THE UNIVERSITY *of* EDINBURGH

PdIoT coursework3 (2022-2023)

Zhenlin Yang, Jiaqi Huang, Qiwen Deng

Abstract

IoT systems and smart wearable devices are more and more often get noted in the industry. Now, many enterprises and organizations have applied machine learning to IoT systems. In our project, we will use two wearable sensors to collect IMU data, use machine learning to analyze them, and will show the classification result on the smartphone. In detail, we will use two sensors one collecting on the 6-axis(accelerometer and gyroscope), and the other collecting the data IMU data on the 9-axis(accelerometer, gyroscope, and magnetometer). Then the collected data will be transmitted to a smart phone (Android Phone in our case) via Bluetooth low energy (BLE), and then will be classified locally in Android App. The machine learning algorithm we finally adopted is a type of transformer and was found to performed very accurate.

1 Introduction

IoT devices are embedded computing devices that can connect to a wireless network and have the ability to transmit data, when many devices such as this are working together we called them the internet of things system. The wearable Internet of things system has now become a hot spot in the Information and Communications Technology(ICT) industry. With the advancement of mobile devices(hardware aspect specifically), mobile systems, communication networks, and machine learning. The Internet of things system is more viable ever than before.
In this project, we especially aim for the machine learning application on IoT systems. We have already had some implementations of machine learning on wearable IoT systems such as works from Apple Watches can detect the Heart rate (HR) dynamics in response to workout intensity and duration and measure key aspects of an individual's fitness and cardiorespiratory health.[4] They used a hybrid machine learning-based model here to recognize and learn user-specific fitness parameters and combine a physiological model of HR and demand during exercise.
The project is aiming to develop a prototype of a complex IoT system which able to classify real-time human activities into different 4 classes or 13 classes with high accuracy. With the project, we will be using two wireless Inertial Motion Unit (IMU) sensors and machine learning techniques.

1.1 Project Aim

Within the project, our product will contain the following features
Essential features:

- ✓ On-device, real-time Human Activity Recognition (ML in the Android app or custom firmware on the Thingy)
- ✓ Basic interface for users to view the current activity
- ✓ Ability for user to pair a Respeck or a Thingy
- ✓ Classification of a subset of activities:
  - Sitting/Standing
  - Walking
  - Running
  - Lying Down
- ✓ Accuracy of 85-90 percent

Desirable features:
Accuracy of 91-95 percent for your basic implementation, plus at least one of the following features:

✓ Classification of all activities - provide cross-validation accuracy

✓ Use of both devices (Respeck, Thingy) or sensors (accelerometer, gyroscope) to improve accuracy

× Intuitive user interface, allowing user logins and the ability to view historic data

Advanced features:

- Notifications for reminders to move
- Step counting

1.2 Brief description of the method adopted

Briefly on the method, the product can be decomposed into the following modules: hardware, system, algorithm, and data.
For the hardware, we have 2 sensors and a smartphone which both use the Nordic NRF52 System on Chip (SoC), containing a low-power Arm Cortex processor and Bluetooth Low Energy (BLE) radio for wireless communication:

- Respeck: A compact IMU device, designed in-house, with a 3-axis accelerometer and gyroscope sensor for physical activity monitoring
- Thingy: An off-the-shelf IMU prototyping platform produced by Nordic Semiconductor with 3-axis accelerometer, gyroscope, and magnetometer sensors
- Smartphone: The smartphone is used as the terminal to support multiple jobs
  - Collect and store the human activity data
  - Compute and show the recognition result
  - Enable more future features(cloud computing/storage for example)

For the system, as we use the smartphone to do the processes, the system is in the form of Android App. It collects, stores, and processes the data.

For the classification algorithm, we adopted two algorithms. Both of them are neuron network(NN) models, one is Long short-term memory network(LSTM) plus convolutional neural network (CNN), the other is Transformer. Both LSTM and Transformer are the language model from NLP but they are found to be effective in processing sequential data in general. While CNN is a neuron network which most commonly applied to analyze images, but they found to be effective and easy to use on spatial data. These methods have the following advantages compared to other machine learning algorithms:

- Capability on big data, it is easier to train and test with NN models, and generally NN model performs better when data is large and complicated.

- Better supports the system and hardware perspective. We have many tools to help us accelerate/transfer the NN model on mobile devices. Thus making it more applicable.

1.3 Physical activities used in the classification

The physical activities that we wanted to recognize can be split into two ways: For essential features, we are aiming to recognize the activities into four classes:

- Sitting/Standing
- Walking
- Running
- Lying Down

For the desirable features, we are recognizing all of the following:

- Sitting
- Sitting bent forward
- Sitting bent backward
- Desk work
- Standing
- Walking
- Climbing stairs
- Descending stairs
- Running
- Lying Down on the left
- Lying Down on the right
- Lying Down on the front
- Lying Down on the back
- General movement, including sudden turns, bending down, getting up from chairs, and anything else

1.4 Summary of results

For the activity recognition results, we have achieved an accuracy of 100% on the essential features on the 2022 data and accuracy of 92% on 2021 data. For the desirable features, we achieved 98% on 2022 data.

2 Literature Survey

2.1 A review of the SOTA for HAR algorithms

With the development of the Internet of Things, human activity recognition based on wearable devices has become a research focus in recent years. The current mainstream recognition methods include two types. One of them is a deep learning approach based on CNN and LSTM, or a combination of both. This type of approach is widely sought after by the research community due to its simplicity and effectiveness in modelling. However, as data scales and features become more complex, simple models present a bottleneck in recognition performance.

With the prevalence of embedded wearable devices in recent years, more and more scenarios require models with higher requirements in terms of size and inference speed. Very large CNN networks as well as LSTM networks are unable to meet such requirements, leading more and more researchers to focus on transformers. Transformers are known for their powerful parallel computing capabilities and can have the advantage of being more flexible than CNN networks and faster than LSTM networks. This allows transformers to be hopefully deployed in equipment with high performance requirements. In the next section, we will focus on current research findings on transformers for human activity recognition.

The earliest research applying transformers to human activity recognition on IMU data from wearable devices was presented by Yoli Shavit et al. [5] They proposed a transformer-based architecture as a general framework for general activity recognition in HAR tasks. The model they designed first extracts features from the IMU data via a convolutional encoder, then uses a CLS token as a vector representation for prediction, which is passed into the transformer along with the extracted feature vectors for final prediction. Figure 1 illustrates the architecture of the model they designed.

