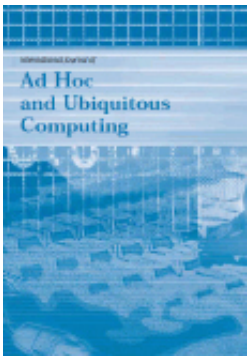


International Journal of Ad Hoc and Ubiquitous Computing

ISSN online: 1743-8233 - ISSN print: 1743-8225
https://www.inderscience.com/ijahuc

NEAT activity detection using smartwatch

Ankita Dewan, Venkata M.V. Gunturi, Vinayak Naik

DOI: 10.1504/IJAHUC.2023.10057574

Article History:

Received:	22 July 2022
Last revised:	13 March 2023
Accepted:	14 April 2023
Published online:	18 January 2024

NEAT activity detection using smartwatch

Ankita Dewan*

Department of Computer Science and Engineering,
IIT Ropar,
Rupnagar, India
Email: 2017csz0012@iitrpr.ac.in
*Corresponding author

Venkata M.V. Gunturi

School of Computer Science,
University of Hull,
Hull, UK
Email: v.gunturi@hull.ac.uk
and
Department of Computer Science and Engineering,
IIT Ropar,
Rupnagar, India

Vinayak Naik

CSIS and APPCAIR,
BITS Pilani,
Goa, India
Email: vinayak@goa.bits-pilani.ac.in

Abstract: This paper presents a system for distinguishing non-exercise activity thermogenesis (NEAT) and non-NEAT activities at home. NEAT includes energy expended on activities apart from sleep, eating, or traditional exercise. Our study focuses on specific NEAT activities like cooking, sweeping, mopping, walking, climbing, and descending, as well as non-NEAT activities such as eating, driving, working on a laptop, texting, cycling, and watching TV/idle time. We analyse parameters like classification features, upload rate, data sampling frequency, and window length, and their impact on battery depletion rate and classification accuracy. Previous research has not adequately addressed NEAT activities like cooking, sweeping, and mopping. Our study uses lower frequency data sampling (10 Hz and 1 Hz). Findings suggest using statistical features, sampling at 1 Hz, and maximising upload rate and window length for optimal battery efficiency (33,000 milliamperes per hour, 87% accuracy). For highest accuracy, use ECDF features, sample at 10 Hz, and a window length of six seconds or more (37,000 milliamperes per hour, 97% accuracy).

Keywords: non-exercise activity thermogenesis; NEAT; smartwatch; activity recognition; battery.

Reference to this paper should be made as follows: Dewan, A., Gunturi, V.M.V. and Naik, V. (2024) 'NEAT activity detection using smartwatch', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 45, No. 1, pp.36–51.

Biographical notes: Ankita Dewan is presently pursuing her PhD at the Indian Institute of Technology (IIT Ropar), located in Punjab, India. With a background in teaching for two years at BML Munjal University, her research primarily revolves around the areas of ubiquitous and MobiQuitous computing, closely intertwined with the field of data science.

Venkata M.V. Gunturi is currently a Lecturer at the School of Computer Science at the University of Hull, UK. He has over ten years of experience in the area of data science. His works are published in highly prestigious journals such as *IEEE TKDE*, *Geo-Informatica* and *IJGIS*. His research work has been supported by both the government (DST, India) and industry (Microsoft India).

Vinayak Naik is an OPERA Awardee Professor of Computer Science at BITS Pilani, Goa. He heads the Computer Science Department and a faculty at APPCAIR, an AI center of excellence. He is a member of the Research Advisory Council of D.Y. Patil International University. Previously, he was an Associate Professor at the IIIT Delhi and worked as a Visiting Associate

Professor at the IIT Bombay. His research focuses on large-scale networked systems, including mobile computing, IoT, ML, and AI. He has received numerous awards for his work, including Best Paper Awards, Research Fellowships, and Teaching Excellence Awards.

1 Introduction

This paper aims to create a system that utilises the accelerometer and gyroscope data collected by the embedded sensors (Milette and Stroud, 2012) in a smartwatch to distinguish and inform the user of the various activities they perform at home. The system will be able to differentiate 13 different activities, cooking, sweeping, mopping, walking, climbing up, climbing down, eating, driving, working on a laptop, browsing on a phone, cycling, sitting in a car, and watching TV. Out of these activities, the top six are considered non-exercise activity thermogenesis (NEAT), which refers to any home activity that requires energy, excluding sleeping, resting, and traditional exercise. The remaining activities, such as driving and cycling, are considered non-NEAT, as they do not require much energy.

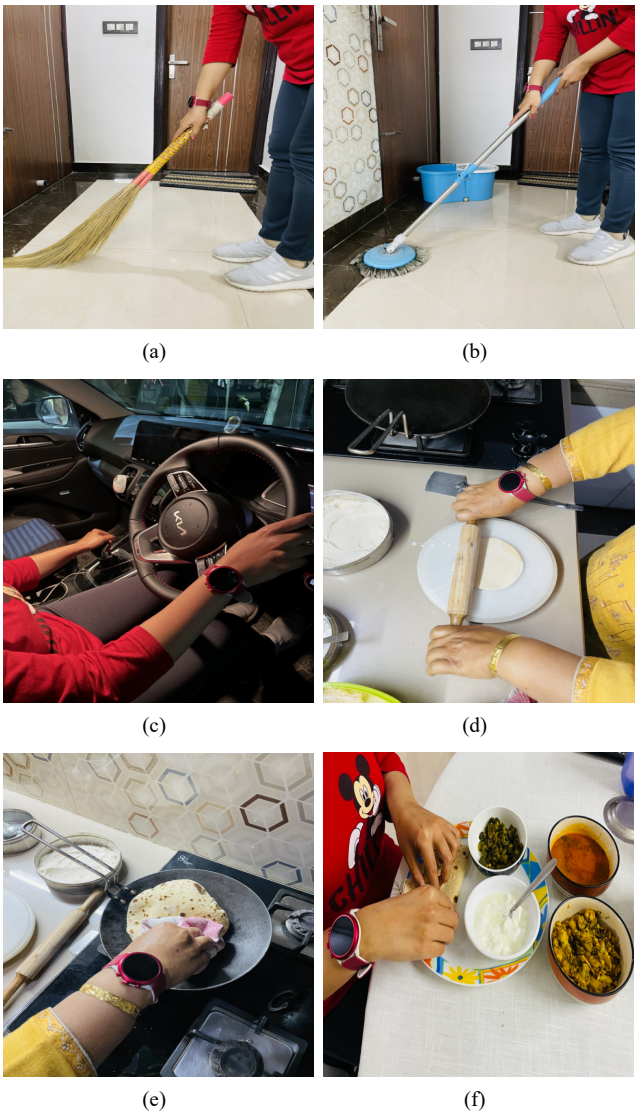
NEAT activity recognition is crucial in health monitoring applications that keep track of the time spent on each activity (Levine, 2002). In Figure 1, we see some of the daily house activities in an Indian home scenario. In the fullness of time, a person can get a good understanding of his time assigned to the NEAT activities and will be given a sound vision of how to plan his days (Villablanca et al., 2015; Levine et al., 2006). NEAT activities have been linked to better health outcomes, such as, reducing the risk of obesity, diabetes, heart disease, and depression (Hamasaki et al., 2016; Villablanca et al., 2015; Levine et al., 2006; Owen et al., 2009). Hence, a mobile application that tracks these activities, especially during a pandemic when people are confined to their homes, could be beneficial in helping people keep track of their physical activity levels.

In our opinion, a perfect solution for NEAT activity recognition must possess the following two characteristics. To begin with, it must use a single and readily available hardware device (for example, a smartwatch or a smartphone). Secondly, as NEAT activities are likely to occur throughout the day, our recognition system should be energy efficient. Any user would probably like the ability to record meaningful data without needing to charge their device numerous times during the day to get significant results.

A smartwatch-based solution is developed in this work to recognise NEAT and non-NEAT activities. If energy efficiency is the goal, reducing the sampling rate of the sensors is the most efficient way of achieving it; we know that battery consumption are directly proportional to the data sampling rate (Zheng et al., 2017). Other than sampling rate, we can also vary the amount of pre-processing of the features, timestamp length of the window, etc. We will discuss this in detail in Section 5 of this paper.

Although good battery life is guaranteed by a low sampling frequency, it should be noted that the 13 activities focused on in this paper cannot be easily distinguished in low-frequency data. Therefore, the user must make an intelligent choice in determining the frequency that suits them to achieve efficient battery life for their smartwatch. The total accuracy (for our 13 activities of interest) obtained by different classification algorithms on data sampled at 1 Hz, 5 Hz, and 10 Hz is demonstrated in Table 1. As the frequency of sampling data increases, the accuracy is expected to increase."

Figure 1 A few NEAT and non-NEAT activities which are different in our developing nation – India, (a) sweeping (b) mopping (c) driving on right side (d) flatting dough in kitchen (e) roasting chapati on tava (f) eating food with hand (see online version for colours)



(a) (b) (c) (d) (e) (f)

Table 1 Accuracy of different classifiers

Frequency	*1 Hz*	*5 Hz*	*10 Hz*
KNN	77 (0.2)	87 (0.5)	88 (0.1)
Multi-layer perceptron	74 (1.1)	85(0.6)	87 (0.4)
SVM	78 (0.4)	88 (0.3)	88 (0.3)
Logistic	64 (0.7)	77 (0.4)	78 (0.6)
Random forest	48 (2.7)	55 (1.5)	56 (1.9)
Naive Bayes	62 (0.8)	71 (0.8)	71 (0.8)
XGBoost	80 (0.4)	91 (0.4)	92 (0.3)

Notes: Along with the standard deviation after five runs on 1 Hz, 5 Hz, and 10 Hz sampled data for two seconds window length and 0% overlap.

