PDIoT Coursework 3 (2022-23)

HUMAN ACTIVITY RECOGNITION APPLICATION

Group G
Shuo Chen – s1931698
Jingkai Yuan – s1941321
Hao Zhou – s1862323

January,2023

Abstract

Human Activity Recognition (HAR) is a challenging time series classification task. It involves predicting human motion based on inertial sensor data and involves deep domain expertise and methodologies from signal processing to raw data to properly extract features to fit machine learning models. Deep learning methods such as convolutional neural networks and recurrent neural networks have shown the ability to achieve even state-of-the-art results by automatically learning features from raw sensor data. Based on accurate model prediction capabilities, we can implement specific applications in detecting health, sports, medical care, etc.

Introduction

Project Aims

Human activity recognition[1](or HAR for short) is a broad field of research involving the recognition of specific movements or actions of people based on sensor data. It can be used worldwide using capture devices such as smartphones and cameras and is capable of capturing human activity data. Along with the steady growth of electronic devices and their applications, advances in artificial intelligence have revolutionized the ability to extract deep hidden information for accurate detection and its interpretation, and by combining different deep learning models such as CNN[2], RNN (LSTM)[3], etc., the accuracy of recognition can be improved substantially and efficiently. With the rapid development of electronic devices, human activity recognition has become more and more widespread and various applications[4] have surfaced, such as pedometers, car accident detection, water fall detection, etc.X

The project uses CNN as the deep learning model with outputs for 14 different types of human activity categories: sitting, sitting bent forward, sitting bent backward, standing, walking, running, lying down on back, lying down on left, lying down on right, lying down on stomach, movement, desk work, descending stairs, and ascending stairs. After being trained on the 2021 and 2022 PDIoT cleaned datasets, the models will be imported into an Android backend app, which will be connected to sensors to collect human activity data in real time and use the models to make predictions accordingly.

In the Android back-end app, we implemented database access, where users can register an account to store the corresponding human activity information data into the database, and also query the data according to the date. Also, in the app, we have implemented the basic function of pedometer, which allows real-time prediction of pedometer at the level of having two seconds delay. The accurate classification of these fourteen types of human activity categories can lay a good identification for various derivative applications in the future. For example, we can use this project as a prototype to create a child sitting health application that uses sensors to collect real-time data, count the sum time of different human activity categories in a certain time period, and give adjustment suggestions to the wearer accordingly.

Brief description of the method adopted

Data Fusion[5]

Respeck[6] is a compact Inertial Motion Unit[7] (IMU) device with a 3-axis accelerometer and gyroscope sensor for physical activity monitoring. Since Respect is worn under the wearer's ribs, when the wearer is sitting or standing, there is not much difference in the direction indicated by the gyroscope in the three-dimensional space, so it is impossible to simply use respect to distinguish between standing and sitting. Similarly, Thingy[8] is an off-the-shelf IMU prototyping platform produced by Nordic Semiconductor with 3-axis accelerometer, gyroscope and magnetometer sensors. It will be worn in the wearer's trouser pocket. When sitting and standing, thinyg's gyroscope will point in two different directions to distinguish between sitting and standing. We train two models with the data of respeck and thingy respectively, and do the logic processing in the back-end.

Convolutional Neural Network

A convolutional neural network (CNN) is a deep learning algorithm that takes an input image, assigns importance (learnable weights and biases) to aspects/objects in the image, and is able to distinguish one from the other. Compared to other classification algorithms, CNN requires much less pre-processing. Although the filters are designed manually in the original approach, with enough training, the CNN has the ability to learn these filters/features. the CNN can receive one-dimensional temporal information as input, in addition to images. In 1D CNN[9], the kernel slides along one dimension while the data is collected from the accelerometer and gyroscope on the wearer. The data can represent the acceleration or gyroscope values for all 3 axes. 1D CNN can perform activity recognition tasks based on accelerometer data, such as whether a person is standing, walking, jumping, etc. The data has 2 dimensions. The data has 2 dimensions. The first dimension is the time step, and the others are the acceleration values of the 3 axes.

Tensorflow Lite

TensorFlow Lite[10] is a set of tools to help developers run models on mobile, embedded, and IoT devices to enable device-side machine learning.

1. Optimized for device-side machine learning by addressing 5 constraints: latency (data does not need to travel to and from the server), privacy (no personal data leaves the device), connectivity (no need to connect to the Internet), size (reduced model and binary file size), and power consumption (efficient inference and no network connection).

2. Supports multiple platforms, spanning Android and iOS devices, embedded Linux, and microcontrollers.

3. Support for multiple languages, including Java, Swift, Objective-C, C++, and Python.

4. High performance with hardware acceleration and model optimization support.

5. Provides end-to-end examples of common machine learning tasks on multiple platforms, such as image classification, object detection, pose estimation, question answering, text classification, and more.

In this project, we take the trained CNN model with TensorFlow Lite (tflite file) as output, build the application on the backend using Android Studio, and change the tflite file to inference mode with real-time data as input.

Back-end multi-threaded operation

(Respeck Thread): The Respeck thread is used to fetch Respeck's 3-axis accelerometer and gyroscope and to synchronise updates to the Respeck Chart. In addition, it needs to transfer Respeck data to the Respeck model. The model thread can use the Respeck model input for the model output.

(Thingy Thread): The Thingy thread is used to fetch Thingy's 3-axis accelerometer and gyroscope. Also, it needs to transfer Thingy data to the Thingy model. The model thread can use the Thingy model input for the model output.

(Model Thread): The Model thread is used to fetch the Thingy data and Respeck data. The data is inserted into the appropriate model and the results of the model are obtained. Moreover, the human activity and the corresponding model output probabilities are then mapped together and sorted in reverse order based on the model outputs.

List the physical activities used in the classification

Respeck is worn below the rib cage, always parallel to the wearer's body. Thingy is worn in the right trouser pocket with the front fixed and parallel to the body

Sitting: Sitting is a basic movement and resting position in which the hips touch the ground or a horizontal surface (such as a chair seat) and the angle between the body and the ground is 90 degrees.

Sitting bent forward: On the basis of the standard sitting posture, the upper body is tilted forward 45 degrees, and the body's center of gravity is located on both feet

Sitting bend backward: On the basis of the standard sitting posture, the upper body is tilted back 45 degrees, and the body's center of gravity is located at the back

Standing: Standing is a human posture that keeps the body upright, with the body at an angle of approximately 90 degrees to the ground. Although a person in this posture appears to be stationary, the center of gravity will sway slightly back and forth in the sagittal plane with the ankles as the center of gravity.

Walking: Walking is typically slower than running and other gaits. The speed is about 5 km per hour.

Running: It is defined athletically as a pace where sometimes both feet do not touch the ground at the same time. The speed is about 10-12 km per hour

Lying down on back: The wearer will lie flat on a horizontal plane, the front of the Thingy will be parallel to the plane

Lying down on left: The wearer's body is perpendicular to the lying plane and face to the left, the front of Thingy is perpendicular to the ground.

Lying down on right: The wearer's body is perpendicular to the lying plane and face to the right, the front of Thingy is perpendicular to the ground.

Lying down on stomach: The wearer lies face down on a horizontal surface, and the front of the Thingy will face the surface

Movement: Movement consists of a series of actions including sudden turns, bending down, getting up from chairs, sitting back to chairs

Desk work: Desk work includes typing on the keyboard, writing with a pen, turning the body slightly to pick up things, leaning forward slightly.

Descending stairs: Go down the stairs at a normal speed, one step at a time, the Thingy will be perpendicular to the body when the leg is raised, and the Thingy will be parallel to the body when the leg is retracted

Ascending stairs: Go up the stairs at a normal speed, one step at a time, the Thingy will be perpendicular to the body when the leg is raised, and the Thingy will be parallel to the body when the leg is retracted.

Summary of results

For the CNN-4 Respeck model, its Average F1-score reached 0.992; for the combination of CNN-2 Thingy model and CNN-14 Respeck model, its Average F1-score reached 0.965. Recorded according to the front-end test In the case of 14 types of human activities, the resolution accuracy is about 0.94.

For the PDIoT application, the memory footprint is about 15%, and the latency is two seconds to do every prediction because it requires two seconds to collect the data. The power consumption is about 5%. The application has a storage footprint of 33.65 MB, and the storage will increase as historical data is stored.

Literature Survey

Human Activity Recognition[1][11] or HAR is the process of interpreting human motion using computer and machine vision techniques. Human motion can be interpreted as an activity, gesture or behavior recorded by a sensor. The motion data is then converted into action commands for computer execution and analysis of human activity recognition codes. It has been proven to play an important role in applications in different fields including sports training, safety, recreation, environmental assisted living, and health monitoring and management.

Typically, HAR is divided into two categories[12] based on the type of data, such as vision-based HAR and sensor-based HAR. vision-based technologies analyze camera data into video or image formats, while sensor-based systems interpret sensor (accelerometer[13], gyroscope[14], radar, and magnetometer[15]) data into time-series form. Among the available sensors, accelerometers are mainly used for HAR due to their low cost, small size and portability.

Machine learning algorithms for the sensor-based HAR time series classification problem are divided into three main categories: discriminative, generative, and hybrid.[16] (See Figure 1)

The first challenge for wearable IMU[17] HAR is feature extraction. Activity recognition is a classification task, so it has a common challenge with other classification problems, namely feature extraction. For sensor-based activity recognition, feature extraction is more difficult because there is inter-activity similarity. Different activities may have similar features (e.g., walking and running). Therefore, it is difficult to generate distinguishable features to uniquely represent activities. Many previous works have used machine learning methods in human activity recognition. They rely heavily on feature extraction techniques, including time-frequency transformations[18], statistical methods, and symbolic representations. However, the extracted features are carefully designed and heuristic. There are no general or systematic feature extraction methods to effectively capture distinguishable features of human activities.

