PDIoT Coursework 3

Group K1

Group Report
School of Informatics
University of Edinburgh

2024

Abstract

This project presents the development of an innovative Android-based application interfaced with the RESpeck sensor, harnessing machine learning to monitor and classify physical activities and respiratory patterns in real-time. The objective was to to develop a prototype to demonstrate the feasibility of using machine learning models, particularly TensorFlow Lite, for immediate activity recognition within a mobile application framework

The methodology involved the integration of the RESpeck sensor with an Android application, employing Bluetooth Low Energy for data transmission. The application was constructed in Android Studio, using Kotlin and Java, and was structured to pre-process sensor data before being classified by the TensorFlow Lite model. Testing was conducted across multiple Android devices to ensure functionality and performance.

Results from the prototype testing indicated successful data communication and classification of activities and critical analysis of the model revealed high testing scores and in cases of discrepancy, provided explanations on any misclassification that happened during activity recognition.

In conclusion this prototype represents good progress in mobile application activity recognition which could be applied to several different systems such as in the fitness and the healthcare industries, however the latter may have a regulatory barrier to entry for medical device approval.

Looking forward, with access to higher performance computing or simply more time, we would be able to increase the rigour of the testing process to improve the model beyond it's current capabilities and investigate new heuristics or model changes and their effects.

0.1 Contributions

0.1.1 Tom Tudor S2050574

- Developed the real-time classification portion of the App
- Report Writing - (Sections 3.1,3.2,3.3,3.5,3.6, 5.2)
- Submission Preparation (1,2,3,4)
- Designed and developed the UI of the app
- Real-Time classification testing and troubleshooting
- App performance testing
- Developed the UI of the App's main menu and live activity page.

0.1.2 Chris Crampton S2020491

- The design of the CNN Model
- The coding implementation of the CNN Model
- Offline testing of the CNN Model
- Online testing of the CNN Model
- Implementation of the data pre-processing mechanism
- Implementation of the data post-processing mechanism
- Analysis and Improvement of the CNN model
- Report Writing (Sections 2.2, 4.4, 4.6.2, 5, 6.1, 6.2)
- Submission Preparation (1, 5, 6, 7)

0.1.3 Marley Adair S1689282

- Integration of ML model with the Android app
- Pre- and post- processing of data on the Android app
- Pre-processing of gradient features for the model during offline training
- Real-Time classification troubleshooting
- Report Writing (Sections 1.1, 1.2, 1.3, 2, 3.4, 3.5, 4)
- Submission Preparation (1)
- Demonstration Slides

Table of Contents

Chapter 1

Introduction

1.1 Project Aims

The goal of this project was to develop an application for Android smartphones that performs human activity recognition and breathing pattern recognition via data from a *RESpeck*, a waist-mounted tri-axial accelerometer and gyroscope. In order to achieve this, our goal was to train a machine learning model based on data collected and provided by the 2023 PDIoT class, and to create the software required to connect to the RESpeck sensor, read data from it, process that data, classify it, and display it to the user.

1.2 Brief Description of the Method Adopted

The model designed is a Convolutional Neural Network (CNN) that takes three second time slices of data across whichever sensors are being used by the RESpeck device, and outputs two separate classifications, one for respiratory symptom and the other for activity being carried out. The process involves generating three 2D-arrays representing the raw data, a Fourier Transform of the Data, and the derivatives between consecutive points. Each of these are analysed independently. The results are concatenated, analysed some more then split off.

For the application design, the provided App for data collection was used as a base. This allowed for the Bluetooth connection and reading and parsing of data from the RESpeck to be implemented quickly, as it only required minor modifications to the existing code base. The main addition to the application that was made as part of this project is the analysis activity, which stores the incoming data in a buffer, and reads it in windows of three seconds, passing it through a pre-processor for feature extraction, and then to the classifier model discussed above. Once classified, the result is merged with a class determined by some hand-crafted heuristics during post-processing, before it is finally displayed to the user via a text box on the screen. A graph of the sensor data is also displayed on the screen, as that was already implemented by the original app.

1.3 List of Physical Activities Used

Our model performs two classifications: physical activity and respiratory style. The classes for each are listed below. Note that data for all pairs is not available; respiratory patterns were only trained on static physical positions. As such, the application will not display that classification when partaking in other physical activities.

Physical activity classes:

- Upright Stationary (Sitting or Standing)
- Lying Down on the Left Side
- Lying Down on the Right Side
- Lying Down Face Up
- Lying Down Face Down
- Walking
- Running or Jogging
- Ascending Stairs
- Descending Stairs
- Shuffle Walking
- Miscellaneous Movements

Respiratory pattern classes:

- Normal Breathing
- Coughing
- Hyperventilating
- Other (Trained on Singing, Laughing, Talking, and Eating)

Chapter 2

Literature Survey

Human activity recognition (HAR) is a well studied field. Many resources exist in literature that look at the question of how to classify activities using on-body sensors. Below, a selection of past attempts to do this are presented, showing that a wide range of sensor and classification models have been used before. While many of these studies performed data collection in order to train and test their models, many also used pre-existing datasets of human activity. This allows some of the various approaches to be compared against each other. Most notably, many of the studies use the University of California, Irvine's Machine Learning Repository's Human Activity Recognition Dataset (UCI HAR), which contains data for six activity classes (walking, ascending stairs, descending stairs, standing, sitting down, lying down).

Mannini et al. [13] performed activity classification using on-body accelerometers using a variety sensor locations and a variety of single-frame classifiers, including Naïve Bayes, GMM, Nearest Mean, SVM, Binary decision trees, ANN, and k-NN. The activities classified were all physical positions and actions, such as sitting, standing, walking, running, and cycling. These were compared and had a range of accuracies, ranging from 92% for GMM, to 98.5% for nearest mean.

Nunavath et al. [17] classified activity data from a single tri-axial accelerometer using recurrent neural networks (RNN). They used both data from the UCI Machine Learning Repository, which contains data from wrist-worn accelerometers, as well as another dataset that contains data from waist-worn accelerometers. The study found that RNN could classify the activities with an accuracy of 84.9% and an F1-score of 82.6%. These authors claim this is comparable to other "state-of-the-art methods" and suggest it could be used in medicine to efficiently keep track of patient activity.

Sani et al. [19] performed activity classification from data collected by accelerometers worn on the wrists and thighs. They compare three types of feature representations, shallow hand-crafted, shallow frequency transformed, and deep CNN derived. These are compared using deep learning classifiers, as well as hybrid classifiers (CNN + SVM and CNN + kNN.) The study found that CNN + SVM obtained the best F1-score of 85.0%. They also comment on various possible pre-processing steps for the raw data, including Fourier Transforms and discrete cosine transforms.

Allen et al. [2] investigated two classification methods based on a GMM and a heuristic rule-based method to identify basic activities, such as standing, sitting, lying, and walking, using a tri-axial accelerometer on the waist. The results were that the GMM system was able to achieve an accuracy of 91.3% when classifying between four activity states (sitting, standing, lying down, and walking) and four state transitions (standing to and from both lying and walking, versus the 71.1% achieved by the heuristics system. The data used was collected by the researchers from six healthy but elderly participants.

Jain et al. [10] studied the feasibility of using tri-axial accelerometer and gyroscopic data from a smartphone carried on a person to perform activity classification. However, they did not collect any data from such devices for evaluation, instead using the UCI HAR dataset. As well as the raw data, they also included pre-processing to add additional features, including magnitudes of the acceleration and angular velocity, Fourier descriptors. and histograms of gradients. They tested both SVM and k-NN classifiers on two subsets of features, as well as all features. In these tests, SVM performed far better than k-NN, reaching an accuracy of 97.1%, versus k-NN's 84.0%.

Minarno et al. [16] also studied the use of smartphone accelerometer and gyroscope data for activity classification. Much like Jain et al., they used the classes and data from UCI HAR, however they also collected their own data from thirty participants. They perform no feature reduction or creation on this data. They compared many classifiers: k-NN, logistic regression, random forest, SVC, EVC, and extra trees. Logistic regression provided the highest accuracy of the models tested, at 98%. They also show that the most common errors made by the model were confusion between sitting and standing.

Ha et al. [8] proposed an approach for using a CNN to classify physical activities from multi-modal data collected by several accelerometers and gyrosocpes mounted on the body. They demonstrate a 2D convolutional model called CNN-pff, trained and tested on data from mHealthDroid, that has a 99.67% accuracy. This model uses partial weight sharing in the first layer, and full weight sharing in its second and third layers, hence the name. The accuracy is compared against that of other models on the same dataset, such as SVM, which achieves 65%; HCRF, which achieves 69%; and a traditional CNN, which achieves 90%.

