This work is made freely available under open access.

AUTHOR: BASHIR, S.A.

TITLE: Change detection for activity recognition.

YEAR: 2017

OpenAIR citation:
BASHIR, S.A. 2017. Change detection for activity recognition. Robert Gordon University, PhD thesis.

This work was submitted to- and approved by Robert Gordon University in partial fulfilment of the following degree:
Doctor of Philosophy, School of Computing Science and Digital Media.

OpenAIR takedown statement:
Section 6 of the "Repository policy for OpenAIR @ RGU" (available from http://www.rgu.ac.uk/staff-and-current-students/library/library-policies/repository-policies) provides guidance on the criteria under which RGU will consider withdrawing material from OpenAIR. If you believe that this item is subject to any of these criteria, or for any other reason should not be held on OpenAIR, then please contact openair-help@rgu.ac.uk with the details of the item and the nature of your complaint.

This ůKŜäłă is distributed under a CC BY-NC 4.0 license.

https://creativecommons.org/licenses/by-nc/4.0

OpenAIR at RGU Digitally signed by OpenAIR at RGU Date: 2018.08.31 16:00:16 +01'00'

Change Detection for Activity Recognition

SULAIMON ADEBAYO BASHIR

A thesis submitted in partial fulfilment of the requirements of Robert Gordon University for the degree of Doctor of Philosophy

November 2017

Abstract

Activity Recognition is concerned with identifying the physical state of a user at a particular point in time. Activity recognition task requires the training of classification algorithm using the processed sensor data from the representative population of users. The accuracy of the generated model often reduces during classification of new instances due to the non-stationary sensor data and variations in user characteristics. Thus, there is a need to adapt the classification model to new user characteristics. However, the existing approaches to model adaptation in activity recognition are blind. They continuously adapt a classification model at a regular interval without specific and precise detection of the indicator of the degrading performance of the model. This approach can lead to wastage of system resources dedicated to continuous adaptation.

This thesis addresses the problem of detecting changes in the accuracy of activity recognition model. The thesis developed a classifier for activity recognition. The classifier uses three statistical summaries data that can be generated from any dataset for similarity based classification of new samples. The weighted ensemble combination of the classification decision from each statistical summary data results in a better performance than three existing benchmarked classification algorithms.

The thesis also presents change detection approaches that can detect the changes in the accuracy of the underlying recognition model without having access to the ground truth label of each activity being recognised. The first approach called 'UDetect' computes the change statistics from the window of classified data and employed statistical process control method to detect variations between the classified data and the reference data of a class. Evaluation of the approach indicates a consistent detection that correlates with the error rate of the model. The second approach is a distance based change detection technique that relies on the developed statistical summaries data for comparing new classified samples and detects any drift in the original class of the activity. The implemented approach uses distance function and a threshold parameter to detect the accuracy change in the classifier that is classifying new instances. Evaluation of the approach yields above 90% detection accuracy. Finally, a layered framework for activity recognition is proposed to make model adaptation in activity recognition informed using the developed techniques in this thesis.

Keywords: Activity Recognition, Change Detection, Classification Algorithm, Sensors.

Declaration of Authorship

I declare that I am the sole author of this thesis and that all verbatim extracts contained in the thesis have been identified as such and all sources of information have been specifically acknowledged in the bibliography. Parts of the work presented in this thesis have appeared in the following publications:

- S.A. Bashir, D.C. Doolan, A. Petrovski "The Impact of Feature Vector Length on Activity Recognition Accuracy on Mobile Phone" Proceedings of the International Conference on Knowledge Engineering and Data Mining London, U.K., 1 - 3 July, 2015.

- S.A. Bashir, D.C. Doolan, A Petrovski " The Effect of Window Length on Accuracy of Smartphone-Based Activity Recognition" IAENG International Journal of Computer Science 43 (1), 126-136.
  (**Part of Chapter 2**)

- S.A. Bashir, D.C. Doolan, A. Petrovski "ClusterNN: A Hybrid Classification Approach to Mobile Activity Recognition" MoMM '15 The 13th International Conference on Advances in Mobile Computing and Multimedia. 11-13 Dec. ACM New York 2015.
  (**Chapter 4**)

- S.A. Bashir, D.C. Doolan, A. Petrovski "Clustering and Nearest Neighbour Based Classification Approach for Mobile Activity Recognition" Journal of Mobile Multimedia Volume 12 12 , 110-214.
  (**Chapter 4**)

- S.A. Bashir, A. Petrovski, D. Doolan “UDetect: Unsupervised Concept Change Detection for Mobile Activity Recognition ” MoMM ’16 The 14th International Conference on Advances in Mobile Computing and Multimedia. 29-30 Nov. ACM New York 2016.
  (**Chapter 5**)

- S.A. Bashir, A. Petrovski, D. Doolan ”A Framework for Unsupervised Change Detection in Activity Recognition” International Journal of Pervasive Computing and Communication 13(2) Pages 157-175.
  (**Chapter 5 and 6**)

Acknowledgements

First, I thank Almighty Allah for making it possible for me to complete this thesis. I express my profound appreciation to my supervisors for their guide and feedbacks on my work.

I also wish to express my sincere gratitude to my wife for her support throughout the period of my study. My wonderful kids Abdusshakur, Abdurroheem, Muhammad and Maryam were very patient throughout the period of my absence to play my direct fatherly role in their upbringing. I thank them for being such lovely and nice kids by keeping their good morals and maintaining high morale. I cannot forget the support I received from the Nigerian Muslim Community in Aberdeen. They provide both unflinching spiritual and outstanding moral support during my entire stay in Aberdeen. I pray that may Almighty Allah reward them in manifolds. I am also grateful to my colleagues at N426, the staff of CSDM and all those who played a part in making this research possible.

This acknowledgement will not be complete if I failed to thank my home government, *The Federal Republic of Nigeria* for providing the generous funding under the auspices of *The National Information Technology Development Agency (NITDA)* to carry out this research. I also acknowledge the support of my employer, *The Federal University of Technology, Minna* for granting me a study fellowship for this research. Finally, I would like to express my gratitude to my late mother; Alhaja Sariyu Oluwalose (May her soul be admitted to paradise) for her special love, care, prayers and efforts to raise me.

Contents

List of Figures

List of Tables

Dedicated to my late mother Alhaja Sariyu Oluwalose (may her soul rest in paradise); my beloved wife Lateefah and my lovely kids: Abdusshakur, Abdurroheem, Muhammad, Maryam and Ali.

Chapter 1

Introduction

Sensor rich devices have become commonplace items in the hands of billions of people worldwide. They are gaining improvements in computing power and are equipped with sensors such as accelerometer, gravity, gyroscope, microphone Bluetooth, and other wireless communication interfaces. This advancement has revolutionised the way people interact and open up innovations in many other application areas. For example sensors on the mobile phone can be utilised individually and collectively to support myriads of applications in healthcare [6], environment monitoring [57] and location awareness [60]. These applications can enhance the user experience by way of providing context-rich information to the users at the right time and place.

Context recognition is one of the application areas of ubiquitous computing. The elements of context include activity [9], surrounding sound events [57], emotion state [76], location [60] and phone state [89]. Monitoring of these contexts can be used to develop applications for providing custom services to the users. For example, monitoring of user physical activities can facilitate the development of health and fitness applications. It can also be used to monitor physical activity of people with motion related ailments to provide an emergency alert to caregivers. Also, users can collaboratively share their activity status. For example, a context service integrated with a phone book can automatically show the activity state of each contact to enable the user to know the status of the contact list and decides whether to call them or not in such situations.

Activity recognition is a component of context that is useful to most applications that customise their services to the user. However, to realise the full potential of developing scalable activity recognition in the face of the diverse user population and non-stationary sensor data, the need to provide adaptive and personalised recognition cannot be underestimated. A good activity recognition system should be able to adapt to different environment and users.

The traditional approach to motion-based activity recognition involves the collection of sample data from a group of users who perform the designated set of target activities to be recognised by the system. The sensor data obtained is pre-processed to extract more informative features to represent the raw data. After this, the data is used to train a classifier that can distinguish different activity patterns from the training set. The assumption behind this practice is that the training data contains all the possible variations that would be encountered in the new unseen data in the future. However, this assumption may not hold, given the fact that, it is not possible to anticipate all possible variations that may occur during recognition time. Different variations can happen in the data due to differences in user characteristics that are not captured during the training time. Research has shown that pattern of activity of a user can serve as a form of personal identification due to the uniqueness of how gait characteristics vary from user to user [43]. Thus, it is practically challenging to create a one-size-fits-all model that caters for all user characteristics embodied in variations in age, weight, gender and other environment factors for activity recognition.

The reasons for the variations in users that lead to model inaccuracy across different users can be attributed to various user characteristics such as weight, age, gender and height. Also, data from the device sensors may exhibit variations due to different positioning of the device during training and recognition. Catering for all these situations would require an enormous data collection efforts to cover all the possible anticipated changes which are impractical and costly. Hence, adaptive systems capable of incremental learning have been proposed as a better solution. Such systems will adapt to the non-stationary nature of the evolving data and subsequently updates the classifier model so as to improve its accuracy on the fly. However, a continuous adaptation of the model without precise detection of degradation in the accuracy of the classifier will lead to consumption of system resources and reduce the efficiency of the recognition model. Therefore, this thesis presents approaches that focus on developing change detection framework that is non-intrusive and resource efficient for motion-based activity recognition.

1.1 Problem Context

The different characteristics possessed by various categories of people introduce a unique challenge to activity recognition. The reason being that there is always a variation between how two users perform the same activity [46, 100]. Also variations exist within individual ways of performing an activity, which may depend on the person well-beings such as stress, level, emotional state, e.t.c. A classifier that is trained from the preliminary data of some users will encounters reduced performance when used by another user with varying characteristics from the reference training data. Model adaptation has been the main approach to cater for reduced classifier accuracy as a result of the aforementioned variations [47]. The adjustment of the model is carried out by updating the recognition model at a periodic interval with new ground-truth information. This adaptation is normally carried out without detecting the change that leads to model inaccuracy. This blind adaptation without precise detection of the indication of variations is a wasteful task. Thus, change detection in the underlying model will eliminate the blind adaptation and allow the system to be used optimally.

Change detection in activity recognition is a unique and challenging problem because it encompasses two intertwined domain of change detection and concept drift detection. In one part, change detection considers detecting variations in an unlabelled data stream while concept change detection focuses on detecting changes in the concept being learned from a stream. That is, the emphasis is on detecting changes in the conditional distribution of the output labels while the conditional distribution of the input attributes remains unchanged [30]. Thus, many of the approaches in this regard rely on the ground truth of the predicted label for monitoring the model degenerated accuracy. However, since drift in activity recognition does not only depends on the change in the distribution of output variable and the ground truth of the predicted activity is not readily available, many of the approaches for real concept drift detection will not work in this case. Similarly, approaches that focus mainly on detecting changes in the distribution of the attributes have shortcomings since the change in activity recognition data does not only manifest in the attributes but also in the simultaneous changes in the labels.

Therefore, to effectively detect a change in activity recognition where there may be a simultaneous change in both the input and output data distribution in the absence of ground truth, techniques suited for this unique problem is required. Hence, the work in this thesis developed frameworks that are uniquely tailored to the challenging task of change detection in activity recognition. In order to address the issue of eliminating blind adaptation of activity recognition model, two main research questions are formulated as follows:

1. How can change in accuracy of activity recognition model be detected in the absence of ground truth labels of the types of activities?
2. How can statistical data characteristics be used to identify changes in activity recognition model accuracy?

1.2 Aim and Objectives of the Research

The aim of this research is to develop techniques for change detection in activity recognition. The source of changes in activity recognition model often arises from the variations in users' characteristics, and it manifests in degenerating performance in the accuracy of the recognition model. The specific objectives of the research are:

1. Develop a framework to incorporate change detection in activity recognition model without access to ground truth for error monitoring.
2. Develop a classification algorithm that utilises reduced statistical data summaries derived from the original data for weighted ensemble nearest neighbour classifier.

3. Develop change detection techniques that in part utilised the baseline data as reference data to detect changes in the new set of classified examples.

1.3 Key Contributions of the Thesis

This thesis contributes to the development of algorithms for change detection in activity recognition. A change detection framework for activity recognition is developed. The framework leverages on the data characteristics without recourse to ground truth to detect classifier model degrading performance. The primary contributions of this thesis are the following:

1. **Development of data characteristics from activity patterns:** The thesis proposed three statistical data characteristics that capture the patterns in an activity dataset. The three data characteristics proposed included the centre, minimum and maximum which, are obtained from micro-clusters created from an activity class. These three characteristics capture the boundaries of a particular pattern in the activity dataset, and they are used as a summary for the whole data in the micro-cluster.

2. **Development of a clustering and nearest neighbour based classifier (ClusterNN) for Activity Recognition:** The classifier employs the statistical data summaries to compute the similarity between new instances and the data summaries. The weighted ensemble combination of the classification decisions from each data summary results in a better performance than three existing benchmarked classification algorithms widely used for activity recognition.

3. **Development of Unsupervised Change Detection Techniques:** To avoid blind adaptation, change detection techniques were developed to monitor the evolution of the performance of activity recognition model. Two approaches for change detection in activity data developed contributed to knowledge in the following ways:

- Unlike other approaches for change detection that utilise ground truth label to monitor the error rate of a model, the detection framework developed here can detect classifier errors without access to the ground truth.
- The approach developed can handle both multi-dimensional and unidimensional data change detections. This method is more useful in the real scenarios of the multi-sensor data stream for activity recognition compared to other approaches that deal with unidimensional data only.

1.4 Thesis Structure

This thesis has seven chapters. The rest of the thesis is organised as follows:

Chapter 2 Literature Review: This chapter provides the theoretical and practical underpinning of activity recognition systems. It identifies the various schemes of adaptation in activity recognition, all of which are blind adaptations without explicit detection schemes. This chapter also discusses various approaches for change detection in unlabelled data in one hand and that of labelled data in another hand.

Chapter 3 Benchmark Algorithms and Datasets: This chapter presents background details about the general research framework of this thesis. The chapter also describes the datasets used for the evaluation of the base learning algorithm and the change detection techniques. Also, the chapter presents the evaluation metrics employed in the research.

Chapter 4 ClusterNN: A Clustering and Nearest Neighbour Based Classifier for Activity Recognition: presents the base classification algorithm developed in this thesis. The algorithm builds on the concept of nearest neighbour and clustering to develop an hybridised algorithm that is suitable for the mobile environment with limited computational resources.

Chapter 5 Unsupervised Accuracy Change Detection for Activity Recognition: Existing approaches to concept change detection rely on the availability of ground truth labels to monitor the error rate of the online classification algorithm. The monitoring of the change in the distribution the error rate serves as the indicator for the change detection in the underlying distribution of the data. This thesis presents a novel approach that does not rely on the availability of ground truth in other to detect the changes in the model accuracy degradation. This chapter presents the methods developed and the evaluation of the method in addition to the results obtained.

Chapter 6 Distance Based Change Detection Technique for Activity Recognition: While the previous chapter is dedicated to detecting the structural change in the accuracy of the model. The chapter presents another improved approach that locates the actual point in time the accuracy of the underlying model begins to degrade.

Chapter 7 Conclusion: This chapter concludes the thesis with the proposed layered framework for activity recognition system based on the developed techniques. It also states the summary of the contributions of the thesis and the proposed relevant future work.

Chapter 2

Literature Review

2.1 Introduction

Recognition of user activities is an important task in pervasive computing. It provides contextual information to the computing platform so as to adapt their functions to the users' context. The physical activities of a user are part of the context that can be fed to a computing platform in order to provide enhanced computing services to facilitate quality user experience. The recognition of activity has a wide area of application such as personalised healthcare monitoring, surveillance and context aware awareness [55, 77, 81]. There are various kinds of sensors for recognising activity depending on the application areas and the types of activity to be recognised. However, a key issue in activity recognition is the need to adapt a recognition model to the varying user characteristics. Various approaches have been prescribed, but they all have the demerit of being blind approaches. Thus, the need for a change detection in the model before starting the adaptation process.

The rest of this chapter is organised as follows: Section 2.2, gives the definition of activity and identify its different categories. Section 2.3 provides a brief overview of the process of machine learning approach for activity recognition. Section 2.4 focused on the discussion of the different type of sensors that are being used for activity recognition. Section 2.5 presents the different approaches to model adaptation. In Section 2.6 the methods of change detection in unlabelled data stream are examined while Section 2.7 discusses the methods of change detection in labelled data stream. Section 2.8 outlines the conclusion from the literature and the summary of the chapter follows in Section 2.9.

2.2 Definition and Types of Activities

Activity recognition is the computational task of detecting the state of physical action of a user from sensor data at a particular point in time. There are many types of activities that have been subject of research in the literature and there is no general consensus on their categorization. An attempt to categorise activities is made by Huynh [36], who proposed 3 categorization of activities namely: gestures, low level and high level activities. Gestures are brief and distinct body part movements such as taking a step, bending the arm, waive the arm e.t.c. These activities can occur within few seconds. Low-level activities on other hand are short duration sequences of bodily movements (running, walking) or a distinct stationary posture (standing or sitting). The recognition and monitoring of low-level activities has been a topic of interests over the years. Shahmohammadi et. al. [85] collected smartwatch accelerometer sensor data on five low-level activities of *running, walking, standing, sitting, and lying down* and evaluated the accuracy of four supervised machine learning algorithms on the collected data and the efficiency of active learning in improving the baseline classifier accuracy. Further examples of low-level activity recognition studies were conducted in [15, 20, 42, 68]. These show that activity recognition is a topical subject with lot of interest. High-level activities are composed of a set of low-level activities, for example, the high-level action of shopping consists of series of locomotion task like walking, standing. A distinctive subset of high level activities known as activity of daily living or instrumental activity of daily living are also common in the literature due to their importance in supporting healthy living in intelligent homes such as monitoring elderly activities in homes, and assisted living of people with limited and impaired mobility [98].

Furthermore, an in-depth look at the literature reveals that there are other types of activities which have received the attention of activity recognition researchers that do not fit directly into the categorization given above. For example, health-related activities such as sleep and stress [6, 46] and fall detection [23, 31, 32] and workshop or assembly activities [59] have also been researched in the literature. A survey on activity recognition [51] codifies a flexible categorization of activities as presented in Table 2.1. These activities can be recognised with sensor data from smartphone sensors or other types of wearable sensors.

Group	**Activities**
Ambulation	Walking, running, sitting, standing still, lying, climbing stairs, descending stairs, riding escalator, and riding elevator.
Transportation	Riding a bus, cycling, and driving.
Phone usage	Text messaging, making a call.
Daily activities	Eating, drinking, working at the PC, watching TV, reading, brushing teeth, stretching, scrubbing, and vacuum
Exercise and fitness	Rowing, lifting weights, spinning, Nordic walking, and doing push ups.
Military	Crawling, kneeling, situation assessment, and opening a door.
Upper body	Chewing, speaking, swallowing, sighing, and moving the head.

Table 2.1: Categories of Activity Adapted from [51]

2.3 Supervised Machine Learning Approach to Activity Recognition

Activity recognition using supervised learning approach involves the use of collected sample of activity data to train a classification algorithm to recognise similar activities from new unseen samples. The generic process of this method can be broken down into four basic tasks namely: (i) Sensor Sampling and Pre-processing (ii)

Segmentation and Feature Extraction (iii) Online or Offline Training of classifier (iv) Model Generation and Deployment.

2.3.1 Sensor Sampling and Pre-processing

The first step in supervised machine learning approach to activity recognition is the sampling of sensor data. Pre-processing steps are required to be applied to the raw data to ensure their reliability and precision. Typical pre-processing approaches include sensor calibration and smoothing. Sensor Calibration is used to eliminate the effects of zero-offset and drift sensitivity error sometimes encountered in accelerometer sensor data. Zero-offsets are incorrect data obtained from the sensor other than the actual expected values. For example, a device at rest is expected to give a magnitude acceleration of $9.8m/s^2$. This value is the sum of acceleration forces along each of the three axes with an x-axis and y-axis having zero values and z-axis having $9.8m/s^2$ due to the gravity effect. Any value greater or less than this is an offset error. Calibration requires the computation of scaling factor and offset values along each orthogonal axis of the accelerometer to compensate for the drift and offset error. A traditional approach to calibration is to determine the positive and negative g-force along each axis of the device [109]. This process is done by precise positioning of the device along the direction of gravity to determine the positive and negative g-forces experienced along its six orientations. After obtaining these values, the offset and the scale factor of each axis is determined with Equations 2.1 and 2.2 respectively. The positive values are the $\{ b_1, b_2, b_3 \}$ while the negative values are the $\{ b'_1, b'_2, b'_3 \}$. The equations show only the calibration for the x-axis. The other axes are computed the same way.

$$offset_x = \frac{b_1 + b'_1}{2} \tag{2.1}$$

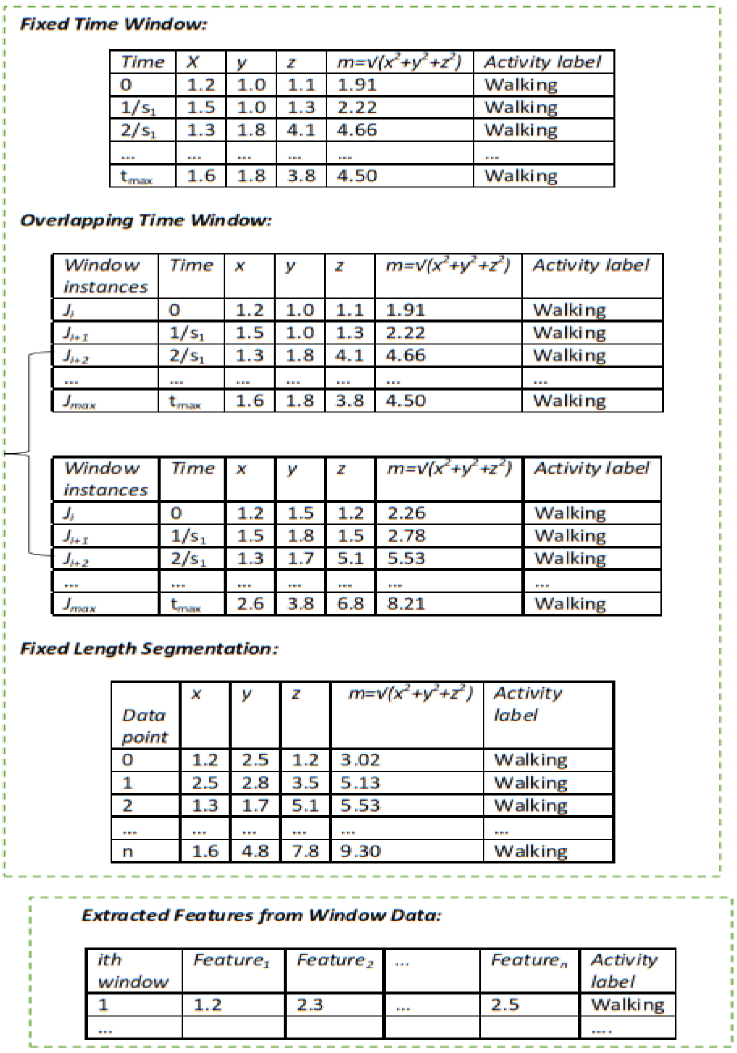
$$scale_x = \frac{|b_1 - b'_1|}{2} \tag{2.2}$$

The demerit of this approach stems from the difficulty of determining the precise *g-direction* to which the device must be pointed. Also, the procedure needs to be repeated several times to obtain accurate values. Doing this will be cumbersome for the end user. Thus a more sophisticated and automatic calibration techniques for accelerometer data have been presented in [108] and [56]. Smoothing or filtering can also be applied to ensure continuous elimination of jitter in the signal. Common filtering techniques include moving average, low pass filter and high pass filter.

2.3.2 Segmentation and Feature Extraction

Segmentation is the process of grouping streaming time series raw sensor data into manageable chunks that contain enough data to characterize an activity. The amount of sample data within a segment or window is dependent on the predefined window data collection strategy. Various windowing strategies have been proposed. Figure 2.1 shows three common approaches to windowing. The first approach collects chunks of data within every fixed period defined by the system while the overlapping window method combines a fixed amount of previous window data with the current window data by a predefined proportion. The third approach does not employ time. The system collects a predefined amount of data of fixed size n from the continuous flow of streaming sensor data. The overlapping window has the advantage of catering for activity transition points while the rest do not.

The segmentation step is necessary because the point by point streaming sensor data is inadequate to capture the signal variations associated with a given activity. Also, an activity can span a period of seconds or minutes. Thus, a single raw sample point cannot contain adequate information to differentiate the performed activity [51].



Fixed Time Window:

Time	*X*	*y*	*z*	*$m=\sqrt{(x^2+y^2+z^2)}$*	*Activity label*
0	1.2	1.0	1.1	1.91	Walking
$1/s_1$	1.5	1.0	1.3	2.22	Walking
$2/s_1$	1.3	1.8	4.1	4.66	Walking
...	...	...	...	...	...
t_{max}	1.6	1.8	3.8	4.50	Walking

Overlapping Time Window:

Window instances	*Time*	*x*	*y*	*z*	*$m=\sqrt{(x^2+y^2+z^2)}$*	*Activity label*
J_i	0	1.2	1.0	1.1	1.91	Walking
J_{i+1}	$1/s_1$	1.5	1.0	1.3	2.22	Walking
J_{i+2}	$2/s_1$	1.3	1.8	4.1	4.66	Walking
...	...	...	...	...	...	...
J_{max}	t_{max}	1.6	1.8	3.8	4.50	Walking

Window instances	*Time*	*x*	*y*	*z*	*$m=\sqrt{(x^2+y^2+z^2)}$*	*Activity label*
J_i	0	1.2	1.5	1.2	2.26	Walking
J_{i+1}	$1/s_1$	1.5	1.8	1.5	2.78	Walking
J_{i+2}	$2/s_1$	1.3	1.7	5.1	5.53	Walking
...	...	...	...	...	...	...
J_{max}	t_{max}	2.6	3.8	6.8	8.21	Walking

Fixed Length Segmentation:

Data point	*x*	*y*	*z*	*$m=\sqrt{(x^2+y^2+z^2)}$*	*Activity label*
0	1.2	2.5	1.2	3.02	Walking
1	2.5	2.8	3.5	5.13	Walking
2	1.3	1.7	5.1	5.53	Walking
...	...	...	...	...	...
n	1.6	4.8	7.8	9.30	Walking

Extracted Features from Window Data:

ith window	*$Feature_1$*	*$Feature_2$*	...	*$Feature_n$*	*Activity label*
1	1.2	2.3	...	2.5	Walking
...					....

FIGURE 2.1: Segmentation of Raw Data with Windowing Approach

Feature Extraction is the process of extracting more meaningful information from each segment of data obtained during the segmentation process. Time and frequency domain features are the two basic categories of features that are commonly used in inertial based activity recognition. *Time domain features* are basic statistical quantities computed over the raw data captured in a window. They can be calculated from the individual orthogonal axis or the magnitude of the three axes data. The descriptions and formulae of some of the commonly used statistical features as described in [109] and [51] are summarised in the Table 2.2.

Frequency domain features on the other hand are derived by transforming raw data in the window into the frequency domain by using Fast Fourier transform (FFT). The FFT coefficients obtained from the transform serve as the amplitudes

Features	**Formulae**	**Description**
Mean	$X = \frac{1}{n}\sum_{i=1}^{n} x_i$	The mean represents the DC component of the motion signal.
Root Mean Squares	$rms = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$	It captures the the intensity of the signal and invariably the activity being performed.
Variance	$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - X)^2$	As a measure of dispersion, this feature indicates level of dynamics of the signal, differentiating between low dynamic activity such as walking and high dynamic activity such running.
Standard Deviation	$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - X)^2}$	Same as variance.

Table 2.2: Time Domain Features

of the frequency components of the signal and its energy distribution. A number of these coefficients can serve as a feature vector, and other features such as energy (Equation 2.3) can be derived from the coefficients as well.

$$E = \frac{\sum_{i=1}^{n} FFT_{coef_i}}{n} \tag{2.3}$$

The energy feature captures the level of intensity of the accelerometer signals and invariably the activity intensity. The features extracted from the raw data will allow machine learning algorithms to discriminate better between one activity data and another rather than using ordinary raw data.

2.3.3 Model Generation and Deployment

Once the data has been processed they are ready to be used for inducing a classification model. Several supervised machine learning algorithms such as K-nearest neighbour (KNN), decision tree, Naive Bayes and Gaussian discriminant analysis have been proposed by different authors. Preece et al. [73] presents a detailed review of many of the existing work on activity recognition. The model generation can be performed directly on the device that is used for data collection or on a remote server system. The choice of the platform for model generation depends on the scale of deployment of the resulting model. A single-user activity recognition can utilise model generated on the user's device while a large population activity recognition will require model generated on a remote system by using a large pool of data from many users. The generalised population model is prone to inaccurate recognition due to differences in users characteristics while single-user model based on each user data results in duplication of efforts. Thus, the compromise is to use model adaptation to maintain the accuracy of the model.

2.4 Sensor Modalities for Activity Recognition Systems

Sensors are crucial components in activity recognition. There are many types of sensors that can be used for activity recognition. Sensors can be used singly to recognise a particular kind of activity or combined to improve recognition accuracy. Multi-modal context recognition can also be done by combining sensors of different types. The choice of sensors depends on factors such as the type of context being sensed, the level of intrusiveness of the sensor, ease of use, level of recognition precision desired, availability and cost. This section presents the commonly used sensors in activity recognition which include, wearable sensors, environment-installed sensors and smartphone sensors. Wearable sensors are attached to the user body e.g. 3D accelerometers; environment-installed sensors are fixed into a position in the home or office environment e.g. wireless sensor network and camera while smartphone sensors consist of many types of sensors integrated into mobile devices.

Inertial Sensors

These sensors are by far the most widely used sensors for activity recognition. Their usage stems from the fact that they are minuscule and can be easily carried as wearable sensors attached to the users' body. Besides, they have low cost and are usable under different environments both indoor and outdoor to recognise ambulatory activities. 3-D accelerometers are the usual inertial sensors used in activity recognition research. They are usually packaged in compact cases and attached to the users at specific positions on their body parts, such as waist, arms, knees, ankles and hips. An accelerometer measures the magnitude and direction of acceleration forces along the three reference axes. These values obtained reflect the level of intensity of the force experienced by the device which invariably indicates the type of activity the user is performing. Before the advent of smartphones, researchers used body-worn accelerometers for recognising activity. The pioneering work of Intille and Bao [12] examined the use of sensors worn on different body parts to accurately determine physical activities. Similar works in [54, 63, 70] also focused on activity recognitions using body worn sensors.

Audio Sensor

The sound encountered in the surrounding of a user while performing an activity is a good clue to the kind of activities. Audio based sensing is useful for recognising high-level activities of daily living such as driving and vacuuming the floor by extracting characteristics sounds associated with such activities. Stork et al. [93] presented a microphone based system that can recognise 22 different sounds corresponding to some human activities in a bathroom and kitchen context. They extracted mel-frequency cepstra coefficients (MFCC) from sound samples and applied segmentation free approach that yielded a recognition rate of over 85%. Another work presented by Zhan and Kuroda [45] employed Haar-like sound features with HMM to recognise 22 different activities related to personal cooking using such as vacuum cleaning, shaving and drinking and other groups related to social events such as shopping and outside dining. They claimed an average accuracy of 97% given their laboratory settings.

Although the use of audio for activity recognition is yielding promising results as reported in the works cited above, the major challenges in real world situations are the interference produced by background noise. Another limitation is that sound cannot be used to recognise some activities such as ambulation and sports activities since they do not produce characteristics sound to discern their audio patterns.

Image Sensor

Use of camera for activity recognition has been studied extensively in the literature. There are two main approaches to recognising human actions from video i) a direct recognition of human action from video sequences ii) inferring the actions through observation of phenomenon such as location and object in the environment of the user. The former is often hard to achieve due to variations in motion, recording settings and inter-personal differences [72, 96]. The latter approach is more practical and commonly used in recognising activity of daily living in the home to monitor the inhabitant activity. For example, Duong et al. [24] presented a work that installed multiple cameras in a room which observes a user performing different actions. They divided the room into regions of squares with the various objects of interest in each region. The list of visited regions is used to infer the actions of the user such as watching television. The major limitation of image sensors is that they can only observe the activity of objects within the vicinity of their installed location. They are not pervasive enough to recognise activity of a user outside their surroundings.

Object Use through Wireless Sensor Network and RFID

Another approach for capturing data to infer users' activities is through their interaction and contact with some sensors in the environment. Wireless sensor networks are used for activity recognition by connecting them to objects that people interact with inside smart homes. The sensor node is usually equipped with various types of sensors such as contact switches, pressure mats, mercury contacts, passive infrareds sensors, humidity sensors and temperature sensors. Data from these sensors indicate activities such as sleeping using pressure mats sensors, bathing using humidity sensors in the bathroom, cooking using temperature sensors in the kitchen and in-the-room using contact sensors [98, 106]. Similarly, RFID (Radio-frequency identification) is also used by attaching tags to objects in the house, and users with RFID reader which can read information stored in those tags. The activity is inferred based on the tag information on the objects interacted with by the user. Philipose et al. [71] implemented an activity recognition that attached RFID tags with objects and user with a glove embedded with RFID readers to recognise their activities.

Physiological and Vital Signs Sensors

These categories of sensors are used to infer activities by sensing physiological parameters of the wearer of the device. The magnitudes of those parameters indicate the intensity of the actions that the user is performing. Vital signs data such as ECG (electrocardiogram), GSR (galvanic skin response), skin temperature and oxygen saturation are being used to improve activity recognition. Parka et al. [70] examined the use of physiological sensors for activity recognition and concluded that they did not provide more useful data for activity recognition. They observed that physiological signals correlate with the intensity level of activities, they are not adequate to reflect the type of activity because physiological signal reacts slower to activity changes. In another work, Tapia et al. [94] demonstrated an activity recognition system that combines data from heart rate monitor and five accelerometers. They arrived at the same conclusion that physiological sensors did not respond well to activity changes. However, Lara et al. [52] assert that by performing additional feature extraction on the physiological data, they can be used to improve recognition accuracy. The main drawback of physiological sensors is their high level of intrusiveness as the user has to attach the sensors to the basal skin to read the signals.

Smartphone Sensors

Today's mobile phones are becoming more and more powerful with many types of sensors packaged into them. A typical smartphone has many of the individual sensors discussed in the previous sections thereby making them a good platform to infer more fine-grained user context. Incel et al. [37] presented a taxonomy of mobile phones based activity recognition.They identify three approaches to inferring activities using a cell phone to include i) location, ii) motion and iii) hybrid phone-based activity recognition.

Location-driven recognition uses the location information of the users to infer their activities. The primary focus was to recognise activities associated with some places [19]. Reality Mining Project [25] used Bluetooth and cell tower data to model user location and regular activity such as being at home, work and elsewhere. They employed HMM algorithm conditioned on the hour of the day and day of the week to build a predictive model and recorded a 95% accuracy.

Motion-based Activity Recognition uses phone embedded sensors like accelerometer, gyroscope, cellular, Wi-Fi and GPS radio data from moving users to infer their activities. Accelerometer enables inference of user activity by capturing the acceleration force experienced by the device. The intensity of this force correlates to the intensity of the activity that is being performed. Works carried out by Bertchold et al. [13], Alvian and Muhammad [82] and many others utilised mobile phone accelerometers for recognition of activity. Sohn et al. [92] used GSM data collected by mobile phones to recognise walking, driving, stationary and daily step count activities of users. They used fingerprinting principle, where GSM observations with a stable set of towers and signal strengths are used to infer not moving and changes in the set of nearby towers and signal strengths indicate motion. Another related work presented by Anderson et al. [7] tracks daily exercises activities of users and allow individual user to share and compare their activity level to others. They employ an Artificial Neural Network (ANN) to analyse GSM cell signal strength and visibility. The multitude of sensors on mobile phones allows the inference of more fine-grained context information. Cenceme [64] draws inferences from GPS, Bluetooth, accelerometer and audio data to infer user context. Lane et al. [46], Alqassim et al. [6] and Natale et al. [67] utilized phone embedded sensors to infer user sleep patterns for well-being monitoring.

It can be concluded that smartphones are versatile tools for performing sensing of user context at a different level of details. The advantage of using smartphone sensors for activity recognition stems from the ubiquity of the phones and popularity of usage among billions of people worldwide. It is less intrusive, and users would not hesitate to carry the phones with them always.