The methods discussed above typically rely heavily on manual feature extraction. This is limited by domain expertise, time-consuming and resource-intensive. Deep learning methods can be used to solve such problems. Recent trends in sensor HAR recognition show that deep learning methods such as convolutional neural networks and recurrent neural networks provide excellent results on challenging activity recognition tasks with nominal feature engineering, rather than using time-consuming manual feature learning on raw data. One of the main ben-

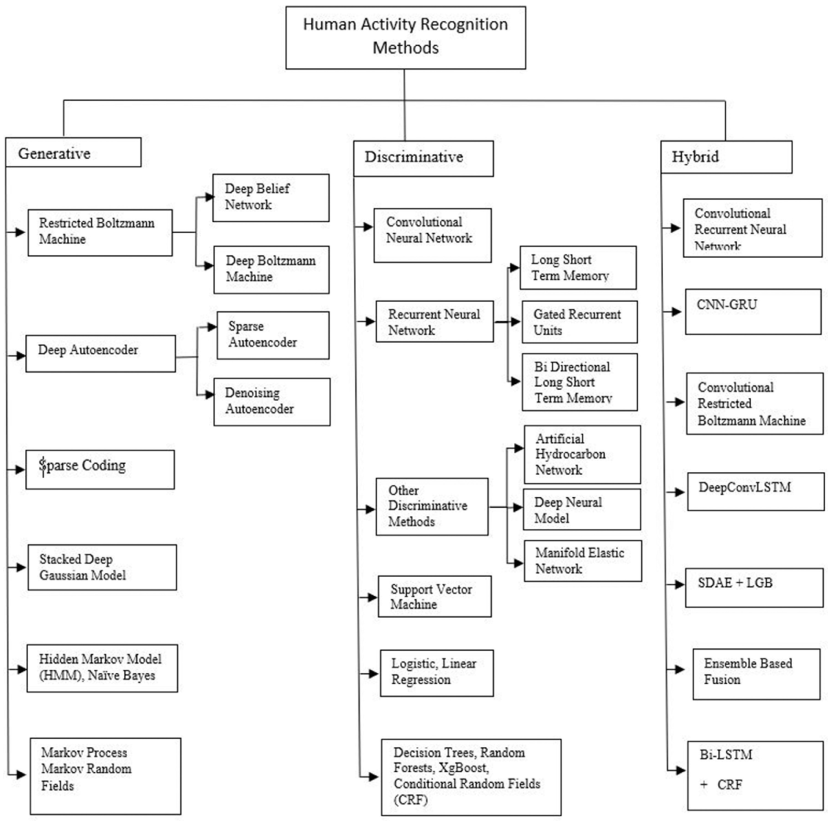

Human Activity Recognition Methods
Generative
Restricted Boltzmann Machine
Deep Belief Network
Deep Boltzmann Machine
Deep Autoencoder
Sparse Autoencoder
Denoising Autoencoder
Sparse Coding
Stacked Deep Gaussian Model
Hidden Markov Model (HMM), Naïve Bayes
Markov Process Markov Random Fields
Discriminative
Convolutional Neural Network
Recurrent Neural Network
Long Short Term Memory
Gated Recurrent Units
Bi Directional Long Short Term Memory
Other Discriminative Methods
Artificial Hydrocarbon Network
Deep Neural Model
Manifold Elastic Network
Support Vector Machine
Logistic, Linear Regression
Decision Trees, Random Forests, XgBoost, Conditional Random Fields (CRF)
Hybrid
Convolutional Recurrent Neural Network
CNN-GRU
Convolutional Restricted Boltzmann Machine
DeepConvLSTM
SDAE + LGB
Ensemble Based Fusion
Bi-LSTM + CRF

Figure 1: Different Categories of HAR

efits of using deep learning methods is the imposed impact of automatic feature learning using deep neural networks to extract temporal features[16]. Various deep learning methods have been applied for temporal information extraction including RNN, LSTM and GRU[19].CNN is another very widely used deep learning method for extracting temporal features.CNN does not require sliding windows to segment time series data. Convolutional operations with small kernels are applied directly to the temporal dimension of the sensor signal so that local temporal dependencies can be captured[20]. Both 1D CNNs and 2D CNNs have been effectively used by researchers to identify HAR activity from sensors.

Long short-term memory (LSTM)[21] units to overcome this problem are typically used to build RNNs for temporal feature extraction. when processing sequential data, the depth of an effective LSTM-based RNN needs to be at least two. Since sensor signals are continuous streams of data, sliding windows are usually used to split the raw data into individual segments, each of which is an input to the RNN unit. Bidirectional LSTM (Bi-LSTM)[22] structures with two traditional LSTM layers for extracting temporal dynamics from the forward and reverse directions are important variants of RNNs in various fields including human activity recognition

Unlike RNNs, temporal CNNs[2] do not require a sliding window[23] for segmenting streaming data. Convolutional operations with small kernels are applied directly to the temporal dimension of the sensor signal, so local temporal dependencies can be captured. Some works use one-dimensional convolution for a single univariate time series signal to extract temporal features. When there are multiple sensors or multiple axes, multivariate time series will be generated and thus a separate 1D convolution needs to be applied. Ha and Choi[24] proposed a new CNN structure with modality-specific 1D CNNs for learning modality-specific temporal features. With the development of CNNs, other types of CNN variants have been considered for effective embedding of temporal features.

The development of deep hybrid models to explore different views of temporal dynamics is another attractive trend in the human activity recognition community. Given the advantages of CNN and RNN, Ordóñez and Roggen[25] proposed to combine CNN and LSTM for local and global temporal feature extraction. the LSTM network implemented by Ronald and Dong[26] boasts a 99% correct rate on the in-house dataset.

In this work, in order to effectively recognize complex human activities such as eating, bathing, basketball, etc., it is necessary to identify spatial and temporal features from data collected using raw sensors of different modalities. This is achieved carefully using a mixture of convolutional and gated recurrent unit (GRU) neural networks[27].

Methodology

Description of the system and its implementation

The system is consists of three components: HAR model, Android application back-end system and front-end system. In addition, the user interface was designed with minimalism in mind, as our product was focused on the Human Activity Prediction functionality and was aimed at the general user. Also, the back-end system was used as the cornerstone of the entire Android application, it was used to connect the user-interface display and HAR model interactively process. The image is shown in Figure 2. The development tools was Android Studio and the testing was developed in the Android Studio as well.

Our backend system has three functionality: registration/login component, HAR model processing and prediction, history data visualisation and storing.

The registration/login component is based on a local database and its implementation. As the app is a demo, user register and login data are stored in the local device instead of server. The back-end system reads the user input and writes it to the local database to complete the user registration. The login operation works by the system reading the database to verify whether the email is registered and whether the corresponding password is correct.

HAR model processing and prediction: the HAR model is processed and run locally. We have not implemented cloud model processing and computing. The processing and prediction of the HAR model includes a number of steps. Firstly, the application receive and collect the data from Respeck and Thingy sensor. The data is processed into the above mentioned sliding window to be used as input. Then, the input will be fed into the TensorFlow Lite Model to obtain the prediction results. After each 2 seconds, prediction result will be updated. The final prediction results are saved to Table 2 in the local database in Figure 2

Viewing History subsystem is a query and data visualisation of Table2 in the database in Figure 2. The system queries all the prediction results in the database based on the date. The proportion of time spent on each behaviour/activities is calculated to output a visualization diagram of each behaviour/activities. Our step account calculates how many steps the user has taken by query how many rows in the database has walking or running. One row represent 1 second or one step. If one of the predictions ends with walking or running, we determine that the user took a step at that moment. Sum of these moments is the output result of step count system. However, step count system is too weakly implemented and not very accurate.

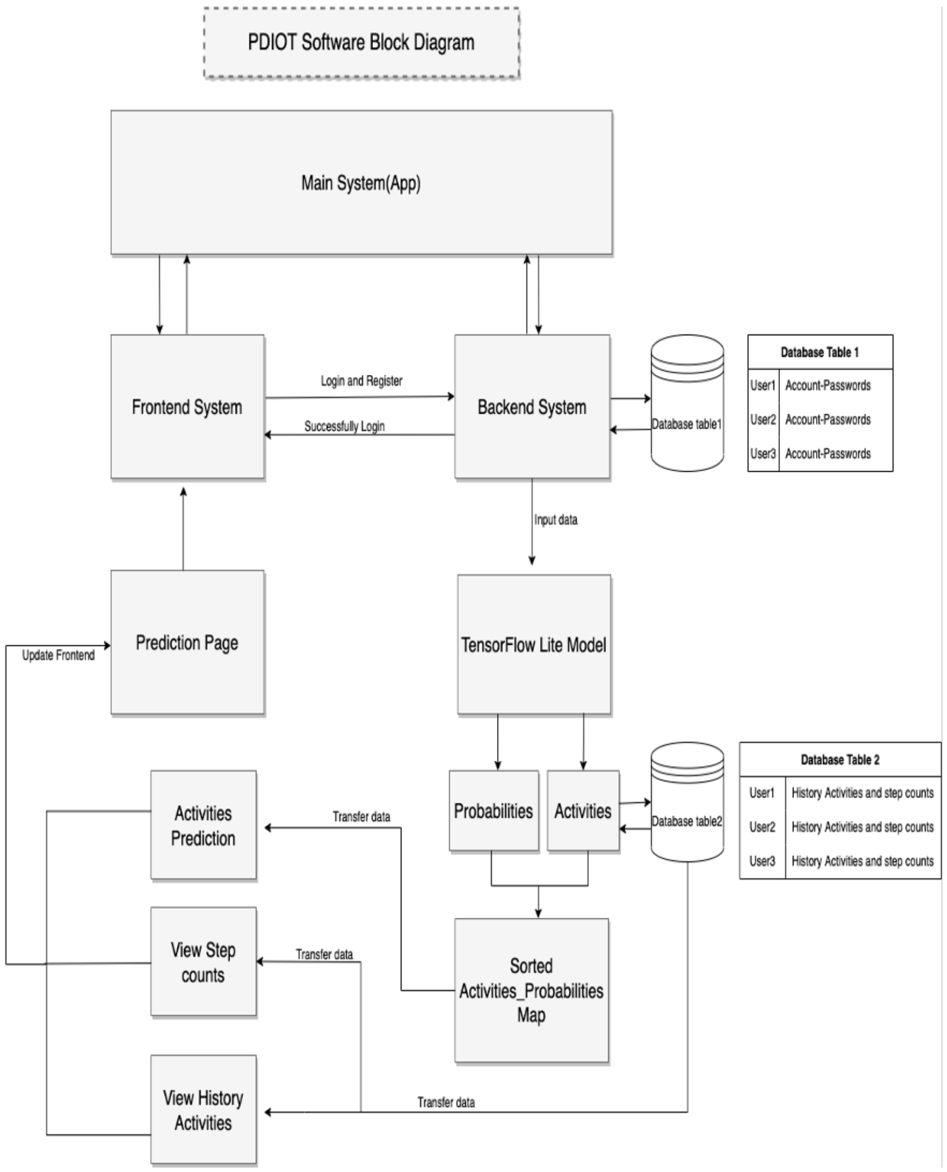


Figure 2: Android application UML image

Hardware and firmware

We used Respeck and Thingy as our project hardware and firmware.

Respeck: A compact Inertial Motion Unit (IMU) device, with a 3-axis accelerometer and gyroscope sensor for physical activity monitoring. The Respeck sensor, worn on the lower left ribcage, sampling accelerometer and gyroscope data at 25Hz.

Thingy: An off-the-shelf IMU prototyping platform produced by Nordic Semiconductor with 3-axis accelerometer, gyroscope and magnetometer sensors. The Thingy sensor, worn in the front right pocket of your trousers, sampling accelerometer, gyroscope and magnetometer data at 25Hz.

Besides these, we didn't make any modifications to Respeck or Thingy.