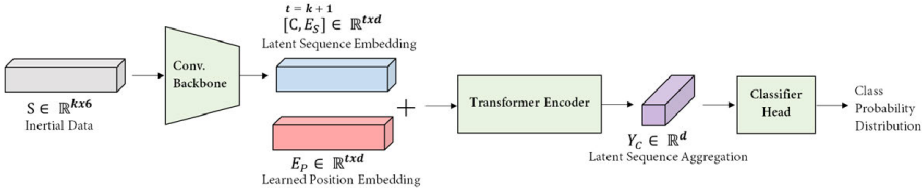

Figure 1: The proposed framework for inertial data classification with Transformers. [5]

This type of architecture uses convolutional networks to smooth and extract features from IMU data, and correlates and enhances some of the features through an attention architecture, greatly enhancing the model's fitting ability. However, the model architecture is deficient in that it requires the extra use of CLS tokens as predictive vectors, which will limit the representational power of

the model. Our model design eschews the use of CLS tokens, opting instead for global averaging of the output representations across the sequence, resulting in ensemble generalisation capability.

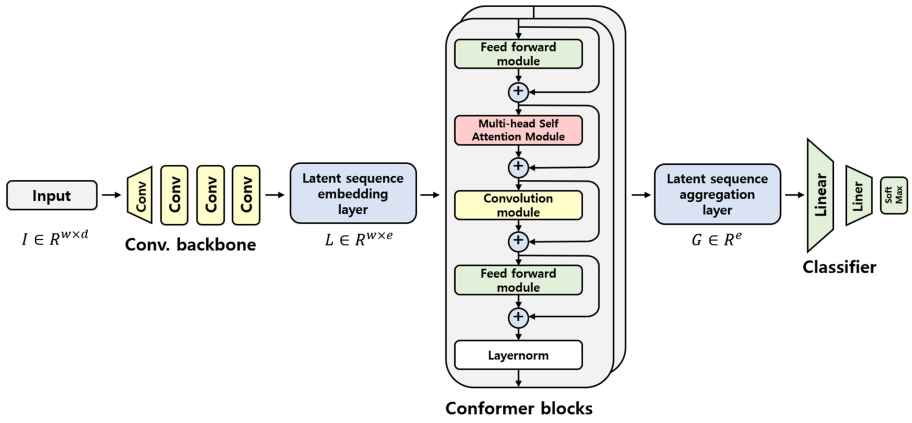

Figure 2: Proposed conformer-based HAR model. [3]

Another similar study uses Conformer as the backbone for human activity recognition, which was proposed by Yeon-Wook Kim et al. [3] They also use a convolutional encoder to extract the features and feed them into a transformer with convolutional enhancement. Figure 2 illustrates the Conformer-based human activity recognition model. This model has a more customisable structure than the previous model, and thanks to its optimisation and improvement of the transformer architecture, the performance of the model has been improved even more. However, too much customisation makes the model design complex and difficult to re-optimise, and the large model structure is not conducive to deployment and model inference on the embedded device side. We designed the model to circumvent these points and achieve similar performance to Conformer with a simplified and customised model structure and a lightweight model size.

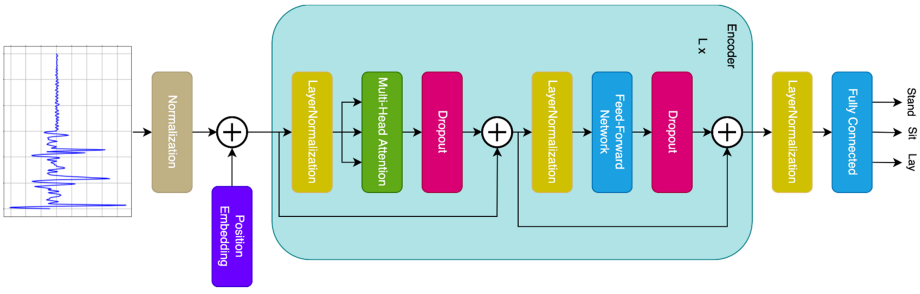

Figure 3: The transformer model for human activity recognition. [1]

In addition to optimising the design of the model architecture, a recent study

has shown that increasing the size of the data allows the original transformer to achieve state-of-the-art results. This study was proposed by Dirgová Luptáková et al. [1] They found that data enhancement and data smoothing were effective in improving the performance of the transformer. Such results benefit from the fact that transformers are still unable to saturate performance in data scaling. Figure 3 illustrates the transformer model for human activity recognition. During the training of our model, we also use some data augmentation methods to expand the data and thus enhance the generalisation ability and performance of the model.

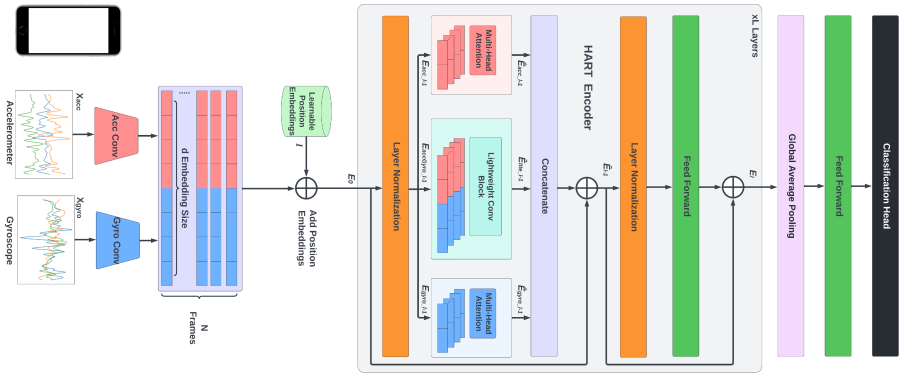

Figure 4: Overview of Human Activity Recognition Transformer. [2]

A few months ago, a lightweight and sensor-based transformer designed for IMU data was proposed by Sannara EK et al. [2] This model makes use of a dual convolutional encoder for the first time to encode accelerometer and gyroscope data separately. An attention module has been specifically designed for both types of data to extract features. Figure 4 presents the overview of human activity recognition transformer. The model we designed was inspired by it, again using a dual encoder as well as multiple attention modules to extract features. The difference is that we only use the single layer transformer encoder module for feature extraction and we extend the input data to multiple sensors in multiple devices rather than just a single device.

Looking at the current progress of transformer research in human activity recognition, there are some issues that still need to be addressed. One of the most serious problems is that human activity recognition models are often difficult to generalise well. In our experiments, the transformer fitted the distribution of the training data quite well, and in particular the model was able to identify the distribution of the data for a specific person. In addition to this, the model also achieves very good results on the validation set. However, when we use data recorded from people who do not appear in the training set, the recognition performance of the model is greatly reduced. This means that IMU data is extremely person-specific in nature. How to design a generalised HAR model for non-person-specific data has become a key research direction for the

future.