2.5 Approaches to Model Adaptation in Activity Recognition

Model adaptation is the process of adapting a base recognition model to the variation in the new unseen data which manifests in decreasing accuracy of the model. The aim is to make the model usable across many people with diverse characteristics such as age and gender. The process involves creating an initial model from labelled training data which is then deployed into an application. The model is adapted later to reflect the current characteristics of the new user. This section examines the several approaches that have been proposed in the literature for model adaptation and personalization of activity recognition. The approaches can be broadly categorised into three namely: Model re-training with crowd-sourced data, personalised single user model and incremental model adaptation.

2.5.1 Model Re-training

The model re-training approach adopts the strategy of updating an existing activity recognition model by replacing it with a newly created model after re-training. The new updated models are created by adding data from a target user to the *already existing data* to re-train the classification algorithm that creates a new model. The update can also be made by combining the target user data with the crowd-sourced data that is similar to the data of the target user. The later approach was adopted in [13, 48] while the former approach was adopted in [5, 13, 33, 41]. Lane et al. [48] proposed Community Similarity Network (CSN) which uses similarity metrics between target user's data and other subscribed users' data to generate a new model to replace existing one periodically.

CSN has a client and a server components. The client side runs on mobile phones. It uses initial model downloaded from the server side to classify and store new data from the phone sensors. The stored data comprising of both user-labelled and unlabelled data is periodically uploaded to the cloud-hosted server. These ground truth data are combined with other users' data in the server to build similarity networks and train an updated model which is pushed to the client periodically. The similarity networks for each user is constructed by using crowd-sourced data from other users. The network is composed of nodes and edges. Nodes represent users connected by weighted edges to the target user. The weights on the edges indicate the level of similarity between users regarding physical, lifestyle and sensor data characteristics. The physical similarity metric between $user_i$ and $user_j$ is computed using the mahalanobis distance between the vectors according to Equation 2.4:

$$sim(\overrightarrow{x}_i, \overrightarrow{x}_j) = \exp(-\gamma(\overrightarrow{x}_i - \overrightarrow{x}_j)^\top)\Sigma^{-1}(\overrightarrow{x}_i - \overrightarrow{x}_j) \tag{2.4}$$

$\overrightarrow{x}_i$ $\overrightarrow{x}_j$ are vectors of values of age, height, weight, well-being and physical activity scores denoting physical characteristics of the two users. Σ is the covariance matrix and γ is a scaling factor determined empirically. Lifestyle similarity is computed according to equation 2.5:

$$sim(i, j) = \sum_{f \in F} T_{f(i)}^\top T_{f(j)} \tag{2.5}$$

where $T_{f(i)}$ represents lifestyle variable of type f for $user_i$ from the set F that contains the GPS mobility patterns, diurnal patterns and distribution of activities performed. GPS mobility patterns were obtained by accumulating GPS location data and taking the frequency of how often the user stays in a location. Activity distribution are computed by finding the frequency of each activity performed while the diurnal lifestyle is computed by accumulating the number of times the user is active for a particular period of time in the week scaled between 0 and 167hrs.

Sensor data similarity, on the other hand, is computed by applying Locality Sensitive Hashing (LSH) to generate a histogram that represents the collection of each user data. The sensor data similarity between users is calculated from the histogram representation of the data through LSH using equation 2.6.

$$sim(i,j) = T_{f(i)}^{\top} T_{f(j)} \tag{2.6}$$

The model re-training phase in the cloud employs the data from the weights of the similarity networks to train three separate classifiers, one from each type of network data using boosting. Finally, the three classifiers are unified and pushed to the mobile clients to replace any existing model there. The major drawback of this approach is that the process of generating the similarity networks is cumbersome and generating a model for each subscribed user is un-scalable as population of clients increases. The method also does not cater for detection of changes before initiating the adaptation process which makes it updating the model unnecessarily. This technique is refined in [5] to cater for scalability by generating a single model for users within a similarity network and uses a natural language technique for removing error in crowd-sourced labels. Although, it also does not cater for detection of model errors before adapting the model. Berchtold et al. [13] presented a service based approach for personalised activity recognition. Their system employs split-level architecture with mobile phones and server-based components. Personalization is achieved by user-generated data that is sent to the server for training a personalised model. The performance is reported to be 97.3% if the training data of the user evaluating the system is used and the phone is kept in the same body position like pants pocket as it was during collection phase. However, the accuracy drops to 60% if the data is collected with the phone placed at different locations. If the target user data is excluded from the training set, the accuracy rate varies between 63.3% and 80% which indicates the importance of personalization. However, this personalization or adaptation method does not incorporate detection of changes, thereby unduly wasting resources.

2.5.2 On-the-Phone Personalised Model Generation

The approaches in this category aim to eliminate the need for adaptation by using individual user data for model generation on the mobile phone without recourse to the cloud or pre-generated model on a computer. In this case, there is no data collection from any other users to pre-train the model. It is only the data from the target user that are collected and used to build a model on the phone directly. At the beginning of using the activity recognition system, users are asked to perform sample activities and the data collected during this set-up phase are used to generate a personalised model.

The use of online personalised activity recognition has been explored as alternative to generalised model adaptation and many of such work are presented in [87]. Kose et al. [41] proposed an improved KNN algorithm for personalised phone based activity recognition. Initially, the approach extracted four features from the raw accelerometer values. They extracted k maximum and minimum data points from each class of activity in the raw x,y,z axes values of the accelerometer dataset. They also extracted k nearest data points to the average of the datasets and standard deviation of the whole datasets in each class. These data points and their corresponding class tags are employed during recognition phase by using nearest neighbour computation with test instances. The proposed approach in Chapter 4 is different form this approach because it explores the patterns in the dataset by applying clustering to the data in each class to reveal the patterns in a given class of activity. Then the patterns in each sub-cluster are summarized by extracting the three statistical summaries that represent the boundary of each sub-cluster. Also, while this approach arbitrarily select K data points, the clustering routines in the proposed approach in Chapter 4 is based on the desired level of data retention of the system.

Bartolo et al. [33] also presented a personalised mobile activity recognition system that performs online training and classification of activity data collected from the target user. They proposed an incremental Naive Bayes algorithm for incremental learning of the underlying activities. The system relies on the user provided labelled data for incremental training and updating the model as required. The work opine that by using an incremental Naive Bayes algorithm, the model can be updated with more of the same user data based on the changing profile of the user.

The main drawback of these approaches is the need for each user to provide data individually to train their models. Also, they do not incorporate change detection to detect when the accuracy of the model degenerates.

2.5.3 Incremental Model Adaptation

Incremental adaptation approach utilizes stepwise adaptation of the initial generalised model to a user specific model by utilizing self-labelled data provided by the user to tune the generalised model to the target user. This approach is different from those presented in section 2.5.1 because they do not perform an explicit retraining of a classification algorithm to generate new model. Rather, they adopt mechanisms that tune the existing recognition model to the target user. The adaptation is carried out incrementally when the user is using the model.

An incremental model adaptation method is presented by Abdallah et al. [3, 4]. The two presented algorithms built offline classifiers by applying similarity measures between various cluster parameters obtained from the dataset and the new cluster of sample from the new user.

The parameters extracted from each cluster include centroid, standard deviation and density. These are defined in Equations 2.7 to 2.9:

$$Centroids = \sum_{i=1}^{n} \frac{x_i}{n} \tag{2.7}$$

$$Density = \frac{Weight}{Volume} \tag{2.8}$$

where weight is the number of sample in the cluster and $volume = \frac{4}{3}\pi Radii^n$ with $Radii = max(Euclidean(x_i, Centroid)\,\forall\, x_i \in cluster$

$$StandardDeviation = \sqrt{\frac{\sum_{i=1}^{n} (Euclidean(x_i, Centroid))^2}{n}} \tag{2.9}$$

During the classification step, clustering is applied to a window of raw accelerometer data points to extract the same parameters as the stored ones. The algorithm uses four measures to compare the stored cluster characteristics and the new one. The new instance is then assigned to the appropriate class using majority voting to determine the final class. The measures are used to compare the data:

- Euclidean distance between stored cluster centroids and incoming cluster centroid. The label of the stored centroid with the minimum distance to the new one is assigned to it.
- Gravitational force between the stored clusters and the incoming cluster centroid. The label of the stored centroid with highest force is assigned to the new cluster.

$$GravitationalForce = \frac{Weight_{centroid_i} Weight_{newcentroid}}{Euclidean(centroid_i,\ newcentroid)^2} \tag{2.10}$$

- Density gain when the new cluster is merged with each stored cluster. The label of the stored cluster with highest density gain is assigned to the new one.
- Arithmetic difference between the stored standard deviations of the reference clusters and the standard deviation of the new cluster is computed. The label of the cluster that gives minimum value is assigned to the new cluster.

After using these measures for classification, a final majority voting is applied to assign a final class label to the new cluster. The adaptation of the model is performed by requesting for user input whenever two measures vote for the same class while the other two votes for different classes or when two measures or more vote for different classes. There is no adaptation when the majority or all measure vote for a class.The adaptation algorithm updates the cluster parameters of the closest cluster to the new user-labelled cluster. A key fundamental drawback of this model adaptation approach is that there is no mechanism to detect explicitly the changes in the accuracy of the model. Also, the detection of misclassification is based on the classifier decisions and not a separate detection method. The detection cannot be used to detect changes in accuracy of any other classification algorithm because it is built into the classifier majority decision.

Another incremental adaptation proposed by Froster et al. [26] is based on learning algorithm guided by error feedback mechanism. The scheme employs weighted KNN with weight assigned to each instance. Initially all instances have a weight set to 1 and each class is represented with a model given in Equation 2.11,

$$M_{c,t} = \{((x^{(1)}, y_i), w_{1,t}), ((x^{(2)}, y_i), w_{2,t}), ..., ((x^{(n)}, y_i), w_{1,t})\} \tag{2.11}$$

having set of examples $x^i \in^n$, weight $w_{i,t} \in^1$ and class label $y = \{c_1, c_2...c_n\}$. Thus, at any point in time t, the classifier C_t has n models corresponding to the number of classes in the classification problem $C_t = \{M_{1,t}, M_{2,t}, ..., M_{n,t}\}$. To classify a new instance x_t, the K- nearest neighbour is determined and the sum of weights of the instances belonging to the same class are computed. The new instance is classified to the class with highest sum of weights among the nearest neighbour instances, $c_t = argmax_{c_i} SW(c_i)$.

Once the classification is performed, the error feedback mechanism based on provision of correct and incorrect indicator (s_t) is used for adaptation. If the classifier gives a correct prediction, the indicator is set to correct otherwise it is set to incorrect. The two signals determine the adaptation step to take. For the case of correct indicator, the classified instance x_t^i, the predicted class c_t and the correct signal $s_t = correct$ are used for adaptation based on Algorithm 1.

According to the Algorithm 1, the classified instance is added to the set of nearest m point of model $M_{c,t}$ to which the instance is classified, and the weight of those instances are incremented. Conversely, if the prediction is not correct the adaptation procedure is based on Algorithm 2. In this case, the weights of the nearest m points to x_t from model $M_{c,t}$ to which the new instance is classified are

Algorithm 1: Correct Signal Adaptation [26]

```
Input: $(x_t, c_t, w_t = 1)$ // new correctly classified instance with correct label and
   weight=1
1  $M_{nn} = NN(M_{(c,t)}, x_t, m)$
   // obtain m nearest neighbours of $x_t$ from $M_{(c,t)}$
2  for each $(x^{(i)}, c_i)$ in $M_{nn}$ do
3      if $c_i == c_t$ then
4          $w_{(i,t+1)} = -\frac{(w_{(i,t)}-2)^2}{2}$
5      end
6  end
```

decremented. Afterwards, instances with weights less than a set threshold k are removed from the model $M_{c,t}$.

Algorithm 2: Incorrect Signal Adaptation [26]

```
Input: $(x_t, c_t)$ // new classified instance with label
1  $M_{nn} = NN(M_{(c,t)}, x_t, m)$ obtain m nearest neighbours of $x_t$ from $M_{(c,t)}$
2  for each $(x^{(i)}, c_i)$ in $M_{nn}$ do
3      if $c_i == c_t$ then
4          $w_{(i,t+1)} = \frac{{w_{(i,t)}}^2}{2}$
5      end
6  end
7  for each $(x^{(j)}, c_j, w_{j,t})$ in $M_{c,t}$ do
8      if $w_{j,t} < k$ then
9          $remove(x^{(j)}, c_j, w_{j,t})$
10     end
11 end
```

The quadratic functions for weight adaptation are meant to ensure the balance between low and high weight values. The high weight values $(w_{i,t} > 1)$ are less affected than the low weight values $(w_{i,t} < 1)$ such that when the system is in a stationary state, the weights will be high $(w_{i,t}\ 2)$ and when in non-stationary state, the weight will be low $(w_{i,t} << 2)$. This is to ensure more time to adapt the classifier. Furthermore, the correct learning for a particular class c with model $M_{c,t}$ is stopped if the sum of weights of the instances exceed the mean of the highest and lowest sum of weight of all classes K and a factor $(1+\alpha)$ where $\alpha \in [0,1]$.

$$\sum_w M_{c,t} > \frac{\max_K(\sum_w M_{K,t}) + \min_K(\sum_w M_{K,t})}{2}.\ (1+\alpha)\ \ \forall\ K = c_i..c_n$$

Similarly, the incorrect learning is halted when the following condition is fulfilled:

$$\sum_w M_{c,t} < \frac{\max_K(\sum_w M_{K,t}) + \min_K(\sum_w M_{K,t})}{2}.\,(1-\alpha)\ \ \forall\, K = c_i..c_n$$

Also, the author proposed the stoppage of incorrect adaptation for a class c if the number of instances in its model $M_{c,t}$ is less than or equal a threshold δ.

The nearest neighbour used for classification is maintained by by ensuring that at any point in time t , k is proportional to some instances in the classification models C_t that is $k_t = R.n(C_t)$, $R \in \{0,1\}$. Also, the amount of nearest instance m used during adaptation of the model is maintained by: $m_t = LR.n(C_t)$, $LR \in \{0,1\}$

This approach requires user input to confirm correct or incorrect classification. This method is not practical in real life as it will be cumbersome for the user to provide such indicator continuously. Also, as other approaches mentioned, it does not detect the changes before initiating the adaptation process. Although, mechanism for terminating the adaptation is proposed the same mechanism for detecting the model inaccuracy change is not proposed.

The preceding approaches may be regarded as supervised adaptation because they require user input. Unsupervised adaptation does not require the effort of the user during the adaptation process. The major difference between the two is that the output of a classifier prediction in unsupervised adaptation is used directly for incremental model update without requiring the user to give correct label if the classifier makes a wrong prediction. This is directly opposite of supervised adaptation that relies on user input to either indicate a correct or incorrect signal or directly provides the ground truth label for wrongly predicted instance by the classifier before performing the model update. Foster et al. [26] proposed an unsupervised self adapting nearest cluster centre adaptation mechanism. The adaptation proceeds after the classifier classifies an instance x^i into a class y_i. The label is used to adapt the corresponding cluster centre CC_i nearest to the instance. This cause the class centre at time i to be moved to CC_{i+1}. The centroid move is performed according to the update rule:

$$CC_{i+1} = (1 - \psi).CC_i + \psi.x^i$$

where ψ is the learning rate that determines the rate of movement of the centre. Zhao et al. [110] employed transfer learning approach that incorporates k-means clustering algorithm and decision tree. They generate a model for user A using decision tree and use the model to classify unlabelled data of user B. The result of the classification of user B is then used to adapt the decision tree model of user A for user B by clustering new input to correlate the data of user B during classification by decision tree.

The advantage of these approaches is that they do not require the intervention of the users to provide labelled data. This is more practical and less intrusive since most users will not be committed to monitoring a recognition system and provide labelled data for all unclassified instance. However, they do not incorporate the detection of change before adaptation.

2.6 Change Detection in Unlabelled Data Stream

An unlabelled data stream is a sequence of data emanating from a source (such as industrial process monitoring sensors) independently without the need to register the data as belonging to a category. The data is solely dependent on the distribution of the source of the data. It can be univariate, that is, having only one attribute or it could be multivariate whereby the attributes are more than one and are distributed jointly according to the source of the data.

Change detection in an unlabelled data stream is concerned with the identification of variations in the *descriptive parameters* of the data distribution such as mean and variance. There are various methods for detecting changes in univariate and multivariate data stream. This section discusses the various categories of techniques for change detection in an unlabelled data stream.

2.6.1 Sequential Change Detection

Sequential change detection are methods that test the sample data as they arrive for a change point in contrast to the traditional hypothesis testing where all the data must be collected before performing the test. A sequential test is carried out to determine if there is a change in the parameter of the data being observed up until the moment of the test. The test can be performed as the data arrived one at a time or as they arrived in batches depending on the problem settings and the technique put in place to detect the change.

The detection can be based on the utilisation of a statistical test of hypothesis or a threshold constant to detect when a significant change occurs in the *parameters* of the data stream being monitored. The threshold value is preset such that when the value of the test statistic exceeds the threshold, change is declared. An hypothesis test defines a null and an alternative hypothesis and a test statistic. The null is rejected when the value of the test statistic is beyond the critical value. Otherwise, the alternative hypothesis is accepted. However, if the test fails to decide on the null or alternative hypothesis the test continue by taking more samples from the rest of the data and repeat the procedure. The following are the major techniques of sequential change detection.

Sequential Probability Ratio Test

This is a classical sequential test developed by Wald in 1947 [99] in the context of industrial production quality control problems. The test originally aims to ascertain that a set of items from many production output items meets the desired specifications. The test procedure involves the setting up of null and alternative hypothesis that corresponds to the desired quality level. The test is premised on making a statistical decision to accept a null hypothesis, reject the null hypothesis or to continue the experiment by considering additional items from the production lot.

In the contest of change detection, the technique can be adapted to determine the change in distribution between a reference window $W_1 = 1..k$ and subsequently observed data $W_2 = k...N$. Assume that the data in W_1 is generated from a distribution P_0 and the data in W_2 is generated from another distribution P_1. Two hypotheses are set up with null hypothesis as that the probability distribution P of new data is P_0 and no change occurs and alternative hypothesis that the probability distribution of the new data is P_1 there is a change in distribution.

- $H_0 : P = P_0$ at w

- $H_1 : P \neq P_1$ at point w

The test statistic is computed as:

$$\vartheta_i = \log \frac{P(x_k|H_1)}{P(x_k|H_0)} \tag{2.12}$$

Two test parameters α and β controls the decision as follows:

(a) $if\ log\frac{\beta}{1-\alpha} < \sum_{i=k}^{t} \vartheta_i < log\frac{1-\beta}{\alpha}$ take another observation.

(b) $if\ \sum_{i=k}^{t} \vartheta_i \geq log\frac{1-\beta}{\alpha}$ reject H_0

(c) $if\ \sum_{i=k}^{t} \vartheta_i \leq loglog\frac{\beta}{1-\alpha}$ accept H_0

α and β are small constants often selected to be 0.05 and indicates the type 1 and type 2 errors that often characterise hypothesis testing. The main challenge of the test is the appropriate selection of the underlying distribution of the data and the parameters of the model.

Sequential CUSUM Test

Cumulative sum test is a sequential test that can be used to detect a small shift or change in the parameter that is being monitored to detect a significant change in its magnitude. The target value is often the expected apriori statistics of the data stream such as the mean of the data. In another clime, the input to the CUSUM is a residual form a predictors such Kalman filter. The residual represents the error between the observed and estimated value of a regression [84].

Algorithm 3: CUSUM Test

```
Input:   target value , $theta$, slack variable $nabla$, threshold limit $\pm\xi$
Data:  $S = \{x_i.....\}$
// Stream of data
Result:  $\tau$
// change indicator
1  $CumSum^-_{t-1} = 0$
2  $CumSum^+_{t-1} = 0$
3  foreach $x_i \in S$ do
4      $CumSum^+_t = \max(0, CumSum^+_{t-1} + x_i - \theta - \nabla)$
5      $CumSum^-_t = \min(0, CumSum^-_{t-1} + x_i - \theta + \nabla)$
6      $CumSum^+_{t-1} = CumSum^+_t$
7      $CumSum^-_{t-1} = CumSum^-_t$
8      if $CumSum^+_t \leq \xi$ $CumSum^-_t \leq -\xi$ then
9          $\tau = 1$
10     end
11     else
12         $\tau = 0$
13     end
14     return $\tau$
15 end
```

The test progresses by computing the cumulative sum of the deviations of the observed data and expected target parameter value. This cumulative sum is then checked at every time step to see if it has deviated significantly from zero within the allowable threshold value. The deviations is monitored in either positive or negative directions or both at the same time. In the case of positive deviation, maximum is used to update the cumulative sum while the minimum is used in the case of negative direction monitor. The CUSUM algorithm for detecting both positive and negative change directions is outlined in Algorithm 3. The CUSUM test procedure requires a threshold parameter to signal the change and the level of tolerance in the deviation between target and observed values. The sensitivity of the algorithm relies mostly on this threshold. In the realm of process control, the slack variable is usually set to be 1σ of the average of the data and the positive and negative threshold is often set to be $\pm 4\sigma$ of the average.

Exponential Weighted Moving Average (EWMA)

The exponential weighted moving average sequential method for change detection can be used to detect the time step at which a significant drift exists in the parameter of the data stream being monitored. The parameter could be the expected value of the mean of the data stream. When a series of samples x_i are observed, they are weighted and added to the previously observed data, and when the sum exceeds a threshold, change is declared in the underlying distribution of the data. The procedure starts by assigning an initial value to the cumulative weighted sum Z_{i-1}. This value is usually the target parameter that is being monitored. Afterwards, each successively observed samples is weighted and added to the previous value to obtain the new sum Z_i. This is defined as: $Z_i = x_i\lambda + (1-\lambda)$ where λ is the weighting factor and x_i is the currently observed sample.

The sensitivity of the EWMA to detect change depends on the value of λ and the threshold ξ. Setting them too high or too low may lead to missed detection. Therefore an optimum value is required to reduce false positive rate. In practise, it is common to set λ to 0.2 and the threshold is often set to $Z_0 \pm 3\sigma\sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda))^{2i}}$ for the positive and the negative threshold limits.

Page Hinkley Test

This is a sequential test for change point detection originally devised by Page Hinkley in 1954 for change detection in signal processing [69]. The approach is similar to CUSUM but rather than computing cumulative sum; it computes a test statistic; the cumulative difference between the observed values and their mean up till the moment of the test. It is defined as $C_T = \sum_{t=1}^{T}(x_t - \overline{x} - \theta)$, where $\overline{x} = \frac{1}{T}\sum_{1}^{T} x_t$ and θ is the accepted magnitude of tolerable changes. The minimum of C_T is defined as $c_t = min(C_t : t = 1...T)$. The two parameters are compared as $PH_{test} = c_t - C_T$. If the result is greater than a threshold ζ, a change is signalled.

2.6.2 Distributional Change Detection on Two Different Windows

In contrast to approaches that examine data in a sequential manner to detect a change, the approaches discussed here aim to detect the change by testing for distributional similarity and dissimilarity between two contiguous data sequences. The first data chunk represents a historic reference data and the second represents the current data. If the test yields a result in favour of a change in the distribution of the two data sequences, change is signalled at the starting point of the second data sequence. Otherwise, a new sequence of data are admitted, and the detection test continues until the end of the whole data stream. The two data sequences are often maintained in windows, and there are different strategies for managing these windows. For example, Kifer et al. [39] employed two equal and fixed size windows where they compare one reference non-sliding window of historical data with a sliding window of the same size containing the most recent data. Another approach presented by Dasu et al. [22] used window of adjacent and equal sizes, where the data in the two sliding windows are compared for distribution change. Bifet et al. [14] also proposed another approach that used all adjacent windows to look for the distribution change within a single accumulated window of moving samples. The size of the window is often selected based on system heuristics.

The detection method of distribution change can be parametric or non-parametric. The parametric approaches use a hypothesis test to determine the distributional similarity between two samples by using the parameters derived from the samples. The mean and variance extracted from the data samples are the commonly used parameters. Many theoretical statistical testing methods have been established for this task. The following are the prevalent methods in change detection literature.

Welch's t-test

This method is a parametric change detection method. It is an adaptation of the Student's t test by Welsh [102]. Given two samples n_1 and n_2 sampled from population N_1 and N_2, the test is used to statistically test the null hypothesis that the means of the population $\overline{N_1}$ and $\overline{N_2}$ with unequal sample variances ${s_1}^2$ and

${s_2}^2$ are equal. The null hypothesis can be rejected depending on the p-value given in Equation 2.13.

$$p - value = \frac{(\overline{N_1} - \overline{N_2})}{\sqrt{(\frac{{s_1}^2}{n_1}) + (\frac{{s_2}^2}{n_2})}} \tag{2.13}$$

Hotteling's t^2 Test

This is another parametric method. The test [35] is used to test the equivalence of the means of two distributions of multivariate data. The test computes a statistic to accept or reject the null hypothesis that the two data samples in windows W_1 and W_2 are drawn independently from two multivariate normal distributions with the same mean and covariance matrices. The t^2 statistic is computed according to the equation

$$T^2 = \frac{N_1 N_2 N_1 + N_2 - n - 1)}{n(N_1 + N_2 - 2)(N_1 + N_2)(\hat{\mu_1} - \hat{\mu_2})^T} \tag{2.14}$$

Kolmogorov-Smirnov's test

The Kolmogorov-Smirnov's test [18] is often used to determine if a set of samples follows a particular reference distribution or if two samples follow the same distribution or not. This test is non-parametric and is based on empirical cumulative distribution function (ECDF) computed from the test samples. To test if two samples follow the same distribution, the test computes ECDF for each ordered number of points N $x_1, x_2, ...x_N$ in each sample of size N_1 andN_2 according to Equation 2.15.

$$ECDF(i) = \frac{\gamma(i)}{N} \tag{2.15}$$

where $\gamma(i)$ is the number of points less than x_i; and the x_i are ordered from smallest to largest value. The Kolmogorov-Smirnov distance between the two test samples is computed as: $D = max_i(ECDF_1(x_i) - ECDF_2(x_i))$. The null hypothesis assuming that the two samples follow the same distribution is rejected with a confidence θ, if: $\sqrt{\frac{(N_1 N_2)}{(N_1+N_2)}}D > K_\theta$. The value of K can be found in the Kolmogorov-Smirnov table.

2.7 Change Detection in Labelled Data Stream

Change detection in labelled data stream is often regarded as concept change detection or concept drift detection [30]. In this domain of change detection, the data that is being monitored is labelled with ground truth for online learning. The direct manipulation of the attribute data is no more adequate to detect a change in the distribution of the data generating process which is conditioned on the data label. Hence an alternative approach to detecting change is being proffered.

The detection of a change in this domain can be done by monitoring different types of indicators [79]. Error rate, precision and recall are the commonly used performance indicators to detect concept change in the underlying online learning model [79]. A principal drawback here is the need for the availability of the ground truth of the predicted instances to ascertain the correct or incorrect nature of the predictions.

2.7.1 Types of Changes in Labelled Data Stream

Concept drift is a phenomenon in classification problem where a classifier built to recognise certain concept from the set of training data becomes inaccurate over time because the distribution of the data being classified has changed from the initial distribution known to the model [30].

The changes in the data can be viewed from three perspectives. In the first case, the change can occur in the class labels of a given set of related attributes for some data instances while the attributes themselves remain unchanged. That is, given a particular sample with set of attribute values and a class label say ‘0’, when changes occur the same sample now has label ‘1’. In the second case, the attribute values of the data may change while the class labels remain unchanged while in the third case of change, the two parts of the data can change simultaneously. In the first and third situations, the classifier will need to be updated with the new emerging distribution of the data while the second situation may or may not affect the decision boundary and hence may not require model update. Another possible but infrequent change, is the change in the prior probabilities of classes termed concept evolution that results in the emergence of new concepts or merging of existing concepts [61].

More formally, concept drift arises as a result of differences in the relationship between input variable x any target variable y between two points in time t_0 and t_1 : $P(x, y)_{t_0} \neq P(x, y)_{t_1}$ where $X \in R^n$ is input attributes and $y \in \{y_i : i = 1...c\ number\ of\ classes\}$. The changes in this relationship can manifest in the form of changes in the class conditional probability $P(x/y)$ where the attributes values changes for given y_i but the class label y remains unaffected. It may also result in posterior probability $p(y/x)$ changes which mean the attributes remain unchanged but the class labels change for a given attributes, or there could be simultaneous changes in posterior and class conditional probability. It is also possible to have prior probability changes leading to the emergence of new concepts.

Types of Drift	**Notation**	**Comments**
Real Drift	$p(y/x)_{t_0} \neq p(y/x))_{t_1}$	This drift affect the decision boundary.
Virtual Drift	$p(x/y))_{t_0} \neq p(x/y))_{t_1}$	Does not affect the decision boundary.
Virtual Drift with Decision Boundary Change	$p(x/y))_{t_0} \neq p(x/y))_{t_0}$ and $p(y/x)_{t_0} \neq p(y/x))_{t_1}$	Simultaneous drift in class conditional probability and posterior probability which affects the decision boundary.
Concept Evolution	$p(y)_{t_0} \neq p(y)_{t_1}$	Concept evolution results in emergence of new classes other than the existing known classes.

Table 2.3: Categories of Drift

Changes that arises from $p(y/x)$ is regarded as real concept drift, while $p(x/y)$ is referred to as virtual drift. The virtual drift can also occur when both $p(x/y)$ and $p(y/x)$ changes simultaneously. The changes in $p(y)$ is referred to concept evolution. Table 2.3 gives a summary of classes of drift in the literature. Three general approaches for change detection in labelled data stream are discussed below:

Sequential Change Detection in Labelled Data Stream

Change detection in label data stream often used the performance indicators of the online learning algorithms. To apply the sequential test to change detection in labelled data stream will require the input to the detector to come from the prediction errors of the underlying algorithm. Different authors have adapted the change detection in the unlabelled data stream to the case of labelled data stream by using the error rate indicators as input rather than the actual raw data from the stream. Gama et.al. [29] have used the Page-Hinkley test for change detection to show the efficiency of prequential evaluation technique on the data stream. Similarly, [83] evaluated Page-Hinkley [69] test among other change detection methods by using the error rate obtained from online learning model built from the non-stationary stream. The basis for using error rate is that in the absence change in the underlying distribution of the data the error rate should be stable or decreases and when there is a change in distribution the error rate will increase. By using Page Hinkley test, they monitor the deviation between the currently observed error rate and the accumulated mean of the error rate up until the moment of the test. If this value exceeds the threshold, change is declared, and the model is reset.

Method Based on Statistical Control

This method considers learning as a process and applies the principle of process control chart to concept change detection by defining upper and lower control limits for the variable of the model being monitored. In this case the error rate. When the error rate of the online model derived from the stream is within the limit of the lower and upper control limit the model is said to be in control when the limit is exceeded, the current model is declared as out of control and change is declared. The drift detection method [28, 40] are some of the approaches that use the idea of process control chart for change detection in the labelled data stream. The method monitors the performance evolution of a classifier and relies on the availability of ground truth to determine when the classifier gives a correct or incorrect prediction. The method incrementally compute the proportion of errors produced by the current model with $p_i = p_{i-1} + (x - p_{i-1})/n$ with $x = 1$ if the prediction is incorrect and $x = 0$ if the prediction is correct. The average error is thus computed incrementally. The standard deviation of the error rate s_i at each time step of the learning process is also computed. Two registers p_{min} and

s_{min} are maintained and they are updated with p_i and s_i respectively whenever $p_i + s_i < p_{min} + s_{min}$. DDM has two thresholds to take a decision on the drift: if $p_i + s_i \geq p_{min} + 2 * s_{min}$ it implies a warning level. Subsequent examples after this point are stored in anticipation of a possible change of context. The examples are used to retrain the model. If $p_i + s_i \geq p_{min} + 3 * s_{min}$ a drift level is signalled after a series of warning state concept drift is declared, the model induced by the learning method is reset and a new model is learnt using the examples stored since the warning level was triggered. The values for p_{min} and s_{min} are reset to 0.

The intuition behind this approach is that in the absence of concept drift the error rate should decrease indicating a stationary distribution. However, if the error rate decreases significantly, it means the classifier is no more in tandem with the distribution of the data. Thus a concept drift has occurred and the model has to be rebuilt. Authors in [10] extends DDM to account for the distance between error point while [16] used DDM as a component for their adaptation algorithms to make them informed.

Distributional Change Detection in Labelled Data Stream

Approaches based on monitoring the changes in the distribution between two windows can be used to track the performance indicators of the online learners as labelled samples prediction errors are evaluated. The techniques in this category employ a statistical testing technique to determine change point between the reference window and a detection window. Apart from this, some techniques employ geometrical similarity measures between two windows and determine whether the threshold limit is exceeded to signal a change.

A notable approach for concept change detection is the adaptive windowing technique [14]. In ADWIN, a window W containing the error rate from the predictions of a learning model grows until there has been a change in the average value inside the window. Therefore, when the algorithm succeeds at finding two distinct subwindows, the split point can be considered as a concept change. Also, Widmer and Kubat [104] proposed change detection approach that analyses the misclassification error rates and the changes that occur in the structures of the learning algorithm (adding new definitions in the rules in this case). When there is variation of these indicators, the size of the training window is reduced. Otherwise, the window grows with more examples to achieve better learning.

2.8 Conclusion from the Literature

Activity recognition from wearable sensors involves use of sensors data collected from reference users to train a machine learning classification algorithm that gives a model that recognises the activity of large population of users. This process has been seen in the literature to suffer from inadequacy of accurate recognition for large user populations with differing characteristics [44, 48, 49, 62, 97, 100].

Various approaches have been proposed in the literature to cater for the problem of degrading accuracy of classification model as a result of varying characteristic and differences in user populations. The approaches are generally termed personalization and adaptation of activity recognition model. However, there are deficiencies in the process of invoking the adaptation process. The approaches do not identify the actual state at which the model degrades. They either assume that the model should be adapted for each of the new users or continuously adapt the model at regular intervals. This definitely indicates an unfounded adaptation of the model that either update or replace the model unnecessarily. They may also miss the best time to update the model.

Therefore, an holistic solution will be to detect the state at which the model performance degrades before updating the model. However, existing approaches in the literature show that the unlabelled change detection are mainly used for detecting changes in the unlabelled data distribution parameters such as mean and variance. They have not been used directly for detecting changes in labelled data. On the other hand, detecting changes in labelled data relies on using ground truth to detect changes in error rate of the model. Thus, it is clear that a method that does not rely on ground truth in order to detect changes in labelled data during classification is desired for activity recognition because it is impractical to ask the user to be providing ground truth at every point of activity prediction by the classifier. Therefore, the research question of this thesis is centred on how to detect change in activity recognition model without having access to the ground truth. The thesis answer this question by proffering alternative novel solutions to detect classifier error without access to ground truth labels during classification of new sample of instances.

2.9 Chapter Summary

This chapter has reviewed the background literatures on activity recognition process and the various sensor modalities being used in activity recognition system. Also, a review of various approaches being used for model adaptation in activity recognition are discussed and their drawbacks are identified. The lack of informed decision before adaptation is a key problem that this thesis aim to address. This chapter has also discussed various approaches for change detection in labelled and unlabelled data. It is observed that change detection in unlabelled data focuses on determining change in the parameters that characterise a given data samples. Such parameters include the mean and variance of the data distribution. These approach cannot be directly applied in labelled data stream because the data samples are condition on their labels. Also, current approaches for change detection in labelled data rely on label availability to monitor the error rate of the model generated from the data. This approach is not realistic for activity recognition because it will be tedious for the user to be providing label for each instance of the data during recognition.

Chapter 3

Benchmark Algorithms and Datasets

3.1 Introduction

This chapter presents background details about the machine learning algorithms and datasets used for the evaluation of the proposed and implemented learning algorithm and the change detection techniques. Also, the chapter discusses the evaluation metrics employed in the research.

3.2 Benchmark Supervised Machine Learning Algorithms

Three machine learning algorithms were used as benchmark algorithms to compare with the new classification method that is developed in this research. The benchmark algorithms are: Naive Bayes, Decision Tree, and Instance-Based Classifier also called K- Nearest-Neighbour. These three algorithms are some of the most commonly used algorithms for state of the art activity recognition system [51, 85].

