# School of Informatics


# PDIoT Coursework 3 (2021-22)

## Human Activity Recognition (HAR) Machine Learning Algorithm and Applications: Based on the Respeck and Android System

**Name:** Chenxi Yang, Pengcheng Jin, Yongjian Liu

**Matriculation Number:** 2119637, 2173036, 2171825

**Abstract:**

This paper presents an Android application that could implement Human Activity Recognition (HAR). The HAR algorithm based on the CNN model that could classify seven activities(Falling, Walking, Sitting, Running, Lying, Descending stairs and Climbing stairs). This model could get 94% accuracy in the training set and 86% at the test set. Based on the HAR results, this software achieves many functions, such as history data view, sedentary reminder and falling alert. In addition, the user could also customise their software, such as Max sitting time in the sedentary reminder and emergency contact phone number in the fall alert. Next, the application supports authentication that could access it at any device and stored data in the Cloud. Finally, this application has an intuitive user interface that could provide a user-friendly experience.

# Content

## 1. Introduction

Human physical activity recognition(HAR) is growing rapidly due to the cost-effectiveness and miniaturisation of wearable wireless sensors(such as gyroscopes and accelerometers). Many deep learning algorithms are applied in the HAR, such as Deep Autoencoder, CNN and LSTM. The application of deep learning makes long-term continuous HAR possible, such as ambient assisted living, sports injury detection, elderly care and rehabilitation[1]. In the meantime, HAR and the Internet of Things(IoT) are becoming increasingly intertwined due to gateways such as smartphones[2]. As the fog node, the smartphone could collect and process wearable wireless sensor data[3]. This allows HAR to be applied in large numbers population.

Therefore this paper will develop a CNN deep learning algorithm to recognise human activity and explore some HAR applications in the Android system smartphone based on the Respeck wearable wireless sensors. The CNN model was developed by Tensorflow and deployed in the Android smartphone by Tensorflow lite. The software was developed in Kotlin and Java language. In addition, there is a server hosted in the Google Firebase that could support the Authentication and data storage functions. Next, this application explores some HAR applications, such as Sedentary Reminder and Fall Alert. There are seven activities that could be recognised in this application:

- Falling
- Walking
- Sitting
- Running
- Lying
- Descending stairs
- Climbing stairs

Finally, by comparing the CNN, LSTM and Google Activity Recognise API, this paper proposed a CNN model that could classify the seven human activities mentioned above. This model could get 94% accuracy in the training set and 86% at the test set. In addition, there is an Android application that realize the functions of authenticate users, view history data, remind sedentary and alert falling.

The rest of the article is organised as follows: Section2 reviews human activity recognition algorithms. Section3 discusses the technology that is used in this application. Section4 provides the test results for the software and CNN model. Section5 compares the difference between our application and LSTM as well as Google Activity Recognise API and presents the weaknesses of this application. Finally, the future work and conclusion are presented in Section6.

## 2. Literature Survey

The HAR was proposed many years ago, and nowadays it has two main directions of development: based on Computer Vision and based on wearable sensors[4][5][6]. Computer vision recognises human activity by extracting and analysing keyframes from video recorded by cameras. It includes image location, object detection, object track and semantic segmentation technology. The application of machine learning has accelerated the development of The HAR computer vision, and there are many proven frameworks for it, such as Tensorflow[7] and Keras[8]. However, computer vision is very dependent on image resolution, ambient lighting and the camera angle. In addition, the magnitude of movement is difficult to capture due to the lack of 3D depth-of-field information. This limits the development of computer vision in HAR.

However, the wearable sensor could overcome these difficulties. Wearable sensors implement it by collecting and analysing wearable sensors data such as accelerometers and gyroscopes. It could reduce the interference caused by external environmental information. In addition, the dimensions of data from the sensors are significantly reduced. The difficulty of feature extraction is reduced significantly. Therefore, this section will focus on the review of the sensor-based HAR algorithm literature.

Since Sensor-based HAR was proposed earlier and artificial intelligence was not widespread in that period, many traditional classification algorithms were used, such as KNN, decision trees. Therefore, the Sensor-based HAR algorithms could be divided into two groups: The traditional and the Deep Learning methods. The traditional method includes KNN, Decision Tree, Naive Bayes, SVM and Logistic Regression. The deep learning method includes Convolutional Neural Network(CNN), Recurrent neural network (RNN), Deep Autoencoder, Sparse Coding and Restricted Boltzmann Machine(RBM)[1]. Deep Autoencoder, Sparse Coding and RBM belong to the generated model, which mainly focuses on the feature extraction and dimensionality reduction. The CNN and RNN belong to the discriminant model, implementing the classification tasks. Nowadays, the traditional method is rarely implemented in HAR due to its low classification accuracy in terms of complex movements, such as ascending stairs[9]. Therefore this paper will give a brief overview of traditional methods and a detailed overview of deep learning discriminant models.

### 2.1 Traditional Methods

KNN, Decision Tree, Naive Bayes(NB) and Logistic Regression could be applied in the HAR [9][10]. The traditional machine learning algorithm has been discussed in a lot of papers [11]. Therefore the basic technology of these algorithms would not be mentioned in this part. Wu et al. use KNN, Decision Tree (C4.5), Logistic and NB to recognise nine activities, which are slow walking, normal walking, brisk walking, jogging, sitting, normal upstairs, normal downstairs, brisk upstairs and brisk downstairs[9]. Most of the algorithm gets high accuracy in simple activities(nearly 90%), except the NB. However, all of the traditional methods have poor

accuracy in complex activities, such as upstairs and downstairs. Then, Weiss uses the Random Forest algorithm to recognise eighteen activities, which gets more than 90% overall accuracy[12].

### 2.2 Deep Learning Methods

#### 2.2.1 Generate Model

Restricted Boltzmann Machine is a generative deep learning model that can learn a probability distribution over its set of inputs[13]. Restricted Boltzmann Machine consists of visible and hidden units, which are confined to form a bipartite graph to realise effective feature extractor algorithm, but it is difficult to apply to smartphones because it produces too many meaningless hidden neurons in RBM[14].

The Deep Autoencoder could project unsupervised data to the low-dimensional space to produce the features [1][15]. By the multiple hidden layers, different features are found in Deep Autoencoder [15]. Sparse Coding could find the relationship between different input data by the dynamic representation of data as a linear combination of basis vectors[16].

The application of these three Generate Models in HAR could be found in Table1.

| Model | Name | Title |
|---|---|---|
| RBM | Erkan Karakus et al[13] | Conditional restricted Boltzmann machine as a generative model for body-worn sensor signals |
| | Jingjing Cao et al[17] | A Sensor-Based Human Activity Recognition System via Restricted Boltzmann Machine and Extended Space Forest |
| Deep Autoencoder | Serge Thomas et al[18] | Ensemble of Deep Autoencoder Classifiers for Activity Recognition Based on Sensor Modalities in Smart Homes |
| | Varamin et al[19] | Deep Auto-Set: A Deep Auto-Encoder-Set Network for Activity Recognition Using Wearables |
| Sparse Coding | Sourav Bhattacharya et al[20] | Using unlabeled data in a sparse-coding framework for human activity recognition |
| | Y. Zhu et al[21] | Sparse Coding on Local Spatial-Temporal Volumes for Human Action Recognition |

Table1: Generate Model Applications

#### 2.2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) performs convolution operations on raw data and has widespread applications in image classification, speech recognition, and mobile and wearable sensors based on human activity recognition[22]. Generally, the convolutional neural network model includes convolutional layer, pooling layer and fully connected layer[22]. The

convolutional layer could detect the local conjunctions of features from the previous layer, and the pooling layer could merge similar features into one[23]. Zebin compares traditional methods, CNN model has significant speed-up in computing and improvement in overall classification accuracy in HAR[24]. Huang presents a shallow HAR CNN that could remove redundant information among channels in multiple sensors fusion[25].

### 2.2.3 Recurrent Neural Network

The Recurrent neural network (RNN) was built to model sequential data, like time series or raw sensor data[1]. However, RNN could have vanished and exploded gradients due to the network using time backpropagation to train[26]. Long short-term memory (LSTM) could overcome it because it contains memory blocks with cells in the hidden layers, storing long-time memory information[27]. Gated recurrent units(GRU) is the variant of LSTM, it uses forget gates to significant reduce the parameters[28]. However, few works used RNN for the HAR due to the learning speed and resource consumption[29]. Most of the methods use hybrid to combine CNN and RNN to get better performance, such as CNN-LSTM[30][31] and CNN-GRU[32].

## 3. Methodology

### 3.1 System and Implementation

This application has three main components: Server, Software, and CNN model. The relationship between these three components is shown in the Figure1. Server and CNN model as the service supporter support the software operation. The server could authenticate and store user data. The CNN model could output the activity result from the sensor data. The software supports three main functions: authenticate, recognise the human activity, and show historical data. In addition, this software also supports some applications about using the recognised result, such as Fall Alert and Sedentary Reminder. The implementation of these components will be explained below.

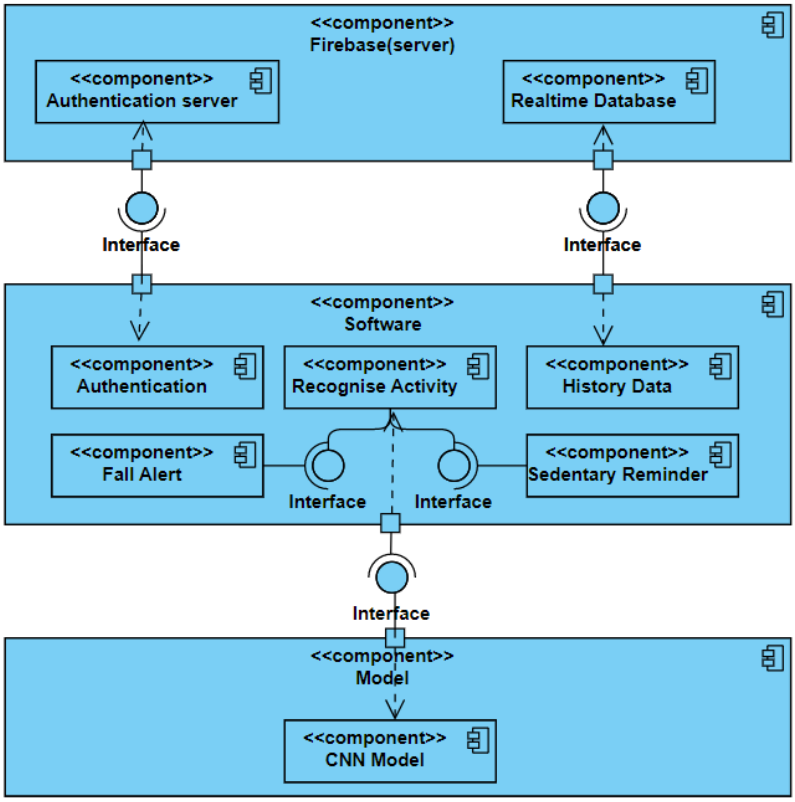



Figure1: System Components Diagram

### 3.2 Hardware and Firmware

This application is based on the Respeck wearable wireless sensor and the Android system. The fundamental requirements of hardware and firmware will be shown in Table2.

| Requirements | Details |
|---|---|

| Respeck | Null |
|---|---|
| SmartPhone | Min CPU: 1GHZ, 2 Cores |
| | Min RAM: 2G |
| | Min ROM: 4G |
| | Camera |
| | Bluetooth |
| | WIFI or Cellular data |
| | Registered to the Network (SIM card) |
| | Min Sdk Version: 26 (Android Oreo or OneUI 1.0 or EMUI 8.0 or MIUI 12) |

Table2: Hardware and Firmware Requirements

### 3.3 Algorithm For HAR

In this part, we introduce the algorithm final used in our application for human activity recognition. We have tried multiple algorithms, with different feature extractors or with different network layouts, some of which will be discussed later, but only the final version is explained here in this part.