3 Methodology

3.1 A description of the system and its implementation

3.2 Hardware and firmware

We used 2 devices and a smartphone. The first sensor is called Respeck, this is a small device that has the size of $4.5 \times 3.7 \times 1.3$ cent meters, and it weights only 17 grams. It is mainly used in clinical applications, e.g. respiratory monitoring. It has an accelerometer for 3-dimensional space and three angular rate sensor in the gyroscope, this indicates that we will have 6 signals for one time unit. The accelerometer consumes 500µA of electric current under a 2.2V - 3.6V voltage supply. It has a sleep mode to save energy when it met the predefined condition. Moreover, the sensor supports Bluetooth Low Energy (BLE), which is more economical on energy usage than other IoT edge protocols while also keeping the performance.

When we need to connect the Respeck to the smartphone, we need the MAC address of the device. The simplest way to do it is to scan the QR code on the Respeck, as it encodes the MAC address. Also, Respeck supports NFC, we can get the MAC if our smartphone support NFC, lastly there is a very straightforward way to do this: manually enter the MAC address. Once we get the MAC address, the smartphone will connect to the Respeck via Bluetooth, we will have the following signals from a light source of the Respect:

- Green: sensor ON and NOT CONNECTED
- Blue: sensor ON and CONNECTED
- Red: sensor ON and DISCONNECTED

We need to place the Respeck in a specific place of our body. The Respeck sensor should be placed on the left lower ribcage, with the blue half against the skin. Make sure that the Respeck is first put into the small plastic bag provided. You should be able to read the Respeck label when placing it on the chest – this ensures the sensor is held the right way up, as shown in the figure below.

The other sensor we used is called thingy. it is larger than Respeck. It also connects to Bluetooth-enabled devices, and send the data from the sensors and actuators to an app or to the cloud. Similar to Respeck, it also includes an NFC antenna and an RGB LED that signals the state(connected/disconnected) of thingy. But is also has a switch to turn on/off the battery. For the power, it is powered with a rechargeable Li-Po battery with a capacity of 1440 mAh, that can be charged via USB.

Similar to Respeck, thingy also has an accelerometer for 3-dimensional space and three angular rate sensor in the gyroscope. Furthermore, it have a 3-axis magnetometer, the magnetometer measures magnetic fields, delivering a fixed

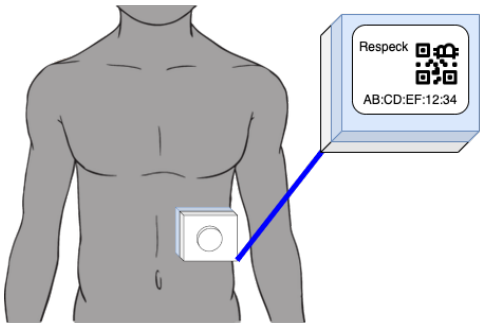

Figure 5: Placement of Respeck sensor on the human body

point of reference (Earth's magnetic field). This data can be fused with the gyroscope and accelerometer data to deliver absolute heading: Not only how many degrees heading have changed, but its relation to magnetic north. In the same way, 9 axis devices also measure attitude (yaw, pitch, roll), or absolute orientation against a frame of reference.

The connection of thingy is the same as how we connect the Respeck. It also support QR code scanning, NFC and manually input the MAC address. One thing different is that we need to configure it with an app to a fixed motion processing unit frequency of 25 HZ.

Instead of putting the sensor against the skin of our left lower ribcage. We are putting thingy in the front right pocket of your trousers, with the circle placed in the upper right corner and the USB port facing downwards.

Regards to the types of sensors, we have the accelerometer and gyroscope for both devices and the magnetometer for thingy only. The following will briefly explain what they are measuring.

An accelerometer works using an electromechanical sensor that is designed to measure either static or dynamic acceleration. Static acceleration is the constant force acting on a body, like gravity or friction whereas dynamic acceleration forces are non-uniform, an example is vibration or shock. Basically, the accelerometer will be able to measure the acceleration of the movement of the device.

A gyroscope senses angular velocity produced by the sensor's own movement.

Lastly, the magnetometer measures the strength and direction of the magnetic field in the vicinity of the instrument.

Figure 6: Placement of Thingy sensor on the human body

3.3 Wireless communication

There are three types of wireless communication between the sensor and the smartphone.

- Quick response code
- Near-field Communication (NFC)
- Bluetooth or Bluetooth Low Energy (BLE)

3.3.1 Quick response code

Quick response code(QR code) is now wildly used in our lives. It is wildly used in storing data for a locator, identifier, or tracker that points to a website or application.
A QR code consists of black squares arranged in a square grid on a white background, including some fiducial markers, which can be read by a camera,

and processed within Reed–Solomon error correction making sure that image can be correctly interpreted (even the QR code sticker is damaged). The stored data is then extracted from the QR code that is present in both the horizontal and vertical components of the image.
The main advantage of a QR code is that it only takes a piece of sticker to make, compare with NFC where we need some chips to make it work. While QR code also does not require any electricity, so it saves power to some extent.

3.3.2 Near-field Communication

NFC is a protocol widely used in close-distance communication, which usually needs 10cm or less. It operates at 13.56MHz on ISO/IEC 18000-3 air interface and at rates ranging from 106 kbit/s to 424 kbit/s. The transmission rate is slower than Bluetooth(2.1 Mbit/s or BLE (1 Mbit/s)), but it would be enough for our project.
NFC tags contain data and are typically read-only, but may be writable. In our project, we just use the NFC tag to read the MAC address into the smartphone, it is read-only for us.

3.3.3 Bluetooth

Bluetooth is different from the other two communication methods we have discussed above in our project. We use Bluetooth to transmit the data collected in real-time, so it is much more frequently used than NFC and QR codes. The reason for this is we want to recognize human activity in real-time will require us to process the IMU data in real-time, so in order to achieve this online classification, we need a powerful terminal to support the computations. However, wearable sensors will not be able to do this due to the limitations on battery size, power consumption, weight, etc. Thus we need to process the real-time data somewhere else.

However, we have options more than Bluetooth and BLE. We can also use WiFi to do this, but Bluetooth and BLE will still be better options:

	Bluetooth	BLE	WiFi
Current consumption on awake	35 mA	15 mA	245 mA
Current consumption on sleep	$9\mu A$	$0.78\mu A$	$30\mu A$
Bit-rate	2.1 Mbps	1 Mbps	600 Mbps (depends)
Latency	80-130 ms	6ms	20ms (depends)

As shown in the table, the advantage of Bluetooth and BLE are required less energy to run, while WiFi is faster and maybe also better on latency if the connection is good. Thus, in our case, the data we transmit will not be very large data, so we may not need the speed to be that fast also wearable devices are preferred on saving energy. Therefore, Bluetooth or BLE is better to be adopted in our case.