Webber et al. [23] studied the trade offs of fusing data at various points in the classification process when handling multiple types or locations of sensors. They used four datasets, MobiAct, UMAFall, IM-WSHA, and UCI HAR, and tested each against models that combine data directly, combine features after extraction, and combine decisions after classification. Of these, decision-level fusion outperformed both other options in all cases, however this came at a significant computational resource cost. As a result, they suggest using a Kalman filter, which allows classification using far less resource while still maintaining a reasonable accuracy.

Masum et al. [14] uses dense neural networks to perform activity classification over 10 categories. The data was collected via smartphones. The study compares the results of the DNN to various other models, such as SVM, k-NN, naïve Bayes, and random forest. They use principal component analysis to perform feature selection. They found that dense networks were able to achieve an accuracy of 94.8%, higher than any of the other methods tried. Unlike most HAR studies, which use sensor data taken at a frequency of between 20 and 100 samples per second, this study uses only 1Hz data. The study also used separate datasets for male and female participants, and provides separate accuracies for each.

Hassan et al. [9] compares various deep belief networks (DBN) in the Taskof classifying physical activities, based on data collected from participants wearing accelerometers and gyroscopes on their bodies. They produce a table that shows the effectiveness of various sizes of neuron layers in the network in performing accurate classification. They found that for the dataset they collected, the optimal size of network was two layers of roughly 40 nodes each, which achieved an accuracy of 97.5%. If too few nodes were used, for example around ten per layer, accuracy dropped to around 90%, while too many (around 500) lead to overfitting, which rapidly caused the accuracy to drop to around 1%.

Bevilacqua et al. [3] used a convolutional neural network to perform classification of activity data collected from participants and labeled into classes based on the Otago exercise program. The researchers used a 2D convolutional network to perform classification. The model consisted of three 2D convolutional layers, separated by max pooling layers, and a final dense layer for output. The study then trained the model based on various subsets of the available sensors. They found that using two or three separate sensors was optimal for performing activity recognition versus resource cost.

Kanjilal et al. [12] compares machine learning approaches to human activity recognition with hand-crafted heuristics. They show that while manual feature extraction is still viable, deep learning provides effective tools to quickly create classification models from large amounts of data, and still achieve high accuracy. They based the study on the UniMiB-SHAR dataset. They show that for certain tasks, such as fall detection, deep learning provides a significant edge over classical algorithms.

Jiang et al. [11] compares two approaches to deep convolutional neural networks to each other and to SVM in the Taskof human activity recognition. Across three datasets, UCI HAR, USC, and SHO, they find that the DCNN consistently match outperform SVM in accuracy. The DCNN both also significantly outperform SVM in resource usage, so the study suggests they be used for performing classification on lower power devices. Additionally, they find that the DCNN models can be made less resource intensive at the cost of accuracy.

Bhuiyan et al. [1] presents a feature extraction method for activity classification from tri-axial accelerometer and gyroscope data based on enveloped power spectrum and linear discriminant analysis. They suggest this as a way of reducing or eliminating the required amount of deep learning for classification, suggesting it as a tool for low-power environments. The study compares it to common general techniques, such as Fourier Transforms, and shows that is is able to obtain high accuracy predictions, for example of 97.1% when used with SVM. Like many HAR studies, they use the UCI HAR dataset for evaluation of their models.

Additionally, there are many studies in literature about the use of worn accelerometer sensors for the analysis of breathing patterns. These are often done for the purpose of analysing respiration during sleep, in order to study conditions such as sleep apnea.

Ryser et al. [18] built a classifier for respiratory patterns that used data from a chest-worn accelerometer. This gave a binary classification for healthy versus abnormal breathing, with an accuracy of 76%. Data for evaluation was collected by participants both wearing the accelerometer and connected to a polysomnogram. The main advantage of the classifier presented is a novel feature included they call the "respiratory peak variance".

McClure et al. [15] used chest and abdominal worn tri-axial accelerometer and gyroscope data to classify breathing patterns. Their model was a 1D convolutional neural network that classified sensor data into several classes, including normal breathing, yawning, sighing, and some forms of sleep apnea. They measured the mean F1-score for each class, with results ranging from 51% for obstructive sleep apnea to 92% for normal breathing.

Tarim et al. [21] also used on-body tri-axial accelerometers to build a model to classify breathing patterns for the purpose of assisting with "the diagnosis and continuous monitoring of sleep apnea." Their model uses a neural network trained for 230 epochs to classify the state of the user's breathing. In testing, it had an accuracy of 96% and a specificity of 80%.

Doheny et al. [6] used wearable accelerometers to classify both breathing patterns and body position during sleep. The model they created was able to determine the subject's physical position with a ROC AUC of 0.94, and the maximum mean absolute error for the estimation of respiration rates was 2.67bpm.

From this, it is clear that there has been a large amount of prior research in the fields of human activity recognition and respiratory analysis. Many different classification models have been trained and compared across a large number of datasets and sensor varieties. Deep learning occurs a lot as a suggested model, having consistently high accuracies in its classification performance in both areas. The accuracies achieved by these models are also fairly consistently high, suggesting that the problem at hand is one that can be solved with modern tooling to an effective degree.

Chapter 3

Methodology

3.1 Description of System and Implementation

Our system is an Android-based activity detection application that utilises a RESpeck sensor via Bluetooth to monitor and classify user's physical activity and respiratory patterns. Developed using Android Studio, the application makes the most of the capabilities available from the TensorFlow and TensorFlow Lite libraries to process and classify the recorded data.

The application is structured to allow for a seamless flow from the recording of the data via the accelerometer and gyroscope found in the RESpeck sensor to the classification of the data. Raw sensor data is first collected and then pre-processed, where it undergoes Fourier Transformation to extract the amplitude information, and then then is further processed to derive meaningful features. The last step of the pre-processing stage is to normalise all of the data to standardise it, ensuring consistency and reliability in the input that is fed into the classifier model.

The classifier itself is a complex structure consisting of multiple layers. It features multiple parallel pathways that utilise 2D convolution layers and max pooling operations, which are used in feature extraction and dimensionality reduction. Following these layers, the data pathways converge, and the information is merged before passing through dense layers that are used in pattern recognition within the data. The model culminates in two output streams, one for the current respiratory pattern, and one for the current physical activity.

The implementation of our system and model was driven by a design philosophy that prioritises accuracy and efficiency. The architecture of the App ensures that the application performs optimally on a range of Android devices without using too great a share of the device's resources. In summary, the system is a blend of relatively advanced data processing techniques and machine learning, all housed within a user-friendly mobile application. This approach ensures that the App not only meets the technical requirements of activity recognition but also as a prototype in its early stages, it is adaptable and scalable across various Android platforms and versions and allows for easy modifications should the need arise.

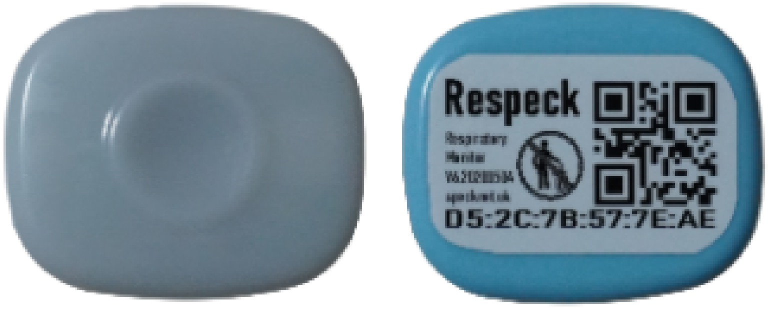


Figure 3.1: The RESpeck Version 6 sensor

3.2 Hardware and firmware

In developing our activity detection and recognition application, we focused on two main pieces of hardware: a basic smartphone and the respiratory sensor known as the RESpeck monitor device Version 6. It can be seen in Figure 3.1. The sensor allowed us to acquire the needed data via its onboard accelerometer and gyroscope. The sensor integrates a Freescale MMA7260QT three-axis accelerometer and a trio of Analog Devices ADXRS300 gyroscopes[7]. The RESpeck sensor was mounted on the front of the subject's chest, on the left lower costal margin, right below the ribs as seen in Figure 3.2. The sensor is small and light (35mm x 45mm; 18g)[5]) thus it is easy for people to wear without feeling uncomfortable. The plastic shell is also colour-coded and marked to ensure that the sensor is worn the right way every time, which guarantees reliability during testing.

Linear acceleration is the rate of change of the velocity of an object. The MMA7260QT accelerometer measures the linear acceleration in m/s/s across three orthogonal directions, referred to as `accel_x`, `accel_y`, and `accel_z` in the code. The accelerometer can detect its orientation in the 3 directions by measuring the amount of acceleration due to gravity, called static acceleration, which allows it to identify which direction is down towards the ground. With this knowledge, combined with measuring the amount of acceleration caused by external forces, called dynamic acceleration, the accelerometer can identify the direction it is moving in, allowing for our reading.