- Decision trees utilised the concept of if-then rules to build a hierarchical model in the form of trees. The tree nodes represent the attributes in the sample data and the edges of the tree represent the possible attribute values. Each branch from the root to the leaf node is a classification rule. C4.5 is the most widely used implementation of a decision tree. It utilised the concept of information gain to determine which attributes should be placed in the top nodes [75] and recursively builds the tree until all attributes are selected, and leaf nodes are reached. A decision tree is easy to understand by humans but suffers from difficulties in obtaining a single optimal tree. Decision Tree algorithm has been found to perform well on activity recognition tasks that employed the wearable and mobile devices sensors [12, 38, 63]

- Naive Bayes is another widely used learning algorithm for activity recognition. It is based on the Baye's probability theorem [17]. To determine the class c_i of an unseen data $x^i \in R^n$ with attributes $a_1, a_2, ..., a_n$ and corresponding values $v_1, v_2, ..., v_n$, the algorithm first computes the prior probabilities of each class $P(c_1), P(c_2), ..., P(c_k)$ from the training data. The conditional probability of the observed data given the classes $c_i \in K$ is also computed with:

$$P(a_1 = v_1|c_i) \times P(a_2 = v_2|c_i) \times ... \times P(a_n = v_n|c_i)) \; \forall c \in K \; numberofclasses$$

Finally the posterior probability of the most probable class c, out of all possible classes K to be assigned to the instance x^i is obtained from:

$$c = argmax_{c_i \in K} \frac{P(c_i)P(a_1 = v_1|c_i) \times P(a_2 = v_2|c_i) \times ... \times P(a_n = v_n|c_i)}{\sum_{i=1}^{K} P(c_i)P(a_1 = v_1|c_i) \times P(a_2 = v_2|c_i) \times ... \times P(a_n = v_n|c_i)} \tag{3.1}$$

The denominator is called the marginal probability or the normalizer. Naive Bayes algorithm is easier to use with categorical attributes. If the attributes are continuous as in the case of most sensor data, a pre-processing step is required to discretized the data. Naive Bayes and the related Bayesian algorithm have been used in activity recognition system including [94] [50].

- K-nearest neighbour is an instance based machine learning classification algorithm [105]. It works by comparing unseen instance with all instances in the data set using a distance function. K nearest training data to the unseen instance is selected and the class labels of the majority of the selected k points are used as the class of the unseen instance. KNN was employed in [53, 74] for activity recognition.

3.3 Justification for the Selected Benchmark Algorithms

The algorithms selected are part of the most widely used in recognition of low-level activity. These algorithms have been used in other recent work for recognition of low level activities [81, 85, 88]. In [88] kNN, decision tree and Naive Bayes classifiers were employed in the evaluation of data collected for different low-level and high level activities. There results indicate that the classifiers are adequate in recognizing the activities accurately under various sensor fusions and placement scenario. Lara and Labrador [51] have stated that these classifiers are among the state of the art algorithms for activity recognition. Another important justification for selecting these classifiers is because of their real time practical usage on mobile phone for recognizing day-to-day low level activities [86, 87]. Various studies have employed these classifiers on mobile phones. For example, Naive Bayes classifier was implemented for mobile activity recognition in [21, 33, 46, 85], decision tree was employed in [27, 58, 64] while kNN was the selected algorithm in [90, 95]. All these point to the effectiveness of the selected algorithms for activity recognition. Also, using more complex classifiers on mobile phones will result in higher resource consumption which will eventually affects the user experience in using the mobile activity recognition application [101].

3.4 Activity Recognition Datasets

The three publicly available real activity recognition datasets used in this thesis are described in this section. The description include the source of the datasets and the distribution of instances across all subjects.

3.4.1 WISDM Dataset

The Wireless Sensor Data Mining (WISDM) activity recognition dataset [44] was obtained from the accelerometer of mobile phones. The researcher collected the data from 36 subjects that performed six designated activities of working, jogging, ascending and descending stairs, sitting and standing. Forty-three features represent each data sample in the dataset. Each 43 features of each instance data were computed from the transformation of 200 raw samples from the tri-axial accelerometer of a mobile phone. The 200 raw accelerometer samples were recorded within a 10-seconds window with a sampling frequency of $20Hz$. The features used were basic statistical features including standard deviation, average, resultants among others [44] described as follows:

- $X0..X9, Y0..Y9, Z0..Z9$ are set of bins of values representing a fraction of accelerometer samples that fell within that bin.
- $XAVG, YAVG, ZAVG$ these features account for the average of the x, y, and z values in each recorded 200 samples.
- $XPEAK$,$YPEAK$,$ZPEAK$ these features approximate the dominant frequency along the x, y, and z-axis values of the accelerometer within each 200 samples point.
- $XABSOLDEV, YABSOLDEV, ZABSOLDEV$ are the average absolute deviations from the mean value for each axis.
- $XSTANDDEV, YSTANDDEV, ZSTANDDEV$ are the standard deviations for each axis.
- $RESULTANT$ is the average of the square roots of the sum of the values of each axis squared$\sqrt{(x_i^2 + y_i^2 + z_i^2)}$

The dataset distribution spread across the six activities. The total samples in the obtained dataset and their distribution across each activity and users are shown in Table 3.1.

User No.	**Walking**	**Jogging**	**Walking Upstairs**	**Walking Down-stairs**	**Sitting**	**Standing**
1	42.11	36.18	11.18	10.53	0.00	0.00
2	50.00	50.00	0.00	0.00	0.00	0.00
3	36.31	30.73	10.61	10.06	4.47	7.82
4	50.00	8.33	13.33	16.67	11.67	0.00
5	41.78	21.92	12.33	12.33	6.16	5.48
6	39.87	35.95	8.50	7.19	5.88	2.61
7	35.03	29.30	12.10	7.64	8.28	7.64
8	50.60	30.95	0.00	0.00	8.33	10.12
9	100.00	0.00	0.00	0.00	0.00	0.00
10	36.52	34.27	13.48	11.24	0.00	4.49
11	37.27	38.51	14.91	9.32	0.00	0.00
12	32.14	36.90	8.93	9.52	7.14	5.36
13	35.52	31.15	13.66	12.02	3.28	4.37
14	36.13	34.55	21.47	7.85	0.00	0.00
15	41.91	39.71	8.82	9.56	0.00	0.00
16	60.78	0.00	6.86	7.84	14.71	9.80
17	42.48	13.27	26.55	17.70	0.00	0.00
18	39.75	34.16	8.07	7.45	4.97	5.59
19	37.99	35.37	10.04	6.99	4.80	4.80
20	23.81	19.78	9.16	9.16	28.21	9.89
21	26.23	15.57	21.31	18.03	6.56	12.30
22	30.97	27.43	24.78	16.81	0.00	0.00
23	25.58	47.29	19.38	7.75	0.00	0.00
24	23.66	47.33	12.21	11.45	3.05	2.29
25	51.52	48.48	0.00	0.00	0.00	0.00
26	40.00	35.76	12.12	12.12	0.00	0.00
27	36.47	31.76	9.41	11.18	6.47	4.71
28	65.42	0.00	14.02	14.02	0.00	6.54
29	32.63	30.53	13.16	12.11	6.84	4.74
30	48.06	0.00	17.83	16.28	6.20	11.63
31	37.67	31.39	10.76	9.42	4.93	5.83
32	36.26	30.41	11.70	7.02	9.36	5.26
33	49.01	9.93	8.61	16.56	11.26	4.64
34	36.61	34.97	11.48	8.74	4.37	3.83
35	31.86	55.75	0.00	0.00	7.08	5.31
36	19.02	36.20	16.56	14.11	7.98	6.13
Total	**38.41**	**29.99**	**11.66**	**9.75**	**5.65**	**4.54**

Table 3.1: Distribution of WISDM Dataset in (%)

3.4.2 Human Activity Recognition Using Smartphone Dataset-HARS

The Human Activity Recognition Using Smartphone (HARS) Dataset [8] was collected from 30 different volunteers with their ages ranging between 19 to 48 years. Each subject performed six designated activities of walking, walking-upstairs, walking-downstairs, sitting, standing, and laying while wearing a smartphone attached to their waists. The sensor data were obtained from gyroscope and accelerometer sensors of the smartphone. Each instance (feature vector and label) in the dataset is represented by 561 features containing both time domain and frequency domain features and a corresponding activity label. The feature vectors was computed from 128 fixed-length sliding windows of raw accelerometer data within 2.56sec and 50% overlap. The released dataset is partitioned into two representing 70% and 30% of training set and testing set respectively. The dataset distribution spread across the six activities. The total samples in the obtained dataset and their distribution across each activity is shown in Table 3.2.

User No.	**Walking**	**Walking Upstairs**	**Walking Down-stairs**	**Sitting**	**Standing**	**Laying**
1	27.38	15.27	14.12	13.54	15.27	14.41
2	19.54	15.89	15.56	15.23	17.88	15.89
3	17.01	17.30	14.37	15.25	17.89	18.18
4	18.93	16.40	14.20	15.77	17.67	17.03
5	18.54	15.56	15.56	14.57	18.54	17.22
6	17.54	15.69	14.77	16.92	17.54	17.54
7	18.51	16.56	15.26	15.58	17.21	16.88
8	17.08	14.59	13.52	16.37	19.22	19.22
9	18.06	17.01	14.58	17.36	15.63	17.36
10	18.03	15.99	12.93	18.37	14.97	19.73
11	18.67	17.09	14.56	16.77	14.87	18.04
12	15.63	16.25	14.38	15.94	19.06	18.75
13	17.43	16.82	14.37	14.98	17.43	18.96
14	18.27	16.72	13.93	16.72	18.58	15.79
15	16.46	14.63	12.80	17.99	16.16	21.95
16	13.93	13.93	12.84	18.85	21.31	19.13
17	16.58	13.04	12.50	17.39	21.20	19.29
18	15.38	15.93	15.11	15.66	20.05	17.86
19	14.44	11.11	10.83	20.28	20.28	23.06
20	14.41	14.41	12.71	18.64	20.62	19.21
21	12.75	11.52	11.03	20.83	21.81	22.06
22	14.33	13.08	11.21	19.31	19.63	22.43
23	15.86	13.71	14.52	18.28	18.28	19.35
24	15.22	15.49	14.44	17.85	18.11	18.90
25	15.05	14.03	12.76	19.90	18.88	19.39
26	15.05	14.03	12.76	19.90	18.88	19.39
27	15.16	13.56	11.70	18.62	21.28	19.68
28	14.14	13.35	12.04	18.85	20.68	20.94
29	15.41	14.24	13.95	17.44	18.90	20.06
30	16.97	16.97	16.19	16.19	15.40	18.28
Total	**16.60**	**14.92**	**13.60**	**17.41**	**18.54**	**18.94**

Table 3.2: Distribution of HARS Dataset in (%)

3.4.3 Opportunity Dataset Description

The Opportunity dataset for human activity recognition from wearable object and ambient sensors [80] also generally referred to as the Opportunity dataset was obtained from the UCI repository. The dataset is devised to serve as the basis for benchmarking algorithms for classification, automatic data segmentation sensor fusion and feature extraction tasks in the context of human activity recognition problems. The collection and release of the dataset to public domain was motivated by the lack of standard benchmark datasets in the activity recognition community for comparing and validating results among activity recognition research community. It is common for each group to use their own in-house collected dataset to validate their algorithms and methods. The dataset was collected in the context of a smart home setting whereby a studio flat simulates a sensor-rich environment with sensors attached to the subjects in the apartment, objects used such as fridge, spoons and plates. Each subject performed five sessions of scripted activities ranging from locomotion activity (sit, lie and walk) hand gesture activity (hold cup, open door and open fridge). The first four sessions of the data were performed by the user in a free form manner without any specific sequence for the user to follow in performing the ADLs. While the last session called, the drill session was carried out by following a prescribed sequence of activities. All the sensor reading were synchronised and recorded in column matrix while the activity label which was annotated by expert were appended to the last columns. The data used in this experiment, however, excludes some sensor channels and activity labels. Only the sensor data from the accelerometer and inertial measurement units (IMU) attached to the subjects as they perform activities in the flat were used. The quaternion channels data from the IMU sensor recordings were also excluded.

Subject ADL Sessions	**Null**	**Standing**	**Walking**	**Lying**	**Sitting**
S1ADL1	26.62	43.78	12.80	14.59	2.21
S1ADL2	23.94	31.93	20.21	19.59	4.33
S1ADL3	23.95	31.19	18.12	22.85	3.89
S1ADL4	24.59	31.83	19.48	21.06	3.03
S1ADL5	25.52	30.71	17.41	22.33	4.04
S1Drill	5.21	41.11	43.62	10.07	0.00
S2ADL1	18.39	34.01	21.16	23.45	2.99
S2ADL2	23.44	30.40	19.35	24.09	2.72
S2ADL3	22.12	33.40	18.45	23.35	2.67
S2ADL4	22.08	27.17	22.58	25.91	2.26
S2ADL5	27.83	28.11	19.80	20.62	3.64
S2Drill	11.49	49.61	32.20	6.70	0.00
S3ADL1	13.17	39.05	25.72	15.22	6.83
S3ADL2	10.50	41.10	29.71	13.58	5.10
S3ADL3	20.18	39.57	16.98	17.60	5.67
S3ADL4	9.23	32.82	32.59	17.56	7.80
S3ADL5	22.07	39.08	16.91	16.28	5.66
S3Drill	3.50	67.91	21.15	7.45	0.00
S4ADL1	30.50	37.19	15.72	12.74	3.85
S4ADL2	29.05	35.03	18.13	14.63	3.16
S4ADL3	29.89	33.68	15.41	16.59	4.42
S4ADL4	32.25	31.04	21.88	11.28	3.54
S4ADL5	21.98	34.08	29.97	10.91	3.05
S4Drill	7.95	55.90	26.51	9.65	0.00
Total	**18.69**	**39.73**	**22.91**	**15.74**	**2.92**

TABLE 3.3: Distribution of Opportunity Dataset in (%)

Change Detection Performance Metrics

- True Positive Detection T_{p_d}: These are the set of instances which are truly misclassified and the detector detects them as misclassified.
- True Negative Detection T_{n_d}: These are the number of instances which the model classified correctly and the detector detects them as not misclassified.
- False Positive Detections F_{p_d}: These represent set of instances which the model classified correctly, but the detector detects them as being misclassified.
- False Negative Detection F_{n_d}: This refers to set of instances which the classification model misclassified but the detector detect them as not being misclassified.
- Overall Detection Accuracy: This is the percentage of correctly detected samples which is denoted by the sum of true positive detection and true

negative detection divided by the total number of samples.

$$\frac{T_{p_d} + T_{n_d}}{Total\ Samples}$$

- Recall: This the proportion of correctly detected unseen instances to the total number of both correctly detected and falsely detected unseen instances. This is given as:
$$\frac{T_{p_d}}{T_{p_d} + F_{n_d}}$$
- Precision: This the proportion of correctly detected unseen instances to the total number of both correctly detected and falsely detected unseen instances. This is given as:
$$\frac{T_{p_d}}{T_{p_d} + F_{p_d}}$$
- Specificity: This the proportion of correctly detected known instances to the total number of both correctly detected and falsely detected known instances. This is given as:
$$\frac{T_{n_d}}{F_{p_d} + T_{n_d}}$$
- Detection Error Rate: This connotes the error rate in the detection scheme given as:
$$\frac{F_{p_d} + F_{n_d}}{TotalSamples}$$

3.5 Chapter Summary

This chapter presents an overview of the methodologies adopted in this thesis. It discussed the benchmark supervised machine learning algorithms: Naive Bayes, Decision Trees and kNN that were used for comparing the new ClusterNN algorithm. It also provides the justifications for selecting these algorithms. The major reason for selecting the algorithms include their less computation resource overhead which make them suitable for resource constraint environments such as mobile phones. Also, this chapter provides details about the three publicly available activity recognition datasets namely the Opportunity, the Wireless Sensor Data

Mining (WISDM) dataset and the Human Activity Recognition Using Smartphone (HARS) that were employed for the myriad set of experiments performed throughout the thesis The description of the features used to generate the data and the distributions of the instances in the datasets are discussed. Finally, the chapter presents the metrics used for the evaluation of the classification method and the change detection methods developed in the thesis.

Chapter 4

ClusterNN: A Clustering and Nearest Neighbour Based Classifier for Activity Recognition

4.1 Introduction

This chapter presents a classification approach called ClusterNN. The algorithm builds on the concept of nearest neighbour and clustering to develop an hybridised algorithm that utilised less amount of training data. This is desirable for using instance based learning in a resource constraint environment such as mobile phones. The algorithm evaluation and results are presented along with the performance comparison with the benchmark algorithms described in Chapter 3.

The rest of this chapter is organised as follows: Section 4.2 highlights the process of formation of the reduced data from the proposed statistical data summaries. Section 4.3presents the online classification stage of the ClusterNN algorithm while Section 4.4 presents the experimental evaluation of the approach. The chapter is summarised in the last Section 4.5.

4.2 Formation of the Reduced Data

One of the key reasons for proposing this algorithm is to develop a change detection that can support adaptive incremental learning strategy for activity recognition. In this regard, an algorithm that is capable of being updated during online classification is needed. The proposed algorithm is well suited for this purpose. The framework termed 'ClusterNN' is based on the application of clustering and nearest neighbour hybrid technique to create a holistic representation of the original dataset into a compact and more discerning set.

The framework has two phases of operation: the offline and online phases. The off-line phase involves extensive experimentation to determine the amount of data to be retained in the dataset to ensure more accurate classification with nearest neighbour approach. The main essence of this stage is to reduce the original dataset as much as possible to yield a more compact and informative reduced set that is suitable for on-line recognition. To achieve this, clustering is applied on the original dataset with the introduction of the notion of percentage data reduction to guide the clustering routine on the number of clusters to create in each class present in the dataset. The sub-clusters produced in each class of activity represent the various patterns that are present within a given class. This will invariably enable the system to leverage on the numerous patterns rather than a single centroid model per class to classify new samples. After this, three statistical summaries: minimum, maximum and centroid are extracted from each sub-cluster. Each of these statistical summaries can be used individually as referenced reduced dataset or their predictions can be combined into an ensemble predictions depending the level of accuracy desired and system resources available. The process is illustrated in Figure 4.1.

The three statistical summaries represent set of discerning data characteristics that summarise the original dataset in a reduced form by reducing the size of the instances. This method does not reduce along the dimension of the features representing the data. The main purpose of the statistical summaries is to reduce the data samples and produce more compact and reduced data that posses more

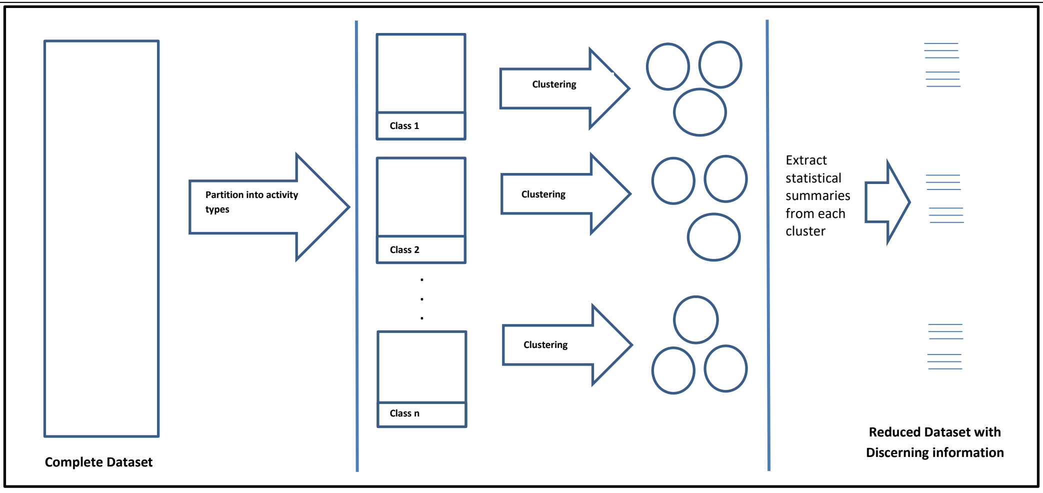

Complete Dataset
Partition into activity types
Class 1
Class 2
Class n
Clustering
Clustering
Clustering
Extract statistical summaries from each cluster
Reduced Dataset with Discerning information

FIGURE 4.1: Statistical Summaries Extraction Process

discerning information than the original data set. The proposed statistical summaries do not have any limitation on the type of features they can summarise. Regardless of the number features and amount of data in the underlying dataset, the approach can extract the desired proportion of data from them. This is made possible because the clustering technique that clusters the base data into different patterns is directed by the proportion of desired data to retain. The three proposed statistical summaries and the data obtained are described as follows:

- *Centroid or Mean*: The is obtained from a cluster of data by computing the average of the set of *d-dimensional* data points that are present in a cluster as $\vec{\mu_l} = \frac{1}{|C_i|} \sum \vec{x}_i$ where C_i is the amount of samples belonging to cluster i. The set of centroid vectors obtained from different clusters created from each class of activity data represents a reduced and new representative set from the centroid perspective.
- Maximum characteristic from a cluster is obtained as a d-dimensional vector in R^d with each component containing the maximum value in each dimension of the set of $d-dimensional$ instance vectors in a cluster. This implies that each cluster produces a maximum vector. The collection of maximum vectors obtained from n sub-clusters of a given class in a dataset represents another representation or view of the original dataset. The intuition of using the maximum characteristic to represent a micro-cluster is that its data represents the upper limit of the points in that cluster. Thus, it serves as the upper

boundary of the data in the cluster. Also, the set of maximum vectors across each type of activity can be used to differentiate high intense activity with high values from low intense activity with low values.

- Minimum characteristic is a vector in R^d similar to maximum characteristic but each of its components contains the minimum value in each dimension of the instance vectors in a cluster. This also represents another view of the original dataset. The intuition of using this in contrast to the maximum characteristic is that it represents the lower limit of the points in a cluster. It serves as the lower boundary of the data in the cluster. Also, the set of minimum vectors across each type of activity can be harnessed to differentiate low intense activity with low values from high intense activity with high values.

The procedure for extracting the data for these statistical summary is shown in Algorithm 4. The algorithm takes the training data and the desired percentage of data to retain as input and produces the Model Data (MD).

The model data (MD) is the set of data resulting from the centroid, minimum and maximum statistical summaries obtained after applying clustering on the dataset. In this algorithm, data samples belonging to each class$_i$ are clustered (lines 1-3 Algorithm 1) by applying a clustering technique on the data. Possible clustering algorithms include k-Means, DBScan, EM and host of others [107]. However, bisecting k-means is used for clustering the dataset in each class. After the clustering step, the list of cluster centres obtained for the class of data is stored in a list.

Also, the minimum vectors and maximum vectors are extracted from each sub-cluster returned for the current class (Algorithm 4 lines 4-8). The number of clusters created per class is proportional to the number of samples in the class and the desired percentage of data retention. This step is repeated for each class in the data. Finally, the set of cluster characteristics i.e. centroids, minimum and maximum obtained from the different clusters of each class and their associated labels are returned from the algorithm. These represent the Model Data (MD) to be deployed for the online recognition. The key feature of the model is that it is more compact and has a reduced resource overhead in terms of memory requirement and time when compared to the ordinary KNN. In addition, the reduced compact set including centroids can be adapted to evolving activity data as new unanticipated changes occur in the input data distribution.

Algorithm 4: Offline Data Characteristic Formation

```
Input: $C_n$ number of classes in the dataset
$K_n$ // percentage of data to retain in each class of examples that serves as cluster
    centroids.
Data: $D = (x^i, y^i)$ $x^i \in R^n$ and $y^i \in R^1$ // set of training examples
Result: MD // Data Summaries
1  foreach $class_k$ in $C_n$ do
2      $data_{classi}$=getData(D, $class_k$)
3      centroidsList[k], clusterAssign= Clustering( $data_{classi}$, $K_n$)
4      centroidList.append(centroidsList[k]) foreach $cluster_k$ in $len(centroidsList_k)$ do
5          $pointsInCluster_k$ = getData(clusterAssign, $cluster_k$)
6          maximumList.append(max($pointsInCluster_k$))
7          minimumList.append(min($pointsInCluster_k$))
8      end
9      MD = [centroidsList, minimumList, maximumList,$class_k$]
10 end
11 return MD
```

4.3 ClusterNN Online Classification

During the online phase, new instances are classified by passing them and the MD to Nearest-Neighbour routine. It employs Euclidean distance to compute the K-nearest neighbour to the new instance and assigns the majority label of the K nearest point to it (Algorithm 5). Since there are more than one cluster characteristics in the MD, each is considered separately, and a majority voting is performed on the outcome of each comparison. The final class given to the new instance is the majority label returned by all of them.

4.4 Experiments

In this section the experiments conducted to evaluate the applicability and accuracy of the proposed algorithm are presented. The datasets: WISDM, HARS and Opportunity were used to perform the experiments.

Algorithm 5: Online Classification

```
Input: $x_{new}$ , $k$
// new unlabelled instance, number of nearest neighbours
Data: $MD$
// Reduced training set obtained from statisical summaries
Result: $y_{new}$
// predicted class
1 foreach $clusterCharacteristics_i$ in $MD$ do
2     $prediction_i$ =nearestNeighbour($clusterCharacteristics_i$, $x_{new}$, $k$ )
3 end
4 $y_{new} = argmax_c\ (prediction_c) \quad c = (1...C)$
5 return $y_{new}$
```

The experiments was performed in two phases. In the first phase, the accuracy of using individual cluster characteristic was examined. As there are three statistical summaries that are used for classification decision during the online phase (Algorithm 5). The centroid characteristic is the mean of the data points in a cluster. Minimum characteristic characteristic is the minimum values across each feature for the data points in a cluster while the maximum characteristic is the minimum values across each feature for the data points in a cluster. The percentage of data retained was varied between 10% to 90% retention rate. The 100% retention rate was not tested because this will be equivalent to having all the dataset present. Therefore, the accuracy of using each characteristic was recorded as the number of data retention was varied. In the second phase of the experiment, all the characteristics were combined to predict the classes of unseen instances used for testing the algorithm.

In carrying out the experiment, the hold-out evaluation strategy was followed. The entire dataset was divided into the training set and test set. The split of the data was proportional to the number of instances in each class for the training set and the testing set. Each dataset is divided into 70% for training and 30% for testing set. The same configuration is used in evaluating ClusterNN and the benchmark algorithms.

4.4.1 ClusterNN Performance on WISDM Dataset

The accuracy of the proposed classifier for activity recognition is presented here. The results of using the first dataset are presented here. The results of using WISDM [44] dataset for evaluation are shown in Tables 4.1, 4.3 and 4.2.

The tables show the results for the three different reduced data samples (centroid, minimum and maximum) employed individually by computing the nearest similarity between them and new instances of the test data. As indicated in Table 4.1, centroid data give its best accuracy in classifying test instances when nearest neighbour is set to 1 and the percentage of data retained is set to 50% of the original dataset. The accuracy of using minimum data for classification decision give its best accuracy when K=1 and data retained is 80% while maximum data characteristic gives its best accuracy when k=2 and data retained is either 80% or 90% of the original training dataset.

However, when the predictions from all the three representative data were combined and the majority voting scheme was used to select the final class of an instance, the best accuracy is obtained when the number of nearest neighbour $K = 2$ and 80% data retained. These results indicate that there is a trade-off between accuracy and the amount of data retained for classification across each of the three data characteristics and their ensemble predictions. Thus, one can select the percentage of data reduction based on the level of the desired accuracy. For this dataset, 50% data retention can be adopted as the optimal data retention given that the overall best accuracy of 81.46% is achieved with centroid characteristic at this point. More so, going beyond this percentage of data reduction does not yield any high significant increase in accuracy across each of the training data representation and their combination. Therefore, going beyond 50% data retention is ineffectual considering that the corresponding accuracy improvement is not very significant. It is also observed that the best accuracy was obtained for the WISDM dataset [44] within the range of nearest neighbours set between 1 and 5.

Table 4.5 shows the comparison of the accuracy of KNN which utilised all the dataset and the three statistical summary with their best accuracies obtained at

Centroid Data									
K	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
1	**76.85**	**79.80**	**80.07**	**80.54**	**81.46**	**80.81**	**80.90**	80.63	81.00
2	76.01	77.77	78.78	79.80	79.70	80.26	80.63	**81.09**	80.81
3	75.83	78.32	78.78	78.32	79.43	79.24	80.17	79.70	78.60
4	76.20	77.31	77.95	78.60	79.24	78.14	79.34	79.61	79.61
5	75.37	76.57	77.58	78.04	78.51	78.14	79.34	79.34	79.43

Table 4.1: Accuracy of Using Centroid Data with Varying Percentage of Data Retained on WISDM Dataset

Maximum Data									
K	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
1	62.55	65.31	**73.15**	**75.28**	77.31	78.87	79.15	79.34	79.98
2	62.92	65.22	72.05	75.00	**77.40**	**79.06**	**80.07**	**80.44**	**80.44**
3	**64.11**	66.42	72.05	73.80	76.11	77.40	77.77	79.34	78.78
4	63.10	65.87	71.77	74.54	77.03	77.95	79.52	78.60	79.43
5	63.10	**66.61**	71.31	73.71	76.38	76.75	79.06	78.51	79.43

Table 4.2: Accuracy of Using Maximum Data with Varying Percentage of Data Retained on WISDM Dataset

Minimum Data									
K	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
1	70.48	71.59	75.09	78.04	**77.95**	**79.89**	**79.98**	**81.73**	80.44
2	69.10	71.49	75.37	77.03	77.58	79.15	79.52	80.72	**80.90**
3	69.37	73.43	75.83	76.57	77.68	78.97	78.78	79.80	79.61
4	69.83	73.62	75.55	77.31	77.86	78.23	79.61	80.72	80.35
5	69.46	72.42	75.00	76.94	77.49	78.14	78.41	79.52	79.98

Table 4.3: Accuracy of Using Minimum Data with Varying Percentage of Data Retained on WISDM Dataset

Ensemble Prediction									
K	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
1	**73.71**	**76.85**	**78.32**	**79.61**	**79.80**	**81.18**	**80.81**	81.09	80.35
2	69.74	72.79	76.01	78.14	78.23	79.34	80.07	**81.18**	**81.00**
3	70.20	73.80	75.74	76.66	77.58	78.14	79.52	80.26	79.24
4	70.39	73.43	75.55	76.66	77.68	77.58	79.98	80.26	79.89
5	69.46	72.88	75.18	76.38	77.58	77.49	79.06	79.06	79.80

Table 4.4: Accuracy of Using Ensemble Prediction with Varying Percentage of Data Retained on WISDM Dataset

the corresponding data reduction level. It is seen that the accuracy of basic KNN is lower than the centroid, minimum and the ensemble prediction accuracies. As indicated further in Figure 4.2, the accuracy of using centroid data for prediction is the best when K=1 and the percentage of data reduction is 50%. This is followed closely by the ensemble prediction which utilises the combined prediction of the three statistical summary data. Although, the ensemble prediction is not suitable for this dataset because it will require the retention of 60% each for the three data representations. However, the minimum data representation which utilised 80% of the data is also better than that of the KNN that used all the data samples in classifying a new instance. The only data representation that gives a less accuracy compared to KNN is the maximum data and this margin of difference is very low.

KNN	**Centroid-50%**	**Maximum-80%**	**Minimum-80%**	**Ensemble Prediction-60%**
80.90	**81.46**	79.34	**81.73**	**81.18**
80.99	79.70	**80.44**	80.72	79.34
79.89	79.43	79.34	79.80	78.14
80.07	79.24	78.60	80.72	77.58
79.98	78.51	78.51	79.52	77.49

Table 4.5: Accuracy of Different Data Transformation Compared with KNN Algorithm on WISDM Dataset

In general, it is evident that using reduced statistical summary samples instead of the entire dataset yield better accuracy for nearest neighbour classification. It is also clear that this approach can transform a training set into a reduced set using clustering and extraction of statistical summary from the clusters to obtain a better nearest neighbour classification accuracy.

It should be noted that the general low accuracy below 90% for this dataset can be attributed to the nature of the dataset in terms quality of features and the subjects used for data collection. The dataset contains data from 32 different users of varying characteristics in performing the designated activity. This produces many variations in the training and testing data. Nevertheless, the performance of the statistical summary data are good given the fact that they can use a reduced dataset for online recognition compare to KNN that requires the entire training instance to achieve good performance.

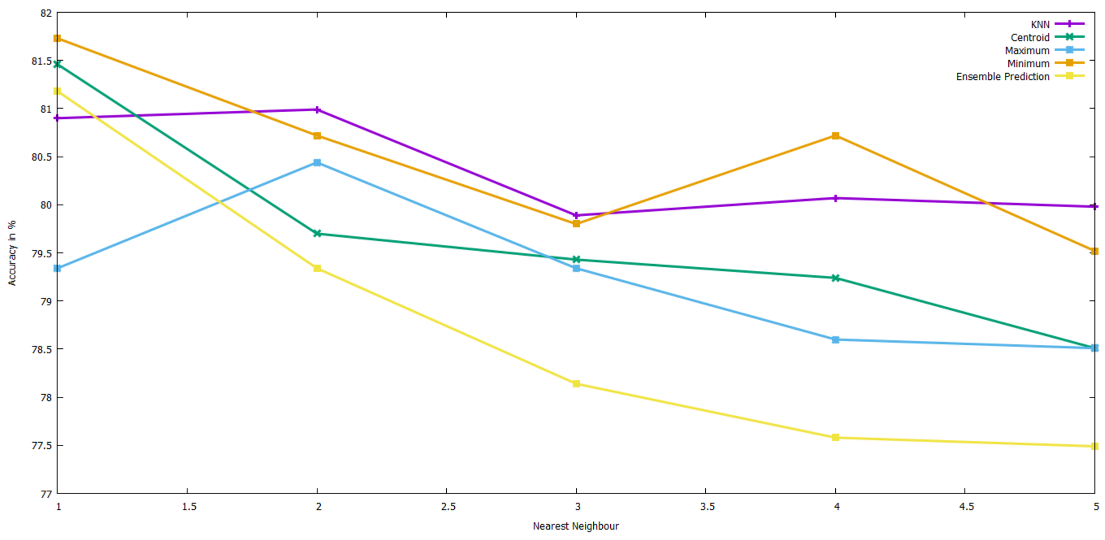


Figure 4.2: Accuracy of Using Different Data Transformation and KNN Algorithm with WISDM Dataset

4.4.2 ClusterNN Performance on HARS Dataset

The results obtained for the HARS dataset are presented in Tables 4.6, 4.7 and 4.8. The tables show the results for the three different representation (centroid, maximum and minimum) of the original dataset with varying amount of data retention employed to classify test instances. As indicated in Table 4.6, centroid data representation gives its best accuracy of 91.65% in classifying test data when nearest neighbour is set to 10 and the percentage of data retained is 40%. The best accuracy of using maximum data representation in place of the original dataset is 89.96%. This is obtained when K=6 and data retained is 80% as shown in Table 4.7. Similarly, the best accuracy obtained for using minimum data representation is 90.87% when K=7 and data retained is at 80% as shown in Table 4.8. Moreover, the accuracy of ensemble prediction of all the three data representations yields the best accuracy of 91.01% at the point when K=7 and a data retention of 60% as shown in Table 4.9.