1.1 Our contributions

1 In the preliminary version of this work (Dewan et al., 2021), we focused on limited number of activities but in current work we are expanding the scope to include more activities. So now, in this paper, we have a total of 13 activities:

- cooking
- sweeping
- moping
- walking
- climbing up
- climbing down
- eating
- driving
- working on laptop
- browsing on the phone
- cycling
- sitting in a car
- watching TV.

2 We deployed our classification models on the server and used them according to the desired accuracy. We deployed four strong classification models (XGB, MLP, SVM, and random forest) on the Heroku server using the Flask API. The prerequisites for using our smartwatch entail establishing a Bluetooth connection between the device and a compatible smartphone. Furthermore, to obtain accurate output signals from the server API, both the smartwatch and smartphone must be connected to Wi-Fi. The classification model utilised by the smartwatch operates on data derived from the accelerometer and gyroscope sensors present in a typical smartwatch.

3 As mentioned above, low-frequency data means low accuracy and vice-versa. So we left this decision to our end-user, which frequency is most suitable to them depending upon the battery efficiency of their watch.

4 We evaluated (trained and tested) the classification models experimentally on actual data sampled by seven volunteers. We show the experimentation graphs for both, accuracies and the smartwatch's battery life.

5 Lastly, we worked on 'others' class which means if the activity detected does not belong to any of the above-mentioned 13 classes then the prediction is made as 'others'. This works on the basis of threshold which is configurable depending on the strictness a user desires.

6 The brief results can be seen in Table 2.

Table 2 Key results

Parameters	*Best battery efficiency*	*Best accuracy*
Features	Statistical	ECDF
Data sampling frequency	1 Hz	10 Hz
File upload rate	As high as possible	No effect on accuracy on changing this parameter
Window length	$\geq$6 seconds	$\geq$6 seconds
Classifier	No effect on battery	XGB (n_estimators = 100)

1.2 Scope of the paper

In this paper, the 'NEAT' motion class encloses the following activities:

1 cooking

2 sweeping

3 moping

4 walking

5 climbing up

6 climbing down, and the 'non-NEAT' motion class enclose the following activities:

7 eating

8 driving

9 working on laptop

10 browsing on the phone

11 cycling

12 sitting in a car

13 watching TV.

These are the maximum activities a person can perform in a home scenario. We can easily identify them using a smartwatch's sensors.

1.3 Outline of the paper

Section 2 discusses the basic concepts and the problem definition at hand. In this, we explain all the activities which we have evaluated and how the raw-data has been processed. Section 3 discusses the work similar to our work, which includes IoT devices and multiple on-body sensors at a high frequency. Next, we discuss our proposed approach (Section 4), the architecture of our model (Subsection 4.2), the experimental evaluation (Subsection 5), how we collected the dataset (Subsection 5.1) (used for training purposes), and what classification algorithms (Subsection 5.2) we are using to train and test. Then, we show the experimentation in which we discuss the variation of battery (Subsection 5.3.1) and accuracy (Subsection 5.3.2) for above mentioned parameters. Although there were many possibilities with the parameters mentioned-above, we shown only a few of them to understand the trend and hence saving the redundant space of this paper. Next, we demonstrate the real-time testing of all the activities (Subsection 5.5) accomplished by an external volunteer whose data does not get included in the training. Lastly, we present our key results (Subsection 5.6), summarising the accuracy-driven and energy-efficient models corresponding to each set of parameters.

2 Basic concepts and problem definition

2.1 Basic concepts in motion classes

1 *Cooking:* The user is working in the kitchen for this motion class. The user's smartwatch has dedicated buttons for starting and stopping the process. As soon as one enters the kitchen, he presses the start button, then the stop button once he leaves. This class has described all aspects of conventional cooking, including actions such as creating Indian flatbread (smoothing and flattening the dough [Figure 1(d)], toasting it over a pan [Figure 1(e)], as well as preparing curry (slicing, dicing, washing, and stirring the vegetables). So, it covers most of the actions that one does in a regular Indian kitchen setup daily.

2 *Sweeping:* This motion class is conducted throughout the entire house. Volunteers sweep the floor with a broomstick while performing this activity [Figure 1(a)]. The person may move objects intermittently while performing this activity.

3 *Wet mopping:* A person uses a wet-mopping stick to mop the floor in this motion class [Figure 1(b)]. We conduct it for the entirety of the house. In addition to dipping the mop into a cleaning solution and wringing it, this motion class includes wringing the mop after using it.

4 *Walking:* When a user walks around at home (but not while sweeping or mopping), they experience this motion class.

5 *Climb up/down:* User will climb and descend the stairs during this motion class. While performing this activity, volunteers were also permitted to hold the side railings.

6 *Eating:* In this activity, the volunteer will eat by using his hand directly or can take the help of a spoon or fork. In Indian culture, people usually prefer eating with their hands [Figure 1(f)].

7 *Driving:* In India, driving is essentially done on the left-hand side, and the steering wheel is on the right; hence we primarily use the left arm for maneuvering the gear and the right arm for the steering wheel [Figure 1(c)].

8 *Working on a laptop:* In this activity, the volunteer is primarily idle and has a minimal motion of typing on a keyboard (at an average speed) or using a touchpad.

9 *Browsing on a phone:* This activity deals with the idle portion of human time. When a person is browsing or texting on his mobile phone, his right hand usually holds the phone in front of his face or a little lower.

10 *Cycling:* In this activity, although the position of the hand remains steady, due to holding the handle, some amount of continuous jerk can be felt by smartwatch sensors. The volunteer was instructed not to remove their hands from the handle in this motion.

11 *Sitting in car:* This activity deals with again idle/zero motion of a volunteer who is ideally not doing any work, but a person may feel slight amount of jerks moving in the car which can be perceived by the smartwatch.

12 *Watching TV:* This activity deals with sluggish motion. In this case, a person sits idle on a couch or bed and will pick up the remote once in a while to change channels.

In all the classes mentioned above, the volunteers were wearing the smartwatch in their dominant hand, i.e., the right hand.

2.2 Problem definition

- *Input:* Our raw data includes a set of time series ($\mathcal{T}$) taken from smartwatch's accelerometer and gyroscope sensors. With the accelerometer and gyroscope sensors in the smartwatch, we have sensor values at any time point p in the time series $t_i \in \mathcal{T}$. Each time

series $t_i \in \mathcal{T}$ corresponds to one of the 13 motion classes.

The input set of time series $\mathcal{T}$ is transformed into overlapping windows by applying $\mathcal{W}$. Each window $w_i \in \mathcal{W}$ is β in length. There is a temporally adjacent window overlapping by θ. In our experiments, we vary both β and θ but for the real-time testing phase, θ remains constant as 0. A class label is assigned to each $w_i \in \mathcal{W}$ corresponding to one of the 13 previous activities, and the activity is represented using a number of characteristics.

- *Objective:* Learning and deploying four strong classification models (using data from $\mathcal{W}$ to train) that are capable of classifying the previously cited 13 activity classes. The trained models are deployed on a server, and with the help of trained models, the live data can be classified by using an API.

3 Related work

During the past decade, researchers have been studying human activity recognition from various perspectives. We divided the work related to detecting home activities into two categories. The first category (Tapia et al., 2004; Skocir et al., 2016; Bianchi et al., 2019; Cicirelli et al., 2016) talks about detecting home activities based on IoT devices/sensors, and the second category talks about detecting home activities with the help of wearable sensors (e.g., Bianchi et al., 2019; Stisen et al., 2015; Jiang et al., 2016; Mannini, 2015; Zhang and Sawchuk, 2012; Murao and Terada, 2014; Guo et al., 2016; Inoue et al., 2015; Ponce et al., 2016; Sun et al., 2010; Thiemjarus et al., 2013; Ronao and Cho, 2016; Bayat et al., 2014; Sun et al., 2017; Shoaib et al., 2014; Ordóñez and Roggen, 2016 which is closer to our work. This category is further divided into two sub-categories, i.e., use of multiple sensors (Mannini, 2015; Zhang and Sawchuk, 2012; Murao and Terada, 2014; Guo et al., 2016; Inoue et al., 2015; Ponce et al., 2016) and use of high-frequency sensor data (Stisen et al., 2015; Jiang et al., 2016; Sun et al., 2010; Thiemjarus et al., 2013; Ronao and Cho, 2016; Bayat et al., 2014; Sun et al., 2017; Shoaib et al., 2014; Ordóñez and Roggen, 2016). Here is a brief overview of the current research literature and its relevance to the present discussion.

