Unobtrusive weakly supervised activity recognition using a single wearable sensor

William Duffy (BSc)

Faculty of Computing, Engineering and the Built Environment of Ulster University

A thesis submitted for the degree of Doctor of Philosophy (PhD)

July 2020

"I confirm that the word count of this thesis is less than 100,000 words"

Abstract

Assessing the activities an individual performs, through smart-devices or proprietary equipment, is a research area which is currently attracting a high level of interest. Current implementations of these systems use either, subjective methods, for instance, questionnaires administered in medical environments, or automated systems which use pre-captured data. Each of these systems has their drawbacks, with questionnaires limited in accuracy by the patient's recollection and automated systems are limited by the data they are provided.

In automated systems and the wider field of machine learning, a major problem is the collection of accurate and comprehensive labels. Sensors can be worn by individuals and data captured ambiently, however, the difficulty lies in capturing the context for this data, with that context being the activity that the user is currently performing. In many implementations, data-collection experiments are performed, these have practicality issues, and as mentioned, are limited to only the activities captured during the experiment. A potential solution is to reduce the burden of providing a fully labelled dataset by allowing a weaker labelling structure. Normally, a supervised classifier requires that each piece of data is provided with an associated label, however, weakly supervised techniques provide methods for handling inaccurate or incomplete annotations and literature has shown their effectiveness for classifying activity data.

Most people now carry some form of smart device, many of which are laden with an array of embedded sensors. This provides an ideal platform for not only recognition of activities but also the ambient collection of data and a method of user-centric label collection. This work, therefore, focuses on, firstly, providing an overview of the current state of the art in weak supervision and activity recognition. It then investigates methods of collecting labels from users and efficiently applying these labels to large quantities of otherwise unlabelled data. Finally, methods of increasing the ambience of these implementations, by reducing label requests, or otherwise are tested.

These methodologies, when combined, provide a novel and significantly more feasible method of collecting activities from users when compared to the current state of the art. The combination of experience sampling and clustering, along with ideas derived from weak supervision, allowed a significant reduction in overall labels and the removal of the requirement to track beginning and end times of activities, while providing similar classification performance to that of a fully-supervised dataset. Studying the confidence of classifiers also allowed further reductions in the number of labels required and could be used to improve classification performance by propagating labels to unlabelled data points. It was also found that deep learning could be used to remove the requirement for a domain expert for feature analysis and extraction, while maintaining classification performance.

NOTE ON ACCESS TO CONTENTS

I hereby declare that with effect from the date on which the thesis is deposited in Research Office of the Ulster University, I permit

1. The Librarian of the University to allow the thesis to be copied in whole or in part without reference to me on the understanding that such authority applies to the provision of single copies made for study purposes or for inclusion within the stock of another library.

2. The thesis to be made available through the Ulster Institutional Repository and/or EThOS under the terms of the Ulster eTheses Deposit Agreement which I have signed.

IT IS A CONDITION OF USE OF THIS THESIS THAT ANYONE WHO CONSULTS IT MUST RECOGNISE THAT THE COPYRIGHT RESTS WITH THE AUTHOR AND THAT NO QUOTATION FROM THE THESIS AND NO INFORMATION DERIVED FROM IT MAY BE PUBLISHED UNLESS THE SOURCE IS PROPERLY ACKNOWLEDGED.

(William Duffy)

Contents

List of Figures

List of Tables

Abbreviations

ES – Experience Sampling

MIL – Multiple Instance Learning

DBSCAN – Density-based Spatial Clustering of Applications with Noise

G-Means – Gaussian Means

NN – Neural Network

CNN – Convolutional Neural Network

RF – Random Forest

SVM – Support Vector Machines

kNN – K-Nearest Neighbours

FSP – Fixed Sampling Prediction

DSP – Dynamic Sampling Prediction

HMP – Human Motion Primitives Dataset

HAPT – Human Activities and Postural Transitions Dataset

ADL – Activities of Daily Living

MLP – Multi-layer Perceptron

TSFRESH - Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests

Acknowledgements

Firstly, I would like to give a special mention to one of my supervisors' Dr Daniel Kelly, who without him I would never have considered a PhD, nor would I have considered an academic career. I am truly grateful for everything, from my time as an undergraduate to now.

I would also like to give my deepest thanks to my supervisors Professor Kevin Curran and Dr Tom Lunney, who along with Daniel have provided me with an unparalleled level of guidance and support.

I also want to thank my friends within the ISRC, I have never had, nor could I ever ask for a better group of people to be around. There have definitely been some memorable moments during the course of our PhD's and many of them I'll never forget.

I would also like to thank my fiancée and family for their patience and belief throughout the last few years. Their level of support has ensured that I always pushed myself to achieve the best possible work.

1. Introduction

Over the last several years, an increasing number of people have been adopting smartphones. With many of these devices containing multiple embedded sensors, ranging from gyroscopes and accelerometers to GPS and Wi-Fi. The signals produced from these sensors can be used to accurately measure a person's orientation, location, and acceleration and may potentially provide a solution to some of the challenges associated with activity recognition.

Advancements in healthcare and extensions in lifespans have resulted in an ever-ageing population with the elderly becoming a larger percentage of the overall population [1]. This has caused additional strain upon healthcare systems and brought associated rises in operating costs [1]. As a result of this, any methods of reducing these costs could be advantageous, for instance, rehabilitation through activity recognition, patient monitoring [2] and the ability to detect the onset of serious medical conditions, for example, gait analysis in Parkinson's patients [3]. These ideas and their cost benefits for a healthcare system are causing activity recognition systems to receive serious interest from the research community. In addition to its uses within the medical field, activity recognition can also be used for sports, for example monitoring the throw of a dart player and gaming [4].

Currently, a major problem exists within the field of activity recognition This problem relates to the infeasibility in collecting accurate and a sufficiently high enough quantity of labels to allow for effective recognition of these activities. This is an issue which is common among many fields of machine learning, however, activity recognition has the added difficulty of requiring human trials to gather data and the idiosyncrasies that these experiments can bring. These non-free living environments can result in unnatural movements, which can provide inaccurate readings [5], as the person is watched perform activities, and often given specific instructions on the order and duration in which activities should be performed, skewing the capabilities of the final algorithms in recognizing these same activities when the user relaxes and returns to natural environments. As these experiments are not trivial to perform [6], the algorithms are also limited to those few activities that are collected during the experiment and cannot be extended to recognize new activities without re-experimentation.

There are several classes of activities which would be useful to detect. The first of which is ambulatory activities, examples of these include sitting, standing, lying, walking up and downstairs, walking and others. These types of activities are particularly useful in quantifying the levels of activities an individual is currently achieving, allowing people to see when they are being sedentary and if they are reaching their guideline minimum levels of activity which have been set out by government agencies.

The other main class of activities are the Activities of Daily Living (ADL), these activities are used to quantify an individual's ability to live independently. Examples of ADL's are managing money, cleaning, and cooking. These activities are ultimately made up of combinations of ambulatory movements. As they are more complex and generally less cyclic than ambulatory movements, they can be more difficult to accurately classify, especially considering the many idiosyncrasies that people can have in performing the same task.

The previously mentioned ubiquitous adoption of smartphones and technology has given rise to new ideas and possibilities within the field of activity recognition. One idea is to bring activity recognition experiments and data collection into naturalistic environments that allow the collection of user-specific training data and labels that more accurately reflects a participant's natural movement. This idea, while also fraught with its unique difficulties, could be a potential solution to the problem of labelled data in activity recognition and could utilize the embedded smart device sensors to provide data for activity discrimination.

1.1. Research Problem and Motivation - The problem of labelled data within activity recognition

A prominent issue and the focus of this research are the problems around labelled activity data. Having a fully labelled dataset is desirable as it provides the maximum amount of training data for a supervised classifier to learn from. This gives the classifier the best chance at recognizing this activity when new unseen data is introduced. Generally, fully labelled data is expensive to record [2]. This is both due to the time and effort involved in the capture process but also the equipment and setup [3]. The typical setup of an activity recognition capture session involves connecting multiple sensors and placing these sensors at various positions on the participants' body. The participant then performs multiple different activities, each activity performed multiple times, over an extended period of time to ensure sufficient data is available to both train and validate the model. Many studies are also performed in laboratory settings, further complicating the capture of useful data [1]. Even with a completed data capture experiment, the recognition model is limited not only to the activities captured but also to the individuals who performed them. Person dependent activity recognition systems, which are trained to recognize activities of a specific person, have been shown to achieve higher accuracy levels when compared to a person independent system. Person independent activity recognition systems are trained on data from a different set of participants than classification is performed on. This is due to the slight idiosyncrasies displayed by each person in performing the same activity. One of the only feasible approaches to creating a real-world person dependent activity recognition system is to enable users to provide labels for the activities that they perform. However,

with the high frequency of data being captured by modern movement sensors, asking users to accurately label their activities is not feasible. For this reason, weakly supervised methodologies are being proposed to solve the issue. These methodologies take considerably weaker labelling information than normal supervised methods [7]. This weaker labelling implies only a small number of labels relative the number of feature vectors are available, and it may not always be clear which feature vectors these weak labels apply to. Weaker labelling could potentially provide a more feasible method for users to provide the learning model with context about activities they have performed. Generally, unlabelled data is available in greater quantities than labelled data. Weak supervision takes advantage of this and tries to gain information from the relationship between labelled and unlabelled data points [2].

One potential solution to the label problem is to use a data capture methodology, such as experience sampling. Experience sampling requests the user to submit a report over specified time intervals. These intervals can be either time or event based. For example, experience sampling could request a label every 15 minutes of in the case of significant accelerometer readings. While this provides a solution to acquiring labels for sensor data, it could be a source of user annoyance [8]. Multiple methodologies exist for performing experience sampling, ranging from requests at random times with no user consideration to context-aware requests. Dynamic decision-theoretic probing estimates the cost versus benefit of potential user input. These methodologies create context based on estimates of user cost. For instance, the cost of probing a user at certain times of the day, during periods of activity or based on previous response times. These estimates can aid the decision on requesting data from the user. Previous work employed a training mode which, based on events e.g. a user typing, requested how interruptible the user was. This data can then be used to infer the cost of when best to interrupt the user [8]. This idea could also be performed on smart devices to estimate how interruptible the user currently is, for example, if the screen is on, what an application is open, the orientation of the device or if an active call is currently underway.

As smartphones contain the technology to collect data ambiently through their embedded sensors, they would make an excellent platform for ambient recognition of activities. In addition to this, smartphones can collect input from users, which could provide solution to the labelling problem by allowing them to label their own data with the activities that they want to capture. However, this type of approach also brings its own complexities which this thesis aims to identify and resolve.

1.2. Research Objectives

As many more people carry smart devices in everyday life and these devices provide new ways in which data can not only be captured ambiently but, information and context can be gathered from

the user easily. Using these smart devices, we can empower the user to have an active role in the creation of their own personal activity recognition model and in the development of activity recognition systems. While it is hoped that the user's active role will have advantages from the standpoint of system development, it may also increase their overall understanding and importantly their trust in an AI. While this can have its drawbacks, such as excessive requests undermining the user's interest and the overall importance of the request made, it provides a chance for free-living activity recognition experiments.

With these considerations, the main objective of this thesis is to prove that within a controlled environment, a recognition system using user acquired weak labels can achieve results like those seen in traditional systems without the need for intrusive and time-consuming data collection experiments.

1.2.1. Research Questions

In order to ensure that the main objective of this thesis remains on track, the objectives must be broken down into attainable research questions

1.2.1.1. Question 1

> *Can weak supervision techniques successfully recognize activities when given significantly weaker labelling information? And if so, are there any limitations on the type of activities which can be successfully recognized?*

Question1 implies an important issue, because if the associated weak labelling recognition approach does not produce results comparative with that of a fully-supervised system, then it does not provide a basis for a viable solution. To ensure that the implications of asking question 1 is fully understood, and its application potential determined, a literature review is performed both of activity recognition and weak supervision systems.

It is not expected that these weaker systems will provide the same levels of performance seen by fully-labelled systems and it cannot be expected that they will. Furthermore, it is postulated that more complex activities, such as activities of daily living, are going to be more challenging to classify. However, if the weaker system can provide performance close to that seen in fully-labelled systems, then it provides a feasible method of activity recognition, something which current fully-labelled systems do not.

The following objectives will be executed to answer this research question:

- Find appropriate datasets which provide examples of both locomotive and ADL activities.
- Investigate methods of transforming weak labels to unlabeled data

- Apply methods to the collected datasets
- Test the data against classifiers which have been proven in recognizing activities
- Analyze results and compare to state of the art

1.2.1.2. Question 2

Are there techniques we can employ to further increase the feasibility of these experiments?

Once it has been proven that activity recognition systems can detect activities when provided with weaker labelling information, further improvements can be made to increase either the recognition performance or the feasibility of the systems. Examples include, detecting phenomenon other than activities that provide context to what activity or when it is being performed within a time-series. We also seek to ascertain if these additional contexts can be used to increase recognition performance or otherwise.

To answer this research, question the following objectives will be executed:

- Implement datasets from previous research question where appropriate
- Firstly, investigate methods to reduce the overall number of user requests made by the activity recognition system
- Using these methods, re-evaluate the performance on classifiers to ensure that overall performance has no dropped given the opportunity for increasingly noisy labels
- The next step is to investigate any phenomena which occur within the data which can be detected without additional supervision. This could be through detecting new activities and the transitions between them
- Next, investigate methods which will increase the efficacy of the gathered labels.
- Finally, test these methods on classifiers to ensure that they are not causing reductions in performance when compared with the state of the art and the previous research question.

1.2.1.3. Question 3

Can we detect activities using a completely feature-free approach?

Establishing a methodology that (1) does not require controlled data collection and (2) one in which experiments do not need to be conducted in order to design/evaluate features for specific activity types, would provide the foundations for the potential development of a tailorable activity recognition system that could be trained on activity types decided by the user.

The following objectives will be undertaken to answer this research question:

- Investigate the state of the art in convolutional neural networks as they provide the ability to recognize activities without the need for features
- Test these deep features and compare them with the normal state of the art in feature extraction.
- Test the deep features when given weaker labelling information
- Analyze the results and ensure that the performance is still acceptable given the reduction of both features and labelling burden.

1.3. Thesis contributions

The main contributions of this thesis are through the unique combination of weak supervision techniques with activity recognition systems. The techniques proposed in this thesis will allow activity recognition to be performed in free-living environments with the user providing labels to the system.

The main contributions and methods provided are:

1. A novel method of collecting and applying labels to activity data using techniques derived from weak supervision and clustering (Chapter 3).
2. Two novel methods of reducing the number of user-requests required when collecting labels from users for activity recognition (Chapter 4).
3. A novel method of recognizing the transitions between activities (Chapter 5).
4. Two novel methods of propagating activity labels from known labelled points to unlabeled points (Chapter 6).
5. A novel method of automatically generating feature vectors from raw activity data and weak labels (Chapter 7).

1.4. Thesis outline

The outline of the chapters contained within this thesis are:

Chapter 2 – Literature review

This section will provide an extensive review of the state of the art in both activity recognition and the machine learning techniques relevant to this thesis. It will discuss the main types of activities which are useful to track in activity recognition systems. The chapter will also investigate the problems with the current data collection methodologies, experiments and approaches which may be useful to collect labels in a less intrusive manner, as well as, the state of the art in sensing technologies for discrimination of activities.

The review will then branch away from the problem background into machine learning, where methods of signal cleaning, signal processing and classification will be discussed. Of these classifiers, the most effective for activity discrimination will be highlighted. Methods of weak supervision will also be investigated.

Chapter 3 – Experience sampling for labelling collection

This chapter will focus on the problems associated with collecting labels through user-reporting. It will discuss the datasets to be used, which will be a mixture of both locomotive and activities of daily living. The chapter will then discuss methods of extracting features from these datasets and the methods in which simulated label collection will be performed. A discussion will also be carried out relating to a prominent issue found when collecting labels from users, how we apply the label to correct feature vectors. Multiple methods of performing this will be reviewed, including clustering methods and otherwise. Classification will then be performed on the data and its performance compared with a normal fully labelled dataset.

Chapter 4 – Reducing the overall number of user-requests

This chapter will focus on methodologies which can be applied to reduce the number of user-requests made in user-trained activity recognition systems. It uses the principle that, as a user provides labels, knowledge will be gained into what the upcoming activities are and therefore, if a prediction system is confident in its prediction it can forgo requesting data from the user and instead use its own prediction. The chapter will investigate two possible methods of performing this and will discuss optimising a threshold value of prediction confidence and its effects on recognition performance and the number of requests made.

Chapter 5 – Detecting the transitions between activities

This chapter will investigate the methods used in predicting the transitions which occur between activities. This means detecting the moment when an individual transition from one activity to another, for example, from sitting to standing. Being able to detect these transitions can have benefits from a medical or rehabilitation standpoint, but, in the case of weak supervision in this work, they are leveraged to segregate moments of activity when additional labels are not available to do this automatically. Like the previous chapter 4, it uses recognition confidence scores and an algorithm for performing transition detection is presented and the results of this system analysed.

Chapter 6 – Label propagation for increasing classifier performance

This chapter will investigate the propagation of labels from known labelled data points to unlabelled data points. If a weakly-supervised activity recognition system can effectively perform this task, then it could allow for a significant increase in the number of labels and potentially increased classifier performance. However, performing this incorrectly could result in many incorrectly labelled data points, which will harm recognition performance. Several methods are employed to try and achieve this, and the results reviewed in this chapter.

Chapter 7 – Reducing the requirement for expert feature analysis

The work is this chapter is intended to reduce or remove the requirement for expert feature analysis. Feature analysis is when defining features are extracted from the raw data to allow the different classes, or in the case of this thesis, the activities to be discriminated effectively by a recognition system. Poor feature selection will result in data which is difficult to segregate and overall poor recognition performance, however, as this needs to be performed by someone with previous domain knowledge, it adds further complication to self-trained activity recognition systems as the system will be forced to use a set of generic features which may not translate well to all activities.

To get over this problem, methods of automated feature extraction exist, such as convolutional neural networks, these methods will be explained and tested within this chapter.

Chapter 8 – Conclusion

This chapter will provide a summary of each of the previous chapters and discuss the contributions provided within this thesis as well as provide indications of future work which could be applied to the methods within this thesis.

1.5. Publications

Several publications have arisen from this work, including peer-reviewed conference proceedings, book chapters and journal articles, each of which will be shown below.

1.5.1. Conference papers

Of the below conference papers, the first contributes to chapter 4 and the second contributes to chapter 3 and the last contributes to chapter 6.

Duffy, W., Curran, K., Kelly, D., Lunney, T. (2018a) Reducing the intrusion of user-Trained activity recognition systems, in: 29th Irish Signals and Systems Conference, June 21-22 2018, Queens University, Belfast, ISSC 2018. https://doi.org/10.1109/ISSC.2018.8585343

Duffy, W., Curran, K., Kelly, D., & Lunney, T. (2018). Addressing the Problem of Activity Recognition with Experience Sampling and Weak Learning. Intelligent Systems Conference (IntelliSys) 2018,

6-7 September 2018, London, United Kingdom.

Duffy, W., Curran, K., Kelly, D., Lunney, T. (2019) User-trained activity recognition using smartphones and weak supervision, in: 30th Irish Signals and Systems Conference, June 17-19 2019, ISSC 2019. https://doi.org/10.1109/ISSC.2019.8904948

1.5.2. Book chapters

The following book chapter contributes to chapter 6.

Duffy, W., Curran, K., Kelly, D., Lunney, T.(2018c) Addressing the problem of activity recognition with experience sampling and weak learning, in: Advances in Intelligent Systems and Computing. 6-7 September 2018, London, United Kingdom, pp: 1–13, https://doi.org/10.1007/978-3-030-01054-6_86

1.5.3. Journal Articles

The first journal article contributes to chapters 3 and 5. The second journal article is currently in preparation for submission to Pervasive & Mobile Computing which relates to chapter 7.

Duffy, W., Curran, K., Kelly, D., Lunney, T. (2019) An investigation into smartphone based weakly supervised activity recognition systems. Pervasive & Mobile Computing, Vol. 56, No. 1, pp:45-56, https://doi.org/10.1016/j.pmcj.2019.03.005

Duffy, W., Curran, K., Kelly, D., Lunney, T. (2020) Convolutional Neural Networks for Automated Feature Extraction in Weakly Supervised Activity Recognition Systems. Pervasive & Mobile Computing [In Preparation]

2. Literature Review

As this thesis encompasses several different concepts to create a naturalistic activity recognition system, the review of the literature must be carefully considered to ensure that a comprehensive state of the art is captured.

Firstly, some problem background will be discussed, including the core concepts of activity recognition experiments and the current problems with capturing labels. Next, the main uses of activity recognition will be captured as this allows us to narrow down our criteria into specific activities that would be advantageous to capture.

Next, the current methods of data capture for activity recognition will be detailed, along with the methods which have been used to capture these labels in less intrusive ways.

At the point in the review, most of the background information around label and data collection for activity recognition will be complete. Now a deep dive will be performed into the many ways in which data can be classified and, importantly, the different methodologies useful to weak and semi supervision.

2.1. Problem background

The current approaches to activity recognition operate by analysing data received from sensor arrangements to make predictions on new unseen data. Prior to a machine being able to recognize a specific activity it must be given the context for which each signal relates to, for example, a waveform signal may indicate a cyclic motion such as walking, and a flat signal may indicate a signal which contains no movement. However, without being provided with this context, a system has no way of making predictions on future data. Gathering this context, otherwise known as labelling or annotating the data, is a difficult, expensive process. It requires time-consuming setups, for instance, a typical data collection study will require gathering participants, obtaining and calibrating appropriate sensors and having the participants perform a sequence of activities [9]. The activities must be performed in enough quantity to provide adequate data for training of supervised algorithms.

Many of these experiments for capturing data and context will be performed within non-free-living environments such as research centres and laboratories. These rigid surroundings and constant observation can put unnecessary pressure on participants, resulting in unnatural movements [10]. Another issue with activity context is that some people may perform several activities at one time [11], such as using a phone and walking.

Extra complication is added to the task as all data received from sensors are subject to unwanted changes, for example, noise [12], sensor placement [13], and orientation [13] can all cause recognition issues, if an activity recognition system has been trained on an individual with the sensor attached firmly upright to their waist and then tested by an individual with the sensor upside down in their pocket it is likely to cause problems [11]. As activity recognition systems are generally implemented on battery-powered devices, we must be mindful of the limited power and compute capabilities. Intense compute operations may result in battery drain and a negative experience for the user of the device [11].

Previous literature has investigated multiple methods of activity recognition, including supervised, unsupervised and semi-supervised techniques. Supervised learning techniques require comprehensive annotation of datasets and sufficient training data for reasonable levels of accuracy [6] whereas semi-supervised systems try to extract supervision signals from otherwise unlabelled data by exploiting its similarity to known labelled data [6].

2.2. Uses of Activity Recognition

Building from the issues raised in section 2.1, a review of the main uses of activity recognition is required. This will allow us to discover the main activities that would be useful to capture in our experimentation.

Multiple uses for activity recognition exist within several fields. These fields include the medical domain, gaming and sports. The most notable of these domains is the medical field, where financial pressures of an aging population are pushing research in the direction of early detection.

Within the medical field, there are multiple areas which could benefit from sensor arranged activity detection. These are:

- **Rehabilitation –** An activity recognition system has been used to monitoring how medication affects people with Parkinsonian tremor [14].
- **Monitoring** - Certain patients, such as those with heart disease and diabetes, can benefit from a carefully defined exercise routine. These activities can be everyday life activities, for example running and walking [15]. Detection of these activities could be automatically performed using bodily attached sensors which could potentially provide practitioners with details as to how well patients are sticking to their routines. Other methods of monitoring using activity detection include fall detectors [16], monitoring of the health status of COPD patients [17] and detecting activities of daily living [2].

- **Ambient Assisted Living –** This includes methods which can provide patients with increased independence by tracking their vital signs and automatically notifying the emergency services in certain events [18]. This section could also include the automated tracking of abnormal activities, in the case of certain diseases such as dementia these abnormal activities could indicate a problem [15].
- **Prevention –** By performing a detailed analysis of the movement an individual is executing and analysing for changes in quantity or ability to perform these activities, it may be possible to detect and prevent non-communicable diseases [19].

However, these examples only provide a small insight into the multiple ways in which activity recognition systems may be used.

2.3. Activities to detect

Physical activity is defined as any bodily movement which has been produced by skeletal muscles that result in an energy expenditure which is above the normal resting level [20]. The goal of an activity recognition system is, therefore, to detect, firstly, what activity is currently being performed and secondly how long that activity was being performed for.

We can think of the activities being segregated into several categories, including locomotive, sports and activities of daily living, each of which brings their own complexities.

2.3.1. Locomotive activities

Locomotive activity is a class of activities that is based around basic human movements such as standing, sitting and walking. Locomotive activities can be very helpful in determining an individual's level of activity. Locomotive activities can show if an individual is sedentary or active [21]. Monitoring these activities can therefore be useful in ensuring that individuals are reducing their chances of diseases by leading an active lifestyle [22]. These activities, along with sporting activities, can also be useful when estimating the number of calories an individual is burning [23].

Many of these activities contain distinct movement patterns that can occur at regular frequencies. Due to these distinctive patterns, there are many examples in the literature where locomotive activities have been classified with high accuracy [23], [24].

2.3.2. Activities of Daily Living (ADL)

The ADL are a sequence of activities proposed in the 1950s by physician Sidney Katz [25]. They are described as a measure of an individual's functional status and range between basic ADL and instrumental ADL. The basic ADL are a requirement for an individual to function independently, examples include dressing, bathing, walking and self-feeding [26]. While the basic ADL are a

measurement of an individual's ability to function independently, the instrumental ADL are used to measure an individual's capability of living in a community setting. Examples of instrumental ADL are managing money, cleaning, preparing meals and using the telephone [26]. In current practice, instrumental and basic ADL are measured using a physician-administered questionnaire. Public datasets are available containing ADL and locomotive movements [9], [27]. Previous literature has successfully classified ADL data, and they provide a solid base solid base for testing future developments.

As ADL are significantly more complex, and in some cases may contain locomotive activities they can be more difficult to classify [24]. Current implementations have also found that activities of daily living classification performance are below that which could be considered reasonable for real-world implementation [24].

2.3.3. Sports

The other category of activities which can be tracked is sports. Some of these activities will likely have some overlap with the locomotive movements, such as walking and running [28]. As many of these movements, are performed in a fixed, repetitive cycle, and are performed by most individuals using the same general movement pattern they should be easier to classify than the activities of daily living.

Like locomotive movements these can be used to classify an individual's current levels of activity and the number of calories they are burning. However, activity recognition systems could also be used to identify problems with athletes, such as detecting collisions [29] or early indicators of injury [30], or providing feedback on an athletes strengths and weaknesses [30]–[32].

2.4. Traditional collection of activities and labels

Within the medical field, quantification of a person's activity levels is usually performed by self-reporting on questionnaires or by occasionally observing the individual [33]. Both of these approaches have their own difficulties with questionnaires providing a large amount of data with questionable reliability and observation being time-consuming [34]. With these issues, any methods of automated activity recognition would be advantageous, and a feasible approach for real world settings when using widely available sensors in conjunction with machine learning.

When considering the development of a machine learning model, we need to consider how ground truth is captured for the training of the model. In current literature, this is performed through directed experiments where participants are requested to repetitively perform a certain movement under the supervision of the investigator [10].

Data collection will generally start just before the individual begins the movement; they will then repeat the movement until they are requested to stop or get to the end of a prescribed course. For example, they may have to walk until the investigator requests. The data produced will then be trimmed to remove any extraneous data, for example, any data captured after the activities were stopped, otherwise the labels may incorrectly refer to a completely different activity. While this is happening, the investigator will be noting the exact times when activities start and end. Once this process is completed, the investigator will collate the labels with the raw data from the sensors. This data collection protocol is repeated for each participant in the study.

As discussed, performing these label collection experiments is a non-trivial task that is costly and time consuming. If any issues occurred during data collection that may have affected the quality of the data, for example, the sensor stopped working, then the entire data collection process will likely need repeated for that participant. Another major issue is when an additional activity class is desired in order to expand the number of activities that the model can recognize. This would require running an additional data collection stage, with the participants asked to perform multiple repetitions of the new activity.