Table 4.10 shows the comparison of the accuracy of KNN and the three data summaries of ClusterNN. The KNN utilised all the training dataset while the centroid, minimum and maximum data summaries utilised a reduced and transformed form of the original dataset. It can be seen that the best accuracy of 90.77% is obtained for KNN when the nearest neighbour for deciding the final label of an instance

Centroid Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	89.45	88.77	89.58	88.77	89.38	87.85	88.16	87.58	88.16
2	87.65	88.29	89.07	88.39	88.33	87.00	86.94	86.80	86.94
3	90.33	89.85	90.70	89.75	90.02	88.97	89.35	89.41	89.35
4	90.50	90.60	90.80	90.30	90.13	89.07	89.62	89.45	89.62
5	90.50	90.40	90.63	90.46	90.70	89.89	89.99	90.13	89.99
6	90.77	**91.14**	**91.28**	91.35	90.97	90.23	90.23	90.26	90.23
7	90.74	90.74	90.77	91.21	90.91	90.74	90.46	89.92	90.46
8	91.14	91.11	90.91	**91.65**	**91.11**	90.94	90.23	90.23	90.23
9	90.40	90.40	90.46	91.11	90.60	90.87	**90.57**	**90.57**	**90.57**
10	**91.62**	90.70	90.84	91.52	91.01	**91.14**	90.40	90.50	90.40

Table 4.6: Accuracy of Using Centroid Characteristic with Varying Percentage of Data Retained on HARS Dataset

Maximum Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	66.03	79.00	82.46	83.58	85.58	86.46	87.17	87.61	87.17
2	62.71	73.19	78.96	80.86	83.24	84.66	85.44	85.88	85.44
3	71.46	80.12	85.31	85.82	87.51	88.26	88.67	89.18	88.67
4	69.19	78.62	84.32	85.07	87.00	87.65	88.19	88.94	88.19
5	72.18	80.56	85.04	86.26	87.72	89.21	89.45	89.79	89.45
6	71.16	79.88	85.24	86.19	87.55	88.70	89.01	**89.96**	89.01
7	72.82	81.37	85.37	**86.63**	88.19	**89.51**	88.94	89.62	88.94
8	72.58	**81.64**	85.61	86.56	88.09	88.84	89.24	89.18	89.24
9	**73.94**	81.37	**86.09**	86.46	**88.80**	89.11	**89.48**	89.75	**89.48**
10	73.29	81.34	85.95	86.09	88.67	89.11	89.35	89.75	89.35

Table 4.7: Accuracy of Using Maximum Characteristic with Varying Percentage of Data Retained on HARS Dataset

Minimum Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	74.69	79.50	83.92	85.31	86.73	87.17	87.44	87.72	87.44
2	70.41	74.55	80.56	82.05	84.26	85.04	85.92	86.02	85.92
3	77.13	82.22	86.73	86.87	88.53	89.24	89.58	89.72	89.58
4	74.89	79.54	85.14	85.71	87.51	88.50	89.04	89.58	89.04
5	77.20	82.97	86.22	87.78	89.18	89.99	90.36	90.53	90.36
6	76.42	81.81	86.60	86.77	88.97	89.68	90.23	90.33	90.23
7	77.50	84.39	87.31	87.58	89.38	**90.70**	**90.43**	**90.87**	**90.43**
8	76.89	83.61	**87.31**	87.34	88.90	89.92	90.06	90.46	90.06
9	**77.71**	**85.27**	87.17	**88.02**	**89.68**	90.23	89.99	90.40	89.99
10	76.93	84.56	87.21	87.82	89.04	90.26	89.99	90.53	89.99

Table 4.8: Accuracy of Using Minimum Characteristic with Varying Percentage of Data Retained on HARS Dataset

	Ensemble Predictions								
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	78.18	84.73	87.95	87.72	88.50	88.16	87.99	87.92	87.99
2	73.97	79.71	84.32	85.17	86.09	86.16	86.53	86.66	86.53
3	80.18	86.63	**89.68**	89.14	90.16	89.85	90.02	89.99	90.02
4	78.69	84.83	88.46	88.56	89.68	89.58	89.41	89.75	89.41
5	79.64	86.53	88.43	89.62	**90.50**	90.91	**90.70**	**90.77**	90.70
6	79.67	85.88	88.94	89.38	90.43	90.53	90.33	90.74	90.33
7	80.22	86.60	89.24	**89.99**	90.33	**91.01**	90.70	90.70	**90.70**
8	80.32	87.04	89.11	89.96	89.92	90.57	90.13	90.63	90.13
9	80.25	87.11	89.07	89.62	90.16	90.67	90.33	90.50	90.33
10	**80.59**	**87.28**	88.87	89.68	89.79	90.97	90.09	90.57	90.09

TABLE 4.9: Accuracy of Using Ensemble Prediction with Varying Percentage of Data Retained on HARS Dataset

is set to 8. This accuracy is less than the centroid data representation with K ranges between 1 and 10 and percentage of data reduction between 10% and 60% level. This shows that a reduced set using centroid data is better than the KNN approach that uses all the training dataset.

Since other characteristics utilized lower amount of data, their accuracy can be traded-off for the smaller amount of data required when compared to the basic KNN that utilized all the training data.

K	KNN	Centroids-40%	Maximum-80%	Minimum-80%	Ensemble Prediction-60%
1	88.19	88.77	87.61	87.72	88.16
2	86.29	88.39	85.88	86.02	86.16
3	89.35	89.75	89.18	89.72	89.85
4	89.11	90.30	88.94	89.58	89.58
5	89.99	90.46	89.79	90.53	90.91
6	89.99	91.35	**89.96**	90.33	90.53
7	90.50	91.21	89.62	**90.87**	**91.01**
8	**90.77**	**91.65**	89.18	90.46	90.57
9	90.53	91.11	89.75	90.40	90.67
10	90.40	91.52	89.75	90.53	90.97

TABLE 4.10: Accuracy of the Three Different Data Summaries of ClusterNN Compared with KNN on HARS Dataset

These results indicate that there is a trade-off between accuracy and the amount of data retained for classification across each of the three reference data and their ensemble predictions. Thus, the percentage of data reduction can be selected based on the level of desired accuracy and available computational resource. For this

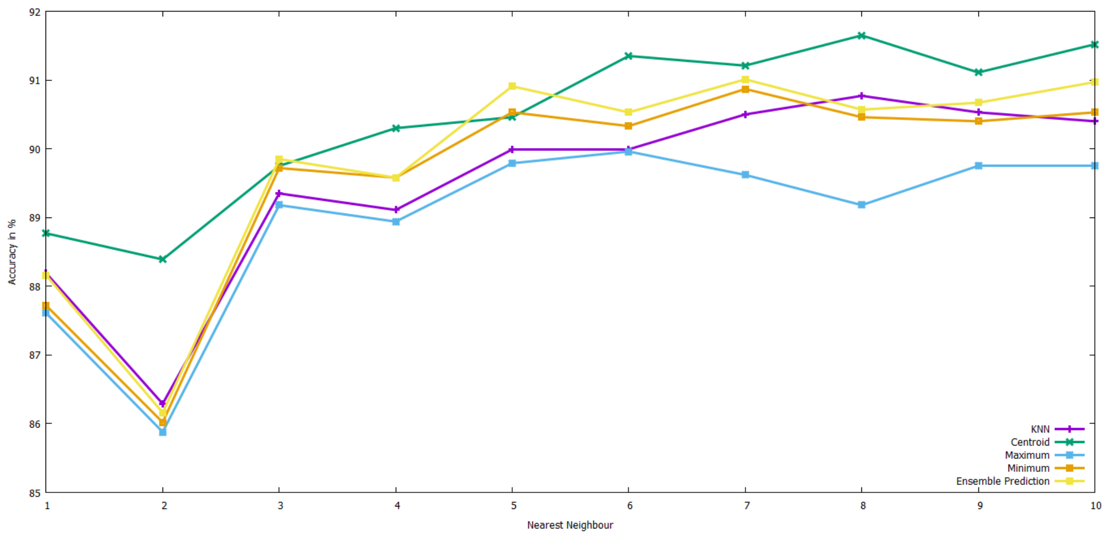


Figure 4.3: Accuracy of Using Different Data Summaries of ClusterNN Compared with KNN on HARS Dataset

dataset, one can take the 10% data reduction and the centroid data as the reference training data for classifying new instances. Figure 4.3 shows the comparative accuracy of using each of the three reduced data as the training set for nearest neighbour classification and their ensemble predictions. It can be observed that the accuracy of minimum and centroid are better than the KNN. This is more important since the two use a less amount of data for prediction. Also the accuracy of the ensemble prediction is also better when k=7 and the data reduction rate is 60%. It is only the maximum data that gives a lesser accuracy than KNN although the difference is very small. With this, it can be concluded that the presented approach of reducing the dataset with different statistical summary that are extracted from the clusters is better than using the KNN with all the training data for predictions.

4.4.3 ClusterNN Performance on Opportunity Dataset

The result for the experiments with Opportunity dataset is presented here. Tables 4.11, 4.12 and 4.13 show the results for the three different statistical summaries data (centroid, maximum and minimum) that were derived from the original dataset with varying amount of data retention employed to classify test instances.

As indicated in Table 4.11, using ClusterNN with centroid data gives its best accuracy of 96.28% in classifying test data when nearest neighbour is set to 8 and the percentage of data retained is 20%. The best accuracy of using maximum summary data in place of the original dataset is 92.48%. This is obtained when K=7 and data retained is 40% as shown in Table 4.12. Similarly, the best accuracy obtained for using minimum data representation is 95.62% when K=7 and data retained is at 30% as shown in Table 4.13. Moreover, the accuracy of ensemble predictions of all the three data yielded the best accuracy of 93.27% at the point when K=5 and a data retention of 40% as shown in Table 4.14.

Centroid Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	93.01	92.81	93.40	90.02	88.91	90.19	89.66	90.77	89.91
2	91.38	92.99	93.32	90.05	89.11	89.40	88.63	90.39	89.64
3	94.18	93.84	94.71	91.68	90.55	91.12	89.76	91.75	90.87
4	94.78	94.37	94.79	91.91	90.77	91.62	90.86	92.30	91.63
5	**93.75**	94.61	94.67	92.06	91.12	92.41	90.55	92.46	92.44
6	95.00	95.44	95.66	94.15	91.96	92.98	91.76	93.35	93.16
7	94.66	96.12	95.61	93.15	92.21	93.03	91.25	93.21	92.94
8	95.07	**96.28**	96.17	**94.59**	92.10	93.23	92.55	**93.65**	93.62
9	93.58	95.66	**95.37**	92.26	91.78	92.24	91.90	93.11	92.73
10	94.33	93.26	93.80	93.72	**92.54**	**93.28**	**93.86**	93.43	**92.95**

Table 4.11: Accuracy of Using Centroid Characteristic with Varying Percentage of Data Retained on Opportunity Dataset

Maximum Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	70.22	83.57	86.95	90.42	88.49	89.55	88.06	89.61	88.69
2	66.26	78.13	83.66	87.77	86.03	86.90	86.32	87.88	87.03
3	75.56	84.63	90.30	91.42	89.89	90.98	89.22	91.18	91.04
4	72.22	82.65	88.73	90.32	89.89	90.33	89.78	90.94	90.17
5	76.50	84.69	89.96	91.16	90.63	**91.31**	90.60	91.79	91.20
6	75.14	84.18	90.11	91.20	90.08	91.15	90.03	91.96	91.53
7	**77.35**	85.44	89.99	**92.48**	89.63	91.18	90.13	91.62	91.36
8	75.76	**86.18**	90.37	91.85	89.51	90.39	89.39	91.18	90.85
9	76.95	85.95	**90.97**	91.38	90.52	90.99	90.30	91.75	91.42
10	75.68	83.52	88.60	91.86	**91.97**	91.13	**91.69**	**92.20**	**91.83**

Table 4.12: Accuracy of Using Maximum Characteristic with Varying Percentage of Data Retained on Opportunity Dataset

Minimum Data									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	79.30	84.02	92.62	89.54	87.76	89.67	87.83	89.72	89.20
2	73.57	79.14	90.82	87.14	87.02	87.71	86.07	88.02	87.50
3	81.23	86.87	94.41	90.77	90.72	91.50	90.29	91.72	91.23
4	79.33	83.77	94.28	91.49	90.58	91.53	90.42	91.58	91.09
5	82.06	87.30	95.43	92.62	91.53	92.31	**90.64**	92.53	91.86
6	79.63	86.42	95.22	92.12	91.33	92.24	91.09	92.33	91.88
7	82.29	88.41	**95.62**	**92.42**	**91.87**	92.27	91.00	**92.87**	**92.35**
8	80.73	88.21	94.86	92.63	91.46	91.46	91.16	92.46	92.10
9	**81.57**	**89.35**	94.83	91.60	91.40	92.09	90.93	92.4	91.58
10	80.56	88.95	95.39	91.69	91.53	**92.20**	91.08	92.53	92.31

TABLE 4.13: Accuracy of Using Minimum Characteristic with Varying Percentage of Data Retained on Opportunity Dataset

Ensemble Predictions									
K	10%	20%	30%	40%	50%	60%	70%	80%	90%
1	81.98	89.11	92.41	90.40	88.43	89.65	89.14	90.16	89.47
2	77.56	84.10	90.78	88.73	87.05	87.67	86.43	88.16	87.65
3	84.07	91.07	94.33	92.69	90.61	91.57	90.54	91.85	91.12
4	82.49	89.52	94.42	91.83	89.91	91.38	90.20	91.58	90.89
5	84.03	91.14	95.85	**93.27**	91.70	**91.98**	91.67	92.91	92.40
6	82.83	90.01	95.06	91.63	91.29	92.41	90.95	92.53	91.59
7	84.45	91.21	95.31	93.95	**91.65**	92.86	91.24	**93.01**	92.51
8	**84.24**	91.29	94.82	92.91	91.09	92.26	90.64	92.57	92.24
9	84.96	91.74	95.18	92.08	91.17	92.27	91.44	92.67	92.34
10	84.75	**91.38**	**95.76**	92.41	91.31	92.73	**91.35**	92.97	**92.33**

TABLE 4.14: Accuracy of Using Ensemble Prediction with Varying Percentage of Data Retained on Opportunity Dataset

Table 4.15 shows the comparison of the accuracy of KNN and the three data representations. The KNN utilised all the training dataset while the centroid, minimum and maximum data representation utilised a reduced and transformed form of the original dataset. The highest accuracy of 93.62% is obtained for KNN when the nearest neighbour for deciding the final label of an instance is set to 9. This accuracy is less than the centroid data representation with K ranges between 3 to 10 and percentage of data retention between 10% and 90% level. This shows that a reduced set using centroid data is better than the KNN approach that uses all the training dataset. Similarly, the minimum data summary yield a better accuracy than kNN when data retained is 30% and K ranges between 3 to 10. The same observation is recrded for the ensemble data with its accuracy outperforming kNN at 30% and K ranges between 3 to 10. However, the maximum data summary with its best accuracy of 92.48% is slightly less than the kNN best accuracy of 93.62%.

Since ClusterNN uses a lower amount of data than the kNN, it implies that the proposed statistical summaries contain adequate discerning patterns to guarantee good accuracy than KNN. In the situation where kNN outperformed ClusterNN, the lower amount of training data used can be traded-off against the accuracy of kNN that utilized all the training data.

K	KNN	Centroids-20%	Maximum-40%	Minimum-30%	Ensemble Prediction-30%
1	91.76	92.81	90.42	92.62	92.41
2	88.83	92.99	87.77	90.82	90.78
3	91.80	93.84	91.42	94.41	94.33
4	91.89	94.37	90.32	94.28	94.42
5	92.32	94.61	91.16	95.43	95.85
6	93.27	95.44	91.2	95.22	95.06
7	92.51	96.12	**92.48**	**95.62**	95.31
8	92.80	**96.28**	91.85	94.86	94.82
9	**93.62**	95.66	91.38	94.83	95.18
10	93.52	95.26	91.86	95.39	**95.76**

Table 4.15: Accuracy of the Three Different Data Representation Compared with KNN Algorithm on Opportunity Dataset

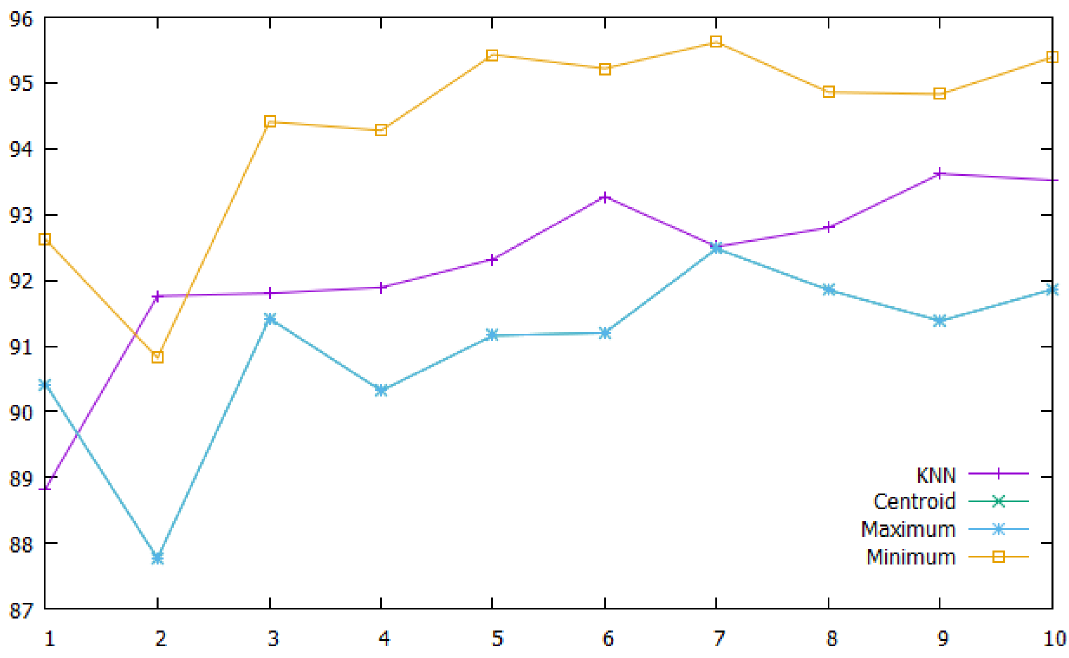


Figure 4.4: Accuracy of Using ClusterNN Different Summary Data and KNN Algorithm with Opportunity Dataset

These results indicate that there are varying accuracies based on the amount of data retained for classification across each of the three reference statistical summary data and their ensemble predictions. Thus, the percentage of data reduction can be selected based on the level of desired accuracy and available computational resource. For this dataset, one can take the 20% data reduction and the centroid data as the reference training data for classifying new instances. The ensemble data can also be used at 30% data retention.

Figure 4.4 shows the comparative accuracy of using each of the three reduced data as the training set for nearest neighbour classification and their ensemble predictions. It can be observed that the accuracy of minimum and centroid are better than the KNN. This is more important since the two use a less amount of data for prediction. Also the accuracy of the ensemble prediction is also better when k=7 and the data reduction rate is 60%. It is only the maximum data that gives a lesser accuracy than KNN although the difference is very small. With this, it can be concluded that the presented approach of reducing the dataset with different statistical summary that are extracted from the clusters is better than using the KNN with all the training data for predictions.

4.4.4 Comparative Evaluation of Weighted Ensemble ClusterNN with other Algorithms

Having observed the relative performances of each statistical summary extracted from the original dataset and the ordinary majority voting ensemble, an improved weighted voting scheme for taking the final decision on the classification of samples was implemented and evaluated.

The weighted ensemble voting scheme assigned weight to each of the decision made from each characteristic with centroid having more weight than the minimum and maximum characteristics. The results obtained with comparative performance with three other algorithm is presented in Table 4.16.

	Weighted Ensemble ClusterNN	**KNN**	**Decision Tree**	**Naïve Bayes**
WISDM	92.50	81.99	86.00	87.10
HARS	97.97	90.77	90.20	91.20
Opportunity	98.99	92.57	93.56	94.12

TABLE 4.16: Comparative Evaluation of ClusterNN with other Algorithms

Statistical Test Result 4.4.1. **Paired T-Test and Confidence Interval: Weighted Ensemble ClusterNN** *vs* **KNN**

	N	Mean	StDev	SE Mean
Weighted Ensemble ClusterNN	3	96.49	3.49	2.01
KNN	3	88.44	5.66	3.27
Difference	3	8.04	2.17	1.25

95% Confidence Interval for mean difference: (2.65, 13.44)

T-Test of mean difference = 0($vs \neq 0$): T-Value = 6.42 P-Value= 0.023

Statistical Test Result 4.4.2. **Paired T-Test and Confidence Interval: Weighted Ensemble ClusterNN** *vs* **Decision Tree**

	N	Mean	StDev	SE Mean
Weighted Ensemble ClusterNN	3	96.49	3.49	2.01
Decision Tree	3	89.92	3.79	2.19
Difference	3	6.567	1.171	0.676

95% Confidence Interval for mean difference: (3.657, 9.477)

T-Test of mean difference = 0($vs \neq 0$): T-Value = 9.71 P-Value= 0.010

Statistical Test Result 4.4.3. **Paired T-Test and Confidence Interval: Weighted Ensemble ClusterNN** *vs* **Naive Bayes**

	N	Mean	StDev	SE Mean
Weighted Ensemble ClusterNN	3	96.49	3.49	2.01
Naive Bayes	3	90.81	3.53	2.04
Difference	3	5.68	0.98	0.57

95% Confidence Interval for mean difference: (3.244, 8.116)

T-Test of mean difference = 0($vs \neq 0$): T-Value = 10.03 P-Value = 0.010

With only 15% data retention, the weighted ensemble of the statistical summary decisions outperformed both KNN, Naive Bayes and Decision Tree algorithms for both the WISDM and HARS and Opportunity datasets. The weighted ensemble is applied to the decision from each characteristic by assigning more weight to the more accurate ensemble component and lesser weight to the less accurate ensemble components. The results and the statical tests are presented in Table4.16and statistical test 4.4.1,4.4.2 and 4.4.3 respectively.

The paired t-test on the result in Table 4.16, shows that WISDM dataset has lesser accuracy under each algorithm compared with HARS and Opportunity datasets. This can be attributed to the underlying features of the datasets. However, with the statistical summaries, the accuracy of ClusterNN outperformed all the algorithms that used the complete dataset. Also, the statistical summaries utilised lesser training instances unlike the remaining algorithms that used all the available reference training dataset. In conclusion, the ClusterNN with its reduced dataset is more accurate and robust for activity recognition.

4.5 Chapter Summary

This chapter has presented an approach for extracting statistical summaries and its associated data from the original large dataset. The proposed statistical summary are the centroid, maximum and minimum features extracted from the sub-clusters created from each class of activity. The advantage of the extracted features and its associated data is that they resulted in a reduced set of data for the instance-based classification approach.

An instance-based classifier for activity recognition that uses the extracted characteristic data as reference data for classifying new instances is presented. The classifier computes the similarity between a new instance and the reference data summaries and assigned the labels of the nearest similarity to the instance. The decisions from each reference feature are combined using majority voting to arrive at the final label. Evaluation of the approach shows that the accuracy of using each of the characteristic for similarity measure depends on the number of sub-clusters created in the dataset. Also, the weighted ensemble voting scheme of the classifier outperforms all the benchmarked algorithms.

Chapter 5

Unsupervised Change Detection Framework for Activity Recognition

5.1 Introduction

The traditional approach for inertial sensors based activity recognition involves: i) collection of labelled data from the subjects that perform sample activities to be recognised ii) classification model generation by using collected data to train and test classification algorithms iii) a model deployment stage where the learnt model is transferred to the mobile device for identifying new unseen data.

This approach for activity recognition performed the model generation phase on remote systems and pushed the generated model to the client applications to recognise new user activities. The drawback of this approach is that the model is static and does not reflect possible changes in the distribution of new evolving data. Another approach that aims to eliminate this induced the model by using the user self-annotated data from the device so that the model can be tuned to the individual user.

However, the two approaches are still not immune from changes that may occur in the underlying distribution of the unseen incoming data due to differences in user characteristics and drifts in sensor data [44]. This usually results in decreasing performance in the accuracy of the model as new users with different characteristics begin to use it. For example, a model that is trained to recognise walking activity may take a new data from another slightly different distribution that corresponds to jogging for another user and classify it as walking. At this point, the model has suffered from the phenomenon called concept change. The source of the change can be known or unknown. But for activity recognition problem, it has been shown to be caused by some factors such as dissimilarities between the data of users used during training and those using the activity recognition model. [44, 48]. It may also be caused by the displacement of the sensors and orientation effect on the sensor readings [11, 97].

Various approaches have been developed for activity recognition model adaptation during online operation. All of these approaches [4, 48, 110] are blind in the sense that they do not identify concept changes before they start the adaptation process. In the field of stream mining, a related problem exists and two approaches have been identified for handling concept drift. They are named the informed and uninformed adaptation approaches [30]. The informed approaches react to the occurrence of concept drift by ensuring that the drift point is detected before taking any action. In contrast to this, uninformed approaches incrementally and continuously update the model each time a sample or set of samples are integrated into the model. The drawback of the uninformed adaptation approaches is that they can react slowly to concept drift and consume system resources as they continuously adapt the old concept which may need to be replaced out-rightly or maintained without adaptation [30]. Hence an informed adaptation scheme is better and well suited for better management of concept drift.

Many of the approaches [28, 34, 91] for the detection of concept drift assume that the data are labelled. But, in the domain of activity recognition, labels are not easy to come by during online recognition. This is because users will be required to provide a label for each activity being performed. This is impractical and tedious to do. Hence, the approach that is based on unsupervised detection which eliminates the need for ground truth to detect changes is highly desirable.

This chapter presents a new approach called Unsupervised Change Detection for Activity Recognition (UDetect) which unlike other approaches that use ground truth [28, 34, 91], does not require the presence of labels to detect the drifts. The approach extends the Shewart control charts[103] parameters used in statistical process control domain, for change detection in activity recognition. The approach aims to detect the variability between reference users data and the predicted data from the same or another user. In particular, the contribution of the UDetect approach is that it can detect changes in activity recognition model without using ground truth for error monitoring unlike other approaches [28, 34, 91] that requires ground truth for the detection of concept drift. Instead, the approach uses a data discrimination method and a base classifier to detect the changes by using the parameters computed from the reference data of each class to discriminate outliers in the new data being classified to the same class.

The rest of this chapter is organised as follows: Section 5.2 presents the concept change detection framework. Section 5.3 and 5.4 present the experimental evaluation results and discussions. The summary of the chapter concludes the presentation of this chapter.

5.2 Unsupervised Change Detection Framework

The proposed and implemented method of unsupervised change detection is a technique that detects changes in the activity recognition model. The change in the model manifests in the form of reduced model accuracy. In this case, the detected change is viewed from the perspective of detecting incoherency between the data that represents the original model of the activity and the new data that are classified to the same activity. The central idea of the framework is that if a classifier is trained on a set of data, the model obtained will continue to be accurate if the new unseen data fit coherently with the training data with some level of deviation; but if the new data classified to a particular class varies widely from the pattern of the reference data, it means the data should belong to a different class rather than where it is classified. Therefore, the technique relies on monitoring the variation between the reference data of a class and the new data classified to the class. The method does not assume the presence of ground truth with each arrival of new samples to be classified. Hence, it reflects a realistic scenario for detecting concept drift in activity recognition. The method employs the base classifier presented in the previous chapter. The classifier is pre-trained on a set of target concept to classify new samples, and when the classifier is deployed for recognition of new samples, batches of samples classified to the same class are maintained. Intuitively, if the classifier is not misclassifying samples to a class, the distribution of the attributes in the samples classified to the same class should be stable. A change in the distribution of the samples classified to the same class is signalled if the distribution parameters of the samples deviate significantly from the previous reference parameters. The method relies on this assumption and monitors the parameter computed from the batches of data that are classified to the same class. If this parameter is within a threshold, no change is detected, but if this parameter exceeds a threshold a change is signalled in that particular class of data.

The realisation of the technique is formulated as a two level architectural framework comprising of the off-line phase and the online phase. The off-line phase extracts the change parameters from the reference dataset. It also performs the function of converting the multidimensional data into uni-dimensional data. The online phase functions include the classification of new samples and the detection of changes in each class of activity present in the datasets. The details of these two components are discussed further in the following sub-sections.

5.2.1 Conceptual Framework of the Offline Component

The functional components of the off-line phase of the framework is shown in Figure 5.1. The main functions performed by the off-line component include (i) Windowing and Segmentation (ii) Window summary computation (iii) Change parameter computation and (iv) Model Generation.

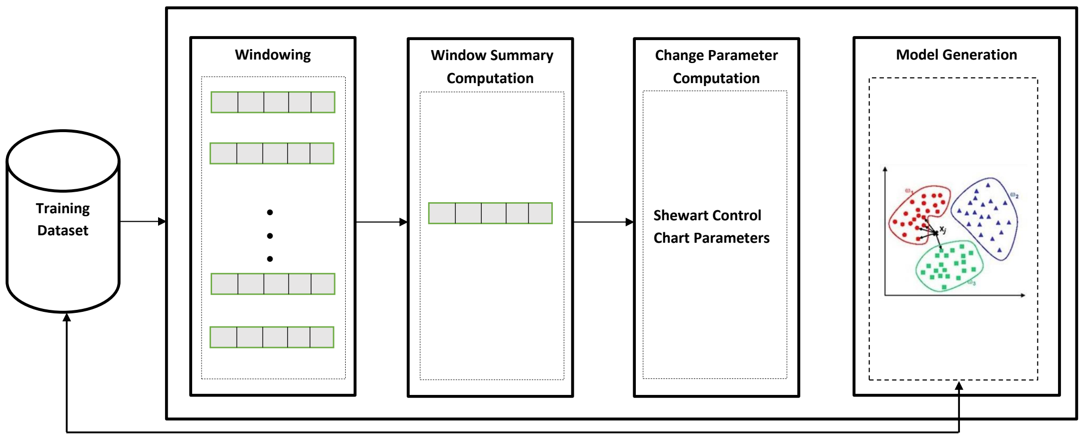

Training Dataset
Windowing
Window Summary Computation
Change Parameter Computation
Shewart Control Chart Parameters
Model Generation

Figure 5.1: Off-line Components of the Unsupervised Change Detection Framework

5.2.1.1 Windowing

The windowing sub-component performs the function of partitioning and segmenting the incoming data into chunks of fixed sizes. Each multi-dimensional chunk is passed to the next layer for further processing. Different windowing strategies can be used. The experiment performed with overlapping and non-overlapping windows suggests that non-overlapping window performs better.

5.2.1.2 Window Summary Computation

The window summary computation is the second layer of the off-line component that is saddled with the function of converting each chunk of multidimensional instances obtained from a window into a single summary value that will be used as input to the change parameter computation. After accumulating a batch of multidimensional data of a fixed size n from a given class of activity data in the reference dataset, the summary value of the data is computed as the average distance between each point in the batch of data and their mean. Algorithm 6 is used for computing the summary values. The set of *avgdtc* statistic computed from the different activity classes are then accumulated and used by the next component to compute the change parameters according to each class of activity.

```
Algorithm 6: Window Summary Computation
Input: $D = \{(x^1, y_i), (x^2, y_i), ..., (x^n, y_i)\}$
// D = Reference Dataset
Output: $avgdtc_{y_i,...,y_k}\{1..m\}$
// set of window summary values computed for each class of activity $y_i$ to $y_k$
1 foreach class $y_i$ in $D$ do
2     while data $\in$ class $y_i) \neq$ empty do
3         $X = getWindowData(n, y_i, D)$
          // obtained fixed non-overlapping data chunk of size n from the window of
             activity of class $y_i$
4         $\overline{\varpi c} = Centroid(X, n)$
          // compute the centroid of n data items in X
5         $avgdtc_{y_i}[k] = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x^{(i)} - \varpi c)^2}$
          // $x^{(i)} \in R^n$ are the instances in the window and $\overline{\varpi c} \in R^n$ is defined as the
             centroid of the data in the window computed
6     end
7 end
8 return $avgdtc_{y_i,...,y_k}\{1..m\}$
// set of window statistics computed for each class of activity.
```

5.2.1.3 Change Parameters Computation

The change parameter computation sub-component computes the change parameters for each class of activity present in the reference dataset. The computation is done by making use of Algorithm 7. The input to the algorithm are the set of window summary values computed from the preceding component for each class of activity. The algorithm is based on the statistical process control method by adapting the Shewhart individuals control chart parameters [65] to identify the significant variations in the window summary values computed from the newly classified incoming samples. The constants in the algorithm depend the nature of the distribution and size of the data. Assume that the *avgdtc* statistic samples follow a normal distribution with mean μ and standard deviation $\sigma_{avgtc} = \frac{\sigma}{\sqrt{n}}$, then $1 - \alpha$ is the probability that the mean value of any sample of *avgdtc* will fall between:

$$\mu + z_{\frac{\alpha}{2}}\sigma_{avgdtc} = \mu + z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \quad and \quad \mu - z_{\frac{\alpha}{2}}\sigma_{avgtc} = \mu - z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \tag{5.1}$$

Hence, Equation 5.1 can be used as upper and lower control limits for the *avgdtc* sample statistic. But, the entire population mean μ and standard deviation σ are not known. However, they can be estimated from the preliminary set of *avgdtc* values computed from the reference dataset that indicates the normal patterns of the activities. The mean μ is estimated by computing the mean of the *avgdtc* from the preliminary reference samples as $\overline{avgdtc_{y_i}} = \frac{1}{n}\sum_{k=1}^{n} avgdtc_{y_i}$ The standard deviation σ can be estimated from the ranges of two successive samples i.e $R_{avgdtc} = avgdtc_{i+1} - avgdtc_i$.

An established relationship between the range R and standard deviation σ of a sample from a normal distribution is $W = \frac{R}{\sigma}$ [65]. The random variable W is called the relative range. The parameters of the distribution of W are a function of the sample size n. The mean of W is d_2. Thus, an estimator of σ is $\frac{R}{d_2}$. The values for d_2 has been established and the table of values is given in Appendix B. Therefore, the mean μ of the population is estimated with the average of the preliminary samples and the standard deviation σ is estimated with $\frac{\overline{R_{y_i}}}{d_2}$. Finally, the upper control limit for the individual sample of *avgdtc* is computed from Equation 5.1 as:

$$UCL_{avgdtc} = \mu + 3\frac{\sigma}{\sqrt{n}} = \mu + \frac{3\overline{R_{y_i}}}{d_2\sqrt{n}} \tag{5.2}$$

Since the range is computed from 2 consecutive samples the equivalent value of d_2 is 1.126. Also, since each *avgdtc* sample is treated individually $\sqrt{(1)} = 1$. Hence, the UCL_{avgdtc} becomes:

$$UCL_{avgdtc} = \mu + 2.66\overline{R_{y_i}} \tag{5.3}$$

Similarly, the LCL_{avgdtc} is given as:

$$LCL_{avgdtc} = \mu - 2.66\overline{R_{y_i}}* \tag{5.4}$$

The computed parameters for each class of activity are stored in a variable called *parameterList* Algorithm 7 *line* 7. The parameters for each class are used separately to monitor any change that may occur in the data classified to each of the activity class during the online classification of new unseen instances.

Algorithm 7: Change Parameter Computation

```
Input: $avgdtc_{y_i,\ldots,y_c}\{1..m\}$
// set of window summary values of different classes obtained from reference dataset
Output: $parameterList$
// change monitoring parameters for each class of data
1 foreach class $y_i$ in $avgdtc\{1..m\}$ do
2     $\overline{R_{y_i}} = \frac{1}{n} \sum_{k=1}^{n-1} avgdtc_{k+1} - avgdtc_k$
3     $\overline{avgdtc_{y_i}} = \frac{1}{n} \sum_{k=1}^{n} avgdtc_{y_i}$
4     $UCL_{avgdtc_{y_i}} = \overline{avgdtc_{y_i}} + 2.66 \times R_{y_i}$
5     $LCL_{avgdtc_{y_i}} = \overline{avgdtc_{y_i}} - 2.66 \times R_{y_i}$
6     $parameterList.append(UCL_{avgdtc_{y_i}},\ LCL_{avgdtc_{y_i}})$
7 end
8 return $parameterList$
```

5.2.1.4 Model Generation

The final layer of the off-line component is responsible for building a classifier prototype by making use of the reference dataset which represents the current activity patterns codified into the model. The main essence of the model generation is to train a learning model to be able to recognise the current pattern of activity that is present in the system. The classifier accuracy is then monitored when it is deployed to the online stage to recognise activity of the new set of data from the same set of users who performed the sample activities or entirely new users whose characteristics may totally diverge from those present in the training set.

5.2.2 Conceptual Framework of the Online Components

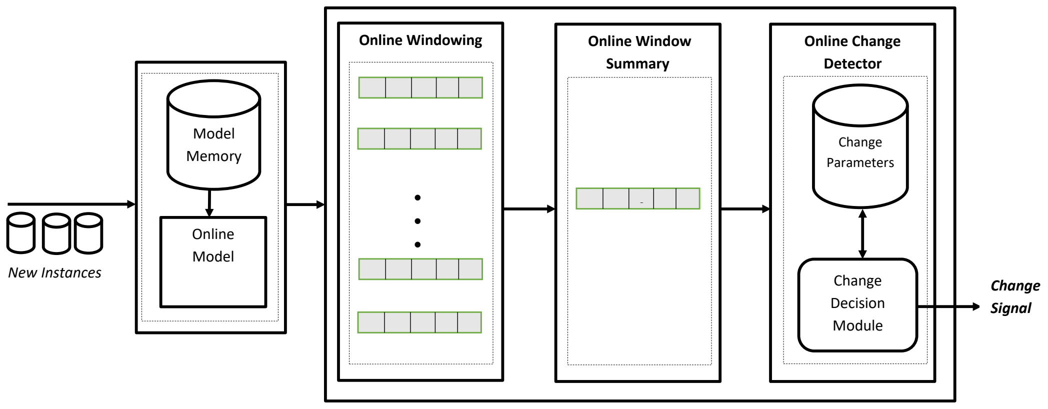


Figure 5.2: Online Components of the Unsupervised Change Detection Framework

Figure 5.2 shows the architecture of the online detection method. The components of the framework consists of: (i) The Online Model (ii) Online Windowing (iii)Online Window Summary Computation (iv)Online Detector.

5.2.2.1 Online Model

The online model is a classification model that has been pre-trained during the off-line stage. The model is meant to classify new samples of activity data. As the data are classified they are passed to the dedicated temporary buffers from where they are sent to the appropriate window for the windowing and segmentation operation to be performed on the data.