The first category requires lots of IoT instruments which are a bit costly. For instance, Tapia et al. (2004) talks about retrofitting their devices in the existing homes which comprises three components, i.e., the environment which fills with state change sensors taped to objects, an experience sampling tool (ESM) that allows users to label their activities in context and lastly recognition algorithms that recognise patterns and categorise activities. They had to install 80 sensors in a particular home to capture data, which was quite a hassle. Although they tried to cover the maximum home activities a person can perform, their accuracy varied from 25% to 89%. Skocir et al. (2016) talk about just entering and exiting a person inside a room to adjust the settings of heating, ventilation, and air conditioning (HVAC). Again, the IoT devices are used to sense a human presence, making detection possible using simple algorithms like ANN. Similarly, Bianchi et al. (2019) uses the concept of IR motion sensors in every room to recognise most of the activities of the person, along with a wearable three-axis accelerometer sensor to identify the posture of a person. Cicirelli et al. (2016) talk about anomaly detection along with recognition of 11 activities with the help of an H_2O autoencoder. Again multiple sensors were involved in solving the purpose. Furthermore, there are tons of research going on in this field. They use IoT devices in every part of the home and can guess a user's activity by knowing the location of a person inside the house. The main idea behind this research is to make an intelligent home environment with the help of sensor devices and Wi-Fi connectivity.

The next category (e.g., Bianchi et al., 2019; Stisen et al., 2015; Jiang et al., 2016; Mannini, 2015; Zhang and Sawchuk, 2012; Murao and Terada, 2014; Guo et al., 2016; Inoue et al., 2015; Ponce et al., 2016; Sun et al., 2010; Thiemjarus et al., 2013; Ronao and Cho, 2016; Bayat et al., 2014; Sun et al., 2017; Shoaib et al., 2014; Ordóñez and Roggen, 2016 of activity recognition talks about detection using wearable devices similar to our case. The classification models developed in these works can anticipate a wide range of activities. Research done in this area has been broadly divided into two categories:

a multiple sensors were used (including a customised hardware) (Stisen et al., 2015; Jiang et al., 2016; Mannini, 2015; Murao and Terada, 2014; Inoue et al., 2015; Ponce et al., 2016; Krishnan and Cook, 2014; Shoaib et al., 2014)

b one sensor was used (e.g., smartphone and smartwatch) (Zhang et al., 2010; Sun et al., 2010; Ronao and Cho, 2016; Bayat et al., 2014; Sun et al., 2017).

In the first category, classification models are developed by using data from multiple sensors (e.g., Tapia et al., 2007; Murao and Terada, 2014; Krishnan and Cook, 2014; Ponce et al., 2016; Inoue et al., 2015). For example, Inoue et al. (2015) proposed a model which incorporated accelerometer data from three locations (one on the wrist, one in the breast pocket, and one on the back hip) for the purpose of recognising activity in a hospital setting. Likewise, the method proposed in Tapia et al. (2007) requires five accelerometers (located on the body in various locations) and a heart monitor to assess human activity such as rowing, cycling, walking, etc. Based on accelerometer data (embedded in the legs, arms, and hips), Murao and Terada (2014) recognises body postures (standing, sitting, etc.), movement (walking, running, etc.), and hand gestures (throw, chop, punch, etc.). With the help of multiple sensors mounted on the wrist, chest, and ankle, Ponce et al. (2016) developed a classification model for detecting 18 of the most common household activities. Despite the impressive results of these approaches, they are not

relevant in our problem setting since any technology that uses multiple devices to recognise NEAT activities would not be accepted by consumers. Ideally, data sampling and prediction should be limited to one easily accessible device (e.g., a smartphone or smartwatch, which are readily available).

We can further segment the second category of solutions (i.e., those which utilise a single device) into two subcategories:

1 a high frequency (e.g., 20 Hz or above) data sampling is performed (Sun et al., 2010; Ronao and Cho, 2016; Bayat et al., 2014; Sun et al., 2017)

2 low-frequency data sampling is performed (Zhang et al., 2010).

For NEAT activity recognition, solutions based on high-frequency data are unsuitable because the battery would be consumed faster by such solutions. There is a direct correlation between high-frequency data sampling rate and battery consumption, as cited in Zheng et al. (2017) and Dewan et al. (2019). NEAT activities are likely to spread throughout the day, so a solution should be energy efficient, as a consumer would like to collect data for a full day to get meaningful insights into their NEAT activities.

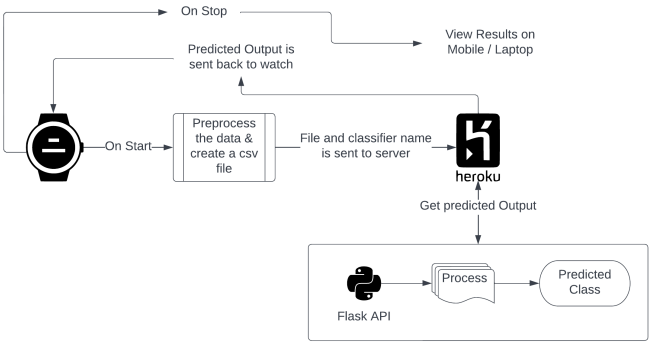
Zhang et al. (2010) operates at a low frequency of 1 Hz. In this paper, the volunteers wore their smartphones on their belts to identify the activities. Seated activities, standing activities, lying activities, walking activities, posture changes, and gentle motions are distinguished here. Even though these are some common activities (for instance, walking), we focus on several activities that are not trivial (for example, cooking, mopping, and sweeping), which are integral in a solution for NEAT activity monitoring. Patel et al. (2008) proposes a novel hierarchical machine learning classifier specifically designed to address closely related classes. It employs a temporal pattern mining approach that recognises, for instance, the likelihood of eating following the act of cooking. This work is a valuable complement to our own.

4 Proposed approach

4.1 Pre-processing and features

For training, we divide raw sensor values into overlapping windows ($\mathcal{W}$) consisting of time series data $\mathcal{T}$. A certain degree of overlap between two temporally adjacent windows w_i and w_{i+1} in $\mathcal{W}$ is determined by the parameter θ. The values of θ are 0, 0.3, 0.5 and 0.7 (Issa et al., 2022). No overlap occurs at $\theta = 0$. In $\theta = 0.3$, $\theta = 0.5$ and $\theta = 0.7$, there is an overlap of 30%, 50% and 70% respectively between two successive windows. By creating an overlapping 50%, we are, in effect, adding another data point between non-overlapping (but temporally adjacent) windows. As a result, we can reduce the likelihood of outliers, resulting in a more robust model.

Figure 2 Architecture for testing on live data



The windows are of length ω. In our study, ω equates two, four, and six seconds.

- *Accelerometer individual axis features:* With the help of an accelerometer, we calculate the rate at which the device's velocity changes. Accelerometers produce results in all three dimensions, i.e., x-axis, y-axis, and z-axis. There will be three sets of acceleration values for a particular window $w_i \in \mathcal{W}$ consisting of a_x, a_y, and a_z. A window of three time series is analysed using five statistical features:

  1 mode

  2 max

  3 median

  4 lower quartile

  5 standard deviation.

  The result is 15 possible combinations, composed of three axes paired with five statistical features. We used raw sensor values and do not apply a filter before using them to calculate statistical features. Similarly, we analyse a window of three time series by finding five data points at equal intervals between the minimum and maximum value, which we call ECDF (empirical cumulative distribution function) features.

- *Gyroscope individual axis features:* We measure an angular speed by a gyroscope sensor attached to or worn by a person. The gyroscope also displays three dimensions for the x-axis, y-axis, and z-axis, just like the accelerometer. We denote the resulting time series by g_x, g_y, and g_z. Gyroscope sensors can also be used as a motion sensors to detect a device's orientation in conjunction with the accelerometer values. A time-series window with 15 statistical features has the previously mentioned five statistical features and five ECDF features for each axis.

4.2 Model architecture

After preparing the back-end models, we deploy them on a server. Figure 2 gives an overview of how the live data

flows in the system and how we get the desired output in a smartwatch by using Wi-Fi connectivity.

The process starts with a person wearing a smartwatch with an active Wi-Fi connection; it automatically synchronises with the smartphone's Wi-Fi. With the press of the start button, we begin capturing the raw inputs from the sensors. We then pre-process this raw data using static or ECDF features and save it into a CSV file in the smartwatch itself. The processed CSV file gets sent to the Heroku server regularly, hitting an API that expects two parameters – the CSV file and the classifier's name. After the models process the CSV, the server sends back the result to the smartwatch. We repeat this process until the user presses the stop button and transfers the entire prediction and actual activity class file to a smartphone. We use this method to test the accuracy of all activities (we will show this in Subsection 5.5).

4.2.1 Detection of 'others' class

Along with the instantaneous detection of the 13 NEAT and non-NEAT activities, a user performs multiple other activities too. So to generalise our approach towards deployment, we detect those non-listed unknown activities named the 'others' class. We use various clustering algorithms and outlier detection methods to detect this activity. We will discuss in brief each of them.

1 *Gaussian mixture model:* Its library has a predict_proba function, which gives us a probability of a test point belonging to one of the classes. Even if the points lie far from all the clusters, yet it gives a high probability of belonging to one of the clusters.

2 *OPTICS algorithm:* We find different numbers of clusters for our 13 classes and calculate the mean (the average value of all data points belonging to one class) and variance (the spread of all the data points of a class from its class mean) of these clusters. Then, we compared the distance between an outside class data point and mean of each cluster and compared it with three times the variance since for a normal distribution 99% of the data lies within the three times the standard deviation.