2.4.1. Unobtrusive collection of data

While the data collection protocols described in the previous section can provide accurate labels, the protocol conditions are not entirely representative of natural human movements or natural environments [1]. A method which can passively capture the activities and their associated labels without requiring direction or monitoring by investigators and with the participant in their natural environment would produce a model which more accurately represents real world activity. The model could, therefore, be more effective when deployed in real world settings and achieve more accurate classification results when tested on real world data.

Smart devices present a potential opportunity to passively capture activity data in real-world scenarios. These are an attractive option for activity recognition systems as they contain a large number of sensors, including accelerometers which have been used successfully for activity recognition [34]–[36]. They also contain other potentially useful sensors such as gyroscopes as well as communication interfaces like Wi-Fi [15]. Using smart devices makes unobtrusive collection of data relatively simple as most people already carry the devices everywhere, and the sensors can run in the background without affecting the device's user experience.

However, there are still issues with capturing activity data from smart devices. For example, some activity classification techniques can be sensitive to sensor positioning and orientation relative to the participants body. If a user is required to keep attention of the orientation and/or position of their

device then this could be intrusive [37]. This could also be difficult if the individuals are elderly or disabled [21].

As unnatural data collection environments can cause inadvertent abnormal movements, Bayat et al conducted experiments to reduce the intrusion of these data collection protocol by performing a semi-naturalistic data collection[38]. Specifically, they removed the observers and allowed the participants to record their own start and end times of the activities. The activities they performed were controlled by a worksheet which they were given prior to the experiment starting [38]. This experiment showed that users are capable of providing their own labels while maintaining high levels of recognition accuracy.

However, uncontrolled label collection can cause problems which don't occur within laboratory settings. The first of which is noisy labelling where the annotator is inadvertently giving the system poor quality labels, for example, not recording the exact start and end times of activities. This can result in feature vectors having the wrong label and these will actively confuse the decision plane of a classifier causing a reduction in classifier performance [19]. Other problems include short activities being very unlikely to be annotated and confusion of activities by the annotator, for example, confusion between lying and sitting on reclining chairs [19].

In summary, when compared with laboratory experiments, which ideally require as many participants as possible across a wide spectrum of people. Utilizing the sensors within the devices that people carry with them every day makes it much easier to collect a large amount of data [23]. However, the great challenge comes from the difficulty in labelling this data and ensuring that any labels which are provided are accurate [6].

2.5. Experience sampling

With the collection of labelled data being such a significant problem, experience sampling is a type of diary study technique which captures labels from users directly [8]. Capturing data from users, while potentially frustrating for some, provides a method of avoiding complex data capture experiments.

Within many scenarios this would mean giving the user a questionnaire, but to collect activity labels the requests could come from their smart device. This would most likely take the form of a prompt on the device's screen for input.

Multiple methodologies exist for applying experience sampling. They range from simple randomly sampled methods to context-aware systems. Some example sampling methodologies are:

- Random sampling – This method makes user requests at random times with no regard to the usefulness of the data to be captured.
- Landmark sampling – Landmark attempts to use contextual cues before making a request, for example, if the user is in an unknown location.
- Uncertainty sampling – This model makes a request when it finds data which it has very low certainty in.
- Decision-theoretic – This model compares the user state with the expected gain from the captured data, allowing a cost versus benefit analysis to be performed before a user request.
- Dynamic decision-theoretic – An extension to decision-theoretic, allowing the caching of user-states for construction of multiple models.

Issues exist with many of these implementations. Random sampling, landmark and uncertainty sampling do not consider user-state before making a request, which could lead to a request being made at an inopportune time. Dynamic decision-theoretic has been shown to perform the best of the mentioned implementations [8]. In the context of activity recognition, unless we have contextual information about the activities then we are forced to use random sampling to obtain initial activity information from the user.

Previous work into context-aware experience sampling systems has used Bayesian networks [39]. One of these works made predictions on user interruptibility to infer the expected cost of interrupting the user with a request. However, validation was not performed [39].

It has also been found, in the context of emails, that importance does not dictate the speed which a reply is received. Instead, individuals are more likely to respond to social emails [40]. It could be inferred from this that a request for data from an activity recognition system, regardless of its importance, is likely to be ignored if the user is currently engaged to social applications. The speed at which people respond to emails has also been predicted, with strong indicators including the duration of user online time and last outgoing message time [41]. This responsiveness speed could be used to assess opportune user request times.

2.6. Sensors

Historically, activity in the form of steps has been collected using simple pedometers, these, however, provide no benefit in recognizing more complex activities. Accelerometers record the acceleration signals produced by the user, on a tri-axial device it will produce acceleration values for the X, Y and Z axes. This acceleration data can prove useful for the classification for some activities. However, a drawback is that activities which have similar acceleration profiles can be difficult to differentiate [42].

To reduce the effect of movements with similar acceleration profiles other sensors have been included, for instance, gyroscopes [42]. Some wearables and other smart devices are being equipped with biosensors such as photoplethysmograms, which can be used as a basic ECG, allowing detection of heart rate and rhythm variability [43], estimation of blood pressure [44]. The camera on smartphones has even been used as a rudimentary pulse oximeter [45]. Even though these types of device provide a feasible approach to unobtrusive activity recognition, they can run into issues with the diversity between sensors and their capabilities in different brands and models [46].

The placement of the sensors can have a critical impact on the classification ability of certain activities; however, ergonomics must also be considered. Sensors have been placed on the chest [27] and waist [47] to record movements which involve the whole body. Wrist-worn devices have also been used to diagnose sleep apnoea [48]. Orientation can also be a problem for many activity recognition systems, if they have been training to classify activities from a certain orientation then a different orientation of the sensing device can cause classification problems [37].

Other sensors exist which may be useful for activity recognition, however, not all of these will be useful within a wearable device. Some additional examples include:

- Magnetometers – these sensors use the Earth's magnetic field to detect the devices orientation. These can be useful for detecting an individual's posture during activities [49].
- Gyroscopes – these sensors detect pitch, roll and yawing movements. They can be useful for detecting activities which contain some kind of rotation [50].
- Temperature sensors – these sensors return the current temperature, in an activity recognition setting they will return the skin temperature [51].
- Electrocardiogram – also known as ECG, these sensors are used to track electrical activity across the heart [52].
- GPS – this sensor can be used to detect where on earth a person is and make estimates about the speed they are travelling; this speed can then be used to identify what activity they are performing. For example, if they are doing 60mph it is likely they are driving a car [53].
- Camera – image processing has also been used for activity recognition [54].

2.7. Data Pre-processing

Before any sensor data can be utilized for training a classifier, it is typical for the signal to be pre-processed. Some of the common pre-processing steps include signal cleaning, separating the signal into windows and extracting meaningful features.

2.7.1. Filtering

Filtering is performed to remove or extract certain information from a signal, such as using a Butterworth filter to reduce the effect of noise on sensor data [55]. Certain signals can also be isolated using filters such as a low-pass filter to separate the gravity acceleration from body acceleration [38].

2.7.2. Windowing

A commonly performed step prior to feature extraction is windowing, this segregates the data into smaller moments of activity.

Multiple methodologies exist for splitting the data [56]. Ideally, we would split the windows into the start and end times of the activities, however, this can be a difficult method to achieve as it may not be clear when one activity starts or another ends [56]. Another common approach in activity recognition is to use windows of fixed length intervals. This interval can either be sensor-based or time-based, in sensor-based the window will stop after a pre-defined number of sensor event whereas time-based will close the window after a set time. Although these approaches eliminate the need to know the exact start and end times of activities, it presents different problems. If the window is too long then we can have multiple activities within the same window and if the window is too short we may have insufficient information for successful classification , such as only collecting a partial activity [56]. Many approaches will use an overlapping window to overcome these issues [27].

Generally, window size is randomly selected and very little literature exists in selecting the optimal window size, but smaller windows provide the best recognition results [57].

2.7.3. Feature extraction

As each window of raw data can contain many data points from each axis of a sensor, this will result in extremely high dimensional data. This high dimension data can ultimately increase the amount of computing required for both training and classification. The extracted features are intended to provide better discriminatory information for a classifier than the raw data while also reducing dimensionality. These features are derived from the data in multiple ways, including time and frequency-based analysis. When a window of data is reduced into a smaller number of features this is known as feature extraction and the resulting new data is its feature vector. One feature vector will be derived from each window and its intention is to provide a comprehensive view of the data for discrimination of classes while reducing the number of dimensions the classifier must consider for training.

2.7.3.1. Time-domain features

Time-domain features show how a signal changes over a given time-frame and are mainly gathered using statistical methods such as variance [58], [59], mean value [10], [60], [61], or the number of

peaks [62]. He et al, attempted to classify several locomotive activities, including standing still, jumping, walking and running. They used an autoregressive model of the time-series and successfully increased the classification performance over standard time and frequency domain features [63].

2.7.3.2. Frequency domain features

Frequency-domain features provide information about which frequency bands a signal is in, and is gathered by functions such as Fast Fourier Transform [64] and Spectral Energy [65]. These types of features have been shown to perform the best for activity recognition, but they can be computationally expensive to calculate [66].

2.7.3.3. Deep learning feature recognition

Through the use of convolutional neural networks (CNN), features can be derived from images automatically [67]. As an image can be considered a 2-dimensional signal, this same principle of feature extraction can be used for accelerometer signals [68], [69].

2.7.3.4. Other feature extraction methods

There are also other methods which can be used to generate features, including heuristic methods such as signal vector magnitude [47], [70], and signal magnitude area [65], [71].

Generally, time and frequency domain features will require handcrafting to define the best features to collect from a signal [72]. For example, some features may be more suited towards detecting what song is playing rather than what activity a person is performing.

2.7.4. Dimensionality reduction

After feature extraction has been performed, further dimensionality reduction can be performed. Reducing the number of dimensions can both improve the compute performance of classifiers and in some cases increase the performance of classification. The two categories of dimensionality reduction are feature transformation and feature selection [73]. An important note is that to be useful in reducing the overall compute complexity of the algorithm, the features to be derived must also be calculable easily. Investigations into the features which both provide the highest levels of discrimination between classes that can still be executed quickly have also been performed [74].

2.8. Classification of data

Machine learning has two main categories, classification and regression. They are both intended to resolve different issues, with regression being used to predict a numerical quantity. For example, making a prediction of the exact number of degrees a person can flex their arm would be a regression problem. Conversely, classification is making a prediction of a target label. For example, making a

detecting that someone is walking versus any other activity is classification. The algorithm which makes the classifications is known as a classifier.

For activity recognition systems, classification is generally the method used [75] and over the next few sections, several different methods of training classifiers will be discussed. The first is supervised learning, where each individual feature vector within the training set has been provided its own label and these labels plus their associated data are provided to the classifier [76]. The next learning method is semi-supervised training, where the training set contains some labelled feature vectors, but also contains a pool of unlabelled data [77]. A standard supervised classifier cannot use unlabelled examples; however, semi-supervised techniques try to gain additional insight from their relationship to the labelled feature vectors. Finally, unsupervised learning analyses feature vectors that have no labels assigned to them [76].

With any predictive model, trades-offs and problems exist. Many machine learning classifiers will have parameters which impact how well the decision plane will fit to the data and these parameters will affect the trade-off between bias and variance [78]. Bias is the ability for a classifier to accurately learn the patterns in the data, if it is unable to accurately identify these patterns then it is said to have a high bias and it is underfitted, which may negatively affect predictive performance. Variance is the difference between how well the training and test set fit together. If we configure a predictive model to have very low bias and fit the training set extremely tightly, then we run the risk of having poor performance if there is a high variance between the training and test sets, this is known as overfitting. A core problem is finding a bias configuration which allows the model to generalise well and effectively manage variations in the data between the training and test sets. Another issue is the curse of dimensionality, which is when error counterintuitively increases with an exponential increase in the number of features [79]. This increase in features will also correspond with an increased computational complexity. Therefore, it is important to consider each of the features chosen and to ensure that it is actively supporting the predictive capabilities of the model.

2.8.1. Evaluation methods

Multiple methods exist for testing a predictive model against unseen data with pre-collected datasets. These methods include:

2.8.1.1. Train test split

A train test split is when the dataset is split into two distinct sub-sets which do not change throughout the testing. Normally, two thirds of the data will be used for training and the remaining third used for testing [80].

2.8.1.2. Cross validation

This method iteratively works through the dataset by splitting it into sub-sets so that all of the data has been training and tested on separately. The two main methods of performing this are k-folds, where the dataset is split into "k" folds, and for each iteration, one-fold is held out as testing data and the rest of the dataset is used for training [80].

The other method of performing cross-validation is leave-one-out, which is when a dataset is available which contains multiple participants. Each of these participants is considered a single fold, and the classifier will be trained on X-1 folds, leaving the final participant for testing. This is repeated until all the participants have been tested on. This method can be useful to reduce the bias incurred when the classifier is provided with training data from the testing participant which can occur in K-fold cross validation [81].

In each of these methods, the average performance of each of the folds is taken as the final performance. Since cross-validation will ultimately test on all the data, whereas, the train test split mentioned in the previous section will only test on the test data, it is usually the better of the two methods for showing overall performance.

2.8.2. Performance metrics

In order to measure the performance from the classifier, several different metrics can be used including

- Accuracy – This is the total number of correct predictions divided by the total number of predictions which have been made [82]. This metric only useful in relatively balanced datasets, as a major imbalanced dataset can have the appearance of high accuracy if all predictions are for the majority class.
- Precision – This metric is the total number of true positives divided by the total number of positives the model has predicted [80].
- Recall (sensitivity) – This is the number of true positives divided by all the samples which the model should have identified as positive (true positives + false negatives) [82].
- F1-score – This is the mean of both precision and recall, and it provides an identification of predictive performance regardless of the class balance of the dataset [80].

Over the following sub-chapters, several different methods of this classification will be looked at in more detail.

2.8.3. Supervised

Supervised learning is when each feature within the training set has been provided with an associated label. This provides the optimal amount of information for the classifier to learn from, however, each of the problems associated with machine learning exist.

In order to reduce the bias-variance trade-off, a sub-set of the dataset is used to tune the parameters of the model. This is normally done sequentially through a GridSearch, which will try different combinations of parameters until it finds the optimal values. This can also assist in reducing the overall effects of noise within the training set.

The next chapters provide detailed information on several important supervised classifiers.

2.8.3.1. Support Vector Machines (SVM)

Support Vectors are defined as data points which lie on the transition between two classes. The SVM algorithm uses support vectors to create a decision plane in the position that achieves the greatest separation between the two classes [83]. When compared with conventional classification methods they can handle very large feature spaces without incurring a significant performance impact [84]. As this work primarily focuses on activity recognition within naturalistic settings with minimal low-powered sensors, any savings to compute or battery performance is advantageous.

The SVM can create its hyperplane based on the selected kernel, there are multiple kernels available including linear, polynomial and radial basis function (RBF). Each of these have been used for activity recognition in the literature but RBF appears to provide the best performance. Each of these kernels contains several parameters which must be tuned depending on the situation. The RBF kernel contains two critical parameters referred to as gamma and C. These parameters are used to control the strictness of the hyperplane when training. While an SVM could create a hard margin when splitting two classes, the idea of a soft margin where feature vectors from the wrong class may be ignored and placed in the incorrect side of the margin can lead to better generalization. The C value decides how much impact each support vector has upon the margin, a higher value means the algorithm will try to tune the hyperplane, so each feature vector is on the correct side. The gamma function controls how much weighting feature vectors close to the hyperplane exert on the boundary, the higher the gamma the more influence a close feature vector has on the hyperplane than one which is far away.

SVM has been used extensively throughout the literature for activity recognition [11], [23], [85]–[90]. Some works have tried to improve the performance of SVM classifiers used on accelerometer data by performing a discrete cosine transform before classification [91]. Reductions in the computational cost of SVM classification has also been shown on low-powered devices such as smartphones by adapting them to use fixed-point arithmetic [85].

In activity recognition experiments, SVM have been shown to classify locomotive activities such as walking, using stairs, standing, sitting and laying effectively [85]. When tested on other mixed activities, including driving, running, bicycling and using stairs the performance of the SVM could be improved if sensor location data was available. The different locations included different jean pockets e.g. front or back pocket or a coat pocket [86].

When tested against the PAMAP and the MHEALTH datasets, SVM was compared against several other classifiers, including ADABOOST, Neural Network, Random Forest and Boosted Decision Trees. SVM provided the highest average results across both of these datasets [90].

2.8.3.2. Neural Networks

There are multiple different configurations of neural network which can be used for classification. The standard configuration, shown in Figure 1, is a series of input nodes which are fully connected to a series of output nodes. These final nodes will represent the final output, in the context of an activity recognition system, it will be the output class.

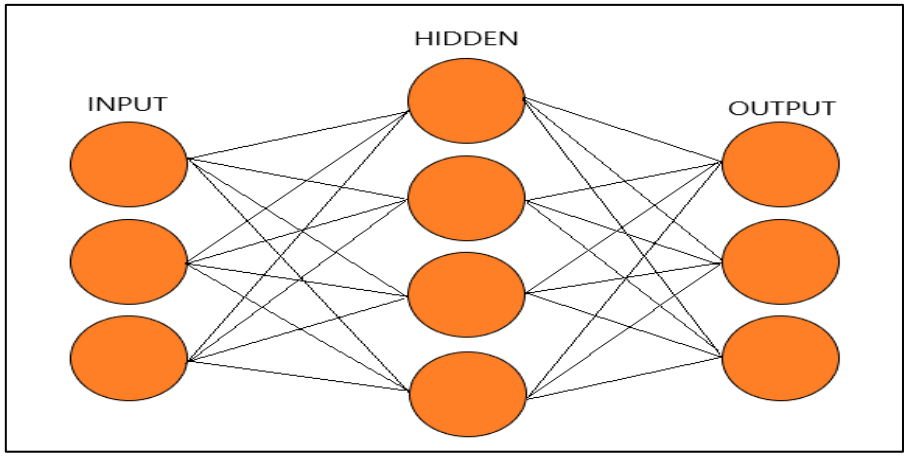

Figure 1 - Neural Network

When data is passed into the network, for each node, the data is multiplied by the weights and summed together. This summed value is known as the node's summed activation, the output of the node is then obtained by passing this summed activation into an activation function. There are multiple examples available of activation functions, each of which is intended to introduce non-linearity into the network. Some examples are ReLu and Softmax.

$$R(z) = \max(0, z)$$

Equation 1 – ReLu Equation

Equation 1 shows the ReLu function, where the input will always be 0 in the case of a negative number and the output will be in the input in the case a positive number is inputted. The output of this activation function is shown in Figure 2.

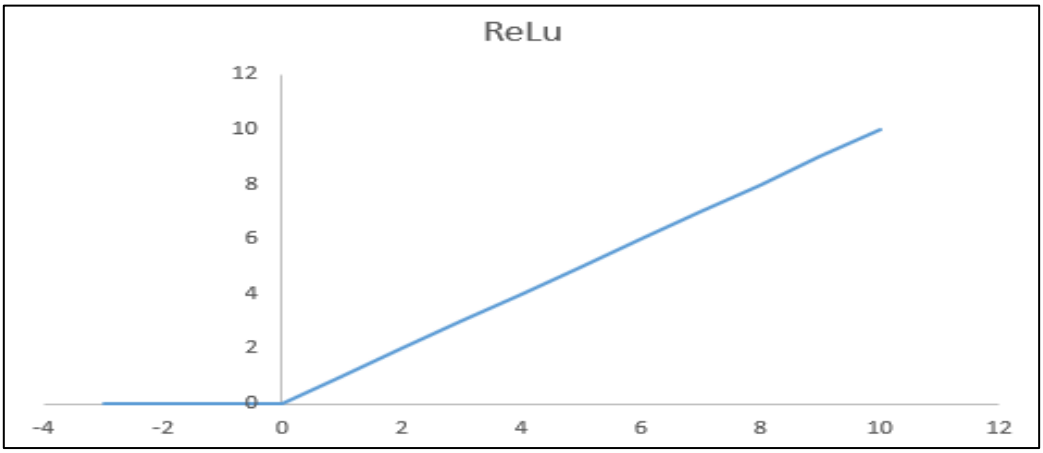

Figure 2 - ReLu Activation Function

Shown in equation 2, the softmax activation function is generally used in the final layer of a neural network for classification.

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Equation 2 – Softmax Equation

Neural networks have seen widespread use in activity recognition [24], [38], [92]–[94], they have been used as multi-layer perceptron's on data collected from smartphones [92]. They have been used to capture both locomotive movements [93], and mixtures of both activities of daily living and locomotive activities [24]. Neural networks have also shown that they can achieve high classification performance with activity recognition [94]. In certain circumstances, they have been shown to outperform decision trees [92], logistic regression [92], SVM [38] and Random Forest [38].

2.8.3.3. Deep learning

Deep learning is a subset of neural network-based learning, where instead of none or a single hidden layer, multiple hidden layers exist between the input and output layers. An interesting characteristic of deep learning is that it can be capable of generating its own features [95]. This reduces the requirement for a domain expert to select the most appropriate features to best describe the sensor data.

While this is still a comparatively new concept within the world of machine learning, it has shown promise in several different fields, most notably in image processing [95]. However, it has been shown to be effective for both classification of activities and generation of features for activity recognition [95]–[99].

Deep learning introduces new types of hidden layers, these include long-short term memory layers (LSTM) and convolutional layers. Instead of being able to only process a single feature vector, an LSTM layer allows the network to get feedback so it can learn from a sequence of data. This could be potentially interesting from an activity recognition standpoint because an individual performing an activity is likely to happen over a sequence of time rather than just in a single instance. Unlike LSTM layers, convolutional layers are able to generate features from images [100] and signals [95]. Deep-belief networks are another configuration of deep learning neural network which have been successfully shown to classify activities when using non-convolutional features [101].

Transfer learning is another domain within deep learning which attempts to exploit the knowledge gained from one domain to learn about a different domain. This is normally performed by replacing the output layer of a neural network with different classes. A solution such as this could be useful in tackling the problem of different smart devices having different accelerometers and other sensors [46]. While the sensors should produce similar profiles, they make not be close enough for a traditional machine learning classifier to discriminate.

2.8.3.4. K-Nearest Neighbours (kNN)

KNN is a lazy classifier which uses distance-based measures such as Euclidean distance to find the nearest labelled examples to the unknown data point. The “K” denotes the number of nearest data points to use, once the nearest labelled points are found a plurality vote is held and the unknown data point is assigned the highest voted label.

The kNN algorithm has been used for activity recognition [15], [23], [28], [76], [102], [103] and it has been used with wearable sensors [76], and in that context was shown to outperform an MLP by a significant margin [58].

The algorithm is useful as it can handle noisy data better than other classifiers and its performance is generally still good even when presented with a large dataset. However, an obvious disadvantage is that the K value needs to be carefully selected and the computational cost can be high [104].

2.8.4. Unsupervised

Unsupervised learning is the opposite of supervised learning and it infers that every feature vector has no associated label. Even though no labels exist, which would make it unsuitable for the recognition of activities it can be used to find patterns within the data without any supervision.

The two main methodologies which exist are clustering, which attempts to group similar data based on metrics, and component analysis.

2.8.4.1. Zero-shot learning

As the main aims of this thesis are to allow for data collection for activity recognition outside of laboratory environments, it is not feasible to capture labels to the same comprehensive levels that can be performed within carefully controlled experiments. As it is not always going to be feasible to label every activity [105], any chance to recognize activities, or add additional context to the small number of labels which have been collected is advantageous.

Zero-shot learning is a type of learning methodology where no labels exist, however, through other context it may be able to learn a new label without any specifically being provided. There are very few activity recognition systems which are capable of detecting activities which have been provided with no training data [106]. One potential example is the transitions between activities. It is very unlikely that a system will ever be provided with accurate labels for these transitions. However, knowledge of classified activities, and the unclassified moments of activity between classified activities, could be leveraged to identify activity transitions.

Zero-shot learning has been previously used for activity recognition. It was used in combination with a semantic attribute sequence model to detect activities with approximately 70% precision and recall [106]. Matching networks have also been tested for zero-shot learning based activity recognition [105].

2.8.5. Weakly Supervised vs Semi-supervised

A distinction is required when referring to weakly and semi-supervised systems. Weak supervision infers that weak labels have been used, which are labels which are noisy and incomplete [107]. Semi-supervised on the other hand uses correct labels, however, they will have incomplete coverage of the entire dataset and many unlabelled feature vectors may exist [77].

Weak labels can be advantageous as they increase the feasibility of data collection experiments as you no longer need to worry about the tedious problem of collecting every single label in an accurate fashion[107].

Multiple issues can occur when using weak labels, the main being that the noise generated from weak labels can become compounded, resulting in diminishing classifier performance with increasing levels of labelling noise. This can cause significantly worse performance than a normal fully supervised classifier [108].

2.8.5.1. Classification methods

The last chapters of this literature review have discussed supervised classification methods which require a fully labelled training set for classification. The next sections will discuss semi-supervised classification, which, unlike supervised, does not require a fully labelled dataset. Instead, semi-supervised uses some labelled data, but also examines the unlabelled data for any available insight.

As the main aim of this thesis is to recognize activities without the requirement for comprehensively labelled datasets, these semi-supervised techniques could prove useful to meeting these aims.

2.8.5.1.1. Multiple instance learning

Multiple instance learning is a semi-supervised model which differs from standard supervised learning models in that the data is placed into collections known as bags, the name attributed to the fact that each label can have multiple instances (set of feature vectors) assigned to it [109]. Initially, multiple instance learning was developed to solve the problem of the musk drug prediction problem [109] and has shown efficacy in other applications including, activity recognition [2], spam filtering [110], image categorization [111], text classification [112], visual tracking [113] and computer-aided diagnosis [114]. Like other semi-supervised methodologies, multiple instance learning has been shown to better deal with label ambiguity.

As mentioned, MIL works by placing feature vectors in sets, known as bags, then labels are gathered for the bag of feature vectors. However, these bag-level labels will not be directly transferable to the underlying feature vectors inside. This is because it is impossible to know exactly which feature vectors each label applies to. To resolve this issue, the two rules of multiple instance learning are:

1. *A bag will be labelled negative if all instance within it are negative*

2. *A bag will be labelled positive if at least one instance within it is positive* [115].

Multiple instance learning has been applied to multiple classification methods, including support vector machines (mi-SVM) [116], neural networks [117], diverse density [118] and random forests [113]. It is not without issues though as MIL approaches have been shown to have significantly longer convergence times than standard supervised learning approaches. While a key issue with MIL approaches is not knowing which of the instances in the bag is the true positive instance [116], it does provide a method of gathering labels which can be significantly less labour intensive as it does not

require as stringent notation of activity start and end times. Using this principle of limited labelling, MIL has been applied to activity recognition by gathering all of the activities within the bags, then using MIL techniques to analyse and discover the underlaying feature vector labels within the bags [2].

Learning Axis-Parallel

Learning Axis-Parallel was the first approach aimed at solving the problems associated with multiple instance learning [109]. It works by trying to fit an axis parallel hyper rectangle to the feature space in a way that the rectangle would contain no instances from negative bags and at least one instance from a positive bag. Multiple methodologies have been proposed for this concept, the most effective being an algorithm which begins at a source position and grows a rectangle until it meets the criteria of no negative bag instances and at least one positive bag instance [109].

Diverse Density

Diverse density is a theory which measures of how far each of the instances of negative bags are and how close the instances of positive bags are to a specific point in the feature space [118]. This concept, therefore, begins by trying to find the position in the feature space which is considered the most diversely dense. This approach has been extensively used in literature, on problems such as the MUSK drug prediction [118].

Multiple instance SVM

There are several approaches to implementing a multiple instance SVM which have been detailed in the literature. Two methods have been suggested, one of which classifies at a bag level (MI-SVM) and one which classifies at the instance level (mi-SVM) [116]. MI-SVM has been successfully used for classifying activities of daily living [2], providing near supervised results with less labels.

2.8.5.1.2. Citation kNN

Citation kNN is the multiple instance variation of the supervised kNN algorithm. It takes a majority voting approach to multiple instance learning and currently has unknown performance for the problem of activity recognition, but has competitive results in handling the Musk drug prediction problems [119].

2.8.5.1.3. Label propagation

Label propagation is a semi-supervised methodology in which unlabelled and labelled data are presented together and knowledge is gained from the distribution of labelled instances to unlabelled instances. An example is a k-nearest neighbour's classifier trained on a group of labelled data points; the unlabelled data points are then labelled via the classifier. The system will then converge once all

instances are labelled. Label propagation has shown promise in handling noisy labels, one of the potential drawbacks of human-annotated activity labels [2].