5.2.2.2 Online Windowing

The online windowing component performs the same functions as the offline windowing module. They both perform the function of segmenting the data in the window into fixed size chunks. There are separate windows dedicated to each class of activity and the data in each of the window are processed when the predetermined number of samples are available in the window. Each fixed size multidimensional chunk of data obtained from the window is passed to the window summary computation component to convert the entire chunk into a single value know as *avgdtc* statistic or window summary value.

5.2.2.3 Online Window Summary Computation

The online window summary calculation is carried out on the formed data of size n. The computation is akin to the off-line format of the component because the same parameter as the off-line component is required for the detection module. Hence, Algorithm 6 is employed in computing the online window summary as it was done in the off-line computation of the same parameter.

5.2.2.4 Online Detector

Algorithm 8: Online Change Detector

```
Input: $UCL_{avgdtc_{y_i}}$, $LCL_{avgdtc_{y_i}}$
// change parameters computed in the offline phase
Output: change signal
1  foreach instance $x_k$ classified to a stream of window activity do
2      $x_w = getWindowStep(n)$
       // Form a window Size from the Stream
3      $w_{statistic} = computeWindowSummary(x_w)$
       // Compute Window Summary
4      if $(w_{statistic} >= LCL_{avgdtc_{y_i}})$ and $(w_{statistic} <= UCL_{avgdtc_{y_i}})$ then
5          No change
6      end
7      else
8          change detected
9          initiate adaptation
10     end
11 end
```

The online detector module is the main component in the online detection framework that determines when a change occur in the new instances being classified. The change can manifest in different ways. One manifestation of change is the diminishing accuracy of the online model that classifies the samples, but this requires knowing the ground truth of the classified sample which is not realistic in this scenario of activity recognition. Therefore, an indirect approach to detect the change is to determine if the data classified to a class is divergent widely from the original baseline data of the class. Base on this, the online detector uses the already computed change parameters from the training samples to decide whether the new window summary fits coherently within the existing data in the samples or whether it diverges widely from the baseline data. If the new summary value fits coherently with the change parameters no change is detected and the sample is said to be within control but if the value diverges widely then a change is signalled. The online detection algorithm is presented in Algorithm 8. In this algorithm, as new samples are classified into designated windows, and the window summary statistic is computed on a fixed amount of window data, the static is compared to the change parameters and a decision is taken to declare a change or no change.

5.3 Experimental Study

The objective of the experiments is to identify when the accuracy of the underlying model begins to degrade without having access to the ground truth. This change point is due to the differences between reference data used for training and new unseen data during online activity recognition. To simulate this scenario, the data of one user was used for training and another subject data for evaluating the change point detection. In other words, a known amount of one user data is used as training data to create a bespoke up-to-date model and also used to compute change parameters. The training and the test data are then combined and passed to the model so that if there are differences between the distributions of the activity data between the users, the method should be able to identify the change points after the first user data. Hence, the first set of data to test is from the original user while the rest are from another user. This configuration is employed for all the experiments in this section.

5.3.1 Experiment with HARS Dataset

The Human Activity Recognition Using Smartphone Dataset-HARS described in Chapter 3 was used for this experiment. The configuration of the experiment involved setting the data of one user as reference dataset to serve as training set for the underlying model and to compute the change parameters (UCL and LCL).

Another user data is then set as the test set. Each class of activity has its own dedicated window for detecting the variability in the data classified to a given class. A batch size of 3 was utilised to compute the window summary statistic (*avgdtc*) from the window dedicated to each activity. The detection results obtained by setting the data of the user with ID 19 as training set while the combination of user 19 and 14 are set as test data are charted in Figures 5.3 to 5.8. The first 360 samples of the testing data belong to user 19 while the remaining 323 data points of the total 683 belongs to user 14. Figure 5.3 shows the chart of the values of the individual window summary statistic *avgdtc* that are obtained from the window of activity class 'walking'. A change is detected at the time step 489 and 647 on the chart. These are the points where the values of *avgdtc* exceed the upper control limit. and the distribution of the data changes from the initial data distribution resulting in deterioration in the accuracy of the model. It also indicates that the samples around these time steps are misclassified which makes their computed parameter goes out of control limit. Table 5.2 shows that there is misclassification error in this class amounting to error rate of 0.15.

The summary of the number of changes detected and the corresponding error rates in each class of activity across the different pairing of one user with another user is presented in Tables 5.1 and 5.2 respectively. The correlation test performed on the two set of data from the two tables using Minitab [2] statistical analysis software yields the result presented in statistical test result 5.3.1.

Statistical Test Result 5.3.1. **Correlation: Number of Changes** *vs* **Error Rate in HARS Dataset**

Pearson correlation of Number of Changes and Error Rate = 0.934 $P-Value =$ 0.000

The result indicates that the Pearson correlation between error rates and the number of changes detected in each class of activity for this sample dataset is 0.934 while the p-value is 0.000. Since the $p-value$ is smaller than 0.01, the null hypothesis about the correlation between the error rate and number of changes detected $H_0 = \rho = 0$ can be rejected in favour of the alternative hypothesis $H_0 : \rho > 0$. This implies that there is sufficient statistical evidence at the $\alpha : 0.01$ level to conclude that there is a significant relationship between the error rate and the number of changes detected in a class of activity.

		Number of Changes Detected per Class of Activity					
Reference User	**Test User**	**Walking**	**Walking Up-stairs**	**Walking Down-stairs**	**Sitting**	**Standing**	**Lying**
19	14	2	2	0	3	5	0
30	2	4	0	5	2	1	0
3	4	2	0	0	1	2	0
5	6	0	1	0	3	1	0
7	8	1	3	3	1	2	0
9	10	1	2	0	2	2	1
11	12	1	5	0	1	2	0
13	15	3	0	0	0	2	0
16	17	3	2	0	3	0	0
18	20	4	4	0	2	1	0
21	22	0	2	0	0	2	0
23	24	5	0	5	1	1	0
25	26	0	0	0	0	0	0
27	28	0	1	0	3	0	1
29	30	1	0	1	2	0	0

Table 5.1: Summary of Detections in HARS Dataset

		Error Rate per class of Activity					
Reference User	**Test User**	**Walking**	**Walking Up-stairs**	**Walking Down-stairs**	**Sitting**	**Standing**	**Lying**
19	14	0.15	0.24	0	0.32	0.36	0
30	2	0.10	0.00	0.45	0.19	0.11	0.00
3	4	0.16	0.05	0.01	0.09	0.21	0.01
5	6	0.01	0.08	0.00	0.30	0.06	0.01
7	8	0.09	0.27	0.12	0.09	0.24	0.00
9	10	0.11	0.18	0.00	0.13	0.25	0.10
11	12	0.14	0.46	0.00	0.14	0.06	0.04
13	15	0.32	0.02	0.00	0.05	0.14	0.01
16	17	0.05	0.17	0.00	0.26	0.03	0.00
18	20	0.39	0.44	0.01	0.18	0.08	0.00
21	22	0.02	0.24	0.00	0.04	0.05	0.00
23	24	0.47	0.00	0.45	0.15	0.13	0.00
25	26	0.00	0.00	0.00	0.00	0.00	0.00
27	28	0.02	0.11	0.01	0.33	0.00	0.05
29	30	0.10	0.00	0.10	0.20	0.00	0.01

Table 5.2: Error Rates in HARS Dataset

Similarly, changes are detected in the activity class 'Walking-Upstairs', 'Sitting' and 'Standing' shown in Figures 5.4, 5.6 and 5.7 respectively. The change points are indicated by the out of control limit points in the charts. The proportions of

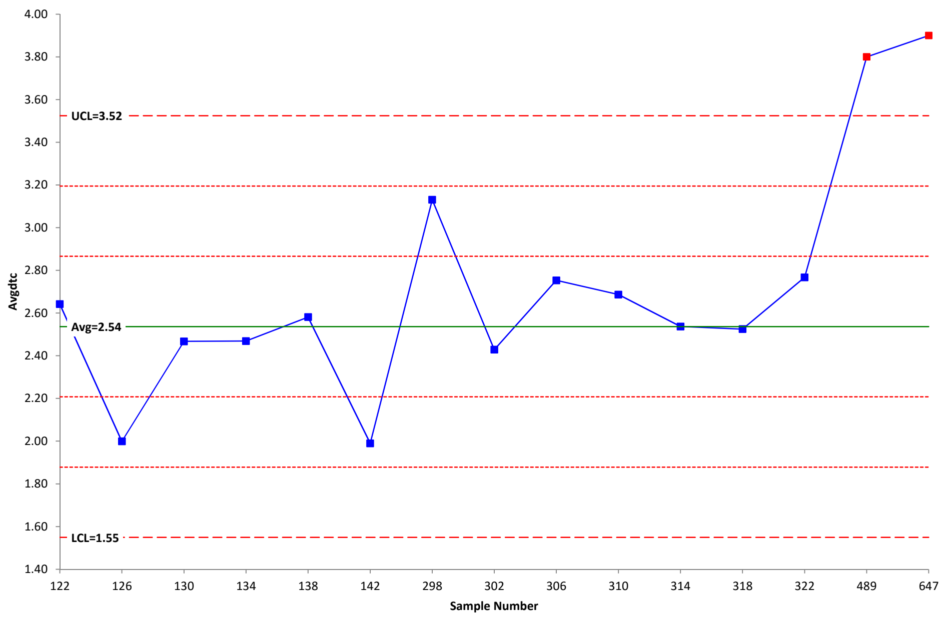


FIGURE 5.3: Parameter Chart of Walking Activity in User 19 Against 14

misclassified samples in these classes as shown in Table 5.2 corroborates the non-homogeneity of the data that are classified to the window dedicated to each class. Thus, the approach is able to detect changes in the distribution of the initial user data that belongs to the original activity and those that comes from another user. It should be noted that the points that are within control limits indicate instances from test data have the same and correct class as the initial training data.

There is no change is detected in the activity class 'Walking-Downstairs' (Figure 5.5) and class 'Laying' (Figure 5.8). This is evident by the absence of out of control points in the two charts for the two classes. This is because there is no variability in the training data of the user and the test data from another user and hence the proportions of their misclassified samples are 0 for each of the two classes as shown in Table 5.2.

5.3.2 Experiment with Opportunity Dataset

This experiment utilised the Opportunity activity recognition dataset that is described in Chapter 3. The dataset has more data points and is obtained from more inertia sensors than the HARS and WISDM dataset. The experiment was

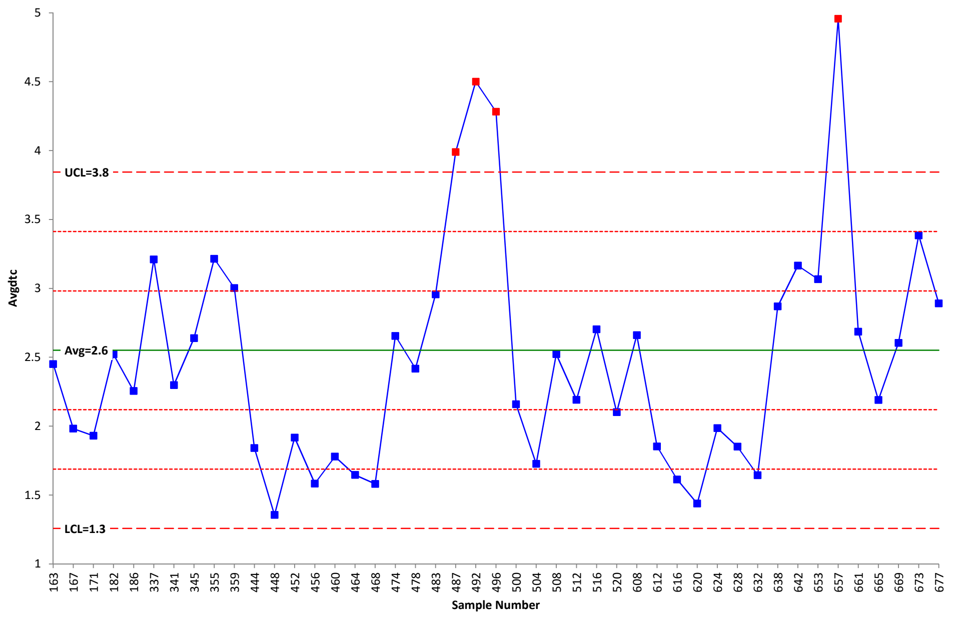


FIGURE 5.4: Parameter Chart of Walking-Upstairs Activity in User 19 Against 14

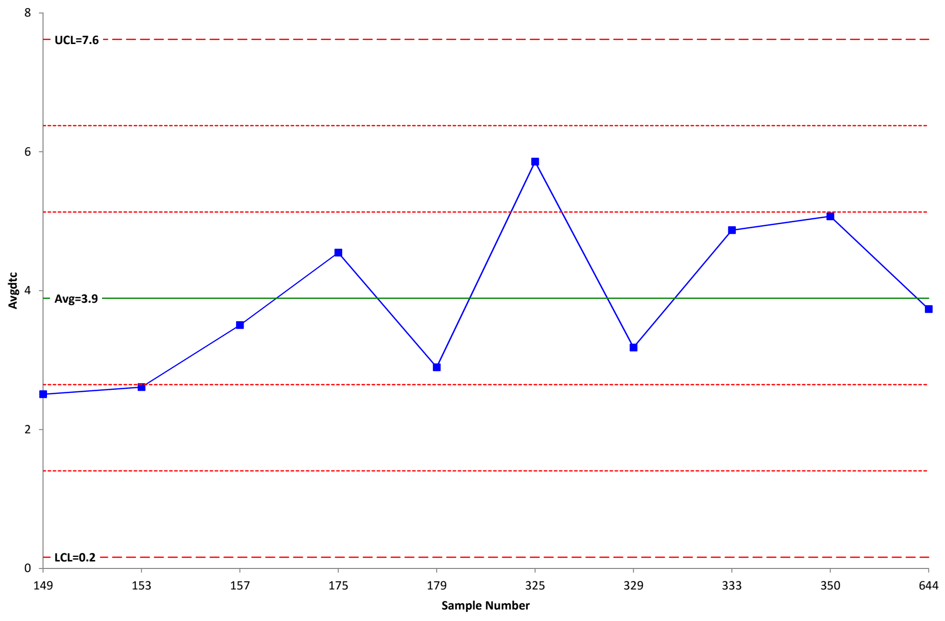


FIGURE 5.5: Parameter Chart of Walking-Downstairs Activity in User 19 Against 14

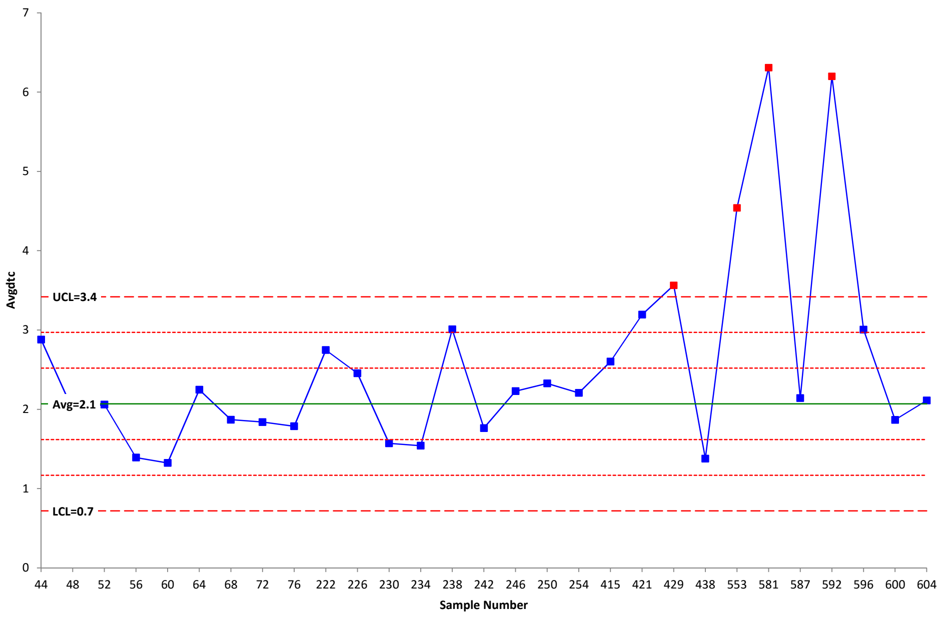


FIGURE 5.6: Parameter Chart of Sitting Activity in User 19 Against 14

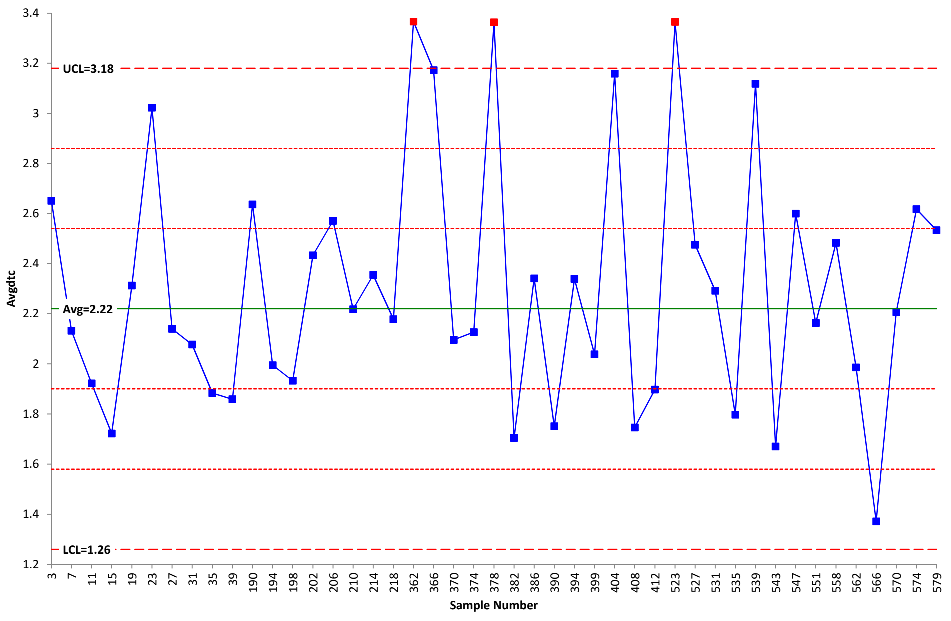


FIGURE 5.7: Parameter Chart of Standing Activity in User 19 Against 14

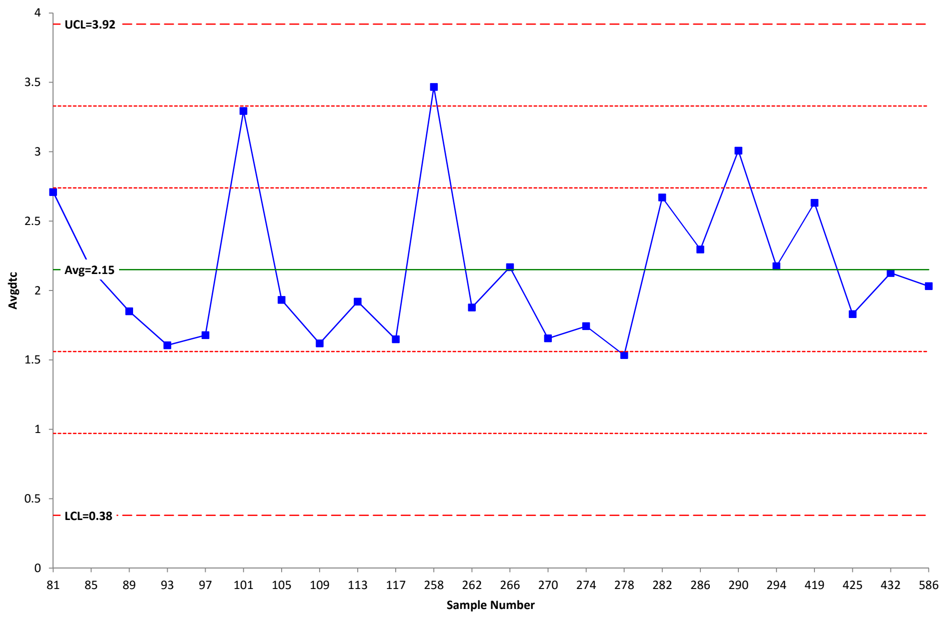


FIGURE 5.8: Parameter Chart of Laying Activity in User 19 Against 14

performed by setting the data of ADL session of one user as reference data for training the initial recognition model and another user data as the test data. The summary of the number of changes detected and the corresponding error rates in each class of activity across the different users is presented in Table 5.3 and 5.4 respectively. A correlation test was performed between the two tables using Minitab [2] statistical software. The result of the test is shown in test result 5.3.2.

Statistical Test Result 5.3.2. **Correlation: Number of Changes** *vs* **Error Rate in Opportunity Dataset**

Pearson correlation of Number of Changes and Error Rate = 0.972 $P - Value = 0.000$

The result indicates that the Pearson correlation between error rates and the number of changes detected in each class of activity for this sample dataset is 0.972 while the p-value is 0.000. Since the $p-value$ is smaller than 0.01, the null hypothesis about the correlation between the error rate and number of changes detected $H_0 = \rho = 0$ can be rejected in favour of the alternative hypothesis $H_0 = \rho > 0$. This implies that there is sufficient statistical evidence at the $\alpha = 0.01$ level to conclude that there is a significant relationship between the error rate and the number of changes detected in each class of activity. The implication is that the

		Number of Changes Detected Per Class of Activity			
Reference Data	**Test data**	**Standing**	**Walking**	**Lying**	**Sitting**
S1ADL1	S2ADL1	3	5	2	3
S2ADL2	S2ADL2	3	4	1	1
S3ADL3	S3ADL3	3	3	1	1
S4ADL4	S4ADL4	2	3	1	1
S1ADL5	S2ADL5	2	4	1	1
S1DRILL	S2DRILL	3	2	1	1
S3ADL1	S4ADL1	3	3	1	5
S3ADL2	S4ADL2	3	3	1	5
S3ADL3	S4ADL3	3	4	1	5
S3ADL4	S4ADL4	4	2	1	3
S3ADL5	S4ADL5	3	5	1	5
S3DRILL	S4DRILL	2	3	2	1
S3ADL2	S1ADL2	6	5	5	0

TABLE 5.3: Summary of Detections in Opportunity Dataset

higher the error rate in a given class of activity window, the more the number of changes detected. The change point indicates the points where instances are being misclassified that leads to error of the model and wide inconsistencies in the data classified to a class.

Sample charts of the change points in the data of one of the experiments are presented in Figures 5.9 to 5.12. The experiment was performed by setting the data of ADL session 2 of user 3 as reference training dataset while the combination of this data and that of ADL session 2 of ser 1 were set as test data. The first 27825 samples of the testing data belong to ADL session 1 of user 4 while the remaining 32224 data points of the total 60049 belongs to ADL session 1 of user 1. Figure 5.9 shows the chart of the computed window summary statistic (*avgdtc*) obtained from the window designated for 'Standing' activity.

		Error Rate Per Class of Activity			
Reference Data	**Test data**	**Standing**	**Walking**	**Lying**	**Sitting**
S1ADL1	S2ADL1	0.12	0.37	0.13	0.22
S2ADL2	S2ADL2	0.11	0.26	0.04	0.03
S3ADL3	S3ADL3	0.10	0.24	0.00	0.01
S4ADL4	S4ADL4	0.09	0.24	0.02	0.01
S1ADL5	S2ADL5	0.06	0.28	0.06	0.06
S1DRILL	S2DRILL	0.14	0.11	0.01	0.00
S3ADL1	S4ADL1	0.13	0.19	0.03	0.39
S3ADL2	S4ADL2	0.15	0.19	0.04	0.38
S3ADL3	S4ADL3	0.18	0.27	0.02	0.42
S3ADL4	S4ADL4	0.21	0.13	0.02	0.27
S3ADL5	S4ADL5	0.16	0.37	0.02	0.38
S3DRILL	S4DRILL	0.08	0.23	0.13	0.00
S3ADL2	S1ADL2	0.45	0.36	0.40	0.00

TABLE 5.4: Error Rates in Opportunity Dataset

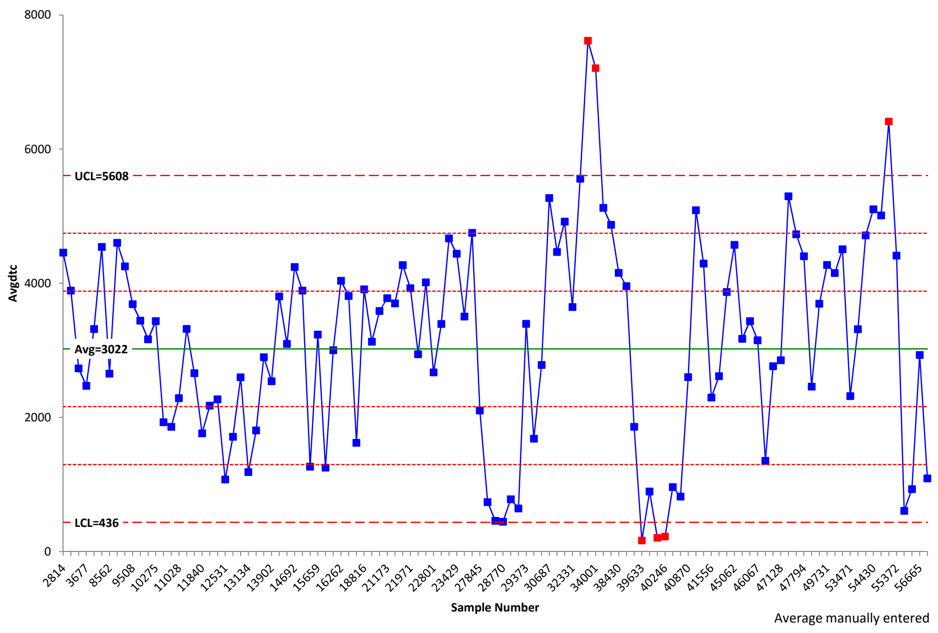


FIGURE 5.9: Parameter Chart of 'Standing' Activity in User 3 ADL 2 Against User1 ADL2

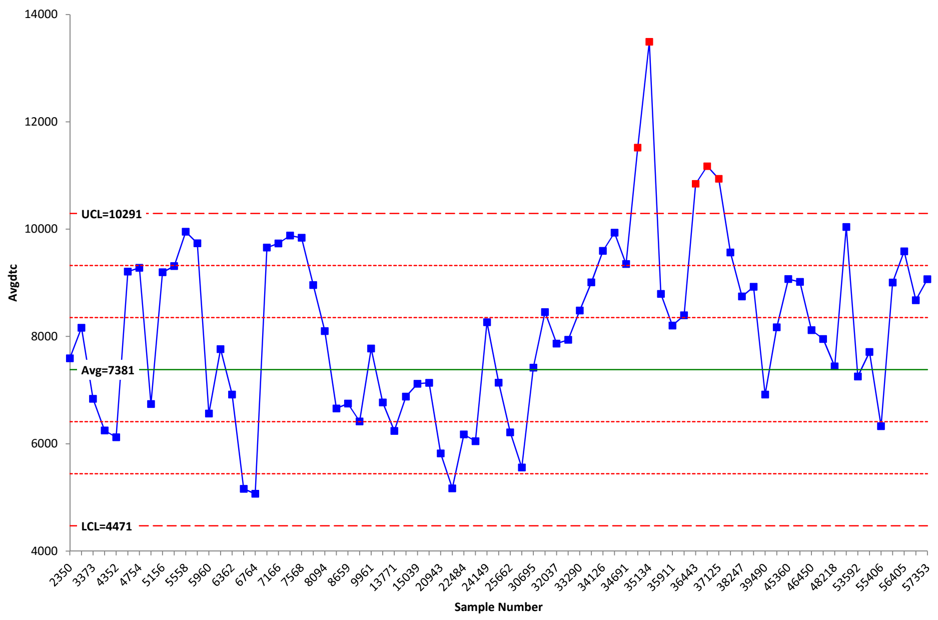


FIGURE 5.10: Parameter Chart of 'Walking' Activity in User 3 ADL 2 Against User1 ADL2

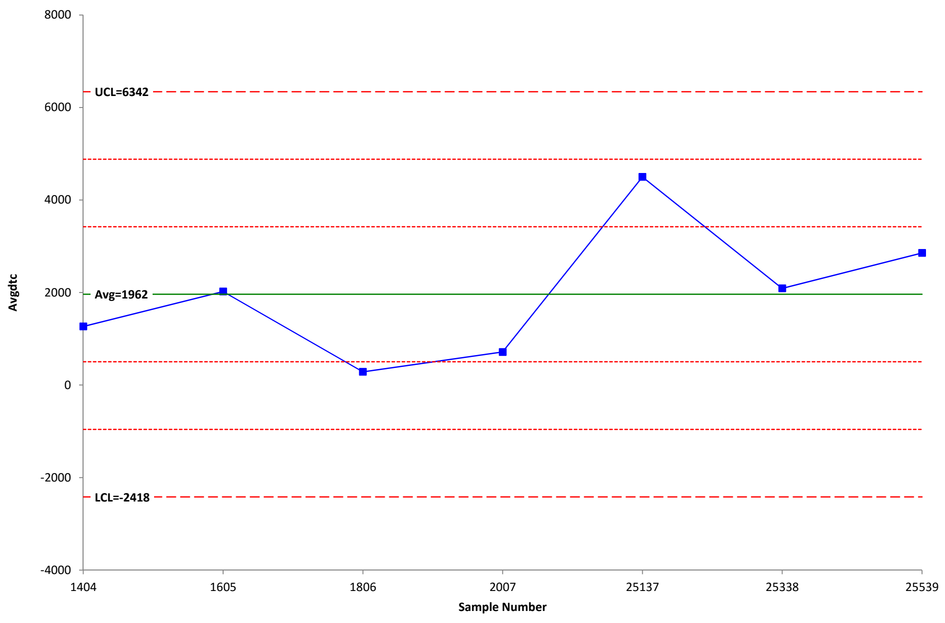


FIGURE 5.12: Parameter Chart of 'Sitting' Activity in User 3 ADL 2 Against User1 ADL2

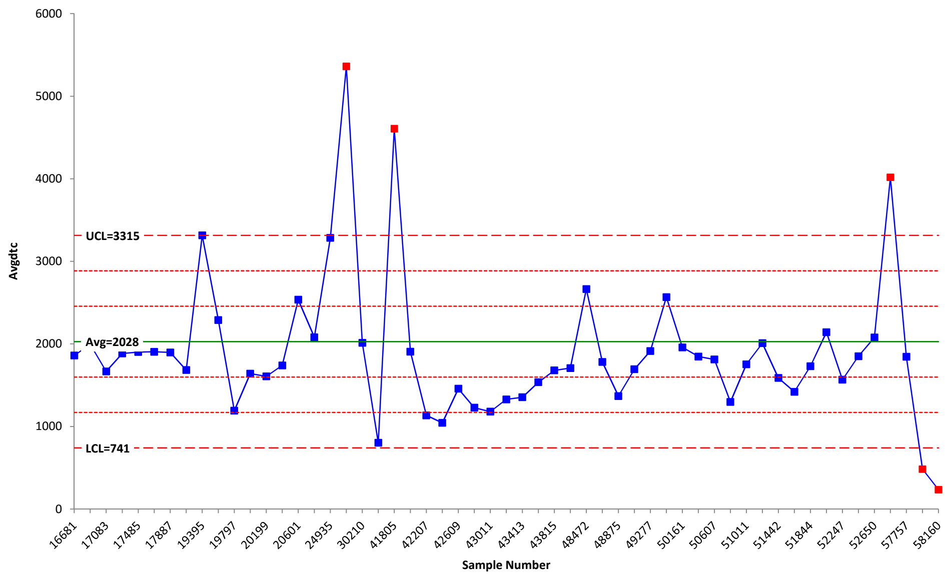


FIGURE 5.11: Parameter Chart of 'Lying' Activity in User 3 ADL 2 Against User1 ADL2

The figure shows that changes are detected at the data points 33520, 39633, 40045, 40246 and 54978. These are the points where the window summary statistic values (*avgdtc*) exceed either the upper or lower control limits. These points indicate where the data distribution changes from the initial reference data distribution and the deterioration in the accuracy of the underlying classification model. It also indicates that the samples around these time steps are misclassified which makes their computed statistic goes out of control limits. The more the out of control points the more the proportions of the misclassified samples that are classified into this window.

Similarly, changes are detected in the activity class ‘Walking’, and ‘Lying’ as shown in Figures 5.10 and 5.11 respectively. The change points are indicated by the out of control limit points in the individual chart and moving range chart of the change detection parameters. The changes detected show the variation in the activity of the initial user and the test user data. However, there is no change detected in the ‘Sitting ’ activity as shown in Figure 5.12. This is especially so as there is no data from the other test users data are classified to this window of activity. These results indicate the applicability and usefulness of the implemented method in detecting concept change point in activity recognition that involves diverse user characteristics and profiles.

5.3.3 Experiment with WISDM Dataset

This experiment used the Wireless Sensor Data Mining (WISDM) activity recognition dataset [44] that was described in Chapter 3. A batch size of 3 was utilised to compute the window summary statistic for all the experiments in this part. Each of the activity types has its own dedicated window for detecting the variability in the data classified to a given class. The number of changes detected and the error rates in each class of activity for different pairing of the users data are presented in Table 5.5 and 5.6. The result in these tables were subjected to Pearson correlation test to determine the relationship between the error rates and the number of changes detected. The result of the test is presented in test result 5.3.3.

Statistical Test Result 5.3.3. **Correlation: Number of Changes** *vs* **Error Rate in WISDM Dataset**

Pearson correlation of number of changes and error rate = 0.958 P-Value = 0.000

The result indicates that the Pearson correlation between error rates and the number of changes detected in each class of activity for this sample dataset is 0.958

while the p-value is 0.000. Since the $p-value$ is smaller than 0.01, the null hypothesis about the correlation between the error rate and number of changes detected $H_0 = \rho = 0$ can be rejected in favour of the alternative hypothesis $H_0 = \rho > 0$. This implies that there is sufficient statistical evidence at the $\alpha = 0.01$ level to conclude that there is a significant relationship between the error rate and the number of changes detected in a class of activity.

The charts of the *avgdtc* statistic values obtained from the window of each activity against the sample numbers are presented in Figures 5.13 to 5.18. The charts in were obtained by setting the data of the users with IDs 5 and 6 as reference data while the combination of users 5&6 and 19&20 were used as test data.

Figure 5.13 shows the chart of the sequence of the individual window summary statistic (*avgdtc*) obtained from the window of activity class 'walking'. A change is detected at the time step 518 *and* 530 on the chart. These are the points where the sample statistic goes out of the upper control limit. It implies that the distribution of the data has changed from the initial data distribution and points to the deterioration in the accuracy of the model. Table 5.6 shows the error rates indicating the proportions of misclassification for each class. It can be observed that there is a classification error in this class. Similarly, changes are detected in the activity class 'Walking-Upstairs', 'Sitting' and 'Standing' shown in Figures 5.15, 5.17 and 5.18 respectively. There are no changes detected in the jogging and walking-downstairs activities as shown in Figures 5.14 and 5.16 respectively which is corroborated by the absence of any misclassification and invariably zero error rates in these class of activities for this experiment as shown in the 9th entry of Table 5.6. The approach is able to detect changes between the distribution of the initial user data and those that comes from another user.