3.4 Machine Learning methods for activity recognition

3.4.1 Data Preprocessing

The data we used were collected by more than 80 volunteers. All volunteers sampled 14 types of human activity data, each was collected for 30 seconds at a frequency of 25 Hz. Those data are guaranteed to be at some level of quality as they will be all 'human-filtered' by specialists(our markers). The following will be some examples where the data might not be considered:

- Empty values
- Abnormal trends
- Outliers

The recording data sometimes may be empty due to the unstable connection of Bluetooth(caused by the noise or the distance of communication). The abnormal trends sometimes happen due to the loose sticker or not wearing the sensor properly. As for the outliers, they may be caused by not wearing the sensor properly or not labeled correctly or anything else.

3.4.2 Sliding window

The sliding window approach is one of the basic rules of how we process the IMU data. The data will be processed as the figure shown below: We can decide

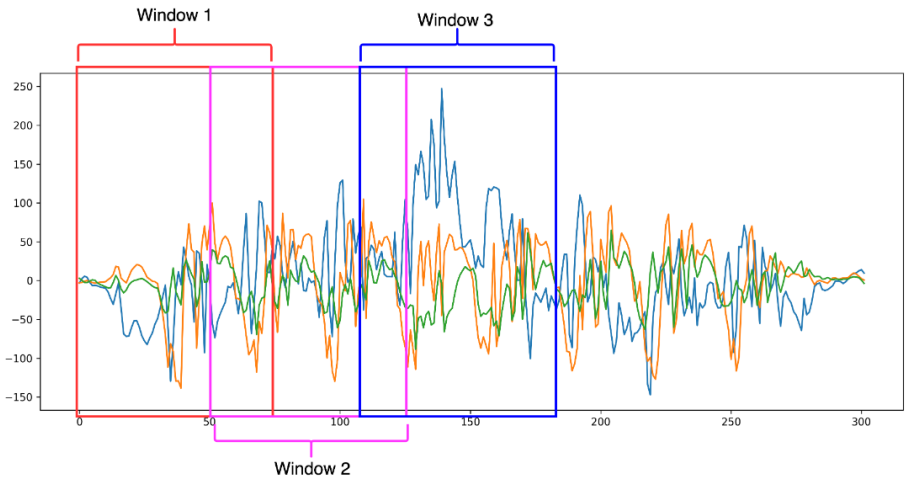

Figure 7: Each data within the window will be proceeded by the algorithm

the window length and how much they overlap with each other. This will be a balance to be considered between the speed and the accuracy of the model. In the project, we decided to use a continuous real-time forecast in the provided data frame. We chose a window size of 50 (2 seconds) and a step size of 25. Hence, the prediction results will be updated every single second.

3.4.3 NN models

In recent years, human activity recognition can be modeled by neural network models for IMU data. The vast majority of approaches achieve state-of-the-art results in terms of real-time performance by using CNNs or LSTMs or a combination of both. Recently, transformer networks have made breakthroughs in the field of NLP and CV. Many researchers have tried to apply such model structures to the field of human activity recognition, but the model structures are very heavy and not suitable for embedded applications. Therefore, we propose a lightweight, sensor-based transformer network structure dedicated to IMU-embedded mobile devices, which we call M-HART.

3.4.4 Transformer

The full name of M-HART is Multi-Device Fusion Human Activity Recognition Transformer. It fuses accelerometer and gyroscope features from both Respeck and Thingy devices to predict human activity. It features a lightweight structure, short real-time inference time, low energy consumption, and adapts to data from multiple devices with different sensors. It is found experimentally that our model has fewer FLOPs compared to CNN/LSTM models while maintaining state-of-the-art results. Figure 8 shows the architecture of M-HART.

We used Respeck and Thingy to collect data on 14 different human activities recorded from 124 volunteers as a benchmark to train and evaluate the performance of the model. Each piece of data is professionally and manually reviewed and screened to filter out all erroneous data, and sampled to 25Hz, and cropped to 30 seconds. We roughly merged Respeck and Thingy data for the same volunteers and split the frames by sliding windows. We choose a sliding window size of 25ms and coverage of 10ms to reduce the inference speed of the model and the system latency while maintaining performance. Our 14-class benchmark contains a total of 102,540 frames, and the training and validation sets are prepared by a stratified sampling ratio of 8 to 2. Also, we merged the 14-class benchmark into the 4-class benchmark by label fusion for evaluating the performance of the model in essential activities. Our 4-class benchmark contains a total of 97119 frames and the training and validation sets are prepared using exactly the same preprocessing as the 14-class benchmark.

For the architecture of the model, the input to M-HART is a window of data $x \in R^{W,M*3*S}$ of W window time-length, composed of data provided from S sensors in M devices, each having three channels (X, Y, Z axis). In order to better fuse the data from S sensors in M devices, the same axis of the same sensors in multiple devices are next to each other in the sequence. In the preprocessing stage, the continuous input data is segmented by a window of size W samples with partially overlap. In our report, we use a setting where S = 2 since our prepared datasets contain only accelerometer and gyroscope data. We also use M = 2 since the data from both Respeck and Thingy are used together. The window size W = 25 as we mentioned before.

At the beginning of the M-HART process, x is framed and linearly projected,

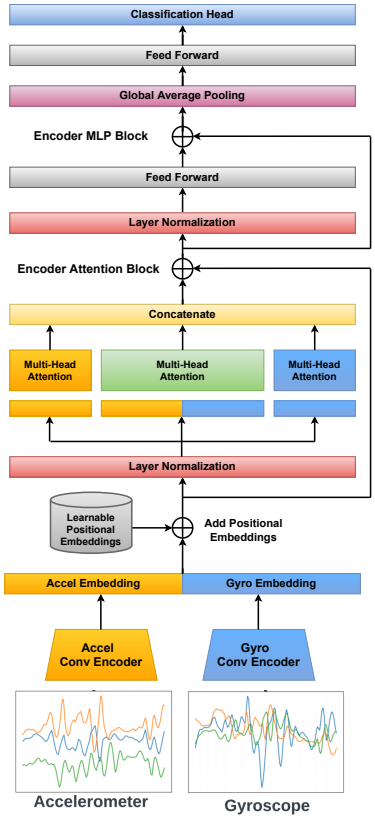

Figure 8: M-HART Architecture: lightweight, multiple devices, a sensor-based transformer that has a smaller size and faster inference speed than all existing HAR transformers, while maintaining state-of-the-art recognition performance.