2.8.5.1.4. Active learning

Active learning has shown promising results for activity recognition [6]. Where semi-supervised learning methodologies learn from already gathered labels, active learning selects the instances that it deems to provide the most knowledge. The labels are gathered from a human annotator known as an 'Oracle' [120].

Instances are ranked by their informativeness, this informativeness is gathered using experience sampling methodologies. Examples include uncertainty sampling, which ranks the instances based on how unsure the classifier is about them. E.g. points which are the furthest from the hyperplane [2].

Active learning can be performed via several methodologies, for example, pool-based which initially has a large set of unlabelled instances and a very small set of labelled instances [120]. Active learning cycle of pool-based active learning where the classifier begins with a few labels of the training set L, cautiously selects another few instances from the pool U and sends them to the oracle. The classifier then chooses the next unlabelled instances to learn from based on the newest data from the oracle and the cycle repeats.

Noisy oracles and inefficient query selection strategies are the main concerns with active learning approaches [2]. A solution has been proposed for the reduction of errors caused by oracles, this involved gathering multiple labels for the same instance, this brings an issue of trading off accuracy with how overbearing the application will be on the oracle [121].

2.8.5.2. Semi-Supervised Summary

As this work intends to operate with reduced labels relative to normal full ground truth classification problems, the semi-supervised classifiers mentioned within this chapter will be crucial. They can effectively use the combination of labelled data points with unlabelled data points in order to increase classification accuracy without the additional cost of gathering labels. However, questions remain around how effective the system will be and the level of label reduction that can be applied before classification performance is reduced beyond a point at which it could be considered useful.

Aspects of the MIL techniques, where bags are created, could be leveraged to create a personalized data collection methodology. Users could provide a single label for many feature vectors and then use MIL methods to estimate what activity the feature vectors within the bag apply to. In effect, the weakly supervised problem could be transformed into a supervised problem in which a number of supervised classification techniques could potentially be utilized to recognize activities.

2.9. Ensemble Learners

Ensemble theory is the idea that a supervised model's predictive performance can be increased by using multiple models. This idea can be used to reduce the likelihood that overfitting, for example, Decision Tree classifiers tend to fit the training data exactly, which can result in overfitting if there is variance between the training and test sets. However, the Random Forest classifier, which contains multiple Decision Trees can reduce this bias by providing each of the Trees with a smaller sub-set of the original dataset.

There are multiple methods for ensemble learning including:

- Bootstrap aggregating – This methodology is also known as bagging; it is used by Random Forest and as mentioned previously within this chapter, it provides a random subset of the training set to each of the internal classifiers to increase the variance between the internal classifiers.
- Boosting – This methodology uses an incremental approach to building the ensemble. It trains the first model then iteratively creates more internal models but increases the weighting associated with previous incorrect classifications. This method reduces the bias and has a chance of overfitting.
- Stacking – This uses layers of internal classifiers and the previous layer of classifiers is used to train the next layer of classifiers. This is used to increase the performance of the predictive model.

Examples of boosting and bagging algorithms are:

2.9.1.1. Random Forest

Random Forest is an ensemble learner, meaning it uses a combination of multiple classifiers to improve the classification performance. In the case of Random Forest, it will build a collection of Decision Trees, each of these trees will be trained on a random subset of the training data and the mode of each of the predictions of the trees will be the assigned label for the unknown data point. This ensemble method is intended to overcome the limitations of standard Decision Trees such as overfitting.

In order to classify the data, the Random Forest will first split it into random subsets. A user-defined number of Decision Trees are created, and each is provided with one of these subsets for training. Once then presented with unseen data, each tree will make its prediction on what class it should belong to. The final class is then generated by averaging all of these predictions, usually with the mode result becoming the Random Forests final prediction.

There are multiple parameters which can influence the final prediction, the first is the number of Decision Trees used for prediction. Generally, increasing the number of trees can improve the performance at a cost of compute performance [122]. The other major parameter is the depth of the trees, were having a deeper tree can potentially improve the trees representation of the data but again at the cost of compute.

In addition to classification, Random forest can also be used to aid in feature selection, it does this by measuring the importance of each feature for each of the Decision Trees. This importance can then be used to reduce the dimensionality of the data and potentially increase the performance of the classifier [27].

While it has not been adopted for activity recognition to the extent that SVM has, it has still seen widespread use [27], [123]–[126]. It has shown to discriminate locomotive activities very well [27], and also provided high levels of accuracy when presented with impersonally trained data, exceeding the results of neural networks [126].

For detecting activities, the Random Forest has shown efficacy in detecting household activities, including doing laundry, dusting, cleaning dishes, sweeping, as well as certain locomotive activities, including walking and jogging [124]. It also was capable of classifying stairs, but not to the same levels of performance as the other activities within this study [124].

As previously described, personalized data is when the classifier is trained on the same person in which it is tested on, while impersonal is when the training and test data are collected from independent people. Weiss et al conducted experiments to evaluate the effect of personalised models. Random Forest was shown to achieve the highest levels of performance on average across both impersonal and personalised datasets when compared to SVM, Decision Tree, Neural Network and Logistic Regression [126]. Only a Neural Network was shown to outperform Random Forest on personalised data. In the experiments, locomotive activities had been tested including walking, standing sitting, laying, using stairs and jogging.

2.9.1.1. XGBoost

XGBoost is another ensemble learning algorithm which again uses a collection of Decision Trees. However, unlike Random Forest which uses full-grown decision trees (which can be prone to overfitting [127]), XGBoost uses weak learners. A weak learner is any classifier which can provide labels at a rate which only marginally outperforms a random guess [128].

The XGBoost algorithm contains multiple parameters which need to be tuned correctly to provide the maximum performance. The n_estimators and max_depth parameters are shared with the Random Forest classifier in that it controls the number of trees used and the depth of the trees respectively.

XGBoost is still a relatively new algorithm but has shown promise in the context of activity recognition [127], [129]–[133]. It has outperformed k-NN, Random Forest and SVM [130].

2.10. Model training method

Several different methodologies exist for constructing a training set. They can be generically trained, where the participants of the test set are not present within the training set. They can be personally trained, where the test and training set contains the same participant, or they can be a hybrid of these two approaches.

Each of these methods of creating a training set has its own advantages and disadvantages which will be discussed in the proceeding sections. Finally, a section will compare the results of testing these training methods on participants performing activities to see which of the three gives the best results.

2.10.1. Generic

One of the main challenges associated with activity detection is that each individual is going to have their own idiosyncrasies [46], potentially resulting in different sensor readings. A model trained on one person may not translate well to another person. Impersonal models are those where data is captured by a collection of people and applied to an unseen user without any further labelled data. In contrast to this, personal models are training data captured from the individual who the model will be tested on [134]. This implies that all personal models require an initial training phase, potentially having a negative effect on user adherence but personalized models have been shown to be significantly more accurate [134]. The final type of personalized model is a hybrid model, which uses a mixture of both approaches. Although if it is not possible to have a personalized collection of data, it is better to have learned from more participants than to get more data from each participant [134]. It has been shown that personalized models perform substantially better than impersonal models [126].

A generic model is any model which has been trained on the individuals who the system won't be applied to. An example of this would be a dataset containing ten participants. The system would be trained on the first five participants and tested on the other 5. This is done in many activity recognition experiments to reduce bias.

Generally, training a classifier this way with result in lower performance than if the classifier is trained on the person that it was tested on. This is likely due to subtle differences in sensor placement, movement style.

2.10.2. Personalized

A personalized model is one in which the classifier has been trained on the people it will eventually be tested. When trained this way the classifier will generally have higher recognition performance because it has already seen the exact way in which the person will place the sensor and how they will move.

However, training classifiers this way is difficult outside of a data collection experiment. If a system is to be deployed in real world settings, it would be unfeasible to perform detailed data collection and labelling for all users of that system. A more feasible approach would be enable a person to provide their own data and labels, which a classifier can then learn from.

2.10.3. Hybrid

A hybrid model is a combination of both a generic model and a personalized model. The system could begin with a model which has been generically trained on some individuals, giving the classifier some idea of the activities to be captured. Then when the system has been deployed to a user, they can provide additional information to the system to help increase the classification performance further.

These hybrid models have been shown to provide the best results as they avoid the pitfalls associated with other model types. It can capture more accurate information than generic models and it is able to capture activates without significant user intervention.

2.10.4. Performance comparison of the training method

Previous work by Lockhart et al conducted experiments to evaluate the effect the three different training methods has on performance [108]. Experimentation trained models on walking, jogging, using stairs, sitting, standing and laying activities, with the three different model training methods. These experiments were carried out on an Android smartphone, which meets the criteria of naturalistic activity recognition that this work is aiming towards. The accelerometer data from the phone is collected at 20Hz. Six features consisting of average acceleration, standard deviation, average absolute difference, average resultant difference, binned distribution and frequency are gathered from each access resulting in 43 total features. These features had been captured across a 10 second sliding window [134].

When then tested on a neural network, personal models achieved an accuracy of 98.7%, the hybrid was slightly behind with 92.1% and impersonal models lagged significantly with 67.8% [134].

Personalized activity recognition systems can show high levels of performance; however, performance will only remain good if data recording conditions remain consistent. For example, if the orientation of the sensor is changed, the individual whose activities are being classified or the position on the individual's body is changed, then performance will reduce significantly [46].

It is likely that of the three the hybrid model will be used as it provides the value in terms of user intrusion versus classification performance.

2.11. Computation and battery-powered devices

Considering that all smart-devices are battery-powered, care needs to be taken when considering which pre-processing and classification techniques to use [15]. Larger deep learning networks have been shown to have a significant impact on the battery life of a smartwatch, however, networks with less hidden layers and fewer parameters reduce this impact [135].

The computational complexity of training a model on a battery-powered device by capturing the acceleration values on the device and transmitting them to an online server for analysis has proved to reduce the computational complexity. Although there is still some computation required when actually capturing and transmitting the data, training a classifier on the server both removes a major portion of the overhead and also gives some freedom in the classifier to be used [136].

In an experiment which was classifying nine activities using an MLP. Power was saved by reducing the frequency in which samples were being taken from the accelerometer. A balance was then formed on how little data could be processed while maintaining a comparable classification performance to other works [94]. A different experiment compared an SVM with Dynamic Time Warping, versus no activity recognition system and found that after 140 minutes, the SVM had used approximately 5% less battery than dynamic time warping. However, the SVM had still used a significant amount more battery than no AR system running on the device.

The chosen features can also have a significant impact on the energy consumption of activity recognition systems. One study compared the classification accuracy and energy consumption of different types of features. It found that time-domain features use significantly less energy than frequency domain features, and the sampling rate also had an impact, with lower sampling rates consuming less energy. However, adding frequency domain features did not always improve classification accuracy, meaning the types of features extracted should be carefully considered [137].

2.12. Activity transitions

Activity transitions are the activities which occur between or when transitioning to a new activity. For example, transitioning from standing to sitting will require some movement. Detection of these

moments can be useful as it provides a way to segregate the moments of activities starting and ending and can provide additional context for labels close to the activity. For instance, being able to classify the transition 'standing to sitting' provides context about the data before and after the transition. Specifically, data before the transition must relate to standing and data after the transition must relate to sitting.

When considering the context of certain transitions between activities, it is worth remembering that certain transitions are impossible. They are impossible because they will usually require another activity to be performed before the person transitions. For example, it should not be possible for a person to transition from laying to walking as walking first requires a transition from laying to standing [15]. Understanding these different temporal constraints for certain transitions could potentially reduce the compute complexity of an activity recognition system, and where an impossible transition occurred it may highlight problems with the classifier.

2.13. Summary

The literature review began by first explaining the problems existing within the area of activity recognition. The uses of activity recognition and the different types of activities to be detected are discussed. These included locomotive activities, activities of daily living and sports activities. Details are given on the normal methods of collecting these activities, through laboratory-based experiments through to unobtrusive methods of capturing these activities.

Experience sampling is identified as a potentially useful method in the capture of activity labels, multiple methods of experience sampling exist which have trade-offs between how intrusive they are and how useful the data they provide is likely to be. Sensors and how the data can be used is also discussed, with multiple methods of filtering, windowing and feature extraction detailed. Once the data has been processed it can be applied to classifiers and the most common ones are detailed within this literature review.

As we can never be sure how much useful data a weakly supervised activity recognition system will receive, the different types of models are discussed. These include generic models which only use data captured from other people, personalized models which only use the data and labels captured from the person who the system will be used on and then hybrid which uses a combination of the two. Of these, hybrid models are the ones most likely to be used for future experimentation as they can both get the performance increases associated with personalized models whilst also potentially reducing intrusion via generic models.

From the chapters in this literature review, it is clear than significant problems exist within the area of activity recognition using sensor arrangements. While it has been shown that a collection of large amounts of unlabelled data is possible using smart devices, the collection of labels for this data is a non-trivial issue. Performing the data collection experiments seen in literature, which are intended to capture both the raw signal data and accurately capture the labels associated with that data, are infeasible in daily life. Some previous work approached the use of weak supervision for the capture of activity labels, however, gaps in knowledge still exist. For instance, the testing of hybrid models which use the generic data from a sample of the population and then any labels which the user can provide.

3. Experience Sampling for Label Collection

The focus of this chapter is to investigate the problem of labelled data and the different methods in which labels can be collected and utilized. The chapter will address the following questions relating to labelled activity data:

1. Can activities be recognized with weak labels?
2. If they can, what activities can be recognized?
3. What effect does the weakness of the labels have on the classification performance?

This chapter will begin by detailing the datasets to be used in the experiments. It will then simulate labels collected from the user; these labels can be collected using a diary study technique known as experience sampling. Once the experience sampling labels have been gathered, weak supervision techniques will be applied to propagate the labels to feature vectors and then, finally, the labelled feature vectors will be used for classification.

3.1. Methodology

Hypothesis: An accurate activity recognition model can be trained on weaker than normal labels.

To test the above hypothesis, suitable datasets must first be obtained. Ideally, a combination of different types of activities, such as ADL and locomotive activities, should be included in the dataset.

3.1.1. Datasets

Two datasets have been chosen to test the hypothesis that weak supervision can provide a solution to the labelling problem within activity recognition. The first of which is the HAPT dataset, which provides locomotive signals collected from a smart device. The second is the HMP dataset which provides ADL signals from an accelerometer.

These datasets have been selected as they are both contain at least an accelerometer, a sensor which is commonly found on even lower specification smartphones. They also provide two distinctly different types of activities, which will allow robust evaluation of techniques under different conditions.

To fully test our hypothesis, different levels of labelling restriction will be applied to the datasets, beginning with a higher number of labels and eventually restricting it to an extremely low number of labels relative to the number of feature vectors. It is postulated that extreme restrictions in labels will show the limits of the datasets and may result in entire activity labels being missed and therefore not modelled by any classifier.

3.1.1.1. HAPT Dataset

The Human Activities and Postural Transitions (HAPT) dataset [55] uses a waist-mounted smartphone to collect 6 activities. The signals from the gyroscope and the accelerometer are sampled at 50Hz. In total, 30 participants are included in the dataset with the age range being between 19-48 years. The dataset contains a mixture of static and dynamic activities as well as the postural transitions which occurred on the static activities. For the purposes of this experiment, the postural transitions are being ignored for now but will be re-visited in later chapters of this thesis. The static activities performed are standing, sitting and lying and the dynamic activities are walking, walking upstairs/downstairs.

This dataset is particularly useful as if we refer to the original problem, discussed in Chapter X.X, the intended use of this type of activity recognition system is to accept the labels provided by a user for a personalized activity recognition experience within naturalistic settings. To be performed within these naturalistic settings, it would require the use of a device which is able to perform statistical analysis of data, allow user input and provide ambient sensing technology. Smartphones meet these criteria and are carried by a significant portion of the population, and as this dataset is captured on smartphones then it could provide a close representation of the possible performance of this type of activity recognition system.

The HAPT dataset also provides a balanced number of classes for each of the participants, which helps ensure that each of the classes gets fairly represented in both the training and test data.

3.1.1.1.1. Feature extraction

For the experiments within this section, the features have been pre-computed. They are generated from raw 3-axis accelerometer and gyroscope signals. The derived features are a mixture of time and frequency domain features. Prior to extraction of features the signals are de-noised using both a median filter and then a low-pass Butterworth filter with a cut-off frequency of 20Hz. Five different time series are produced from these signals. Acceleration values produced from “body” movements and the acceleration values produced from gravity. The signals are split using a low pass Butterworth filter with a cut-off frequency of 0.3Hz. The jerk of the body signals is then produced by calculating the rate of change of the acceleration values with respect to time. The time series now produced is the acceleration due to gravity, the body acceleration, the body gyroscope acceleration and jerk of the body signals. The magnitude of each of these is calculated using norms. Fast Fourier transforms are calculated from these signals. The signals are split into windows which are 128 samples in length, approximately 2.6 seconds of data. Several variables are calculated from each of these signals, including mean, standard deviation, max/min values, median absolute deviation, signal magnitude area, energy of signal, interquartile range, signal entropy, correlation coefficients, the autoregression coefficient, angle between the vectors, skewness, kurtosis, mean frequency, frequency component

which has the largest magnitude and energy of the frequency interval of the Fourier transform window. The mean of the angle between the vectors is also calculated. This results in a 561-length feature vector.

3.1.1.2. **HMP Dataset**

The HMP Dataset [138], [139] uses a single wrist-worn tri-axial accelerometer to capture acceleration values from participants at 32Hz. The dataset captured the following activities, brushing teeth, climbing / descending stairs, combing hair, drinking from a glass, eating meat, eating soup, getting up from bed, lying down in a bed, pouring water, sitting into a chair, standing up from a chair, using the telephone and walking.

To effectively test the hypothesis that weak supervision could provide a viable method of data capture for activity recognition for ADL, several activities within the dataset will be modified or removed. Drinking from a glass, eating meat and eating soup will be combined into one activity known as eating and drinking. Lying down and getting up from a bed will be combined into use bed and the same will be applied to getting up from and sitting into a chair which will become use chair. The locomotive activities are also being removed. These changes are being made as ADL classification is already a significant challenge with a fully labelled dataset, with the restrictions placed on labels by the hypothesis for weak supervision it is going to be an unrealistic task. Therefore, combining similar activities and those which will be tested by the previous dataset provides an opportunity for the hypothesis to be proven, and in the case that the hypothesis is proven correct, increasing the challenge can come as a future improvement. With these changes, it will leave a total of 7 activities. They are Brush Teeth, Use Bed, Eating & Drinking, Use Chair, Use Telephone, Pour Water, Comb Hair.

The dataset contains data from 11 males and 5 females, with a range of ages between 19 and 81. The participants also ranged in weight from 56kg to 85kg. The activities within this dataset had been collected in trials, with a participant repeating the same activity until the end of the trial period. Not all the participants performed all activities and they did not perform an equal number of trials for each activity. For example, Brush Teeth has ten trials performed by one female participant and two trials by one male participant, rather than a smaller number of trials performed by a wider range of participants. The total number of trials for each activity is shown in Table 1 and it is immediately visible that some of the activities are severely underrepresented, such as Brush Teeth, Use Telephone and Comb Hair. As the hypothesis for this chapter will restrict the number of labels, this imbalance could cause problems for the classes who aren't represented as well.

As this dataset has been captured on a wrist-worn device, it represents a different perspective than the usual hip-attached activity recognition systems. Like the smartphones mentioned in section 3.1.1.1, wrist-worn activity recognition systems such as fitness bands and smartwatches are becoming increasingly more common and we postulate that this method of data capture may be more useful for the capture of ADL. This is because many of them contain gestures which are performed with the arm or hand, e.g. brushing teeth or combing hair, which would not easily be recognized by a hip-attached device. With careful consideration of these points, we feel that the HMP dataset provides a good medium for testing our weakly supervised methodologies.

Table 1 - HMP Dataset Activity Details

Activity	Number of trials	Total Time Spent Performing Activity
Brush Teeth	12	15.53 minutes
Use Bed	129	44.24 minutes
Eating and Drinking	108	58.55 minutes
Use Chair	202	36.55 minutes
Use telephone	13	11.03 minutes
Pour water	100	32.6 minutes
Comb Hair	31	17.03 minutes

3.1.1.2.1. Feature Extraction

Before extracting features from the HMP dataset, the data is first smoothed using a Median filter and Butterworth filters are applied. The first Butterworth filter is a low pass filter which has a cut-off frequency of 20Hz which is intended to de-noise the signal further, followed by another Butterworth

set at 0.3Hz. From this signal 6 features are extracted from each of the three-axis. These features are mean, standard deviation, median absolute deviation, max, min and interquartile range. This results in a 46-width feature vector. The smaller feature vector, when compared to the HAPT dataset is related to the fact that the HMP dataset only has a tri-axial accelerometer and no gyroscope. It is likely that overall classification and weak labelling prediction performance will be impacted as the gyroscope can be useful in discriminating activities that contain a lot of rotation.

3.1.2. Experience sampling

The experience sampling method is a diary study technique used to gather data from a user. This concept can be related to the field of activity recognition by collecting data about the activities that the user is currently performing. Collecting data in this way can reduce the requirement for formal, unnatural data collection experiments and allow the user to capture the activities they want to track. However, a major drawback to this method is that we only get back what we ask of the user. Making as many requests as possible will be beneficial to the system as it provides additional data to aid in classification. However, excessive requests for input will be considered intrusive and will likely reduce the chances that the user will accept the system. It is critical that the questions are designed to capture effective data from the user.

Previous work investigating experience sampling for activity recognition used several different types of sampling questions posed to the user. These ranged from the user providing a single label for the data to multiple labels. Some examples are:

- “What are you doing now”

This request is intended to catch a single label for the current activity that the user is performing

- “What did you do since last time”

This request will capture all the activities the user has performed since the last time they had been prompted.

- “What did you do most of the time since the last request”

This is intended to capture the activity the user has been performing for most of the time since the last request. Each of these requests can provide useful information, however, through our own experimentation into the most useful experience sampling questions we found that “What are you doing now” and “What did you do most of the time since the last request” to be particularly useful. “What did you do since last time” can provide multiple labels, however, it provides a higher chance of noisy labels due to the additional difficulty in populating multiple weak labels than a single weak label.

Due to their effects on classifiers decision hyperplanes and the additional burden associated with users providing multiple pieces of information, noisy labels are much worse than no label at all.

In order to test the hypothesis, several different approaches of processing the labels gathered from experience sampling will be evaluated. A concept derived from multiple-instance learning, known as bagging, will be utilized such that sets of feature vectors will be placed into "bags". Each bag will have one single label associated with them, known as the bag-level label. However, the difficulty in this approach comes from the likelihood that not all the feature vectors inside the bag will correspond to the same activity class as the bag-level label. For instance, if requests were made every ten minutes, then each bag would be assigned a single label and would contain 10-minutes of feature vectors. A key problem to consider here is how would we handle a bag with a bag level label of standing if, for example, it contained 1-minute of feature vectors which corresponded to walking? And with this, how do we map the bag-level label to these internal feature vectors without creating noisy labels?

In the proceeding chapters, we will evaluate some of the methods of approaching this problem, ranging from simpler methods, such as giving all feature vectors within the bag the same label to methods which are engineered from the context of the experience sampling request. For example, if we ask for the majority label within the bag, then we know that the largest group of similar feature vectors is likely the correct ones for the provided label. To aide with visualization of the different approaches in these proceeding chapters, Figure 3 has been provided.

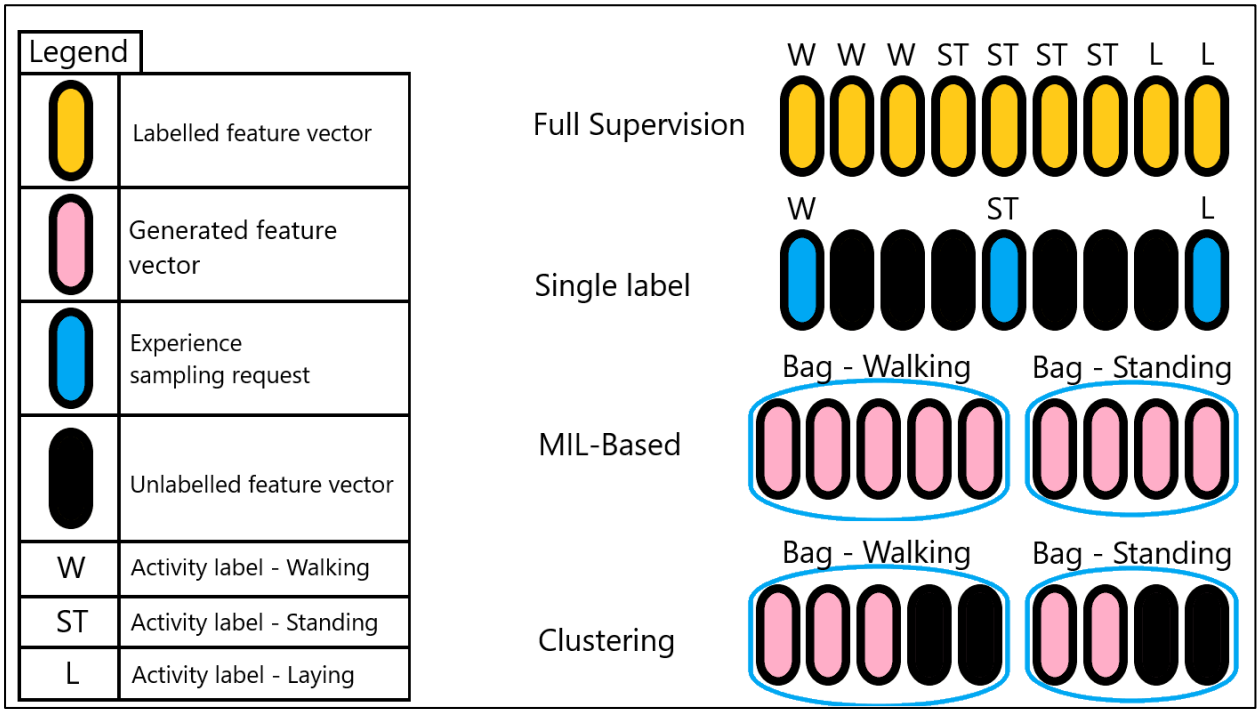

Figure 3 - Visualisation of weak labelling approaches

3.1.2.1. Single Label

This approach will evaluate using two different label collection methodologies, the first being instance-based label collection, which finds the label for a feature vector which lies directly on the experience sampling intervals. E.g. for a one-minute experience sampling window, the feature vector which lies on exactly one minute will be used. A potential issue is that when a user is about to input the data, they will move the device, and the movement of using the device will be incorrectly labelled as the locomotive activity they are performing. An example of this collection process can be seen under the "Single label" section of Figure 3, where the blue objects represent an experience sampling request, the other black objects represent missed feature vectors which will remain unlabelled. While this will not present an issue in our experiments, due to the use of simulated user requests, if it were to be used on an online system it would require that either data collection is halted until the request is completed or to apply the label to feature vectors that occurred at a time prior to the time the user completed the request.

3.1.2.2. Multiple-Instance Learning-based label collection

The second approach is to use multiple-instance learning concepts and collect bag-level labels. Where traditional fully supervised models learn from individual feature vectors, multiple instance learning based models place data into collections. These collections are known as "bags", where a single label can have multiple sets of feature vectors associated with it [109]. These bags are then processed by a modified supervised classifier, an example is a bag level classifier known as MI-SVM [116]. Multiple Instance Learning operates on the principles that; a bag will be labelled negative if all instances within it are negative and a bag will be labelled positive if at least one instance within it is positive [115]. A key problem in the case of this work is that identifying truly "negative" bags is a difficult task. Since we are only capturing a single label per bag, it is likely that a bag will contain activities other than the activity identified as the majority activity. It is therefore extremely difficult to accurately define any bag as negative due to the unknown activities which may occur in the bag. This combined with an aim to predict individual feature vectors rather than bag labels means this work will only use multiple-instance classifiers for comparison. While this may be a more feasible approach than collecting a label for a single feature vector-based, the problem of modelling the relationship between feature vectors and labels within a bag is very difficult.

When we simulate the capture of a label from experience sampling, we will have a single label which represents the entire bag, and an example of this can be seen in Figure (one on the previous page). However, there is no guarantee that the bag level label is representative of all the activities that occur

within the bag. Therefore, a method which tries to automatically identify feature vectors relevant to the bag label is required. For the purpose of conducting repeatable evaluation-based experiments, experience sampling will be simulated by contacting the original ground truth at certain intervals. Ideally, to minimize the user burden associated with data collection, longer bags would be an advantage, but they could reduce the amount of data available to the classifier. With all likelihood, the activities will either be longer or shorter than the specified time interval, requiring some method to decipher which feature vectors within the bag correctly apply to the activity label. In a real-world scenario, the user request would ask for which activity was performed the most during the time interval. In order to simulate this the label which occurs the most within the bag will be captured from the ground truth. If in any cases there is no clear majority label within the bag, the bag will be skipped.