Wireless communication

At a specific sampling frequency (25Hz), the PDIOT app and the sensors communicate wirelessly via Bluetooth. The Android app will connect to Respeck and Thingy respectively. At first, the PDIOT app needs to set a specific sampling frequency of 25Hz.

At a specific sampling frequency (25Hz), the PDIOT app and the Thingy sensor communicate wirelessly via Bluetooth. If the user's smartphone supports NFC pairing,, then user can simply tap the phone on the front side of the Thingy (here shown without the rubber case) to get the Thingy ID autocompleted in the corresponding field. Otherwise, the user has to manually enter the Thingy ID (MAC address), which is made up of 12 letters or numbers. The Thingy ID can be found on the bottom edge.

For Thingy, the different light colours represent the different states of the sensor.

- Blue light $\rightarrow$ sensor ON and NOT CONNECTED
- Green light $\rightarrow$ sensor ON and CONNECTED
- No light $\rightarrow$ sensor OFF

When the connection is successful, the Thingy will turn from a blue light to a green light. If the connection is not successful, the Thingy will remain Blue light. If you have any connection issues, you can click on Restart connection to restart the Bluetooth service forcefully.

At a specific sampling frequency (25Hz), the PDIOT app and the Respeck sensor communicate wirelessly via Bluetooth. If the user's smartphone supports NFC pairing, then the user can simply tap the phone on the white surface of the Respeck to automatically fill in the Respeck ID in the appropriate area. Also, the user can scan the Respeck's QR code. Each Respeck should have a QR code printed on its back. Once the QR code has been scanned, the Respeck ID will be automatically filled in the corresponding field. In addition to this, the user can manually enter the Respeck ID (MAC address), which is made up of 12 letters or numbers. The Respeck ID can be found on the back.

For Respeck, the different light colours represent the different states of the sensor.

- Green light blink $\rightarrow$ sensor ON and NOT CONNECTED
- Blue light $\rightarrow$ sensor ON and CONNECTED
- Red light $\rightarrow$ sensor ON and DISCONNECTED

When the connection is successful, the Thingy will turn from a green light to a blue light. If the connection is not successful, the Thingy will remain green light. If you have any connection issues, you can click on Restart connection to restart the Bluetooth service forcefully.

Therefore, when both Respeck and Thingy are successfully connected, the Respeck's light should be blue and the Thingy's light should be green. And the connections are successful.

Machine Learning Methods for activity recognition

Our Algorithm choice for human activity recognition is Convolutional neural network 1D. Convolutional neural network is a feed-forward neural network, but it reduces the amount of computation and the number of parameters. Convolutional neural networks consist of these types of layers: input layer, convolutional layer, Activation layer, pooling (Pooling) layer and fully connected layer (fully connected layer is the same as in a regular neural network). By stacking these layers together, a complete convolutional neural network can be constructed. In practical applications, the convolutional layer and the ReLU layer are often referred to together as the convolutional layer, so the convolutional layer undergoes a convolutional operation that is also subject to an activation function. [28] The following figure 3 illustrates the architecture of our network.

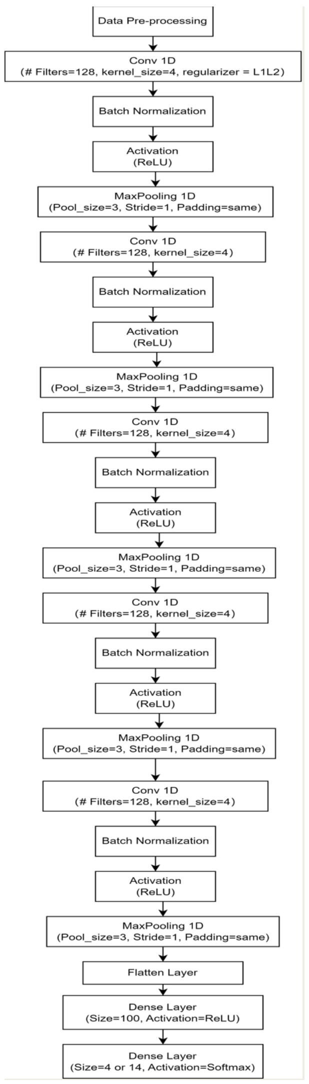

Data Pre-processing
Conv 1D
(# Filters=128, kernel_size=4, regularizer = L1L2)
Batch Normalization
Activation
(ReLU)
MaxPooling 1D
(Pool_size=3, Stride=1, Padding=same)
Conv 1D
(# Filters=128, kernel_size=4)
Batch Normalization
Activation
(ReLU)
MaxPooling 1D
(Pool_size=3, Stride=1, Padding=same)
Conv 1D
(# Filters=128, kernel_size=4)
Batch Normalization
Activation
(ReLU)
MaxPooling 1D
(Pool_size=3, Stride=1, Padding=same)
Conv 1D
(# Filters=128, kernel_size=4)
Batch Normalization
Activation
(ReLU)
MaxPooling 1D
(Pool_size=3, Stride=1, Padding=same)
Conv 1D
(# Filters=128, kernel_size=4)
Batch Normalization
Activation
(ReLU)
MaxPooling 1D
(Pool_size=3, Stride=1, Padding=same)
Flatten Layer
Dense Layer
(Size=100, Activation=ReLU)
Dense Layer
(Size=4 or 14, Activation=Softmax)

Figure 3: CNN Structure

The convolutional layer is responsible for extracting features, the pooling layer is responsible for feature selection, and the fully-connected layer is responsible for classification. Each convolutional layer in a convolutional neural network consists of a number of convolutional filters, and the parameters of each convolutional unit are optimized by a back-propagation algorithm. The purpose of the convolutional operation is to extract different features from the input. The first convolutional layer may only extract some low-level features and more layers of the network can iteratively extract more complex features from the low-level features. In our convolutional layer, we use 128 kernels and kernel size 4. Then we use ReLu as activation function. We also used batch normalization to accelerate the coverage of network. After each convolutional layer we use a Max-pooling layer to implement downsampling. Max-pooling is done by filtering to keep only the largest values on the feature map. The pooling layer further reduces the spatial size of the data, and therefore the number of parameters and the computational effort, which also controls overfitting to some extent. Finally, the extract features will become flatten data(1D data) in fully-connected layer. We then input this data into the Softmax layer to implement the classification of the human activities.

Our data are the continuous changes of the six features of the accelerometer (x,y,z) and gyroscope (x,y,z) from sensors in a given time. For data-preprocessing, we split data into sliding window. The sliding window approach is one of the most efficient ways to process Human Activity Recognition data. Sensor data comes in the form of Time Series. One single datapoint is not enough to represent an activity, we need a larger snapshot of the signal for that. Because the data is 1D Time Series information, it is more efficient to use 1D convolution here.[29] One-dimensional CNN can perform activity recognition tasks based on accelerometer and gyroscope data, such as human stance, walking, jumping, etc. This data has 2 dimensions. The first dimension is the time step and the other dimension is the acceleration value on the 3 axes. The figure 4 below illustrates how the kernel moves over the accelerometer data. Each row represents the time series acceleration for a certain axis. The kernel can only move along one dimension of the time axis.

We built models for both Respeck and Thingy's data separately. Both models use CNN as the classification algorithm.

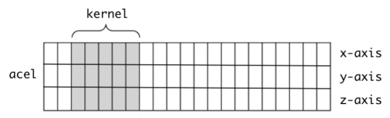
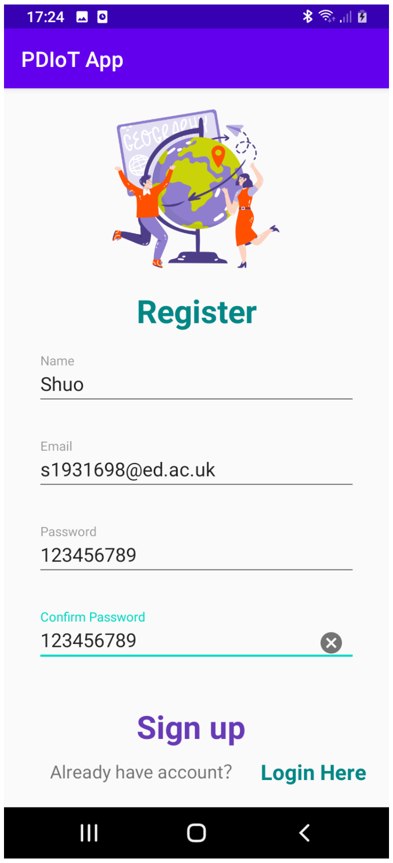


Figure 4: CNN 1D Exmaple

Mobile application

The name of the APP is PDIoT APP. The memory space occupied by the whole application is about 35.5-36m. We have implemented many features on our mobile applications. For example, Registration, Login, Prediction, Step counting and Viewing history activities. These features are developed in parallel front and back end. Each function and interface will be described one by one below.

Registration


17:24

PDIoT App

Register

Name
Shuo

Email
s1931698@ed.ac.uk

Password
123456789

Confirm Password
123456789

Figure 5: Registration Page

Figure 5 shows the Registration page of application. Based on feedback from peer review we have simplified the information required for registration. The registration only requires new users to enter their name and email address. And New users will need to set a password that is longer than 8 digits. The system will judge the user's input. System will check if the user input is empty, if the input email is in the correct format and whether the password has more than 8 digits. If there is an input error, a pop-up window will alert users. Based on feedback from peer review, we have optimized the input box. When the user finishes typing Name and hits the next key on the keyboard, user can jump to the next input box. When all the user's input is correct, click Sign Up and the system will save the user's data to the database and register successfully. Because this APP is a demo, the user data will only be stored in the local database. Not the cloud database. Subsequent work can store the user data in the remote server database. The application will then automatically jump to the login page. Login Here texts showed on the register page is also a link. Click it will also jump to the login page.