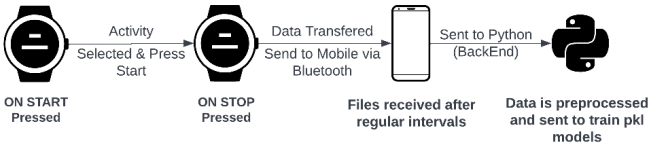
Unfortunately, we did not get any accurate results from these experiments since it is challenging to find an outlier in highly sparse data. Henceforth, we apply the threshold concept, i.e., for every window, if more than 60% instances do not belong to the same class, then we say that the entire window belongs to the 'others' class. We chose 60% because any number above 50% would have been acceptable. Basically to have some prediction for maximum possible windows low threshold will be helpful.

5 Experimental evaluation

5.1 Dataset used

We collect real-time data at a frequency of 10 Hz from a home environment. The data was collected using Fossil's sports smartwatch, which runs an Android platform and is powered by Google's Wear OS as shown in Figure 3. All the seven volunteers wore their watches on their dominant hands (mostly right hands). There is a list of activities listed on the watch and a start and stop button at the top. The volunteer would enter the ground truth labels in the smartwatch application before commencing any activity by clicking on any list item. So the volunteers selected the activity they wanted to do (cooking, sweeping, mopping, walking, climbing up, climbing down, eating, driving, working on laptop, browsing on the phone, cycling, sitting in a car, and watching TV) before pressing the start button. With the press of the stop button, the volunteer can halt the activity. To collect data, the volunteer had to keep their mobile phone in their pocket or hands because, using Bluetooth technology, we transfer data from their smartwatch to their smartphone every few minutes. According to our preferences, we can set a config to transfer data from the smartwatch to the smartphone at any given interval.

Figure 3 Architecture for data sampling



The watch or phone did not contain any 'GPS' or any other sort of 'tag' information. A total of four to five hours of data was collected but could consider only three and a half hours (balanced data) due to an imbalance in activity classes. The imbalance was natural since it took significantly less time to climb up and down the stairs, than to cook or do any other household work such as sweeping or mopping, or even, for instance, sit idle/watch TV/work on a laptop. To prevent the issue of misclassification caused by dominant classes, we take equal samples from each category. At last, the data is collected and sent to the backend for some pre-processing (statical or ECDF). We then tend to use this pre-processed data to create classification models.

5.2 Candidate algorithms

Here, we are using the most common and most effective classification models. These are in-built classifiers found in most machine learning libraries (for example, sklearn in Python). For the classification purpose, we chose the following classifiers:

a MLP (multi-layer perceptron with three hidden layers having 13 neurons each)

b SVM (support vector machine with kernel = 'rbf')

c random forest

d XGB (extreme gradient boosting)

e AutoML [automated machine learning with parameters as time_left_for_this_task: $5 * 60$ (five minutes) per_run_time_limit: 50 seconds initial_configurations_via_metalearning: 0] (Patel et al., 2020a).

5.3 Experimental goals

5.3.1 Effect of parameters on battery

We have some parameters on which we performed the battery experiments, i.e., we calculate the battery depletion rate. The four basic parameters are:

1 *Features:* We factor in two kinds of features, statical and ECDF. We considered five statistical features:

   1 mode

   2 max

   3 median

   4 lower quartile

   5 standard deviation.

   Similarly, empirical cumulative distribution function is a step function for n data points that jumps up by $1/n$ each time. For ECDF, we find out five data points at equal intervals between the minimum and maximum value. We witness the battery depletion for both ECDF and statistical features in Figure 4. The computation of ECDF is more complicated than simple statistical features that we are calculating for pre-processing, subsequently, the rate of battery depletion.

2 *Window length:* We denote the window's length by ω. In our study, ω equates to two, four, and six seconds as shown in Figure 5. The longer the window length, the lesser the battery consumption. Like 'file upload rate', a smaller window length will mean multiple windows; subsequently, it will pass more instances through PKLs on the server.

3 *Sensor sampling rate:* Although there are four types of modes available in our android, i.e., normal, UI, game, and fastest, we are using the slowest two, i.e., normal and UI (considering the low frequency $\rightarrow$ low battery depletion phenomena). These are the delays that the sensors provide. We get approximately 3–4 pings/second using normal mode, and UI mode delivers approximately 13–15 pings/seconds. As we insert the values of the accelerometer and gyroscope, we consider the latest value that the sensor has dispatched after every second and every 1/10th of a second for normal and UI mode, respectively. Our application's *onSensorChanged()* callback method receives sensor events based on the data delay (or sampling rate). We can witness the battery depletion in Figure 6, and, the higher frequency data sampling rate, i.e., 10 Hz (UI mode), depletes the battery at a faster rate as compared to the 1 Hz (normal mode).

4 *File upload rate:* This is the time we fix for sending our file to the server for processing through PKLs models (prepared classification models). We observe the battery depletion graph in Figure 7. We vary the time intervals from ten seconds (10 kms) to 70 seconds (70 kms). The smaller the gap for sending files, the higher would be the battery depletion since the server is getting pinged frequently. Time for sending file to the server can be after every few seconds or after full storage capacity its a personal choice but optimal one would be 'as high as possible' since this is hardware dependent. We can increase this time duration to some minutes or few hours till the time our application does not freeze.

We performed the above four experiments for one hour each and noted the battery level at intervals of ten seconds. We used to start the battery off with full charge and noted the depletion rate concerning the last value reported. As the absolute battery value may vary at full charge (100%), we had to normalise the values to start from one absolute value. We did this to get a better visualisation of the graphs.

5.3.2 Effect of parameters on accuracy

5.3.2.1 Training and evaluation metrics

We use two-second, four-second, and six-second window lengths in our experiments. We vary a window overlap parameter θ between 0 (i.e., no overlapping between consecutive time windows) and 0.7 (i.e., 70% overlap between consecutive time windows). Taking a set of $\mathcal{W}$ values with their corresponding θ values, we divide it into a 4:1 ratio, that is, assigning 80% of the data for training, and the rest 20% is set for testing. Our dataset is randomly divided into training and test sections (ten times) to obtain reliable results. We train on each of the ten sections (and tested the learned model). Lastly, we present the average of the F1-scores and accuracy values obtained across these ten test datasets.

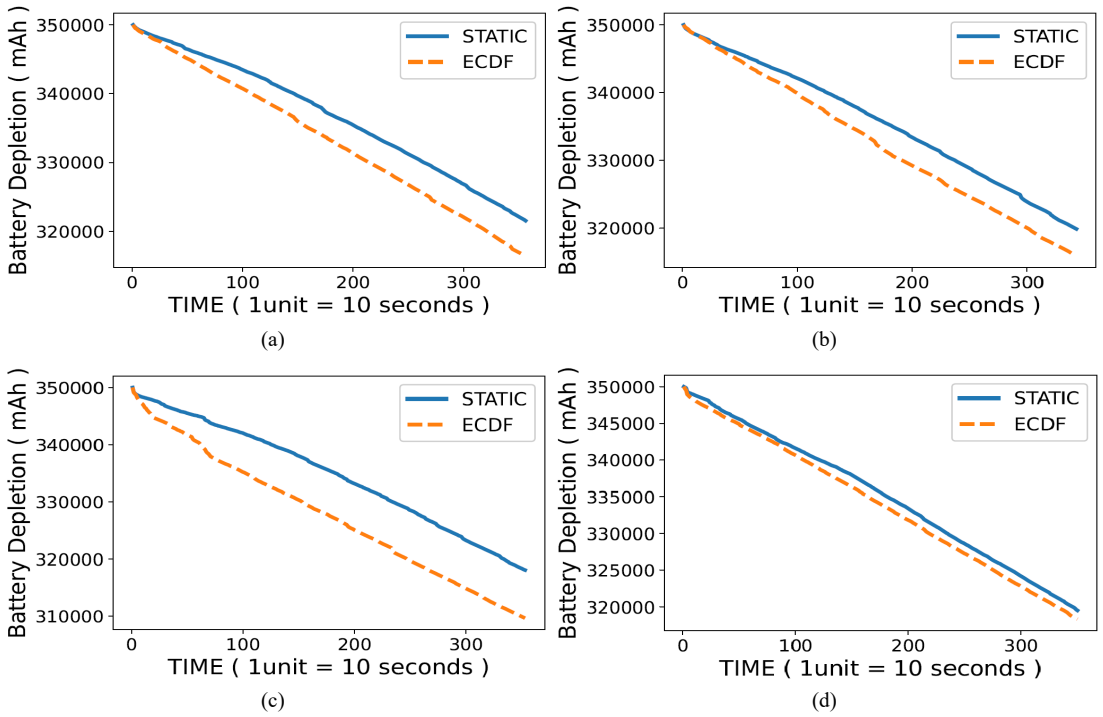
1 *Effect of varying window length and overlapping percentage on the final accuracy:* The average of ten results is shown in the graphs of Figures 8, 9, 10 and 11. In all these figure:

   a represents two seconds window

   b represents four seconds window

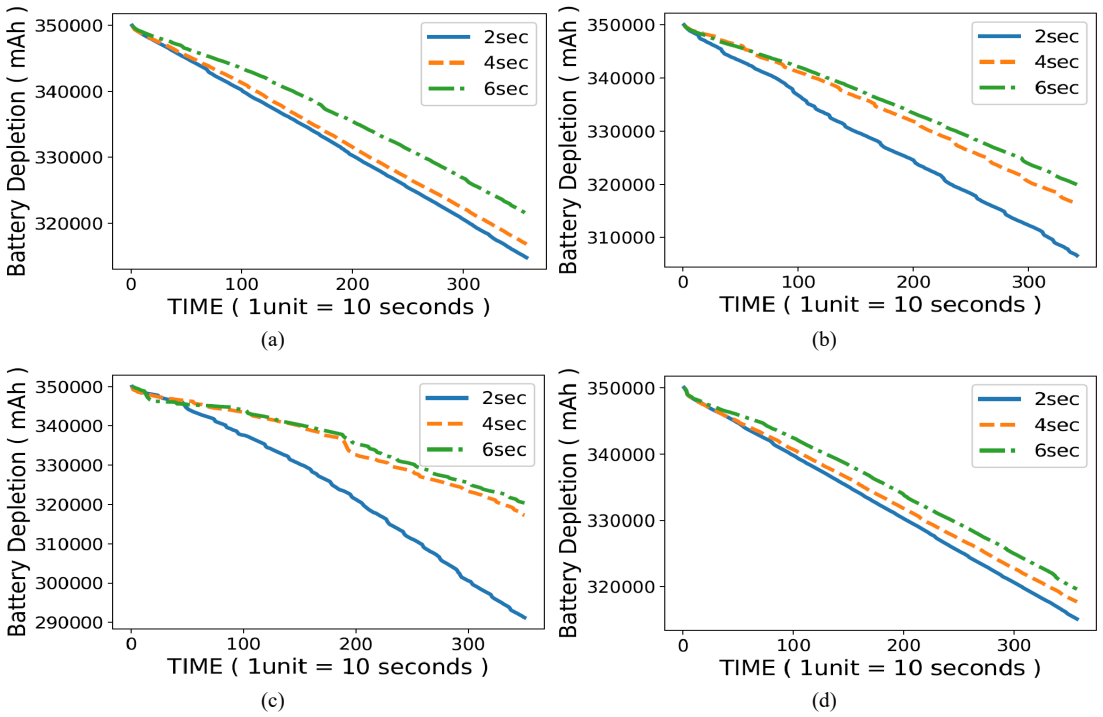
   c represents six seconds window.

Figure 4 Rate of battery depletion while using ECDF features as compared to the static features, (a) file to server = 5,000 ms (b) file to server = 10,000 ms (c) file to server = 20,000 ms (d) file to server = 40,000 ms (see online version for colours)



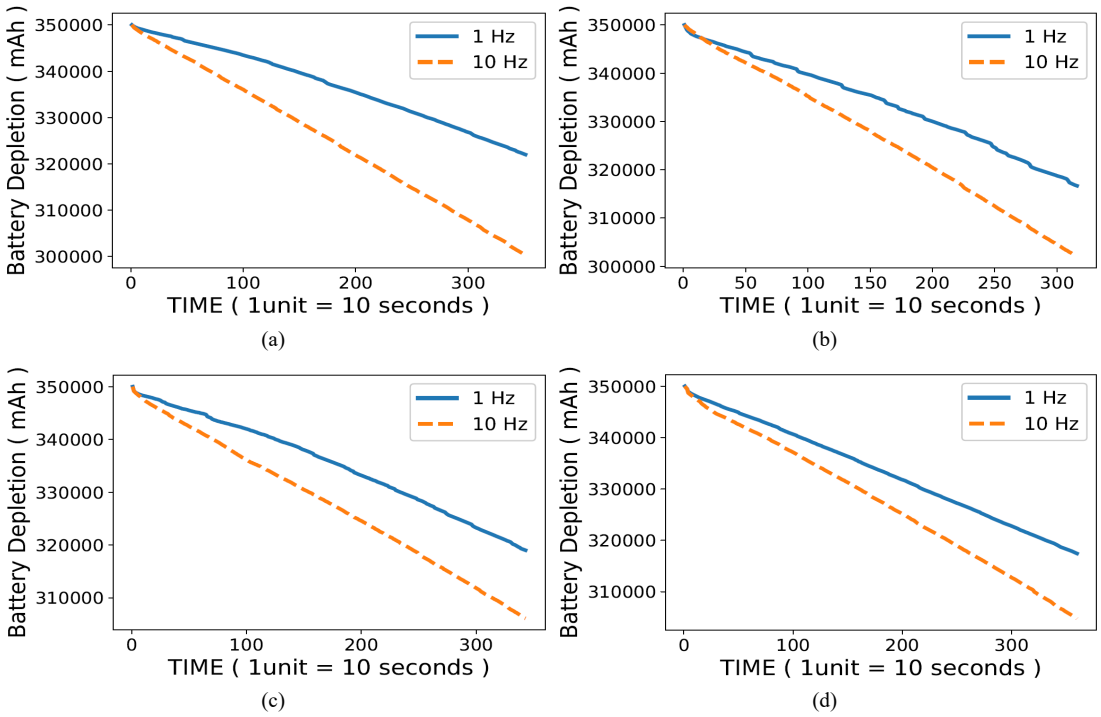
Notes: The X-axis represents the time where one unit represents ten seconds, and Y-axis represents the battery left (mAh) in the smartwatch.

Figure 5 Rate of battery depletion while taking different window length, such as, two, four, and six seconds, (a) file to server = 5,000 ms (b) file to server = 10,000 ms (c) file to server = 20,000 ms (d) file to server = 40,000 ms (see online version for colours)



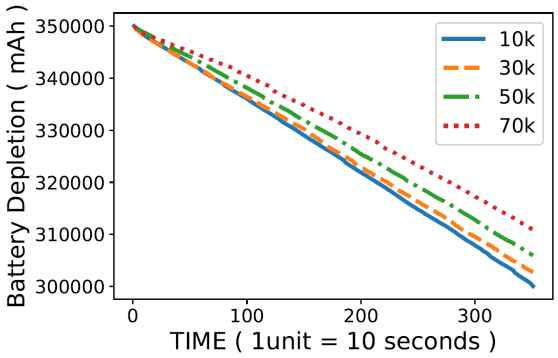
Notes: The X-axis represents the time where one unit represents ten seconds, and Y-axis represents the battery left (mAh) in the smartwatch.

Figure 6 Rate of battery depletion while using 1 Hz sampling rate as compared to 10 Hz sampling rate, (a) file to server = 5,000 ms (b) file to server = 10,000 ms (c) file to server = 20,000 ms (d) file to server = 40,000 ms (see online version for colours)



Notes: The X-axis represents the time where one unit represents ten seconds, and Y-axis represents the battery left (mAh) in the smartwatch.

Figure 7 Rate of battery depletion while varying the file upload rate to the server (see online version for colours)



Notes: The X-axis represents the time where one unit represents ten seconds, and Y-axis represents the battery left (mAh) in the smartwatch.

There is a refinement of at least 2 to 3% accuracy for each classifier to increase the window length. The higher window length denotes lesser amount of battery consumption (Figure 5) and more amount of data to recognise the activity. Similarly, when we increase the overlapping percentage of windows, there is an improvement of at least 2 to 3% in accuracy for each classifier.

2 *Effect of varying frequencies and pre-processing features on the F1-scores:* As we increase the frequencies of pings from the sensors, not just does the battery consumption increase (Figure 6) but the F1-scores boost too. Henceforth, the accuracies also increase. The higher the frequencies of sensors, the better the quality of information to predict. Likewise, we have chosen two types for pre-processing features – ECDF and statics. ECDF consumes additional battery as compared to statics (as shown in Figure 4) but gives more promising F1-scores and henceforth the accuracies for our 13 activities.

We see in Table 3, for 0% overlapping among windows and a two seconds window length, ECDF features give better F1-scores and accuracies as compared to statical features. The same observation can be seen in Table 4, i.e., for 50% overlapping among windows and a two seconds window length. In all of the eight tables, we see that the F1-scores and accuracies are better for 10 Hz than 5 Hz.

Please note that we alter the frequencies of sensor data by adjusting the modes and designating the timer at which we desire our pings. To work on 10 Hz data, we use the UI mode, but for 5 Hz, we downsampled the 10 Hz data.

5.4 Comparison of feature selection algorithm – TsFresh vs. ECDF vs. static features

Tsfresh (Patel et al., 2020b) provides a comprehensive set of feature extraction methods, ranging from simple statistical moments to more advanced features such as entropy, fractal dimension, and autocorrelation. The large

number of features (789 for each of the six axes) provided by tsfresh makes it a useful tool for time series analysis, as it provides a rich set of features that can be used to characterise the structure and behaviour of time series data. Tables 5(a) and 5(b) shows the accuracy comparison for 5 Hz and 10 Hz sampling data and we can observe that for four seconds and six seconds window the accuracy of Tsfresh is better as compared to static and ECDF. But in Tables 5(c) and 5(d) we can observe the computation time of Tsfresh as compared to static and ECDF is much much higher for both 5 Hz and 10 Hz sampling data. There are several other feature extraction tools that are similar to tsfresh such as – featuretools, tsfeat, rfeat, sktime, and tsaug which we will be exploring in depth in our future work.