#### 3.3.1 Training Dataset

In our specific task, the core dataset is given by instructors after sorting out all the data that the whole class had collected, i.e., the Respeck Recordings Clean dataset so our model concentrates on this specific data set. This concentration highlights the performance on the given dataset as the most important indicator of how the model works on our particular task and how it should be evaluated against other models.

The dataset can be visited at:
https://github.com/specknet/pdiot-data/blob/master/2021/Respeck_recordings_clean.csv

#### 3.3.2 Model Overall Architecture

We use CNN Network for this task, with several batch normalisation, activation, max-pooling layers for better performance. The overall architecture is shown in the following Figure2.

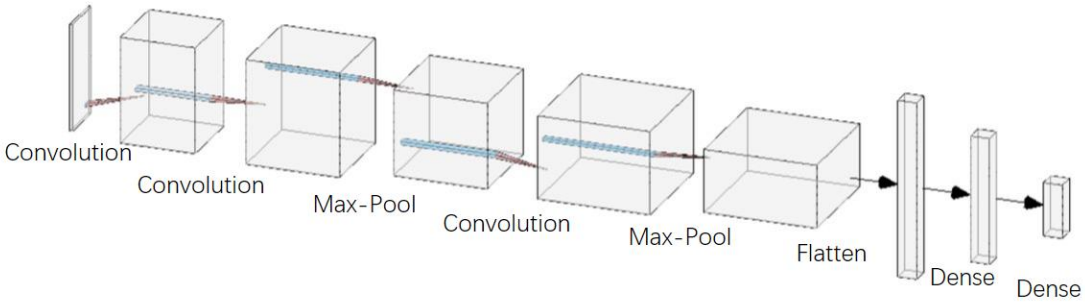



Figure2: CNN Model

### 3.3.3 Layers in the Model

The layout and key specifications of the layers are shown below, and details will be interpreted in the following sections.

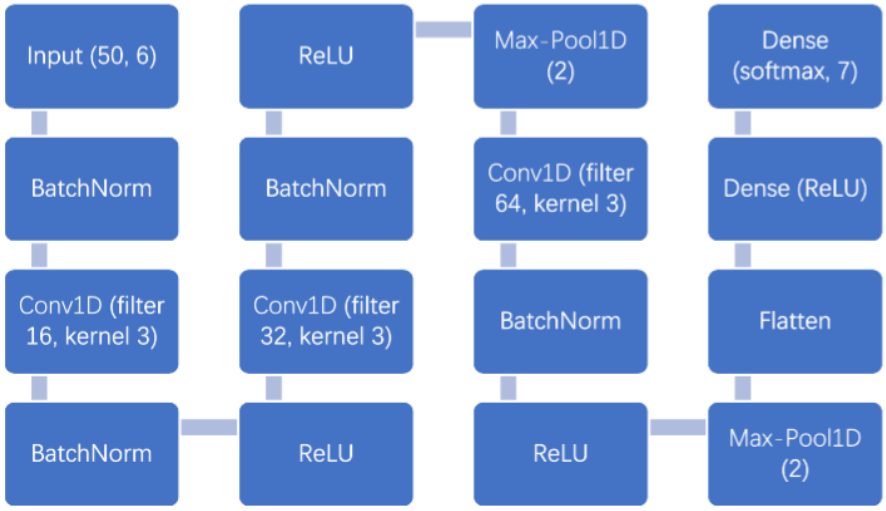



Figure3: Layout and Key Specifications of the Layers in the Model

#### 3.3.3.1 One Dimensional Convolutional Layers

As the essential parts of the whole CNN network, convolutional layers serve as the most significant feature extractors. And because the input data are basically data of different dimensions of the sensors in different time points, we use one-dimensional convolutional layers to capture features at each dimension of the sensors at different times without mixing different dimensions of sensors at every single receptive field.

There are 3 convolutional layers in our model, all of which have the same kernel size of 3. The kernel size decides how much data is viewed each time by the kernel, and because our Network is deep and the input data dimension is not very high, this small kernel size is enough in our application scenario.

The three layers have different filters, 16, 32, and 64 respectively. Padding is not applied in this model.

#### 3.3.3.2 Batch Normalization Layers

Batch normalisation layers basically subtract the mean value from the input value and scale the result with standard deviation so that the mean value is close to 0 and the standard deviation close to 1.

In Keras, in the inference phase, the batch normalisation will apply mean and deviation values that are seen in the training process. This simplifies its application for inferencing and especially facilitates processing input data.

A batch normalisation layer is applied first before any other operations on the input data. This is particularly helpful in this scenario when different sensors are feeding data with various scales, deviations and mean values.

Batch normalisation layers are also utilised after each convolutional layer to set the output of respective layers to normal distribution to avoid gradient vanishing or gradient exploding.

#### 3.3.3.3 Activation Layers

There are activation layers after every convolutional layer. Activation layers are some non-linear functions that change the whole Network from linear into non-linear, making it more powerful to represent more functions.

The output layer uses softmax as the activation function, which maps the input data to the range from 0 to 1, thus satisfying our requirement to get the possibility for every category for the multiple classifications.

The other activation layers, used for convolutional layers, have the ReLU activation function. This function is also a non-linear function, whose value is 0 for inputs less than 0 and equals to the input value otherwise. Although the function seems simple, it is helpful in CNN because its gradient is easy to calculate, and its gradient will not disappear when the input value is large as other functions like the sigmoid function does. The zero part is capable of giving the Network a certain sparsity, which is useful to avoid overfitting, and overfitting is a particularly significant problem in this scenario because our training data is not very large.

#### 3.3.3.4 Max Pooling Layers

Pooling layers are also popular components in CNN algorithms. They change input data by some operations on certain pairs and result in fewer output data, and therefore they can reduce the dimension of the data and layers, avoid over-fitting, and can also extract more prominent and more important features from the data, avoiding some abnormal values.

In this model, we use max-pooling layers, which in each stride get a certain number of data and output the maximum value to the next layer. Intuitively, this kind of pooling concentrates on the most prominent values in the data. This also makes the relative order of the time data

invariant, which is meaningful in this scenario. We set the pool size to 2 and stride to 2, which gets the maximum value from two input values each time, and reduces the input dimension to half.

### 3.3.4 Training of the Model

Training is also an important aspect to consider in the process of getting the final model for use.

For the given dataset, we first divide them into windows of samples at consecutive timestamps, where the window size, which is the same as what is input to the model, is 50, and the step size of dividing the windows is 25 timestamps. And then we take 20% of the data as a test set, and the remaining 80% as a training set, when we split the two sets with the same portion of labels, i.e. stratified with the labels.

We use 10% of the data as the validation split in the training process. The optimiser is SGD, i.e. stochastic gradient descent optimisation, and the learning rate is set to 0.1, with loss of binary cross entropy, and optimisation metrics of accuracy. The batch size is 16, trained 10 epochs.

## 3.4 Mobile Application

### 3.4.1 Wireless communication

The user activity data is collected by the Respeck, a low power wearable wireless sensor. Respeck could measure acceleration in three orthogonal axes and angular velocity. Bluetooth transfers the data in 25HZ. The smartphone would receive data to perform the analysis. The smartphone connection process is as follows:

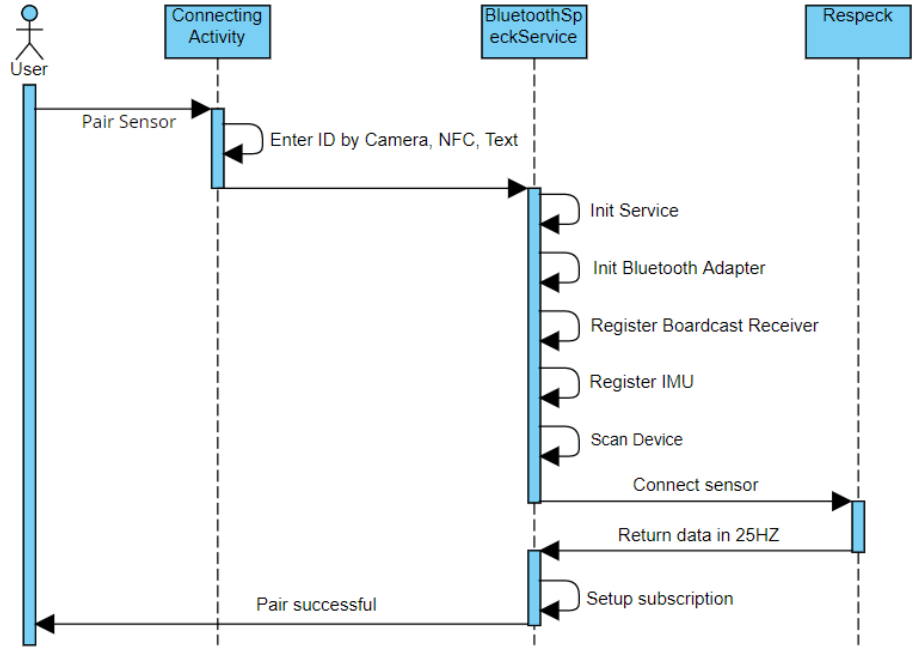



Figure4: Wireless Communication Sequence Diagram

In addition, to check whether the sensor connections are successful, this application uses a special algorithm to detect it. Firstly, the data receiving process should be claimed. This software initialises a three hundred elements array to receive the sensor data. The sensor data is received by the software each 1/25 second, and each data contains 6 values, which are three orthogonal axes and angular velocity. Therefore, the array could store 2 seconds of sensor data (It could be shown as a sliding window). Whenever new sensor data is passed in, it will be added to the end of the array. Then the first six elements in the array would be dropped. It could ensure that new data is added and avoid memory leaks. This algorithm would select a small range of data in the array as the prior array. This prior array must be far away from the end of the initial array to avoid false alarms. The position of the a priori array is usually chosen between 100-150. Then the software will pick the same position as the posterior array one second later. By comparing the priori array and posterior array, this software could judge whether the sensor was connected successfully.