The gyroscope on the other hand is a MEMS gyroscope (Micro-Electro-Mechanical Systems). The gyroscope operates on the principle of a vibrating structure. When the device experiences rotational movement, the Coriolis effect comes into play. This causes a vibration in the structure that is perpendicular to the direction of rotation. By measuring the vibrations, the gyroscope can accurately calculate the rate of rotation around its axis. The RESpeck features three such gyroscopes allowing for angular velocity to be measured in all directions.

The application is designed to run on Android devices running on Android 6.0 as a minimum and is targeted at Android 11.0. Our goal was to ensure broad accessibility thus the App is not resource intensive allowing a wide range of smartphones to be

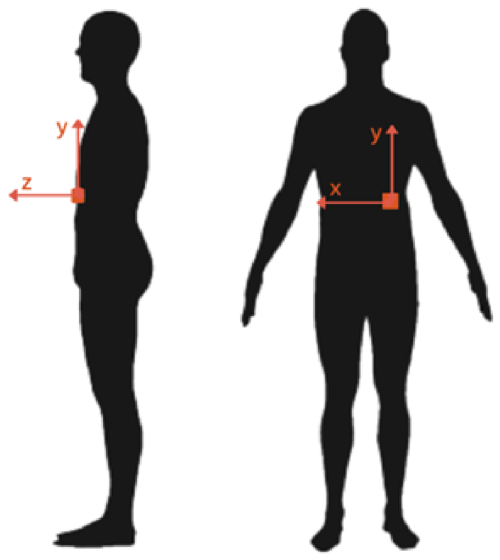


Figure 3.2: Placement of the RESpeck sensor with axis direction definitions shown. Acquired from a paper by Charalampous[5]

compatible ensuring smooth operation across a wide range of devices. The application incorporated TensorFlow Lite to process the data received from the RESpeck sensor using our machine-learning model. With some amount of pre-processing, this setup allows for efficient on-device data processing ensuring a quick response to the user without the need for an external server.

3.3 Wireless communication

In our activity detection and recognition application, the RESpeck sensor that we used utilises Bluetooth Low Energy (BLE) which allows for data to be sent quickly, and at a low power cost. BLE also allows for relatively long ranges of 50-100m[4] and the packets sent over Bluetooth are reliable and not prone to error. The wireless communication code was primarily taken from the PDIoT data collection App used in Coursework 1 with some minor modifications, to allow the gyroscope data to be more easily accessed.

The `BluetoothSpeckService` class within our Android application plays an important part in the communication process. It scans for the RESpeck sensor using a provided device ID, entered manually or via a QR code found on the sensor. Upon successful detection, a BLE connection is initiated. The data collected from the sensor is then sent via real-time transmission at a rate of 25 Hz. Each packet sent from the RESpeck sensor to the smartphone features a timestamp, the accelerometer data in the x, y & z directions and lastly the gyroscope data in the x, y & z directions.

Upon receipt of the packets, the App then utilises a packet handler and packet decoder. These files allow for the decoding of the raw byte stream and convert it into meaningful, quantitative measurements of acceleration and angular velocity. The packet decoder has been specifically made for the RESpeck V6 device. Using the BLE communication protocol of the RESpeck sensor creates a robust, efficient and user-friendly system that aligns with our project aim. The implementation of this wireless communication setup not only adheres to the technical requirements of the sensor but also maintains the efficiency and reliability of our application.

3.4 Algorithm for activity recognition

Data read from the RESpeck is written into a circular buffer, and every 75 data points (3 seconds due to the 25Hz sampling rate from the RESpeck) the classifier is run on those data. The classifier itself contains four main steps: feature extraction, standardisation, model evaluation, and post-processing.

The initial pre-processing steps are used to transform the input in order to produce more features and normalise the results, ready for the model. The six streams of raw data (three acceleration axes, three gyroscopic axes) are still retained as features. In addition to those, a Fast Fourier Transform is run independently on all six streams, from which we extract the amplitude and phase. We then discard the phase as it does not have any bearing on the type of activity, just when it is done. As such we only use the Fourier amplitude for the model inputs as that denotes the frequency of the spikes which is a major distinguishing feature between the classes of data. For example, the difference between hyperventilating and normal breathing is very apparent in frequency analysis.

Additionally, the gradients of each stream are taken as features. As the data is coming directly from a sensor, there is potentially a large amount of noise present, which would be exaggerated by taking gradients. In order to minimise this, the raw data is smoothed using convolution-based smoothing before the gradients are taken. The aim of this is to further extract distinguishing features between some types of data that may have similar frequencies but different wave amplitudes. Note that the amplitude of a wave is distinct from a Fourier Transform amplitude which refers to the frequencies of the wave.

After these steps are performed, a model input has been produced consisting of three batches of $75 \times 6 = 1350$ features (Raw Data, Fourier Amplitudes, Derivatives). It should be noted that each of these features is an entirely separate input. As a final pre-processing step, each feature is standardised based on the mean and standard deviation of all the values it takes in the training data. This is calculated by 6 2D arrays that correspond to the mean and standard deviation for each of the three input batches for all pieces of the training data in the same location. Each new element (i,j) in a new batch of data is then standardised by the corresponding elements in the respective mean and standard deviation arrays for that input type. These values are saved as a .tfrecord and loaded for each model run. These standardised features are then given to the classifier model.

Upon review of an article explaining some of the limitations of choosing RNNs [20] we were steered towards using a CNN that has time segments as a dimension. These RNN limitations included weaker parallelisation due to their sequential nature, weaker representational power and difficulty in explaining results. The weaker representational power, referring to a neural networks ability to approximate an ideal classification function, was a deal breaker given the majority of the accuracy metrics we were concerned with relied on offline and independent testing. The parallelisation flaws also dissuaded us from the RNN approach as we knew there would be a lot of testing during development, and with lack of access to a high performance computer, it was important that we could take advantage of performance improvements where possible.

After that, the model has two consecutive convolutional layers and max pooling layers for the raw data and gradients. Due to a mild discrepancy in frequency bins from the Fourier transform with certain classes, we applied a preliminary max pooling layer of length five across each stream to remove insignificant variation amongst roughly similar frequencies that represented the same activity. The idea behind this was that for short and sharp peaks, like with coughing, there could be some variation in which frequency bin a particular data point would fall into, but all of the bins it could feasibly fall into would be still be within a certain range with high probability. This more coarse approach for feature extraction from the Fourier Transform amplitudes was inspired by a paper investigating the performance improvements when taking into account coarser depictions of the data in parallel with more fine analysis [22]. Each of these is then connected to a dense layer, before all the inputs are concatenated and then followed by two more dense layers. To improve accuracy and aid with testing we elected to separate the model into two separate outputs: Respiratory Symptoms and Activity. Both outputs have a final dense layer of their own before the final layer which calculates the probabilities of each class. This has been visualised in the figure below.

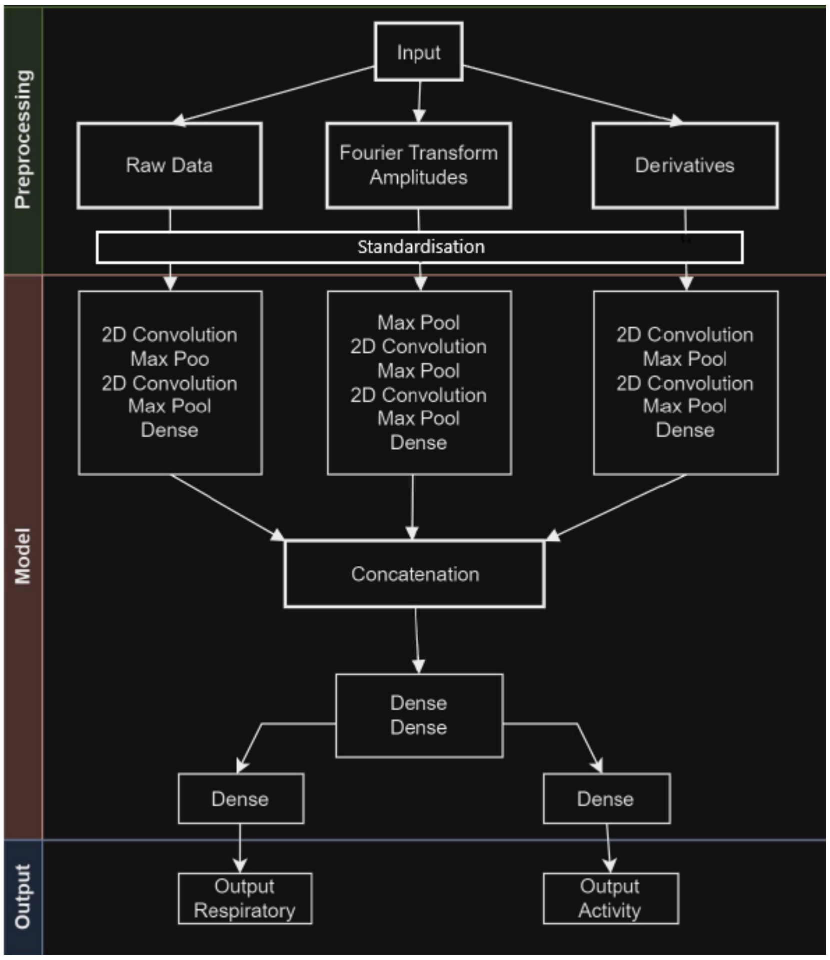


Figure 3.3: Diagram of the models structure including pre-processing and classification