in a sensor-wise manner, by specific convolutional encoders for each sensor data. More specifically, the accelerometer and gyroscope data from Respeck and Thingy are combined by sensor type and processed by convolutional encoders designed for accelerometers and gyroscopes, respectively. The accelerometer-oriented convolutional encoder processes $x_{acc} \in R^{W,6}$ and a gyroscope-oriented convolutional encoder processes $x_{gyro} \in R^{W,6}$. Given hyper-parameter embedding dimension d, each sensor-wise convolutional layers have d/S amount of filters with kernel size of K and stride size of T. We use $d = 64$ and $S = 2$ since there are 2 different types of encoders, and also K = 3 and T = 1 in our convolutional layers. The projection of the accelerometer and gyroscope input data generates N embeddings composed of $\hat{x}_{acc} \in R^{N,d/S}$ and $\hat{x}_{gyro} \in R^{N,d/S}$. The two outputs of the convolutional encoders are concatenated together to form the projected embedding vectors. This embedding vectors are then added to learnable position embeddings $I \in R^{N,d}$ to finally have $E_0 \in R^{N,d}$. The

operations all together can be written as below:

$$E_0 = concat(Conv_{acc}(x_{acc}), Conv_{gyro}(x_{gyro})) + I \tag{1}$$

E_0 will be used as the first M-HART encoder input, which is then processed with layer normalization to standardize the embeddings across different sensors. Since the normalized embeddings were initially concatenations from two different sensors, we can separate back the accelerometer and gyroscope embedding vectors. M-HART partitions and uses the different embeddings as follows: the accelerometer embeddings $E_{acc} \in R^{N,d/2}$ are processed by an accelerometer-wise Multiheaded Self-Attention (MSA_{acc_l}); the gyroscope embeddings $E_{gyro} \in R^{N,d/2}$ are processed by a gyroscope-wise Multiheaded Self-Attention (MSA_{gyro_l}); the both sensors embedding $E_{accGyro} \in R^{N,d}$ are processed by a general Multiheaded Self-Attention ($MSA_{accGyro_l}$). The outline of the process is presented below:

$$E_{acc_{l-1}}, E_{gyro_{l-1}}, E_{accGyro_{l-1}} = LN_{l-1}(E_{l-1}) \tag{2}$$

$$\hat{E}_{acc_{l-1}} = MSA_{acc_l}(E_{acc_{l-1}}) \tag{3}$$

$$\hat{E}_{gyro_{l-1}} = MSA_{gyro_l}(E_{gyro_{l-1}}) \tag{4}$$

$$\hat{E}_{accGyro_{l-1}} = MSA_{accGyro_l}(E_{accGyro_{l-1}}) \tag{5}$$

Multi-headed attention modules designed using specific sensor data can model data from the same modality to capture more unique data patterns. Combining single-modal data and mixed-modal data helps to extract richer information as well as more flexible attention patterns. Afterwards, we concatenate $\hat{E}_{acc_{l-1}}$, $\hat{E}_{gyro_{l-1}}$ and $\hat{E}_{accGyro_{l-1}}$ together to create $\hat{E}_{l-1}$. The outline of the process is presented below:

$$\hat{E}_{l-1} = concat(\hat{E}_{acc_{l-1}}, \hat{E}_{gyro_{l-1}}, \hat{E}_{accGyro_{l-1}}) + E_{l-1} \tag{6}$$

$$E_l = MHARTEncoderBlock_l = FF_{l-1}(LN(\hat{E}_{l-1})) + \hat{E}_{l-1} \tag{7}$$

Lastly, the outputs of the last M-HART encoder block are combined through a GAP layer, where the outputs are passed to a feed-forward layer and then finally given to the classification heads.

For the experimental setup, we trained 14-classified and 4-classified M-HART models on two benchmarks, respectively. Both models use the similar training hyperparameters but different model hyperparameters. The 14-classified M-HART model uses an embedding size of 64, a fully connected layer size of 256, and 8 attention heads. While 4-classified model uses a hyperparameter of half the size of the 14-classified model, which only uses an embedding size of 32, a fully connected layer size of 128, and 4 attention heads. This compresses a 14-classified model of 662KB size into a 4-classified model of 180KB. Compressing the 4-classified model is because we experimentally found that identifying essential activities does not require overly complex model fitting capabilities,

and reducing the model hyperparameters not only improves the inference speed of the model, but also maintains a high recognition performance.

We trained the 14-classified and 4-classified models 100 and 20 epochs, respectively, using a batch size of 128. We use adam as the optimizer and use a learning rate of 0.001. In addition, we also used warmup steps 10 and global clipnorm 3.0 as trick for model optimization. Finally, we train the model with optimal performance using early stopping and callbacks that retrace the optimal model. For evaluation, we use accuracy, precision, recall, and F1 scores on the validation set to evaluate the performance of the model. We also use the confusion matrix to analyze the distribution of bad samples to better understand the limitations of the model performance. The specific experimental results will be shown in the later sections.

3.5 Mobile Application

This section will discuss the overall structure of the human activity recognition mobile application. The explanation of the specific function will be summarised in each module.

3.5.1 Application Set up

The UML class diagrams of the project are created by the Visual Paradigm, which is software to reverse the instance to the class diagram. Each class diagram shows the methods and attributes relationship between the activity file and its XML layout file.

Connection:

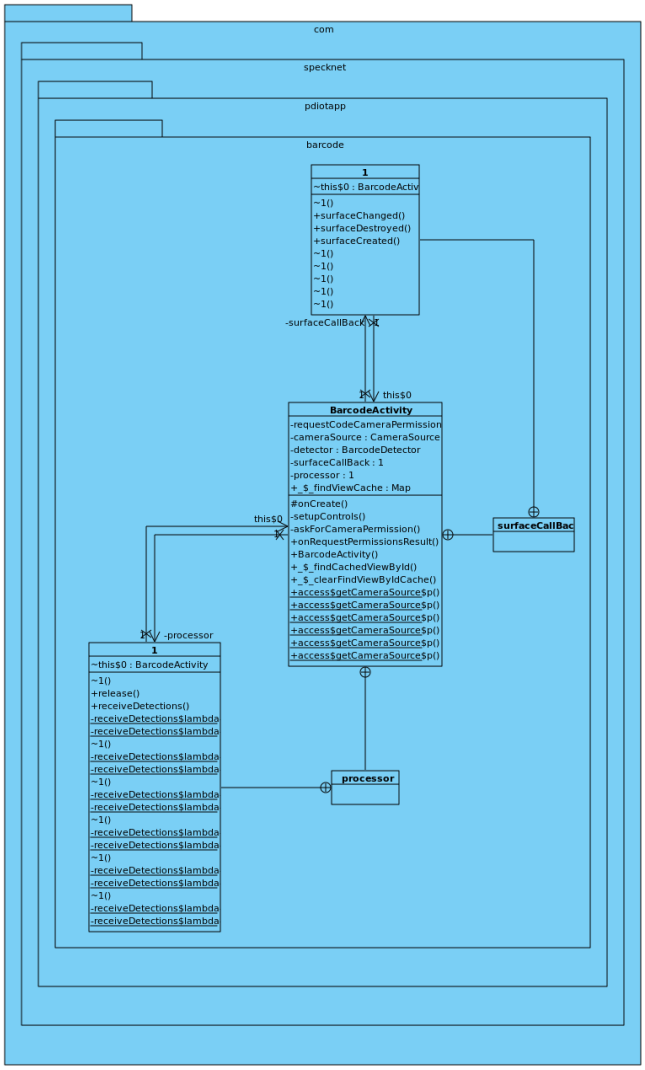

Figure 9: barcode

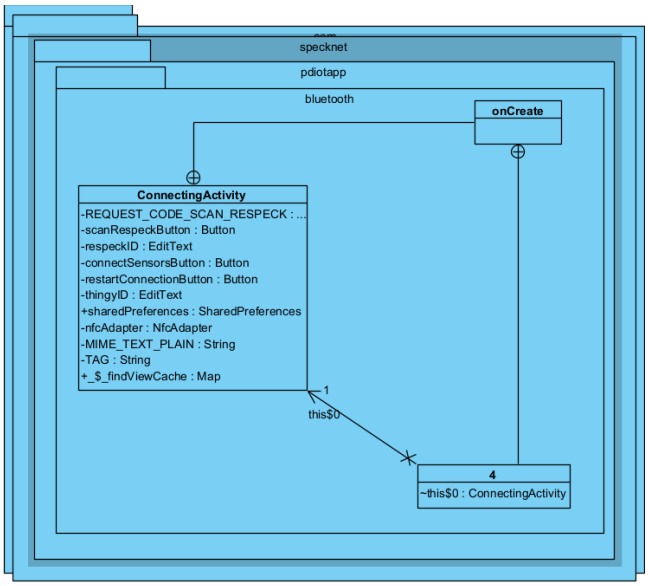

Figure 10: bluetooth

The figures above show the processing of how the sensors(Thingy and RESpeck) connect to the mobile phone.

Figure 9 contains the activity of the barcode recognition. When the user chooses to use the scanner to get the information from the barcode on a device, this activity class will request the permission of the user's mobile phone camera to scan and read the QR code. This activity will create a thread to read the machine's identifier and save it during this process.

Figure 10 includes the mobile application's processing using Bluetooth, creating the connection with the inertial measurement unit devices (RESpeck and Thingy). It will use the device's recognition code collected by scanning the barcode or directly typing by the user. This activity uses the pre-written standard in the same package named BluetoothSpeckService to initialize a Bluetooth adapter, which creates, maintains, and discords the connection.

Only when both Thingy and RESpeck are connected to the mobile application, it has the ideal performance of user activity recognition.

LiveData:

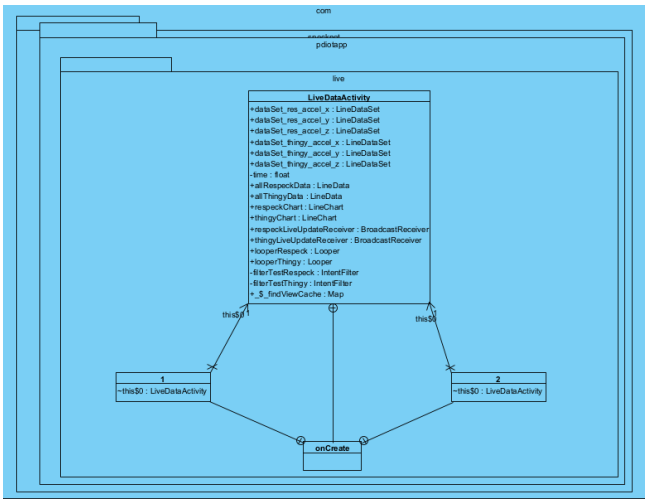

Figure 11: Live Data Graph

The live data passed by the RESpeck and Thingy will be constantly stored and displayed in dynamic graphs. The time step of the display is one, so with the frequency 25hz, it will append 75 data points of acceleration on three axes for each sensor user worn on every second. The data will be stored in the form of LineDataSet so that user can check their time series with three fold lines. There will be a RESpeck graph to represent the motion of RESpeck and a Thingy graph for the motion of Thingy.

Log in and Sign up:

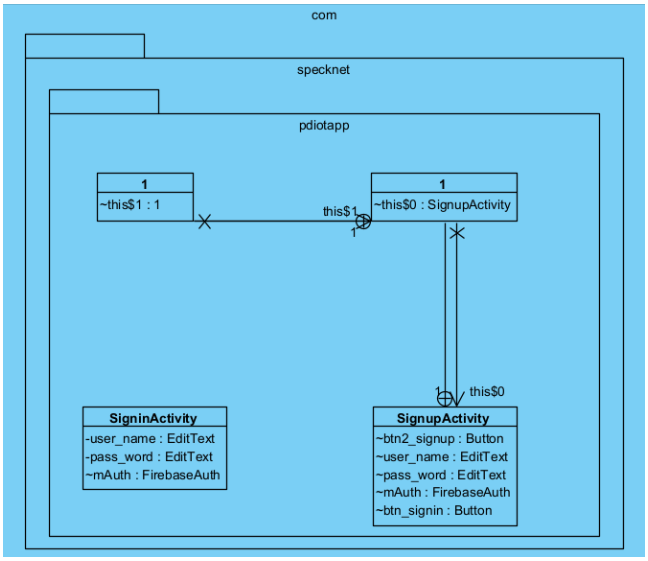

Figure 12: Log in and Sign up

Once the user opens the application, they will need to go through the login fragment. The sign-up and sign-in activities are bonding with XML page activity_login.xml and activity_signup.xml. When users use the application, they will be required to log in so that their data can be stored in the google firebase based on the user's account. If they currently do not have an account, they can press the button to switch to the sign-up page. Users will sign up with a username in the form of an email address and a password equal to or larger than six characters to ensure the account's safety. When users fail to create the account, it will have announcements in the text box to tell the user how to fit the needs of signing up for an account. Every time the user tries to log in, it will need authorization from the firebase server matching the account information in the database. After login, they will get the greeting message, and their user name will be remembered the next time they log in. Users can log out at any time.

3.5.2 Prediction

Prediction:

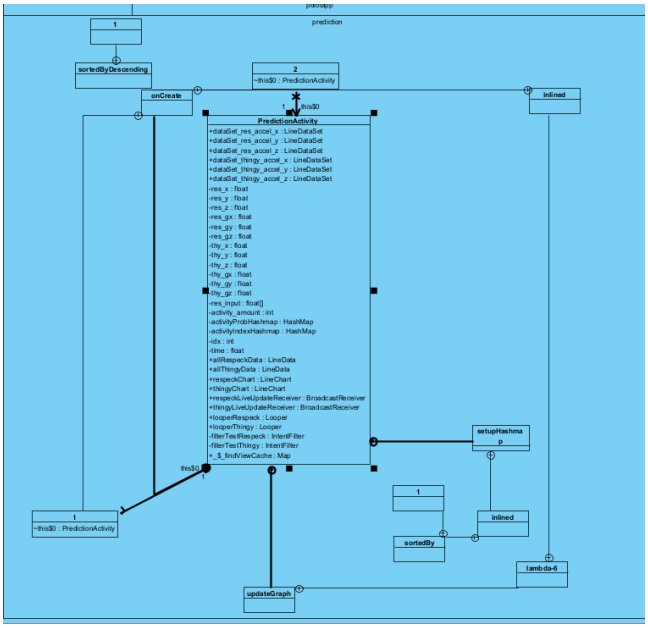

Figure 13: Prediction

Algorithm 1 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  if $n < 0$ **then**
    $X \leftarrow 1/x$
    $N \leftarrow -n$
  else
    $X \leftarrow x$
    $N \leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow N/2$
    else $\{N$ is odd$\}$
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