3.1.2.3. Experience sampling request intervals

An important step of the experience sampling process is calculating how often to poll the user. A shorter duration produces smaller and more frequent bags and bag labels. This has the effect of capturing more labels overall, which should theoretically reduce the chances of having noisy labels. Conversely, longer durations produce larger and less frequent bags and bag labels. This may result in more noisy labels, but from a user intrusion perspective it is advantageous as the user does not need to provide as much data.

As our experimentation will use pre-recorded datasets, the aforementioned user request processes can only be simulated. Initially, our training set will contain sensor data but no labels. It will then simulate the experience sampling requests over varying lengths. The advantage of this simulated approach is in the repeatability of the experiment. For each run the data will be consistent and therefore we should have reasonably consistent results when experimental parameters are changed, save for the randomness of some classifiers. If we were to use real users for collecting experience sampling labels and data, we would need to re-record the data for each person, for each time we changed the sampling window. Even if we took the time to record all this data at different sampling windows, with different participants, the results would still not be directly comparable as it is likely conditions and the participants performances would not be the exact same.

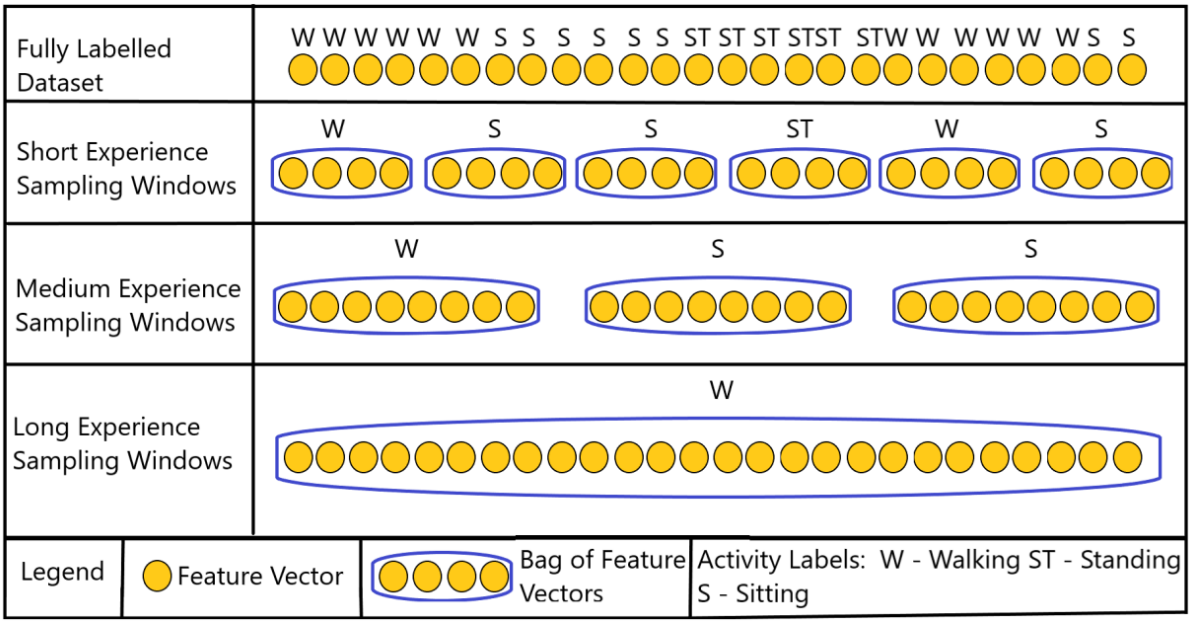

Figure 4 - Visualisation of bag lengths

3.1.2.3.1. Short experience sampling windows (1-5 minutes)

It can be expected that the short experience sampling windows will perform with similar activity recognition performance to fully supervised learning. At one-minute intervals, even with 95% of labels removed, the system will likely still have enough data to classify the 6 activities with reasonable accuracy.

3.1.2.3.2. Medium-long experience sampling windows (10-20 minutes)

We postulate that this could be the optimal window size as it provides an acceptable break between data collection requests for users, while potentially providing enough data for classification.

3.1.2.3.3. Long experience sampling windows (30+ minutes)

These windows will contain an extremely low number of labels relative to the number of feature vectors.

Considering the different effect, the sampling window will have on the overall system, we are going to evaluate both the collecting a label for a single feature vector and the multiple-instance learning approaches on several different time intervals between experience sampling requests. These are 0.5, 1, 1.5, 10 and 30 minutes.

3.1.3. Single Instance Learning

Single instance is learning is a 'naïve' method of translating bag-level labels to the feature vectors within the bag, we perform this by assigning the bag level label to each feature vector within the bag. As single-instance learning is a naïve learner in that it will incorrectly label a large number of feature

vectors, but has been shown to be effective in certain cases [140]. An example of single-instance learning being applied to data can be seen in Figure 3 under the title of “MIL-Based”, it shows that a single label is gathered for each bag and the labels are applied to all feature vectors within the bag.

Referring back to Figure 3 under “MIL-Based”, we can see the first bag has been given the label “walking”, however, if we refer back to the ground truth, denoted as “Full Supervision”, we can see that within our bag the first three feature vectors apply to walking and the final two apply to standing. Therefore, if we used Single Instance learning we would have inadvertently labelled the two standing feature vectors as walking, which would have knock on effects on final classification accuracy.

3.1.4. Clustering for Multiple-Instance based collection of labels

Since each user request only collects a single label which encompasses multiple feature vectors, we investigate methods to reduce the likelihood that a label will be applied to an incorrect feature vector. Based on the assumption that the collected label is the one performed the most over the specified time frame, the proposed methods will identify similar feature vectors within the bag. We propose a clustering-based algorithm to search for the largest single cluster within the bag.

Experience sampling will only have gathered bag level labels. Single instance learning assigns each feature vector within the bags the same label as the bag label. The proposed clustering-based algorithm aims to discover the largest cluster within the bag and only apply labels to these feature vectors. Figure 1 demonstrates that for a certain time interval, experience sampling will gather one label for a bag of multiple feature vectors but does not label any specific feature vectors. Single Instance Learning populates each of these bag labels to the entire bag, resulting in a high per bag recall rate, but potentially poor precision rate. However, the cluster analysis approach aims to overcome this problem by only assigning the appropriate feature vectors in the bag to the bag label. The aim is to maintain a similar recall rate to Single Instance Learning while improving the precision rate.

Since many clustering algorithms require the number of clusters to be specified, this presents a problem. The core aim of the proposed bag clustering is to cluster the bag into a set of K clusters, where each cluster represents a different activity. Each user request provides only a single label.

Due to the weakly supervised conditions, the number of unique activities within a bag is not known and therefore the number of clusters that bag is also not known. The number of clusters, and the largest cluster, must therefore be estimated. We propose the use of a clustering algorithm which does not need the number of clusters within the data to be specified.

An example of this approach can be seen in Figure 3 under the heading “Clustering”. Unlike the MIL-based approach which will naively place labels on feature vectors with no consideration, the clustering-based actively leaves feature vectors unlabelled if they are likely to be incorrect.

3.1.4.1. Density-based spectral clustering of applications with noise (DBSCAN)

DBCAN is a clustering algorithm which uses density-based information to find clusters without requiring knowledge of the number of clusters. It groups these points together by considering how many nearest neighbours they have.

It contains two main parameters, eps and minPoints. Eps is the maximum distance between two points for which they can be considered neighbours. MinPoints is the minimum number of points for a region to be considered dense.

3.1.4.2. Gaussian Means (G-Means)

K-Means clustering is a method of clustering unknown data points into several clusters. Prior to performing this clustering, the algorithm must be provided with the parameter K, this value K constitutes the number of clusters to be found.

As this system is capturing a single label and will have no knowledge of the underlying distribution, there is no way to know the K value prior to clustering. In order to estimate the K value, we can use an extension of K-Means known as Gaussian Means. The Gaussian Means algorithm attempts to find the value of K by incrementally increasing K until it appears that each cluster comes from a Gaussian distribution. This is paired with a parameter known as the strictness value, this value is how well the clusters must fit a Gaussian distribution before the K value is accepted. If the clusters do not fit a Gaussian distribution to the required strictness, then the K value will be increased.

3.1.4.3. Parameters

As both the G-Means and DBSCAN algorithms require parameters that need tuning, they will be discovered using a Grid Search.

Parameter selection is performed once for each algorithm. After tuning has been performed, the same parameters will be used by the clustering algorithm for all experiment conditions to ensure results are directly comparable. In order to initially tune the parameters, a balanced set of experiment conditions were needed. Thus, a one-minute experience sampling time interval was selected. The one-minute interval restricts the labels somewhat but not to an extreme so it should provide a good baseline for tuning the parameters.

3.1.4.4. Clustered label algorithms

Once either the G-Means or DBSCAN algorithms have successfully clustered the feature vectors within a bag, the largest of these clusters is found. In this case, the experience sampling request will indicate

Algorithm 1 - Weak Label Propagation via G-Means

1: {Input: b = feature vector of bag to be clustered, s = g-means strictness value}
2: b_GM = GMEANS(b, s)
3: c = findBiggestCluster(b_GM)
4: return c

which activity has been performed the most over the given time interval. Therefore, the experience sampling label can be assumed to be the largest activity cluster within the bag of feature vectors. The activity label is then assigned to the feature vectors which are found to fall within this largest cluster and any remaining feature vectors, assigned to other clusters, are left unlabelled.

3.1.5. Classification

The literature shows that Support Vector Machines perform well on activity recognition problems and they can be modified to allow them to be hardware-friendly to mobile devices [85]. As such, the modelling of weakly labelled feature vectors in this work will be computed using an SVM. Specifically, an error-correcting output codes support vector machine (ECOC SVM) will be used. An ECOC SVM allows an SVM to act as a multi-class classifier. This is advantageous for activity data as generally there are multiple activities to classify.

An ECOC models multiclass problems by providing an ensemble of binary one vs one classifiers [10]. The error-correcting codes output by the ECOC SVM means that although there are extra data overhead, these codes can help the system recover from errors such as poor input features or a flawed training algorithm, something which may be appropriate for weak supervision techniques since there is a high likelihood noisy labels will be included in the training data.

These experiments will also include the use of TreeBaggers, which is an implementation of a Random Forest which is found within the Matlab application. To overcome the problem of overfitting seen with Decision Trees, the TreeBagger utilizes an ensemble of Decision Trees via Bootstrap Aggregation.

The final classifier used within this chapter is Gradient Boosting. Gradient Boosting, similar to the TreeBagger, is another ensemble learner which uses a collection of Decision Trees. However, they differ in how they operate. Where TreeBagger uses bootstrap aggregation, Gradient boosting uses boosting. Where TreeBagger uses fully-grown Decision Trees to make decisions, Gradient Boosting uses very shallow Decision Trees. The changes mean that Trees grown from TreeBagger have much less variance, which depending on the application, can improve or degrade results.

3.2. Results

The following section will describe several experiments conducted to evaluate the performance of the different label collection methodologies. Firstly, experiments will be conducted to find the maximum performance of supervised classification as a baseline comparison. Experiments will then be conducted to evaluate different experience sampling methodologies, beginning with single feature vector labels and then testing the multiple-instance learning-based collection.

3.2.1. Evaluation Protocol

Throughout the following experiments on two datasets, performance will be evaluated through several metrics. The main metric used is accuracy, the equation for which is shown in Equation 3, it is the total number of correct predictions divided by the total number of predictions made.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Equation 3 – Accuracy

F1-Score will also be used as part of the evaluation protocol, which is calculated as shown in Equation 4. Generally, F1-Score is a better alternative to accuracy when the dataset is imbalanced.

$$F1 - Score = \frac{2TP}{2TP + FP + FN}$$

Equation 4 – F1-Score

As we will be dealing with labels generated through weak labelling algorithms, it is important to consider the precision of these labels. Poor precision scores will indicate a noisy training set which will cause significant problems during training. The equation for Precision is shown in Equation 5.

$$Precision = \frac{TP}{TP + FP}$$

Equation 5 – Precision

3.2.1.1. HAPT Dataset Validation

Validation for the HAPT dataset is performed using a train/test split and not cross validation. As there are thirty participants, 20 of them are used within the training set and the remaining 10 are used in the testing set. Although the original hypothesis stated that this could be used as a system where the user trains on their own labels data, we are unable to perform it this way due to the lack of data for each participant. However, our training and testing set will remain independent, with no data from any participant in the test set represented in the training set. This is likely going to increase the difficulty for the algorithms in this chapter, and, if they do perform well here, it is likely that the algorithm would perform better when tested under real circumstances with the users own data.

3.2.1.2. HMP Dataset Validation

Validation on the HMP dataset will be performed through 10-fold cross validation, the reason this has been selected over the train/train split in the previous dataset is it would be impossible to segregate the data based on individuals. The reason it is impossible to segregate is in the configuration of the dataset, not all activities are performed by all participants and some activities are only performed by one or two participants. If we did use a train/test split then it would be likely that the training set and the test set would not contain the same number of activities, which would cause problems when it came to classification.

The reason 10-folds have been selected over a lesser number of folds is to increase the size of the training data as much as possible. Once the experience sampling intervals get longer, the less the number of folds the smaller the training sets and therefore the less time experience sampling can operate before it runs out of data. Normally, this would be performed by using Leave-One-Out cross validation, however, due to the configuration of this dataset it is not possible. This is because not all the activities have been performed by all participants, and in some cases, the activities have only been performed by a single participant. If Leave-One-Out cross validation was used, then in most cases the training data would not contain all of the activities, making it an impossible task for the classifier when tested against the test set.

3.2.2. HAPT Dataset

Firstly, experiments will be conducted on the locomotive activities of the HAPT dataset. These activities are standing, sitting and lying, walking and walking up and downstairs. We theorize that the performance will be higher on these activities than the activities of daily living as they contain signals which are generally more repetitive and distinct.

3.2.2.1. Fully Supervised Performance

We will perform standard supervised learning by using the pre-computed feature vectors with all activity labels included. This will provide a baseline for the maximum possible performance which can then be compared with the performance of the experience sampling methods.

The results shown in Table 2 provide a baseline for weakly supervised performance to be compared against. With an overall average of 96.4% the ECOC SVM performed significantly better than the Tree Baggers average of 90.1%. ECOC SVM will therefore be used for the remaining experiments in this chapter due to its superior performance on supervised data.

Table 2 - SVM vs Tree Bagger on fully labelled dataset

Activity	**Accuracy**	
	SVM	**Tree Bagger**
Walking	99.4%	94.0%
Walking Upstairs	96.8%	86.8%
Walking Downstairs	96.4%	82.1%
Sitting	89.2%	88.6%
Standing	96.9%	89.2%
Laying	99.8%	99.8%

3.2.2.2. Experience sampling

With a baseline of maximum performance for recognition of the activities in this dataset now established, we can now test the performance of the different experience sampling methods. Table 3 shows the number of labels found through experience sampling for each length of the sampling window. The sampling window is the length of the time between experience sampling requests, with longer windows representing a scenario where there is an extremely low number of labels relative to the number of feature vectors. The fully supervised section represents a dataset which is fully-labelled, so even a one-minute interval significantly reduces the number of labels available for classification.

Due to the sampling frequency of the HAPT dataset, for each of the labels represented in Table 3, except for the Fully Supervised section, contains 23 feature vectors per minute of sampling. This means the one-minute window collects a single label which needs to be translated to 23 feature vectors and the 3 three-minute window collects a single label which needs translated to 69 feature

vectors and so on. The larger windows will be required to translate a single label to a large amount of feature vectors, representing a significantly higher level of difficulty compared to the shorter windows.

Table 3 - Number of labels produced by ES

Sampling Window (min)	Fully Supervised	1	3	5	10	30
Number of labels	7415	322	107	64	32	10

At the one-minute sampling window, the average number of labels per person was 11, which, like the numbers seen in Table 3, will reduce linearly to less than one label per person at the 30-minute window.

3.2.2.3. Classification Performance

After the experience sampling methodology has been applied and weak labels have been collected, they can be analysed in several different ways. The first method will be the single label approach where a single label will be applied to a single feature vector at the point when the experience sampling interval occurred. This method of simulating experience sampling is based on users providing labels at equal intervals and at the exact time in which the activity was performed. Therefore, while this method of simulating experience sampling is not feasible in real world scenarios, it provides an excellent evaluation scenario by removing human timing errors as a variable and therefore focuses the experiment purely on evaluating the effect reduced labels have on classification performance.

Following the initial experience sampling interval evaluations, multiple-instance learning based algorithms are evaluated and compared to investigate which methods produce the best results with the longest sampling window possible.

3.2.2.3.1. Collecting a label for a single feature vector performance

In this experiment, we will apply each collected label to a single feature vector which lies directly on the moment the request was made. The advantage of this is that there is a high chance that the feature vector has been labelled correctly but a major disadvantage is that the number of labelled feature vectors will be low as we are constrained to just the one labelled vector per request.

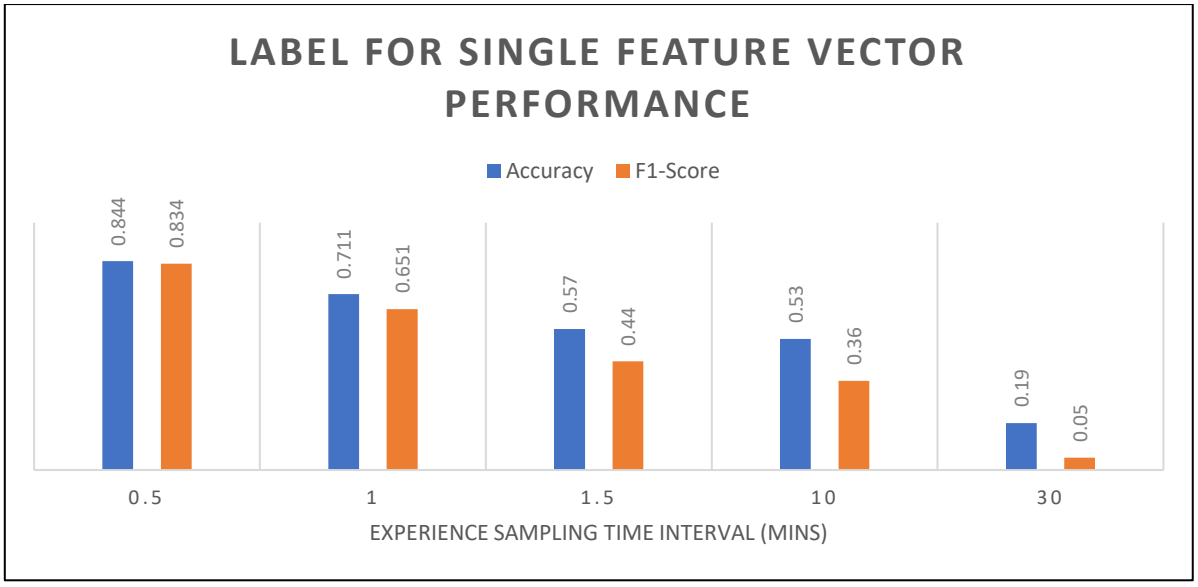

Figure 5 - Label for Single Feature Vector Performance

At each of the experience sampling intervals in Figure 5, which represent the time between the requests being made, a label was collected and applied to the feature vector which resided directly on the experience sampling interval. Figure 5, then shows the results of classifying the labelled feature vectors split by a non-cross validating train/test split with the ECOC-SVM. In this case the training set contained 70% of the data and the test set contained approximately 30% of the data. We can see that even the smaller sampling window of 0.5 minutes causes a reduction in performance compared to fully supervised which had 96.4% accuracy versus 84.4% produced by this experiment. This reduction in performance then gets gradually worse for the longer sampling windows.

To further investigate why the performance degrades so quickly, we can look at the classification performance of each individual activity at 2 different sampling windows.

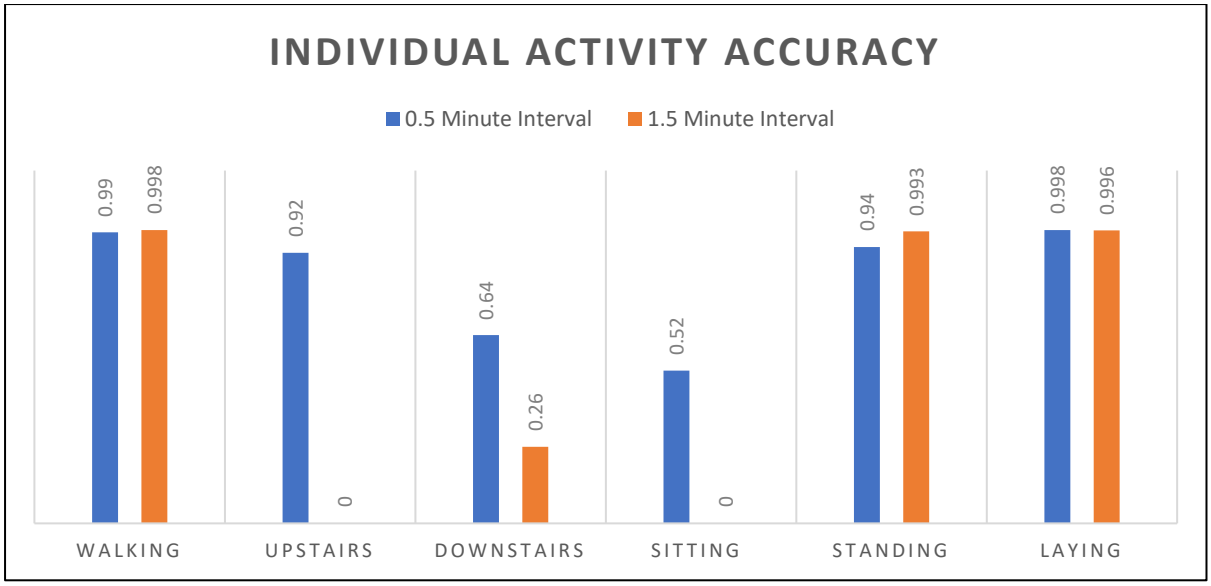

Figure 6 - Individual Activity Accuracy

From figure 6 it is clear why the classification accuracies are dropping so quickly with only an additional minute added to the sampling interval. Results indicate that activities such as sitting and walking downstairs are being poorly classified. Upon further investigation, poor classification is a result of the activities in question being short in duration and therefore less likely for an experience sampling interval to fall upon them.

3.2.2.3.2. Multiple-Instance Learning based

With the performance of a single labelled feature vector per experience sampling interval evaluated, methods of propagating the label to more than one feature vector can be tested.

Single-Instance Learning

Single instance learning will indiscriminately assign the bag label to all feature vectors in the given bag. This differs from the approach in Section 3.2.3.3.1, in that rather than applying the label directly to the feature vector that the request resided on, it places all feature vectors between the requests in a bag and applies the collected label to them all.

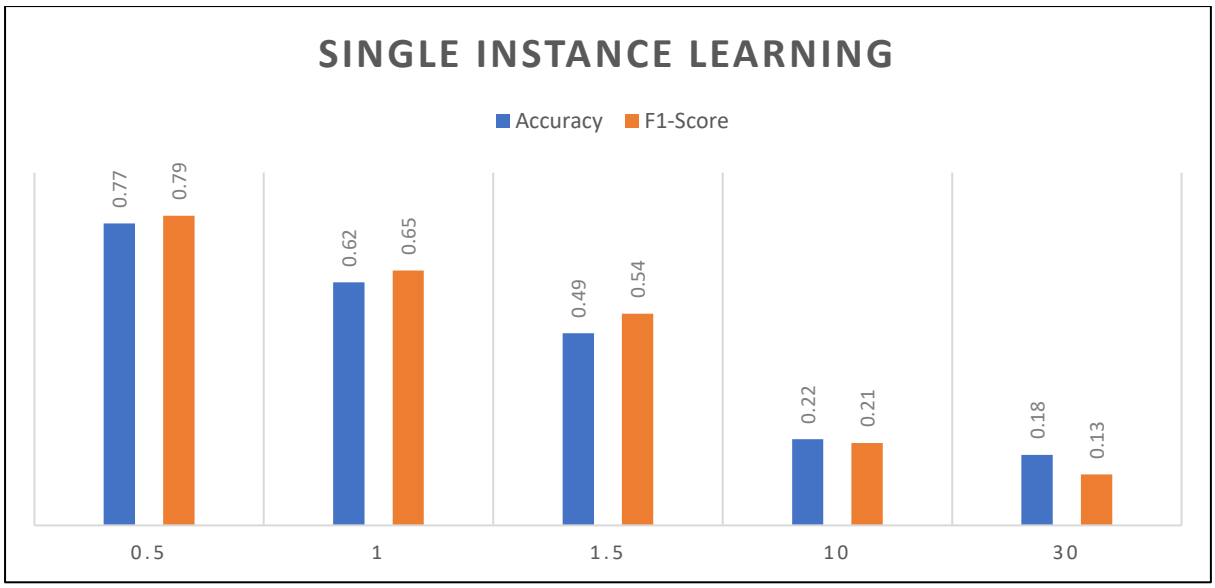

Figure 7 - Single Instance Learning Performance

Figure 7 shows the F1-Score and accuracy for propagating a single label to each bag. As expected, the performance is significantly reduced each time the time interval is increased. Table 4 shows the comparison of the F1-scores and accuracies found in Figure 7 with the results from the previous experiment, Section 3.2.2.3.1, which used a single label for a single feature vector. When comparing the accuracy results using ANOVA, the drop-in performance is drop statistically significant with a *p*-value of 0.498. This maybe because the method from Section 3.2.2.3.1 will have a high precision but low recall due to it collected one accurate label, whereas SIL will likely have a lower precision but higher recall.

Table 4 - Single Label vs SIL Performance

Single label for Single Feature Vector			SIL	
Accuracy	**F1-Score**	**ES Time Interval**	**Accuracy**	**F1-Score**
0.84	0.83	0.5 mins	0.77	0.79
0.71	0.66	1 mins	0.62	0.65
0.57	0.44	1.5 mins	0.49	0.54
0.53	0.36	10 mins	0.22	0.21

0.19	0.05	30 mins	0.18	0.13

Clustered Label Collection

All the methods tested up until this point could be considered 'naïve' in that they do not attempt to use any intelligent method of analysing features and labels. The previously evaluated methods have therefore either resulted in a very high number of noisy labels, or insufficient labels.

In the following sections novel clustering based techniques will be applied in order to identify relevant feature vectors within each bag and overcome some of the limitations of the previously evaluated methods. In particular, the clustering methods will aim to find the biggest cluster in the bag of feature vectors. This biggest cluster links to our experience sampling question of "what activity have you been performing the most". We postulate that if we find this biggest cluster then it is likely that we will have found the feature vectors which relate to the activity which was performed the most. However, an issue exists of how we perform clustering if the number of activities performed within the bag is unknown. The number of clusters within the bag is therefore unknown. A clustering method which does not require the number of clusters to be specified, such as G-Means and DBSCAN will be tested.

Firstly, we will compare G-Means and DBSCAN on the quality of the labels they have produced, and afterwards we will train a classification model using these labels to evaluate which algorithm performs best overall.

DBSCAN

Two parameters are required when tuning the DBSCAN algorithm. The ideal values for both the eps and minPoints parameters have been found in previous experimentation through a Grid Search. The value of 10 is used for both parameters. The results of the DBSCAN experiment can be seen in Figure 8. It should be noted that these results do not represent the performance of a machine learning classifier, they are a measure of the quality of the labels produced by the proposed weak label processing algorithm when compared to the original ground truth.

The F1-Score and Precision values are calculated through direct comparison of the generated weak labels with the ground truth. The true positives, true negatives, false positives and false negatives are all calculated for an individual class and averaged. This then allows the calculation of F1-score and Precision via the methods shown in Section 3.2.1.