5.3.4 Comparative Evaluation of UDetect with Adaptive Windowing Technique (ADWIN)

The experiments here perform a comparative evaluation between the UDetect technique and adaptive windowing (ADWIN)[14] detection method. ADWIN relies on

		Amount of Change Points per Class of Activity					
Reference Data	**Test data**	**Walking**	**Jogging**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**
User3&1	User5&2	2	3	4	8	5	3
User7&8	User12&9	3	1	6	5	2	1
User19&14	User20&15	3	1	5	8	9	2
User27&22	User29&23	4	3	6	6	5	2
User13&10	User18&11	4	3	3	8	2	4
User6&4	User36&35	5	3	5	9	4	2
User21&16	User24&17	5	5	5	6	4	2
User33&28	User34&30	6	1	7	6	4	5
User32&26	User31&25	5	1	3	7	3	6
User5&6	User19&20	2	0	3	0	2	1

Table 5.5: Summary of Detections in WISDM Dataset

		Error rate per Class of Activity					
Reference Data	**Test data**	**Walking**	**Jogging**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**
User3&1	User5&2	0.10	0.25	0.33	0.69	0.41	0.23
User7&8	User12&9	0.15	0.01	0.44	0.43	0.13	0.03
User19&14	User20&15	0.14	0.00	0.36	0.70	0.84	0.11
User27&22	User29&23	0.23	0.25	0.47	0.70	0.38	0.12
User13&10	User18&11	0.19	0.26	0.28	0.75	0.07	0.28
User6&4	User36&35	0.38	0.17	0.38	0.84	0.35	0.10
User21&16	User24&17	0.35	0.45	0.34	0.57	0.15	0.07
User33&28	User34&30	0.43	0.01	0.58	0.56	0.33	0.42
User32&26	User31&25	0.37	0.04	0.16	0.77	0.22	0.50
User5&6	User19&20	0.18	0.00	0.34	0.00	0.20	0.12

Table 5.6: Error Rates in WISDM Dataset

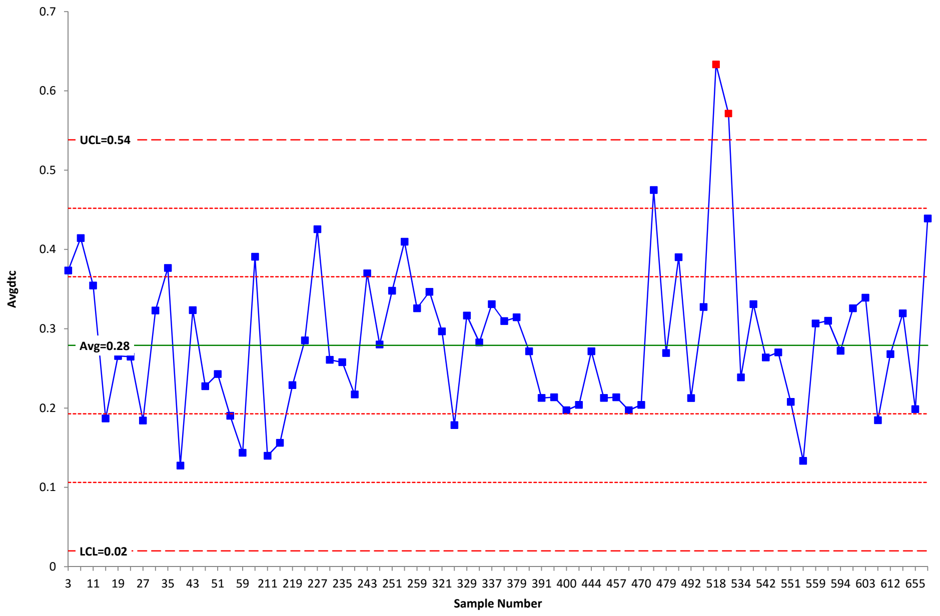


Figure 5.13: Parameter Chart of Walking Activity in User 5&6 Against 19&20

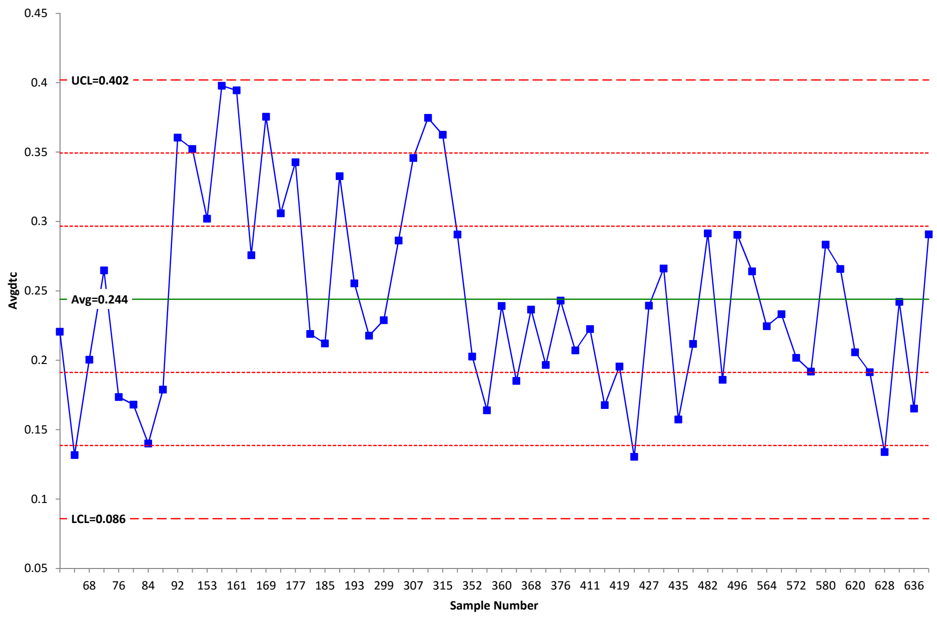


FIGURE 5.14: Parameter Chart of Jogging Activity in User 5&6 Against 19&20

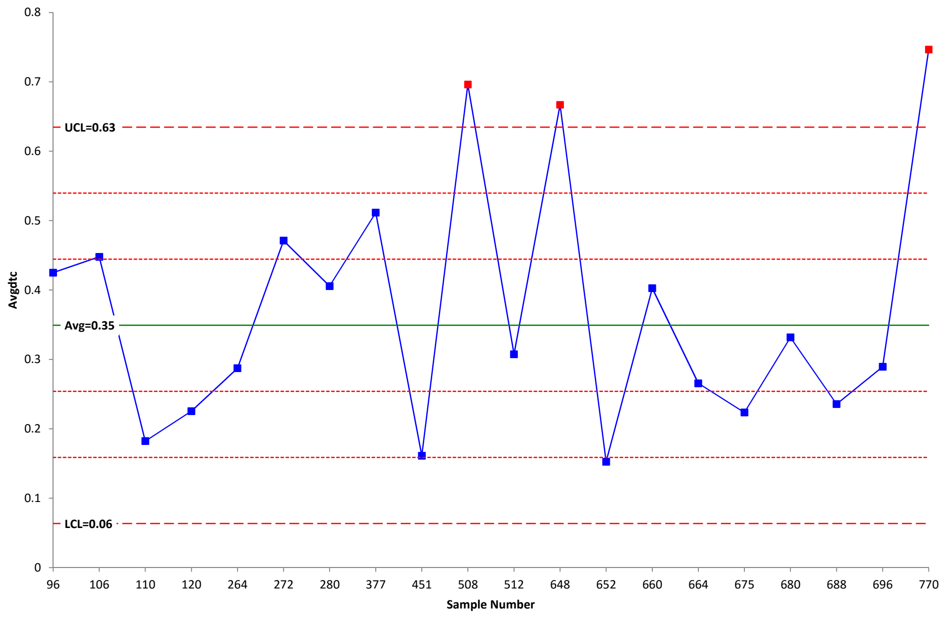


FIGURE 5.15: Parameter Chart of Walking-Upstairs Activity in User 5&6 Against 19&20

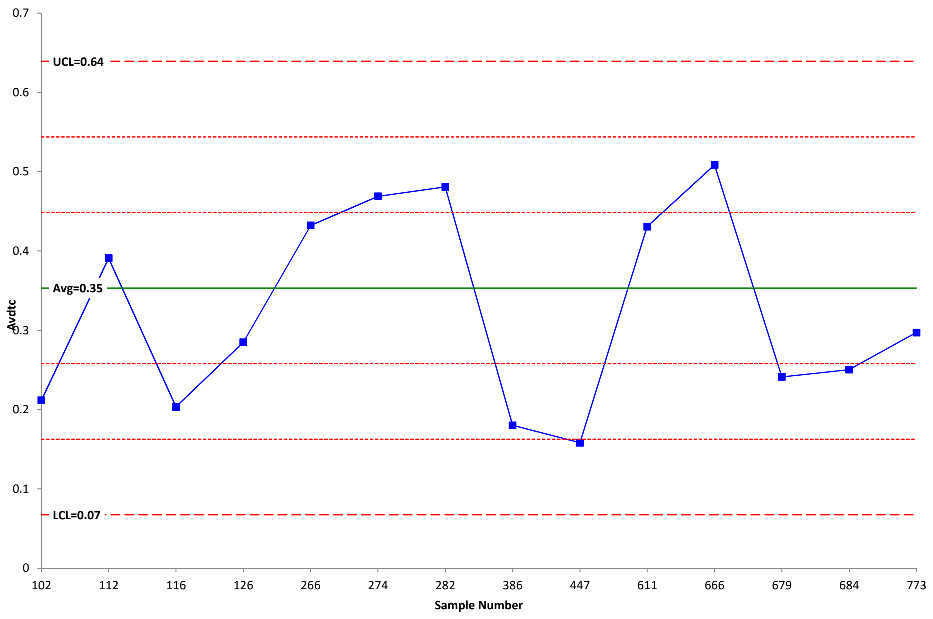


Figure 5.16: Parameter Chart of Walking-Downstairs Activity in User 5&6 Against 19&20

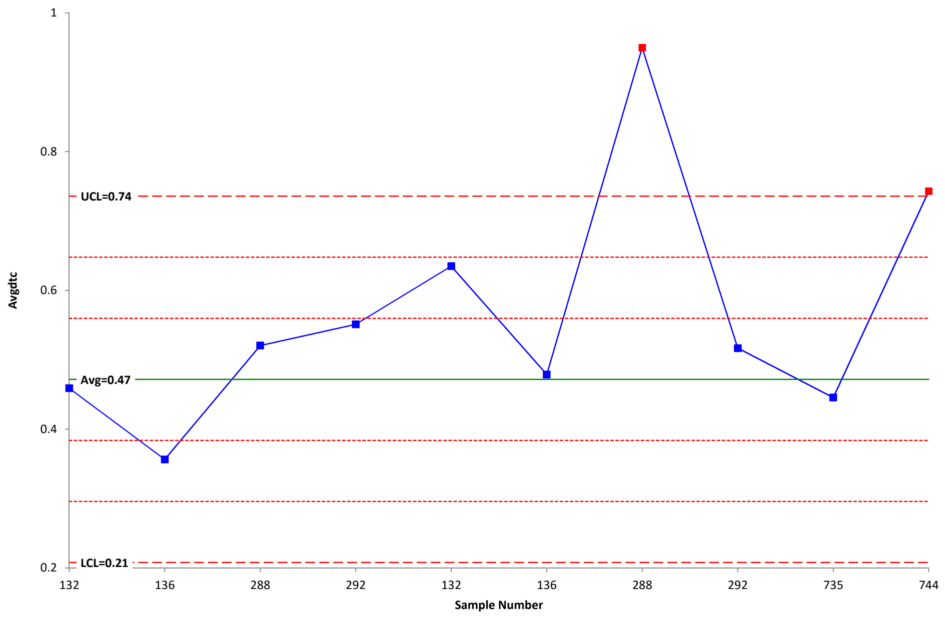


Figure 5.17: Parameter Chart of Sitting Activity in User 5&6 Against 19&20

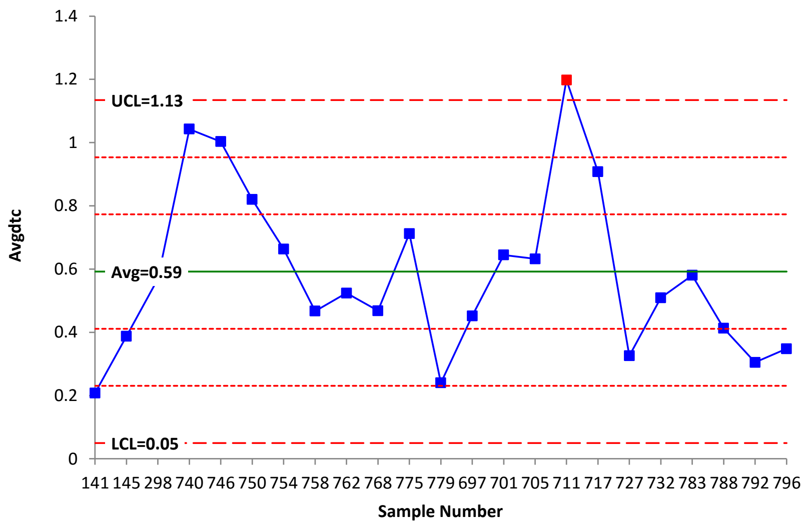


FIGURE 5.18: Parameter Chart of Standing Activity in User 5&6 Against 19&20

the assumption of the presence of ground truth label for each data sample that is being classified by a classifier. Based on this, the error rate of the model is used as input to the ADWIN window to determine the occurrence of change in the underlying concept. In ADWIN, a window W containing the error rate from the predictions of a learning model grows until there has been a change in the average value inside the window. Therefore, when the algorithm succeeds at finding two distinct sub-windows, the split point is considered as a concept change. UDetect on the other hand, does not use error rate because it does not assume the presence of ground truth to determine the classification error.

In other to compare these two methods, the initial accuracy of the KNN classifier was recorded for each data pair used in the previous section. Then, the two methods were deployed to detect changes in the model accuracy. The detected changes was subjected to active learning to provide the correct labels at the points where the changes was detected. After the provision of the true labels at the different points of detected changes, the correct labels was used for new classification and the accuracy of the model is then recomputed to determine the efficacy of the correction made as a result of the changes detected by the two methods (UDetect and ADWIN). The gains between the initial accuracy without change detection is then compared with the accuracy after providing correct labels during detections. The improvements to the classifier under the two change detectors with active learning intervention for each of the dataset are presented in Tables 5.7, 5.8 and 5.9 for the Opportunity, WISDM and HARS dataset respectively.

Reference Data	**Test Data**	**Accuracy Before (%)**	**Accuracy After UDetect (%)**	**Accuracy After ADWIN (%)**	**UDetect Gain (%)**	**ADWIN Gain (%)**
S1ADL1	S2ADL1	63.57	96.61	90.53	33.04	26.96
S1ADL2	S2ADL2	73.52	96.07	91.58	22.55	18.07
S1ADL3	S2ADL3	75.59	95.91	93.57	20.32	17.98
S1ADL4	S2ADL4	76.49	97.18	90.86	20.68	14.37
S1ADL5	S2ADL5	76.44	95.63	90.26	19.19	13.81
S1DRILL	S2DRILL	74.13	95.13	93.08	21.00	18.95
S3ADL1	S4ADL1	69.05	97.86	92.88	28.82	23.83
S3ADL2	S4ADL2	61.34	97.32	91.73	35.98	30.39
S3ADL3	S4ADL3	58.82	96.25	92.75	37.43	33.93
S3ADL4	S4ADL4	50.62	96.21	93.19	45.60	42.58
S3ADL5	S4ADL5	58.94	95.52	92.56	36.58	33.62
S3DRILL	S4DRILL	54.71	95.50	91.32	40.79	36.61
S3ADL2	S1ADL2	68.30	96.56	91.27	28.27	22.98
				Mean	**30.02**	**25.70**

Table 5.7: Comparative Evaluation of UDetect Against ADWIN in Opportunity Dataset

Reference Data	**Test Data**	**Accuracy Before (%)**	**Accuracy After UDetect (%)**	**Accuracy After ADWIN (%)**	**UDetect Gain (%)**	**ADWIN Gain (%)**
User3&1	User5&2	54.17	93.62	90.77	39.45	36.60
User7&8	User12&9	67.24	94.18	90.94	26.94	23.70
User19&14	User20&15	56.97	95.11	90.43	38.15	33.47
User27&22	User29&23	46.71	93.67	90.39	46.96	43.68
User13&10	User18&11	50.93	93.27	90.72	42.33	39.79
User6&4	User36&35	52.54	93.92	91.57	41.38	39.03
User21&16	User24&17	32.38	95.88	91.36	63.50	58.98
User33&28	User34&30	29.17	93.97	91.78	64.81	62.61
User32&26	User31&25	52.25	93.58	91.42	41.33	39.17
User5&6	User19&20	69.92	93.51	90.83	23.59	20.91
				Mean	**42.84**	**39.79**

Table 5.8: Comparative Evaluation of UDetect Against ADWIN in WISDM Dataset

The results for the HARS dataset show that the average gain in accuracy of the classifier under UDetect is better than that of ADWIN. The results were further subjected to paired t-test to determine whether to accept or reject the null hypothesis that there is no difference in the performance of the two methods. $H_0 = \mu_{ADWIN} - \mu_{UDetect} = 0$ $i.e$ $\mu_d = 0$ The result of the paired t-test for this dataset is presented in statistical test result 5.3.4.

Reference User Data	**Test Data**	**Accuracy Before (%)**	**Accuracy After UDetect (%)**	**Accuracy After ADWIN (%)**	**UDetect Gain (%)**	**ADWIN Gain (%)**
User19	User14	43.03	95.79	92.49	52.76	49.46
User30	User2	56.95	93.22	92.05	36.27	35.09
User3	User4	81.07	92.45	91.32	11.38	10.25
User5	User6	85.54	94.12	91.40	8.59	5.86
User7	User8	72.24	92.30	91.14	20.05	18.90
User9	User10	74.15	95.11	92.54	20.96	18.39
User11	User12	71.88	94.92	91.57	23.04	19.70
User13	User15	82.01	92.00	91.89	9.99	9.88
User16	User17	82.88	95.45	92.21	12.57	9.32
User18	User20	63.33	93.42	92.47	30.08	29.13
User21	User22	88.47	93.37	90.27	4.90	1.79
User23	User24	62.73	95.71	92.98	32.98	30.25
User25	User26	100.00	100.00	100.00	0.00	0.00
User27	User28	81.94	94.00	91.73	12.06	9.80
User29	User30	87.47	92.74	90.55	5.27	3.08
				Mean	**18.73**	**16.73**

TABLE 5.9: Comparative Evaluation of UDetect Against ADWIN in Opportunity Dataset

Statistical Test Result 5.3.4. **Paired T for UDetect Gain** *vs* **ADWIN Gain in HARS Dataset**

	N	Mean	StDev	SE Mean
UDetect Gain	15	18.73	14.3	3.69
ADWIN Gain	15	16.73	14.04	3.63
Difference	15	1.999	1.152	0.297

95% CI for mean difference: (1.361, 2.636)

T-Test of mean difference $= 0 (vs \neq 0)$: T-Value = 6.72 P-Value= 0.000

Looking at the statistical test result 5.3.4 it can be asserted that the confidence interval for the mean difference between the two methods does not include zero, which suggests a difference between them. The small p-value (p = 0.000) further suggests that the data are inconsistent with the null hypothesis $H_0 = \mu_{ADWIN} - \mu_{UDetect} = 0$ *i.e* $\mu_d = 0$, that is, the two methods do not perform equally. Specifically, UDetect approach ($mean = 18.73$) performed better than ADWIN ($mean = 16.73$) in terms of using their change detection to initiate adaptation by active learning. A similar test was performed on the results from WISDM and HARS dataset and the statistical test results obtained are presented in statistical test results 5.3.5 and 5.3.6.

Statistical Test Result 5.3.5. **Paired T for UDetect Gain** *vs* **ADWIN Gain in WISDM Dataset**

	N	Mean	StDev	SE Mean
UDetect Gain	10	42.84	13.28	4.20
ADWIN Gain	10	39.79	13.21	4.18
Difference	10	3.049	0.905	0.286

95% CI for mean difference: (2.402, 3.697)

T-Test of mean difference = $0(vs \neq 0)$: T-Value= 10.66 P-Value = 0.000

The test results for WISDM and Opportunity datasets indicate that the confidence interval for the mean difference between the two methods does not include zero, which suggests there is a difference between the mean of UDetect and ADWIN in the two datasets. The small p-value ($p = 0.000$) in both cases also suggests that the data are inconsistent with the null hypothesis $H_0 : \mu_{ADWIN} - \mu_{UDetect} = 0 \; i.e \; \mu_d = 0$, that is, the two methods do not perform equally. Thus, the null hypothesis can be rejected at a confidence level $\alpha = 0.01$ in favour of alternative hypothesis that $\mu_{UDetect} > \mu_{ADWIN}$ in both datasets. Specifically, UDetect approach ($mean = 30.02$) performed better than ADWIN ($mean = 25.70$) for Opportunity dataset in terms of using their change detection to initiate adaptation by active learning. In the same vein, UDetect approach ($mean = 42.84$) performed better than ADWIN ($mean = 39.79$) for WISDM dataset. It can thus be concluded that UDetect change detection performance is better than ADWIN under the circumstances of the experiment scenarios.

Statistical Test Result 5.3.6. **Paired T-test for UDetect Gain** *vs* **ADWIN Gain in Opportunity Dataset**

	N	Mean	StDev	SE Mean
UDetect Gain	13	30.02	8.86	2.46
ADWIN Gain	13	25.7	9.14	2.53
Difference	13	4.321	1.428	0.396

95% CI for mean difference:$(3.458, 5.183)$

T-Test of mean difference = 0 (vs $\neq$ 0): $T - Value = 10.91$ $P - Value = 0.000$

5.4 Chapter Summary

This chapter has presented a new change detection method for activity recognition. The method is based on extracting change parameters from the reference dataset and training the initial model with the dataset during the offline stage. In the online stage, new incoming unseen data are processed in chunks from the window of data classified to the same class and a statistical summary called *avgdtc* that characterised each chunk is computed. The average distance to centre statistic computed from each batch is monitored by using the pre-computed UCL and LCL parameters from the reference data of the associated class. Points that exceed the control limits are the change points and correspond to the point where the classifier performance degrades.

The change points that indicate where the model is misclassifying the samples to the wrong class can be used to initiate the adaptation step thereby improving the accuracy of the classifier. The main benefit of this method compared to the traditional drift detection approach in data stream domain is that it does not rely on the ground truth to detect drift in the data and thus is the more realistic approach for activity recognition.

The method is evaluated using real activity recognition datasets obtained from mobile phones of diverse subjects and another large dataset that is obtained from more complex inertial sensors attached to users who perform the designated activities. The results indicate that the method is able to identify the precise drift point in the data. Also, comparison of the method with ADWIN approach reveals that UDetect performance is better than the ADWIN method.

Chapter 6

Distance Based Change Detection Technique for Activity Recognition

6.1 Introduction

This chapter presents another novel technique for concept drift detection in activity recognition. The problem of concept drift in activity recognition is a very challenging and important task that requires an effective solution that is robust to the multi-dimensional and multi-sensor nature of the activity recognition data The previous chapter exploits the problem of change detection from the perspective of the unidimensional stream by converting the multi-dimensional block of data to the unidimensional summary that serves as input to the change detector. While this approach is advantageous in its ability to employ many existing unidimensional change detector methods, it does not leverage or exploits the peculiarities of the original nature of the multi-dimensional sensors rich activity data.

The chapter details the newly developed technique for change detection in activity recognition. It also presents the results of the evaluation and the performance of the method on three real-life activity recognition datasets. The developed technique implements an ensemble change detection approach that applies a change detection test on the multidimensional data directly thereby yielding a high detection accuracy.

The rest of this chapter is organised as follows: Section 6.2 discusses the overall bird eye view of the method. In Section 6.3 details conceptual framework of the detection methods, highlighting the key architectural and functional components that form the realisation of the novel technique is presented. The last two sections, Section 6.4 and Section 6.4.1 detail the experimental study and results obtained with discussions.

6.2 Ensemble Detector Overview

The approach for concept change detection presented here is based on distance similarities between the refrence data characteristics and the new classified sample. The approach employs the n-dimensional attributes of the data to compute the distance similarity. This results in allowing each feature to be considered in the similarity computation.The curse of dimensionality is eliminated by ensuring that most informative features of the sample are used in the comparison computation.

The key components of the approach are shown in Figure 6.1. The realisation of the components is in a two-phase implementation framework. In the first phase, the base classification framework presented in Chapter 5 is used to create a set of reduced reference dataset that serves as reference data for the ensemble change detection. The reference dataset represents various patterns that exist in each of the class of activity. The patterns are indirectly encapsulated in the set of cluster characteristics extracted from the training samples. In the second phase dubbed the online phase, the system monitors the classification performance of a recognition model by identifying misclassified samples through distance similarity between the samples and the reference holistic dataset.

The assumption behind the comparison is that if the new incoming data is consistent with the holistic reference data, then there is no change. Otherwise, the change is declared and the process of adapting the model to the new change can be initiated. The change detection problem is formulated as testing the similarities

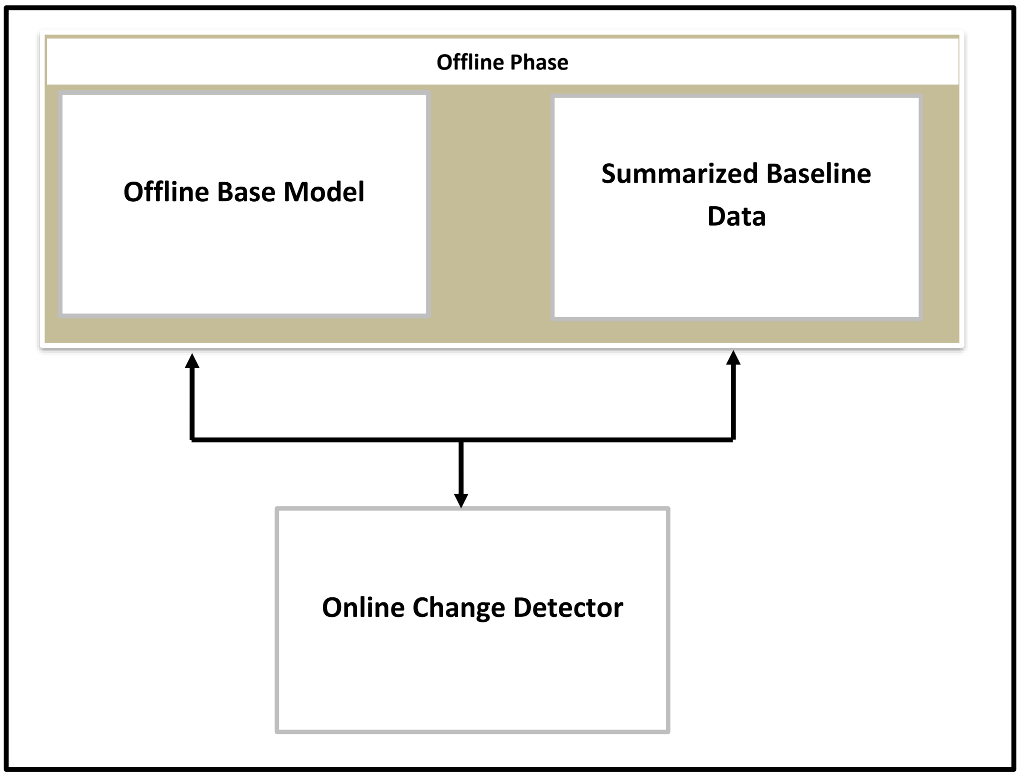


Figure 6.1: Overview of Components of Ensemble Detector

between the reference datasets and the new set of incoming classified samples to see if there is consistency between them or not. Given the reference cluster characteristics obtained from the training dataset as $X_{ref} = \{(X^1, Y_i), (X^2, Y_i)X^3, Y_i\}$ and a new classified chunk or a single sample x_k

$$\begin{cases} change, & \text{if } d_{measure}(X_{ref}, x_k) \geq threshold \\ \neg change, & \text{otherwise} \end{cases} \tag{6.1}$$

The template of the detection framework is given in Algorithm 9. The algorithm shows the main steps involved in the detection framework.

Algorithm 9 gives the overview of the approach. The high-level description of the detection algorithm is explicated and elaborated upon further in the following sections.

Algorithm 9: Template Change Detection

```
1  Build the base classifier model
2  $X$ = Summaries from holistic baseline data
3  foreach instance $x_k$ classified to a stream of window activity do
4      $x_{form} = windowStep()$
       // Form a data chunk from the window
5      $averageDistance = d_{measure}(x_{form}, X)$
       // Compute distances between incoming window summary and baseline summary
6      if (averageDistance < threshold) then
7          No change
8      end
9      else
10         change detected
11         initiate adaptation
12     end
13 end
```

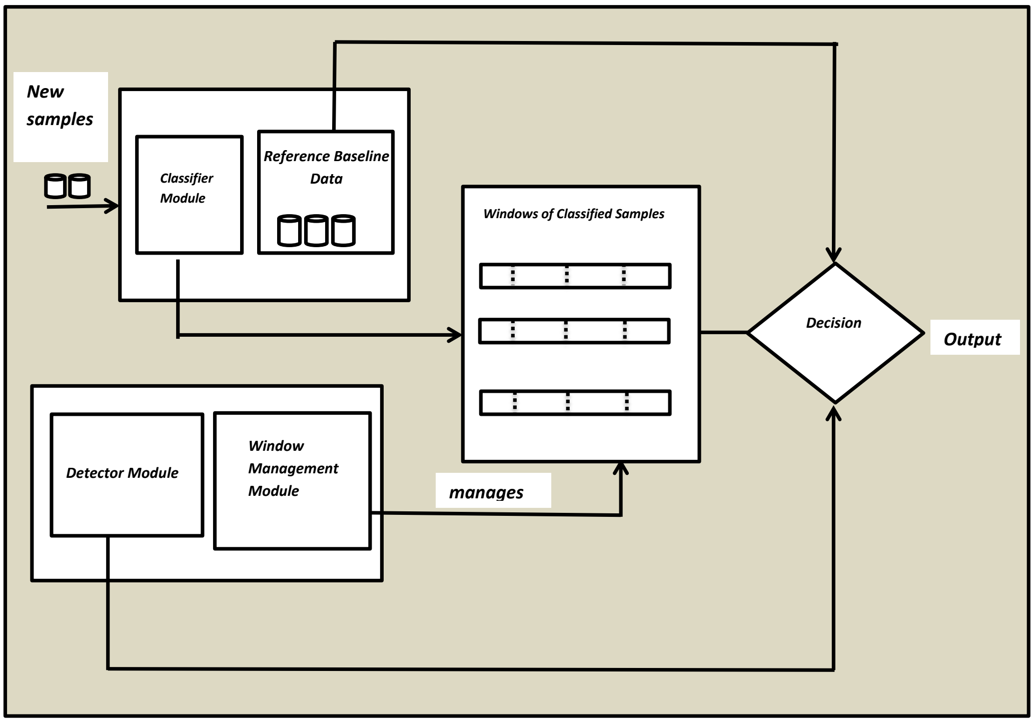


FIGURE 6.2: Main Components of Ensemble Detector

6.3 Conceptual Framework of the Detection Scheme

The ensemble change detection framework components is depicted in Figure 6.2. The main components are: (i) Classifier Module and Reference Baseline Data (ii)Ensemble Detector (iii)Window Management and Segmentation

6.3.1 Classifier Module and Reference Baseline Data

This component is responsible for taking new example of activity and classified them to the appropriate section of the window that is dedicated to each class of activity. The classifier module which could incorporate any classification algorithm to decide on the appropriate class to which an example should be classified. The reference baseline data, on the other hand, are refinement and reduced form of the original training set that was formulated during the off-line phase of the algorithm as described in Chapter 5. Both elements are integrated into the online detection framework, which utilised them in an efficient manner to decides if the new data is consistent with the reference baseline data or not. If there is a consistency between them, it means there is no change and the classifier is still consistent with the original view of the activity. Otherwise, it means there is no consistency between the reference data and the new set of data. In this case, the classifier needs to be updated to reflect the regime in the activity recognition context that resulted to the concept change.

6.3.2 Ensemble Detector Framework

The ensemble detector is a multi-view and multi-resolution change detection approach that combines the decisions of three individual change detectors to achieve a high-level accuracy and precision in its change detection. The individual detector is implemented by using the main scheme of the Algorithm 9.

In the beginning, the module extracts set of characteristic data from the original training set. This is achieved by performing clustering on each class of activity data to derive various patterns in the particular activity class. The definition of the extracted characteristics is given formally in Definition 6.3.1.

Definition 6.3.1. **Cluster Characteristics:**

Given a set of training examples $X_1, \ , X_2, \ X_3, \ ...X_n$ with corresponding labels $y_1, \ , y_2, \ y_3, \ ...y_n$ with $y_i \in 1...c$ classes of activities. For every class c there is a formation of optimal k sub-clusters with $d - dimensional$ data points that represents the various patterns in the class. The three data characteristics obtained from each sub-cluster with m instances $\in R^d$ consist $C_{center}\, mn\, and\, mx$ each $\in R^d$. where:

$C_{center} = \frac{\sum_{i=1}^{m} X_i}{m}$

$mn = min(X^j_{i...m}) \ \forall \ j = 1...d$

$mx = max(X^j_{i..m}) \ \forall \ j = 1...d$

Once these data summaries are in place, a classifier such as Naive Bayes or KNN is used to classify new samples which, are kept in separate windows dedicated to each type of activity. The window for each activity data is kept under the control of window management module described in 6.3.3 to decide on how the window of data are formed. The two repositories comprising the new classified sample (x_{form}) and the extracted characteristics (minimum (mn), maximum (mx), and centroid (C_{center}) vectors) from each class of activity in the reference data become the two inputs to the change detector. It should be noted that the data from the window (x_{form}) that will be used as a comparison to the reference data can be formed in different ways depending on the scheme adopted. The scheme adopted here is to compare each new instance classified to the window with the reference data. Alternatively, a fixed amount of data from the window can be used for distance similarity comparison with the reference data.

The main similarity function adopted here involves computing the Euclidean distance between the new classified instance and the set of characteristics. In essence, for a particular characteristic in d- dimensional space with n samples, the average distance between the new instance and the n samples are computed. If this value is greater than a threshold then a change is detected with respect to this characteristic otherwise no change is detected. This routine is repeated for each of the *min max and center* characteristics as illustrated in Algorithms 10, 11 and

12 respectively.

Algorithm 10: minimumViewDetector

```
Input: $minVectors = \{z^{(1)}, z^{(2)}, ..., z^{(n)}\}$
// set of minimum vector charateristics
1  $k = setWindowSize()$
2  $avgDistance_{min} = 0$
3  while $Data\ Stream\ != null$ do
4      $x_{form} = windowFormation(k)$
       // Form a window size k from the stream
5      $x_{new} = \frac{x_{form}}{k}$
       // $x_{new}$ new sample to test for change
       // Form a window size k from the stream
6      $avgDistance_{min} = \frac{1}{n}\sum_{k=1}^{n} Distance(x_{new}, z^{(k)})$
7      if $avgDistance_{min} < threshold_{min}$ then
8          $changeSignal = change$
9      end
10     else
11         $changeSignal = \neg change$
12     end
13 end
```

To detect change from the perspective of the minimum characteristic, the average distance between a new classified instance to a window of activity type and each of the element of minimum characteristic data for that class is computed according to Algorithm 10 *line 6*. Then, this average is compared with the pre-determined threshold and if the condition of change is met, a change is declared from the perspective of the minimum characteristic (Algorithm 10 lines 7-9).

Similarly, the detection from the maximum and centroid characteristics is based on computing the average distance between a new classified sample to a window of activity type and the maximum or centroid data obtained from the training dataset of that class accordingly. The average is compared with the pre-determined change threshold using Algorithms 11 and 12 respectively.

Algorithm 11: maximumViewDetector

```
Input: $maxVectors = \{max^1, max^2, ..., max^n\}$
// set of maximum vector charateristics
1  $avgDistance_{max} = 0$
2  while $Data\ Stream\ != null$ do
3      $x_{form} = windowFormation(k)$
       // Form a window of size k from the stream
4      $x_{new} = \frac{x_{form}}{k}$
       // $x_{new}$ new sample to test for change
5      $avgDistance_{max} = \frac{1}{n}\sum_{i=1}^{n} Distance(x_{new}, max^{(i)})$
6      if $avgDistance_{max} < threshold_{max}$ then
7          $changeSignal = change$
8      end
9      else
10         $changeSignal = \neg change$
11     end
12 end
```

Algorithm 12: *centerViewDetector*

```
Input: $centroidVectors = \{c^1, c^2, ..., c^n\}$
// set of centroid vector charateristics
1  $avgDistance_{center} = 0$
2  while $Data\ Stream\ != null$ do
3      $x_{form} = windowFormation(k)$
       // Form a window of size k from the stream
4      $x_{new} = \frac{x_{form}}{k}$
       // $x_{new}$ new sample to test for change
5      $avgDistance_{center} = \frac{1}{n}\sum_{i=1}^{n} Distance(x_{new}, c^{(i)})$
6      if $avgDistance_{center} < threshold_{center}$ then
7          $changeSignal = change$
8      end
9      else
10         $changeSignal = \neg change$
11     end
12 end
```

To enhance the confidence level of the change detected, the decisions from the perspective of each characteristic are combined to arrive at a final change signal. From the experiment conducted a majority decisions shows adequate performance in the detection rate. The ensemble algorithm is depicted in Algorithm 13. According to the Algorithm 13, a change is declared only if two or more detectors agree on the decision otherwise a no change is declared.