Login

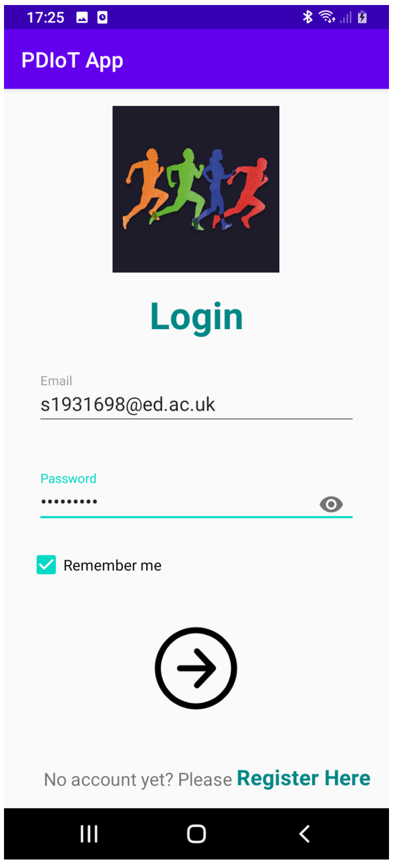
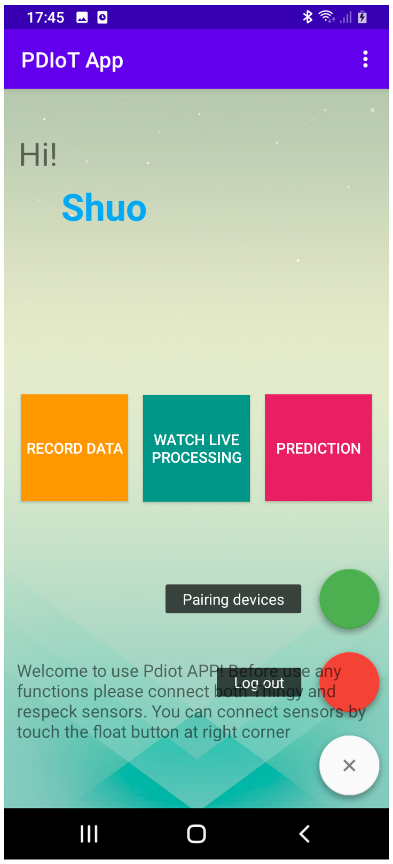


Figure 6: Login Page

After registration, users will come to the login screen as shown in figure 6. Users only need to enter the email address and correct password that they just registered to enter the main interface of the app. The system also makes a judgement on the user's inputs. The system will determine whether the inputs is empty, whether the email address entered exists in the database, and whether the password matches the email address. If there is an input error, a pop-up window will alert users. The arrow icon is the login button. Click on it to log in to the main screen. We have also added the remember me function for the convenience of our users. After restarting the app or logging out, APP can automatically fill in user's email and password without entering them again.

Main Page

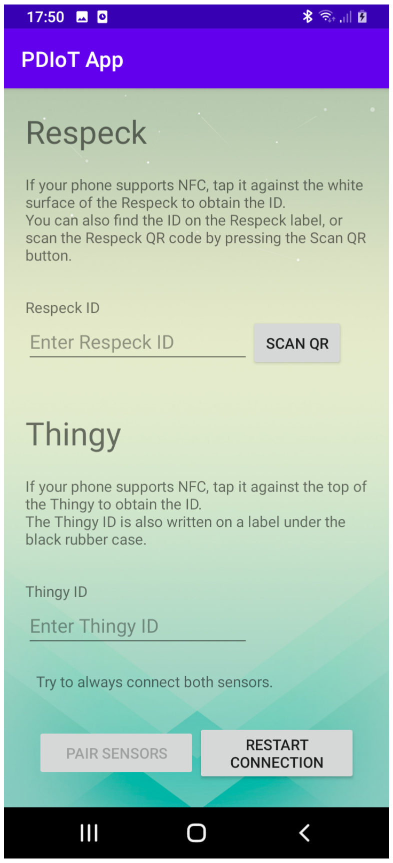


Welcome to use Pdiot APP! Before use any functions please connect both Thingy and respeck sensors. You can connect sensors by touch the float button at right corner

Figure 7: Main Page

As shown in figure 7, The main page is designed to provide the user with a variety of options. The different buttons in the main screen all correspond to different functions. The functions have been named and displayed. Users can jump to the corresponding interface by clicking on the buttons. But before using all functions, user need to connect sensors firstly. According to Peer review feedback, we changed the background of main page so that user can read the texts clearly.

Pairing Sensor


17:50

PDIoT App

Respeck

If your phone supports NFC, tap it against the white surface of the Respeck to obtain the ID.
You can also find the ID on the Respeck label, or scan the Respeck QR code by pressing the Scan QR button.

Respeck ID

Enter Respeck ID

Thingy

If your phone supports NFC, tap it against the top of the Thingy to obtain the ID.
The Thingy ID is also written on a label under the black rubber case.

Thingy ID

Enter Thingy ID

Try to always connect both sensors.

Figure 8: Pair Sensor Page

Figure 8 shows the Pairing Devices page. There is instruction on how to connect sensors in the pairing sensor page. If the user's phone supports NFC, the user can approach the sensor with the phone and the APP system will automatically connect the sensor. If NFC is not supported, the APP will pop up a text to remind the user.Moreover, user can choose to enter the sensor ID manually or use the SCAN QR function. Once click SCAN QR button, App will turn on the phone's camera to scan the QR code on the surface of the sensor. After a valid QR code has been scanned, the app will automatically fill in the input box with the sensor id. Then user can click the pair sensor button and system will connect to the corresponding sensors. User only need to pair these sensors once. Their IDs will be remembered by the app whenever user start it again. The App also has a restart connection function. Clicking on the button will reconnect the sensor once more. After a successful connection, user can use the other functions of the App.

Watch Live Processing

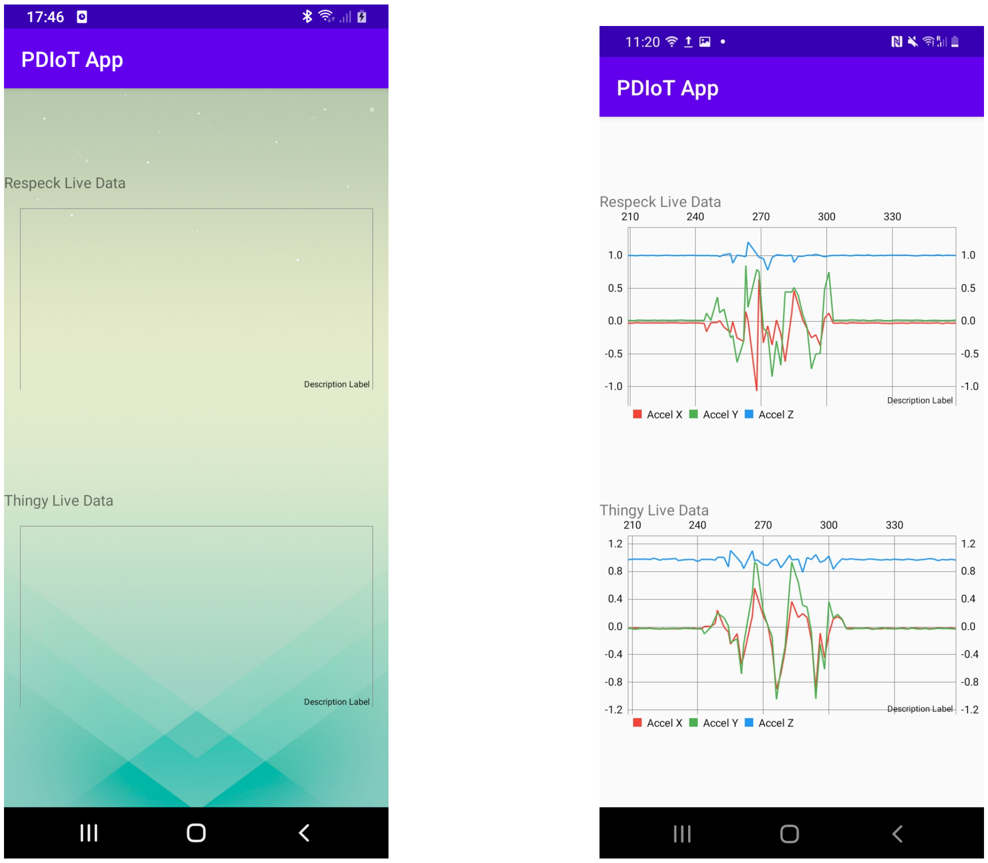


Figure 9: Watch Live Page and Live data example

Figure 9 left is the interface of watch live activity. You can view incoming data from both sensors in the "Watch live processing" activity. This will show you two live graphs of the accelerometer data from the Respeck (top) and Thingy (bottom). Both sensors should run at 25Hz. Figure 9 right is the example of what the live data diagram looks like. The red colour represent Accelerator X-axis, green colour represent Accelerator Y-axis, blue colour represent Accelerator Z-axis.

Record data

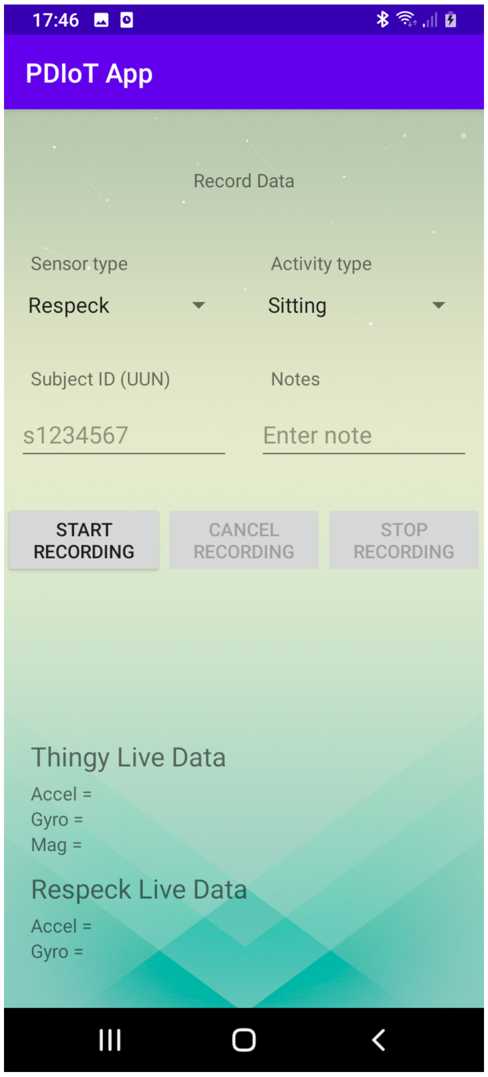


Figure 10: Record Data Page

Figure 10 illustrates the record data interface. You can record data in the Record Data activity. Choose the appropriate Sensor type and Activity and please use the university student number as the subject ID. You can enter any additional notes you have about the upcoming recording.