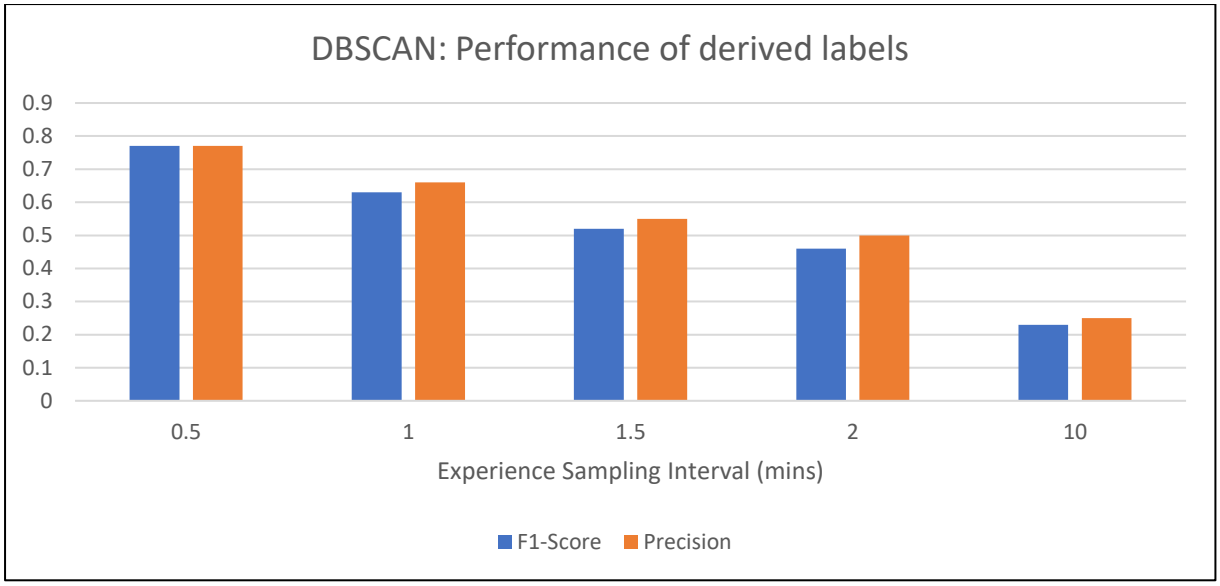

Figure 8 - DBSCAN Weak Label Propagation Performance

G-Means

G-Means contains a tuning parameter which controls the strictness of the algorithm in evaluating that the data comes from a Gaussian distribution. The value has been tuned by our own previous experimentation, explained in Section, and the value of 4 is found to perform best. This is the maximum value which can be applied, lower values increase the F1-score but also increase the false discovery rate. The results of the G-Means experiment can be seen in Figure 9, again as mentioned in Section 3.2.1, these results show the quality of the labels produced when compared to ground truth and do not represent performance of a machine learning classifier. Performance of machine learning classifiers will be tested in the next section.

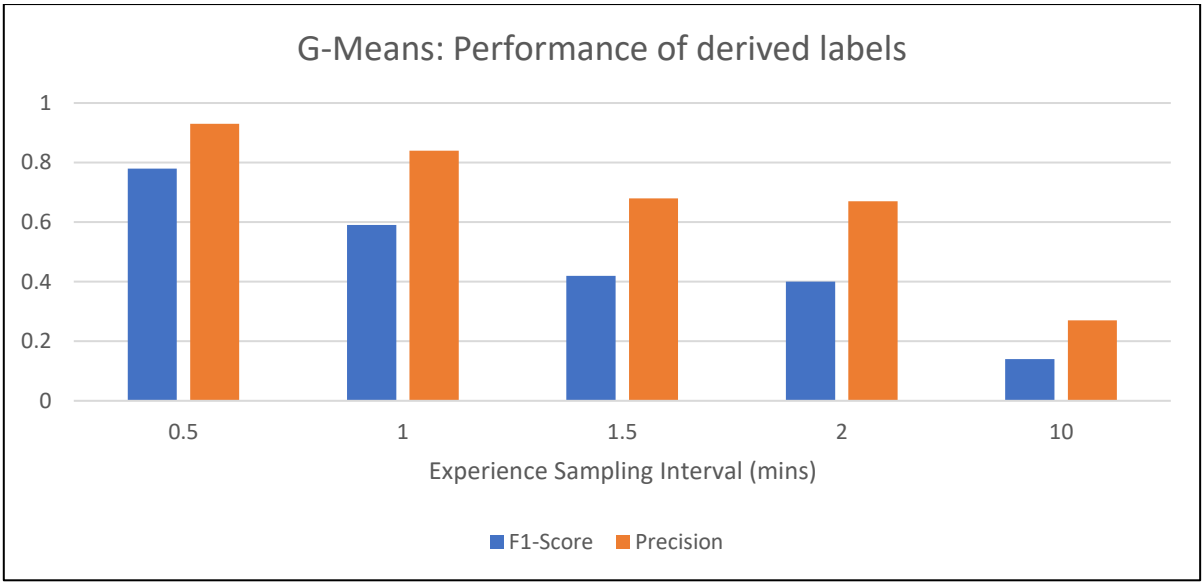

Figure 9 - G-Means Weak Label Propagation Performance

Classification

As up to this point, we have evaluated the quality of the labels produced compared to the ground truth of the training set. In this section, we will evaluate the performance of an SVM classifier when trained using labels generated by the proposed methods. The fully supervised performance shown in Table 3 provides a baseline as the highest level of performance which can be attained. Figure 10 shows the results of an SVM trained on both the labels found using the DBSCAN and G-Means algorithm.

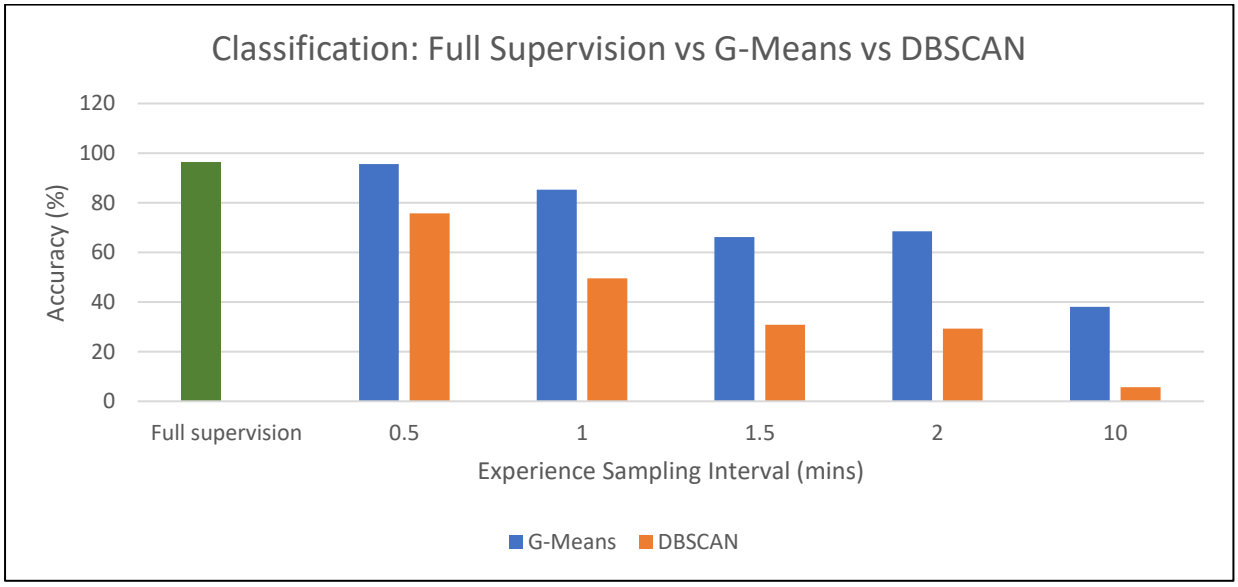

Figure 10 - Performance comparison of Weak Label Propagation Methods

From results shown in Figure 10, the proposed algorithm, which utilizes the G-Means clustering algorithm is fully capable of producing results similar to that seen in fully-labelled training sets when a shorter experience sampling interval is used. The algorithm which employs the DBSCAN clustering method however does not perform nearly as well and performance quickly drops below what could reasonably be considered acceptable.

3.2.3. HMP Dataset

The previous section described experiments on the HAPT dataset, evaluating proposed techniques on locomotive activities. This section will now describe experiments conducted on the HMP dataset which will evaluate techniques using activities of daily living.

Using the same methodology as the HAPT experiments, a baseline for the maximum performance available from this dataset. Based on results from experiments on the HAPT dataset, it was shown that the proposed G-Means based technique attained the highest overall classification performance. This G-means based technique will therefore be used for weak supervision classification in this section.

3.2.3.1. Fully Supervised Performance

To set a baseline performance the HMP dataset was used to train and test a classifier using fully supervised data Fully supervised experiments have been run with 10-fold cross-validation. The average of the ten-folds is used to represent the results seen in Figure 11. The recognition rates are relatively low compared to what is seen with locomotive activities in the previous section. The average classification performance seen is 66.1%

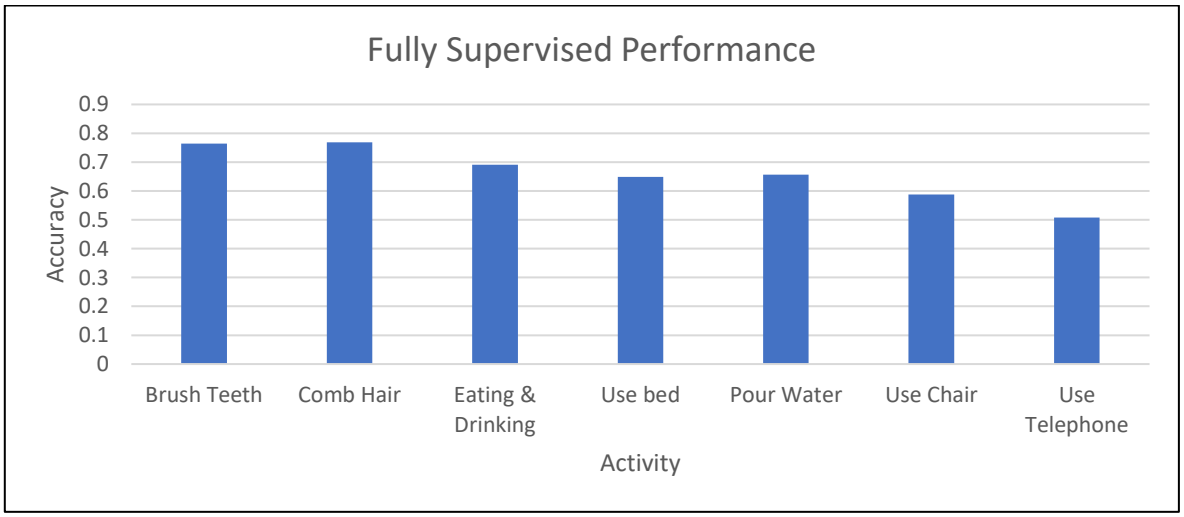

Figure 11 - HMP Dataset: Fully Supervised Performance

3.2.3.2. Experience sampling

Results in the previous Section 3.2.2 show a baseline performance that is much lower than the baseline performance of the HAPT dataset. While this is normal for ADL datasets, and is in line with the literature on ADL recognition, it presents a challenging problem for weak labelling.

Using the same methodology as the HAPT dataset, different experience sampling window durations will be evaluated. Table 5 shows how many labels will be collected by experience sampling at each of the different sampling window durations.

Table 5 - HMP Dataset: Labels Produced by ES

Sampling Window (min)	0	1	3	5	10	15	20	30	60
Number of labels	9073	220	73	44	21	14	10	6	2

3.2.3.3. Classification

Will baseline performance established, evaluations of the weak supervision algorithms on the HMP dataset will be described in the following section.

The evaluations follow the same methodology previously described to evaluate the HAPT dataset in Section 3.2.2. The main difference is in this HMP experiment is that we are running cross-validation rather than a train/test split. In this case, the dataset will be segregated by a sliding window which will move across the dataset 10 times. The data inside of the sliding window becomes the test data for that iteration and the rest of the data becomes the training data. For each of these iterations, the training data will be turned into MIL bags and experience sampling used to capture a single label for each of these bags.

After the experience sampling process has gathered bag labels, the proposed G-mean based weak supervision algorithm will be used to generate labels for the final classification. As before, these labels will be generated from the bag-level labels captured by experience sampling. After weak labels have been processed, we again have a supervised problem which can be modelled by any supervised classifier.

Figure 12 shows the classification results of when the SVM has been trained using the weak labels generated by the experience sampling and clustering algorithms on the HMP dataset at several different time intervals and then tested against the test data. Results show that, even at the smaller

window sizes, classification performance is poor with, for example, the 30 second window achieving an accuracy of 0.45.

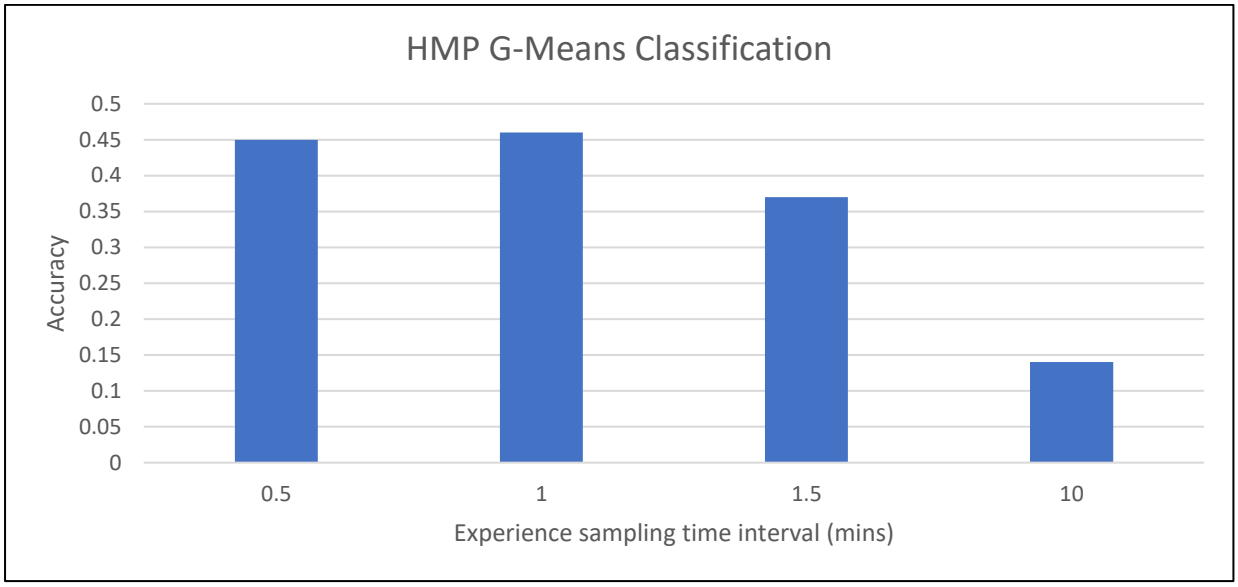

Figure 12 - HMP: Weak Label Classification Performance

To investigate why the overall performance is poor, the results of the G-Means based weak label algorithm is analysed. Table 6 shows the confusion matrix of the labels generated to the training set by the weak label generation system at the 30-second interval. The confusion matrix presents the number of correct predictions versus the total number of predictions in tabular form. However, as we are using cross-validation, one confusion matrix would be produced for each of the folds, therefore, an average of the confusion matrices across all folds is created and display in Table 6.

	Brush Teeth	Comb Hair	Eating & Drinking	Use bed	Pour Water	Use Chair	Use Telephone
Brush Teeth	385	1	0	2	0	0	1
Comb Hair	2	309	6	0	0	4	2
Eating & Drinking	0	7	1089	18	2	3	0
Use bed	0	0	16	884	13	2	0
Pour Water	0	0	1	13	655	14	0
Use Chair	0	0	20	11	23	702	0
Use Telephone	0	1	3	0	1	3	170

Table 6 - HMP: Weak Label Confusion Matrix

Table 6 shows that the labels being generated by the G-Means system are good, with only a small number of incorrect classifications. This shows that the G-Means based weak supervision algorithm is effective in generating these labels. So, if effective labels are being generated by the weak labelling system, then why was classification performance so bad in Figure 12? To investigate this further, we re-train the SVM on the weak labels in the training data generated by the weak labelling system at the 30 second interval, test them against the test data and present these results in Figure 13. This time, however, we will break down the results by activity and we see that performance is good except for the "use bed" activity.

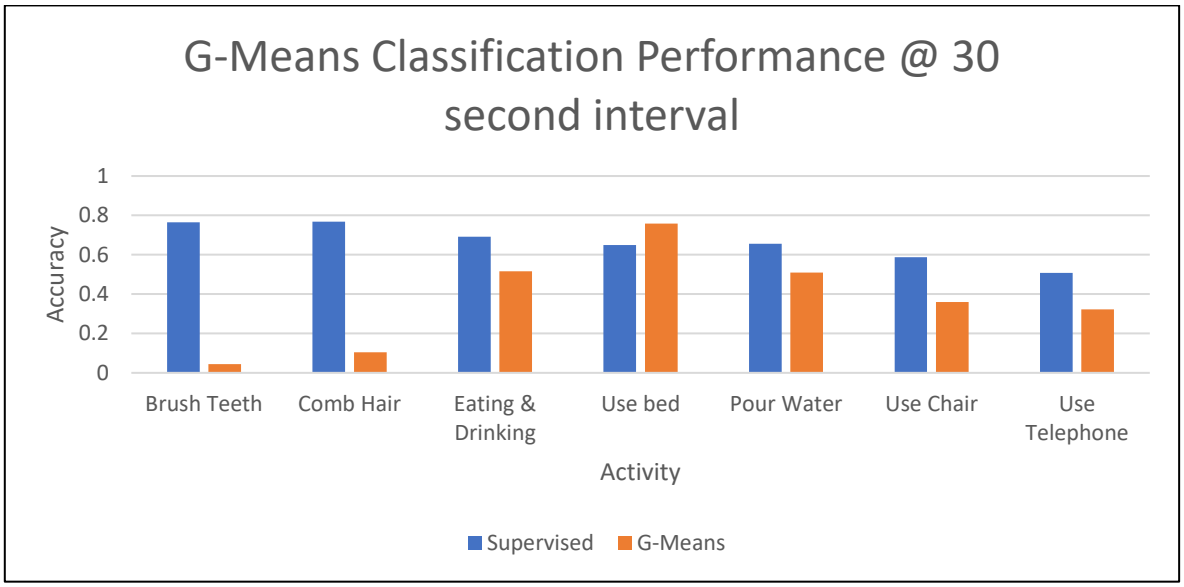

Figure 13 - HMP: Weak Label Individual Activity Performance

A potential cause of this poor performance is the unbalanced classes. For example, in Table 6 it can be seen that brush teeth, comb hair and use telephone have very small numbers of features vectors available for training relative to the others. This does appear to have an impact as the activities with the highest numbers of training examples seem to be the highest performing.

To further investigate this, an alternative classifier, which can better handle imbalanced datasets, will be considered. Specifically, a gradient boosting based model will be utilized, in a study performed to test the ability of several classifiers to handle imbalanced data, Gradient Boosting performed favourably [141]. The same protocol as before will be run, with the experiment first establishing a baseline performance of Gradient Boosting being trained on the fully supervised dataset. After the

baseline is established, Gradient Boosting was trained on weak labels generated by the G-means based weak supervision algorithm.

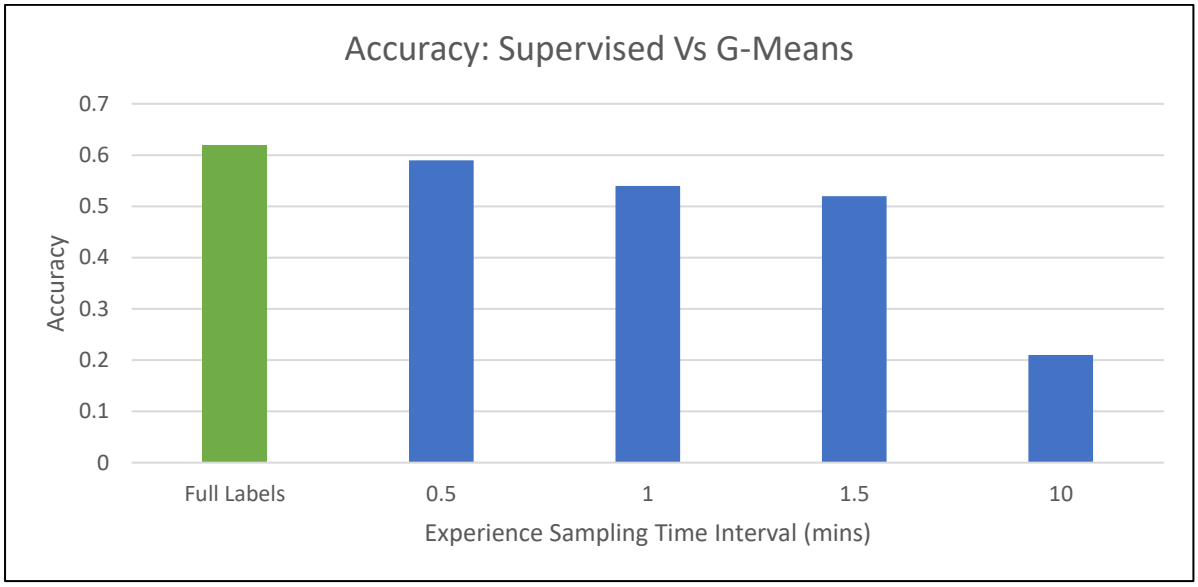

Figure 14 - HMP: Supervised vs Weak Classification Performance

Figure 14 shows that using gradient boosting has given a small reduction in the accuracy of fully supervised, with a drop from 0.66 to 0.62 but it has markedly improved the weakly supervised performance. It has taken the performance at 0.5 minutes from 0.45 to 0.59, a drop in just 3% accuracy when compared to the fully supervised baseline.

3.3. Discussion

When testing these two algorithms on the two different types of activities, it can be expected that the longer the experience sampling bags the worse performance becomes. This happens for multiple reasons:

- There are fewer overall labels – Due to the limitations of this method of label collection, the only labels which will be generated are the ones which are provided for the bag. And even when the system does try to find all the labels in the bag it is still likely to miss some. This problem becomes much worse if the bags are longer as they are more likely to contain a larger number of unique activities, and these additional labels have no way to be labelled.
- Longer bags mean short activities will never be labelled – Shorter activities, for instance, picking something up maybe missed entirely as the system is asking the question “what was the activity you performed the most within the defined window of time”. If the bag is long and the activity is short, then the shorter activity will be missed entirely

- The quality of the labels may drop – Similar to the previous points, the longer the bags the more chance additional activities will have occurred within these bags and therefore the higher chance that the system will incorrectly label a feature vector. These incorrectly labelled feature vectors can negatively affect the final classification performance. The performance could be affected worse than a missed feature vector as while a missed feature vector provides no value, an incorrect one will confuse the decision plane.

These are inherent weaknesses which come with weak supervision, however, collecting data over a longer timeframe may counteract the severity of this problem. However, due to the limitations of many datasets and the infeasibility in capturing very long-term ground truth, it would be extremely difficult to test this hypothesis.

When the performance of the two algorithms in propagating the labels is tested for locomotive activities, the F1-score appears to show that DBSCAN either outperforms or matches the performance of G-Means, albeit with lower precision. The higher F1-score is likely due to higher recall values, but, the lower precision means a significantly higher number of feature vectors are being labelled incorrectly. As mentioned previously, these incorrectly labelled feature vectors provide conflicting information for a classifier to learn from. A similar effect is seen with the activities of daily living, again, with the precision being higher with G-Means yet the F1-Score is higher with DBSCAN.

Of the two classifiers tested, Table 2 shows that the SVM performed best in the supervised baseline and was chosen to be used for the rest of the experiment. The impact of DBSCAN's lower precision is apparent when we consider the final classification performance in Figure 10. At the shortest bag size of 0.5 minutes, the weak labels match the performance of fully supervised for the locomotive activities, but in both cases, the performance drops as the bags get longer.

On the locomotive activities, at the one-minute experience sampling interval, performance has dropped below fully supervised, however, it could still be considered acceptable. Although still a short time interval to be asking for user-input, it is still a much more feasible method of data collection compared to normal methods. The training set contained 546 points where activity changes. Under normal data capture collection, a label would be required for the start and endpoint of each of these activities resulting in 1092 labels which need to be captured. The 0.5 and 1-minute bags respectively only require 617 and 322 labels to be captured, these figures represent 8% and 4% of the number of labels used by the fully supervised training set.

The activities of daily living remain a challenge for activity recognition systems and it was no better in this case. With a baseline performance of 66.1% accuracy on the fully supervised system, it was

expected that weak supervision was going to be poor. Interestingly though, performance drops at the 0.5-minute window are insignificant.

3.4. Limitations

As each of the sub-chapters within this section have used a pre-collected dataset and could only simulate the user-requests, it brings about limitations of the results.

When a user is requested to input the activity they are performing, they could potentially become confused as to what label to provide. For instance, if an office worker is working on their PC and they get a notification to input their current activity. Do they respond to say they are sitting, or working, or working on a computer, or reading a screen? There are a huge number of variations of activities they could provide which could end up confusing the activity recognition system. As all these experiments have been performed on pre-collected data, we had no way to model or simulate this problem. However, a potential solution would be to provide some form of user-training prior to them using the system.

3.5. Conclusion

While these experiments have only been simulated using a previously captured dataset, the results show that with shorter experience sampling bag lengths, we can either match or have modest reductions in classifier accuracy for a significant reduction in the number of labels required. Longer experience sampling bag shows the limits of the dataset with insufficient labels being captured during the training phase to identify all the activities in the test set. To conclude this chapter, we return to the original three questions.

1. Can activities be successfully recognized with weaker than normal labels?
2. If they can, what activities can be recognized?
3. Just how weak can the labels be made before performance degrades below what could be considered reasonable performance?

Firstly, can activities be successfully recognized with weaker than normal labels?

Yes. The labels generated by the G-Means system can classify activities at near fully supervised levels of accuracy. Not only does this reduce the overall number of labels required it also removes the requirement to accurately capture the beginning and end times of activities and, instead, allow the user to provide the labels.

Secondly, what activities can be recognized?

The G-Means based algorithm has shown capability in classifying both activities of daily living and locomotive activities near the performance of full supervision. Although we do see reductions in overall classification performance, these reductions may be acceptable in certain scenarios. E.g. a scenario such as tracking an elderly person where it is not vital that every moment is perfectly tracked.

Finally, just how weak can the labels be made before performance degrades below what could be considered reasonable performance?

Obviously, the shorter the sampling window, the more accurate the labels provided to the system are and therefore the higher the system's performance. Due to the limitations associated with pre-collected datasets, we are limited in what can be realistically tested. We postulate that over a long enough period a system will have good enough labels collected for reasonable classification performance. However, in the case of these experiments, we have seen 1 to 1.5-minute sampling windows as the maximum before performance degrades badly.

The next chapter will try and address some of these issues by considering the problems we have encountered in this chapter and trying to provide solutions.

4. Reducing the overall number of user-requests

In the previous chapter, experiments were conducted on different methods of capturing labels from users and conducted on different methods of propagating these labels to feature vectors. While the previous chapter addressed the issue of using weak labels for activity recognition, several problems still exist. In particular, asking users for labels can still be considered intrusive and can affect the likelihood of accepting the system and continuing to provide labels.

Therefore, this chapter aims to answer the following question.

> *Can we reduce the overall number of user requests for the current activity by analyzing information from previous user requests and using this knowledge to predict the confidence of the classifier in future requests?*

Any reduction in the overall number of requests will increase the likelihood of that system being accepted by the end-user. In this chapter we will first introduce our methodology and the dataset in which we will test our hypothesis and move into results and finally conclusions.

4.1. Methodology

This methodology will begin by first discussing the ways in which user-requests can be reduced and continue to explain the decision-making theories in choosing when to make predictions and when to make a user-request.

4.1.1. Dataset

This chapter will use the Human Activities and Postural Transitions dataset. Further details of this dataset have already been explained in chapter 3.1.1.1. We have chosen to focus these experiments only on the HAPT dataset as the activity data is more balanced than the HMP dataset seen in Chapter 3.1.1.2.

4.1.2. Feature extraction

The features used for the following experiments dataset are pre-computed. They are computed from the raw 3-axis accelerometer and gyroscope signals and are a mixture of time domain and frequency domain features. The signals are de-noised using a median and low pass Butterworth filter. Five different time series are produced from these signals. Acceleration values produced from “body” movements and the acceleration values produced from gravity. The signals are split using a low pass Butterworth filter. The jerk, which is a measurement of the rate of change of acceleration with respect to time, of the body signals is then produced by calculating the rate of change of the acceleration values with respect to time. The time series now produced are the acceleration due to gravity, the

body acceleration, the body gyroscope acceleration and jerk of the body signals. The magnitude of each of these is calculated using norms. Fast Fourier transforms are calculated from these signals.