After the model has provided a pair of output classes, a layer of post-processing is performed. During year-wide testing, it became apparent that the model would often struggle to differentiate between the various static classes of activity (Sitting/standing, and the orientations of lying down.) As a result, it was decided to merge these output classes from the model into one, and then decide on which classification was meant based on hand-crafted heuristics. To produce these heuristics, the assumption was made that during a static activity, most of the detected acceleration would be due to gravity. Therefore, the classification is chosen based on the sextant of the volume that the average acceleration over the duration of the input sample falls in. That is to say there will be a spike corresponding to gravitational acceleration which will be positive or negative across the three axes. This can be used to orientate the RESpeck and from there determine if the wearer is upright or lying down, and if so in which orientation they are lying. All other classifications are left as is, simply as the probability value associated with that output layer.

3.5 Mobile application and software organisation

Our application is structured into a two-pronged architectural framework, with a clear distinction between the real-time data handling and the analysis phase that follows. The application's organisation is centred around modularity and functional clarity, ensuring each component performs a distinct role in the activity detection process. An overview of the system architecture can be found in Figure 3.4

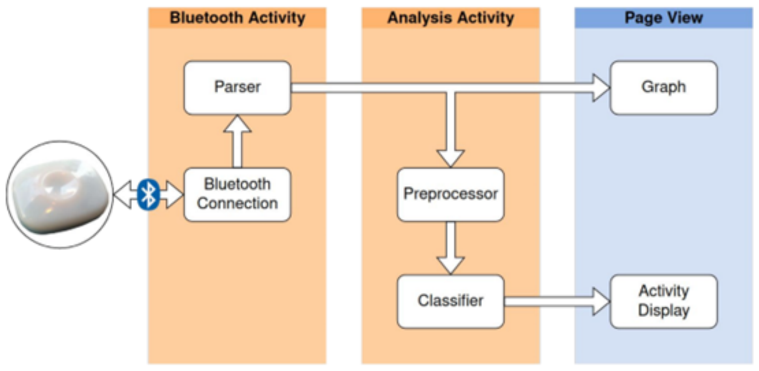


Figure 3.4: Visual representation of the application's architecture

Firstly, the Bluetooth Activity component is the initial point of contact with the RESpeck sensor. It is responsible for establishing and maintaining a BLE connection as mentioned in the Wireless Communication section above. This module encapsulates a parser that interprets the raw sensor data into a usable format. Once the byte-encoded data has been converted into quantitative measurements, the data is then ready for pre-processing before analysis.

This module's design ensures a user-friendly experience when pairing the sensor, as the RESpeck can be paired either with a direct ID input or by scanning a QR code using the smartphone camera. This simplifies the pairing process allowing for ease of use, and the LED found on the RESpeck allows for the sensor to give its status allowing for relatively simple troubleshooting.

Following this, once the data flow from the sensor has begun, it is sent to the Analysis Activity. It is comprised of a pre-processor and a classifier. The preprocessor applies the necessary transformations to the incoming data, such as the Fourier Transformations and the calculated differentials. Once pre-processed, the data is then sent to the classifier. The classifier employs a TensorFlow Lite model which was previously constructed and trained using Python. The model is then run with the data to execute the live activity classification. The buffer containing the streams of raw data that is then pre-processed and use for classification is updated every two seconds with one seconds worth of data (25) overlap between consecutive predictions. This figure was chosen to make the app as lightweight as possible.

The user interface, designed to be simple and efficient to use, features two main pages plus a home navigation page. Firstly, the live classification page presents the classification results to the user through the Activity Display, which is updated every

2 seconds as mentioned above. Complementing this is a Graph view that provides a visual representation of the sensor's accelerometer and gyroscope data, allowing users to see if the sensor is working as intended. It allows the users to see the movements captured by the RESpeck sensor. This visual feedback is not just functional but also engaging.

The other page of note is the connection page. It is the page responsible for establishing communication with the RESpeck sensor. It was taken from the initial PDIoT data collection application. The connection process is easy to follow via on-screen instructions and can be initiated via a direct ID, or by scanning a QR code.

The development environment for the application was chosen to support the integration of both Kotlin and Java since both languages can be utilised in Android App development due to their great compatibility and performance with the OS. The TensorFlow Lite model, crucial for the app's classification capabilities, was transferred from its Python development environment to the mobile application with minimal issues thanks to the compatibility between the Python TensorFlow library and Kotlin/Java's TensorFlow Lite package.

Throughout the development process, a keen focus was maintained on ensuring that the application remained lightweight and non-resource intensive. This consideration allows the App to maintain compatibility across a broad range of Android devices, ensuring that the widest possible user base can engage with our system without the need for high-spec hardware.

Our application's structure reflects a clear separation of roles. The Bluetooth Activity is responsible for establishing, maintaining and decoding the RESpeck sensor connection and data, whilst the Analysis Activity is tasked with the core functionality of data interpretation and user interaction. Compartmentalising the app, allows it to be maintainable and scalable. It also makes sure that each component can be modified without affecting others in potentially negative ways.

The user interface design is kept simple, granting clarity and user-friendliness during App interaction. The live classification page has been designed to be plain and instinctive, offering prompt and clear responses to the user's ongoing activity at a consistent rate. On the other hand, the connection page allows the user to connect to a non-standard Bluetooth device with minimal issues due to its simple instructions and interface.

3.6 Testing

3.6.1 Application

For our activity detection App prototype, the testing approach was tailored to basic functionalities and to make sure that the application's stability and performance were at an acceptable level. Whilst certain testing methods such as unit testing and large-scale usability testing were not conducted due to the application only being a prototype and featuring pre-made components, we still performed focused testing to ensure the application's stability.

Our primary testing was that of integration testing. This was essential in finding out if the pre-existing Bluetooth connectivity component worked as intended with the newly developed live activity detection component that we developed for the application.

To make sure that the application functioned correctly across different devices and operating system versions, we tested the application on multiple different Android devices and Android emulators. We found that the App worked as intended on all the different systems and application behaviour was not affected by the operating system version.

Most of the testing however was that of testing the accuracy of our live detection feature. Testing of the model found it to have a relatively high accuracy in tasks 1, 2, and 3, detailed in the analysis section. The model deployed in the application also utilised the gyroscope as well as the base accelerometer to improve its accuracy further. Despite this, the application required several rounds of testing and modification before the live activity detector produced acceptable results. The issue arose due to the differences in the pre-processing of the data done when training and testing the model and pre-processing the data within the application during live readings. The differences in pre-processing were due to the differences between Python and Kotlin's libraries involving Fourier Transformations and the differences in the shape of the data between Python model training and the live data.

Results of performance testing found that the classifier only takes 16ms to classify 3 seconds of data. The responsiveness of the App is particularly significant since the App is expected to function in real-time scenarios, processing and classifying activity data to provide immediate insights.

Power consumption was categorised as ”Low” by the Android Studio benchmark indicating efficient energy use. This characteristic is beneficial for prolonged use as efficiency in power usage is important for user convenience. In particular battery usage is important when considering the type of user who may make use of it and the different durations of when they may use the app.

Post-testing, CPU usage stood at 46 million cycles, occupying 19% of the test device's processing power during active use. This moderate level of CPU utilization suggests that the app's computational operations such as the pre-processing and classification via the TensorFlow Lite model, are optimised for performance without overburdening the device. Similarly the application's RAM usage was only 177MB. Most modern smartphones have between 4GB-12GB of RAM. The relatively low RAM usage thus indicates that the App can be used at its full potential even in lower spec devices that may not run with the most modern hardware. This optimisation also ensures that the application can run along side other apps, without hindering the performance of the smart device.

Overall, the performance testing results affirm that the application is well-optimised for real-world usage as it is fast, efficient and non resource-intense. The app's design makes sure that necessary functionality is delivered without compromising the user's experience. This balance between resource consumption and App performance is crucial in not only producing fast, reliable activity recognition but also makes sure that sustained use of the application is possible on a range of devices.