### 3.4.2 Database(Firebase)

The sensor and user customised data should be stored and analysed in the application. There are two methods to achieve it: stored locally and hosted in the Cloud. SQLite and CSV could be applied to store data locally. However, local storage is easily lost and cannot be accessed by other devices. Therefore, this application chooses to store data in the Cloud, which is hosted in the Google Firebase.

Firebase could accelerate and scale application development without managing infrastructure[33]. It supports Realtime Database, Machine Learning, Authentication, Cloud Message, etc. This application uses Realtime Database as the data storage. There are four reasons for using the Realtime Database. The first one is that it brings multiple devices and off-site access. The user could access their data at any device and any position due to the data being hosted in the Cloud. The second one is the outstanding database performance. The Realtime Database uses NoSQL(JSON) datatype to store the data. The flat structure JSON tree speeds the data storing and querying, shown in Figure5. Every user has a subtree in the JSON tree named the Unique User ID(UID). Then the UID sub-tree is divided into three parts: MaxTime value, phone value, and Respeck subtree.

The user max sitting time is stored in the MaxTime, and the emergency contact phone number is stored in the phone. Finally, the subtree Respeck contains many keys named data and time, which means the time of data creation, such as '20211122012547729'. Every key in the subtree Respeck has seven values: x, y, z, gz, gy, gz and time. The seven values store the data from the wearable sensor(Respeck). The third one is that Firebase could support Offline uploading. When the user records their data without connecting to the Internet, the data will automatically be stored in the SmartPhone as a buffer. Then the buffer would be uploaded when connecting to the Internet. The last one is that it easily integrates into the Android application. The uploading and downloading process are completed by a simple method, which is shown in Figure6.

In addition, database security is also considered. Firstly, General Data Protection

Regulation(GDPR) and California Consumer Privacy Act(CCPA) are applied to the Firebase. Secondly, the data is encrypted by HTTPS in the transmission. Thirdly, all the data in the Realtime Database are protected by some special rules, which could ensure the user only access their only subtree. The Database rules are shown in Figure7.

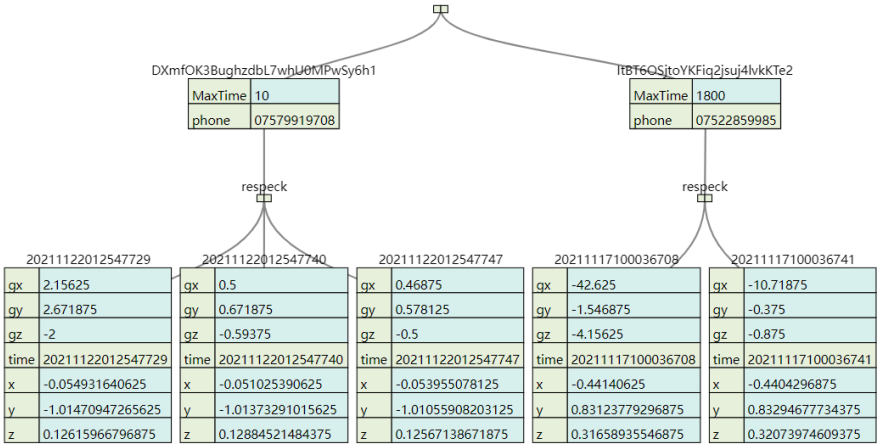



Figure5: Firebase JSON Database Structure

```
//add data to database(Cloud)
val respeck = Respeck(x, y, z, gx, gy, gz,formatted) as Respeck
val database =  FirebaseDatabase.getInstance().reference
database!!.child(uid).child("respeck").child(formatted).setValue(respeck)

//get Respeck data from Cloud
val ref2 = FirebaseDatabase.getInstance().getReference(uid).child("respeck")
val menuListener2 = object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        for (child in dataSnapshot.children) {
            val respeck = child.getValue(Respeck::class.java)
            respeckX.add(respeck!!.getX())
            respeckY.add(respeck!!.getY())
            respeckZ.add(respeck!!.getZ())
            respeckGX.add(respeck!!.getGX())
            respeckGY.add(respeck!!.getGY())
            respeckGZ.add(respeck!!.getGZ())
            respeckTime.add(respeck!!.getTime())
        }
        drawRespeckChart()
    }
    override fun onCancelled(databaseError: DatabaseError) {
        // handle error
    }
}

ref2.addListenerForSingleValueEvent(menuListener2)
```

Figure6: Firebase Uploading & Downloading Process

```
{
  "rules": {
    ".write": "auth != null",
        ".read": "auth != null",
      "$user_id": {
        ".write": "$user_id === auth.uid",
          ".read":"$user_id === auth.uid"
    }
  }
}
```

Figure7: Database rules

### 3.4.3 Authentication(Firebase)

The previous section mentioned that the different user data is divided into subtrees by UID. Therefore, this part will spread how to generate the UID.

The authentication is also hosted on the Firebase, which is already discussed above. The user could register their account by email address and password. Then Firebase could check the email address and return the status of Existing Email or Register Successful. Each user would be assigned a unique ID(UID) to be recognised by Firebase. UID is invisible to the user. Therefore, the user could log in directly by email address and password next time. In addition, the user is allowed to reset their password by email. After the user submits the reset password request, Firebase sends an email to the user email address. The user could easily follow the instructions in the email to reset the password. The email will be shown in Figure8.

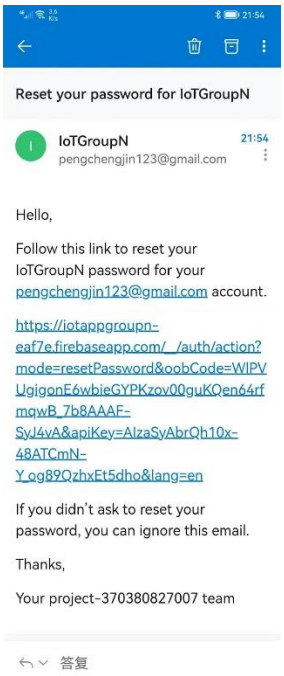



Figure8: Reset Password Email

### 3.4.4 Activity Recognise(TensorFlow Lite)

After storing the data from the Respeck, the analysis should be applied. The activity recognises model is already discussed in Section 3.3. The application will invoke the model every second by the Handler() method to recognise the user activity. In the Handler, the collected sensor data will be loaded from ArrayList format data to TensorBuffer as the input feature, which will be given to the process() method of the Model object generated by Android Studio when importing the Tensorflow Lite model. Then in the floatArray format of the outputFeature0AsTensorBuffer member of the output of the process() function, there are probabilities of all the labels, where the label with the highest probability is treated as the most possible activity recognised. Finally, the recognised result will be updated to the User Interface as text and image.

### 3.4.5 History Data

The sensor data is updated into the Cloud every 1/25 seconds(25HZ). The upload and download process is already discussed in Section 3.4.2. The history data are shown as two line charts, which will be discussed in Section 3.4.8.

### 3.4.6 Sedentary Reminder

This is an application for human activity recognised. The application would check the current human activity every second. If the classifier result is 'Sitting', a counter is applied to add one. Otherwise, the counter will be set to zero. It ensures that only continuous Sitting activity would be counted in the Sedentary Reminder. When the counter is equal to the max sitting time, a small popup message is displayed on the screen(Toast) and a notification in the status bar and drawer. The Max Sitting Time could be set in the application, and the default is 10 seconds for testing. The pseudo-code is as follows:

```
sittingTime=0
#run it every second
Handle runnable(1000ms){
   #show Toast and notification when sitting too long
   if sittingTimemore than Max sitting time: show Toast
   if sittingTime equal to Max sitting time: show notification
   #run the recognise model
   result=run CNN model
   if result=sitting: sittingTime =sittingTime +1
         otherwise: sittingTime =0
}
```

### 3.4.7 Fall Alert

The is an application for human activity recognised. The Fall Alert would check the user whether they were falling. If the user were falling, the application would show an Alert Dialog. The user should click the Dialog within 10 seconds. Otherwise, the application would call the emergency contact, which is set by the user before(The default phone number 999). The pseudo-code is as follows:

```
#run it every second
Falling=false
dialogTime=0
Handle runnable(1000ms){
   #count the dialog showing time
   if dialogTime more than 1: dialogTime=dialogTime+1
   #run the recognise model
   result=run CNN model
   #show notification and   Alert Dialog when falling
   if result = Falling:
   Falling=true
   show notification & invoke Alert Dialog
```

```
    if dialogTime is up to 10 seconds:
            call emergency contact ()
}
# Alert Dialog setting
 Alert Dialog{
    set Title="Falling warning"
     & set Message =Falling, Are you hurt?
    & setPositiveButton=call emergency contact ()
     & setNegativeButton=cancel dialog& dialogTime=0 & Falling=false
}
call emergency contact (){
    dialogTime=0
    Falling=false
    call
}
```

### 3.4.8 User Interface

#### 3.4.8.1 Respeck Data Showing

The Respeck data are shown in the line chart. The horizontal coordinate is time and the vertical coordinate is angular velocity or acceleration. The line chart is based on the MPAndroidChart, a powerful and easy to use chart library in Android[34].

#### 3.4.8.2 Recognise Model Result Showing

The result is shown as text and image. Therefore, there is a TextView and ImageView in the Layout file. All of the Image resources and text are stored in the file. After the application recognises the activity, the new image is updated by changing the ImageView's resource, and the text is updated by changing the TextView.

#### 3.4.8.3 Input Box

The user's input is listened to by the Text fields supported by the Google Material Design[35] Compared to the default text input component in Android, Material Design Text fields are more beautiful and customisation, such as custom icons and helper text. It could be applied by adding dependency and declaring basic information in the layout file. Finally, a listener is added to it for listening to the user's input.

#### 3.4.8.4 Navigation Drawer

Navigation Drawer is the main navigation component in the application. The user could change different functions(activities) by clicking the Navigation Drawer. It is opened by a navigation menu icon in the top application bar. The Navigation Drawer could be divided into two parts: HeaderView and MenuView.

HeaderView contains the user's email address obtained from the Firebase. The header view is an XML layout file that could be integrated into NavigationView by activity code.
MenuView is the main part of changing different functions. There are three steps to integrate it into the application: setting menu layout, setting NavigationView layout, and adding the listener.