3.6 Software organisation

3.7 Testing

4 Results

In this section we will focus on critical analysis using quantitative methods and performance analysis presented as graphs and balanced interpretation of the results. In the methodology section we have described in detail the experimental setup, including the preparation of the dataset, the training of the model and the evaluation methods. Therefore, this section will not dwell on the experimental procedure, but will focus on the results. We present the experimental results in two main ways, one by reflecting the model's recognition performance and the other by reflecting the model's hardware performance. The following sections provide a detailed description of each.

4.1 Benchmark of Implementation

This section will focus on the performance of the application and the model on hardware, including memory usage, power consumption, processing cycles and latency. Table 1 shows the results presented using the Andriod Profiler in Andriod Studio. We can see that the application has many advantages such as low power consumption, low computational overhead, low latency and a small memory footprint. These are made possible by the simplicity and ease of use of our applications, the efficiency of our data pipeline and the light weight and fast inference of our models.

	memory	power	processing cycles	latency
Classification	$180 \pm 30MB$	Light Level	$11 \pm 3\%$	52 ms
Main Page (B Closed)	$80 \pm 10MB$	Light Level	$\approx 0\%$	116 ms
Main Page (B Acting)	$100 \pm 30MB$	Light Level	$2 \pm 1\%$	175 ms
Pairing Sensors	$160 \pm 10MB$	Light Level	$3 \pm 1\%$	223 ms

Table 1: Hardware performance benchmarks

4.2 Performance of both offline and in real-time

Table 2 and Table 4 show the recognition performance of the 4-category and 14-category models on the validation set respectively. These results are averages obtained through 10-fold cross validation. We can find that the recognition performance of both the 4-category model and the 14-category model achieves a recognition accuracy of over 99% in the majority of categories.

From Table 2, we can see that the model is able to classify all the essential activities very well. This result is due to the fact that the four types of activity have very different data patterns. Sitting/Standing and lying down are static activities, while walking and running are dynamic activities. Static and dynamic

	precision	recall	f1-score	support
Sitting/Standing	1.00	1.00	1.00	6775
Lying down	1.00	1.00	1.00	5660
Walking	1.00	1.00	1.00	5179
Running	1.00	1.00	1.00	1810
accuracy	1.00	1.00	1.00	19424
macro avg	1.00	1.00	1.00	19424
weighted avg	1.00	1.00	1.00	19424

Table 2: Classification report for 4-category model

activity differ significantly in accelerometer data. Static data is usually presented in a parallel straight line, whereas dynamic data is presented in periodic fluctuations. This makes it easy for the model to learn the difference between static and dynamic data. In addition to this, axis values of accelerometers in sitting and standing are very different from those in lying down. Accelerometer data from running and walking also have very different frequencies. These differences allow the model to classify each category very accurately. Figure 14 shows the confusion matrix for the 4-category model. Since the classification is 100% accurate, no more analysis will be done.

For the testing of real-time performance, three of our team members and a volunteer who was not involved in the IOT data recording worked together to check the real-time performance of the model. Table 3 shows the average results from our multiple tests. We can clearly see that the 4-category model is 100% accurate in terms of real time results. This is the same as our offline performance and indicates that the model has good generalisation capabilities.

	person1	person2	person3	person4
Sitting/Standing	1.00	1.00	1.00	1.00
Lying down	1.00	1.00	1.00	1.00
Walking	1.00	1.00	1.00	1.00
Running	1.00	1.00	1.00	1.00
Accuracy	1.00	1.00	1.00	1.00

Table 3: Real time performance for 4-category model

Table 4 shows the performance of the 14-category model. It performs the same as the 4-category model with an accuracy of almost 100%, except for minor errors in desk work and movement. Such excellent results were made possible by the fusion of the Respeck and Thingy dual devices, allowing the model to identify differences in each category from data from two completely different positions and angles. We found through experimentation that using only Respeck or Thingy data, the model is heavily confounded in some categories. For example, when using only the Respeck, the model cannot accurately distinguish between sitting, standing and desk work because the device is worn

on the stomach. When using the Thingy only, the device is worn on the thighs, making it impossible for the model to accurately differentiate between bent forward, bent backward, sitting and desk work. This means that fusing data from different locations and angles is important to accurately distinguish between different activities. The model we have designed takes advantage of this by fusing and integrating data from Respeck and Thingy to give the model the ability to summarise activity characteristics from multiple perspectives.

	precision	recall	f1-score	support
Climbing stairs	1.00	1.00	1.00	1696
Descending stairs	1.00	1.00	1.00	1695
Lying down left	1.00	1.00	1.00	1315
Lying down on back	1.00	1.00	1.00	1643
Lying down on stomach	1.00	1.00	1.00	1498
Lying down right	1.00	1.00	1.00	1091
Walking at normal speed	1.00	1.00	1.00	1696
Running	1.00	1.00	1.00	1810
Movement	1.00	0.99	1.00	1431
Desk work	0.99	1.00	1.00	829
Sitting bent backward	1.00	1.00	1.00	1563
Sitting bent forward	1.00	1.00	1.00	1355
Sitting	1.00	1.00	1.00	1222
Standing	1.00	1.00	1.00	1664
accuracy	1.00	1.00	1.00	20508
macro avg	1.00	1.00	1.00	20508
weighted avg	1.00	1.00	1.00	20508

Table 4: Classification report for 14-category model

Why the fusion of Respeck and Thingy data achieves this result is a question worth analysing. After our testing in real time, we have concluded that there are potential patterns in model recognition. Thingy is worn on the thigh, which makes its characteristics key to distinguishing between a range of activities based on sitting and standing. The Respeck is worn around the waist, which gives it angular information to differentiate well between upper body activities. When the model is able to distinguish well between sitting and standing, and can also distinguish well between the types of upper body activities, the model essentially has the ability to identify 14 different activities, as all activities can essentially be broken down into the two different joint sub-activities mentioned above. Figure 15 shows the confusion matrix for the 14-category model. Since the classification is 100% accurate, no more analysis will be done.