You will be able to verify that your sensors are running as expected by watching the Live Data fields at the bottom of the screen.Hit Start Recording when you are all set up. When you are done with a recording, hit Stop Recording. If something goes wrong during the recording you can cancel it by pressing the Cancel Recording button. Recorded files are saved on the phone's internal memory as csv files, on the path: Android → app → data → com.specknet.pdiotapp → files → Filename.csv

Prediction

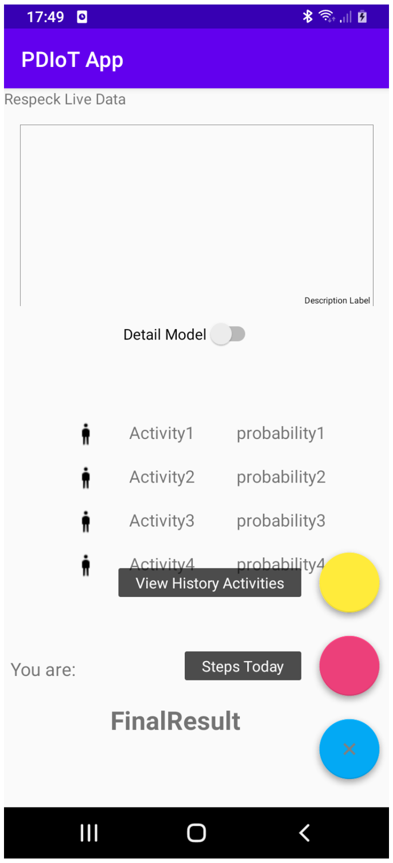
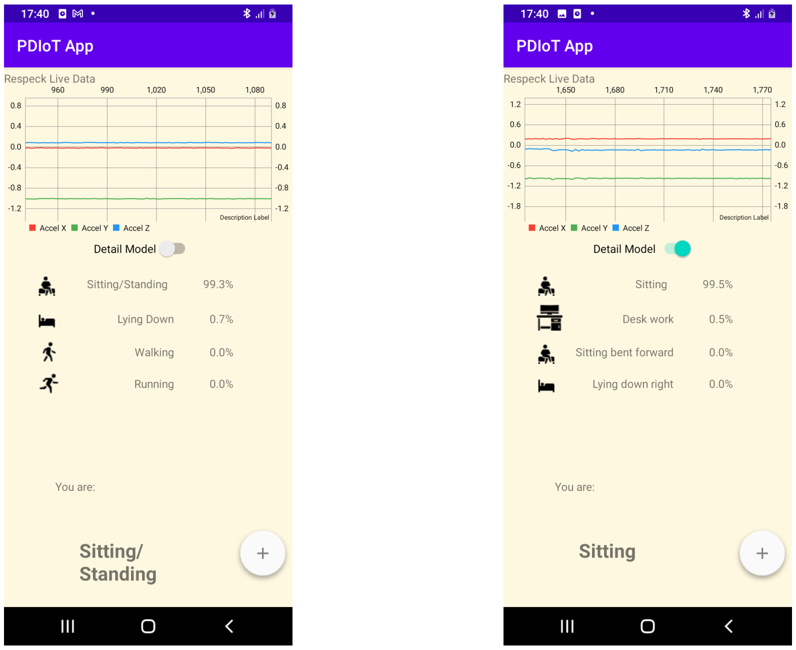


Figure 11: Prediction Page

Once user enter the prediction interface as shown in figure 11. App System will start to do the prediction. Respeck Live data graph will display the real-time data from sensor. The TextView below lists the four most probable actions and their corresponding probabilities. The ImageView will also change accordingly to the behaviour. The final prediction result will be displayed at the bottom. There is a delay of 1-2s when predicting. If the user keeps a behaviour, the prediction will be stable when the behaviour is stable. The results of the predictions are divided into two classes, one general class includes 4 activities (running, walking, sitting/standing and lying),one detailed class include 14 activities.(Sitting, Walking at normal speed, Lying down on back, Desk work, Sitting bent forward, Sitting bent backward, Lying down right, Lying down left, Lying down on stomach, Movement, Standing, Running, Climbing stairs, Descending stairs). A switch under live data graph for user to choose the mode. When App in default mode, it will show the result in general classes with 4 activities and use corresponding machine learning model. When App in detail mode, it will show the result in 14 more detailed activities and use other machine learning model for 14 classes. The predicted results will be saved in a local database and can be viewed again by the user using the view history activity.



Sitting/Standing	99.3%
Lying Down	0.7%
Walking	0.0%
Running	0.0%

Sitting/ Standing

Sitting	99.5%
Desk work	0.5%
Sitting bent forward	0.0%
Lying down right	0.0%

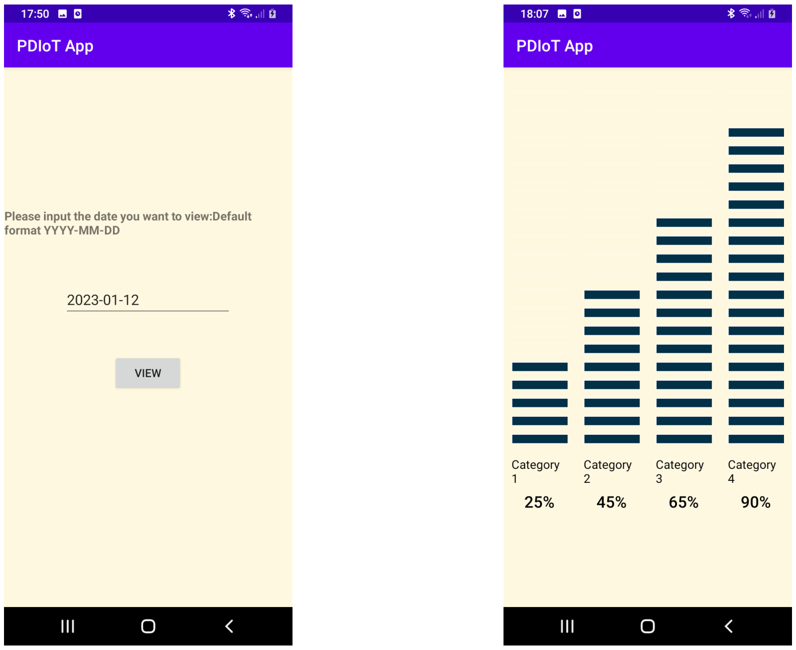
Sitting

Figure 12: Prediction result example

Figure 12 shows how prediction works. Left one is default mode,right one is detailed mode. There is a float button menu at the right corner of the interface. By clicking on the corresponding button, the user can use more extended functions(View History Activities and Steps Today).

The peer review's feedback states that the process crashes when using our prediction function. We have now fixed this bug, which was caused by the front-end not being properly initialised

View History Activities



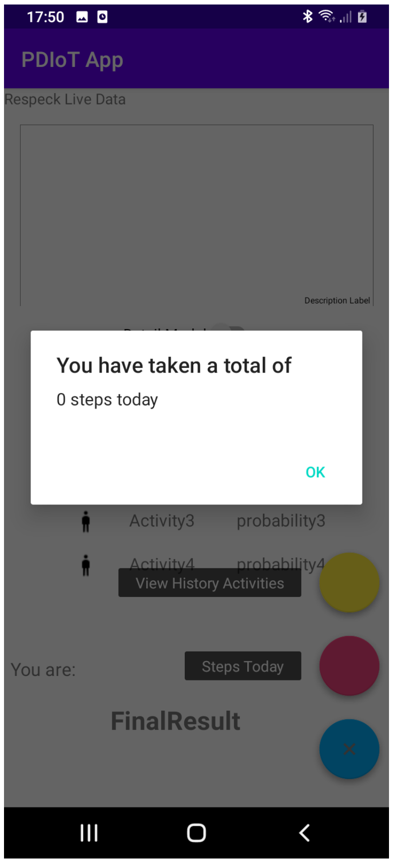
Please input the date you want to view:Default format YYYY-MM-DD

2023-01-12

Figure 13: View History page Example

Figure 13 shows the view history interface. User need to input the date in correct format(YYYY-MM-DD). We have taken the advice given in the feedback. When the user opens the view history interface, the input box is automatically filled in with the date of the day. This is more convenient for user. Once user have filled in the date, click on the view button and the page will change to a distribution graph of the day's activities like shown in figure 13.

Steps Today


You have taken a total of

0 steps today

Figure 14: Steps reminder Window

Once the user clicks on the Steps today button, a dialog box will pop up to show the number of steps for the day as shown in figure 14

Software organisation

The application consists of the front-end development and the back-end development.

The back-end contains initialisation, multi-threaded operations, HAR model processing and interaction with the front-end(User interface).

Initialisation

1. Using the FindViewByID function to locate the corresponding textview, imageview and buttons.
2. Initialising the internal values of textview and imageview to avoid system crashes due to lack of initialisation.
3. Initialising the SQL database by fetching the username, fetching the current date
4. Initialising RespeckCharts.

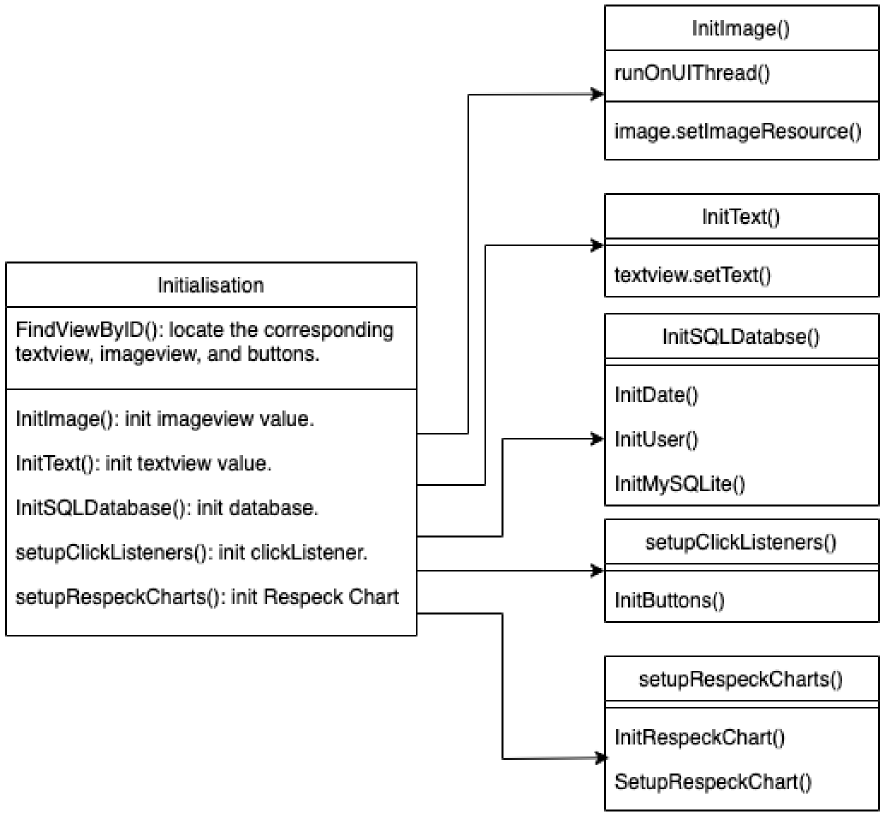

Initialisation
FindViewByID(): locate the corresponding textview, imageview, and buttons.
InitImage(): init imageview value.
InitText(): init textview value.
InitSQLDatabase(): init database.
setupClickListeners(): init clickListener.
setupRespeckCharts(): init Respeck Chart
InitImage()
runOnUIThread()
image.setImageResource()
InitText()
textview.setText()
InitSQLDatabse()
InitDate()
InitUser()
InitMySQLite()
setupClickListeners()
InitButtons()
setupRespeckCharts()
InitRespeckChart()
SetupRespeckChart()

Figure 15: Initialisation functionality Image

Multi-threaded operations

1. (Respeck Thread): The Respeck thread is used to fetch Respeck's 3-axis accelerometer and gyroscope and to synchronise updates to the Respeck Chart. In addition, it needs to transfer Respeck data to the Respeck model. The model thread can use the Respeck model input for the model output.