3.6.2 Model

We tested multiple variations of the model to ascertain small improvements to the overall accuracy. As we were limited in processing power and time we compared a control model without a modification, and the same model with said modification on five runs each. For each run we used the standard 80-20 split of training data to test data and ran each model with 30 epochs. We then graphed the validation accuracies versus the training accuracies to determine at what point overfitting occurred. We used the highest validation accuracy recorded before overfitting as our measurement and compared the models.

We tested standardising the input and found the improvement to be 4.89% on average. When testing the difference made by adding a max pooling layer before the first Fourier convolutional layer, we found there to be an improvement of 2.21% on average.

The decision behind three second time intervals was the result of manual inspection of graphical representations of the training data. The trade-off was that the model would be far more efficient to run and test with smaller input shapes due to the decrease in nodes used, but it needed to be a large enough window that it could capture enough important features of the data to perform a well informed prediction. We found there to be negligible performance improvement when using windows of 100 data points (four seconds) which rendered it undesirable due to the increased processing required. 50 data points (two seconds) caused a significant accuracy drop off which rendered it undesirable as it appeared to have no hope of effectively classifying the data with such a small window.

Chapter 4

Results

Offline analysis of the task 4 classifier using leave-one-out testing across each student resulted in an accuracy of 81.63%. We ensured that between runs the model was reset to prevent test and training data contamination. This process made use of early stopping to keep the weights of the model that performed the best on validation data in each epoch. This used a patience value of 5 which means that if there is no improvement on validation accuracy in five epochs time, it automatically stops and uses the best weights recorded so far.

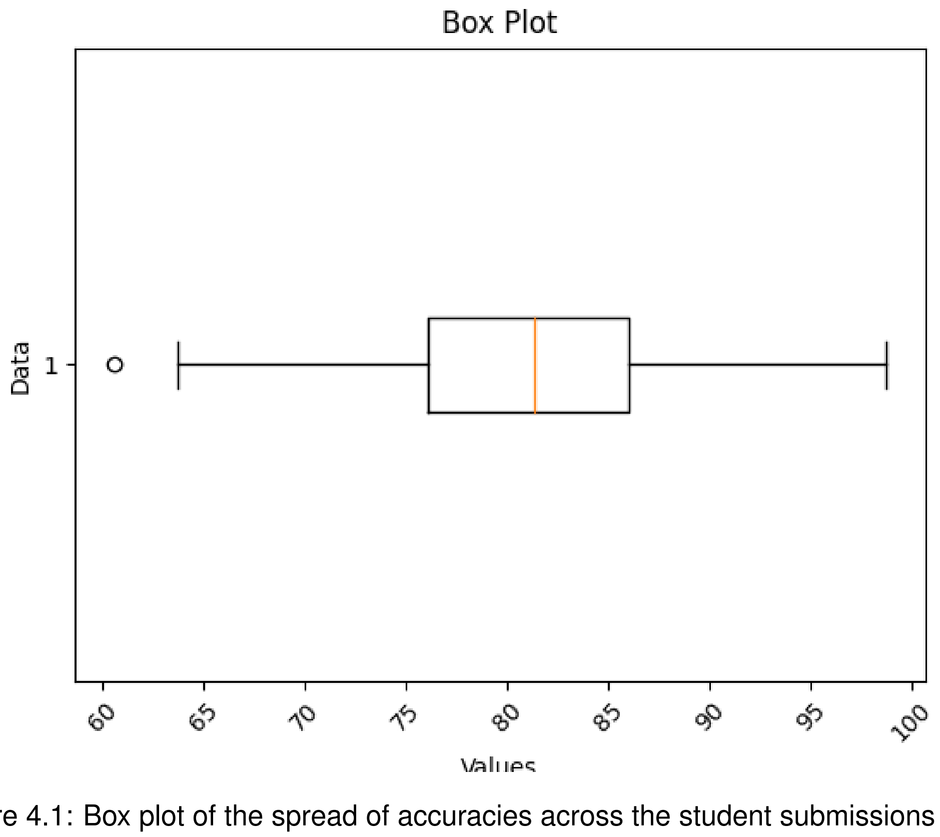


Figure 4.1: Box plot of the spread of accuracies across the student submissions for task 4

The results for the LOO analysis of task 4 have a mean of 81.63% and a standard deviation of 8.29%. This shows a large difference between student data sets which seems too large a disparity to be merely put down to noise. Obviously, there will be some variance between students but the gap between the minimum of 60.3% and 98.2% suggests a significant difference in some aspect of the data collection, whether that be the sensor placement, sensor reliability, fitness or something else entirely.

The results of the LOO analysis of task 1 have a mean accuracy of 95.31%. This is significantly higher than all the other tasks. As this task is the only not not include the respiratory class as an output, this suggests that physical activity classification, even with the more classes present in task 1, is a much easier challenge for the CNN to perform. It is possible this is a result of the higher amounts of physical motion in these activities providing more useful signals to the model, as the range of input values received from the sensors will be much larger.

The results of the LOO analysis of task 2 have a mean accuracy of 84.61%, and task 3 has a mean accuracy of 80.40%. This shows that the introduction of the new “Other” respiratory class leads to a significant reduction in accuracy. It is possible that this is amplified due to the larger range of breathing patterns that fall under it (Singing, laughing, eating) compared against the other classes, which all contain only one style each.

The reported accuracy for task 3 also suggests that the use of the gyroscope to provide additional features has a fairly small influence on the accuracy, since it increases only by 1.23 percentage points. Since these tasks consist only of classifying static positions, performing these activities does not induce much angular velocity, as breathing does not rotate much the position of a correctly attached sensor.

4.1 Critical Analysis

In order to gain a better understanding of the model’s results, a confusion matrix was generated for each task based on a provided sample set of data. This makes visible where issues are occurring that lower accuracy. The confusion matrix for task 1, shown in Figure A.1, shows that the classifier for task 1 is highly accurate. The only mistakes it makes are confusing shuffle walking for standing, and confusing shuffle walking and ascending and descending stairs from misc. movements.

It is understandable why these errors would occur. Shuffle walking is a slow movement, which may not be picked up well by the sensor. As such the resulting data will have strong similarities to that of someone standing still. Miscellaneous movements is a catch-all for unknown motion. As such, its decision boundaries are likely messy, and will result in cases that cannot otherwise be classified falling within them. This includes people carrying out other listed activities in a more extreme or exaggerated fashion, albeit differently.

In task 2, the confusion matrix (shown in Figure A.2) shown an interesting pattern: there is a faint diagonal, parallel to the main, but shifted five cells to the right. This shows that the classifier is sometimes unable to determine the difference between coughing and hyperventilating, however the activity classification within those cases is still accurate.

This pattern is continued in task 3, with various other 5-long diagonals appearing in the confusion matrix. (Figure A.3) This suggests that the activity classifier is more accurate than the respiratory one. Here, it is clear to see that the strongest confusions happen between “other” and rest of the classes.

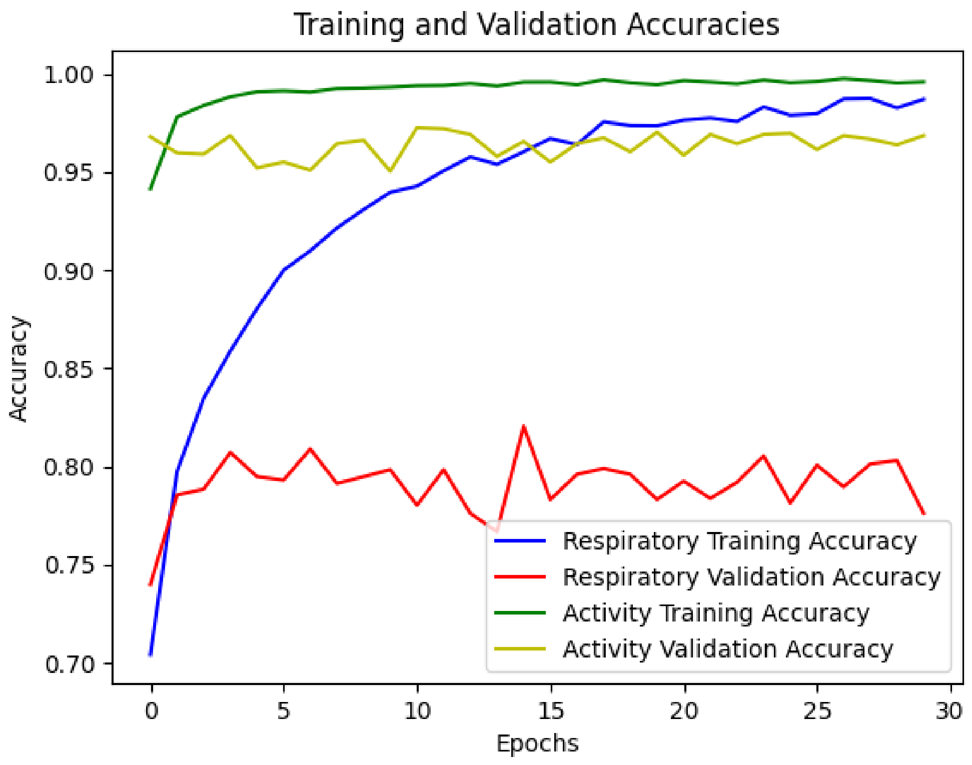


Figure 4.2: Graph demonstrating the progression of training and validation accuracies for both types of classification

The figure above illustrates well the respective difficulties of learning the different types of class and leads to some observations. Firstly, the respiratory training accuracy starts at a lower point but shows a rapid improvement, stabilizing around a high accuracy level swiftly. This suggests that the model is effectively learning the respiratory patterns from the training data. Unfortunately the respiratory validation accuracy is noticeably lower than the training accuracy and does not exhibit the same upward trend. This gap is indicative of overfitting, where the model learns the training data too well, including noise and fluctuations that do not generalise well to unseen data. The inability to learn respiratory symptoms as well suggests we should spend more time inspecting the graphs to help devise more specific metrics or features that can more easily differentiate between the two.