For the testing of real-time performance, three of our team members and a volunteer who was not involved in the IOT data recording worked together to check the real-time performance of the model. Table 5 shows the average results from our multiple tests. We can clearly see that the 14-category model

is overally 99% accurate in terms of real time results. This is similar to our offline performance and indicates that the model has good generalisation capabilities.

	person1	person2	person3	person4
Climbing stairs	0.95	1.00	1.00	0.95
Descending stairs	1.00	1.00	1.00	0.95
Lying down left	1.00	1.00	1.00	1.00
Lying down on back	1.00	1.00	1.00	1.00
Lying down on stomach	1.00	1.00	1.00	1.00
Lying down right	1.00	1.00	1.00	1.00
Walking at normal speed	1.00	0.95	1.00	0.95
Running	1.00	1.00	0.95	1.00
Movement	1.00	0.95	1.00	0.95
Desk work	0.95	1.00	1.00	0.95
Sitting bent backward	1.00	1.00	1.00	1.00
Sitting bent forward	1.00	1.00	1.00	1.00
Sitting	1.00	1.00	1.00	1.00
Standing	1.00	1.00	1.00	1.00
accuracy	0.99	0.99	0.99	0.98

Table 5: Real time performance for 14-category model

References

[1] Iveta Dirgová Luptáková, Martin Kubovčík, and Jiří Pospíchal. Wearable sensor-based human activity recognition with transformer model. *Sensors*, 22(5):1911, 2022.

[2] Sannara EK, François Portet, and Philippe Lalanda. Lightweight transformers for human activity recognition on mobile devices. *arXiv preprint arXiv:2209.11750*, 2022.

[3] Yeon-Wook Kim, Woo-Hyeong Cho, Kyu-Sung Kim, and Sangmin Lee. Inertial-measurement-unit-based novel human activity recognition algorithm using conformer. *Sensors*, 22(10):3932, 2022.

[4] Achille Nazaret*, Sana Tonekaboni*, Greg Darnell, Shirley Ren, Guillermo Sapiro, and Andrew C. Miller. Modeling heart rate response to exercise with wearable data. 2022.

[5] Yoli Shavit and Itzik Klein. Boosting inertial-based human activity recognition with transformers. *IEEE Access*, 9:53540–53547, 2021.

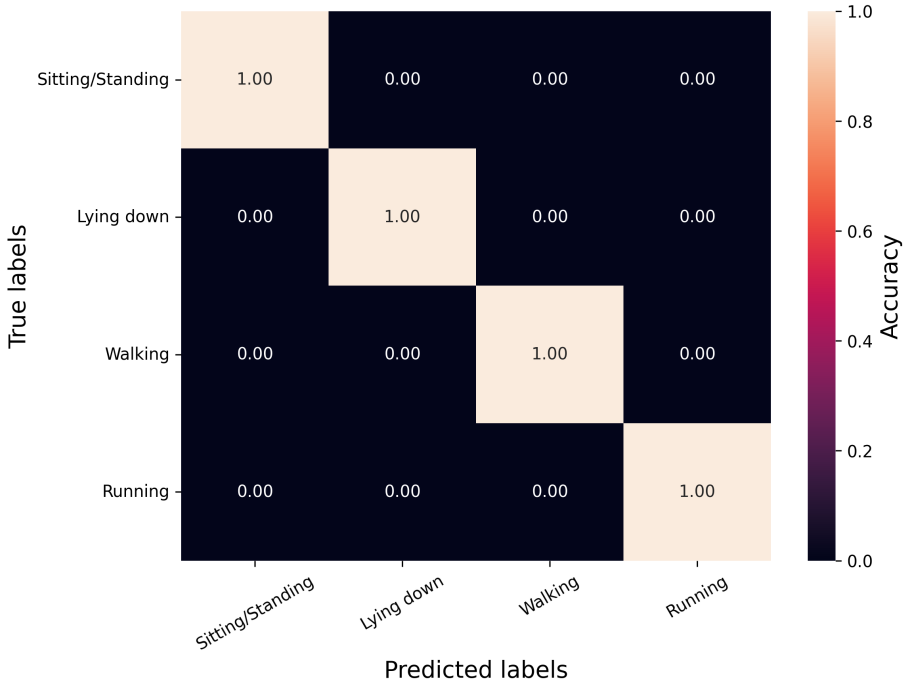

Figure 14: Confusion matrix for 4-category model

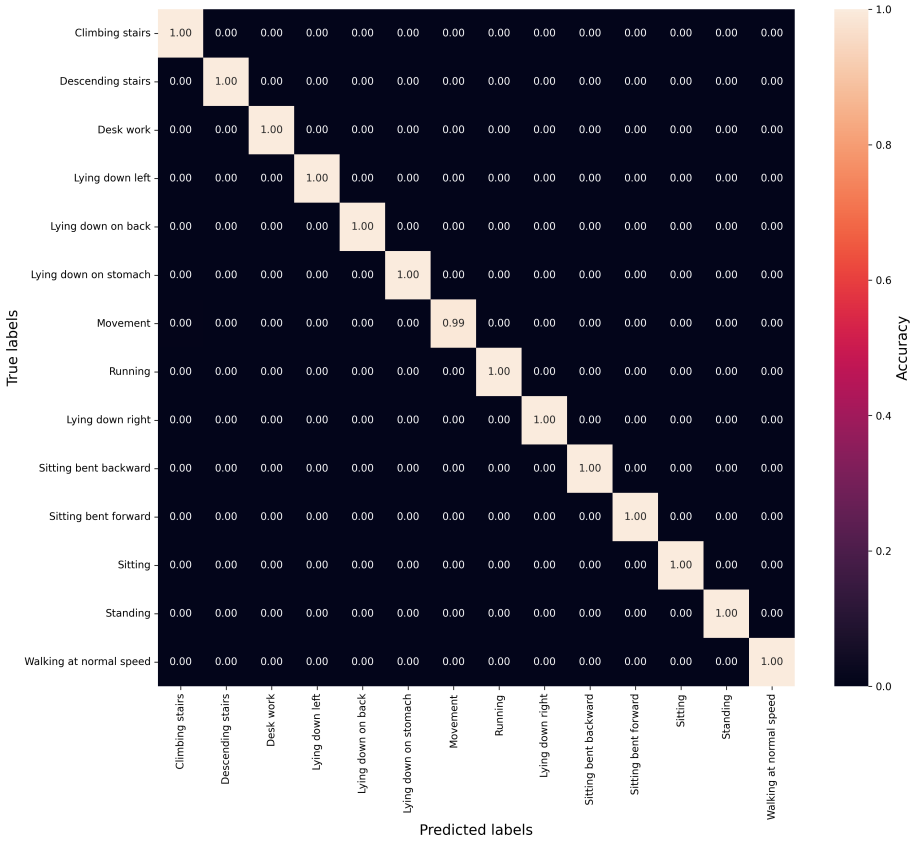

Figure 15: Confusion matrix for 14-category model