2. (Thingy Thread): The Thingy thread is used to fetch Thingy's 3-axis accelerometer and gyroscope. Also, it needs to transfer Thingy data to the Thingy model. The model thread can use the Thingy model input for the model output.
3. (Model Thread): The Model thread is used to fetch the Thingy data and Respeck data. The data is inserted into the appropriate model and the results of the model are obtained. Moreover, the human activity and the corresponding model output probabilities are then mapped together and sorted in reverse order based on the model outputs.

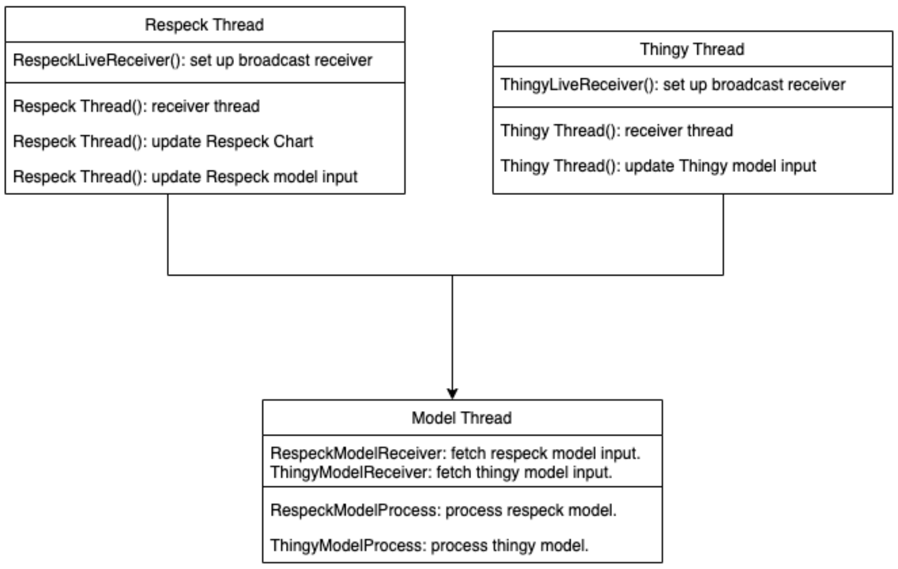

Respeck Thread
RespeckLiveReceiver(): set up broadcast receiver
Respeck Thread(): receiver thread
Respeck Thread(): update Respeck Chart
Respeck Thread(): update Respeck model input
Thingy Thread
ThingyLiveReceiver(): set up broadcast receiver
Thingy Thread(): receiver thread
Thingy Thread(): update Thingy model input
Model Thread
RespeckModelReceiver: fetch respeck model input.
ThingyModelReceiver: fetch thingy model input.
RespeckModelProcess: process respeck model.
ThingyModelProcess: process thingy model.

Figure 16: Multi-threaded operations functionality Image

HAR model processing

1. Fetching the data from Respeck and to fill the Respeck data list. Fetching the data from Thingy and to fill the Thingy data list as well.
2. Once the Respeck data list satisfies the Respeck model inputs, the data is then imported into the model to obtain the output of the Respeck model. Also, when the Thingy data list satisfies the Thingy model inputs, the data is then imported into the model to obtain the output of the Thingy model.
3. Discussion by scenario: predictions for four categories of human activity or prediction for fourteen categories of human activity.
4. (Prediction for four categories of human activity): When the user choose the concise mode, it represents the model will predict for four categories of human activity(Sitting/Standing, Walking, Running, Lying Down). Our system will invoke the CNN-4 Respeck model only. Because the Respeck model has the strong and reliable performance in predicting four categories of human activity. And the Thingy model can be ignored. This will make the application faster and with less memory expenditure.
5. (Prediction for fourteen categories of human activity): When the user choose the details mode, it represents the model will predict for fourteen categories of human activity(Sitting, Walking at normal speed, Lying down on back, Desk work, Sitting bent

forward, Sitting bent backward, Lying down right, Lying down left, Lying down on stomach, Movement, Standing, Running, Climbing stairs, Descending stairs). Our system will invoke the CNN-14 Respeck model and CNN-2 Thingy model at the same time. Because it is impossible to distinguish whether a person is standing or sitting by the Respeck model alone. The Thingy's two-class model is used to distinguish whether a user is standing or sitting. Based on two models, the system satisfies the need for reliable and accurate predictions of human activities.

6. (Details of the implementation of both models): A tricky approach was devised on the back end to adjust the Respeck model's predictions of user standing and sitting based on Thingy's two-class model's predictions of user standing and sitting. When the user is in detail mode and the user is in the Schrodinger state of standing or sitting, the back-end adjusts the Respeck model outputs of standing and sitting based on the predictions of the Thingy model for standing and sitting.

   Sitting = Sitting(Respeck) + Standing(Respeck) when the Thingy model output is sitting.
   Standing = Standing(Respeck) + Sitting(Respeck) when the Thingy model output is standing.

   In effect, the output of the Thingy model is a judgement on whether to stand or sit. And the final outputs are all based on the Respeck model, as the Respeck model predicts the twelve activities with high quality, while the standing/sitting problem is solved by the Thingy model.

Question: Why can't we simply add the two models output together and divided by two?
Answer: Because it is not fair for all twelve activities except standing and sitting. Simply adding the output of both models would double the probability of standing or sitting, while shrinking the probability of the remaining activities to half their magnitude.

Therefore, when users wish to use concise mode, this system can provide a fast, reliable and accurate prediction result based on a single Respeck model. At the same time, when the user wishes to use a detailed mode to understand the exact activity, this system can provide a reliable and accurate prediction result based on the Respeck model and the Thingy model.

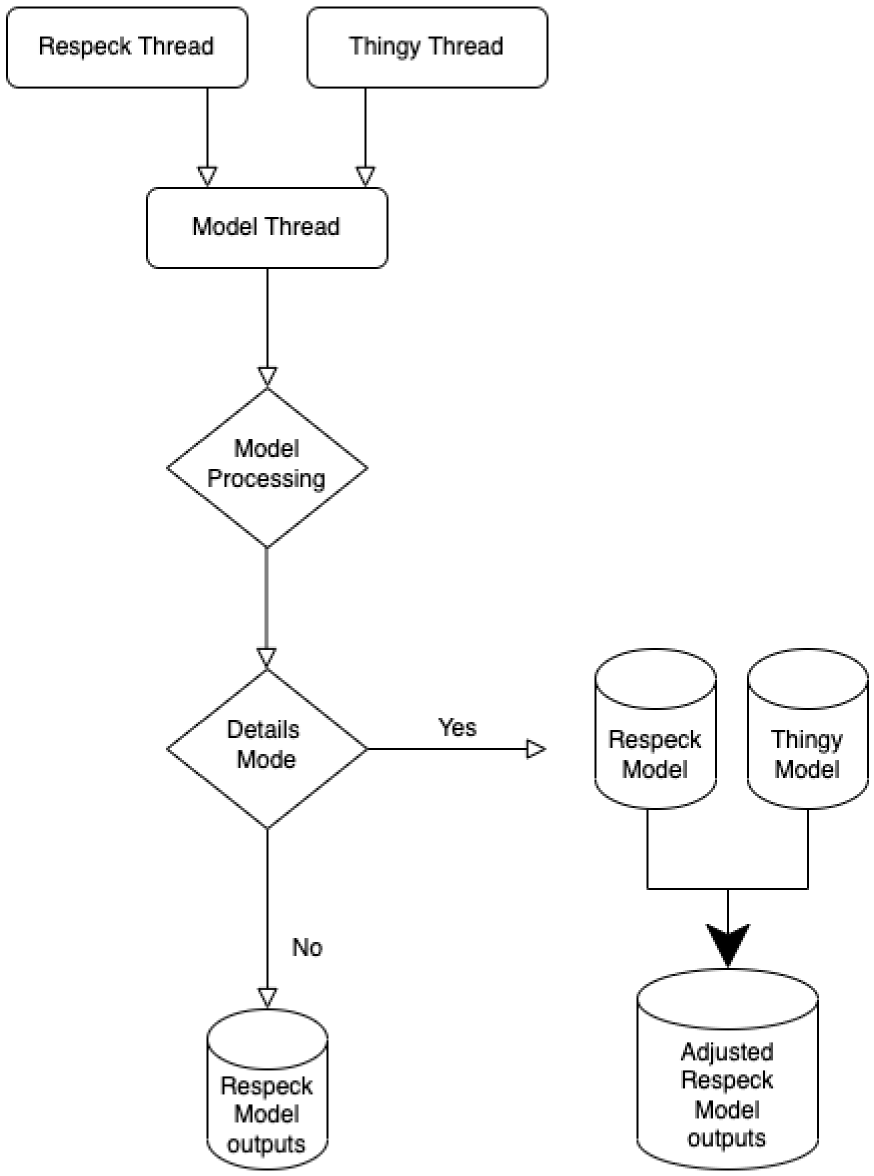


Figure 17: HAR model processing

Interactions with the front-end(User Interface)

1. Based on the output of the Respeck model, a map is created that maps the human activity to the corresponding predicted probabilities.
2. Sorted this map in the reverse order.
3. Got the top four probabilities and their associated activities, updated User Interface textview and imageview.

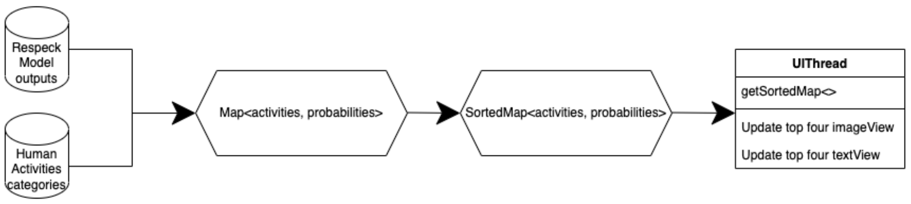
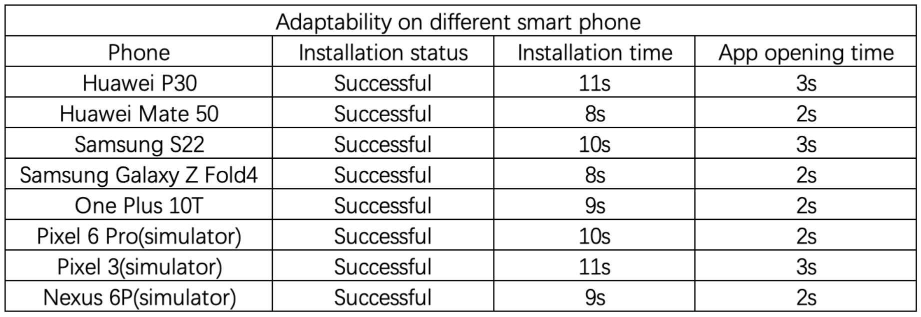

Respeck Model outputs
Human Activities categories
Map<activities, probabilities>
SortedMap<activities, probabilities>
UIThread
getSortedMap<>
Update top four imageView
Update top four textView

Figure 18: Interactions with the front-end(User Interface)

Application Testing

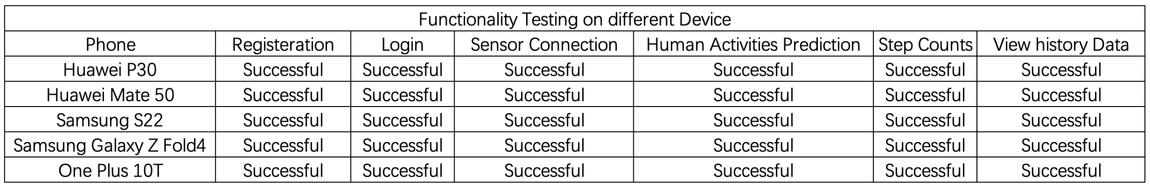
There are a number of different aspects to testing the application. For the installation of the application, we recorded the installation time of the application and the opening time of the application. The adaptability result is shown below.