#### 3.4.8.5 Floating Action Button(FAB)

The FAB is a circular shape button with an icon in its centre. This application uses the FAB as the entrance of pairing the sensor. FAB is supported by Material Design, so it is easy to integrate into the application. Two steps are used to integrate it: declaring the XML layout file and adding a clicks listener.

### 3.5 Software organisation

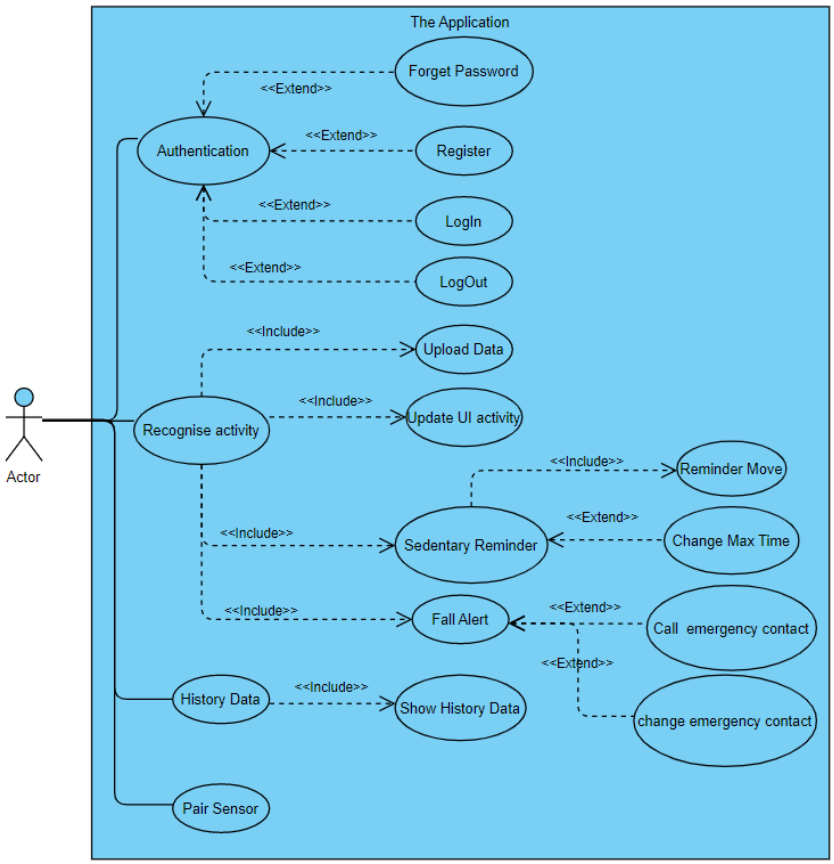



Figure9: Use Case Diagram

Seventeen use cases could interact with users. All the use cases could be divided into four main functions: Authentication, Recognise, History Data, and Pair Sensor. Therefore the four main functions will be discussed following. Only the relationship and organisation between four functions is considered in this part due to relative technology implementation were already

discussed above.

The authentication function is based on the Firebase, which is already discussed above. The main purpose of this function is that it could provide multi-device access and classify multiple user data in the Cloud. The Register, Login and Logout functions could be implemented in the application. However, the Forget password function must be implemented in the authenticated websites because this application could not support enough security to identificate users who forgot the password.

Pair Sensor function technology is already discussed above. Therefore its relationship and connection with other functions are discussed in this part. Pair Sensor is the most basic function in the application. The Recognise, History Data function is based on the Pair Sensor function. Recognise Activity is the main function in the application. It contains recognised results showing, sensor data uploading and the recognition result application. The recognised result showing includes sensor data receiving, CNN model recognised and UI updated. Sensor data uploading is the primary function in History Data. All the data was uploaded to the Firebase Realtime Database. Finally, this app achieved two applications on how to use recognised results: Sedentary Reminder and Fall Alert.

History data allows users to view sensor history data. The application will download the data from the Firebase Realtime Database when the user enters this function. All data are shown in the line chart.

# 4. Testing

## 4.1 Software

This software follows agile development through multiple iterations to reach the final version of the software. This application has six iteration versions. The details are as follows:

Version 1: Recognise Activity(Pair Sensor, import CNN model & update recognise UI)
Version 2: Authentication(Login, Logout, Register, Forget password)
Version 3: Database build(Upload & download the history data)
Version 4: Sedentary Reminder
Version 5: Fall Alert
Version 6: UI Optimisation

As agile development is used, exhaustive testing is necessary. Therefore, this application will focus on the Unit Test, Function Test, Performance Test and Compatibility Test.

### 4.1.1 Unit Test

The unit test is the smallest test unit. It could find the bug early and help to enhance software reusability. The unit test runs on the physical devices because many Android framework APIs are used. The test result tables are as follows.

| Instrument Test (Unit Test) - ConnectingActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:20/10/2021 | | | | Test Version:1.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Static resource loading | Judging whether the static resources are loading | Null | Detect whether layout file is exists | layout file exists | layout file exists | Success |
| 2 | Bluetooth Speck Service | Judging whether it is scanning the Respeck | Null | Find the services on the phone | Service exists | Service exists | Success |
| 3 | NFC reader | Judging whether NFC could read the ID | Null | Let Respeck near the Phone, check whether ID is read. | ID exists | ID exists | Success |
| 4 | Camera reader | Judging whether Camera could read the ID | Null | Scan the Respeck QR code, check whether ID | ID exists | ID exists | Success |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | is read. | | | |

| Instrument Test (Unit Test) - LiveDataActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:20/10/2021 | | | | Test Version:1.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Sensor data receive | Judge whether the sensor data are received | Null | Print the sensor data | 1/25 second per data, each has 6 numbers. | 1/25 second per data, each has 6 numbers. | Success |
| 2 | CNN model | Judge whether CNN could output the result | Sensor data | Print the output result | Activity | Activity | Success |
| 3 | Static resource loading | Judging whether the static resources are loading | Null | Detect whether layout file is exists | layout file exists | layout file exists | Success |
| 4 | Dynamic UI update | Judging whether the dynamic resources are updated | Activity Text | Judge the updated image and text | New text and image | New text and image | Success |

| Instrument Test (Unit Test) - LoginActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:05/11/2021 | | | | Test Version:2.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | User input listening | Judge whether the application could listen to the user input | User input | Find the input box resource | User input text | User input text | Success |
| 2 | Upload email address and password to the Firebase | Judge whether the application could send the login data to the Firebase | Email address and password | Get the return status | Login successful | Login successful | Success |

| Instrument Test (Unit Test) - RegisterActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 05/11/2021 | | | | Test Version:2.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | User input listening | Judge whether the application could listen to the user input | User input | Find the input box resource | User input text | User input text | Success |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | Upload email address and password to the Firebase | Judge whether the application could send the registration data to the Firebase | Email address and password | Get the return status | Register successful | Register successful | Success |

| Instrument Test (Unit Test) - MeActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 05/11/2021 | | | | Test Version:2.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Static resource loading | Judging whether the static resources are loading | Null | Detect whether layout file is exists | layout file exists | layout file exists | Success |
| 2 | Logout | Judge whether the user logout | Null | Detect Login status | Not in the Login | Not in the Login | Success |

| Instrument Test (Unit Test) - ResetActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 05/11/2021 | | | | Test Version:2.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | User input listening | Judge whether the application could listen to the user input | User input | Find the input box resource | User input text | User input text | Success |
| 2 | Upload email address to the Firebase | Judge whether the application could send the reset data to the Firebase | Email address | Get the return status | Reset email sent successfully | Reset email sent successfully | Success |

| Instrument Test (Unit Test) - HistoryActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:10/11/2021 | | | | Test Version:3.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Static resource loading | Judging whether the line chart is loading | Null | Detect whether line chart is exists | line chart exists | line chart exists | Success |
| 2 | Get data from Firebase | Judge whether data are received from Firebase | Null | Find the data | History Data is downloaded | History Data is downloaded | Success |

| Instrument Test (Unit Test) - LiveDataActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:15/11/2021 | | | | Test Version:4.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Timer | The test gap is 1 second, so need to check whether is 1 second | Null | Calculate timer gap | 1 second | 1second | Success |
| 2 | Get Max sitting time | Judge whether the application could get the Max sitting time from the Firebase | User ID | Download the Max sitting time | 1800 seconds | 1800 seconds | Success |
| 3 | Sedentary Reminder test | Judge whether the application could recognise sitting | Sensor data | Judge the CNN output | Sitting | Sitting | Success |
| 4 | Sedentary Reminder notification | Judge whether the application could send notification | Sitting | Judge whether notifications are shown | Notification shown | Notification shown | Success |

| Instrument Test (Unit Test) - LiveDataActivity | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:16/11/2021 | | | | Test Version:5.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Timer | The test gap is 1 second, so need to check whether is 1 second | Null | Calculate timer gap | 1 second | 1second | Success |
| 2 | Get emergency contacts phone | Judge whether the application could get the emergency contacts phone from the Firebase | User ID | Download the emergency contacts phone | 07579919708 | 07579919708 | Success |
| 3 | Fall Alert test | Judge whether the application could recognise falling | Sensor data | Judge the CNN output | falling | falling | Success |
| 4 | Fall Alert Call emergency contacts | Judge whether the application could Call emergency | Sitting | Judge whether call is sending | Call emergency contact | Call emergency contact | Success |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | contacts | | | | | |

### 4.1.2 Functional Test

The Functional tests focus on the initial functional requirement. The test tables are as follows.

| Functional Test - Recognise | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 20/10/2021 | | | | Test Version:1.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Bluetooth Speck Service | Judging whether it is scanning the Respeck | Null | Find the services on the phone | Service exists | Service exists | Success |
| 2 | CNN model | Judge whether CNN could output the result | Sensor data | Print the output result | Activity | Activity | Success |
| 3 | Dynamic UI update | Judging whether the dynamic resources are updated | Activity Text | Judge the updated image and text | New text and image | New text and image | Success |

| Functional Test - Authentication | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 05/11/2021 | | | | Test Version:2.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Login | Judge whether the application could send the login data to the Firebase | Email address and password | Get the return status | Login successful | Login successful | Success |
| 2 | Forgot password | Judge whether the application could send the reset data to the Firebase | Email address | Get the return status | Reset email sent successfully | Reset email sent successfully | Success |
| 3 | Register | Judge whether the application could send the registration data to the Firebase | Email address and password | Get the return status | Register successful | Register successful | Success |

| Functional Test - Database | |
|---|---|
| Test date: 10/11/2021 | Test Version:3.0 |

| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
|---|---|---|---|---|---|---|---|
| 1 | Get data from Firebase | Judge whether data are received from Firebase | Null | Find the data | History Data is downloaded | History Data is downloaded | Success |
| 2 | Get Max sitting time | Judge whether the application could get the Max sitting time from the Firebase | User ID | Download the Max sitting time | 1800 seconds | 1800 seconds | Success |
| 3 | Fall Alert Call emergency contacts | Judge whether the application could Call emergency contacts | Sitting | Judge whether call is sending | Call emergency contact | Call emergency contact | Success |

| Functional Test - Sedentary Reminder | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 15/11/2021 | | | | Test Version:4.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Sedentary Reminder | Judge whether the application could recognise sitting | Sensor data | Judge the CNN output | Sitting | Sitting | Success |

| Functional Test - Fall Alert | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date: 16/11/2021 | | | | Test Version:5.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | Fall Alert test | Judge whether the application could recognise falling | Sensor data | Judge the CNN output | falling | falling | Success |

