model is shown in Figure 4.1(a) below.

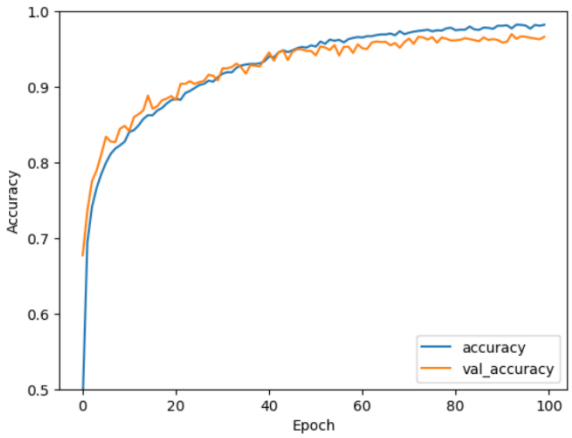


Figure 4.1(a): Line chart of validation accuracy for the final model

In our project, we have a total of three models, they are Respeck model, Thingy model and a hybrid model of the two sensors. For each model, we use 20% of the test data to test the accuracy of each activity and organize them in the table and a confusion matrix.

First of all, for the Respeck model, our test results are shown in Figure 4.1(b) below. It can be seen that for a single Respeck model, our overall accuracy rate can reach 97%.

		precision	recall	f1-score	support
Sitting	0	0.92	0.95	0.94	259
Sitting bent forward	1	1.00	0.98	0.99	261
Sitting bent backward	2	0.99	0.98	0.98	281
Standing	3	0.96	0.94	0.95	273
Lying down left	4	1.00	1.00	1.00	269
Lying down right	5	1.00	0.99	1.00	259
Lying down on stomach	6	0.99	1.00	1.00	287
Lying down on back	7	1.00	0.99	0.99	278
Walking	8	0.97	0.93	0.95	272
Running	9	1.00	1.00	1.00	258
Climbing stairs	10	0.90	0.96	0.93	216
Descending stairs	11	0.92	0.97	0.94	234
Desk work	12	0.94	0.93	0.94	273
Movement	13	0.91	0.87	0.89	256
accuracy				0.97	3676
macro avg		0.97	0.97	0.97	3676
weighted avg		0.97	0.97	0.97	3676

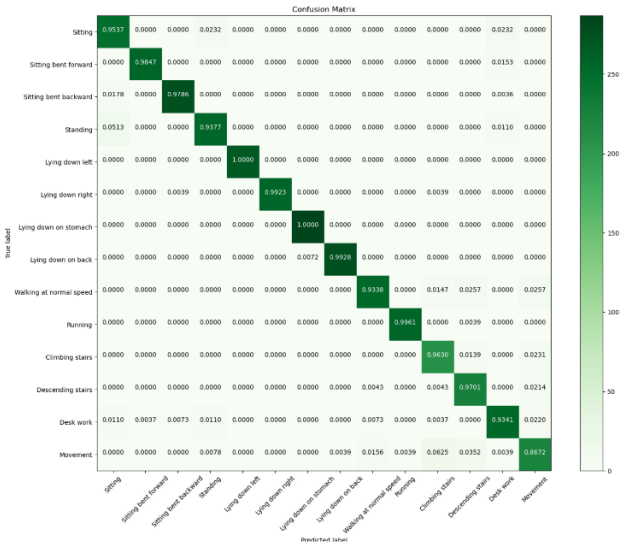


Figure 4.1(b): Accuracy table and Confusion Matrix for Respeck model on 20% test data

As shown in the results in the figure above, we can see that for the results of the Respeck model alone, most of the activities perform well. However, the recognition effects for sitting straight, standing straight, walking, desk work and general movement are average, the accuracy rates for these activities are under 0.95 while others are nearly 1.00, and such results are predictable. This is because the wearing position of Respeck is on the upper body, and the difference between these actions and some other actions is mainly in the movement state of the lower body, so it is difficult to distinguish these activities only with the Respeck sensor, which leads to some inaccurate results.