The signals are split into windows which are 128 samples in length, approximately 2.6 seconds of data. Several variables are calculated from each of these signals, including mean, standard deviation, max/min values, median absolute deviation, signal magnitude area, the energy of the signal, interquartile range, signal entropy, correlation coefficients, the autoregression coefficient, angle between the vectors, skewness, kurtosis, mean frequency, frequency component which has the largest magnitude and energy of the frequency interval of the Fourier transform window. The mean of the angle between the vectors is also calculated. This results in a 561-length feature vector.

4.1.3. Experience sampling

As described in chapter 3.1.2, experience sampling is a diary study method for gathering data from a user. Experience sampling is simulated in this experiment by asking a user for data relating to each activity they are currently performing. This process is simulated by finding which feature vectors lie on the experience sampling intervals. E.g. for a one-minute experience sampling window, the feature vector which lies on exactly one minute will be used. The annotation for this feature vector is then obtained from the original set of annotations.

4.1.4. Ensemble Learners

An ensemble learner is a type of machine learning algorithm which uses a set of classifiers to aid in deciding. An example of this is the Random Forest classifier, which uses an ensemble of Decision Trees. Ensembles can provide better results than a single classifier, for instance, a Decision Tree is provided with the entire training set and it will create decision boundaries which allow it to perfectly identify the examples which exist in the training set. However, this perfection in identifying training set samples can result in poor performance in unseen data due to it being overfitted and under generalised.

Random Forest can provide solution to this overfitting issue by providing a random sample of data to an ensemble of Decision Trees. Each individual tree is not provided with the full dataset or the chance to make perfect decisions on the entire dataset and therefore should be more generalised.

When unseen data is presented, the Random Forest will feed this data into each of the Decision Trees and each of these trees will provide their vote for the class they think that the data belongs to. Generally, the Random Forest will then take the mode of these votes as the overall output label.

Additionally, ensemble learners provide the chance to quantify the amount of confidence a classifier has in any given prediction. In this work, we propose to harness this characteristic of ensemble

learners to evaluate the spread of the votes provided by the internal classifiers and compute a measure of confidence. For example, if internal classifiers return many different classes, then confidence in that prediction is likely to be low.

4.1.5. Reducing user requests

Usability of user-trained activity recognition systems can be increased by reducing the frequency of requests or increasing the time between requests. We propose two methods of achieving this, both of which estimate the confidence in a prediction by analysing an ensemble learners' votes. When the confidence is above a threshold, the learner's will use its prediction but if the confidence is below a threshold then the system will simulate a user request. One potential advantage this brings is that with repeated use it is likely that the system will become more confident and therefore capable of making more accurate predictions. However, a potential issue could be the computational intensity required for online classification of a constant stream of data, and the subsequent retraining of classifiers with these newly labelled data points.

Figure 15 shows how the system will operate. Firstly, the ensemble learner needs to be trained, this is a critical step as poor training will undermine all future predictions. Therefore, a short training phase of the user providing some initial correct labels could be undertaken. Each new feature vector which is generated from the raw activity data is placed into an ensemble learner. This ensemble learner then makes predictions and the average of these votes is found for each activity predicted. If this average exceeds a set threshold then the system will make a prediction. If the average is low, then it is likely that the system is not confident in this data point and a user-request will be made.

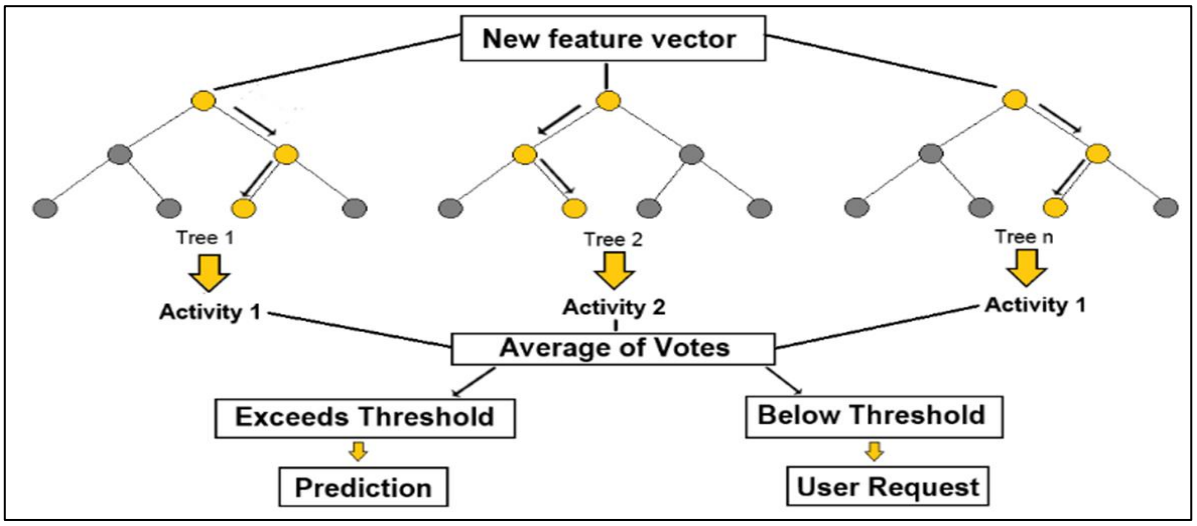

Figure 15 - Ensemble Learner Confidence

As shown in Figure 15, two methods are proposed for confidence classification. The first method reduces the number of requests and while the second method attempts to classify each data point while maintaining a minimum distance between data requests.

For each of these methods, the Random Forest classifier is chosen as it has proven efficacy in classifying activity data [142] and provides access to the internal Decision Tree votes. It is first trained on the labelled feature vectors gathered from the training phase. To ensure that each experiment is repeatable, a random seed is set. Classifier confidence can then be inferred by averaging these votes, and the classifier retrained each time a new label is gained. While likely to result in some noisy labels, tuning of a threshold value will be key. The optimal value is low enough to allow the system to make predictions but also high enough to reduce the likelihood of incorrect labels.

The systems earliest predictions are those most likely to be incorrect. This is because the system will not yet have been provided with sufficient data to model the relationship between features and labels. To take account of this evolving performance, the proposed methodology will utilize a dynamically weighted threshold. The threshold will initially be high for early predictions and as the system learns

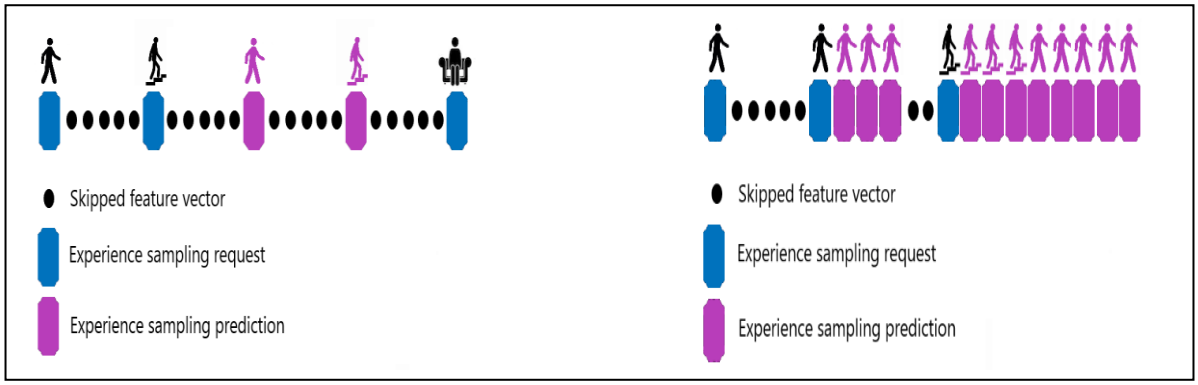

Figure 16 - Visualisation of FSP & DSP

more, the threshold will be reduced.

4.1.5.1. Fixed Sampling Prediction

A fixed sampling prediction method is proposed in order to reduce the number of user requests by predicting each of the experience sampling request points. The request points will be at fixed times, for example, every 5 minutes. At each of these requests, the classifier will make a prediction and if the average of the votes of this prediction is above a threshold value then it will be accepted. Otherwise, it will request input from the user. The pseudocode for this method is shown within Algorithm 2.

Algorithm 2 – Fixed Sampling Prediction

```
1: {Input: RF = random forest trained on all labelled feature vectors, data = feature vector of bag,
Threshold      = prediction threshold}
2: Confidence = 2-width matrix of zeros the same length as data
3: TreePredictions = GetIndividualTreeOutput(RF, data)
4: For each output o in TreePredictions:
5:      ModePrediction = mode(TreePredictions{output})
6:      Confidence{output, 0} = RF.predict{data}
7:      Confidence{output, 1} = Percentage of Trees in TreePredictions which voted for in
ModePrediction
8: BagMode = mode(Confidence{:, 0})
9: BagModePercent = Percentage of predictions in Confidence which contain the BagMode **
10: If bagModePercent >= Threshold:
11:     return bagMode
12: else
13:     return -1
```

4.1.5.2. Dynamic Sampling Prediction

A dynamic sample prediction method is also proposed in order to consider all data points and not just data points in fixed sampling windows. The system will iterate through all feature vectors and measure

Algorithm 3 – Dynamic Sampling Prediction

```
1: {Input: RF = random forest trained on all labelled feature vectors, data = feature vector of bag,
Threshold      = prediction threshold}
2: Confidence = 2-width matrix of zeros the same length as data
3: TreePredictions = GetIndividualTreeOutput(RF, data)
4: For each output o in TreePredictions:
5:      ModePrediction = mode(TreePredictions{output})
6:      Confidence{output, 0} = RF.predict{data}
7:      Confidence{output, 1} = Percentage of Trees in TreePredictions which voted for in
ModePrediction
8: BagMode = mode(Confidence{:, 0})
9: BagModePercent = Percentage of predictions in Confidence which contain the BagMode **
10: If bagModePercent >= Threshold:
11:     return bagMode
12: else
13:     return -1
```

the confidence the classifier has in each. As it iterates through the data points, once the classifier finds a feature vector which it is not confident in classifying, it will make a user-request. Limitations will be put on the number of requests made, and the system will only be permitted to make a request if a minimum duration has passed since the previous request. It is likely that a higher confidence threshold will be required for this method as significantly more data points are being checked, bringing a higher probability of noisy labels. The pseudocode for this approach is shown in Algorithm 3.

4.1.6. Classification against the test set

There are a few phases involved in testing this system against the test set.

1. The first phase is to run an initial data collection phase, this will be performed with experience sampling user-requests and at this point the algorithms won't yet be employed. This is because we need to gather some data to training the algorithms before they can start to make predictions on user-requests. For these experiments, 5 correct labels will be captured at the beginning of the experiment.
2. At this point, some labels have bene gathered and the two algorithms can be trained on these gathered labels. For the remaining data in the training set, the algorithms will begin to make predictions and measure their confidence in these predictions. If the prediction confidence is above the threshold then the user-request will be skipped. This will continue until all data within the training set has been used.
3. Now that the algorithms have been run, a classifier will be trained on all available data produced in the training set, either from experience sampling or the algorithms. This final classification will be achieved using a Support Vector Machines (SVM) classifier. As this work uses several activities, it would be advantageous to have a multi-class SVM. For this reason, and as validated in the previous chapter, an error-correcting output codes SVM (ECOC SVM) is used.

4.2. Results

To find the maximum possible accuracy that experience sampling alone can achieve, a baseline will be set by performing the experiment without either method active. A sampling window of 0 represents a fully supervised model with all feature vectors having a supervised label.

In this experiment, in order to ensure that any labelling noise is caused by the algorithms and not by other weak supervision processing. The labels collected by experience sampling will be directly applied to the feature vector on the which the request occurred on. Further information on this process can be found in Chapter 4.1.3.

Table 7 - Labels Produced By Experience Sampling

Sampling Window Length (mins)	0	1	5	10	15	20
Number of labels	**7415**	**302**	**64**	**34**	**24**	**19**
Accuracy	96.5%	91.1%	84.4%	73.5%	73.8%	71.1%

As is visible from Table 7, the shorter experience sampling windows performed better as more data is available. A 1-minute sampling window significantly reduces the number of labels required while

maintaining an accuracy of 91.1%. Finding the ideal value for the prediction threshold will be key for ensuring the best results. It will be tuned individually for each method, as the proposed dynamic sampling prediction method is likely to require a much stricter value when compared to the static sampling prediction method. For both methods, the threshold will be tuned to the 5-minute sampling window, as it provides significantly less data than standard supervised and is also unlikely to run into insufficient dataset problems.

Table 8 - User-request prediction threshold tuning

5-Minute Sampling Window								
Fixed Sampling Prediction					Dynamic Sampling Prediction			
Confidence Threshold	**User Requests**	**Predictions made**	**Incorrect Predictions**		**Confidence Threshold**	**User Requests**	**Predictions made**	**Incorrect Predictions**
50%	23	36	18		50%	7	6719	5617
60%	43	16	2		60%	30	6472	4104
70%	49	10	0		70%	53	3638	1209
80%	56	3	0		80%	56	1643	364
90%	59	0	0		90%	58	120	0

4.3. Confidence Threshold Tuning

4.3.1. Fixed Sampling Prediction

The results in table 1 show that having a very high threshold will reduce the number of incorrect user-request predictions, but it also reduces the overall number of user-request predictions made. While 60% has reduced the number of requests by the largest amount, it has resulted in two incorrect user-request predictions. A 70% threshold, however, has still reduced the number of user-request predictions by 17% and none of these predictions were incorrect.

Based on this information, a threshold value of 70% was chosen for future experimentation.

4.3.2. Dynamic Sampling Prediction

As is visible in table 1, the confidences required for the second method are higher. Anything below 90% results in an unacceptably high number of incorrect user-request predictions which are likely to cause classification problems. While 90% only reduces the overall number of user requests by 1 relative to Fixed Sampling Prediction at this threshold, it provides an extra 120 successfully labelled feature vectors which could potentially increase classification accuracy.

Using this information, a threshold value of 90% was chosen for the Dynamic Sampling Prediction threshold.

Table 9 - FSP & DSP Performance

Fixed Sampling Prediction						Dynamic Sampling Prediction				
1	**5**	**10**	**15**	**20**	**Sampling Window Length**	**1**	**5**	**10**	**15**	**20**
302	64	34	24	19	**Total Sampling Requests**	1844	183	491	21	16
191	49	26	18	11	**User Requests**	286	58	27	16	11
106	10	3	1	3	**Predictions**	1553	120	459	0	0
2	0	0	0	3	**Incorrect Predictions**	7	0	0	0	0
0.91	0.84	0.74	0.74	0.67	**Classifier Accuracy**	0.91	0.8	0.77	0.72	0.63

4.4. Classification Results Against Test Data

Now that we have found the best parameters for each of our user-request prediction methods, we can train a classifier with each of these methods and test the results against the test set. The idea here is that we ensure that the user-request predictions being produced by the algorithm do not affect the overall classifier accuracy when tested against the test data.

The process of this will first generate the experience sampling and user-request predictions from the Random Forest. Once these have been placed against their respective feature vectors, it is now a normal supervised problem where an SVM can be trained against. The SVM is trained and best parameters found through a GridSearch and the SVM then tested against the test data.

4.4.1. Fixed Sampling Prediction

Table 3 shows that with the lower sampling window lengths, the number of predictions is successfully reduced. The 1, 5 and 10-minute sampling windows having the number of requests reduced by 35%, 16%, and 9% respectively. The longer windows showed less success, with the 15minute window only providing 1 correct prediction and the 20-minute window providing no correct predictions. It is possible however that these problems are being caused by the length of the dataset. When compared with the results found in table 2, we can see that the prediction accuracy of this method is almost identical to the full experience sampling classification accuracies. The 2 incorrect predictions on the 1-minute sampling window only resulted in an overall accuracy reduction of 0.1%.

4.4.2. Dynamic Sampling Prediction

Table 3 shows that the second method provides more data for the classifier when the sampling window is kept to 10 minutes or below. In the 1, 5 and 10-minute sampling windows it provides 1546, 120 and 459 extra features for the classifier, while also reducing the number of requests by 85%, 68% and 95% respectively. However, these reductions in user-requests are not directly comparable to Fixed Sampling Prediction, which will overall have less requests due to its static sampling frequency of requests whereas Dynamic Sampling Prediction is opportunistic. Like Fixed Sampling Prediction, the

longer 15+ windows gained no benefit from this method. Like the first method, the number of user requests relative to experience sampling was reduced but not by the same margin.

4.5. Comparison of methods

Both the Fixed Sampling and Dynamic Sampling prediction methods reduced the number of user requests relative to experience sampling. However, Dynamic Sampling Prediction has provided the benefit of significantly more labelled feature vectors providing a slight increase in classification accuracy of the test set. With the 5-minute window length, Dynamic Sampling Prediction has reduced the number of requests by 9% and has gathered a much higher number of successful predictions than Fixed Sampling Prediction. Counterintuitively, though the classifier accuracy has decreased. While the experience sampling and Fixed Sampling Prediction collects the same labelled feature vectors, we postulate that the decrease in classifier accuracy is due to the dynamic sampling prediction method having the ability to pick different feature vectors for labelling. It is likely that the feature vectors selected provided less discriminative data. The 10-minute window lengths have a similar number of requests for both methods, however, the dynamic sampling prediction made 459 more successful predictions than Fixed Sampling Prediction. These extra predictions also increased the classifier accuracy; however, the increase was not significant.

Longer windows (15+ minutes) simply do not have enough information for enough training of the classifier. The Fixed Sampling method made a small number of predictions, however, most of these predictions were incorrect. The Dynamic Sampling method made no predictions for the longer windows. This was likely caused by the combination of the high threshold plus the lack of training data meaning the classifier will never be confident in any predictions. These issues are due to the limitations of collected datasets, we only have a small sample of data to work and this doesn't provide much opportunity for a system such as this to become confident in making predictions. In reality, a system with a long time interval may not begin to make predictions at the beginning, however, after enough training time has passed, should start to make predictions and ideally would get to a level of confidence that it no longer needs to make any user-requests.

The Fixed Sampling method performs very well in reducing the number of user interruptions, especially at the smallest experience sampling window lengths. At the one-minute window length, it has reduced the number of requests by 35% with minimal effect on classification accuracy of the test set. However, the Dynamic Sampling method performs better at classification and prediction. It reduced the number of requests relative to experience sampling but not by the same amount that Fixed Sampling Prediction reduced requests. It is possible in a real-world scenario that the number of requests made will decrease quicker with the second method due to the amount of extra data it is

providing to the classifier. Another issue noted is the performance cost of the second method, as it is continually providing an online prediction of each data point and retraining the classifier where required. Its performance impact is significant. This could be a serious problem on smart devices as it will impact both the performance of the device and its battery life.

Table 10 - FSP & DSP Individual Activity Performance

1-Minute Sampling Window							10-Minute Sampling Window					
Walking	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**	**Activity**	**Walking**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**
99.6%	90.2%	85.2%	80.5%	89.6%	99.8%	**Full Experience Sampling**	94%	67.1%	71.9%	98.4%	14.2%	98.7%
97.8%	91.3%	85.7%	80.5%	89.6%	99.8%	**Fixed Sampling Prediction**	94%	67.1%	71.9%	98.4%	14.2%	98.7%
98.8%	95.1%	87.6%	68.5%	96.2%	99.8%	**Dynamic Sampling Prediction**	88.9%	91.1%	33.8%	78.9%	61.3%	99.8%

Table 4 shows the classification accuracy against the test set for each individual activity for the 1 and 10-minute experience sampling windows. For the 1-minute windows, we can see that Fixed Sampling Prediction matches or exceeds the classification accuracy of the experience sampling method and this is achieved with approximately 9% less user requests required. The Dynamic Sampling method exceeds the performance of Fixed Sampling Prediction and experience sampling in all but sitting and laying activities. This confusion of sitting and laying may be caused by static activities having similar acceleration profiles. Although not reducing user intrusion as much as Fixed Sampling Prediction, it still uses 5% fewer requests than full experience sampling. For the 10-minute sampling window, Fixed Sampling Prediction and experience sampling performed the same, except Fixed Sampling Prediction achieved this performance with 9% fewer user requests. The results of the dynamic sampling methods are difficult to compare to the results of the fixed sampling method as the dynamic method would have found different feature vectors than the fixed feature vectors of the experience sampling and Fixed Sampling Prediction. Dynamic Sampling Prediction performed better on walking upstairs, standing and laying, but the remaining activities have seen reductions, particularly walking downstairs.

4.6. Conclusion

At the beginning of this chapter our challenge was.

Can we reduce the overall number of user requests for the current activity by looking at what has been learned from previous user requests and using this knowledge to analyse the confidence of the classifier in future requests?

Two methods have been developed which attempt to solve the problem in separate ways. The first method has shown an up to 35% decrease in the number of user requests while maintaining comparable levels of classification accuracy to methods which have higher levels of user intrusion. The second method also reduced the number of requests but not to the degree of the first method. It did, however, provide significantly more labelled feature vectors which increased the classification accuracy in some instances. As the intended usage of these methods is on smart devices where users can be provided with a minimally intrusive personalized system of activity recognition, the second method will need to be refined to reduce its performance impact. Potential ways of doing this could be through reducing the number of feature vectors which are predicted upon or only retraining the classifier after a certain number of predictions have been made. Future work could look at methods which can predict the state of the user to further reduce intrusion. These could be capturing data from the user about their tolerance to interruption and attempting to find activities or time frames where it is appropriate to ask for input.

5. Detecting the transitions between activities

In the previous chapter, we investigated if it was possible to make predictions on future user-requests, with the idea that this would make the system more acceptable to the end-user. However, during our investigation it was noted that when classifier confidence was extremely low in a new data point that this may refer to a transition point between activities. Having information about the transitions between activities would further increase the context of what the user is doing without having to request any additional information from them.

Therefore, this chapter aims to answer the following question:

> *Can we detect the transitions between activities? For example, sitting to standing. If the transitions can successfully be predicted, then it may provide clearer information on what activities lie between user requests.*

In this chapter we will first introduce our dataset, with the main requirement being that the dataset has labelled transition points. This is important as many activity recognition datasets will not contain or capture transitional information.

5.1. Methodology

5.1.1. Dataset

This chapter will use the Human Activities and Postural Transitions dataset. The HAPT dataset has been discussed in detail in the previous chapter 3.1.1.1. This dataset has been chosen over the HMP dataset as the HMP dataset does not contain transitions, which are the key concept of this chapter.

5.1.2. Feature extraction

The features extracted in this chapter are pre-computed and are provided with the dataset. They contain a mixture of frequency domain and time domain features, which have been computed from both a gyroscope and triaxial accelerometer. The signals are de-noised and additional context, for example, gravity and body acceleration are gathered using Butterworth filters.

From the produced signals, features are gathered from 2.6 seconds of data, which corresponds to 128 frames from the accelerometer and gyroscope. A 561-width feature vector is then produced.

Further information on this feature extraction process can be found in Chapter 4.1.2.

5.1.3. Experience sampling

As described in chapter 3.1.2, experience sampling is a diary study method for gathering data from a user. Experience sampling is simulated in this experiment by asking a user for data relating to each activity they are currently performing. This process is simulated by finding which feature vectors lie

on the experience sampling intervals. E.g. for a one-minute experience sampling window, the feature vector which lies on exactly one minute will be used. The annotation for this feature vector is then obtained from the original set of annotations.

5.1.4. Ensemble Learners

An ensemble learner is a type of machine learning algorithm which uses a set of classifiers to aid in deciding. An example of this is the Random Forest classifier, which uses an ensemble of Decision Trees. Ensembles can provide better results than a single classifier, for instance, a Decision Tree is provided with the entire training set and it will create decision boundaries which allow it to perfectly identify the examples which exist in the training set. However, this perfection in identifying training set samples can result in poor performance in unseen data due to it being overfitted and under generalised.

Random Forest can provide solution to this overfitting issue by providing a random sample of data to an ensemble of Decision Trees. Each individual tree is not provided with the full dataset or the chance to make perfect decisions on the entire dataset and therefore should be more generalised.

When unseen data is presented, the Random Forest will feed this data into each of the Decision Trees and each of these trees will provide their vote for the class they think that the data belongs to. Generally, the Random Forest will then take the mode of these votes as the overall output label.

Additionally, ensemble learners provide the chance to quantify the amount of confidence a classifier has in any given prediction. In this work, we propose to harness this characteristic of ensemble learners to evaluate the spread of the votes provided by the internal classifiers and compute a measure of confidence. For example, if internal classifiers return many different classes, then confidence in that prediction is likely to be low.

5.1.5. Activity Transitions

To further improve the usability and effectiveness of an activity recognition system we propose training a model to provide further insight into the feature vector labels and specifically detect feature vectors which correspond to transition activities. Due to the weakly supervised nature of this work, it is not feasible to classify transitions into specific types of transitions such as sitting-to-standing. We, therefore, classify all types of transitions the same. Research into ensemble learners in chapter 4.2.1 provided insight into the confidence in a prediction. In our proposed activity transition technique, we leverage the outputs of each individual tree in a Random Forest classifier and analyse these votes to measure the confidence the classifier has in its prediction in order to detect transition activities. It is not feasible to expect a user to manually provide labels, including weak labels, relating to when transitions have occurred. Experience sampling can therefore only collect labels of actual activities

rather than the transitions between them. We can, however, look for transitions in bags by identifying feature vectors which the classifier has extremely low confidence.

Algorithm 3 shows how the transition detection is implemented. A supervised random forest classifier is trained on any previously identified feature vectors and labels. Each feature vector is then tested on the trained random forest classifier and feature vectors which the classifier has low classification confidence of for all classes are given a transition label. Confidence is measured by counting the number of classes that the individual trees voted for against each feature vector, like that seen in Chapter 4.2.1. If the internal classifiers within the ensemble have made votes for every class known at that time by the classifier, then overall confidence can be inferred to be extremely low. As an example, consider an ensemble learner that had learned 6 classes of activity and is presented with new unseen data. If the internal classifiers had voted for all 6 classes of activity, with no obvious preference for one of the six classes, then the proposed method will infer that confidence in all six classes is low in the unseen data point. If the random forest is not confident that the feature vector relates to a specific activity, the feature vector is therefore classified as a transition. Figure 17 provides a visualization of how this proposed method would operate, where the experience sampling derived labels are gathered for each of the bags and then translated onto the individual feature vectors using the weak label algorithms explained in Chapter 3.1.2.1. The ensemble learner can then be trained on those feature vectors produced and the ones which remained unlabelled within the bags and are usually discarded by the weak label algorithms can be confidence tested using this ensemble learner. Feature vectors which show as extremely low confidence can then be thought of as either a transition or an unknown activity.