### 4.1.3 Database Performance Test

The performance test is focused on the server and client's responsiveness and stability under a particular workload. The test tables are as follows.

| Performance Test –Database | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:18/11/2021 | | | | Test Version:6.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | Max Sitting time obtain time test | Calculate the time spend when requesting Max Sitting time | number | Request data from Firebase | At most 2 second | Average 1 second | Success |
| 2 | Emergency contact phone obtain time test | Calculate the time spend when requesting Emergency contact phone | number | Request data from Firebase | At most 2 second | Average 1 second | Success |
| 3 | History data obtain time test | Calculate the time spend when requesting History data | number | Request data from Firebase | At most 5 second | Average 3 second(Most 24 seconds) | Fail |

### 4.1.4 Software Performance Test

In this part, the CPU usage, battery consumption, RAM will be tested in this part.

| Performance Test – Client | | | | | | | |
|---|---|---|---|---|---|---|---|
| Test date:18/11/2021 | | | | Test Version:6.0 | | | |
| No. | Name of case | Description | Input data | Action | Expected output | Actual output | Success/Fail |
| 1 | CPU utility test | Monitor CPU utility when using the application | Null | Monitor CPU utility | Null | No more than 20% of total 2.36GHZ | Success |
| 2 | UI fluency test | Detecting interface lag | Null | Monitor UI fluency | No significant lag | No significant lag | Success |
| 3 | Battery consumption test | Calcite battery consumption when the application is running | Null | Monitor battery consumption in 10 minutes | Null | 75.68mAh /10min | Success |
| 4 | RAM consumption test | Calcite RAM consumption when the application is running | Null | Monitor RAM consumption | Null | Average RAM usage 70MB | Success |
| 5 | Cellular data consumption test | Calcite Cellular data consumption when the application is running | Null | Monitor Cellular data consumption | Null | 7.89MB/10min | Success |

### 4.1.5 Compatibility Test

This will focus on the application's compatibility with other devices.

| Compatibility Test – Device & Android System | | | | | | |
|---|---|---|---|---|---|---|
| Test date:03/11/2021 | | | | Test Version:6.0 | | |
| No. | Device | Android System | Screen Size | Screen Relation(px) | UI Success/Fail | Run Success/Fail |
| 1 | Huawei VOG-AL00 | EMUI11.0(Android10) | 6.47 inch | 2340*1080 | Success | Success |
| 2 | SamSung-G9500 | OneUI1.0(Android9) | 5.8 inch | 2960*1440 | Success | Success |
| 3 | SamSung-G9880 | oneUI3.0(Android11) | 6.9 inch | 3200*1440 | Success | Success |
| 4 | Nexus S | Android5.1 | 4.0 inch | 480*800 | Fail | Success |

## 4.2 Model

The machine learning model is a significant part of this activity recognition application, where the prediction, i.e. activity recognition is actually done, so testing focused particularly on the performance of the model is of great importance.

For comparison, we also tested on the 5 Class Model, i.e., model for the 5 basic activities (falling, lying, walking, running and siting/standing) as mentioned in the course, in addition to our main 7 Class Model.

### 4.2.1 Test on Datasets

When developing the model, we use the test split from the abovementioned training dataset, i.e., the Respeck Recordings Clean dataset, where we split 20% of the data as test set. After the development is finished, and peer review has been made, we also did some tests on the unseen test set, although this dataset is intended to be used for testing other group's model, because we believe this unseen dataset better shows how the model actually works, and it is helpful to mention these results in this report.

#### 4.2.1.1 Evaluation Scores

The central part of our evaluation of the model is some quantitative scores of the model on the given dataset. They are about how the predicted labels match the ground truth labels given in the dataset.

Precision is the quotient of the number of corrected predicted positive samples over the number of all the samples predicted as positive.

The recall is the quotient of the number of corrected predicted positive samples over the total number of actually positive samples.

F1 score is the harmonic mean of precision and recall.

Macro precision, macro recall or macro f1 score is the class-wise average of the corresponding score.

Accuracy is the total number of samples divided by the number of correctly categorised samples.

In human activity recognition applications, accuracy is a frequently used score to describe the performance of the model, but as the dataset is biased, the f1 score should also be paid much attention to.

| Model Evaluation Scores on training set and unseen test set | | | | | |
|---|---|---|---|---|---|
| Test date:14/12/2021 | | | | | |
| No. | Name of case | Macro Precision | Macro Recall | Macro F1 | Accuracy |
| 1 | 7 Class Model on the test split of the training set | 0.93 | 0.93 | 0.93 | 0.94 |
| 2 | 5 Class Model on the test split of the training set | 0.98 | 0.97 | 0.97 | 0.96 |
| 3 | 7 Class Model on the unseen test set | 0.84 | 0.87 | 0.81 | 0.86 |
| 4 | 5 Class Model on the unseen test set | 0.87 | 0.87 | 0.82 | 0.87 |

Table3: Model Evaluation Scores on training set and unseen test set

#### 4.2.1.2 Confusion Matrix

The confusion matrix is a visualisation method used in machine learning to show the classification results, where each horizontal axis indicates the predicted labels and the vertical ones the actual labels. Therefore, a well-performing model will have a confusion matrix with great diagonal values, with other values being very low, whereas the non-diagonal positions with relatively higher values indicate that the corresponding two classes are easy to be misclassified by the tested model.

If some classes are shown on the confusion matrix to be easily misclassified, we change the model design accordingly or temporarily change our design target.

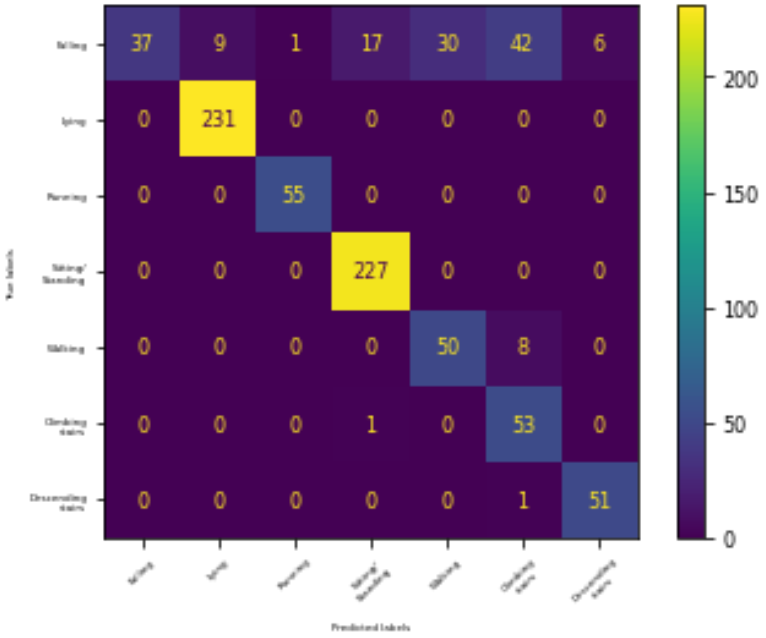



Figure10: Confusion Matrix of the 7 Class Model on the Unseen Test Set

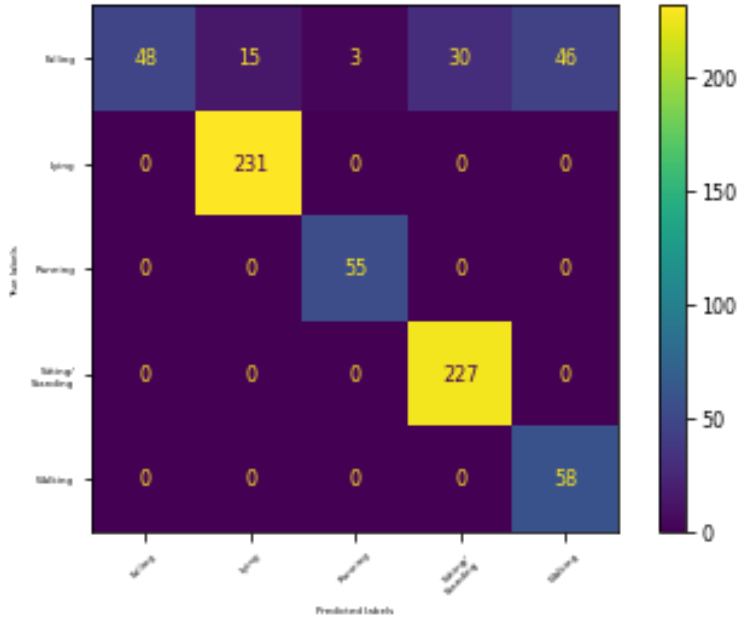



Figure11: Confusion Matrix of the 5 Class Model on the Unseen Test Set

#### 4.2.1.3 ROC Curve

In the ROC curve, the vertical axis is true positive rate, which is true positive divided by the number of actual positive samples, and the horizontal axis is false positive rate, which is false negative divided by the number of actual negative samples.

As this is a multi-class scenario, the testing uses a macro version of the ROC curve, which is the class-wise average of each ROC curve.

Once calculated, the AUC, i.e. area under the curve, is an important indicator of how the model

performs and inspires us to adjust the model.