Next, we will turn to the Thingy model. Figure 4.1(c) below shows the accuracy table and Confusion Matrix of the Thingy model under 20% data. For the Thingy model alone, the overall accuracy is around 93%.

		precision	recall	f1-score	support
Sitting	0	0.81	0.94	0.87	268
Sitting bent forward	1	0.87	0.90	0.89	269
Sitting bent backward	2	0.91	0.78	0.84	259
Standing	3	1.00	0.99	1.00	276
Lying down left	4	0.96	0.99	0.97	250
Lying down right	5	0.96	0.99	0.97	278
Lying down on stomach	6	0.99	0.92	0.95	265
Lying down on back	7	0.95	0.95	0.95	274
Walking	8	0.97	0.97	0.97	257
Running	9	1.00	0.98	0.99	258
Climbing stairs	10	0.95	0.97	0.96	229
Descending stairs	11	0.85	0.93	0.89	217
Desk work	12	0.91	0.88	0.90	257
Movement	13	0.91	0.85	0.88	288
accuracy				0.93	3645
macro avg		0.93	0.93	0.93	3645
weighted avg		0.93	0.93	0.93	3645

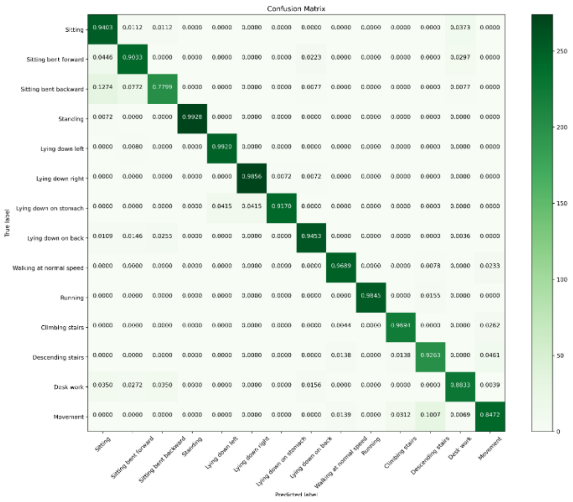


Figure 4.1(c): Accuracy table and Confusion Matrix for Thingy model on 20% test data

From the above figure, we can observe that for the Thingy model alone, the overall accuracy rate is not as good as the Respeck model, and the performance of using only Thingy in the recognition of many activities is not good or even poor. However, we can observe that when using Thingy, the recognition of some actions such as Standing and Walking performs very well, and far exceeds the accuracy when using Respeck. This is because the Thingy sensor is worn in the trouser pocket, which makes it more sensitive to the movements of the lower body, so that these activities can be distinguished by the difference in the movements of the lower body. Based on such results, it is not difficult to find that the results of these two sensors are complementary. If we combine the recognition results of these two sensors, we should be able to make our recognition results more accurate. Under this premise, we use the third model, which uses the data of Respeck and Thingy sensors at the same time, and it can synthesize the recognition results of Respeck and Thingy to make the final recognition judgment. It is worth noting that for some actions, the recognition accuracy of Thingy is too low, which will cause the recognition result of adding Thingy to reduce the accuracy. Therefore, we will adjust the proportion of Respeck results and Thingy results in each activity according to the observation of the accuracy rate. In the action where the Thingy accuracy rate is too low, we will not even use Thingy recognition at all to maximize the accuracy rate. In the same way, in some activity recognition, we will also perform such operations on the recognition results of Respeck. After constant adjustments, the final accuracy and confusion matrix results of this hybrid model are shown in Figure 4.1(d) below.

****
Classification report
****

		precision	recall	f1-score	support
Sitting	0	0.97	0.99	0.98	1339
Sitting bent forward	1	1.00	1.00	1.00	1341
Sitting bent backward	2	0.99	0.99	0.99	1340
Standing	3	0.99	0.97	0.98	1345
Lying down left	4	1.00	1.00	1.00	1350
Lying down right	5	1.00	1.00	1.00	1349
Lying down on stomach	6	1.00	1.00	1.00	1357
Lying down on back	7	1.00	1.00	1.00	1356
Walking	8	0.99	0.98	0.99	1321
Running	9	1.00	1.00	1.00	1318
Climbing stairs	10	0.98	0.99	0.98	1118
Descending stairs	11	0.98	0.99	0.99	1149
Desk work	12	0.99	0.98	0.98	1347
Movement	13	0.98	0.97	0.98	1350
	accuracy			0.99	18380
	macro avg	0.99	0.99	0.99	18380
	weighted avg	0.99	0.99	0.99	18380