Adaptability on different smart phone			
Phone	Installation status	Installation time	App opening time
Huawei P30	Successful	11s	3s
Huawei Mate 50	Successful	8s	2s
Samsung S22	Successful	10s	3s
Samsung Galaxy Z Fold4	Successful	8s	2s
One Plus 10T	Successful	9s	2s
Pixel 6 Pro(simulator)	Successful	10s	2s
Pixel 3(simulator)	Successful	11s	3s
Nexus 6P(simulator)	Successful	9s	2s

Figure 19: Adaptability on different smart phone

In addition, it's necessary to test the application functionality including register/login, connect the sensor, human activity prediction, step counts, view history data. The functionality testing result is shown below.



Functionality Testing on different Device						
Phone	Registeration	Login	Sensor Connection	Human Activities Prediction	Step Counts	View history Data
Huawei P30	Successful	Successful	Successful	Successful	Successful	Successful
Huawei Mate 50	Successful	Successful	Successful	Successful	Successful	Successful
Samsung S22	Successful	Successful	Successful	Successful	Successful	Successful
Samsung Galaxy Z Fold4	Successful	Successful	Successful	Successful	Successful	Successful
One Plus 10T	Successful	Successful	Successful	Successful	Successful	Successful

Figure 20: Functionality test on different smart phone

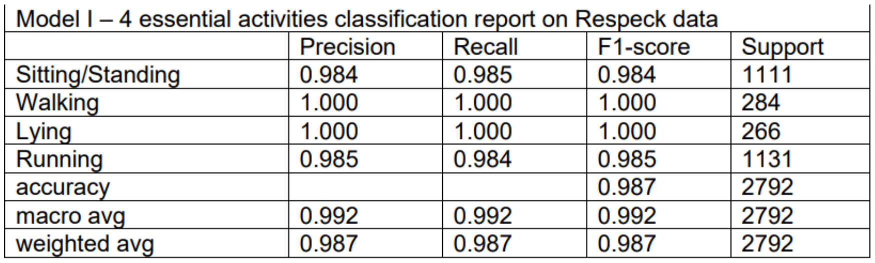
During the development, we encountered an imperceptible little bug that had caused the software to crash several times. The back end needs to initialize every variable in the front-end, including textview, imageview, etc., otherwise, the software will randomly crash. Finally solved by troubleshooting + unit testing

Result

Model Classification Report

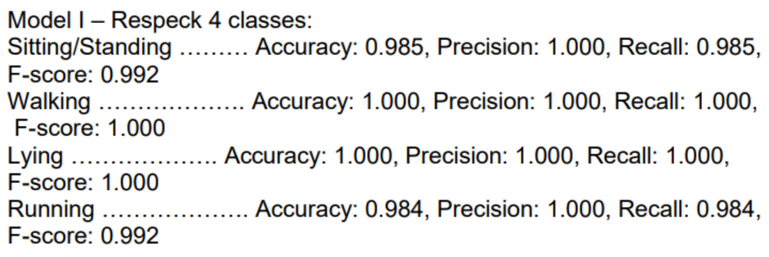
Firstly, we test and evaluate our machine learning model performance by peer review offline. Other group use Google IMU sensor(Respeck and Thingy) test data run our machine learning model to get the classification report.

Here is the result for model1 (CNN-4 respeck) offline testing. This model is for essential 4 activities classification.



Model I – 4 essential activities classification report on Respeck data				
	Precision	Recall	F1-score	Support
Sitting/Standing	0.984	0.985	0.984	1111
Walking	1.000	1.000	1.000	284
Lying	1.000	1.000	1.000	266
Running	0.985	0.984	0.985	1131
accuracy			0.987	2792
macro avg	0.992	0.992	0.992	2792
weighted avg	0.987	0.987	0.987	2792

Figure 21: Model1 Classification Table

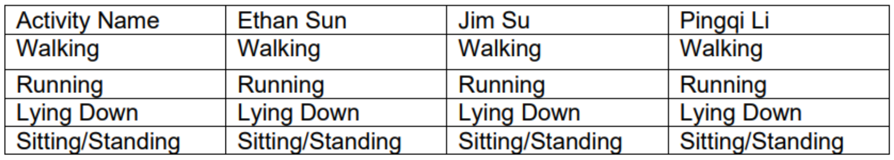


Model I – Respeck 4 classes:
Sitting/Standing ……… Accuracy: 0.985, Precision: 1.000, Recall: 0.985,
F-score: 0.992
Walking ………………. Accuracy: 1.000, Precision: 1.000, Recall: 1.000,
F-score: 1.000
Lying ………………. Accuracy: 1.000, Precision: 1.000, Recall: 1.000,
F-score: 1.000
Running ………………. Accuracy: 0.984, Precision: 1.000, Recall: 0.984,
F-score: 0.992

Figure 22: Model1 Classification Accuracy

As you can see from the table above our CNN-4 respeck model achieves an accuracy of 98.4 to 100 for the essential activities. For Walking and Lying, the accuracy even up to 100. Average F1-score for CNN-4 respeck model is 0.992.

We then performed a Real-time test of the CNN-4 respeck model on the app.

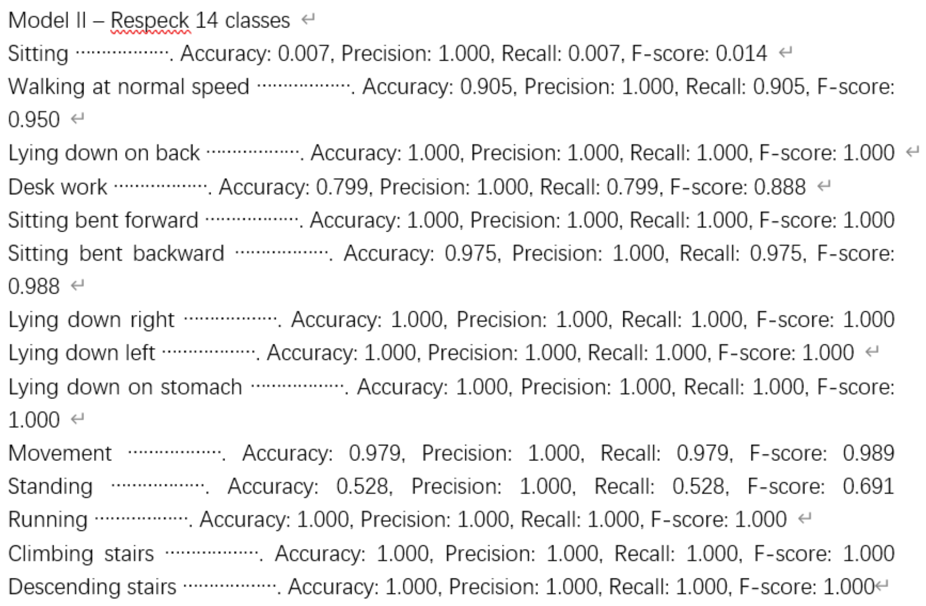


Activity Name	Ethan Sun	Jim Su	Pingqi Li
Walking	Walking	Walking	Walking
Running	Running	Running	Running
Lying Down	Lying Down	Lying Down	Lying Down
Sitting/Standing	Sitting/Standing	Sitting/Standing	Sitting/Standing

Figure 23: Real-time CNN-4 model test on APP

This table shows the real-time prediction result from peer review. The user behaves 4 essential activities after wearing the Respeck sensor.Then observe the predicted results on the app and fill in the table. The results of the tests are accurate.The results correspond to each activity.

Then,We firstly test the offline results of our Model2 (CNN-14 respeck model) and Model 3 (CNN-2 Thingy model) Separately.



Model II – Respeck 14 classes ↵
Sitting ……………… . Accuracy: 0.007, Precision: 1.000, Recall: 0.007, F-score: 0.014 ↵
Walking at normal speed ……………… . Accuracy: 0.905, Precision: 1.000, Recall: 0.905, F-score: 0.950 ↵
Lying down on back ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000 ↵
Desk work ……………… . Accuracy: 0.799, Precision: 1.000, Recall: 0.799, F-score: 0.888 ↵
Sitting bent forward ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000
Sitting bent backward ……………… . Accuracy: 0.975, Precision: 1.000, Recall: 0.975, F-score: 0.988 ↵
Lying down right ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000
Lying down left ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000 ↵
Lying down on stomach ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000 ↵
Movement ……………… . Accuracy: 0.979, Precision: 1.000, Recall: 0.979, F-score: 0.989
Standing ……………… . Accuracy: 0.528, Precision: 1.000, Recall: 0.528, F-score: 0.691
Running ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000 ↵
Climbing stairs ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000
Descending stairs ……………… . Accuracy: 1.000, Precision: 1.000, Recall: 1.000, F-score: 1.000↵

Figure 24: Model2 Classification Accuracy

From figure 24, we can observe that only using repeck to do classification can not distinguish Sitting and Standing. Accuracy for sitting is only 0.007 for Model2. And accuracy for standing is only 0.528 for Model2. But for all other activities except desk work, Model 2 can classify them with high accuracy(greater than 0.9 and up to 1) and high F-score. The accuracy of desk work is only 0.79 because the data from desk work and sitting are relatively similar.