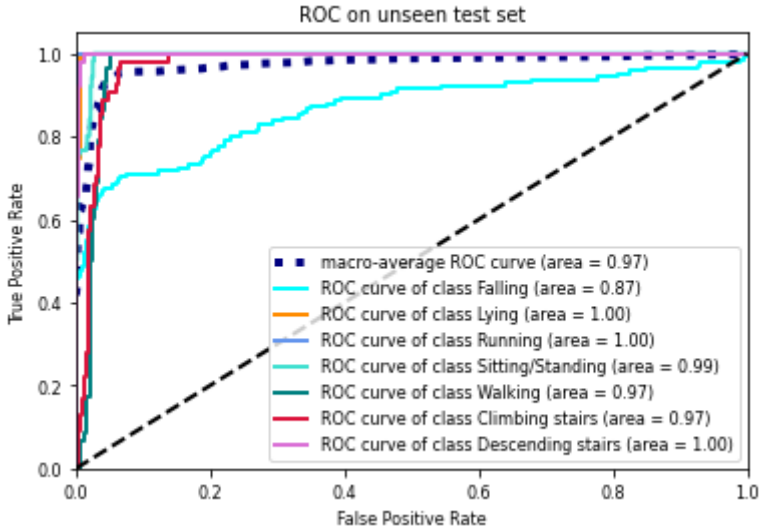



Figure12: ROC Curve and AUC of the 7 Class Model on the Unseen Test Set

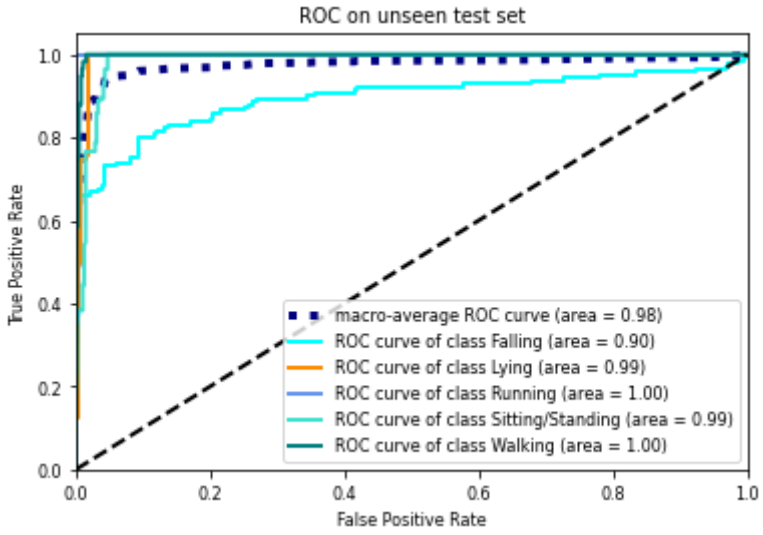



Figure13: ROC Curve and AUC of the 5 Class Model on the Unseen Test Set

### 4.2.2 Testing in Practice

For this wearable application, it is essential to understand how this model works in actual practice. Therefore, in addition to tests run by programs on the given dataset, it is useful to do more tests when connecting it to our real IoT devices and when using it as a normal user.

#### 4.2.2.1 Quick Testing with the Sensor

Our model takes input data from only the Respeck sensor, which is placed on the chest in a practical scenario. Hence, instead of testing the IoT system fully every time, it is a beneficial trick to first do some tests only on the sensor without placing it on someone's chest. This is a degraded version of a simulation test.

In this way, sitting and standing corresponds to placing the sensor on the table upright, whereas lying corresponds to placing the sensor on the table in other directions except for one direction which is the inversion of sitting and standing. Falling is a sudden movement of the sensor from a higher position to a lower position, with some rotation in that process. Walking is similar to pushing the sensor forward while moving it up and down, and running is a much faster version of running.

#### 4.2.2.2 Practical Testing

The quick simulation with barely the sensor only serves as a basic role to roughly check the correct connection of the sensor to the software and the connection of the model to any other parts of the software codes, and roughly see the prediction performance of the machine learning model. But for a more practical assessment of the performance, we still need to equip the sensor on the tester's body to see how it works in real manipulation.

In order to reduce the bias of the test related to the shape of the body of the testers or the habitual movements of the testers, the practice tests are also made by each of the three group members.

## 5. Results

We build two CNN models to recognise five activities and seven activities. The seven classification model could get 94% and 86% accuracy in the training set and test set. The five classification model has better accuracy. These data could be found in the Table3.

Due to the lack of competing products using Respeck sensor, this application will use two benchmarks to separate compare the model, power consumption, RAM and CPU consumption.

### 5.1 Compared to LSTM

In this work, we finally used the CNN model as our core HAR algorithm, but other networks are also popular in HAR, one of them is LSTM.

LSTM is the abbreviation of long short term memory, and is a kind of recurrent neural network architecture. It uses several gates to store the hidden states of the data, whence it can process the sequential information better, especially when some important relations in the series are shown in some data point with some gap between them, and it is more insensitive to the length of this gap than a traditional RNN. It also handles the vanishing gradient problem better than traditional RNN.

Intuitively, the human activity recognition task is also about time series, so it is tempting to try the LSTM model for this task. The comparison of our model and an optimised LSTM model is shown below.

| Comparison of CNN and LSTM results | | | | | |
|---|---|---|---|---|---|
| Test date:16/11/2021 | | | | | |
| No. | Name of case | Macro Precision | Macro Recall | Macro F1 | Accuracy |
| 1 | CNN | 0.93 | 0.93 | 0.93 | 0.94 |
| 2 | LSTM | 0.90 | 0.91 | 0.90 | 0.92 |

Table4: Comparison of CNN and LSTM results

It can be seen that LSTM also achieved reasonable results on this task, with similar results as the CNN model. The difference between their results is small, so it is likely that if we try more optimisation on the LSTM parameters, LSTM might do even better than our final CNN model. But as the difference is not large, and it is more important to focus on the overall methodology and implementation in the task in this limited time, so finally we chose to use the CNN model.

Moreover, it is also interesting to see that there is not much difference when we use LSTM, while intuitively it is more suitable for the HAR task. This might be because the patterns of HAR do not contain very long gaps of timestamps, and the serial properties are also not very obvious, making it enough to use CNN to capture the core features. In fact, it is interesting to dig even deeper into this question.

### 5.2 Compared to Google Activity Recognise API

Google Activity Recognise API is one of the Google Play Services that could accurately detect users' activity using low power signals from multiple sensors in the device[36] The developer does not have to handle the sensor data because this is a highly integrated API. The smartphone will send an activity broadcast that the application could receive whenever the user changes activity. The Activity Recognise API could detect seven activities: still, on foot, walking, running, in vehicle, on bicycle, and tilting. We built an Activity Recognise API demo in the Android system to recognise the seven activities above and show them in the user interface. Through performance monitoring, we found that the power consumption and CPU usage of running this application are no different from standby situations due to the computation and recognition being implemented in the Android system rather than this application. The performance of Activity Recognise API demo is in Figure14.

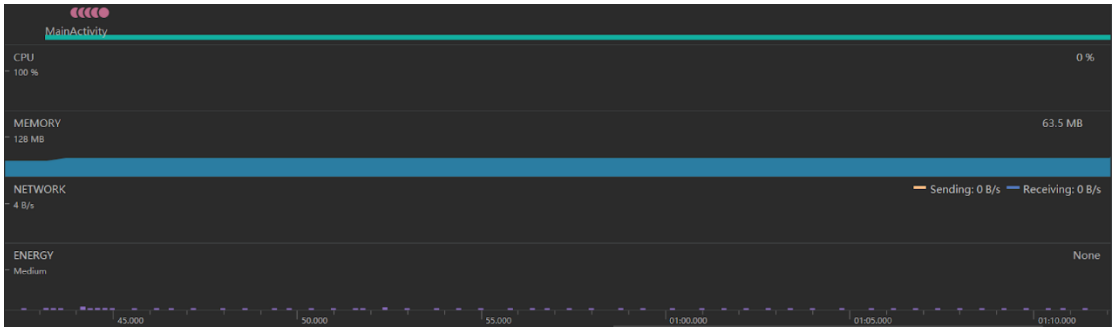



Figure14: Activity Recognise API Demo Performance Test

Section4 discussed the overall performance of the application. However, to compare with the Activity Recognise API, the performance of the detailed functions should be measured. Figure15 is the Login, Recognise and data uploaded performance graph. In Figure15, every wave peak in the Network means the application receives the sensor data and runs the CNN model. The CPU usage is positively correlated with network usage by corresponding to the same time point in the CPU. And the max CPU usage is no more than 40%. At the same time, when recognised activity is running, the RAM usage and battery consumption are increased significantly.

This result is different from the Activity Recognise API Demo Performance Test. Obviously, the Activity Recognise API has the lower CPU, RAM and battery usage. There are two reasons for explaining it:

- Firstly, Activity Recognise API detect frequency is lower than using Respeck. The main purpose of the API is to detect user activity has changed, rather than fine-grained

recognition.

- Secondly, Activity Recognise API is the Android System underlay service. Activity recognition is always performed regardless of the existence of software. Therefore, the CPU usage and power consumption have been optimised before being embedded in the system.

However, it does not mean that using the Respeck sensor to perform HAR is not good. Respeck could show the fine-grained physical orientation data in human activity. The developer could optimise the recognition model and perform better applications through this data. Secondly, the API uses the phone's internal sensors such as gyroscope and light sensor for measurement. Respeck is able to provide better precision compared to them. Therefore, Activity Recognise API is more suitable for daily monitoring of the general human, while using the Respeck sensor to perform HAR is suitable for precise monitoring of the specific population, such as patients with chronic diseases and athletes.

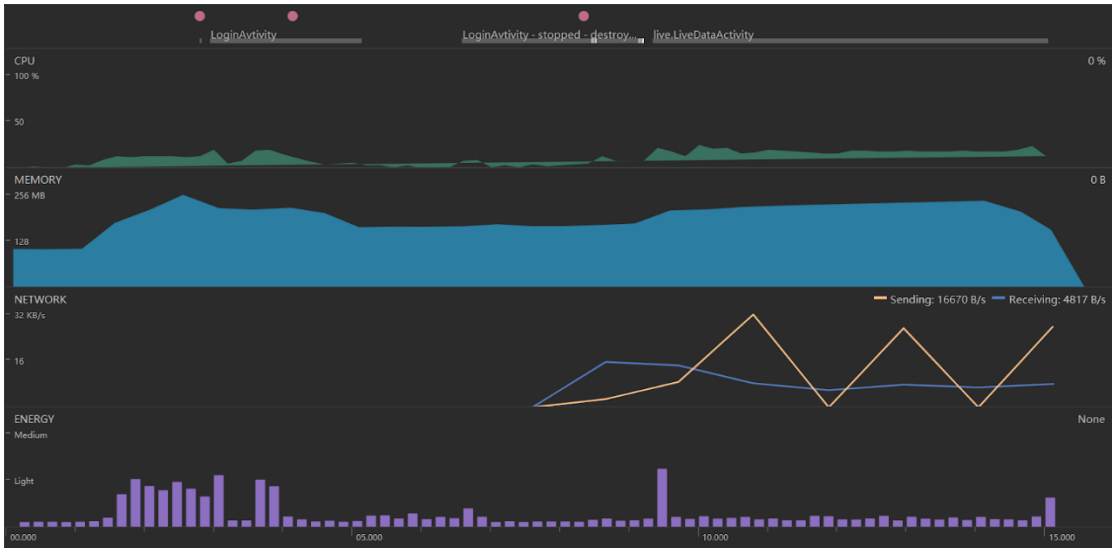



Figure15: The Application Login, Recognise and Data Uploaded Performance Test

### 5.3 Drawbacks

There are some other issues found in the testing of the software. So in this part, the issues will be explained and discussed how the software could be optimised in the future.
The history data downloading and line chart shows are slow when there is a lot of historical data. There are two reasons: data value type is complex and the low performance of MPAndroidChart. Figure16 shows the History data downloading and showing performance graph. The downloading progress consumption is four seconds, and the line chart showing needs 20 seconds. Complex data value type leads to heavy databases, for example, the 'X' value of Respeck is 0.474365234375. Actually, the value could be simplified to 0.474. Therefore, simplification of data is necessary. Next, the performance of MPAndroidChart declines when facing huge amounts of data. Segmented drawing seems to be a solution. The history data could be divided into small sections, then each section would be drawn continuously.