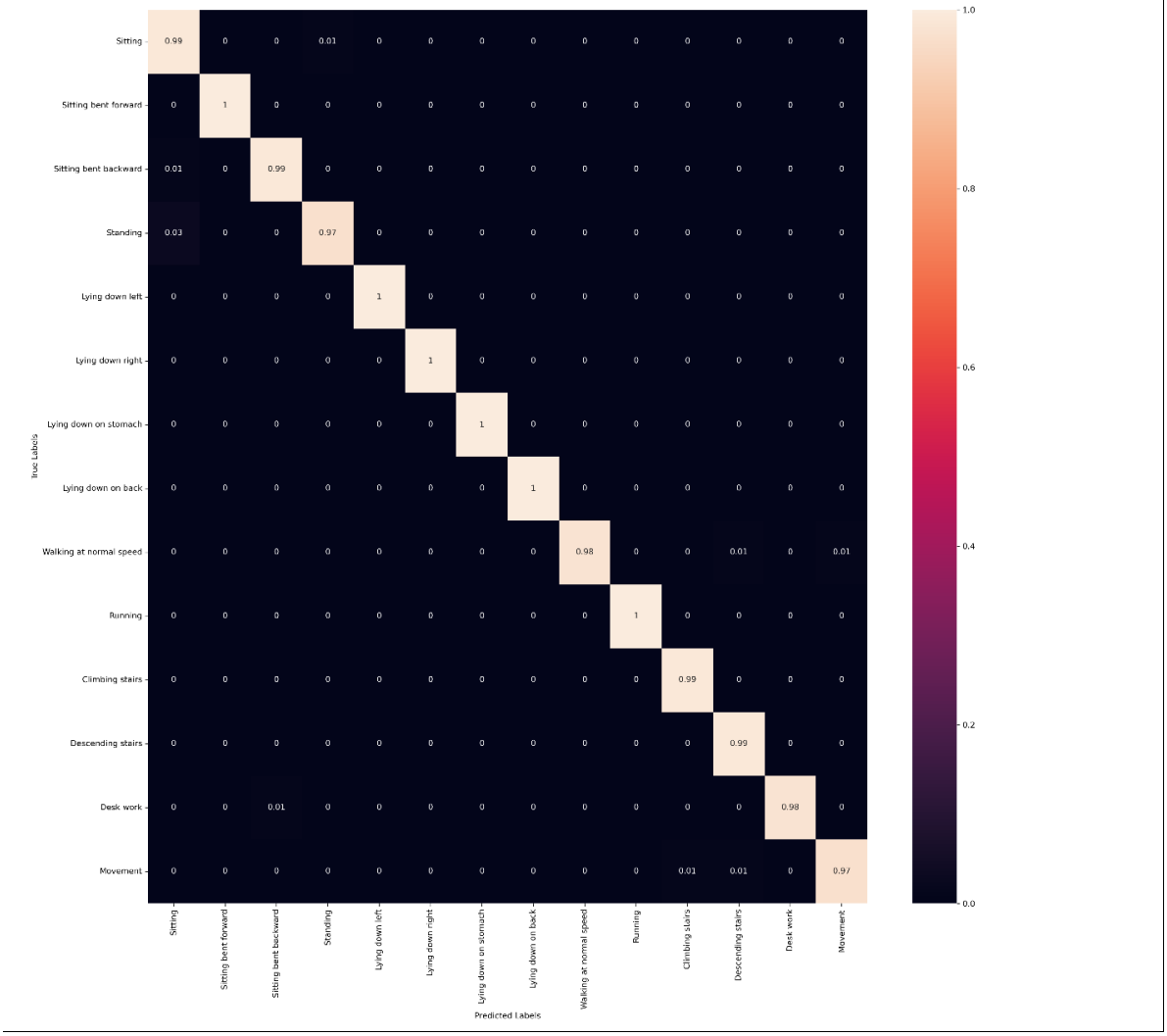


Figure 4.1(d): Accuracy table and Confusion Matrix for Respeck & Thingy model

From this result, we can see that after our continuous adjustment, this model has achieved a very high accuracy rate in the recognition of each activity. This result undoubtedly meets our needs and achieves our goals.