On the other hand, the activity training accuracy begins at a higher point and maintains a steady incline, plateauing near the 1.0 mark. This is indicative of a robust learning process where the model is capable of capturing the essence of the activity data with high precision. The activity validation accuracy closely mirrors the training accuracy, although it is slightly lower. The close proximity of these two lines suggests that the model generalises well and is learning the underlying patterns effectively without overfitting. The slight divergence between the training and validation lines is expected and within acceptable limits, pointing to a well-tuned model.

The critical takeaway here is that while both models are learning their respective tasks effectively, there appears to be room for improvement in the respiratory classification model, particularly in terms of its ability to generalise. Techniques to address this might include further data augmentation, regularization methods, or revisiting the complexity of the model architecture for the respiratory data.

Overall, the accuracies demonstrate a promising performance of the activity detection application, especially in the realm of activity classification. The trends observed in the graph provide valuable feedback for iterative model refinement and highlight the importance of monitoring both training and validation accuracies to ensure the development of a reliable and generalizable model.

Chapter 5

Conclusions

5.1 Reflection on the project

It was remarkable the quantity of inspiration we picked up from the literature we surveyed, which has further cemented the importance of analysing state of the art work in our minds. It was incredibly useful with our limited processing power to be able to see what others have tried and tested, to ascertain whether or not we felt it would make a difference to our work.

We should have made more effort to plan (and stick to) appropriate stages of completion when starting out with more emphasis put on other commitments. We underestimated the impact external factors would have on our ability to work at various times and should have touched base more frequently in times when we didn't have labs and such to make sure everyone was able to cope with the workload they had been allocated at that time.

One of the most significant surprises we encountered was the significant drop off in performance when the model was used with online prediction versus offline performance. There are a few notable differences between the two implementations that we consider responsible for the discrepancy. The first of these is the trimmed and cleaned nature of the data in training, compared with the much noisier raw data that is provided from direct sensory input. The poor performance of the model in online use could have been thrown off by noise, data gaps it was previously unexposed to, and erroneous placement of the sensor during use. As such, we would consider training the model with uncleaned data to prevent overfitting towards data that is too perfect and unrealistic for real world online application. Furthermore we noted differences in some of the pre-processing calculations as identical libraries were not used during testing and implementation. We used python for generating the model and as such python pre-processing libraries were also used in testing. In application deployment we used Kotlin libraries. The results gathered were largely the same but with small differences on a lot of values due to various bit-length decisions and numerical representations that were exaggerated post standardisation.

We have taken on feedback from our peer review and demonstration and understand that will make sure that we will provide more detailed feedback next time we review other applications. We will also make sure that our application is in a more reliable state before demonstrating and when it is to be reviewed.

The team achieved the project objectives in the time available and the prototype, its code and documentation, the results from testing it and this report would form a good basis for future work. The project was an excellent learning experience and we reflect below on what we have learnt. Considering the overall success of the project, we may have been overly critical of ourselves, but perhaps that is not a bad thing.

5.2 Areas for future work

In future, if we were to have access to high performance computing, it would be preferable to carry out the testing in a much more rigorous manner that doesn't fall prey so easily to statistical anomalies from small sample sizes and repetitions. This could also improve the model by allowing for more complex and deeper models. We would also invest in generating a large training set to work on to help improve accuracy and allow for tolerance in noise. It was incredibly apparent the difference that was made between the accuracies when training was carried out on the initial release of data from the students who trimmed their data correctly first time, and the second larger batch that came through once everyone had submitted correctly. Furthermore, students also collected data wearing a Thingy sensor. This had a much more noticeable, and therefore distinguishable, pattern across some of the classes that the RESpeck struggled to classify. For example the difference between sitting and standing could have been distinguished similarly to how we decided which orientation of lying down someone was doing. This being possible because the Thingy was carried in the trouser pocket and as such would be perpendicular when seated with respect to gravity. As mentioned above, movements that are slower would be much harder to distinguish on the RESpeck than the Thingy as the RESpeck is upright for both normal walking and shuffle walking, but the Thingy data had noticeable sharp spikes on the accelerometer when shuffle walking due to the sharp start-stop nature of the movement. Similarly, the Thingy by itself would not be able to distinguish between the different respiratory symptoms and as such for future works, given we have access to both sensors, We would advocate investing in synchronising the sensor data collection to allow a much more informed prediction of the persons activity. We were told that we could not record with both the RESpeck and the Thingy at the same time which made it impossible to synchronise both of the inputs into one model with any meaning. This is because if the results are not lined up perfectly, or at least by a constant factor, the model would get confused with any other pieces of misaligned data. The synchronisation of simultaneous data capture is of course, non-trivial, so this could require a significant amount of work as there would need to be less than 40 millisecond error to not interfere with the 25Hz data capture rate and the nature of internet of things technology such as this means that one could not always rely on state of the art equipment that displays optimal efficiency.

A potential area where our activity detector could be expanded upon is that of geriatric healthcare. The population is constantly aging which putting pressure on healthcare providers to ensure the safety and well-being of the ever growing elderly population. Our system's ability to detect and interpret activity and respiratory patterns could be harnessed to address one of the most common concerns in geriatric care: the timely detection and response to falls.

Falls are a leading cause of injury among older adults, often leading to long-lasting health complications. Integrating our activity detection system into the geriatric healthcare infrastructure could hugely help caregivers and medical professionals respond to such incidents. Expanding the system's capabilities to include detection of erratic movements followed by patterns indicative of distressed breathing or prolonged inactivity could identify when a patient has had a fall.

Once a potential fall is detected, the system could automatically alert caregivers or medical staff. This alert could be sent via multiple different methods such as push notifications, emails or it could integrate with existing emergency response systems within a healthcare facility. The immediacy of this theoretical alert system could significantly reduce response times, ensuring that patients receive prompt medical attention in order to prevent injuries caused by a long fall.

Bibliography

[1] Rasel Ahmed Bhuiyan et al. “A robust feature extraction model for human activity characterization using 3-axis accelerometer and gyroscope data”. In: *Sensors* 20.23 (2020), p. 6990.

[2] Felicity R Allen et al. “Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models”. In: *Physiological measurement* 27.10 (2006), p. 935.

[3] Antonio Bevilacqua et al. “Human activity recognition with convolutional neural networks”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part III 18*. Springer. 2019, pp. 541–552.

[4] *Bluetooth 4.0 Core Specification*. Bluetooth SIG. 2010.

[5] Stylianos Charalampous. “Transformer-based Human Activity Recognition Calibration system”. In: (2022).

[6] Parastoo Dehkordi et al. “Assessment of respiratory flow and efforts using upper-body acceleration”. In: *Medical & biological engineering & computing* 52 (2014), pp. 653–661.

[7] Teodora Georgescu. “Classification of Coughs using the Wearable RESpeck Monitor”. In: (2019).

[8] 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)*. IEEE. 2016, pp. 381–388.

[9] Mohammad Mehedi Hassan et al. “Human activity recognition from body sensor data using deep learning”. In: *Journal of medical systems* 42 (2018), pp. 1–8.

[10] Ankita Jain and Vivek Kanhangad. “Human activity classification in smartphones using accelerometer and gyroscope sensors”. In: *IEEE Sensors Journal* 18.3 (2017), pp. 1169–1177.

[11] Wenchao Jiang and Zhaozheng Yin. “Human activity recognition using wearable sensors by deep convolutional neural networks”. In: *Proceedings of the 23rd ACM international conference on Multimedia*. 2015, pp. 1307–1310.

[12] Ria Kanjilal and Ismail Uysal. “The future of human activity recognition: deep learning or feature engineering?” In: *Neural Processing Letters* 53 (2021), pp. 561–579.