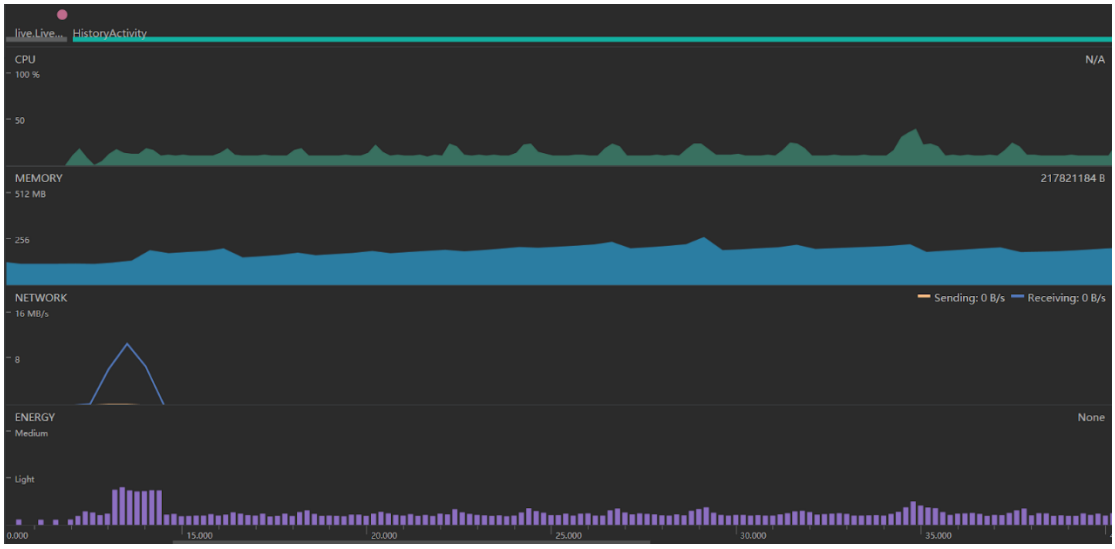


Figure16: History Data Downloading and Showing Performance Test

Our model performance could also be improved. First, for the training data, the Respeck clean dataset is biased as there is too little falling data, which has influenced the accuracy on especially falling activities. For better training data, we might make use of more data. Especially falling data, from the collection of the whole class, or utilise some data augmentation techniques to compensate for the lack. Second, the model could be made more complex to try to improve its performance. For example, we might also get input data from Thingy and combine it with Respeck data to reach better results. More network architectures besides the above-mentioned CNN and LSTM could also be tried and compared and even combined to find optimal models for this task.

## 6. Conclusion and Future Work

This paper presents a HAR system in the Android smartphone, using the Respeck sensor. The classifier is the CNN model because the LSTM model has similar accuracy and Google Activity Recognise API could only support coarse-grained recognition. The CNN model has 86% accuracy in the test set that could recognise seven activities: Falling, Walking, Sitting, Running, Lying, Descending stairs and Climbing stairs. In addition, this software has some other features, such as user authentication, history data viewing, sedentary reminder and fall alert. The user could also customise their software, such as max sitting time in the sedentary reminder and emergency contact in the fall alert.

However, this software has some drawbacks, such as server data type is complex, software line chart drawing slowly, recognition model accuracy and the compatibility is not enough. To improve this, there are some paths for future work.

1. Optimise the data that are uploaded to the server;
2. Reduce the line chart drawing time by segmented data;
3. Balance the training set data and add more training data to improve the model;
4. Use CNN-LSTM fusion model to improve the accuracy;
5. Test it in more devices for reducing software incompatibility.

## 7. Bibliography

[1]H. Nweke, Y. Teh, M. Al-garadi and U. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges", Expert Systems with Applications, vol. 105, pp. 233-261, 2018.

[2]D. Castro, W. Coral, C. Rodriguez, J. Cabra and J. Colorado, "Wearable-Based Human Activity Recognition Using an IoT Approach", Journal of Sensor and Actuator Networks, vol. 6, no. 4, p. 28, 2017.

[3]M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing", 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014.

[4]C. Rodriguez, D. Castro, W. Coral, J. Cabra, N. Velasquez, J. Colorado, D. Mendez and L. Trujillo, "IoT system for Human Activity Recognition using BioHarness 3 and Smartphone", Proceedings of the International Conference on Future Networks and Distributed Systems, 2017.

[5]W. Lu, F. Fan, J. Chu, P. Jing and S. Yuting, "Wearable Computing for Internet of Things: A Discriminant Approach for Human Activity Recognition", IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2749-2759, 2019.

[6]L. Minh Dang, K. Min, H. Wang, M. Jalil Piran, C. Hee Lee and H. Moon, "Sensor-based and vision-based human activity recognition: A comprehensive survey", Pattern Recognition, vol. 108, p. 107561, 2020.

[7]"TensorFlow", TensorFlow, 2022. [Online]. Available: https://www.tensorflow.org/. [Accessed: 20- Jan- 2022]

[8]K. Team, "Keras: the Python deep learning API", Keras.io, 2022. [Online]. Available: https://keras.io/. [Accessed: 20- Jan- 2022]

[9]W. Wu, S. Dasgupta, E. Ramirez, C. Peterson and G. Norman, "Classification Accuracies of Physical Activities Using Smartphone Motion Sensors", Journal of Medical Internet Research, vol. 14, no. 5, p. e130, 2012.

[10]S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen and M. Srivastava, "Using mobile phones to determine transportation modes", ACM Transactions on Sensor Networks, vol. 6, no. 2, pp. 1-27, 2010.

[11]A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development, pp. 1310-1315, 2016.

[12]G. Weiss, K. Yoneda and T. Hayajneh, "Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living", IEEE Access, vol. 7, pp. 133190-133202, 2019.

[13]E. Karakus and H. Kose, "Conditional restricted Boltzmann machine as a generative model for body-worn sensor signals", IET Signal Processing, vol. 14, no. 10, pp. 725-736, 2020.

[14]C. KyungHyun and R. Tapani and I.Alexander, "Enhanced Gradient and Adaptive Learning Rate for Training Restricted Boltzmann Machines", ICML, pp. 105-112,2011.

[15]G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", Science, vol. 313, no. 5786, pp. 504-507, 2006.

[16]Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu and M. Lew, "Deep learning for visual understanding: A review", Neurocomputing, vol. 187, pp. 27-48, 2016.

[17]J. Cao, W. Li, Q. Wang and M. Yu, "A Sensor-Based Human Activity Recognition System via Restricted Boltzmann Machine and Extended Space Forest", Internet and Distributed Computing Systems, pp. 87-94, 2018.

[18]S. Thomas, M. Bourobou and J. Li, "Ensemble of Deep Autoencoder Classifiers for Activity Recognition Based on Sensor Modalities in Smart Homes", Communications in Computer and Information Science, pp. 273-295, 2018.

[19]A. Varamin, E. Abbasnejad, Q. Shi, D. Ranasinghe and H. Rezatofighi, "Deep Auto-Set", Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2018.

[20]S. Bhattacharya, P. Nurmi, N. Hammerla and T. Plötz, "Using unlabeled data in a sparse-coding framework for human activity recognition", Pervasive and Mobile Computing, vol. 15, pp. 242-262, 2014.

[21]Y. Zhu, X. Zhao, Y. Fu, Y. Liu, "Sparse coding on local spatial-temporal volumes for human action recognition", InAsian conference on computer vision, pp. 660-671, 2010

[22]N. Junagade and S. Kulkarni, "Human Activity Identification using CNN", 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020.

[23]Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[24]T. Zebin, P. Scully and K. Ozanyan, "Human activity recognition with inertial sensors using a deep learning approach", 2016 IEEE SENSORS, 2016.

[25]W. Huang, L. Zhang, W. Gao, F. Min and J. He, "Shallow Convolutional Neural Networks for Human Activity Recognition Using Wearable Sensors", IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-11, 2021.

[26]S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 06, no. 02, pp. 107-116, 1998.

[27]S. Pienaar and R. Malekian, "Human Activity Recognition using LSTM-RNN Deep Neural Network Architecture", 2019 IEEE 2nd Wireless Africa Conference (WAC), 2019.

[28]F. Gers, "Learning to forget: continual prediction with LSTM", 9th International Conference on Artificial Neural Networks: ICANN '99, 1999.

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

[30]R. Mutegeki and D. Han, "A CNN-LSTM Approach to Human Activity Recognition", 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2020.

[31]S. Deep and X. Zheng, "Hybrid Model Featuring CNN and LSTM Architecture for Human Activity Recognition on Smartphone Sensor Data", 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2019.

[32]S. Gupta, "Deep learning based human activity recognition (HAR) using wearable sensor data", International Journal of Information Management Data Insights, vol. 1, no. 2, p. 100046, 2021.