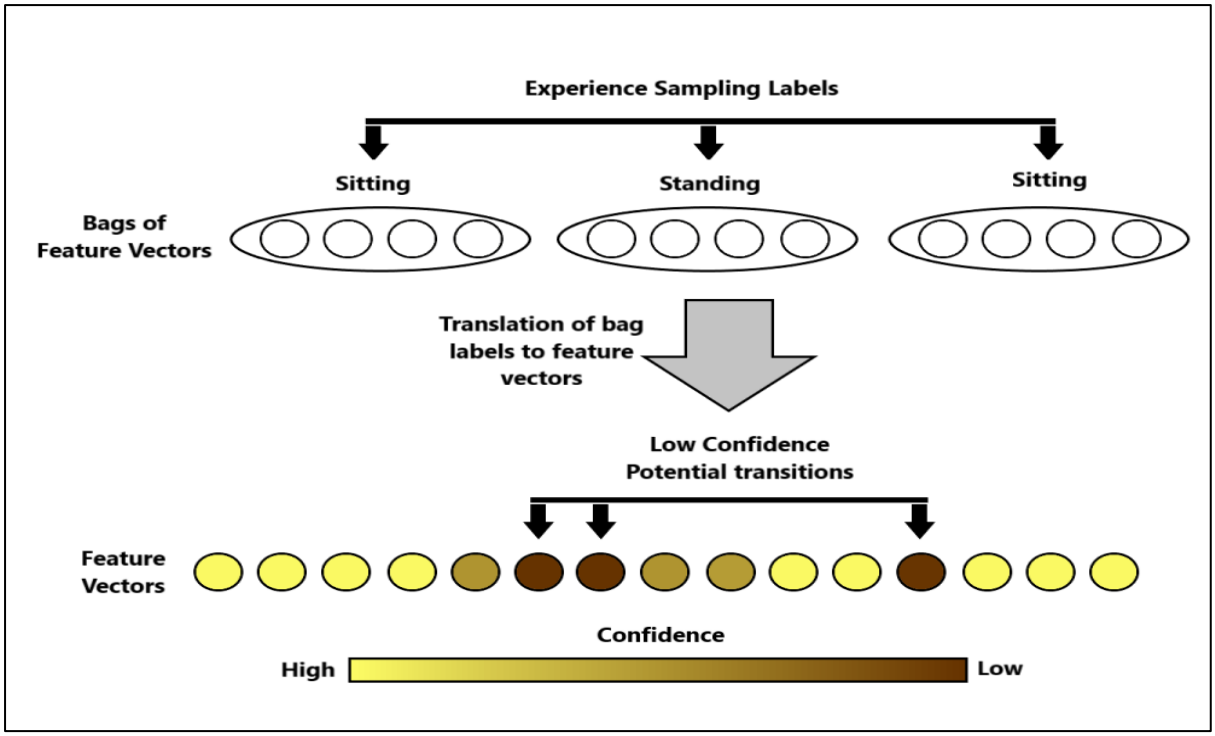

Figure 17 -Confidence Estimation of Feature Vectors

Algorithm 4 – Transition Detection

1: {Input: RF = random forest trained on all labelled feature vectors, trainingData = full training data without labels}
2: Transitions = vector of zeros the same length as training set
3: Predictions = Get individual tree output from RF on training set
4: For each output in Predictions:
5: classesOutput = number of classes predicted in output
6: if classesOutput equal to number of classes within training set:
7: transitions[output] = 1

5.2. Results

In order to evaluate the effectiveness of the proposed unsupervised transition label detection system, an experiment is conducted. The experiment was designed to compare a fully supervised model, including supervised transition labels, compared to a weakly supervised system with no transition labels. The weakly supervised dataset is generated using the G-Means based weak supervision algorithm described Chapter 3. A 30-second window experience sampling time interval is used. The fully labelled dataset was provided with fully supervised transition labels. Critically, however, no transition labels were provided to the proposed activity transition detection technique. Instead, the proposed technique was used to automatically identify activity transitions.

Once the weakly supervised training set has been gathered and the predictions made for the transitions, both the fully labelled and weakly labelled training sets are fitted to individual SVMs. The results of this fitting are validated against a test set. A comparison of accuracy between a weakly labelled training set and a fully labelled training set is shown in Figure 18. Results show that fully supervised and weakly supervised methods perform very close in almost all activities. While the fully labelled dataset predictably performs better overall, the weak labelled training set performs marginally better than the fully labelled training set in predicting the activity transitions. Crucially, it has performed with accuracy comparable to a fully supervised system even after being provided with no labels for transitions.

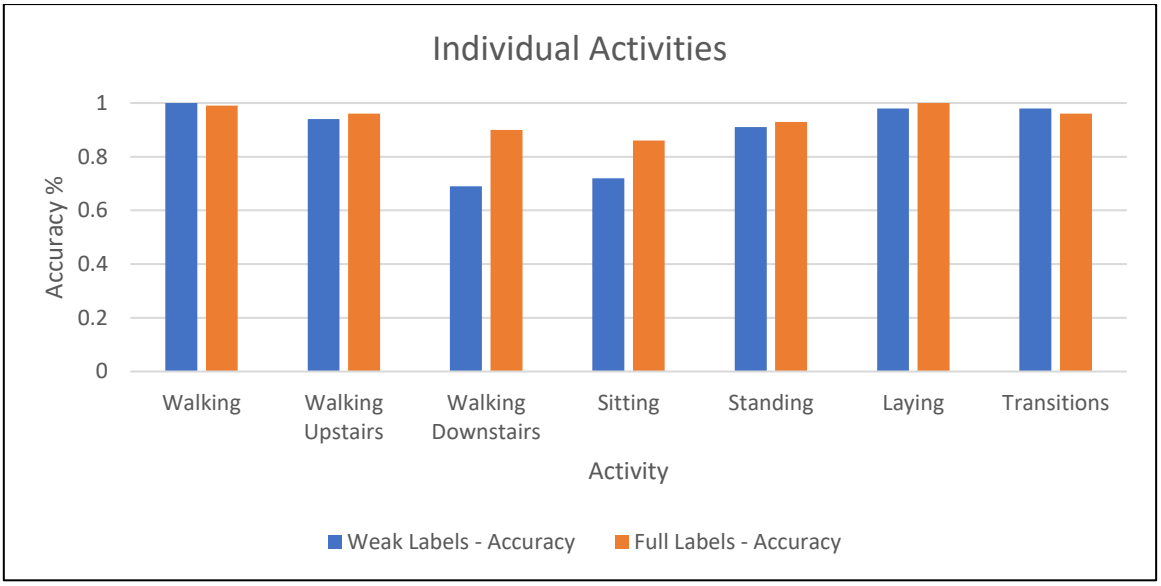

Figure 18 - Transition Performance of Full vs Weak Labels

To further understand the performance of the proposed transition label detection system, the same experiment was repeated using a 1-minute experience sampling window. Using the same methodology, an SVM trained on fully supervised data is compared with an SVM trained on using weakly supervised data. The weakly supervised data is generating using the G-mean based weak supervision algorithm and transitions are detected using the proposed transition detection technique.

Table 11 - Confusion Matrix of Weak Transition Detection

	Walking	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**	**Transitions**
Walking	494	0	2	0	0	0	0

Walking Upstairs	28	441	2	0	0	0	0
Walking Downstairs	83	46	289	0	0	0	2
Sitting	0	0	0	368	132	0	8
Standing	0	0	0	38	507	0	11
Laying	0	0	0	0	0	534	11
Transitions	2	0	0	0	1	0	163

Table 5 presents results of the experiment where a confusion matrix is used to show performance of individual activities, including transitions. Even with the even weaker information of the 1-minute window, the system is still able to successfully generate accurate transition labels.

5.3. Conclusion

At the beginning of this chapter our challenge was.

Can we detect the transitions between activities? For example, sitting to standing. If the transitions can successfully be predicted, then it may provide clearer information on what activities lie between user requests.

In this chapter we have shown that when presented with a dataset that contains the transitional information between activities, it is indeed possible to capture these transitions without additional labelling burden on the user.

This was performed by using the idea of classifier confidence with ensemble learners. Beginning with no labelled transitions, we had been able to discover the feature vectors within the training set that contained transition data, and successfully labelled them to a level of performance that matched that of a classifier provided with fully labelled transitions. Therefore, this system can detect unseen activities, in this case the transitions. This means that a potential future development of this system would be to automatically detect experience sampling request points, rather than having a fixed time interval.

6. Label Propagation for increasing classifier performance

Across this thesis, several methods of handling weak labels and translating them onto feature vectors have been presented. However, even after the weak labels have been applied it is still likely that some unlabelled feature vectors will exist.

Instead of discarding these additional unlabelled feature vectors, we can try and gain extra insight from them. One method to do this could be to look at the similarity between feature vectors by using distance metrics, such as the Euclidean distance from a labelled feature vector to an unlabelled feature vector. The aim is that if we populate labels from labelled feature vectors to unlabelled feature vectors, we can provide additional information for a classifier to learn from. However, methods such as these could potentially create noisy labels.

Therefore, in this chapter we will try to answer the following question:

> *Can we use similarity metrics between feature vectors to infer similar labels? If performed successfully, this could allow significant increases to the number of labelled feature vectors and classifier performance.*

This section will therefore detail our methodology used to propagate labels from labelled to unlabelled feature vectors in experience sampling-based activity recognition systems and present its results.

6.1. Methodology

Within this methodology, two algorithms are proposed, both of which will attempt to populate a small number of labels to a larger set of unlabelled data.

Both algorithms will use distance-based methods to populate the labels, with the main metric being pairwise Euclidean distance.

6.1.1. Dataset

All three of the sub-chapters use the Human Activities and Postural Transitions (HAPT) dataset. Further details of this dataset have already been explained in chapter 3.1.1.1. We have chosen to focus experiments only on the HAPT dataset as the activity data is more balanced than the HMP dataset seen in Chapter 3.1.1.2. The HMP dataset also does not meet some of the requirements of the experiments within these sub-chapters, for instance, one of the chapters tracks the transitions between the activities, these transitions are present within the HAPT dataset but not within the HMP dataset.

6.1.2. Feature extraction

Features are computed from both accelerometer and gyroscope data. Both embedded sensors capture tri-axial data providing 6 axes from which features can be computed.

Raw values are applied to a median filter, and then de-noised using a low pass Butterworth filter. The filter had a corner frequency specified at 20Hz. A second Butterworth filter with a corner frequency of 0.3Hz is then applied to the accelerometer data; this splits the accelerometer data into body acceleration and acceleration due to gravity. This body acceleration, along with angular velocity is then used to calculate signal Jerk.

Signal magnitude is also calculated from the body acceleration, gravity acceleration, body Jerk, body gyroscopic data, and the gyroscopic Jerk. This magnitude is calculated using Euclidean Norms.

A fast Fourier transform is applied to body acceleration, body Jerk, body gyroscopic data, bodily acceleration magnitude, gyroscopic magnitude, and gyroscopic Jerk magnitude. From these signals we then calculate mean, standard deviation, median absolute deviation, maximum value, minimum value, signal magnitude area, energy measure, interquartile range, signal entropy, correlation coefficients between two signals, auto-regression coefficient, frequency component with the largest magnitude, mean frequency with a weight average, skewness, kurtosis, frequency interval energy of the Fourier transform window and the angle between vectors. These calculations result in a 561-width feature vector, each of which is captured from a sliding window of approximately 2.56 seconds.

6.1.3. Experience sampling

As described in chapter 3.1.2, experience sampling is a diary study method for gathering data from a user. Experience sampling is simulated in this experiment by asking a user for data relating to each activity they are currently performing. This process is simulated by finding which feature vectors lie on the experience sampling intervals. E.g. for a one-minute experience sampling window, the feature vector which lies on exactly one minute will be used. The annotation for this feature vector is then obtained from the original set of annotations.

6.1.4. Single Label Propagation

Algorithm 4 Single label propagation

1: $xPoints = index\ of\ each\ position\ Y\ which\ is\ not\ zero$
2: $for\ each\ x\ in\ xPoints$
3: $cLabel = value\ of\ Y\ at\ position\ x$
4: $cFeature = feature\ vector\ of\ X\ at\ position\ x$
5: $For\ each\ z\ in\ X$
6: $distances(z) = pairwise\ distance\ of\ cFeature\ and\ X(z)$
7: end
8: $lowest = \min(distances)\ which\ is\ not\ index\ x$
9: $Y(lowest) = cLabel$
10: end
11: $For\ x = 1{:}K$
12: temp = xPoints
13: $xPoints = index\ of\ each\ position\ Y\ which\ is\ not\ zero$
14: $Remove\ temp\ from\ xPoints$
15: $Repeat\ lines\ 2\ to\ 10\ with\ new\ xPoints$
16: end

Algorithm 4 shows a method for Single Label Propagation, it operates by procedurally iterating through each of the labelled feature vectors and comparing its distance from each of the unlabelled feature vectors. After the closest unlabelled feature vector is found, the label associated with the closest feature vector is copied onto the unlabelled feature. This process is repeated until all the labelled feature vectors have copied a single label.

Once this entire process has been completed, it will be repeated “K” times. This K-value becomes a parameter which must be tuned. On each successive run, the previous set of newly labelled feature vectors are removed to prevent the system from continually re-finding the same feature vectors.

As this process iteratively works through the dataset, higher values of K are inevitably going to have an exponential impact on compute performance.

6.1.5. Unique Aggregate Label Propagation

Algorithm 5 Unique aggregate label propagation

```
1: xLabels
= 2
− width matrix containing each known label and its index in Y
2: uLabels = unique labels in xLabels
3: for each x in uLabels
4:      tLabel = label in xLabels at position x
5:      temp = indexes of Y with label tLabel
6:     temp2 = feature vectors in X with indexes in temp
7:     featuresAvg = mean of features in temp2
8:     for each z in X
9:          distances(z)
= pairwise distance between featuresAvg and X(z)
10:     end
11:     counter = 0
12:     while(counter ≤ M)
13:          smallest(counter)
                    = min(distances) which is not index x
14:     end
15:     Y(smallest) = tLabel
16: end
```

Algorithm 5, which we refer to as Unique Aggregate Label Propagation, firstly finds the number of unique labels within the set of labelled feature vectors. For each of those unique labels, it aggregates all the associated feature vectors into sets containing the same label. For example, all feature vectors containing the "walking" label will be grouped together. The average feature vectors for each set is computed resulting in an average feature vector for each label.

Like Single Label Propagation, these average feature vectors can be compared with the unlabelled feature vectors to find the closest unlabelled feature vectors to the averaged labelled feature vectors. A parameter "M" is introduced to control the number of similar feature vectors to copy labels onto.

An example iteration of this algorithm for the "walking" label would be as follows.

1. Find all labelled feature vectors containing the "walking" label.
2. Compute average feature vectors of "walking" labels, providing us with a single feature vector to represent all walking feature vectors.
3. Search the unlabelled feature vectors for the M closest feature vectors and copy the walking label to these.

Unlike Single Label Propagation, higher numbers of "M" do not cause an exponentially higher compute requirement so it maybe more applicable to certain applications, such as mobile or battery-operated devices.

6.1.6. Experience Sampling

Experience sampling within this sub-chapter will simulate the collection of a single label for each bag. The label will be collected directly on the experience sampling interval time, meaning that there will be a high likelihood that the label provided will be accurate.

This method has been chosen over the other methods mentioned in Chapter 3.1.2.1, as in order to prove the efficacy of this system and to prove the hypothesis that labels can be populated, our original labelled feature vectors must be accurate.

6.1.7. Validation

Validation is performed within this experiment using a train/test split, as the dataset contains a total of 30 participants, 20 of them have been placed within the training set and 10 within the test set. The participant data within the training set is completely independent from the test set to reduce bias.

6.1.8. Classification

As Support Vector Machines have been shown to perform well on activity data, an error-correcting output codes support vector machine (ECOC SVM) will be used to model the weakly labelled data and perform classification. An ECOC SVM allows for multi-class classification, which is particularly advantageous for activity data due to multiple different activity classes possible. ECOC SVM utilizes an ensemble of binary one vs one classifiers [10]. While the error-correcting codes output by the ECOC SVM results in extra data overhead, this work proposes techniques which can leverage these codes to enable the system to recover from errors caused by weak labels such as poor input features or a flawed training data.

6.2. Results

The following experiments will first evaluate the performance of standard supervised classification with full ground truth. The labelled data will then be restricted via experience sampling and finally, new labelled data will be introduced through weak supervision methods.

6.2.1. Supervised

An ECOC SVM model was trained using pre-computed feature vectors and fully supervised activity labels. The results shown in Table 12 provide a baseline for weakly supervised performance to be compared against, with an overall average accuracy of 96.4%.

Table 12 - Supervised Per Activity Performance

Activity	Accuracy
Walking	99.4%
Walking Upstairs	96.8%
Walking Downstairs	96.4%

Sitting	89.2%
Standing	96.9%
Laying	99.8%

6.3. Weakly Supervised

6.3.1. Experience Sampling Only

Table 13 shows the number of labels found through experience sampling for each length of the sampling window. Even a one-minute interval significantly reduces the number of labels available in the training set, any feature vector which has not been assigned a label is discarded as it is not usable by a supervised classifier. Classification performance results of the SVM models trained on only experience sampled feature vectors and labels will be detailed in a later section alongside classification performance results of Single Label Propagation and Unique Aggregate Label Propagation.

Table 13 - Labels Generated By Experience Sampling

Sampling window (min)	0	1	3	5	10	15	20	30	60
Number of labels	7415	322	107	64	32	21	16	10	5

6.3.2. K and M values

As both proposed algorithms use parameters which control the number of outputted labelled data points, experiments will be performed to find the most appropriate values for both. These experiments will be performed on the 1, 10- and 20-minute experience sampling windows. These times have been selected as they provide data relating to both short and medium length sample windows. The longer windows are not being used due to dataset limitations.

The experiment will operate by simulating the collection of experience sampling labels in the method explained in Chapter 3.1.2.1, providing several accurately labelled feature vectors. The Single Label Propagation and Unique Aggregate Label Propagation algorithms will then be run on the small number of labelled data points gathered by experience sampling and populate them to the rest of the unlabelled data within the training set.

The K-values will be tuned first and the results are shown in Table 14.

Table 14 - K-Value Parameter Testing

1-minute sampling window			
K value	1	2	3
New labels	1274	2547	5094
Percent correct	96.8%	94.8%	91.7%
10-minute sampling window			
K value	1	2	3
New labels	128	256	512
Percent correct	97.8%	95.3%	93.4%
20-minute sampling window			
K value	1	2	3
New labels	64	128	256
Percent correct	98.4%	96.9%	95.2%

Secondly, the M-values are tuned, and the results shown in Table 15.

Table 15 - M-Value Parameter Testing

1-minute sampling window			
M value	10	20	30
New labels	378	434	491
Percent correct	98.8%	98.1%	97.1%
10-minute sampling window			
M value	10	20	30
New labels	85	144	169
Percent correct	97.0%	94.7%	90.4%
20-minute sampling window			
M value	10	20	30
New labels	69	127	186
Percent correct	96.2%	95.9%	94.8%

The bigger the values the more labels that will be generated, however, this could also result in an increased number of incorrect labels compared to the ground truth.

6.3.3. Single Label Propagation

We can see from table 3 that increasing the K-value does increase the number of new labels generated, however, the bigger the K-value used, the higher chance that incorrect labels will be created. A K-value of three will be used for the remainder of the experiment. This is due to it having significantly more labels than the other K-values, while only having a slight reduction in quality. K-values of above 3 have been tested but they run into the issue of diminishing returns. They take an extremely long time to run and while they produce more labels for classification, they have a higher number of noisy labels.

6.3.4. Unique Aggregate Label Propagation

Based on the results in table 4, an M value of 20 is selected for algorithm 5. The reduction in label quality between the values of 10 and 20 is insignificant compared to between 20 and 30. The value of 20 also provides substantially more labels than 10, which will be useful to the classifier.

6.3.5. Classifier Results

Fig. 19 shows the overall classification accuracy for each experience sampling window length, these have been achieved by training an SVM on the labels generated both by experience sampling and the algorithms mentioned within this section. The graph shows that as the sampling window length increases above 5 minutes the classifier accuracy of experience sampling starts to decrease significantly whereas, the two proposed weakly supervised algorithms decrease much slower until the 30-minute mark. Both algorithms allow the experience sampling window to be increased to 20 minutes while maintaining an accuracy of above 70%. Algorithm 1 performs better overall, but not significantly. After the 20-minute window accuracy sharply decreases for both algorithm 4 and 5. This is due to the experience sampling window being so long that entire activities are being missed.

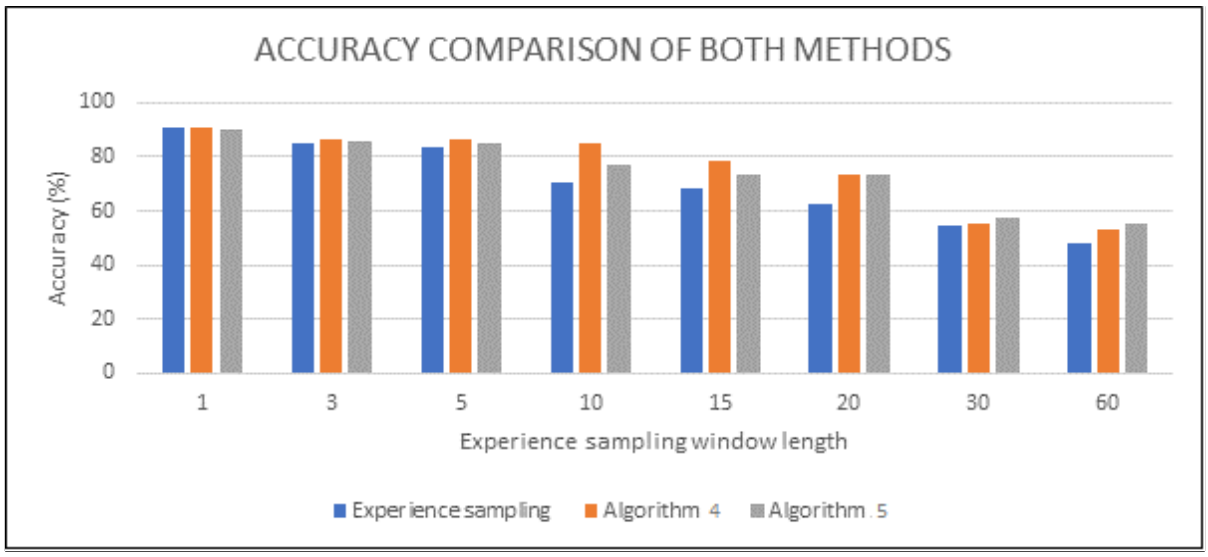

Figure 19 - Accuracy Comparison of Propagation Algorithms

Table 16 shows the classifier accuracy for individual activities in three different sampling window lengths. The one-minute sampling length provides the closest comparison to fully supervised, while lengths greater than 20 minutes run into the limitations of the dataset. For each of these window lengths, we can see that certain activities appear to be affected differently based on the length of the sampling window. Walking, standing and laying do not appear to be significantly impacted even with a sampling window length of 20 minutes. However, walking upstairs, walking downstairs and sitting are all reduced. Walking upstairs is affected the most with a significant reduction from 1 to 20-minute sampling length. It could be that certain activities need more data for accurate classification, or the dataset could be unbalanced.

Table 16 - Per Activity Performance Comparison of Propagation Algorithms

1-minute sampling window						
Activity	**Walking**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**
Experience sampling	98.2%	91.5%	84.2%	81.9%	87.1%	99.8%
Algorithm 4	95.8%	94.4%	88.2%	79.9%	88.2%	99.8%
Algorithm 5	97.9%	90.4%	84.8%	79.0%	87.4%	99.8%
10-minute sampling window						
Activity	**Walking**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**
Experience sampling	91.6%	52.8%	55.4%	19.3%	95.0%	96.6%
Algorithm 4	94.4%	70.2%	70.8%	30.2%	97.2%	95.6%
Algorithm 5	87.4%	58.9%	76.3%	46.5%	91.6%	97.2%
20-minute sampling window						
Activity	**Walking**	**Walking Upstairs**	**Walking Downstairs**	**Sitting**	**Standing**	**Laying**

Experience sampling	95.5%	5.3%	41.9%	44.1%	84.9%	94.9%
Algorithm 4	93.3%	38.9%	71.1%	55.8%	83.3%	94.9%
Algorithm 5	92.8%	18.8%	47.3%	75.3%	68.4%	88.8%

Overall, the two proposed algorithms appear to be effective at increasing the classifier accuracy of the worst-performing activities. With algorithm 4 increasing the accuracy of walking upstairs and walking downstairs by 33.6% and 29.2% respectively at the 20-minute experience sampling window.

6.3.6. Performance results

Although algorithm 5 has not achieved the classification results that algorithm 1#4 did, it is significantly faster as shown in figure 20. Although in recent years smart devices have greatly improved in terms of battery life and compute performance [143], they would be the ideal platform for an activity recognition system and anything which reduces the impact of these systems would be advantageous.

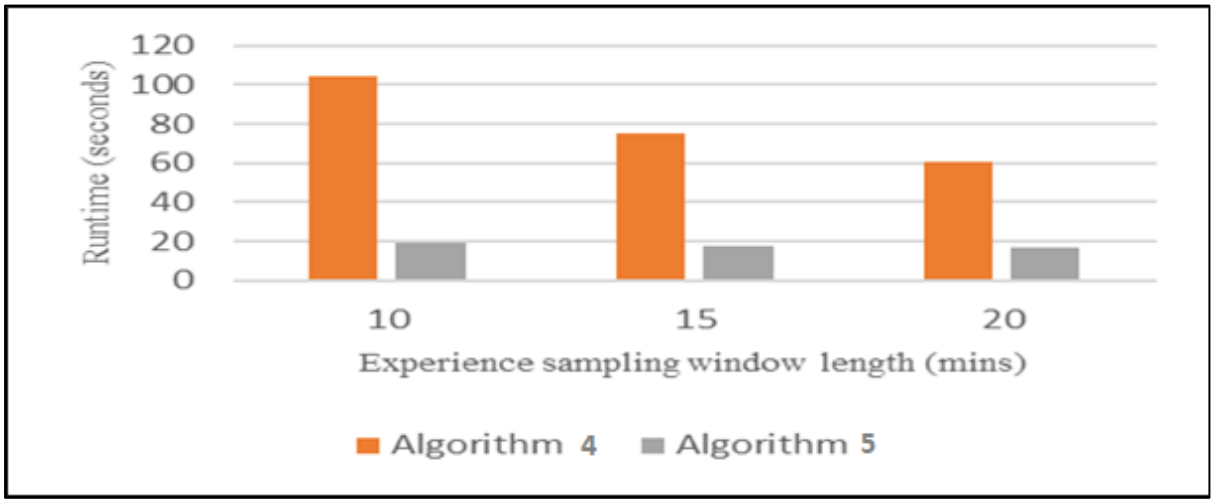

Figure 20 - Compute Performance of Propagation Algorithms

6.4. Conclusion

Within this chapter the challenge was:

> *Can we use similarity metrics, such as the distance Euclidean distance between feature vectors to infer similar labels? If performed successfully, this could allow significant increases to the number of labelled feature vectors and classifier performance.*

This section introduced two different techniques which could be used to propagate labels from labelled feature vectors to unlabelled feature vectors. They allow the tested dataset sampling windows to be effectively doubled, with classifier accuracy of 74% even with a 20-minute sampling window. However, above this window length generally results in problems as entire activities are missed and the limitations of the dataset are pushed. The problem is shown in both the experience sampling labels and the weak supervision labels.

7. Reducing the requirement for expert feature analysis

In previous chapters, the proposed techniques focused on novel techniques to propagate weak labels from infrequent user-sampled requests to unlabelled feature vectors. These techniques used the principle that a person will be providing occasional labels to a self-contained system, which would use the proposed algorithms and the gathered weak labels to create supervision signals which could be used to train a classifier capable of recognizing these activities. However, the step of feature extraction has been overlooked thus far and has been performed using standard non-automated methods, such as manual experimentation with statistical and frequency-based features conducted by a domain expert. Recent developments into feature extraction have produced the TSFRESH algorithm [144], which is capable of automatically generating features from a time series, however, these types of brute-force feature extraction systems can be computationally expensive [145] which may not be suitable for smart phone applications. In addition, with deep learning being such a pervasive topic currently across the machine learning domain and with its capabilities in providing a completely feature free approach, we think it appropriate to test its capabilities in both classification of weak labels and its performance in removing features.

Extraction of relevant features is an extremely important step to classifying activities, poor extraction of features will almost certainly result in poor recognition performance. This knowledge can be used to generate our research question for this chapter.

“How can we generate the best possible features for discrimination of activities without breaking our criteria of a self-contained system?”

This chapter will therefore deal with automated generation of features for activity data. Within literature, several examples exist of automated feature generation, however, much of this work has focused on image processing. The current state of the art in automated feature generation is using convolutional neural networks, which are a class of deep-learning. A convolutional neural network is a type of neural network which contains convolutional layers which perform transformations on the inputted data. When viewed from a computer’s perspective, an image is series of series of numbers and has similarities to a signal produced by an accelerometer, and previous literature has proven it capable of generating features from signals other than images. For example, a CNN was used in conjunction with transfer learning to recognize activities sensed with inertial measurement units [146].

To date, no work has evaluated deep learning-based features in a weakly supervised activity recognition system. This chapter will therefore focus on furthering our understanding of deep learning-based features for weakly supervised activity recognition.

In order to evaluate if deep learning features can be used to classify activities when combined with weak labels, this chapter will focus on answering the following research questions:

1. How well does deep learning perform on statistical features? If it is unable to classify activities with performance like other classifiers from other chapters, such as SVM, then it is highly unlikely that it will perform well on raw accelerometer data.
2. How well do deep learning features work when classified? If the performance is initially low in comparison to statistical features then it is likely that providing even weaker information, such as weak labels is going to reduce performance to unreasonably low levels.
3. If the performance of both classification and feature extraction is comparable to typical methods, then how well will deeply learned features perform when provided with weaker labelling information.