5.5 Real-time testing

In the real-time testing, we use XGB classifier since it delivered the best accuracy for training and testing. We chose the data sampling rate of 10 Hz since it was an optimal choice among the two modes (normal and UI). We chose the file upload rate as once every 70,000 ms as it turns out to be more battery efficient and six seconds for the window length as it gives the best results of accuracy and battery efficiency. We choose an utterly different volunteer to conduct all activities in this live testing phase. The volunteer performed each activity for a varied amount of time; hence we considered the ratio of time duration of correctly classified instances and the time duration of the activity. We can see the results in Figure 12.

5.6 Key results

We deploy our models on lower frequencies, i.e., 10 Hz and 1 Hz. The four parameters (to reckon the battery depletion rate) varied among features, file upload rate, data sampling rate, and window lengths.

The two kinds of *features* taken into deliberation are ECDF and statistical, and among these two, ECDF being the complex one devours more battery as compared to statistical. On the other hand, if we analogise the accuracy score of both, we can observe that the ECDF accuracy is considerably better than statistical. *Hence, we report statistical to be energy-efficient and ECDF to be accuracy-driven*.

File upload rate is the time interval after which we send our file to the server for the prediction of results. There is a significant drop in server hits if we increase this time interval which automatically helps improve the battery efficiency. This number does not affect accuracy, but it should be as high as possible since it will be energy-efficient. Our watch used to hang when we tried for longer durations like an hour or so, hence we varied our experiments till 70 seconds only. *Hence, we report file upload rate to be 'as high as possible' to be an energy-efficient and independent parameter for accuracy*.

Data sampling rate is an Android out of the box provides us with four different frequencies with a facade – normal, UI, game, and fastest. The order of battery depletion goes like fastest > game > UI > normal. Since we are working on the battery efficiency goal, we consider UI mode (10 Hz data sampling rate) and normal mode (1 Hz sampling rate). The higher the frequency, the higher the information available to make a definitive decision and enhance accuracy. *Hence, we report 1 Hz (normal mode) to be energy-efficient and 10 Hz (UI mode) to be accuracy-driven*.

Lastly, we consider *window length* in which we take into account the time duration our classifier needs to learn and predict the activity. The higher the duration we provide, the better would be the accuracy, but it is valid till a specific limit. After a certain point, the accuracy stops increasing. That is what we scrutinised for window length beyond six seconds. We considered window lengths of two, four, and six seconds. Lower window length means an increased count of predictions made and hence an increase in battery consumption. *Hence, we report six seconds and above to be both energy-efficient and accuracy-driven*.

So the best set of parameters for good battery efficiency would be → statistical features, file upload rate is chosen as once every 70,000 ms, 1 Hz data sampling rate, six seconds or above window length and lastly XGB as classifier (no effect though since classifier is present on the server). The best parameters for good accuracy would be → ECDF features, file upload rate as once every 70,000 ms, 10 Hz data sampling rate, six seconds or above window length and lastly XGB as classifier (as shown in Table 6).

6 Conclusions

We develop a system that tells the user what kind of activity they are doing by taking data from the smartwatch that the user has worn on his wrist. We have a total of 13 activities:

1 cooking
2 sweeping
3 moping
4 walking
5 climbing up
6 climbing down
7 eating
8 driving
9 working on laptop
10 browsing on the phone
11 cycling
12 sitting in a car
13 watching TV.

Figure 8 Effect of different overlapping windows percentage on overall accuracy for 5 Hz frequency and ECDF features, (a) two seconds window (b) four seconds window (c) six seconds window (see online version for colours)

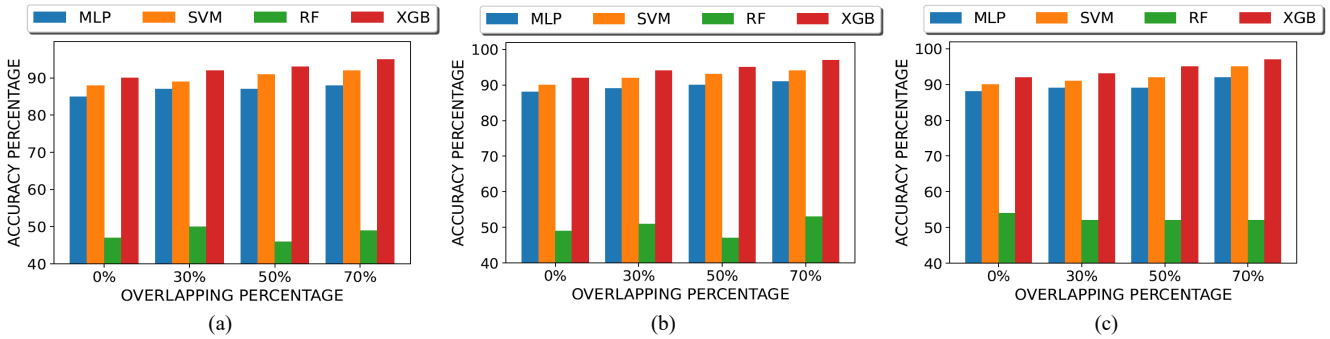


Figure 9 Effect of different overlapping windows percentage on overall accuracy for 5 Hz frequency and static features, (a) two seconds window (b) four seconds window (c) six seconds window (see online version for colours)

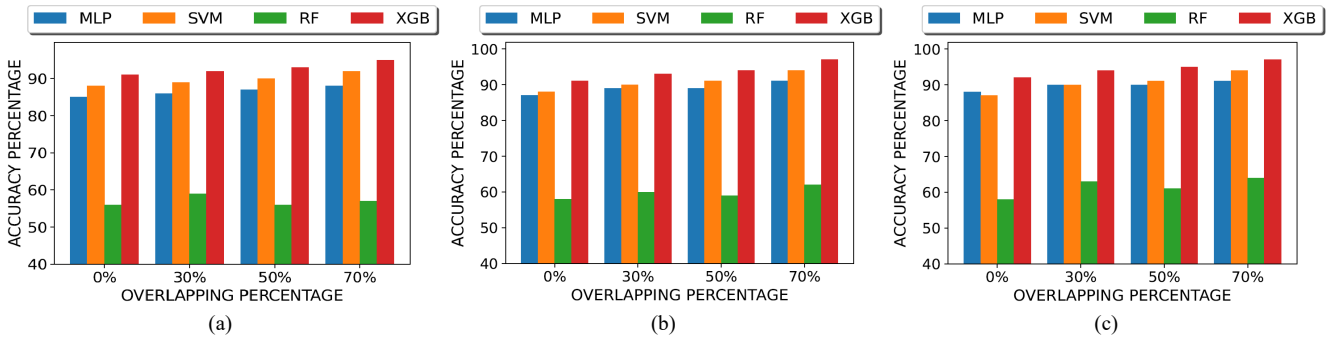


Figure 10 Effect of different overlapping windows percentage on overall accuracy for 10 Hz frequency and ECDF features, (a) two seconds window (b) four seconds window (c) six seconds window (see online version for colours)

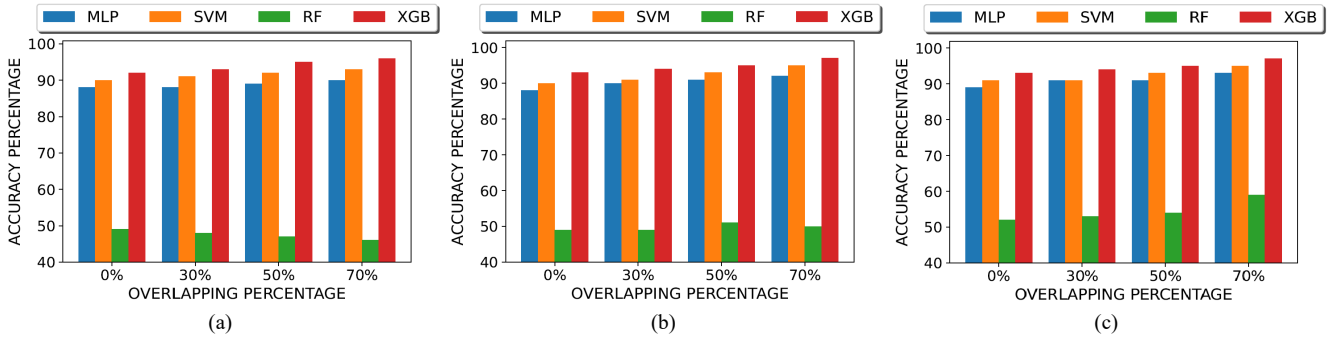


Figure 11 Effect of different overlapping windows percentage on overall accuracy for 10 Hz frequency and static features, (a) two seconds window (b) four seconds window (c) six seconds window (see online version for colours)