Real-time accuracy:

In the analysis of the accuracy rate of real-time APP recognition, since we know that Thingy cannot provide high accuracy in many activity recognition, we only select some activities with high accuracy for Thingy to test. In this test, our three team members will wear the sensor and use the APP to test these activities, and the results of the test will be summarized in the table. We still start with the single Respeck mode first.

Respeck

Activity Name	Try 1 - student A	Try 2 - student B	Try 3 - student C
Sitting	sitting straight	sitting straight	sitting straight
Standing	sitting straight	desk work	sitting straight
Walking	Walking	general movement	general movement
Running	running	running	running
Lying down on the back	Lying down on the back	Lying down on the back	Lying down on the back
Lying down on front	Lying down on front	Lying down on front	Lying down on front
Lying down on right	Lying down on right	Lying down on right	lying down on the right
Lying down on left	Lying down on left	Lying down on left	Lying down on left
desk work	sitting straight	desk work	desk work
sitting bent forward	sitting bent forward	standing	sitting bent forward
sitting bent	sitting straight	sitting bent	sitting bent
backwards		backward	backward
movement	general movement	general movement	general movement
ascending stairs	general movement	ascending stairs	ascending stairs
descending stairs	descending stairs	descending stairs	descending stairs

Figure 4.1(e): Real time recognition for Respeck mode

Figure 4.1(e) shows the recognition results when only Respeck is used. From this result, we can see that the recognition results of most activities are very accurate. For the three activities of desk work, sitting bent forward, sitting bent backward and ascending stairs, the recognition results are correct in most cases, but there are occasional cases where actions are recognized as other actions. As for the two activities of standing and walking, these three activities will often be identified as other activities when only Respeck is used.

With the known test results of offline mode, such results are in line with expectations, so for the recognition of Thingy, we only do the less accurate part of Respeck to test whether the recognition results of Thingy can help Respeck recognition. The real time test results for Thingy only will be shown in the following Figure 4.1(f).

Thingy

Activity Name	Try 1-Student A	Try 2-Student B	Try 3-Student C
Standing	Standing	Standing	Standing
Walking	Walking	General Movement	Walking
Desk work	Sitting Straight	Desk work	Desk work
Ascending stairs	Ascending stairs	Ascending stairs	General Movement
Sitting bent forward	Desk work	Sitting Straight	Sitting bent forward
Sitting bent backward	Sitting bent forward	Sitting bent backward	Sitting Straight

Figure 4.1(f): Real time recognition for Thingy mode

From this result, we can see that Thingy's recognition results for standing, which has the lowest accuracy of Respeck are very accurate. In addition, it can also improve the recognition results of Respeck for the three activities of Walking, Desk work and Ascending stairs. For Sitting bent forward and backward, Thingy's results are not very good, so these two activities should still mostly rely on Respeck's recognition results.

Based on the real-time test results of the first two modes, we continuously adjusted the proportion of Respeck and Thingy results for each activity in the third hybrid model, and finally we determined these proportions based on the actual measurement results. The Figure 4.1(g) below shows the final test results in this hybrid mode.

Respeck & Thingy

Activity Name	Try 1-Student A	Try 2-Student B	Try 3-Student C
Sitting straight	Sitting straight	Sitting straight	Sitting straight
Running	Running	Running	Running
Lying down on back	Lying down on back	Lying down on back	Lying down on back
Lying down on stomach	Lying down on stomach	Lying down on stomach	Lying down on stomach
Lying down on right	Lying down on right	Lying down on right	Lying down on right
Lying down on left	Lying down on left	Lying down on left	Lying down on left
Standing	Standing	Standing	Standing
Walking	General Movement	Walking	Walking
Desk work	Desk work	Desk work	Sitting bent forward
Ascending stairs	Ascending stairs	Walking	Ascending stairs
Descending stairs	Descending stairs	Descending stairs	Descending stairs
Sitting bent forward	Sitting bent forward	Desk work	Sitting bent forward
Sitting bent backward	Sitting bent backward	Sitting bent backward	Sitting bent backward
General Movement	General Movement	General Movement	General Movement