[13] Andrea Mannini and Angelo Maria Sabatini. “Machine learning methods for classifying human physical activity from on-body accelerometers”. In: *Sensors* 10.2 (2010), pp. 1154–1175.

[14] Abdul Kadar Muhammad Masum et al. "Human activity recognition using accelerometer, gyroscope and magnetometer sensors: Deep neural network approaches". In: *2019 10Th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE. 2019, pp. 1–6.

[15] Kristin McClure et al. "Classification and detection of breathing patterns with wearable sensors and deep learning". In: *Sensors* 20.22 (2020), p. 6481.

[16] Agus Eko Minarno et al. "Single triaxial accelerometer-gyroscope classification for human activity recognition". In: *2020 8th international conference on information and communication technology (ICoICT)*. IEEE. 2020, pp. 1–5.

[17] Vimala Nunavath et al. "Deep learning for classifying physical activities from accelerometer data". In: *Sensors* 21.16 (2021), p. 5564.

[18] Franziska Ryser et al. "Respiratory analysis during sleep using a chest-worn accelerometer: A machine learning approach". In: *Biomedical Signal Processing and Control* 78 (2022), p. 104014.

[19] Sadiq Sani et al. "Learning deep and shallow features for human activity recognition". In: *International conference on knowledge science, engineering and management*. Springer. 2017, pp. 469–482.

[20] Vinita Silaparasetty. "What are some of the challenges and limitations of RNNs?" In: *LinkedIn* (2023). URL: `https://www.linkedin.com/advice/1/what-some-challenges-limitations-rnns`.

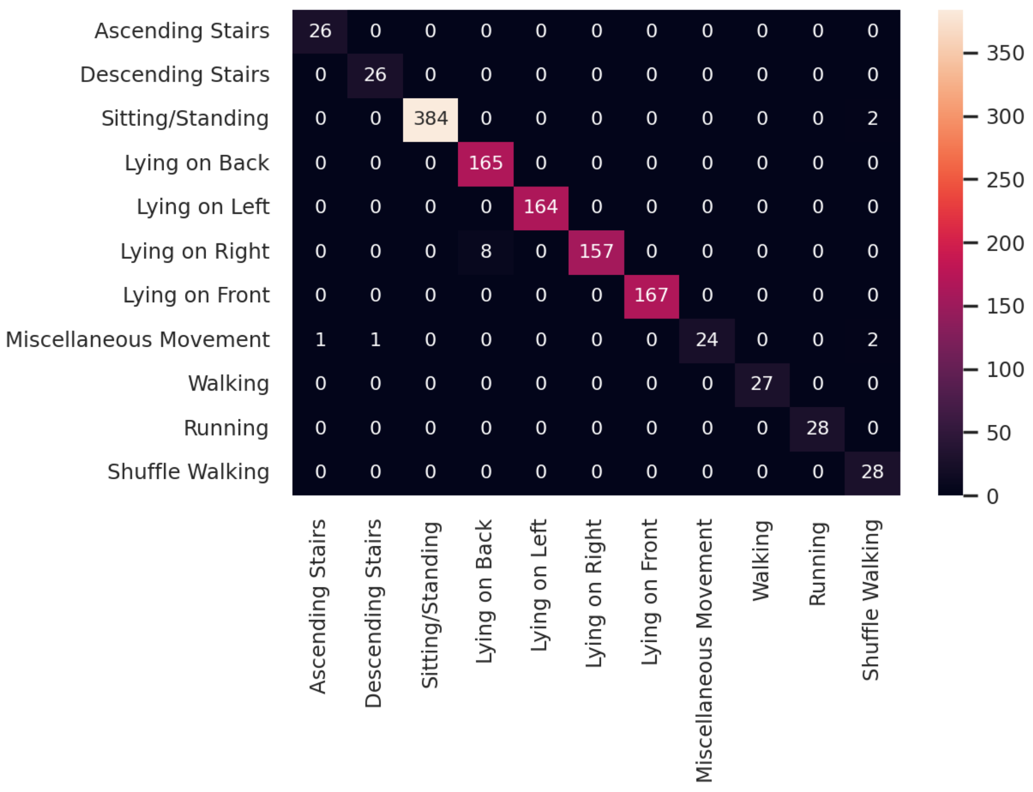
[21] Ergün Alperay Tarim et al. "A Wearable Device Integrated with Deep Learning-Based Algorithms for the Analysis of Breath Patterns". In: *Advanced Intelligent Systems* 5.11 (2023), p. 2300174.

[22] Jose Juan Dominguez Veiga et al. "Feature-free activity classification of inertial sensor data with machine vision techniques: method, development, and evaluation". In: *JMIR mHealth and uHealth* 5.8 (2017), e7521.

[23] Mitchell Webber and Raul Fernandez Rojas. "Human activity recognition with accelerometer and gyroscope: A data fusion approach". In: *IEEE Sensors Journal* 21.15 (2021), pp. 16979–16989.