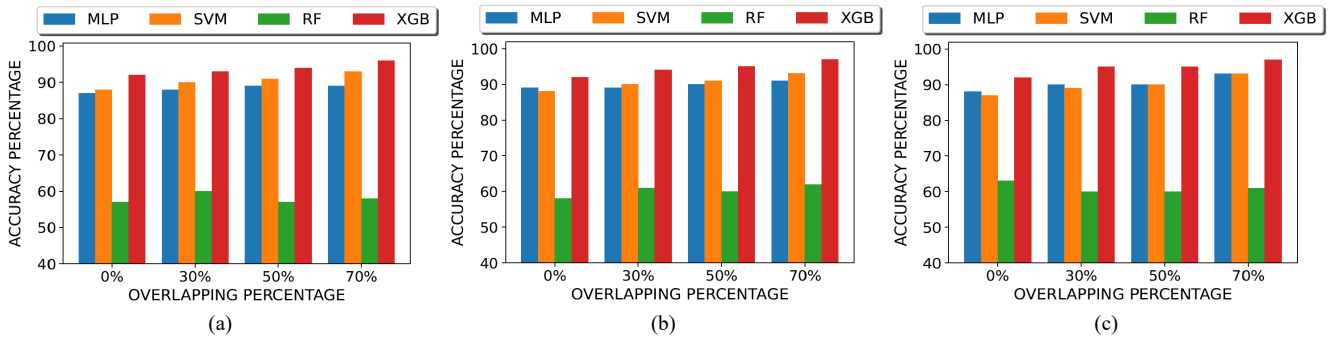


Table 3 F1-scores and overall accuracy from the strong classifiers for ECDF vs. static for overlapping $\theta = 0$ and window length = two seconds

(a) overlapping θ = 0, frequency = 5 Hz, features = ECDF									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.75	0.87	0.83	0.87	0.76	0.82	0.85	0.88	0.95
SVM	0.88	0.95	0.92	0.93	0.9	0.91	0.89	0.91	0.95
RF	0.56	0.42	0.06	0.45	0.02	0.09	0.41	0.12	0.71
XGB	0.95	0.97	0.94	0.97	0.96	0.97	0.96	0.96	0.97
AUTOML	0.88	0.92	0.89	0.93	0.89	0.92	0.92	0.95	0.98
(b) overlapping θ = 0, frequency = 5 Hz, features = static									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.72	0.84	0.81	0.83	0.71	0.8	0.85	0.88	0.93
SVM	0.76	0.9	0.86	0.87	0.79	0.86	0.85	0.88	0.94
RF	0.54	0.56	0.08	0.43	0.05	0.08	0.35	0.69	0.8
XGB	0.81	0.9	0.88	0.88	0.82	0.86	0.9	0.9	0.94
AUTOML	0.78	0.9	0.89	0.88	0.75	0.83	0.87	0.88	0.96
(c) overlapping θ = 0, frequency = 10 Hz, features = ECDF									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.79	0.9	0.83	0.87	0.78	0.85	0.85	0.89	0.94
SVM	0.9	0.96	0.93	0.95	0.93	0.95	0.89	0.92	0.96
RF	0.56	0.62	0.05	0.5	0.02	0.06	0.39	0	0.76
XGB	0.94	0.98	0.96	0.98	0.98	0.98	0.96	0.96	0.99
AUTOML	0.92	0.96	0.93	0.95	0.91	0.94	0.94	0.95	0.98
(d) overlapping θ = 0, frequency = 10 Hz, features = static									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.77	0.88	0.81	0.84	0.75	0.83	0.85	0.87	0.93
SVM	0.75	0.88	0.85	0.88	0.8	0.87	0.85	0.87	0.94
RF	0.54	0.68	0.09	0.46	0.02	0.04	0.35	0.48	0.8
XGB	0.83	0.91	0.88	0.88	0.85	0.88	0.9	0.91	0.95
AUTOML	0.84	0.91	0.87	0.89	0.82	0.9	0.87	0.89	0.94

Table 4 F1-scores and overall accuracy from the strong classifiers for ECDF vs. static for overlapping $\theta = 0.5$ and window length = two seconds

(a) overlapping θ = 0.5, frequency = 5 Hz, features = ECDF									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.78	0.88	0.83	0.87	0.77	0.83	0.84	0.9	0.94
SVM	0.9	0.96	0.95	0.94	0.89	0.92	0.92	0.94	0.96
RF	0.56	0.41	0.06	0.44	0.03	0.01	0.41	0.16	0.75
XGB	0.96	0.98	0.98	0.97	0.95	0.96	0.98	0.97	0.99
AUTOML	0.86	0.91	0.89	0.86	0.8	0.87	0.84	0.88	0.94
(b) overlapping θ = 0.5, frequency = 5 Hz, features = static									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.77	0.87	0.83	0.84	0.74	0.82	0.85	0.89	0.92
SVM	0.83	0.92	0.89	0.88	0.8	0.87	0.89	0.9	0.94
RF	0.56	0.62	0.08	0.46	0.04	0	0.36	0.27	0.82
XGB	0.87	0.91	0.92	0.91	0.84	0.89	0.93	0.94	0.96
AUTOML	0.85	0.91	0.91	0.87	0.81	0.88	0.88	0.9	0.96
(c) overlapping θ = 0.5, frequency = 10 Hz, features = ECDF									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.82	0.91	0.86	0.88	0.82	0.88	0.85	0.9	0.93
SVM	0.93	0.97	0.96	0.95	0.93	0.95	0.92	0.95	0.96
RF	0.55	0.61	0.03	0.52	0.02	0.01	0.4	0	0.72
XGB	0.98	0.98	0.98	0.98	0.97	0.98	0.98	0.99	0.99
AUTOML	0.87	0.9	0.89	0.89	0.83	0.9	0.86	0.89	0.95
(d) overlapping θ = 0.5, frequency = 10 Hz, features = static									
	Cook	*Sweep*	*Mop*	*Walk*	*Climb up*	*Climb down*	*Eat*	*Drive*	*Laptop*
MLP	0.8	0.9	0.86	0.87	0.78	0.85	0.88	0.91	0.95
SVM	0.84	0.93	0.9	0.91	0.83	0.9	0.88	0.91	0.95
RF	0.56	0.7	0.08	0.48	0.03	0	0.35	0.18	0.82
XGB	0.89	0.94	0.94	0.92	0.88	0.93	0.93	0.95	0.96
AUTOML	0.85	0.93	0.91	0.92	0.87	0.91	0.88	0.9	0.97

Figure 12 Live data accuracy for all activities (see online version for colours)

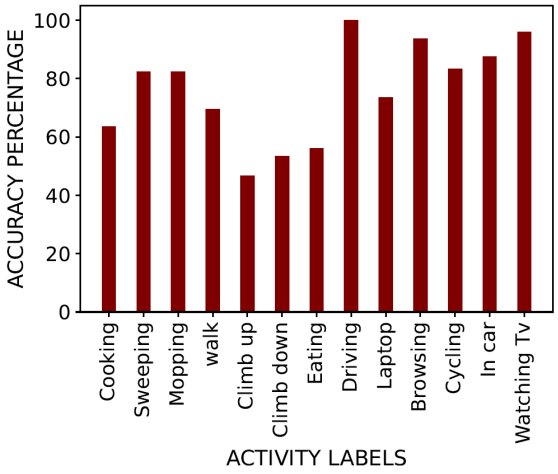


Table 5 Comparison of Tsfresh vs. static vs. ECDF features selection algorithms w.r.t. accuracies and computation time

(a) accuracies for 5 Hz sampling data			
10 Hz	*2 seconds*	*4 seconds*	*6 seconds*
Tsfresh	90%	93%	92%
Static	91%	91%	92%
ECDF	97%	92%	92%
(b) accuracies for 10 Hz sampling data			
10 Hz	*2 seconds*	*4 seconds*	*6 seconds*
Tsfresh	93%	94%	94%
Static	92%	92%	92%
ECDF	97%	93%	93%
(c) computation time for 5 Hz sampling data			
10 Hz	*2 seconds*	*4 seconds*	*6 seconds*
Tsfresh	3,931.671	2,283.186	1,636.572
Static	18.701	8.942	6.355
ECDF	22.159	11.342	8.442
(d) computation time for 10 Hz sampling data			
10 Hz	*2 seconds*	*4 seconds*	*6 seconds*
Tsfresh	4,324.854	2,593.414	1,933.929
Static	19.299	9.414	6.864
ECDF	23.561	11.907	8.327

The process to get this output is collecting the raw input data from the smartwatch sensors by pressing a start button and then dispatching the data to the server to get interpreted and passed through classification models deployed there. The communication between the smartwatch and the server happens through Wi-Fi connectivity. We evaluate the final results in terms of battery depletion rate for each parameter – pre-processing features (static and ECDF), File upload rate, data sampling rate as 1 Hz or 10 Hz, and timestamp length of the window. The battery depletion rate would enable a user to make the best choice for determining an optimal set of parameters. Since these classification models are not deployable on a smartwatch, we had to take support from the intermediate server, but in the future, we shall try to deploy everything on the smartwatch to mitigate the quotient of the internet availability.