Figure 4.1(g): Real time recognition for Respeck & Thingy mode

Through the test of this result, we can find that after combining the results of Respeck and Thingy, the recognition accuracy is greatly improved, and this mode is also the one with the highest recognition accuracy among the three modes. The recognition of most activities is completely accurate, only in the 4 activities Walking, Desk work, Ascending stairs and Sitting

bent forward, it may still occasionally be recognized as other similar actions. This is because neither Respeck nor Thingy can fully achieve accurate recognition of these actions, so no matter how we adjust the respective proportions, we cannot prevent occasional recognition errors. But we can also find that the probability of these recognition errors is very low, and the recognition is correct most of the time, so this is a completely acceptable result. After achieving this result, we can say that our app has achieved its design goals.

4.2 Benchmarking of your implementation in terms of processing cycles, memory usage, power consumption, latency in generating results

The performance of the PDIoT app was evaluated using Android Studio's Android Profiler. The app was found to occupy 32.36 MB of memory for storage. However, it should be noted that additional storage may be required if the user begins to use the app and generate data that needs to be stored locally. Additionally, the app's cache may consume some additional storage space. The app utilised an average of 29.5 MB per hour in terms of RAM usage. This indicates that the app is a small application and does not consume a significant amount of CPU power or energy. The app was found to have a CPU usage of around 10% while running.
As for the other performance benchmarks, the predicting latency is 2 seconds, which is determined by the ML algorithm. The system's scalability should be relatively good, as the system uses machine learning which can handle a large amount of data. The sensitivity and specificity of the system should be high as the system uses IMU sensors which are specifically designed for movement recognition.

5. Conclusions

5.1 Reflection on the project

To sum up, in the process of this project, through joint efforts and cooperation in machine learning, mobile application interface and software, we finally successfully completed all the functions and realized a complex IoT system for human activity recognition. After completing the project, we felt a sense of accomplishment in having completed a challenging task. We felt proud of the working system we have built and have gained a deeper understanding of the technical and practical aspects of IoT. We also learned about the importance of collaboration and teamwork in a group project.
At the same time, in the process of reviewing this project, we also concluded some deficiencies and some aspects that we could have done better. For example, at the beginning of the task, our

division of work is not very clear. At the very beginning, we hope that everyone can participate in all three aspects of the task so that everyone can get the knowledge and experience of all three aspects. But when we actually operated on it, we found that if we want three people to do the same task, we need more communication to really improve the efficiency. Otherwise, due to the different content of learning and our own thinking, it is difficult for the three people to combine the things they made. This caused us to be inefficient in the first few weeks and had to solve some additional problems due to the differences in our respective works. After discovering this situation, we made corrections on work division in time and invested more time in the following weeks to make up for the previous mistake.

5.2 Possible extension of the project and the improve of the implementation

Based on the time constraints of our courses and our current technical level, our APP still has some parts that can be improved and some functions that can be extended in the future.

Possible extensions:
Multi-person recognition: We can work on the ability to recognize multiple people at the same time, and have the system distinguish between them.
Multi-language support: This function can add multi-language support to the system, so that it can be used by people speaking different languages.
Social interaction: This adds the functionality that allows users to share their activity data with friends and family and compete with them in activities.
Virtual reality integration: We can explore the possibility of integrating the human activity recognition system with virtual reality technology, so that users can experience their activities in a more immersive way.
Personalized training plan: This is aiming at adding the functionality that allows the system to provide personalized training plans for the user, based on their activity data and goals.
Voice control: We can work on adding voice control functionality to the system, so that users can control it hands-free.
Integration with smart home: We can explore the possibility of integrating the human activity recognition system with smart home technology, so that the system can automatically adjust settings such as lighting and temperature based on the user's activity.
Possible Improvement:
The step counting algorithm can be optimised: currently the steps are counted by calculating how long the user spent on running, walking and doing general movement. To do a better job, potential improvement could be using peak detection (detects the peaks in the data, which correspond to steps taken) or zero-crossing (detects the zero-crossings in the accelerometer data) algorithm.