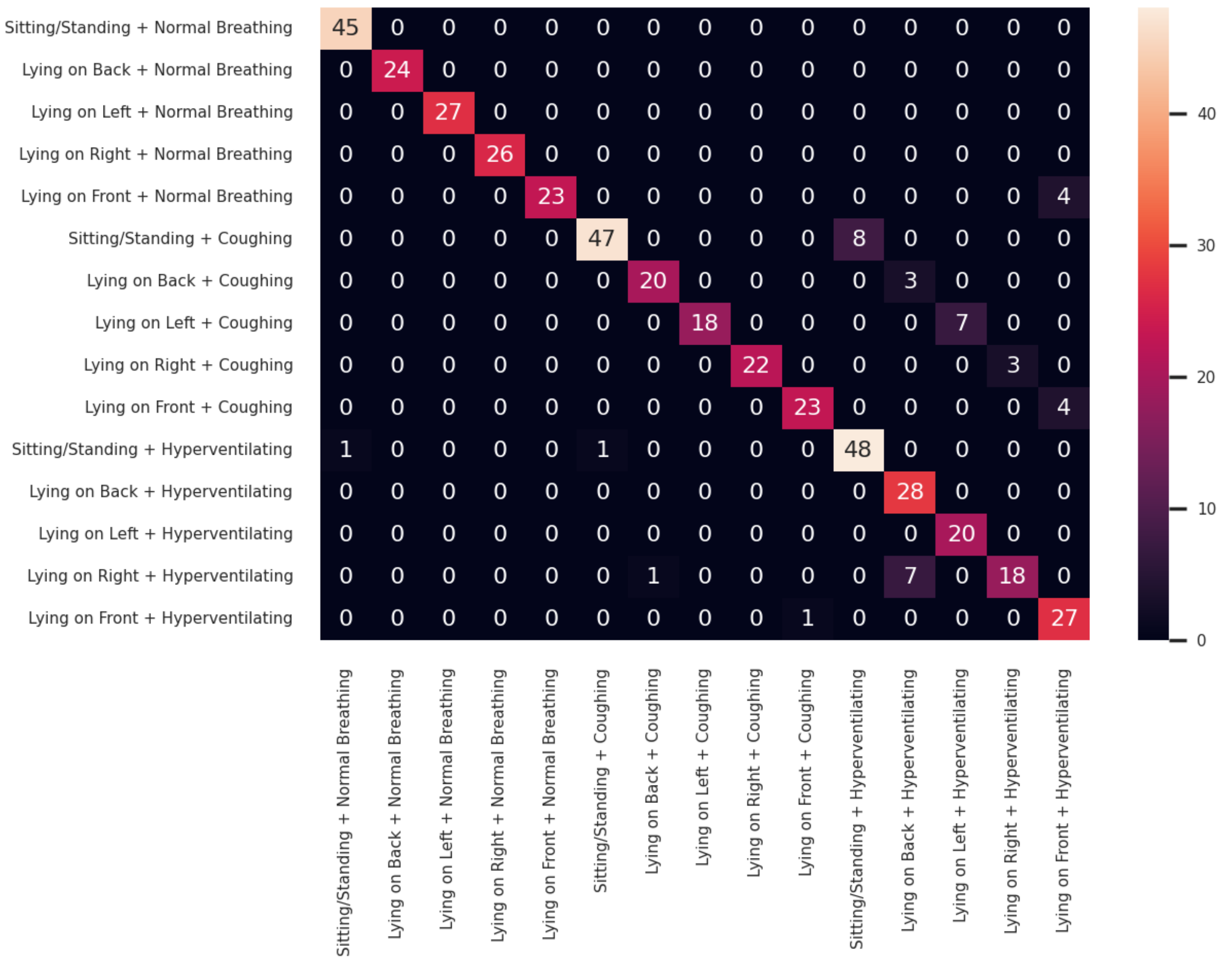
Appendix A

Confusion Matrices



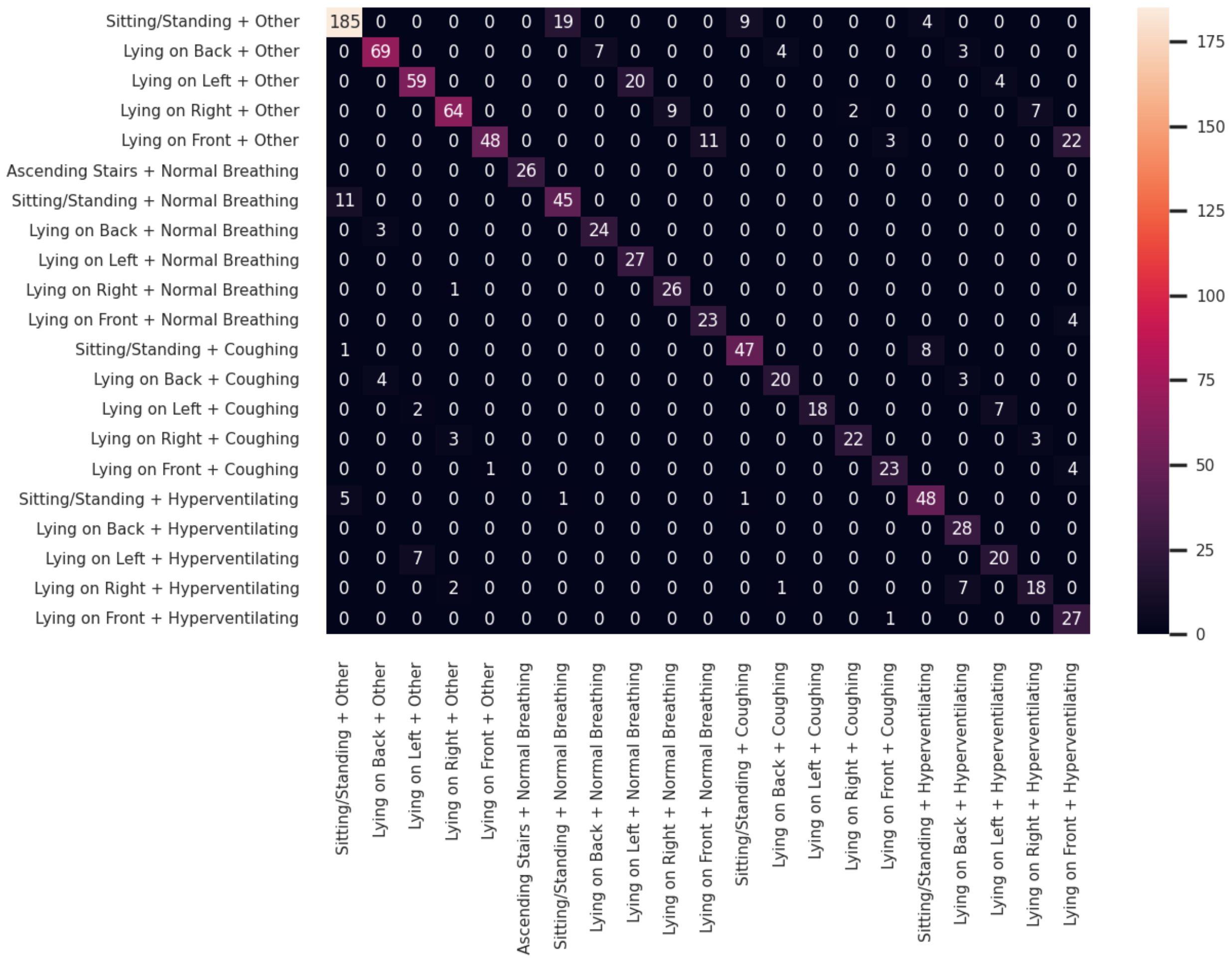
	Ascending Stairs	Descending Stairs	Sitting/Standing	Lying on Back	Lying on Left	Lying on Right	Lying on Front	Miscellaneous Movement	Walking	Running	Shuffle Walking
Ascending Stairs	26	0	0	0	0	0	0	0	0	0	0
Descending Stairs	0	26	0	0	0	0	0	0	0	0	0
Sitting/Standing	0	0	384	0	0	0	0	0	0	0	2
Lying on Back	0	0	0	165	0	0	0	0	0	0	0
Lying on Left	0	0	0	0	164	0	0	0	0	0	0
Lying on Right	0	0	0	8	0	157	0	0	0	0	0
Lying on Front	0	0	0	0	0	0	167	0	0	0	0
Miscellaneous Movement	1	1	0	0	0	0	0	24	0	0	2
Walking	0	0	0	0	0	0	0	0	27	0	0
Running	0	0	0	0	0	0	0	0	0	28	0
Shuffle Walking	0	0	0	0	0	0	0	0	0	0	28

Figure A.1: The confusion matrix for task 1



	Sitting/Standing + Normal Breathing	Lying on Back + Normal Breathing	Lying on Left + Normal Breathing	Lying on Right + Normal Breathing	Lying on Front + Normal Breathing	Sitting/Standing + Coughing	Lying on Back + Coughing	Lying on Left + Coughing	Lying on Right + Coughing	Lying on Front + Coughing	Sitting/Standing + Hyperventilating	Lying on Back + Hyperventilating	Lying on Left + Hyperventilating	Lying on Right + Hyperventilating	Lying on Front + Hyperventilating
Sitting/Standing + Normal Breathing	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Back + Normal Breathing	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Left + Normal Breathing	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Right + Normal Breathing	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0
Lying on Front + Normal Breathing	0	0	0	0	23	0	0	0	0	0	0	0	0	0	4
Sitting/Standing + Coughing	0	0	0	0	0	47	0	0	0	0	8	0	0	0	0
Lying on Back + Coughing	0	0	0	0	0	0	20	0	0	0	0	3	0	0	0
Lying on Left + Coughing	0	0	0	0	0	0	0	18	0	0	0	0	7	0	0
Lying on Right + Coughing	0	0	0	0	0	0	0	0	22	0	0	0	0	3	0
Lying on Front + Coughing	0	0	0	0	0	0	0	0	0	23	0	0	0	0	4
Sitting/Standing + Hyperventilating	1	0	0	0	0	1	0	0	0	0	48	0	0	0	0
Lying on Back + Hyperventilating	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0
Lying on Left + Hyperventilating	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
Lying on Right + Hyperventilating	0	0	0	0	0	0	1	0	0	0	0	7	0	18	0
Lying on Front + Hyperventilating	0	0	0	0	0	0	0	0	0	1	0	0	0	0	27

Figure A.2: The confusion matrix for task 2



	Sitting/Standing + Other	Lying on Back + Other	Lying on Left + Other	Lying on Right + Other	Lying on Front + Other	Ascending Stairs + Normal Breathing	Sitting/Standing + Normal Breathing	Lying on Back + Normal Breathing	Lying on Left + Normal Breathing	Lying on Right + Normal Breathing	Lying on Front + Normal Breathing	Sitting/Standing + Coughing	Lying on Back + Coughing	Lying on Left + Coughing	Lying on Right + Coughing	Lying on Front + Coughing	Sitting/Standing + Hyperventilating	Lying on Back + Hyperventilating	Lying on Left + Hyperventilating	Lying on Right + Hyperventilating	Lying on Front + Hyperventilating
Sitting/Standing + Other	185	0	0	0	0	0	19	0	0	0	0	9	0	0	0	0	4	0	0	0	0
Lying on Back + Other	0	69	0	0	0	0	0	7	0	0	0	0	4	0	0	0	0	3	0	0	0
Lying on Left + Other	0	0	59	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	4	0	0
Lying on Right + Other	0	0	0	64	0	0	0	0	0	9	0	0	0	0	2	0	0	0	0	7	0
Lying on Front + Other	0	0	0	0	48	0	0	0	0	0	11	0	0	0	0	3	0	0	0	0	22
Ascending Stairs + Normal Breathing	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sitting/Standing + Normal Breathing	11	0	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Back + Normal Breathing	0	3	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Left + Normal Breathing	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0
Lying on Right + Normal Breathing	0	0	0	1	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0
Lying on Front + Normal Breathing	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	4
Sitting/Standing + Coughing	1	0	0	0	0	0	0	0	0	0	0	47	0	0	0	0	8	0	0	0	0
Lying on Back + Coughing	0	4	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	3	0	0	0
Lying on Left + Coughing	0	0	2	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	7	0	0
Lying on Right + Coughing	0	0	0	3	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	3	0
Lying on Front + Coughing	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	4
Sitting/Standing + Hyperventilating	5	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	48	0	0	0	0
Lying on Back + Hyperventilating	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0
Lying on Left + Hyperventilating	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
Lying on Right + Hyperventilating	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	7	0	18	0
Lying on Front + Hyperventilating	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	27

Figure A.3: The confusion matrix for task 3