Table 6 Key results

Parameters	*Summary on battery consumption*	*Summary on accuracy*	*Constants*
Features	Statistical (battery depletion – '38,000' milliamperes in one hour) and ECDF (battery depletion – '43,000' milliamperes in one hour)	Statistical (91%) and ECDF (97%)	Fixed data sampling frequency as 10 Hz, window length as six seconds, file upload rate as 50 k, and classifier as XGB
Data sampling frequency	1 Hz (battery depletion – '36,000' milliamperes in one hour) and 10 Hz (battery depletion – '43,000' milliamperes in one hour)	1 Hz (92%) and 10 Hz (97%)	Fixed features as ECDF, window length as six seconds, file upload rate as 50 k, and classifier as XGB
File upload rate	Once every 10 seconds – 50,000 ma in one hour, once every 30 seconds – 47,000 ma in one hour, once every 50 seconds – 43,000 ma in one hour, and once every 70 seconds – 39,000 ma in one hour	No effect on accuracy on changing this parameter	Fixed features as ECDF, data sampling frequency as 10 Hz, window length as six seconds, and classifier as XGB
Window length	2 sec – 43,000 ma in one hour, 4 sec – 33,600 ma in one hour and 6 sec – 30,000 ma in one hour	2 sec – 91%, 4 sec – 93%, and 6 sec – 97%	Fixed features as ECDF, data sampling frequency as 10 Hz, file upload rate as 50 k, and classifier as XGB
Classifier	No effect on accuracy on changing this parameter	MLP – 91%, SVM – 93%, RF – 56% and XGB – 97%	Fixed features as ECDF, data sampling frequency as 10 Hz, window length as six seconds and file upload rate as 50 k
	Best battery efficiency is approx 33,000 ma in one hour for an accuracy of 87%, parameters are – statistical features, normal mode, file to server once every 70 seconds and window length as 6 seconds	Best accuracy of 97% depletes approx 37,000 ma in one hour, pparameters are – ECDF features, UI mode, file to server once every 70 seconds and window length as six seconds	

We implement that if the user does not perform an activity from the above chosen 13 activities, then our model will point to another class termed the 14th class, i.e., 'others', we term this when there is no majority voting to any of the above-stated classes.

We used different clustering techniques to learn about the 'others' class, but it was impossible to separate the outliers from the actual plethora of activities in high-dimensional data. Henceforth, we applied the concept of dominating class (40% and above instances, if they belong to the same class for a particular window, then we say yes to that class; otherwise, the label is 'others'). We can change this threshold to 30% or 50%, depending on the level of strictness/mildness we want in our system.

References

Bayat, A., Pomplun, M. and Tran, D.A. (2014) 'A study on human activity recognition using accelerometer data from smartphones', *Procedia Computer Science, The 9th Intl. Conf. on Future Networks and Communications (FNC'14)/The 11th Intl. Conf. on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops*, Vol. 34, pp.450–457.

Bianchi, V., Bassoli, M., Lombardo, G., Fornacciari, P., Mordonini, M. and De Munari, I. (2019) 'IoT wearable sensor and deep learning: an integrated approach for personalized human activity recognition in a smart home environment', *IEEE Internet of Things Journal*, Vol. 6, No. 5, pp.8553–8562.

Cicirelli, F., Fortino, G., Giordano, A., Guerrieri, A., Spezzano, G. and Vinci, A. (2016) 'On the design of smart homes: a framework for activity recognition in home environment', *Journal of Medical Systems*, Vol. 40, No. 9, pp.1–17.

Dewan, A. et al. (2019) 'A hierarchical classifier for detecting metro-journey activities in data sampled at low frequency', in *MoMM 2019: The 17th Intl. Conf. on Advances in Mobile Computing & Multimedia*, pp.46–55.

Dewan, A., Gunturi, V.M.V., Naik, V. and Dutta, K.K. (2021) 'Neat activity detection using smartwatch at low sampling frequency', in *2021 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pp.25–32.

Guo, H. et al. (2016) 'Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble', in *Proc. of Intl. Joint Conf. on Pervasive and Ubiquitous Computing*, pp.1112–1123.

Hamasaki, H., Ezaki, O. and Yanai, H. (2016) 'Nonexercise activity thermogenesis is significantly lower in type 2 diabetic patients with mental disorders than in those without mental disorders', *Medicine*, January, No. 2.

Inoue, S., Ueda, N., Nohara, Y. and Nakashima, N. (2015) 'Mobile activity recognition for a whole day: recognizing real nursing activities with big dataset', in *Proc. of Intl. Joint Conf. on Pervasive and Ubiquitous Computing*, pp.1269–1280.

Issa, M.E. et al. (2022) 'Human activity recognition based on embedded sensor data fusion for the internet of healthcare things', in *Healthcare*, Basel.

Jiang, X. et al. (2016) 'Air: recognizing activity through IR-based distance sensing on feet', in *Proc. of the 2016 ACM Intl. Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pp.97–100.

Krishnan, N.C. and Cook, D.J. (2014) 'Activity recognition on streaming sensor data', *Pervasive and Mobile Computing*, Vol. 10, No. B, pp.138–154.

Levine, J.A. (2002) 'Non-exercise activity thermogenesis (NEAT)', *Best Practice and Research Clinical Endocrinology and Metabolism*, Vol. 16, No. 4, pp.679–702.

Levine, J.A., Vander Weg, M.W., Hill, J.O. and Klesges, R.C. (2006) 'Non-exercise activity thermogenesis: the crouching tiger hidden dragon of societal weight gain', *Arteriosclerosis, Thrombosis, and Vascular Biology*, Vol. 26, No. 4, pp.729–736.

Mannini, A. (2015) 'Activity recognition using a single accelerometer placed at wrist or ankle (Vol. 45, p.2193, 2013)', *Medicine and Science in Sports and Exercise*, Vol. 47, No. 2, pp.448–449.

Milette, G. and Stroud, A. (2012) *Professional Android Sensor Programming*, 1st ed., Wrox Press Ltd., Birmingham, UK.

Murao, K. and Terada, T. (2014) 'A recognition method for combined activities with accelerometers', in *Proc. of the Intl. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct Publication, UbiComp '14 Adjunct*, pp.787–796.

Ordóñez, F.J. and Roggen, D. (2016) 'Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition', *Sensors*, Vol. 16, No. 1, p.115.

Owen, N., Bauman, A. and Brown, W. (2009) 'Too much sitting: a novel and important predictor of chronic disease risk?', *British Journal of Sports Medicine*, Vol. 43, No. 2, pp.81–83.

Patel, D., Hsu, W. and Lee, M.L. (2008) 'Mining relationships among interval-based events for classification', in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, Association for Computing Machinery, New York, NY, USA, pp.393–404.

Patel, D., Shrivastava, S., Gifford, W., Siegel, S., Kalagnanam, J. and Reddy, C. (2020a) 'Smart-ML: a system for machine learning model exploration using pipeline graph', in *2020 IEEE International Conference on Big Data (Big Data)*, pp.1604–1613.

Patel, D., Shah, S.Y., Zhou, N., Shrivastava, S., Iyengar, A., Bhamidipaty, A. and Kalagnanam, J. (2020b) 'Flops: on learning important time series features for real-valued prediction', in *2020 IEEE International Conference on Big Data (Big Data)*, pp.1624–1633.

Ponce, H. et al. (2016) 'A novel wearable sensor-based human activity recognition approach using artificial hydrocarbon networks', *Sensors*, Vol. 16, No. 7, p.1033.

Ronao, C.A. and Cho, S-B. (2016) 'Human activity recognition with smartphone sensors using deep learning neural networks', *Expert Systems with Applications*, Vol. 59, pp.235–244, ISSN: 0957-4174.

Shoaib, M., Bosch, S., Incel, O.D., Scholten, H. and Havinga, P.J.M. (2014) 'Fusion of smartphone motion sensors for physical activity recognition', *Sensors*, Vol. 14, No. 6, pp.10146–10176.

Skocir, P., Krivic, P., Tomeljak, M., Kusek, M. and Jezic, G. (2016) 'Activity detection in smart home environment', *Procedia Computer Science, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016*, Vol. 96, pp.672–681.

Stisen, A. et al. (2015) 'Smart devices are different: assessing and mitigating mobile sensing heterogeneities for activity recognition', in *Proc. of the Conference on Embedded Networked Sensor Systems*, SenSys, pp.127–140.

Sun, L. et al. (2010) 'Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations', in *Ubiquitous Intelligence and Computing*, pp.548–562.

Sun, X., Qiu, L., Wu, Y. and Cao, G. (2017) 'Actdetector: detecting daily activities using smartwatches', in *2017 IEEE 14th Intl. Conf. on Mobile Ad Hoc and Sensor Systems (MASS)*, pp.1–9.

Tapia, E.M. et al. (2007) 'Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor', in *11th IEEE Intl. Symp. on Wearable Computers*, pp.37–40.

Tapia, E.M., Intille, S.S. and Larson, K. (2004) 'Activity recognition in the home using simple and ubiquitous sensors', in Ferscha, A. and Mattern, F. (Eds.): *Pervasive Computing*, pp.158–175, Springer, Berlin, Heidelberg.

Thiemjarus, S., Henpraserttae, A. and Marukatat, S. (2013) 'A study on instance-based learning with reduced training prototypes for device-context-independent activity recognition on a mobile phone', in *2013 IEEE International Conference on Body Sensor Networks*, pp.1–6.

Villablanca, P.A., Alegria, J.R., Mookadam, F., Holmes, D.R., Wright, R.S. and Levine, J.A. (2015) 'Nonexercise activity thermogenesis in obesity management', *Mayo Clinic Proceedings*, Vol. 90, No. 4, pp.509–519.

Zhang, M. and Sawchuk, A.A. (2012) 'USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors', in *Proc. of the 2012 ACM Conf. on Ubiquitous Computing*, pp.1036–1043.

Zhang, S., McCullagh, P., Nugent, C. and Zheng, H. (2010) 'Activity monitoring using a smart phone's accelerometer with hierarchical classification', in *2010 Sixth International Conference on Intelligent Environments*, pp.158–163.

Zheng, L. et al. (2017) 'A novel energy-efficient approach for human activity recognition', *Sensors*, Vol. 17, No. 9, p.9.