6. References:

[1] Gupta, N., Gupta, S.K., Pathak, R.K. et al. Human activity recognition in artificial intelligence framework: a narrative review. Artif Intell Rev 55, 4755–4808, 2022.

[2]Fatemeh Serpush, Mohammad Bagher Menhaj, Behrooz Masoumi, Babak Karasfi, "Wearable Sensor-Based Human Activity Recognition in the Smart Healthcare System", Computational Intelligence and Neuroscience, vol. 2022, Article ID 1391906, 31 pages, 2022.

[3]Abdelghani Dahou, Mohammed A.A. Al-qaness, Mohamed Abd Elaziz, Ahmed Helmi, Human activity recognition in IoHT applications using Arithmetic Optimization Algorithm and deep learning, Measurement, Volume 199, 2022.

[4] Y. Chen and Y. Xue, "A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer," 2015 IEEE International Conference on Systems, Man, and Cybernetics, Hong Kong, China, 2015, pp. 1488-1492.

[5] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, Hyeonjoon Moon, Sensor-based and vision-based human activity recognition: A comprehensive survey, Pattern Recognition, Volume 108, 2020.

[6] M.S. Seyfioğlu, A.M. Özbayoğlu, S.Z. Gürbüz
Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities, IEEE Trans. Aerosp. Electron. Syst., 54 (4) (2018), pp. 1709-1723

[7] T.N. Nguyen, S. Lee, H. Nguyen-Xuan, J. Lee, A novel analysis-prediction approach for geometrically nonlinear problems using group method of data handling, Comput. Methods Appl. Mech. Eng., 354 (2019), pp. 506-526

[8] E.P. Ijjina, K.M. Chalavadi, Human action recognition in RGB-D videos using motion sequence information and deep learning, Pattern Recognit., 72 (2017), pp. 504-516

[9] Md. Milon Islam, Sheikh Nooruddin, Fakhri Karray, Ghulam Muhammad,
Human activity recognition using tools of convolutional neural networks: A state of the art review, data sets, challenges, and future prospects, Computers in Biology and Medicine, Volume 149, 2022.

[11] E. Ramanujam, T. Perumal and S. Padmavathi, "Human Activity Recognition With

Smartphone and Wearable Sensors Using Deep Learning Techniques: A Review," in IEEE Sensors Journal, vol. 21, no. 12, pp. 13029-13040, 2021.

[12] J. Wang, Y. Chen, S. Hao, X. Peng and L. Hu, "Deep learning for sensor-based activity recognition: A survey", Pattern Recognit. Lett., vol. 119, pp. 3-11, Mar. 2019.

[13] Yeon-Wook Kim, Kyung-Lim Joa, Han-Young Jeong, Sangmin Lee, "Wearable IMU-Based Human Activity Recognition Algorithm for Clinical Balance Assessment Using 1D-CNN and GRU Ensemble Model", Sensors, vol.21, no.22, pp.7628, 2021.

[14] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks", Appl. Soft Comput., vol. 62, pp. 915-922, Jan. 2018.

[15] X. Cheng, L. Zhang, Y. Tang, Y. Liu, H. Wu and J. He, "Real-Time Human Activity Recognition Using Conditionally Parametrized Convolutions on Mobile and Wearable Devices," in IEEE Sensors Journal, vol. 22, no. 6, pp. 5889-5901, 15 March15, 2022.

[16] S. Chung, J. Lim, K. J. Noh, G. Kim and H. Jeong, "Sensor data acquisition and multimodal sensor fusion for human activity recognition using deep learning", Sensors, vol. 19, no. 7, pp. 1716, Apr. 2019.

[17] J. Sun, Y. Fu, S. Li, J. He, C. Xu and L. Tan, "Sequential human activity recognition based on deep convolutional network and extreme learning machine using wearable sensors", J. Sensors, vol. 2018, Sep. 2018.

[18] Biswal, Ankita, et al. "Human Activity Recognition Using Machine Learning: A Review." Progress in Advanced Computing and Intelligent Engineering (2021): 323-333.