Algorithm 13: *EnsembleDetector*

```
1  while true do
2      $decision_{min} = minmumViewDetector()$
3      $decision_{max} = minmumViewDetector()$
4      $decision_{center} = centerViewDetector()$
5      $changeSignal = majorityVote(decision_{min}\, decision_{max}\, decision_{center})$
6      if $changeSignal$ then
7          detect change
8      end
9      else
10         No change
11     end
12 end
```

Threshold Parameters

In other to the detect the change points, the threshold parameters are very important as they determine the sensitivity of the detection module. The threshold used in this case is the maximum inter-micro-cluster distance of each the characteristic. The threshold parameter is computed for each class of activity data characteristic.

$$\delta = max(d(C_i^j, Cj^i)) + \rho \tag{6.2}$$

Equation 6.2 is the threshold employed in this method to determine whether the new points classified to a class cohesively mesh within the baseline data of that class. The data can be nearest to a class but not validly and cohesively belong to the class due to the drifting concept emanating from user dissimilarity or sensor misalignment and displacement. The additive control parameter ρ in the threshold equation allowed the method to be adapted to outliers and regularised the space of the allowable drift in the detection regime within a scope of time interval t_i and t_j during which the model stability is experienced. Experimentally a value of 0.01 proofs to perform well for the accurate detection of the drift in the model.

6.3.3 Window Management

The window management component is a very important component of the framework. Its primary responsibility is to regulate the flow of the data stream. It also caters for segmenting and adapting the window segment sizes to be used for the detection test.

The formation of the data to be used for detection comparison can fall under the various schemes of windowing. The first approach is the sequential sample approach where each sample is a potential candidate to be used in detecting a change. Other alternatives include using a fixed chunk of data that arrives in the window as a unit and compute the characteristic to be used for the basis of detection. The formation of the chunk of data can be in the form of an overlapping window where a fixed amount of data is formed at time t_i and the next data chunk to be formed at time t_j, $(t_j > t_i)$ is composed of fixed points from the previous points. Another approach is to use adjacent window model where there is no overlapped between data used at time t_i and another data at time t_j $(\forall\, t_j > t_i)$.

In the case of adjacent window, the data used are independent and fthere is no reference to the previous data. The overlapping window is useful in a situation whereby there is a dependency among the data in the stream and also where the change being detected is within the confine of the data stream with no reference dataset to compare with. The latter on the other hand is suitable where there is a separate reference data for the data stream. The $windowFormation()$ method in the Algorithms 12 11 and 10 implements the functionality of the window management and caters for the size of data that is passed from the window to the detection module.

The method presented here is not fixed to a particular adaptation module. The aim is to produce a concept change detection that can be used for the basis of any adaptation technique. Once a change is flagged in a class of activity, the adaptation process can be initiated to correct the abnormality that leads to the drift. The advantage of the approach is that it allow a fine-grained recognition of which activity is affected so that the adaptation can be localised to the activity instead of adapting the unaffected activity class. One commonly used approach of adaptation is the active learning approach which employs the user in the loop mechanism to query the user for the correct activity that is being performed at the particular point in time when the change is detected. The corrected instances are added to update the classifier model and adapt to the current situation of the user.

6.4 Experimental Evaluation

The evaluation of the approach presented in this chapter is carried out using three publicly available activity recognition datasets. The datasets comprise of the Opportunity, WISDM and HARS datasets. The details about the datasets are given in Chapter 3.

6.4.1 Experimental Setup

The experiment was conducted in two modes of evaluation. The first mode of the experiment is called cross user detection mode in which a user data is used for generating baseline characteristics data and trained a classifier that recognises new activity from other users. The purpose of the detector here is to detect when the classifier performance degrades in terms of how many instances are being misclassified. The detector attempts to detect the instances that are being misclassified into wrong activity class. The second mode of the experimental setup is focused on detecting totally new and unseen activity that are not part of the original set of existing activities. This mode is called leave-one-out detection experiment because one activity is left out of the existing activities and the detector is set to detect the left out activity in the midst of all the activity data.
Software: The implementation and evaluation of the approach was carried out using Anaconda python distribution [1]. This version of python distribution incorporates a number of open source packages for scientific and analytic data manipulation and computations. The included packages that are employed include numpy, scikit-learn, spyder and matplotlib. Other software packages used for analysing and processing the result include spreadsheet package and Minitab statistical analysis package [2].

Hardware: The hardware platform for the experiment was configured with intel corei7 processor with 8GB memory running Microsoft Windows 10 operating system.

6.4.2 Opportunity Dataset Experiment Results

The result of the experiment with this dataset is presented in this section. As stated earlier, there are two modes of the test experiments performed on the dataset. The cross user evaluation is performed first followed by the leave-one-out detection experimental evaluation. The next sub-sections present the details of the experiments and their results.

Cross User Detection Mode for Opportunity Dataset

The cross user experiment was performed with the Opportunity dataset by using disparate portions of the session of activity performed by one subject to generate the baseline characteristics data and build the learning model while another portion is used to test change detection method. The result of the experiment repeated for 10 different cases is presented in Table 6.1. The number of instances detected to be drifting away from the target class across each activity type is listed in the table including the total detected instances for each case of the experiment.

			Number of Changes Detected					
Experiment No.	**Reference Data**	**Test Data**	**Null**	**Stand**	**Walk**	**Lie**	**Sit**	**Total**
1	User1 ADL1	User2 ADL1	2487	4248	5614	2256	517	15122
2	User2 ADL2	User3ADL4	1882	2909	5220	712	2117	12839
3	User3 ADL3	User4 ADL5	3770	2830	4972	132	900	12604
4	User4 ADL4	User1ADL3	2448	5242	2933	71	123	10817
5	User4 ADL1	User1 ADL5	3862	4580	2441	90	1	10974
6	User1 Drill	User3 Drill	727	16982	4960	1557	0	24226
7	User2 Drill	User4 Drill	263	6308	3442	1453	0	11466
8	User4 ADL2	User3 ADL5	2656	4606	2462	501	1454	11680
9	User3 ADL1	User2 ADL2	4044	1436	2121	453	797	8851
10	User2 ADL3	User3 ADL2	2373	3414	2047	137	840	8811

Table 6.1: Number of Changes Detected in Opportunity Dataset

The change detection performance metrics described in Chapter 3 are employed to quantify the performance of the detection method for this dataset. Table 6.2 presents the results and Figure 6.3 graphically captured the relationships among the various metrics for evaluating the change detector. The average true positive rate (TPR) also known as recall, hit rate or sensitivity of the detector is 0.96 while the average true negative rate (TNR) or specificity is 0.96. The average positive detective (PDV) value also called precision in the parlance of information retrieval is 0.93 while the average negative detective value (NDV) which is the opposite of precision is obtained to be 0.97. The average false alarm rate (FAR) or fall out rate (FOR) of the detector is obtained to be 0.04 and the miss rate (MR) also called false negative rate (FNR) is also 0.04 while the false discovery rate (FDR) is 0.07 and the average detection accuracy is obtained as 0.96.

Exp. No.	**TP**	**TN**	**FP**	**FN**	**PDV**	**NPV**	**TPR**	**TNR**	**FAR**	**MR**	**FDR**	**Acc.**
1	15122	25845	1362	468	0.92	0.98	0.97	0.95	0.05	0.03	0.08	0.96
2	12839	13780	611	451	0.95	0.97	0.97	0.96	0.04	0.03	0.05	0.96
3	12604	16589	765	569	0.94	0.97	0.96	0.96	0.04	0.04	0.06	0.96
4	10817	17757	1566	3134	0.87	0.85	0.78	0.92	0.08	0.22	0.13	0.86
5	10974	17884	786	483	0.93	0.97	0.96	0.96	0.04	0.04	0.07	0.96
6	24226	43544	2111	1047	0.92	0.98	0.96	0.95	0.05	0.04	0.08	0.96
7	11466	31802	1644	548	0.87	0.98	0.95	0.95	0.05	0.05	0.13	0.95
8	11680	13701	621	493	0.95	0.97	0.96	0.96	0.04	0.04	0.05	0.96
9	8851	20077	875	379	0.91	0.98	0.96	0.96	0.04	0.04	0.09	0.96
10	8811	15239	702	379	0.93	0.98	0.96	0.96	0.04	0.04	0.07	0.96
				Average	**0.93**	**0.97**	**0.96**	**0.96**	**0.04**	**0.04**	**0.07**	**0.96**

TABLE 6.2: Cross-User Change Detection Performance for Opportunity Dataset

The implications of these results for the performance of change detection method are manifolds as indicated by the values of the performance metric. In the case of positive detective value (PDV) and negative detective value (NDV) it can be seen that the proportions of detected unchanged points that are truly unchanged are high compared to the proportion of detected changed points that are truly changed. On the other hand, in the case of true positive rate (TPR) and the true negative rate, it is clear that the proportion of unchanged points that are truly detected as unchanged are the same as the proportion of changed points that are truly detected as changed. Similarly, the false alarm rate (FAR) and the miss rate (MR) are the same indicating that the proportion of points detected as changed point which are truly unchanged are more or less the same as the proportion of unchanged points which are detected as changed points. It should be noted that a near perfect sensitive change detector would have a true positive rate or hit rate of 1 and a false-alarm rate or fall out rate of 0. Looking at the detection metrics results it is evident that the performance of the change detection is near the absolute perfect sensitive change detector.

A closer look at the performance trends across each case of the experiment shows that the positive detective value (PDV) has the highest value in the experiments 8 and 2 while the minimum value is at experiment 3. The negative predictive value has the maximum value across four experiments and a minimum value for the experiment number 4. In the case of true positive rate results across each experiment, the largest value is obtained for the experiment number 1 and 2 while the smallest value is obtained for experiment 4. The rest of the experiments have the same true positive rates value. For the true negative rate, it is the experiment 4 that also has the smallest value whereas the highest value obtained is common to all the other experiments except the experiment 1.

In the case of false alarm rate metric results across the experimental studies, it is the experiments 4 that has the highest value while the minimum values are obtained for the experiment $2, 3$ and 5. The miss rate maximum is obtained at experiment 4 indicating the worst performance of miss-detection in this experiment while the minimum miss rate is obtained at experiment 1 *and* 2. False discovery rate smallest value is obtained at experiment 2 while the maximum is at experiment 4 . As a result of these consistent performances, it is evident that the detection accuracy is high and the same for all the experiments except the experiment 4 which has the worst miss rate and false alarm rate. The worst performance in experiment 2 suggests that there is a sharp difference between the test subjects for this experiment compared to the other experiments.

Leave One Activity Out Results for Opportunity Dataset

In the sequential leave-one-out detection, the accuracy of the detection algorithm to correctly detecting novel activity from existing activities was evaluated. In this regard, one activity was left out of the remaining set of activities and the detection algorithm attempts to detect the new unseen activities from the pool of activities that are streamed for novel activity detection.

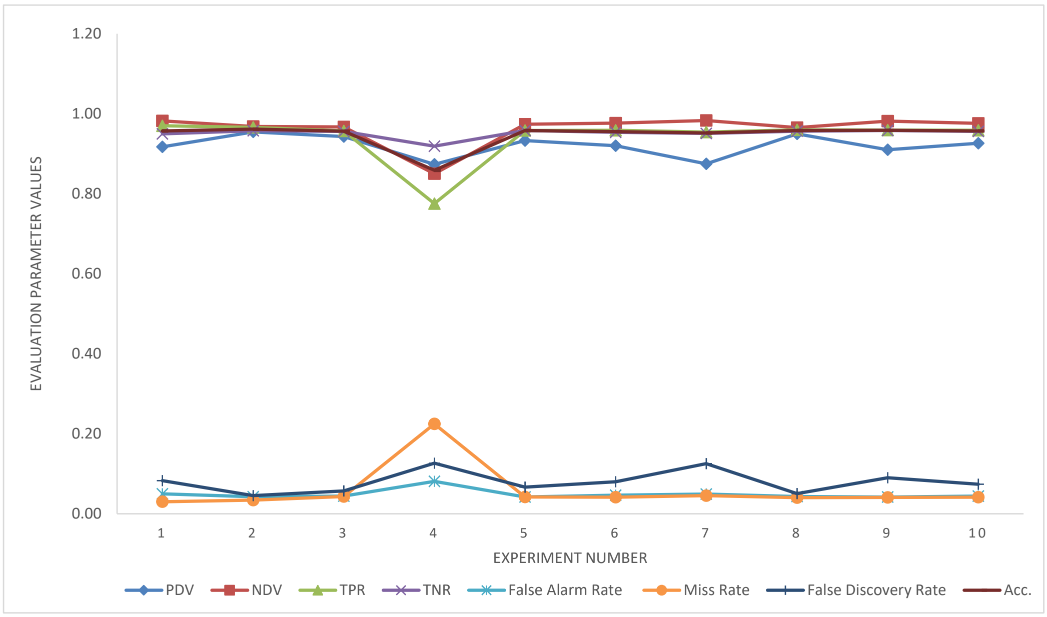


FIGURE 6.3: Performance of Cross User Detection for Opportunity Dataset

	Standing	**Walking**	**Lying**	**Sitting**	**Null**
Standing	334344 96.80%	5872 1.70%	2072 0.60%	2418 0.70%	691 0.20%
Walking	3586 1.80%	193237 97%	1594 0.80%	598 0.30%	198 0.10%
Lying	958 0.7%	1095 0.8 %	132610 96.9%	2190 1.6 %	0 0%
Sitting	203 0.80%	254 1%	483 1.90%	24455 96.30%	0 0%

TABLE 6.3: Confusion Matrix for Leave One Activity Detection for Opportunity Dataset

The base characteristic reference data are composed of only the known activities while the test data from which novel activity is to be detected is composed of both the already existing activities and unseen activity. Table 6.3 presents the confusion matrix of the detection results. The first left out *standing activity* was tested against the rest of the activities and the method is able to detect the left out activity with accuracy up to 96.8%. This implies that the detector is able to distinguish correctly the excluded activity from the pool of all the activities. It can be observed that certain percentages of the excluded activities are mis-detected to be other activities. For example, about 1.7% of the *standing activity* were mis-detected as *walking activity*. In the same vain, just about 0.6% and 0.7% of the *standing activity* were mis-detected to be *lying activity* and *sitting activity* respectively. The reason for the mis-detection can be attributed to the interleaved of activity that may occur as the user attempts to perform the designated novel activity. Also, the novel *walking activity* detection achieved an overall detection accuracy of 97% with less than 1% mis-detection into *lying and sitting activities* except the *standing activity* with mis-detection up to 1.8%. This slightly higher mis-detection of walking into standing can be because of the intermittent standing event and the rest-point in between the walking activity. This is particularly common in the way some people walk especially old people with walking activity that intermittently involves standing activity.

Furthermore, the detection accuracy of *lying* and *sitting* attained 96.9% and 96.3% receptively. The mis-detections of lying activity and sitting activities are close to each other indicating their level of similarity. Although, the sensor value during lying will be different from sitting due to the gravitational component of the accelerometer. Nevertheless, there is a tendency for a classifier to mis-classified a sitting activity to lying activity, especially if the lying point is within the same same level as the sitting point.

In conclusion, the leave-one-out detection experiment achieved a high accuracy indicating the detection technique is robust to detect new activity from existing activity thereby being able to detect when concept drift may occur whenever there are differences in the way one subject perform an activity from the other subjects.

6.4.3 HARS Dataset Experiment Results

The result of the experiment with this dataset is presented in this section. Two modes of experiment were performed on the dataset. The cross user evaluation was performed followed by the leave-one-out experiment.

Cross User Detection Results for HARS Dataset

The cross user experiment is performed with the Human Activity Recognition with Smartphone dataset by splitting the data into two portions. The data of user 19 to

30 were used to generate the baseline characteristics data model and build different learning models while users 1 to 18 dataset were used to test the change detection method. The result of the experiment while detecting the changes experienced under 4 different classifiers namely: KNN, Naive Bayes, SVM and Decision Tree are presented in Table 6.4. The performance metrics results for detecting concept changes in the classifiers are presented. The concept change is manifested by the classification error of the classifiers and the detection framework is meant to detect the misclassified instances of the test activity data.

According to Table 6.4, the average true positive rate (TPR) also known as recall, hit rate or sensitivity of the detector spanning all the 4 test classification algorithms is 0.96 while the average true negative rate (TNR) or specificity is 0.96. The average positive detective value (PDV) also called precision is 0.83 while the average negative detective value (NDV) which is the negative precision is obtained to be 0.99. The average false alarm rate (FAR) or fall out rate (FOR) of the detector is obtained to be 0.04 and the miss rate (MR) also called false negative rate (FNR) is also 0.04 while the false discovery rate (FDR) is 0.17 and the average detection accuracy is obtained as 0.96.

	TP	**TN**	**FP**	**FN**	**PDV**	**NDV**	**TPR**	**TNR**	**FAR**	**MR**	**FDR**	**Acc.**
KNN	793	4799	193	32	0.80	0.99	0.96	0.96	0.04	0.04	0.20	0.96
NAÏVE BAYES	1612	3991	143	71	0.92	0.98	0.96	0.97	0.03	0.04	0.08	0.96
SVM	519	5105	155	38	0.77	0.99	0.93	0.97	0.03	0.07	0.23	0.97
DTREE	1143	4388	253	33	0.82	0.99	0.97	0.95	0.05	0.03	0.18	0.95
				Average	**0.83**	**0.99**	**0.96**	**0.96**	**0.04**	**0.04**	**0.17**	**0.96**

Table 6.4: Amount of Changes Detected in HARS Dataset

The implications of these results for the performance of change detection method are manifolds by looking at each of the performance metric values. In the case of positive detective value (PDV) and negative detective value (NDV) it can be seen that the proportion of detected unchanged points that are truly unchanged are more than the proportion of detected changed points that are truly changed. This implies that the detector is more sensitive to detecting unchanged points than the changed points. In other words, the detector can recognise instances that are correctly classified more than those that are incorrectly classified. On the

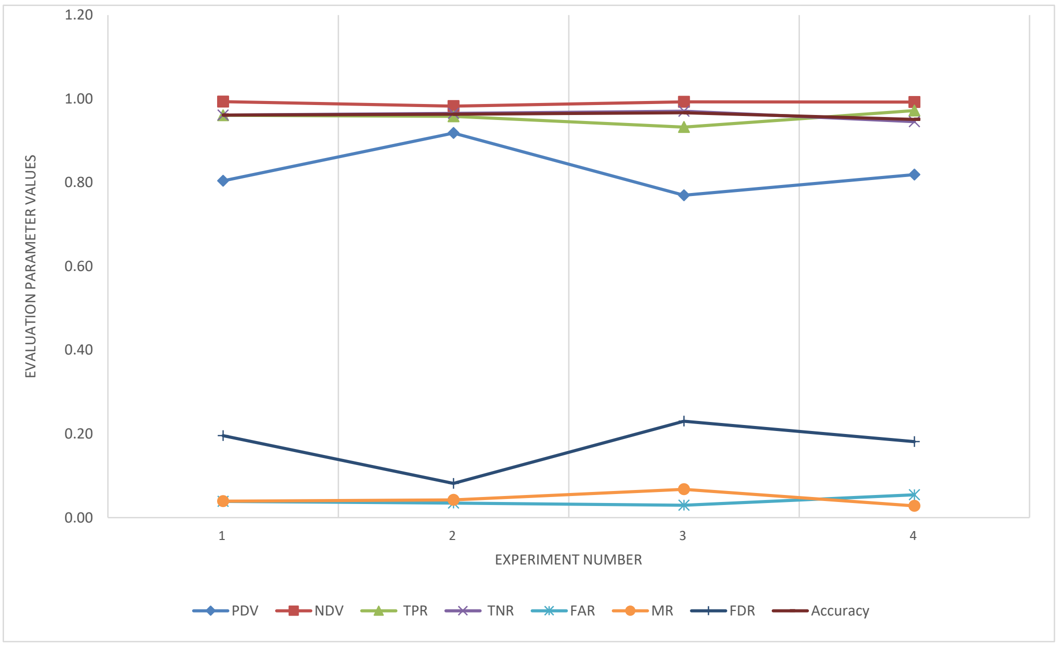


FIGURE 6.4: Performance of Cross User Detection in HARS Dataset

other hand, in the case of true positive rate (TPR) and the true negative rate, it is obvious that the proportions of unchanged points that are truly detected as unchanged are the same as the proportion of changed points that are truly detected as changed. Similarly, the false alarm rate (FAR) and the miss rate are the same (MR) indicating that the proportion of points detected as changed points which are truly unchanged are more or less the same as the proportions of unchanged point which are detected as changed points. It should be noted that a near perfect sensitive change detector would have a true positive rate or hit rate of 1 and a false-alarm rate or fall out rate of 0 . Looking at the detection metrics results it is evident that the performance of the change detection is near the absolute perfect sensitive change detector.

Furthermore, a closer look at the performance trends in detecting the classification errors across each of the classification algorithms shows that the positive detective value (PDV) has the highest value when detecting changes from the *Naive Bayes classifier* while that of KNN has the minimum value. This indicates that the error rate across each algorithm is different and the detector can detect them regardless of the classifier being used. The negative predictive value is relatively uniform for all the detection in the classifiers, indicating that the detector can detect consistently points that are unchanged.

In the case of true positive rate results across each experiment, the largest value is obtained for detecting changes in decision tree while the smallest value is obtained for SVM. The rest of the experiments have the same true positive rates value. For the true negative rate, Naive Bayes and SVM shares the highest value whereas the smallest value is under Decision Tree error detection.

In the case of false alarm rate metric results across the experimental studies, it is the Decision Tree that has the highest value while the minimum values go to detecting changes in SVM classifier. The miss- rate's maximum is obtained under SVM indicating the worst performance of misdetection in this experiment while the minimum miss rate is obtained under the decision tree errors. False discovery rate has the smallest value under Naive Bayes classifier error detection while the highest value is obtained under SVM. As a result of these consistent performances across the experiments, the detection accuracy of above 0.95 is obtained in all the experiments except the experiment involving decision tree in which detector accuracy is exactly 0.95.

Leave One Activity Out Results for HARS Dataset

In the leave-one-activity-out detection, the accuracy of the detection algorithm in detecting novel activity from existing activities was evaluated. In this regard, one activity was left out of the remaining set of activities and the detection algorithm attempts to detect the new unseen activities from the pool of activities that are streamed for novel activity detection. The base characteristic reference data are composed of only the known activities while the test data from which novel activity is to be detected is composed of both the already existing activities and unseen activity.

Table 6.5 presents the confusion matrix of the detection results. The first left out *walking activity* was tested against the rest of the activities and the detection result can detect the left out activity with accuracy up to 94.1%. This implies

	Walking	**Upstairs**	**Downstairs**	**Sitting**	**Standing**	**Laying**
Walking	**1606** 94.1 %	26 1.5%	24 1.4%	17 1 %	17 1%	17 1%
Upstairs	18 1.2%	**1460** 95.2%	26 1.7%	9 0.6%	8 0.5%	12 0.8%
Downstairs	14 1%	20 1.4%	**1342** 96%	7 0.5%	6 0.4%	10 0.7%
Sitting	14 0.8%	11 0.6%	9 0.5%	**1701** 95%	18 1%	38 2.1%
Standing	21 1.1%	19 1%	19 1%	17 0.9%	**1811** 95%	19 1%
Laying	14 0.7%	12 0.6%	8 0.4%	35 1.8%	19 1%	**1859** 95.5%

Table 6.5: Leave One Out Detection for HARS Dataset

that the detector can distinguish the excluded activity correctly from the pool of all the activities. It can be observed that certain percentages of the excluded activities were mis-detected to be other activities. For example, about 1.5% and 1.4% of the *walking activity* were mis-detected as *walking upstairs* and *walking downstairs* respectively. This can be attributed to momentary nature of the two activities that involves the elements of walking. In the same vain, just about 1% of the *walking activity* were mis-detected to each of *sitting activity* , *standing activity* and *laying activity* respectively. The reason for the misdetection can be attributed to the interleaved of activity that may occur as the user attempts to perform the designated novel activity.

Also, the novel *walking-upstairs activity* detection achieved an overall detection accuracy of 95.2% with less than 1% misdetection each into *sitting, standing and laying activities* except the *walking and walking-downstairs activities* with misdetection up to 1.2% and 1.7%. This slightly higher misdetection of walking-upstairs into walking and walking-downstairs can be attributed to the relative interleaving actions of walking that is common to all the three activities. Similarly, the detection accuracy of *walking-downstairs activity* attained 96.0% with misdetection into walking and walking-upstairs reaching up to 1%. This is due to the similarity of walking component that is common them. The same 95% detection accuracy is achieved for sitting and standing activities with while laying activity is slightly higher than the two by 0.5%. It can be concluded that the leave-one-out activity detection experiment for this dataset achieved a high accuracy indicating that the detection technique is robust to detect new activity from existing activity thereby being able to detect when concept drift may occur whenever there are differences in the way one subject perform an activity from the other subjects.

6.4.4 WISDM Dataset Experiment Results

The result of the experiment with this dataset is presented in this section. Two modes of experiment were performed on the dataset. The cross user evaluation was performed followed by the leave-one-out experiment.

Cross User Detection Results for WISDM Dataset

The cross user experiment for this dataset was performed by partitioning the data into two. The first portion consists of data from 18 users while the second portion also contain data from 18 users. But, due to the imbalance nature of the dataset, the division was made in such way that the instances in the two portions of the data are balanced. This was done by selecting the users in each group according to the amount and type of data contributed to the entire dataset. The first part of the data was used to generate the baseline characteristics data and build different learning models while the second part was used to test the change detection method. The result of the experiment while detecting the changes experienced under 4 different classifiers namely: KNN, Naive Bayes, SVM and Decision Tree are presented in Table 6.6.

According to Table 6.6, the average true positive rate (TPR) also known as recall, hit rate or sensitivity of the detector spanning all the 4 classification algorithms is 0.96 while the average true negative rate (TNR) or specificity is also 0.96. The average positive detective value (PDV) also called precision is 0.92 while the average negative detective value (NDV) which is the negative precision is obtained to be 0.98. The average false alarm rate (FAR) or fall out rate (FOR) of the detector is obtained to be 0.04 and the miss rate (MR) also called false negative rate (FNR) is also 0.04 while the false discovery rate (FDR) is 0.08 and the average detection accuracy is obtained as 0.96.

	TP	TN	FP	FN	PDV	NDV	TPR	TNR	FAR	MR	FDR	Acc.
KNN	778	1691	64	57	0.92	0.97	0.93	0.96	0.04	0.07	0.08	0.95
NAÏVE BAYES	794	1688	84	24	0.90	0.99	0.97	0.95	0.05	0.03	0.10	0.96
SVM	655	1851	64	20	0.91	0.99	0.97	0.97	0.03	0.03	0.09	0.97
DTREE	804	1695	64	27	0.93	0.98	0.97	0.96	0.04	0.03	0.07	0.96
				Average	**0.92**	**0.98**	**0.96**	**0.96**	**0.04**	**0.04**	**0.08**	**0.96**

Table 6.6: Cross User Change Detection in WISDM Dataset

By taking a look at each of the performance metric value, it is evident that the method produce a sterling performance in detection accuracy. In the case of positive detective value (PDV) and negative detective value (NDV) , the results indicate that the proportion of detected unchanged points that are truly unchanged are more than the proportion of detected changed points that are truly changed. This implies that the detector is more sensitive to detecting unchanged points than the changed points. In other words, the detector can recognise instances that are correctly classified more than those that are incorrectly classified. This is because there are more unchanged points than changed points in the test data. On the other hand, in the case of true positive rate (TPR) and the true negative rate, the proportions of unchanged points that are truly detected as unchanged are close to the proportion of changed points that are truly detected as changed. Similarly, the false alarm rate (FAR) and the miss rate is the same (MR) indicating that the proportion of points detected as changed points which are truly unchanged are more or less the same as the proportions of unchanged point which are detected as changed points. It should be noted that a near perfect sensitive change detector would have a true positive rate or hit rate of 1 and a false-alarm rate or fall out rate of 0 . Looking at the detection metrics results it is evident that the performance of the change detection is near the absolute perfect sensitive change detector.

Furthermore, a closer look at the performance trends in detecting the classification errors across each of the classification algorithms shows that the positive detective value (PDV) has the highest value when detecting changes from the *decision tree classifier* while that of Naive Bayes has the minimum value. This indicates that the error rate across each algorithm is different and the detector can detect them regardless of the classifier being used. The negative predictive value is relatively

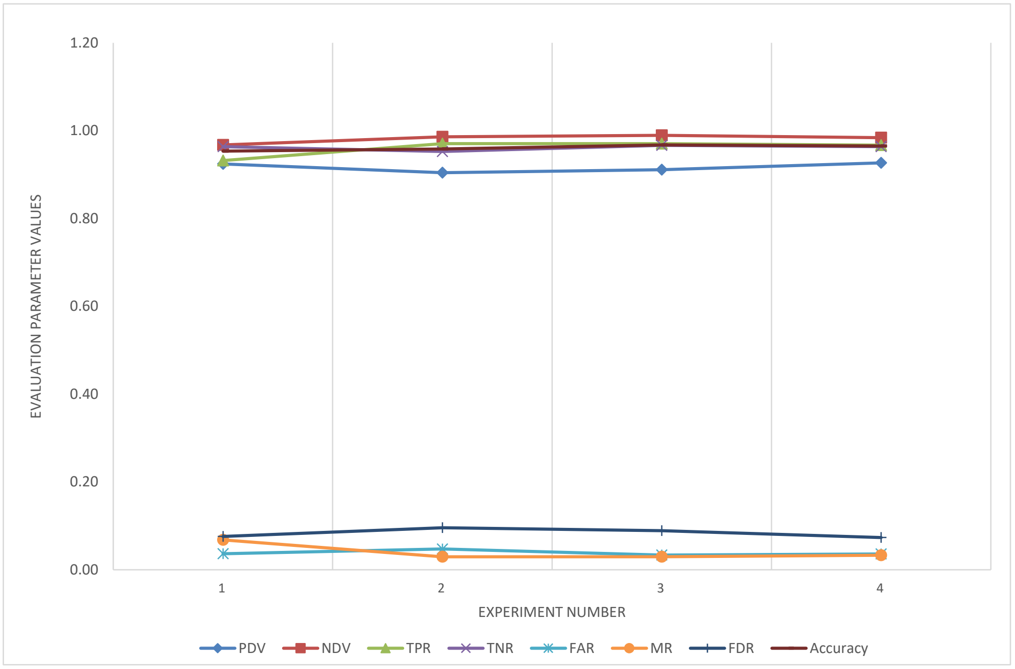


FIGURE 6.5: Performance of Cross User Detection in WISDM Dataset

uniform for all the detection in the classifiers, indicating that the detector can detect consistently points that are unchanged.

In the case of true positive rate (TPR) results across each experiment, the same value is obtained for detecting changes under decision tree, SVM and Naive Bayes while the smallest value is obtained for kNN. For the true negative rate, SVM has the highest value whereas the smallest value is under kNN error detection. In the case of false alarm rate (FAR) metric results across the experimental studies, it is the Naive Bayes that has the highest value while the minimum values is obtained under SVM classifier. The miss-rate's maximum is obtained under kNN indicating the worst performance of mis-detection of mis-classified points while the same amount of miss rate is obtained under the rest of the classifiers. False discovery rate has the smallest value under decision tree classifier error detection while the highest value is obtained under Naive Bayes. As a result of these consistent performances across the experiments with the lowest accuracy obtained under kNN as 0.95 and maximum obtained under SVM as 0.97.

Leave One Activity Out Results for WISDM Dataset

In the leave-one-activity-out detection, the accuracy of the detection algorithm in detecting novel activity from existing activities was evaluated. The setting of the experiment involves one activity being left out of the remaining set of activities and the detection algorithm attempts to detect the new unseen activities from the pool of activities that are streamed for novel activity detection. The base characteristic reference data are composed of only the known activities while the test data from which novel activity is to be detected is composed of both the already existing activities and unseen activity.

	Walking	**Jogging**	**Standing**	**Sitting**	**Upstairs**	**Downstairs**
Walking	**2019** 97%	0 0 %	0 0 %	0 0 %	42 2 %	21 1%
Jogging	16 1 %	**1593** 98 %	16 1 %	0 0 %	0 0 %	0 0%
Standing	2 1 %	0 0 %	**230** 97.5 %	1 0.5%	1 0.5 %	1 0.5 %
Sitting	0 0 %	0 0 %	3 1 %	**300** 98 %	3 1%	0 0 %
Upstairs	3 0.5%	0 0 %	6 1%	0 0 %	616 97.5%	6 1%
Downstairs	3 0.5 %	0 0%	3 0.5%	0 0 %	5 1%	517 98%

Table 6.7: Leave One Out Detection in WISDM Dataset

Table 6.7 presents the confusion matrix of the detection results. The first left out *walking activity* was tested against the rest of the activities and the method can detect the left out activity with accuracy of up to 97%. This implies that the detector can distinguish the excluded activity correctly from the pool of all the activities. It can be observed that certain percentages of the excluded activities were mis-detected to be other activities. For example, 2% and 1% of the *walking activity* were mis-detected as *walking upstairs* and *walking downstairs* respectively. This can be attributed to momentary nature of the two activities that involves the elements of walking.

Also, when the *jogging activity* was left-out and tested against the rest of the activities, the detection method achieved an overall detection accuracy of 98% with 1% misdetection into *walking and standing activities.* This little misdetection of jogging activity into walking and walking and standing activities can be attributed to the relative interleaving actions of standing that is common to all the three activities. Similarly, the detection accuracy of *standing activity* attained 97.5% with 1% mis-detection into walking and 0.5% mis-detection into walking-upstairs, sitting and walking-downstairs.This is due to the similarity of walking component that is common them. The sitting activity was detected up to 98% accuracy while walking-upstairs and walking-downstairs detection attained up to 97.5% and 98% respectively. The little amount of mis-detections of one activity into another activity are due to the transitioning period that exists from one activity to the other.

It can be concluded that the leave-one-out activity detection experiment for this dataset achieved a high accuracy indicating that the detection technique is robust to detect new activity from existing activity thereby being able to detect when concept drift may occur whenever there are differences in the way one subject perform an activity from the other subjects.

6.4.5 Comparative Evaluation with Drift Detection Method (DDM)

This section compares the performance of the developed distance based change detection (DBD) with DDM [28]. While the DMM assumes the presence of ground truth in order to monitor the error rate of a model, the approach presented in this chapter *(Distance Based Change Detector (DBD))* does not assume the presence of ground truth.

The comparative evaluation was carried out by taking the initial accuracy of a base classifier for each data pair used in the previous section. Then, the two methods were deployed to detect changes in the model accuracy, the detected changes were then subjected to active learning to provide the correct label at the points where the changes were detected and retrain the classifier. After the provision of the true labels at the different points of detected changes, the accuracy of the model is then recomputed to determine the efficacy of the correction made as a result of the changes detected by the two methods. The gains between the initial accuracy without change detection is then compared with the accuracy after providing correct labels during detections. The improvements to the classifier under the two change detectors with active learning intervention are presented in Table 6.8. The table shows that the accuracy gain in each experiment increases when active learning strategy is invoked for adaptation at every step of the change signal to the classification model inaccurate outputs. The accuracy gained ranges between 22.95% to 43.35% yielding an average gain of 35.62%. This is an indication that the change detector has enabled the adaptation process to take an informed decision in updating the underlying model.

Reference Data	**Test Data**	**Accuracy Before (%)**	**Accuracy After DBD (%)**	**Accuracy After DDM (%)**	**DBD Gain (%)**	**DDM Gain (%)**
User2 ADL2	User3 ADL4	51.99	95.34	85.34	43.35	33.35
User3 ADL3	User4 ADL5	56.85	94.79	86.79	37.94	29.94
User4 ADL4	User1ADL3	62.78	98.68	88.68	35.9	25.9
User4ADL1	User1 ADL5	61.97	97.88	87.88	35.91	25.91
User1 Drill	User3 Drill	64.37	96.83	88.83	32.46	24.46
User2 Drill	User4 Drill	73.57	96.52	86.52	22.95	12.95
User4 ADL2	User3 ADL5	54.06	94.89	84.89	40.83	30.83
				Mean	**35.62**	**26.19**

TABLE 6.8: Comparative Evaluation of DBD *vs* DDM in Opportunity Dataset

Statistical Test Result 6.4.1. **Paired T-test for DBD *vs* DDM**

	N	Mean	StDev	SE Mean
DBD Gain	7	35.62	6.62	2.5
DDM Gain	7	26.19	6.65	2.51
Difference	7	9.429	0.976	0.369

95% CI for mean difference is: (8.526, 10.331)

T-Test of mean difference $= 0 (vs \neq 0)$: T-Value= 25.56, P-Value= 0.000

The results were further subjected to paired t-test to determine whether to accept or reject the null hypothesis that there is no difference in the performance of the two methods. $H_0 = \mu_{DBD} - \mu_{DDM} = 0$ *i.e* $\mu_d = 0$. The result of the paired t-test for this dataset is presented in statistical test result 6.4.1

It can be seen that the confidence interval (CI) for the mean difference between the two methods does not include zero, which suggests a difference between them. The small p-value ($p = 0.000$) further suggests that the data are inconsistent with the null hypothesis $H_0 = \mu_{DBD} - \mu_{DDM} = 0$ *i.e. population mean* $\mu_d = 0$, this implies that the two methods do not perform equally. Specifically, DBD approach with $mean = 35.62$ performs better than DDM with $mean = 26.19$ in terms of using the change detection to initiate adaptation by active learning.