To Analysis the reason, we plot the confusion matrix of model2:

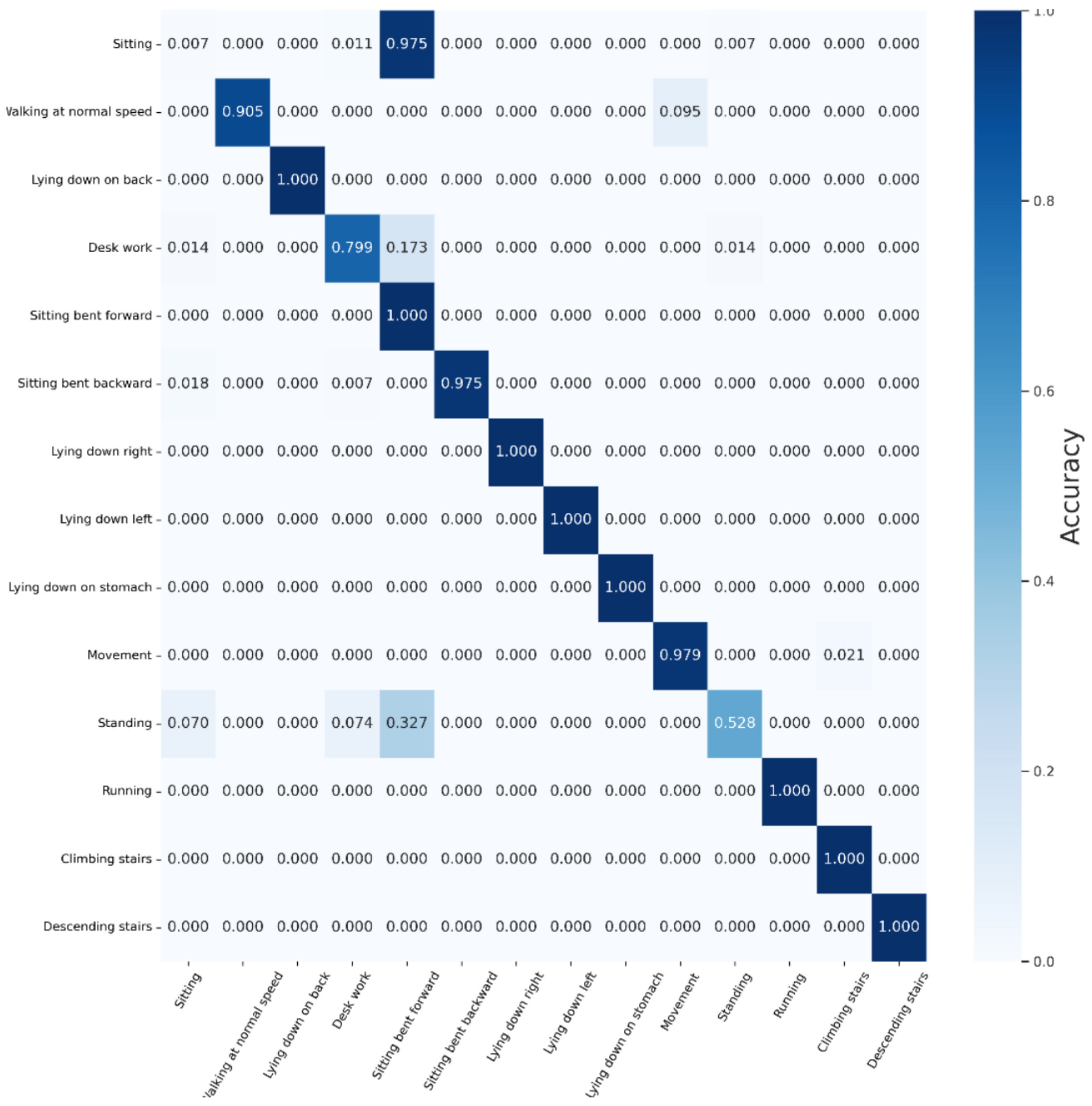


Figure 25: Model2 Confusion Matrix

As shown in Figure 25,Model2 will predict Sitting as Sitting bent forward at most of the time. Sometimes, It will also wrongly predict Standing and Desk Work as Sitting bent forward.

Our CNN-2 thingy model is designed to allow the app to differentiate between standing and sitting. Here is CNN-2 Thingy model classification report

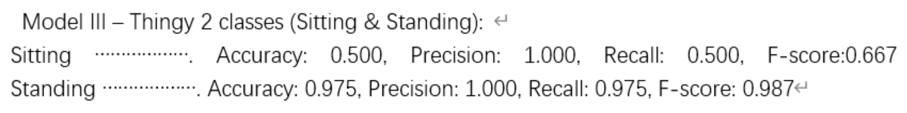

Model III – Thingy 2 classes (Sitting & Standing):
Sitting ………………. Accuracy: 0.500, Precision: 1.000, Recall: 0.500, F-score:0.667
Standing ………………. Accuracy: 0.975, Precision: 1.000, Recall: 0.975, F-score: 0.987

Figure 26: Model3 Classification Accuracy

model3 is still not very accurate for sitting(0.500), but can be used to assist model2 in differentiating between standing and sitting.

After we combine model2 and model3 together, we then used a real-time test to test the app's performance of classification 14 types of activities. The following table shows the actual results predicted on the app.

Activity Name	Ethan Sun	Jim Su	Pingqi Li
Sitting	Sitting	Desk work	Sitting
Standing	Standing	Standing	Standing
Sitting bent forward	Sitting bent forward	Sitting bent forward	Sitting bent forward
Sitting bent backward	Sitting bent backward	Sitting bent backward	Sitting
Desk work	Desk work	Desk work	Desk work
Walking	Walking	Walking	Walking
Running	Running	Running	Running
Movement	Movement	Movement	Movement
Ascending stairs	Ascending stairs	Ascending stairs	Descending stairs
Descending stairs	Descending stairs	Descending stairs	Descending stairs
Lying down on the back	Lying down on the back	Lying down on the back	Lying down on the back
Lying down on the surface	Lying down on the surface	Lying down on the surface	Lying down on the surface
Lying down on the right	Lying down on the right	Lying down on the right	Lying down on the right
Lying down on the left	Lying down on the left	Lying down on the left	Lying down on the left

Figure 27: Model2+Model3 Real time Classification Table

We can see that in the 14 categories of behaviour all predictions are correct except for the errors in sitting and sitting bent forward.The accuracy of model2 combined with model3 is very high and can reach over 90.

Benchmark

The PDIOT app: the memory footprint is around 15%, and the latency is two seconds to do every prediction because it requires two seconds to collect the data. The power consumption is about 5%. The application has a storage footprint of 33.65 MB, and the storage will increase as historical data is stored. After every prediction, the prediction results are stored in the history database and all other useless data are deleted.

1 Conclusion

In conclusion, our HAR mobile application use CNN 1D model to do the prediction/classification. We trained Model1(CNN-4 Respeck) for essential 4 activities has 0.99 offline accuracy and nearly 1 real-time accuracy. For the combination of Model2(CNN-14 Respeck) and Model3(CNN-2 Thingy), it has a over 90 real-time accuracy. On the back end, we implements many basic functions,such as registration/login, sensor connection, prediction, view history and step count. On the front end, We make a clean UI design. The user can easily know what functions are available on each interface. However, there are still some problems with our app. For example, the prediction latency is high, the accuracy of the step count is low, and all the files are kept locally. For future work,we wish to implement a model using shorter sliding window as input. This will reduce latency. The model can accurately identify each step of user. We want to put both the model and the database in the cloud. This will significantly reduce the amount of computing and memory space used by the app locally. We also want the sensors can be worn in a flexible position. The user can wear it on the arm or in a shirt pocket instead of only on the chest. But this requires a lot of re-collection of data to feed the model.

References

[1] Deep learning models for human activity recognition. [Online], 2018. https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/.

[2] A comprehensive guide to convolutional neural networks — the eli5 way. [Online], 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[3] A guide to recurrent neural networks: Understanding rnn and lstm networks. [Online], 2022. https://builtin.com/data-science/recurrent-neural-networks-and-lstm.

[4] Jubil T Sunny, Sonia Mary George, Jubilant J Kizhakkethottam, Jubil T Sunny, Sonia Mary George, and Jubilant J Kizhakkethottam. Applications and challenges of human activity recognition using sensors in a smart environment. *IJIRST Int. J. Innov. Res. Sci. Technol*, 2:50–57, 2015.

[5] A review of data fusion techniques. [Online], 2013. https://www.hindawi.com/journals/tswj/2013/704504/.

[6] DK Arvind, T Georgescu, CA Bates, D Fischer, and Q Zhou. Home-based pulmonary rehabilitation of copd individuals using the wearable respeck monitor. In *EAI International Conference on Body Area Networks*, pages 176–191. Springer, 2021.

[7] What is an inertial measurement unit? [Online], 2018. https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu.

[8] Nordic thingy:52. [Online], 2020. https://embeddedartistry.com/blog/2017/08/16/nordic-thingy52/.

[9] 1d convolutional neural network models for human activity recognition. [Online], 2018. https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/.

[10] A basic introduction to tensorflow lite. [Online], 2020. https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292.

[11] A gentle introduction to human activity recognition. [Online], 2020. https://indatalabs.com/blog/human-activity-recognition.

[12] Md. Milon Islam, Sheikh Nooruddin, Fakhri Karray, and 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*, 149:106060, 2022.

[13] How to measure acceleration? [Online], 2020. https://www.omega.com/en-us/resources/accelerometers.

[14] Gyroscopes. [Online], 2020. https://uk.farnell.com/sensor-gyroscope-technology.

[15] Magnetometers: A comprehensive guide. [Online], 2020. https://gmw.com/magnetometers/.

[16] Saurabh Gupta. Deep learning based human activity recognition (har) using wearable sensor data. *International Journal of Information Management Data Insights*, 1(2):100046, 2021.

[17] David Renggli, Christina Graf, Nikolaos Tachatos, Navrag Singh, Mirko Meboldt, William R Taylor, Lennart Stieglitz, and Marianne Schmid Daners. Wearable inertial measurement units for assessing gait in real-world environments. *Frontiers in physiology*, 11:90, 2020.

[18] Stefan Scholl. Fourier, gabor, morlet or wigner: Comparison of time-frequency transforms. *arXiv preprint arXiv:2101.06707*, 2021.

[19] Huaijun Wang, Jing Zhao, Junhuai Li, Ling Tian, Pengjia Tu, Ting Cao, Yang An, Kan Wang, and Shancang Li. Wearable sensor-based human activity recognition using hybrid deep learning techniques. *Security and communication Networks*, 2020, 2020.

[20] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40, 2021.

[21] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[22] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2020.

[23] sliding window (windowing). [Online], 2018. https://www.techtarget.com/searchnetworking/definition/sliding-windows.

[24] Sojeong Ha and Seungjin Choi. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In *2016 international joint conference on neural networks (IJCNN)*, pages 381–388. IEEE, 2016.

[25] Kun Xia, Jianguang Huang, and Hanyu Wang. Lstm-cnn architecture for human activity recognition. *IEEE Access*, 8:56855–56866, 2020.

[26] Ronald Mutegeki and Dong Seog Han. A cnn-lstm approach to human activity recognition. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 362–366, 2020.

[27] Gated recurrent unit (gru). [Online], 2018. https://blog.marketmuse.com/glossary/gated-recurrent-unit-gru-definition/.

[28] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[29] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.