[33]"Firebase Products", Firebase, 2022. [Online]. Available: https://firebase.google.com/products-build?authuser=0. [Accessed: 20- Jan- 2022]

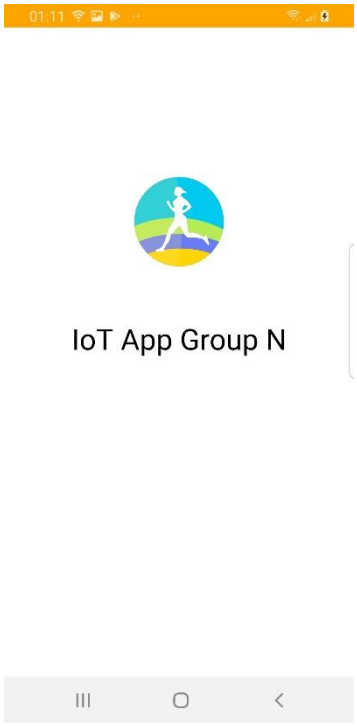
[34]"MPAndroidChart: A powerful Android chart view / graph view library", GitHub, 2022. [Online]. Available: https://github.com/PhilJay/MPAndroidChart#quick-start. [Accessed: 20- Jan- 2022]

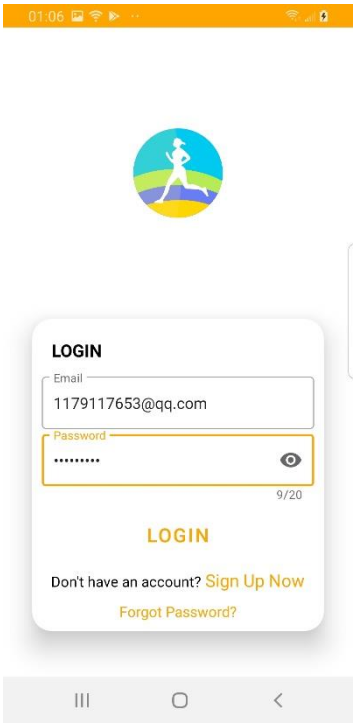
[35]"Material Design", Material Design, 2022. [Online]. Available: https://material.io/components/text-fields/android#using-text-fields. [Accessed: 20- Jan- 2022]

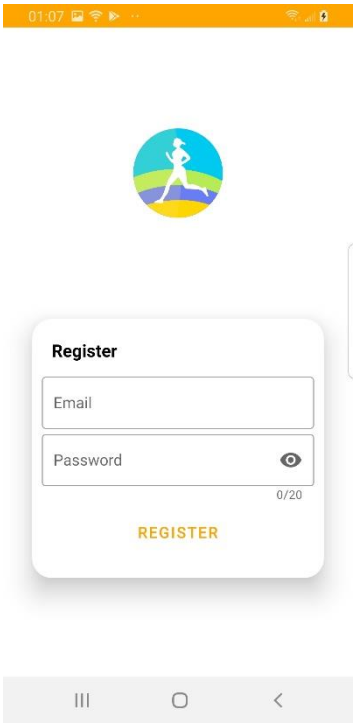
[36]"Activity Recognition API | Google Developers", Google Developers, 2022. [Online]. Available: https://developers.google.com/location-context/activity-recognition. [Accessed: 20- Jan- 2022]

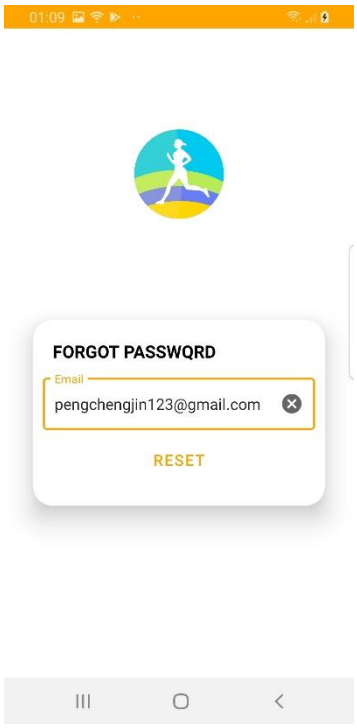
# 8. Appendices

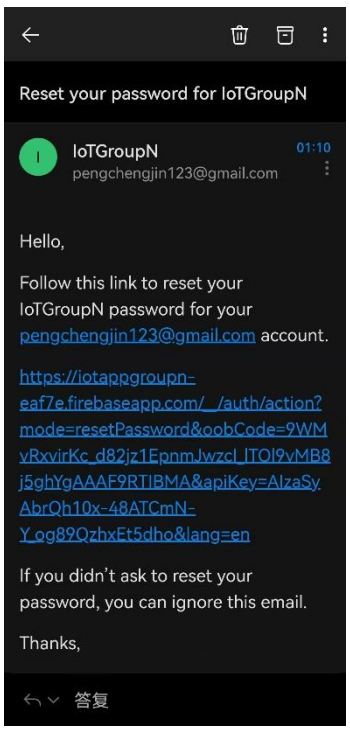
## 8.1 Application Screenshots

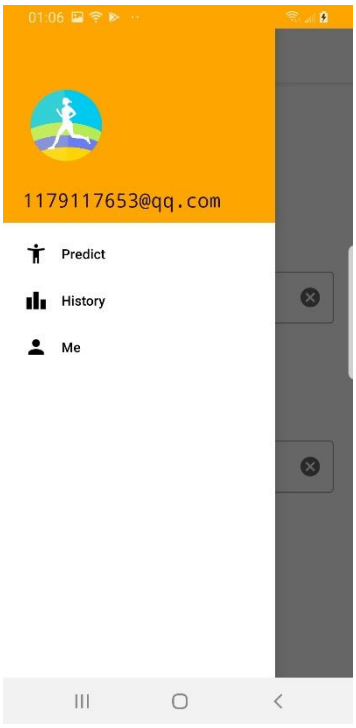


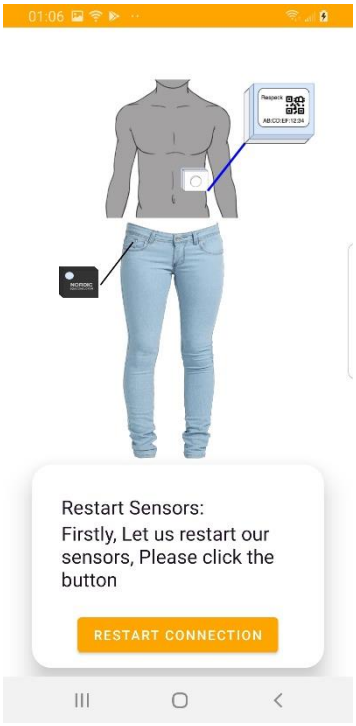



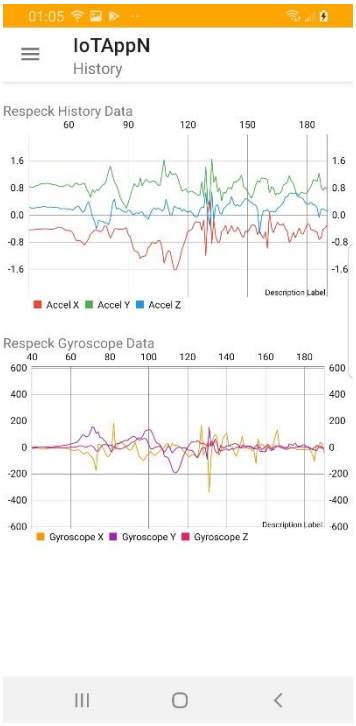



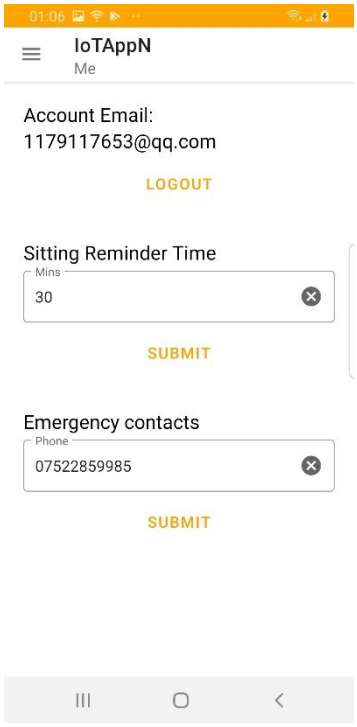



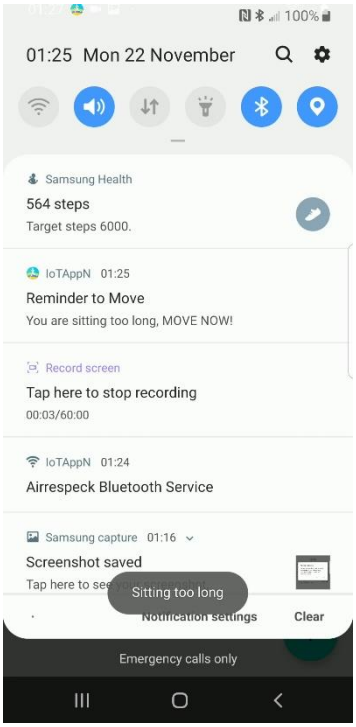



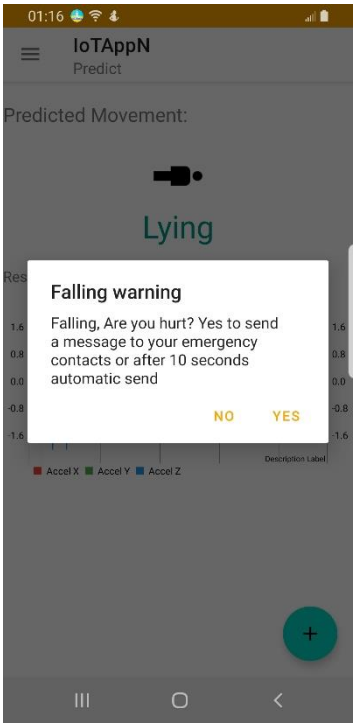

01:04
IoTAppN
Predict
Predicted Movement:
Sitting or Standing
Respeck Live Acceleration Data
300 330 360 390 420
1.6 0.8 0.0 -0.8 -1.6
Accel X Accel Y Accel Z
Description Label


01:06
Restart Sensors:
Firstly, Let us restart our sensors, Please click the button
RESTART CONNECTION


01:05
IoTAppN
History
Respeck History Data
60 90 120 150 180
1.6 0.8 0.0 -0.8 -1.6
Accel X Accel Y Accel Z
Description Label
Respeck Gyroscope Data
40 60 80 100 120 140 160 180
600 400 200 0 -200 -400 -600
Gyroscope X Gyroscope Y Gyroscope Z
Description Label


01:06
IoTAppN
Me
Account Email:
1179117653@qq.com
LOGOUT
Sitting Reminder Time
Mins
30
SUBMIT
Emergency contacts
Phone
07522859985
SUBMIT


100%
01:25 Mon 22 November
Samsung Health
564 steps
Target steps 6000.
IoTAppN 01:25
Reminder to Move
You are sitting too long, MOVE NOW!
Record screen
Tap here to stop recording
00:03/60:00
IoTAppN 01:24
Airrespeck Bluetooth Service
Samsung capture 01:16
Screenshot saved
Sitting too long
Notification settings
Clear
Emergency calls only


01:16
IoTAppN
Predict
Predicted Movement:
Lying
Falling warning
Falling, Are you hurt? Yes to send a message to your emergency contacts or after 10 seconds automatic send
NO
YES
Accel X Accel Y Accel Z
Description Label


66% 01:35
07749 808397
英国
正在拨号...
录音
等待
添加通话
备忘录
静音
联系人