6.5 Chapter Summary

This chapter has presented a novel approach for change detection that is based primarily on detecting classification error of an activity recognition model without recourse to ground truth knowledge. The method relies on the data characteristics extracted from the reference training set and the distance between the new instances and the stored characteristics to determine the correct or incorrect classification of the instances. An incorporated threshold parameter control the decision making of the detector. The threshold parameter is an important setting that make or mar the correct detection of misclassified instances. Furthermore, the decision of the detector is corroborated by incorporating a pluggable adaptation module that updates the model in reaction to the change signal. The tested active learning adaptation method indicates an average accuracy increase of 35.62% in the recognition model which outperforms the well cited DDM[28]. Finally, experimental evaluation of the approach shows a consistent average detection accuracy of up to 95% across the three datasets.

Chapter 7

Conclusion

This chapter summarises the major contributions of this thesis towards addressing the problem of change detection in the classification model for activity recognition. This problem is a challenging task and the approaches presented have been experimented and confirmed to produce favourable performance in solving the problem. In addition to the contributions of the thesis, this chapter also highlights and proposed future work that emanate from this thesis.

7.1 Conceptual Layered Framework for Activity Recognition System

The techniques developed in this thesis are codified into a conceptual layered architectural framework shown in Figure 7.1. The proposed 5 layered architecture for activity recognition system model adaptation consists of: (i) Data Acquisition and Pre-processing, (ii) statistical data summaries formation, (iii) Classification model, (iv) Change detectors and (v) Pluggable adaptation module.

Although, layered architectural frameworks are not new to other research domain, such as in inferential measurement systems [66, 78], nonetheless, this thesis proposes the layered framework to the domain of activity recognition model adaptation. The novelty of the proposed codified layered framework is exhibited in the organisation and implementation of the components that made up the framework.

The functions of each layer and how they achieved their objectives are described as follows:

Layer 1: Raw Sensor Data Acquisition and Pre-processing: The function of the first layer is to acquire raw data from the sensors. It also process this raw data through standard techniques such as segmentation of the sensor into fixed size chunks, and feature extraction to extract features that are more discerning than the raw sensor data. Other functions performed by this layer include feature scaling and normalisation, replacing missing values and feature selection.

Since existing publicly available real life activity recognition datasets were utilized, some of these functions were not implemented from scratch. The dataset used in this research namely: (WISDM, HARS and Opportunity) were pre-processed to replace missing values in some part of the dataset. The approach employed replaced the missing values in a column with the means of the remaining data in the columns. But if the entire column or a substantial part of it are missing, the feature is deselected from the dataset.

Layer 2: Statistical Data Summary Formation: The second layer of the framework is saddled with the function of creating statistical data summaries from the dataset that emerged from *layer 1.* The statistical data summaries are more meaningful and concise representation of the original large dataset. The reduced dataset is important for the mobile environment where there is resource constraints challenges. The proposed statistical data summaries were obtained from the sub-clusters created from each class of activity. The sub-clusters in a given type of activity data are obtained by applying clustering to the data. The clustering yielded different patterns that are present in the type of activity. Once this is done, the centroid statistical data summary and its data points was computed by finding the centroids of the sub-clusters. The amount of centroids data obtained depends on the percentage of data to be retained in the dataset. Similarly, the minimum and maximum statistical data summaries were obtained by finding the minimum and maximum along

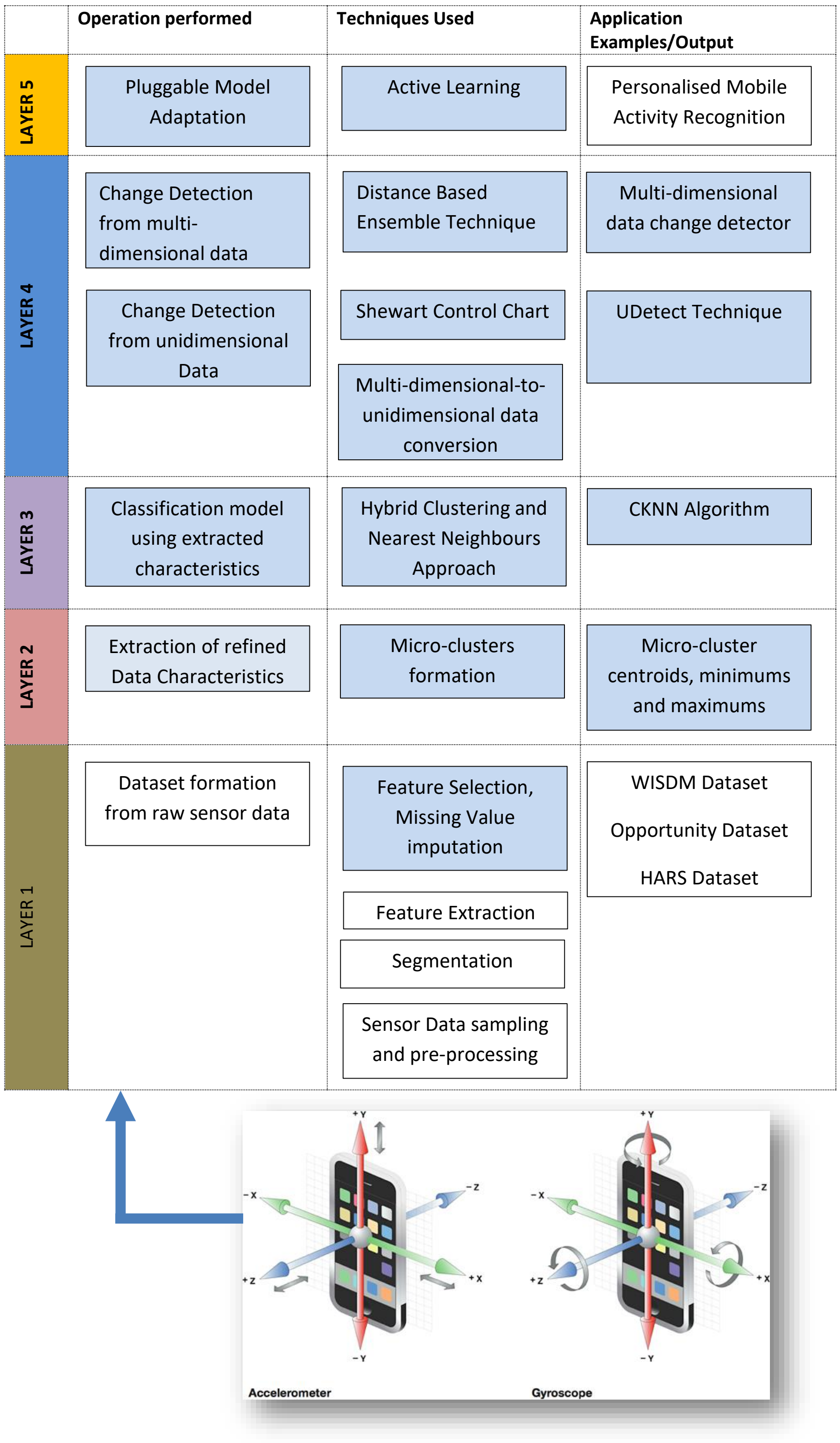

Operation performed
Techniques Used
Application Examples/Output
LAYER 5
Pluggable Model Adaptation
Active Learning
Personalised Mobile Activity Recognition
LAYER 4
Change Detection from multi-dimensional data
Distance Based Ensemble Technique
Multi-dimensional data change detector
Change Detection from unidimensional Data
Shewart Control Chart
UDetect Technique
Multi-dimensional-to-unidimensional data conversion
LAYER 3
Classification model using extracted characteristics
Hybrid Clustering and Nearest Neighbours Approach
CKNN Algorithm
LAYER 2
Extraction of refined Data Characteristics
Micro-clusters formation
Micro-cluster centroids, minimums and maximums
LAYER 1
Dataset formation from raw sensor data
Feature Selection, Missing Value imputation
Feature Extraction
Segmentation
Sensor Data sampling and pre-processing
WISDM Dataset
Opportunity Dataset
HARS Dataset
Accelerometer
Gyroscope
+Y
-Y
+X
-X
+Z
-Z

FIGURE 7.1: Layered Framework for Activity Recognition System

each column of the *d-dimensional* data points that are present in a given sub-cluster. While the centroid statistical data summary represents the centre of the sub-cluster, the minimum and maximum statistical data summaries serve as the upper and lower boundary of the data points in the sub-cluster With this, the pattern exhibited by a given sub-cluster is summarised by the three statistical data summaries.

Layer 3: Classification method: The purpose of this layer is to use the sub-cluster data formed in the previous layer 2 as the basis for classifying new instances. Thus, a classification algorithm termed 'CKNN' was implemented. This classifier for activity recognition classifies new instances by computing the similarity between each instance and the statistical data summaries data from Layer 1. label of the most similar data to the instance is then assigned to the instance.

The evaluation of the approach shows that the accuracy of the classifier depends on the amount of statistical data summaries obtained from the sub-clusters. Also, each statistical data summary has a varying degree of classification power. The centroid characteristic was found to exhibit the best classification power followed by minimum and maximum. However, the weighted ensemble of the individual decisions gives the overall best accuracy. The result shows that there is performance improvements over the KNN, Decision tree and Naive Bayes classifiers. Thus, it can be concluded that the developed approach with the reduced statistical data summaries data can enable superior classification accuracy compared with the benchmarked state of the art algorithms that utilised all the entire datasets.

Layer 4: Change Detection: The change detector stage is focused on methods to detect changes in the data source while the base learner is classifying new unseen instances. The implemented methods are different from existing approaches in that they do not rely on the assumption of the presence of ground truth to diagnose the presence of a change in the data source. The method relies on the reference data from the training set to detect variations in the new set of classified samples. Chapter 5 and 6 details the developed techniques.

Layer 5: Informed Model Adaptation The final layer is the model adaptation layer that relies on the decision from the previous layer to adapt the underlying model once a change has been signalled. The framework can work with different type of model adaptation techniques because there is decoupling between the change detector and the model adaptation in the proposed approach. Active learning adaptation module is incorporated to show the applicability of the proposed change detector.

7.2 Contributions

Based on the implemented components of the derived layered framework, the following contributions to knowledge have emerged from this research.

- The thesis proposed and developed three statistical summaries that can be generated from any dataset to yield more discerning and reduced dataset. The data derived from the summaries summarised the patterns in the original dataset and the resulting reduced data is used for similarity based classification of new samples. Their major benefit is that they can be harnessed for instance-based classification in a resource constraint environment such as mobile phone based activity recognition.
- Another significant contribution of this thesis is the developed classifier for activity recognition. The classifier employed the statistical summaries to compute the similarity between new instances and the referenced data. The weighted ensemble combination of the classification decisions from each statistical summary results in a better performance than three existing benchmarked classification algorithms widely used for activity recognition. Statistical test of the results in Chapter 5 test results 4.4.1, 4.4.2 and 4.4.3 show that there is significant statistical evidence that the algorithm performs better.
- This thesis also contributes to knowledge by proposing the incorporation of change detection in activity recognition model prior to its adaptation. The

implemented approach called 'UDetect' can detect classification model degraded accuracy without access to the ground truth label of the classified instances. This is unlike existing approaches [14, 28, 34, 91] that require the presence of labels to detect the change. The proposed approach is more suitable for activity recognition where labels are scarce and tedious for users to provide label with each data instance. Also, constant provision of labels during online activity recognition is impractical and unrealistic. Therefore, this contribution eliminates this challenge.

- Another main contribution of this thesis is the development of a distance based change detection technique that relies on the developed statistical summaries for comparing new classified samples and detects any drift in the original class of the activity. The proposed approach uses distance function and a threshold parameter to detect the accuracy change in the classifier that is classifying new instances. This is unlike existing approaches [14, 28, 34, 91] that require the presence of labels to detect the change. The distance based approach eliminates the requirement for providing labels. It is more suitable for activity recognition domain where labels are tedious to provide by the user during online recognition.

7.3 Future Work

The following future work are worthy extensions and direction from this research

(a) An important future development work is the development of a holistic activity recognition system that integrates the proposed layered framework into a real life application. Such system will incorporate informed model adaptation so that there will be a detection of changes before initiating the adaptation process.

(b) Development of window management strategy for deciding on the optimal amount of window to use in the formation of a sample to be passed to the detection routine. This problem is important because with the current methods the sample sizes are manually tuned based on the size of the data and the trade-off of the delay in detection. An automated approach that optimises the selection and formation of the window size based on some system parameters such as memory requirements and processing time to ensure that the window formed is adequate and produced on time before the arrival of more instances.

(c) Another important future work with respect to this thesis is the development of adaptation methods other than the active learning approach proposed in this thesis. The adaptation technique can rely on the detection framework to decide on when, and how to adapt the base model to cater for the changes that is detected in the underlying distribution of the model. There are many possible actions that can be taken to adapt the base model such actions include collecting all instances from the point of detection at time t_k up till time t_{k+1} with set of samples collected within the interval time $t_{k+1} - t_k$ denoting the new activity regime that should be adapted to. However, method should be incorporated to determine the actual class of activity without resorting to inquiry from the user.

Appendix A

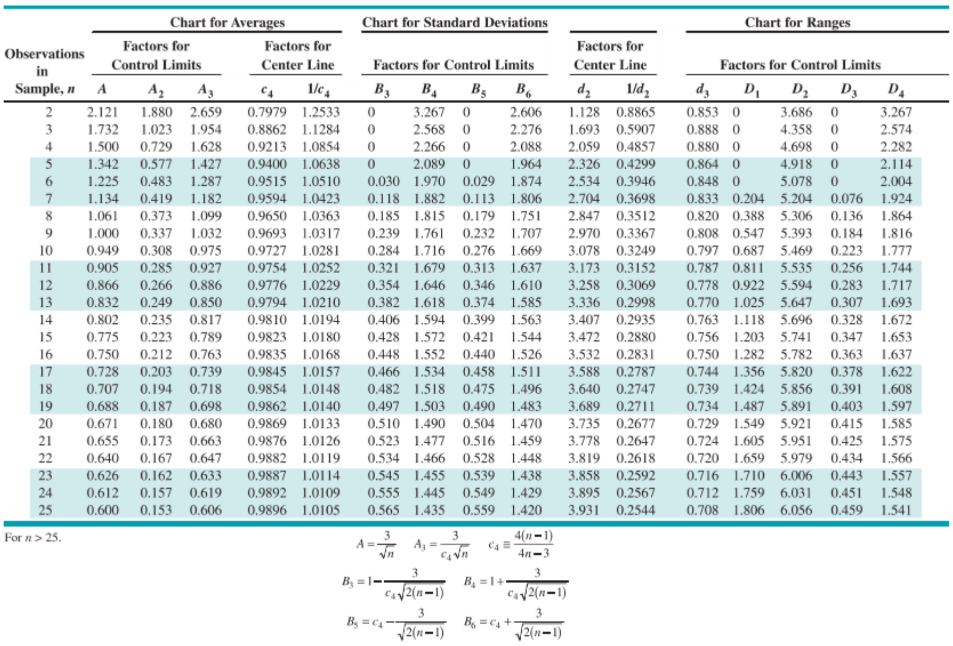
Publications

- S.A. Bashir, DC Doolan, A Petrovski “ The Effect of Window Length on Accuracy of Smartphone-Based Activity Recognition” IAENG International Journal of Computer Science 43 (1), 126-136.
- S.A. Bashir, DC Doolan, A Petrovski “Clustering and Nearest Neighbor Based Classification Approach for Mobile Activity Recognition” Journal of Mobile Multimedia Volume 12 12 , 110-214.
- S.A Bashir, D. Doolan, A. Petrovski “The Impact of Feature Vector Length on Activity Recognition Accuracy on Mobile Phone” Proceed-ings of the Internal Conference on Knowledge Engineering and Data Mining London, U.K., 1 - 3 July, 2015
- S.A. Bashir, D Doolan, A Petrovski “ClusterNN: A Hybrid Classifica-tion Approach to Mobile Activity Recognition” MoMM ’15 The 13th International Conference on Advances in Mobile Computing and Multimedia. 11-13 Dec. ACM New York 2015.
- S.A. Bashir,, A. Petrovski, D. Doolan “UDetect: Unsupervised Concept Change Detection for Mobile Acitivty Recogntion ” MoMM ’16 The 14th In-ternational Conference on Advances in Mobile Computing and Mul-timedia. 29-30 Nov. ACM New York 2016

- S.A. Bashir, A. Petrovski, D. Doolan "A Framework for Unsupervised Change Detection in Activity Recognition" International Journal of Pervasive Computing and Communication 13(2) Pages 157-175.

Appendix B

Table of Control Chart Parameters



Observations in Sample, n	Chart for Averages					Chart for Standard Deviations							Chart for Ranges				
	Factors for Control Limits			Factors for Center Line		Factors for Control Limits				Factors for Center Line			Factors for Control Limits				
	A	A_2	A_3	c_4	$1/c_4$	B_3	B_4	B_5	B_6	d_2	$1/d_2$		d_3	D_1	D_2	D_3	D_4
2	2.121	1.880	2.659	0.7979	1.2533	0	3.267	0	2.606	1.128	0.8865		0.853	0	3.686	0	3.267
3	1.732	1.023	1.954	0.8862	1.1284	0	2.568	0	2.276	1.693	0.5907		0.888	0	4.358	0	2.574
4	1.500	0.729	1.628	0.9213	1.0854	0	2.266	0	2.088	2.059	0.4857		0.880	0	4.698	0	2.282
5	1.342	0.577	1.427	0.9400	1.0638	0	2.089	0	1.964	2.326	0.4299		0.864	0	4.918	0	2.114
6	1.225	0.483	1.287	0.9515	1.0510	0.030	1.970	0.029	1.874	2.534	0.3946		0.848	0	5.078	0	2.004
7	1.134	0.419	1.182	0.9594	1.0423	0.118	1.882	0.113	1.806	2.704	0.3698		0.833	0.204	5.204	0.076	1.924
8	1.061	0.373	1.099	0.9650	1.0363	0.185	1.815	0.179	1.751	2.847	0.3512		0.820	0.388	5.306	0.136	1.864
9	1.000	0.337	1.032	0.9693	1.0317	0.239	1.761	0.232	1.707	2.970	0.3367		0.808	0.547	5.393	0.184	1.816
10	0.949	0.308	0.975	0.9727	1.0281	0.284	1.716	0.276	1.669	3.078	0.3249		0.797	0.687	5.469	0.223	1.777
11	0.905	0.285	0.927	0.9754	1.0252	0.321	1.679	0.313	1.637	3.173	0.3152		0.787	0.811	5.535	0.256	1.744
12	0.866	0.266	0.886	0.9776	1.0229	0.354	1.646	0.346	1.610	3.258	0.3069		0.778	0.922	5.594	0.283	1.717
13	0.832	0.249	0.850	0.9794	1.0210	0.382	1.618	0.374	1.585	3.336	0.2998		0.770	1.025	5.647	0.307	1.693
14	0.802	0.235	0.817	0.9810	1.0194	0.406	1.594	0.399	1.563	3.407	0.2935		0.763	1.118	5.696	0.328	1.672
15	0.775	0.223	0.789	0.9823	1.0180	0.428	1.572	0.421	1.544	3.472	0.2880		0.756	1.203	5.741	0.347	1.653
16	0.750	0.212	0.763	0.9835	1.0168	0.448	1.552	0.440	1.526	3.532	0.2831		0.750	1.282	5.782	0.363	1.637
17	0.728	0.203	0.739	0.9845	1.0157	0.466	1.534	0.458	1.511	3.588	0.2787		0.744	1.356	5.820	0.378	1.622
18	0.707	0.194	0.718	0.9854	1.0148	0.482	1.518	0.475	1.496	3.640	0.2747		0.739	1.424	5.856	0.391	1.608
19	0.688	0.187	0.698	0.9862	1.0140	0.497	1.503	0.490	1.483	3.689	0.2711		0.734	1.487	5.891	0.403	1.597
20	0.671	0.180	0.680	0.9869	1.0133	0.510	1.490	0.504	1.470	3.735	0.2677		0.729	1.549	5.921	0.415	1.585
21	0.655	0.173	0.663	0.9876	1.0126	0.523	1.477	0.516	1.459	3.778	0.2647		0.724	1.605	5.951	0.425	1.575
22	0.640	0.167	0.647	0.9882	1.0119	0.534	1.466	0.528	1.448	3.819	0.2618		0.720	1.659	5.979	0.434	1.566
23	0.626	0.162	0.633	0.9887	1.0114	0.545	1.455	0.539	1.438	3.858	0.2592		0.716	1.710	6.006	0.443	1.557
24	0.612	0.157	0.619	0.9892	1.0109	0.555	1.445	0.549	1.429	3.895	0.2567		0.712	1.759	6.031	0.451	1.548
25	0.600	0.153	0.606	0.9896	1.0105	0.565	1.435	0.559	1.420	3.931	0.2544		0.708	1.806	6.056	0.459	1.541

For $n > 25$.

$$A = \frac{3}{\sqrt{n}} \qquad A_3 = \frac{3}{c_4\sqrt{n}} \qquad c_4 \equiv \frac{4(n-1)}{4n-3}$$

$$B_3 = 1 - \frac{3}{c_4\sqrt{2(n-1)}} \qquad B_4 = 1 + \frac{3}{c_4\sqrt{2(n-1)}}$$

$$B_5 = c_4 - \frac{3}{\sqrt{2(n-1)}} \qquad B_6 = c_4 + \frac{3}{\sqrt{2(n-1)}}$$

Bibliography

[1] Anaconda inc. anaconda python distribution[computer software].

[2] Minitab inc. minitab 17 statistical software [computer software].

[3] Z. S. Abdallah, M. M. Gaber, B. Srinivasan, and S. Krishnaswamy. Streamar: incremental and active learning with evolving sensory data for activity recognition. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 1163–1170. IEEE, 2012.

[4] Z. S. Abdallah, M. M. Gaber, B. Srinivasan, and S. Krishnaswamy. Adaptive mobile activity recognition system with evolving data streams. *Neurocomputing*, 150:304–317, 2015.

[5] S. Abdullah, N. D. Lane, and T. Choudhury. Towards population scale activity recognition: A framework for handling data diversity. In *AAAI*, 2012.

[6] S. Alqassim, M. Ganesh, S. Khoja, M. Zaidi, F. Aloul, and A. Sagahyroon. Sleep apnea monitoring using mobile phones. In *e-Health Networking, Applications and Services (Healthcom), 2012 IEEE 14th International Conference on*, pages 443–446. IEEE, 2012.

[7] I. Anderson, J. Maitland, S. Sherwood, L. Barkhuus, M. Chalmers, M. Hall, B. Brown, and H. Muller. Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mobile Networks and Applications*, 12 (2-3):185–199, 2007.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, 2013.

[9] A. Anjum and M. U. Ilyas. Activity recognition using smartphone sensors. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 914–919. IEEE, 2013.

[10] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86, 2006.

[11] O. Banos, M. A. Toth, M. Damas, H. Pomares, and I. Rojas. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors*, 14(6):9995–10023, 2014.

[12] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[13] M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl. Actiserv: Activity recognition service for mobile phones. In *Wearable Computers (ISWC), 2010 International Symposium on*, pages 1–8, Oct 2010.

[14] A. Bifet and R. Gavalda. Learning from time-changing data with adaptive windowing. SIAM.

[15] X. Bo, A. Huebner, C. Poellabauer, M. K. OBrien, C. K. Mummidisetty, and A. Jayaraman. Evaluation of sensing and processing parameters for human action recognition. In *IEEE International Conference on Healthcare Informatics (ICHI), 2017*, pages 541–546. IEEE, 2017.

[16] A. Bouchachia. Fuzzy classification in dynamic environments. *Soft Computing*, 15(5):1009–1022, 2011.

[17] M. Bramer. *Principles of data mining*. Springer, 2013.

[18] I. M. Chakravarti and R. G. Laha. Handbook of methods of applied statistics. In *Handbook of methods of applied statistics.* John Wiley & Sons, 1967.

[19] D. Choujaa and N. Dulay. Activity recognition using mobile phones: achievements, challenges and recommendations. In *Proceedings of the Workshop on How To Do Good Research in Activity Recognition: Experimental Methodology, Performance Evaluation and Reproducibility in Conjunction with UBICOMP*, 2010.

[20] M.-S. Dao, T.-A. Nguyen-Gia, and V.-C. Mai. Daily human activities recognition using heterogeneous sensors from smartphones. *Procedia Computer Science*, 111:323–328, 2017.

[21] B. Das, A. M. Seelye, B. L. Thomas, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe. Using smart phones for context-aware prompting in smart environments. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 399–403. IEEE, 2012.

[22] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications.* Citeseer, 2006.

[23] M. Á. Á. de la Concepción, L. M. S. Morillo, J. A. Á. García, and L. González-Abril. Mobile activity recognition and fall detection system for elderly people using ameva algorithm. *Pervasive and Mobile Computing*, 34:3–13, 2017.

[24] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 838–845. IEEE, 2005.

[25] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.

[26] K. Förster, D. Roggen, and G. Tröster. Unsupervised classifier self-calibration through repeated context occurences: is there robustness against sensor displacement to gain? In *Wearable Computers, 2009. ISWC'09. International Symposium on*, pages 77–84. IEEE, 2009.

[27] J. Frank, S. Mannor, and D. Precup. Activity recognition with mobile phones. *Machine Learning and Knowledge Discovery in Databases*, pages 630–633, 2011.

[28] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in artificial intelligence–SBIA 2004*, pages 286–295. Springer, 2004.

[29] J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM, 2009.

[30] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, 2014.

[31] S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante, J. Wåhslén, I. Orhan, and T. Lindh. Proposal and experimental evaluation of fall detection solution based on wearable and depth data fusion. In *ICT innovations 2015*, pages 99–108. Springer, 2016.

[32] R. M. Gibson, A. Amira, N. Ramzan, P. Casaseca-de-la Higuera, and Z. Pervez. Multiple comparator classifier framework for accelerometer-based fall detection and diagnostic. *Applied Soft Computing*, 39:94–103, 2016.

[33] J. B. Gomes, S. Krishnaswamy, M. M. Gaber, P. A. Sousa, and E. Menasalvas. Mars: a personalised mobile activity recognition system. In *13th IEEE International Conference on Mobile Data Management (MDM), 2012*, pages 316–319. IEEE, 2012.

[34] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer. Concept drift detection through resampling. In *ICML*, pages 1009–1017, 2014.

[35] H. Hotelling. The generalization of students ratio. In *Breakthroughs in Statistics*, pages 54–65. Springer, 1992.

[36] D. T. G. Huynh. *Human activity recognition with wearable sensors*. PhD thesis, TU Darmstadt, 2008.

[37] O. D. Incel, M. Kose, and C. Ersoy. A review and taxonomy of activity recognition on mobile phones. *BioNanoScience*, 3(2):145–171, 2013.

[38] L. C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork. Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 5250–5253. IEEE, 2008.

[39] D. Kifer, S. Ben-David, and J. Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.

[40] R. Klinkenberg and I. Renz. Adaptive information filtering: Learning drifting concepts. In *Proc. of AAAI-98/ICML-98 workshop Learning for Text Categorization*, pages 33–40. Citeseer, 1998.

[41] M. Kose, O. D. Incel, and C. Ersoy. Online human activity recognition on smart phones. In *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, pages 11–15, 2012.

[42] H. Koskimäki, P. Siirtola, and J. Röning. Myogym: introducing an open gym data set for activity recognition collected using myo armband. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pages 537–546. ACM, 2017.

[43] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Cell phone-based biometric identification. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on*, pages 1–7. IEEE, 2010.

[44] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[45] N. Lane, H. Lu, S. Eisenman, and A. Campbell. Cooperative techniques supporting sensor-based people-centric inferencing. In J. Indulska, D. Patterson, T. Rodden, and M. Ott, editors, *Pervasive Computing*, pages 75–92. Springer Berlin Heidelberg, 2008.

[46] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell. Bewell: A smartphone application to monitor, model and promote wellbeing. In *5th International ICST Conference on Pervasive Computing Technologies for Healthcare*, pages 23–26, 2011.

[47] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 355–364. ACM, 2011.

[48] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 355–364. ACM, 2011.

[49] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Community similarity networks. *Personal and ubiquitous computing*, 18(2):355–368, 2014.

[50] O. D. Lara and M. A. Labrador. A mobile platform for real-time human activity recognition. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 667–671. IEEE, 2012.

[51] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *Communications Surveys & Tutorials, IEEE*, 15(3): 1192–1209, 2013.

[52] Ó. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada. Centinela: A human activity recognition system based on acceleration¡ i¿ and¡/i¿ vital sign data. *Pervasive and mobile computing*, 8(5):717–729, 2012.

[53] S. L. Lau and K. David. Movement recognition using the accelerometer in smartphones. In *Future Network and Mobile Summit, 2010*, pages 1–9, June 2010.

[54] S.-W. Lee and K. Mase. Activity and location recognition using wearable sensors. *IEEE Pervasive Computing*, 1(3):24–32, 2002.

[55] J. W. Lockhart, T. Pulickal, and G. M. Weiss. Applications of mobile activity recognition. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1054–1058. ACM, 2012.

[56] J. Lötters, J. Schipper, P. Veltink, W. Olthuis, and P. Bergveld. Procedure for in-use calibration of triaxial accelerometers in medical applications. *Sensors and Actuators A: Physical*, 68(1):221–228, 1998.

[57] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM, 2009.

[58] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM conference on embedded networked sensor systems*, pages 71–84. ACM, 2010.

[59] P. Lukowicz, J. A. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. Recognizing workshop activity using body worn microphones and accelerometers. In *Pervasive*, volume 3001, pages 18–32. Springer, 2004.

[60] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the international conference on Multimedia*, pages 787–790. ACM, 2010.

[61] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *Knowledge and Data Engineering, IEEE Transactions on*, 23 (6):859–874, 2011.

[62] S. Matsui, N. Inoue, Y. Akagi, G. Nagino, and K. Shinoda. User adaptation of convolutional neural network for human activity recognition. In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO)*, pages 783–787, 2017.

[63] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *International Workshop on Wearable and Implantable Body Sensor Networks, 2006. BSN 2006.*, pages 4–pp. IEEE, 2006.

[64] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[65] D. C. Montgomery. Introduction to statistical process control. *John Wily& Sons, New York, NY*, 2001.

[66] R. R. Moya. *Multi-tier Framework for the Inferential measurement and Data-driven Modeling.* PhD thesis, 2008.

[67] V. Natale, M. Drejak, A. Erbacci, L. Tonetti, M. Fabbri, and M. Martoni. Monitoring sleep with a smartphone accelerometer. *Sleep and Biological Rhythms*, 10(4):287–292, 2012.

[68] Y. Oguri, S. Matsuno, and M. Ohyama. Activity estimation using device positions of smartphone users. In *International Conference on Network-Based Information Systems*, pages 1126–1135. Springer, 2017.

[69] E. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[70] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):119–128, 2006.

[71] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *Pervasive Computing, IEEE*, 3(4):50–57, 2004.

[72] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 6(28):976–990, June 2010.

[73] S. J. Preece, J. Y. Goulermas, L. P. Kenney, and D. Howard. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. *IEEE Transactions on Biomedical Engineering*, 56(3): 871 – 879, 2009.

[74] Z. Prekopcsák, S. Soha, T. Henk, and C. Gáspár-Papanek. *Activity recognition for personal time management.* Springer, 2009.

[75] J. R. Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan Kaufman, 1993.

[76] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 281–290. ACM, 2010.

[77] S. Ranasinghe, F. Al Machot, and H. C. Mayr. A review on applications of activity recognition systems with regard to performance and evaluation. *International Journal of Distributed Sensor Networks*, 12(8):155–175, 2016.

[78] P. Rattadilok and A. Petrovski. Inferential measurements for situation awareness. In *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2013 IEEE International Conference on*, pages 93–98. IEEE, 2013.

[79] I. Renz. Adaptive information filtering: Learning in the presence of concept drifts.

[80] H. Sagha, S. T. Digumarti, J. d. R. Millán, R. Chavarriaga, A. Calatroni, D. Roggen, and G. Tröster. Benchmarking classification techniques using the opportunity human activity dataset. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 36–40. IEEE, 2011.

[81] S. Sani, N. Wiratunga, S. Massie, and K. Cooper. Selfback -activity recognition for self-management of low back pain. In *Research and Development in Intelligent Systems XXXIII: Incorporating Applications and Innovations in Intelligent Systems XXIV*, pages 281–294. Springer, 2016.

[82] I. Schweizer, C. Meurisch, J. Gedeon, R. B, and M. M. Noisemap: multi-tier incentive mechanisms for participative urban sensing. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, PhoneSense '12, pages 9:1–9:5, New York, NY, USA, 2012. ACM.

[83] R. Sebastiao and J. Gama. A study on change detection methods.

[84] M. Severo and J. Gama. Ubiquitous knowledge discovery. chapter Change Detection with Kalman Filter and CUSUM, pages 148–162. Springer-Verlag, Berlin, Heidelberg, 2010.

[85] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. Smartwatch based activity recognition using active learning. In *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017 IEEE/ACM International Conference on*, pages 321–329. IEEE, 2017.

[86] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga. Defining a roadmap towards comparative research in online activity recognition

on mobile phones. In *Pervasive and Embedded Computing and Communication Systems (PECCS), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[87] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga. A survey of online activity recognition using mobile phones. *Sensors*, 15(1):2059–2085, 2015.

[88] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors*, 16(4):426, 2016.

[89] D. Siewiorek, A. Krause, N. Moraveji, A. Smailagic, J. Furukawa, K. Reiger, F. L. Wong, and J. Shaffer. Sensay: A context-aware mobile phone. In *16th International Symposium on Wearable Computers*, pages 248–248. IEEE Computer Society, 2003.

[90] P. Siirtola and J. Röning. Recognizing human activities user-independently on smartphones based on accelerometer data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 1(5), 2012.

[91] P. Sobhani and H. Beigy. *New drift detection method for data streams.* Springer, 2011.

[92] T. Sohn, A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. G. Griswold, and E. De Lara. Mobility detection using everyday gsm traces. In *Ubiquitous Computing. Proceedings of UbiComp 2006.*, pages 212–224. Springer, 2006.

[93] J. Stork, L. Spinello, J. Silva, and K. Arras. Audio-based human activity recognition using non-markovian ensemble voting. In *RO-MAN, 2012 IEEE*, pages 509–514, Sept 2012.

[94] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart monitor. In *Proceeding of International Symposium on Wearable Comp.* Citeseer, 2007.

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

[96] V. K. Thomas B. Moeslund, Adrian Hilton. A survey of advances in vision-based human motion capture and analysis. *Computing Visual Image Understanding*, 2(104):90–126, 2006.

[97] Y. E. Ustev, O. Durmaz Incel, and C. Ersoy. User, device and orientation independent human activity recognition on mobile phones: challenges and a proposal. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1427–1436. ACM, 2013.

[98] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9. ACM, 2008.

[99] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 06 1945.

[100] G. M. Weiss and J. W. Lockhart. The impact of personalization on smartphone-based activity recognition. In *AAAI Workshop on Activity Context Representation: Techniques and Languages*, 2012.

[101] G. M. Weiss, J. W. Lockhart, T. T. Pulickal, P. T. McHugh, I. H. Ronan, and J. L. Timko. Actitracker: A smartphone-based activity recognition system for improving health and well-being. pages 682–688, 2016.

[102] B. L. Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[103] D. J. Wheeler. *Understanding Variation: The Key to Managing Chaos*. SPC Press, second edition edition, 1993.

[104] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[105] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[106] C. R. Wren and E. M. Tapia. Toward scalable activity recognition for sensor networks. In *Location-and context-awareness*, pages 168–185. Springer, 2006.

[107] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[108] J. Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, pages 1–10. ACM, 2009.

[109] J. Yang, H. Lu, Z. Liu, and P. P. Boda. Physical activity recognition with mobile phones: challenges, methods, and applications. In *Multimedia Interaction and Intelligent User Interfaces*, pages 185–213. Springer, 2010.

[110] Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu. Cross-people mobile-phone based activity recognition. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2545–2550. AAAI Press, 2011.