This chapter will aim to provide an answer to these questions. The methodology will first detail the data used, and describe techniques relating to how feature extraction will be performed for statistical feature sections. Methods of gathering statistical features for the comparison tests will then be described and finally, the methods used to classify and generate features from deep learning will be explained. The results section will detail the results of the different experiments, and finally, discussion and conclusion will be performed.

7.1. Methodology

This methodology will introduce the dataset to be used and how deep learning can be used to generate features from raw signal data.

7.1.1. Dataset

As previously described in Chapter 3, the Human Activities and Postural Transitions (HAPT) dataset will be used in this chapter to evaluate different deep learning models. This dataset contains 6 activities, walking, standing, sitting, walking upstairs, walking downstairs and laying. In addition to the 6 activities, the dataset also contains 6 postural transitions. However, as explained in earlier chapters of this thesis, the postural transitions will not be assessed in this experiment. The six activities to be classified within these experiments will therefore be walking, standing, sitting, walking upstairs, walking downstairs and laying.

The dataset contains both accelerometer and gyroscope data and is recorded for 30 participants.

7.1.2. Experience sampling

As previously described in Chapter 3, experience sampling gives the system the ability to request labels from the user at intermittent times. However, in the case of these experiments, it will be simulated.

In a live deployment, the system would ask the user what activity they have been performing the most over a given time frame. However, as we are using data from a pre-recorded public dataset, experience sampling is simulated by contacting the ground truth and retrieving the label which occurred the most within that time frame.

7.1.3. Clustering for Multiple-Instance based collection of labels

After an experience sampling request has been made and gathered a label is collected, we are present with the same problem discussed in previous chapter where the weak label encompasses many feature vectors, and not all feature vectors will apply to this label. To circumvent this, a method of removing labels which are unlikely to be correct is performed. We use our original user-request assumption, where the user is asked what activity they have performed the most in each time frame. To find this most significant activity we can use clustering and search for the largest cluster.

Using clustering creates an additional problem, most algorithms require the number of clusters to be specified, something which is unknown when performing this approach. There are two possible algorithms for clustering without knowing the number of clusters. They are Gaussian Means (G-Means) and Density-based spatial clustering of application with noise (DBSCAN). In chapter one, two weakly supervised algorithms, utilizing these clustering methods, were proposed. Of these, it was shown in chapter one that the G-Means based weakly supervised algorithm outperforms the DBSCAN based algorithm in this application and will be used for the experiments in this chapter.

7.1.3.1. Gaussian Means (G-Means)

As described in an earlier chapter, the K-Means clustering algorithm uses a value ‘K’ to find the ‘K’ clusters in a dataset, while the G-Means algorithm uses no input variable in instead clusters the data with several different values of K. The cluster which closest fits a Gaussian distribution is then taken as correct.

After execution of the G-Means algorithm, the largest cluster returned will be found and the experience sampling label applied to it. G-Means has a single parameter, known as the strictness value, this value specifies how well each of the clusters must fit the Gaussian distribution.

7.1.4. Deep Learning

A neural network is a collection of “nodes” which are connected. An example of a neural network can be seen in Figure 21. The leftmost nodes, represented in yellow, are the input nodes which take the features, or raw data, depending on the configuration of the rest of the network. The middle section represented by the grey dots are the hidden nodes. The hidden layer nodes contain transformed input

values which will eventually allow translation into the final node. The output layer is the final output of the network which can represent a percentage likelihood vote for each class.

These translations between the network layers are performed by activation functions, many of examples of these exist including softmax and ReLu. These activation functions create the nodes output and induce non-linearity into the network. On the output layer, the activation function will activate the node of the predicted class. This final activation is usually performed by the softmax function.

The lines between the nodes can be thought to contain weights, and bias is inserted into the nodes at each layer, with the exception of the input layer. In order to successfully map inputted data to the correct class, these weights and bias must be tuned. This tuning process usually involves backpropagation, which generates a gradient against the selected loss function. The loss function is a way of quantifying how poorly fitted the network is in mapping input to the correct output node. Finally, an optimization technique is applied which will update the weights and bias until they converge on a global minimum loss. Ideally at the end of this process, the network should be able to successfully link inputted data to the correct class.

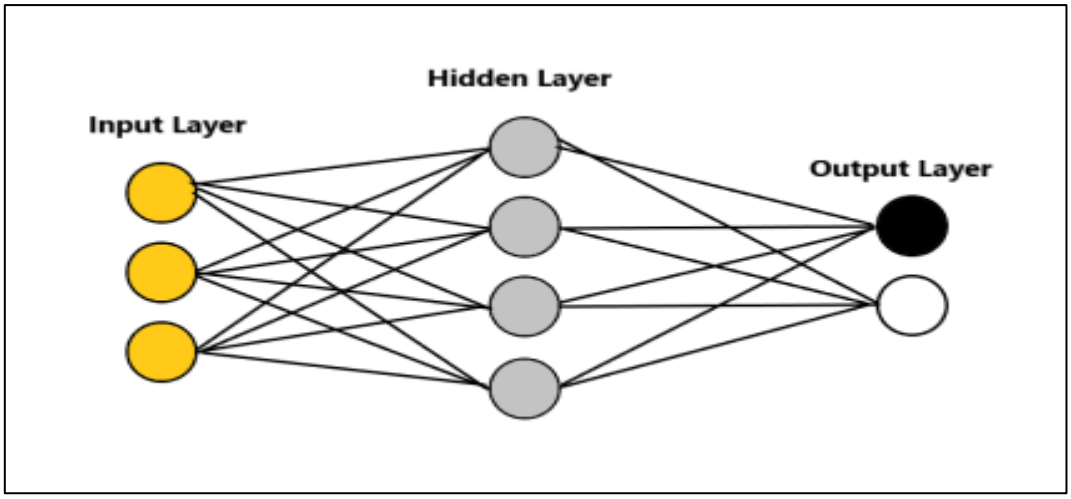

Figure 21 - Neural Network

Deep learning is a machine learning term which means a neural network that has multiple hidden layers. These additional layers allow the learner to extract additional features from raw data. Of the deep learning classes, this chapter will specifically target the convolutional neural network. These convolutional neural networks have been shown to be capable of generating useful features from raw data, they do this by applying filters to the data to find hidden patterns. These generated features are

known as convolved features. Like normal neural networks, these convolutional layers also contain an activation function, weights and bias.

An unfortunate problem is that many deep learning configurations do not have any definitive solution. So, the number and type of internal layers will vary depending on the solution required, and the only way to find the best configuration is through experimentation.

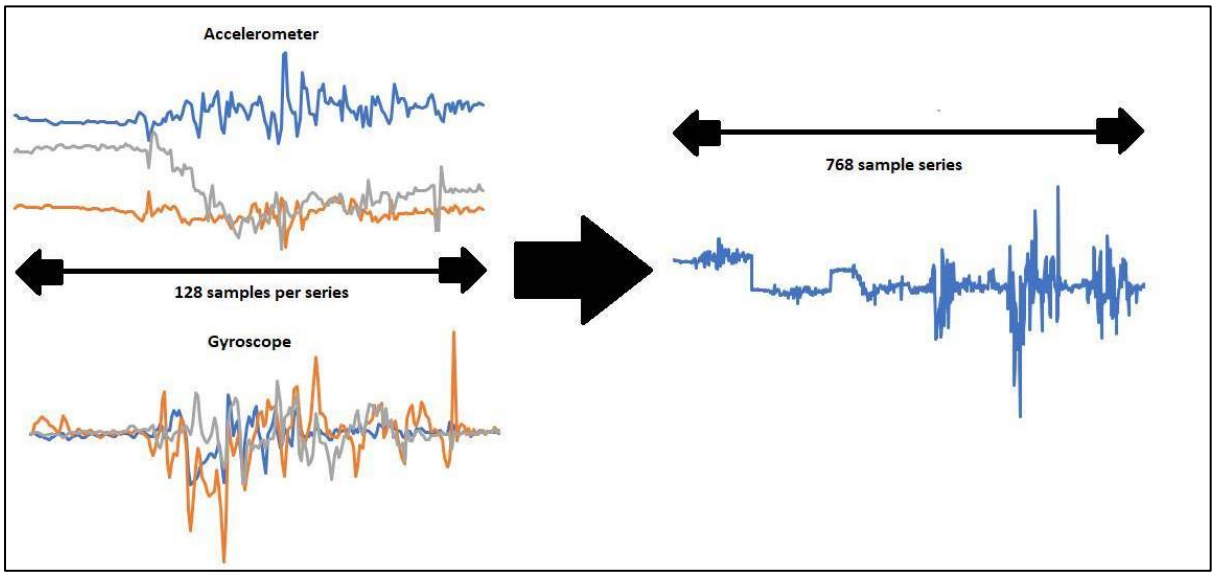

Figure 22 - Converting 6-axis to single axis

Figure 22 shows how the raw data from the 3-accelerometer axis and the 3-gyroscope axis is converted into a single axis. They are simply put on top of one another, changing it from a 128x6 matrix to 768x1 vector.

In the case of the experimentation in this chapter, the algorithms are executed within the Ubuntu environment. The editor used is Spyder and to create the deep learning networks, the Keras API is used with a Tensorflow backend.

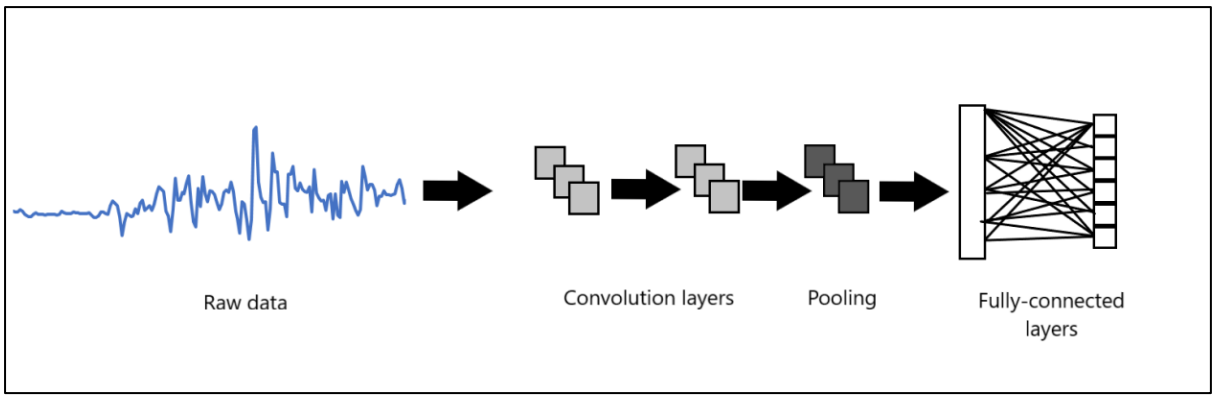

Figure 23 - Raw Data to CNN

An overview of the deep learning network configuration used is shown in Figure 23. A detailed specification of the interior layers is detailed in Tables 17 and 18. For consistency throughout the experiments, the two networks will be the same, except for the input layer size.

Prior to the raw data being provided to the input layer, it is broken down into bags so that experience sampling can be applied. The experience sampling bag-length for all the experiments in this section is 30 seconds, which encompasses 128-samples of raw accelerometer data. 30 seconds was chosen based on experiments performed in previous chapters while also allowing direct comparison of an input layer using raw signals compared with an input layer using the pre-computed feature vectors. When combined across the 3-axis of the gyroscope and 3-axis of the accelerometer this creates a 768-width input vector. The raw accelerometer data is then converted into a convolutional format and inserted into the first convolutional layer. This convolutional formatting creates a Keras tensor. A tensor is a data storage container, like vectors and matrices, however, with a vector storing data in 1-dimension and a matrix in 2-dimensions, a tensor is capable of storing data in N-dimensions.

Once data has been inputted into the first layers, it will be moved through the convolution and pooling layer which have been designed to act as the feature extraction stage in this experiment. Convolution layers apply a sliding filter, in this case of size 3x3 in both of our convolutional layers, which will transform the data into the given output size. The convolution layers have a rectified linear unit activation function, known as “ReLu”. Activation functions are a mathematical function which provides the output of the neuron. ReLu is efficient, however, zero or negative inputs return zero, which will cause problems with backpropagation.

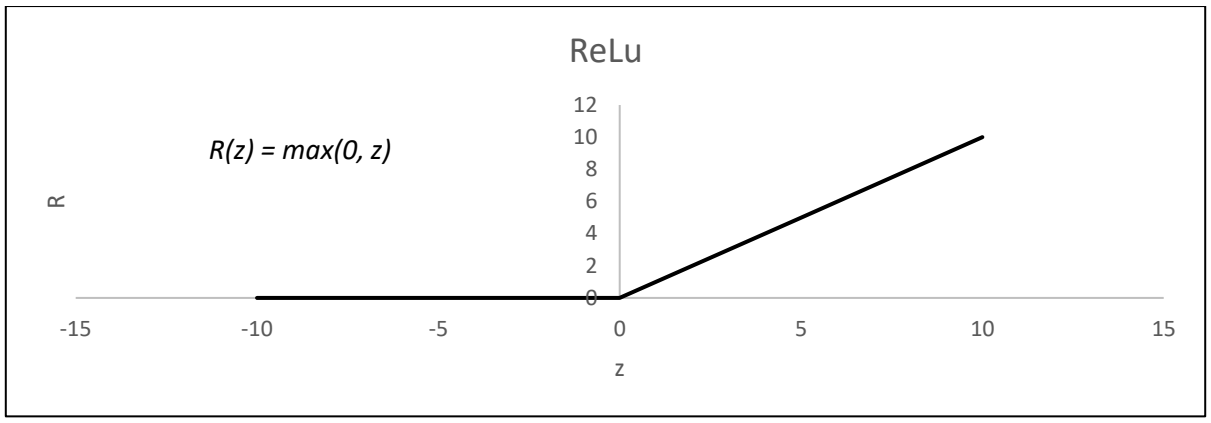

Figure 24 - ReLu Graph & Equation

After the convolution layers, the data is passed to a max-pooling layer of size 3. This will take the max values within 3x3 squares of the data, as shown in Figure 25.

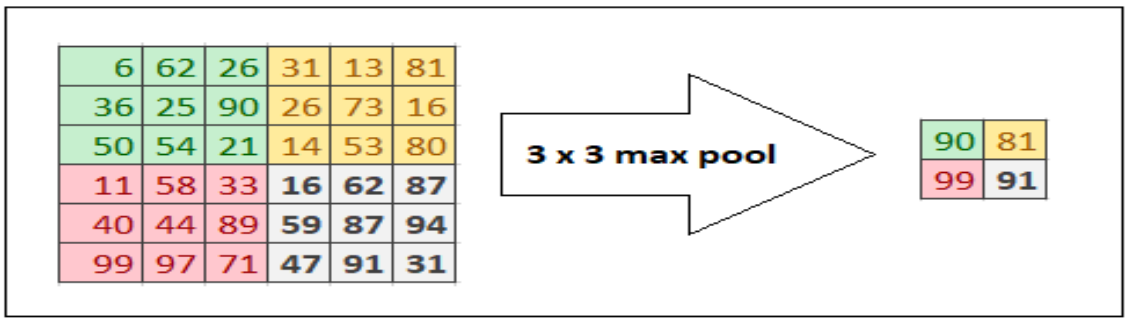

Figure 25 - Max Pooling Layer

Prior to outputs of the convolution layers being passed into a fully-connected layer, the data is put through a flattening layer in order to convert the data from a 2-dimensional format to a 2-dimensional format, an example is shown in Figure 26.

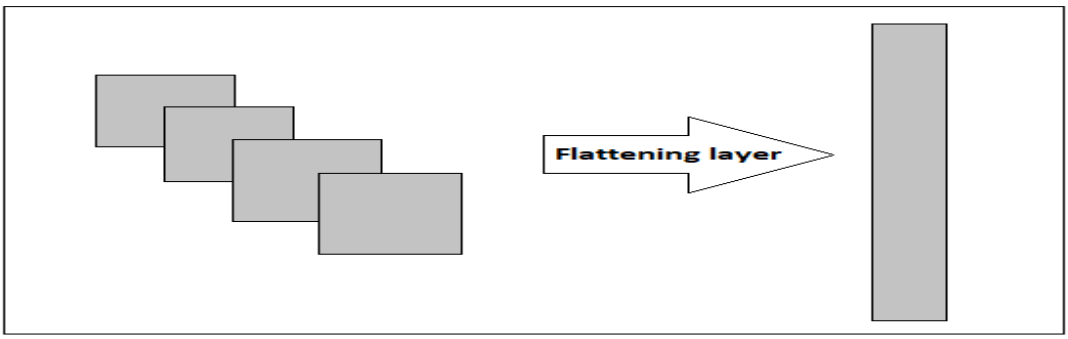

Figure 26 - Flattening Layer

Outputs of the flattening layer are then used as inputs to the fully-connected layer. The first fully connected layer is activated using ReLu, the same activation function used in the convolutional layers. The final layer has 6 nodes and is designed to perform the final classification of the activity data, with each of the nodes representing one of the activities. The final layer uses a softmax activation function, shown in Figure 27. The softmax function ensures that all neurons on the final layer add up to one, as a result of this, each of the outputs of the final layer becomes a percentage probability score towards that class.

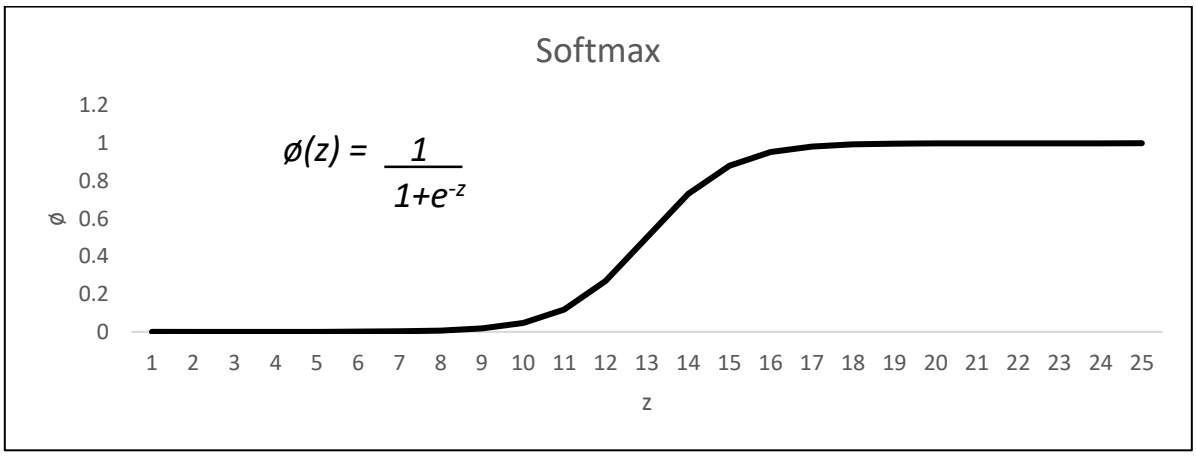

Figure 27 - Softmax Layer

Neural networks are trained using two main parameters, the number of epochs and the batch size. The batch size refers to the amount of training samples processed each time the model is updated and the epochs refers to the total number of passes that will be made through the complete training set. The network is trained using 50 epochs and a batch size of 50 for all experiments. The optimizer used for all experiments is the "adam" optimizer. An alternative optimizer would be stochastic gradient descent, however "adam" optimizer is selected as one of the drawbacks of stochastic gradient descent is a static learning rate which can impede convergence time, while "adam" uses a dynamic learning rate which speeds up convergence.

Table 17 shows the final configuration of the convolutional neural network for making predictions on the statistical features. The input layer size was based on the size of the data to be inputted and the final layer based on our six activities.

Table 17 - CNN Layers – Statistical features

Layer Number	Layer Type	Size	Kernel size	Activation function
1	Input layer	561	N/A	

2	1-dimensional convolutional	2	3	ReLu
3	1-dimensional convolutional	10	3	ReLu
4	Max pooling layer	3	N/A	N/A
5	Fully connected layer	350	N/A	ReLu
6	Fully connected layer	6	N/A	Softmax

As our deep features will be of different input size, Table 18 shows the configuration of the deep network when it is generating deep features.

Table 18 - CNN Layers - Deep Features

Layer Number	Layer Type	Size	Kernel size	Activation function
1	Input layer	768	N/A	
2	1-dimensional convolutional	2	3	ReLu
3	1-dimensional convolutional	10	3	ReLu
4	Max pooling layer	3	N/A	N/A
5	Fully connected layer	350	N/A	ReLu
6	Fully connected layer	6	N/A	Softmax

7.1.5. Experimental Protocol

Several different experiments will be performed to answer the research aims highlighted earlier in this chapter.

7.1.5.1. Evaluation Protocol

All experiments within this chapter will be performed using cross-validation as the method of evaluation. The cross-validation will use 5-folds, and the main performance metric used will be accuracy. The dataset contains 30 participants, so each of the CV folds should contain 6 participants in the test set and 24 in the training set. As the dataset contains six, well balances activities, including walking, standing, sitting, walking upstairs, walking downstairs and laying, accuracy will be chosen for the main classification performance metric.

7.1.5.2. Experiment 1 – Effects of deep features on fully supervised data

This experiment is designed evaluate if a convolutional neural network is capable of classifying human activity to the same standards as the classifiers previously used within this thesis, such as SVM. In order to establish this baseline for comparison of performance, traditional feature vectors are extracted using the same format as those used in previous chapters. Specifically, features are pre-computed from raw 3-axis accelerometer and 3-axis gyroscope signals. Features are comprised mainly of time-domain features. Prior to feature extraction, the raw signals are cleaned by applying a median filter and a low-pass Butterworth filter. An additional Butterworth filter is also used to extract the gravity signals. This will lead to five different time series being produced from the raw data.

The filtered signals are then segregated into 128-sample length windows. This equates to 2.6 seconds of raw data. Features are then derived from these windows, including standard deviation, mean, maximum values, minimum values, signal magnitude area, median absolute deviation, signal energy, IQR, entropy, mean frequency, kurtosis, skewness and the auto-regression co-efficient. Once these features have been extracted from the signal it creates a 561-length feature vector.

These features will be classified with a CNN, the configuration of this CNN is discussed in Section 5.1.4.

7.1.5.3. Experiment 2 – Full Supervision vs Weak Supervision with domain expert generated features

This experiment is designed to evaluate the effect weak supervision has on deep learning of human activity. The methodology from Experiment 1 will be recreated with the one key exception that labels will be applied to feature vectors using the weak supervision algorithms proposed in previous chapters.

The G-means based algorithm is used to process labels collected through experience sampling. The configuration of the G-Means based algorithm is described in detail in Chapter 3. The weak supervision algorithm will be provided with the same 5-folds of training data that the CNN was provided from Experiment 1. However, the weak supervision algorithm will only be provided with experience sampling generated labels and using this small number of labels it will aim to populate labels to the otherwise unlabelled feature vectors present within each fold. This will create a supervised problem which a traditional supervised SVM can be trained on and tested against each fold of the test data.

7.1.5.4. Experiment 3 – Weak supervision with deep learning based features

While Experiment 1 and 2 have been designed to establish baseline performance, experiment 3 is the key experiment where we evaluate the proposed methodology of using deep learning to model the relationship between weak labels and raw accelerometer and gyroscope signal. We aim to understand the feasibility of removing the need for domain experts in design feature vectors in conjunction with allowing users to provide weak labels to model their activity patterns.

In order to allow direct comparison of results, this experiment will use the same methodology as Experiment 2, with the key exception of using raw accelerometer and gyroscope data instead of predesigned and precomputed feature vectors. Raw data contains signals for a tri-axial accelerometer and tri-axial gyroscope. Raw data is then windowed to be 128 samples width, roughly equating to 3 seconds of data. These windows are then bagged with 10 windows per bag, meaning each bag is now approximately 30 seconds long.

Experience sampling will now be simulated for each of these bags, with a single label being collected per bag of raw data. Our G-Means based algorithm is then applied, using the raw data to try and find the largest cluster within the data. Once these largest clusters are found the experience sampling labels are applied to the windows of raw data. Any of the windows which do not have a label applied are dropped as they are not helpful to this current configuration of a CNN.

Finally, the raw data is passed into the CNN along with the generated labels for training and the trained CNN will be tested on the test data and the labels reported.

7.2. Results

7.2.1. Experiment 1: Effect of deep features on fully supervised data

As previously described, this experiment compare the performance of a deep learning model when trained on a fully labelled dataset with regular, statistical features compared to a deep learning model trained on raw signals with features which have automatically been generated by deep learning.

Firstly, we need to evaluate how deep learning performs on a traditional fully supervised activity data problem. In order to assess performance, and to establish a context of performance of methodologies in previous chapter, we will compare it with a more typical classifiers used for activity recognition, such as an SVM. To test this, we will provide the deep learning classifier with the fully labelled dataset and associated pre-computed statistical features.

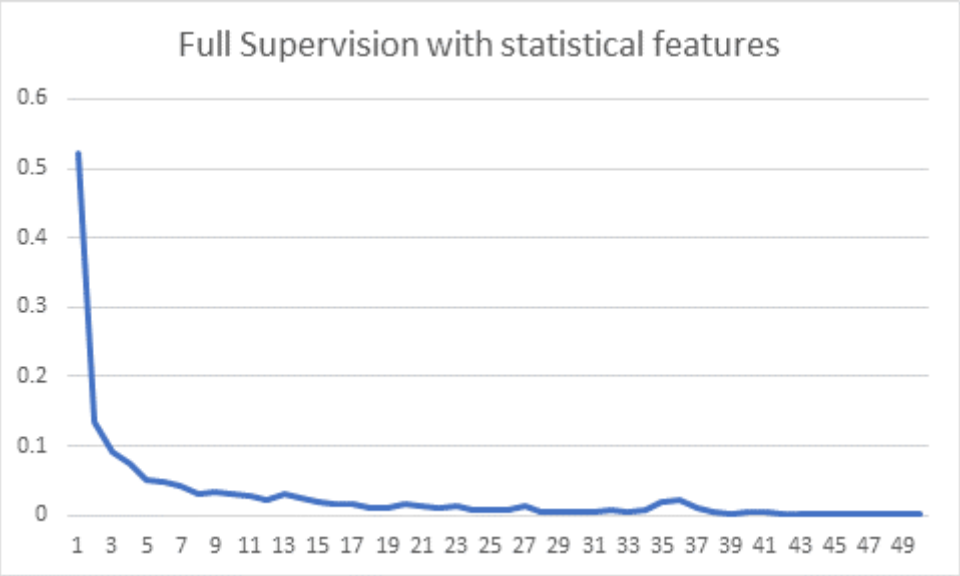

Figure 28 - Supervised loss with statistical features

Figure 28 shows an averaged loss over the 5 folds of the cross validation. It shows that most of the learning is made within the first 10 epochs and the remaining are marking limited change to the network.

Table 19 - SVM Vs Deep Learning Fully Labelled Performance

SVM		**Deep Learning**
Accuracy	**Split**	**Accuracy**
93.6%	1	93.6%
88.6%	2	93.3%
92.5%	3	96.0%
96.6%	4	96.3%
95.2%	5	95.7%
93.3%	**Average**	**94.98%**

Result are shown in Table 19, where the average accuracy achieved by deep learning was 94.98%, while the SVM achieved 93.3%. Using ANOVA, we find that although the deep-learning has performed

slightly better by average, the *p*-value is 0.3 which is not considered a significant increase in performance over SVM. Therefore, results indicate that deep learning can match the performance of other, typical, classifiers. Thus, we now test its ability to classify data without any pre-computed statistical features.

Table 20 - Deep Learning Features with Full Labels

Split	Accuracy
1	88.9%
2	84.7%
3	83.6%
4	89.1%
5	89.5%
Average	**87.16%**

Results in Table 20 shows a drop in performance when using the deep learning generated features instead of statistical features. The *p*-value when compared to the deep-learning performance from the previous experiment is 0.0005. We can therefore conclude that this drop in performance is statistically significant and is a direct result of changing the input layer from precomputed features to raw signals. Overall, the performance has dropped by an average of 7.82%, and all folds of the cross-validation have seen reductions in performance.

While this is a significant drop in classification performance, it removes a critical complexity in the activity recognition pipeline, removing the need for a domain expert between the data collection stage and model training stage, and provides a baseline to evaluate weak learning.

7.2.2. Experiment 2: Full supervision versus weak supervision with domain expert generated features

With a baseline performance established using fully supervised data for both standard statistical features classified with deep learning, and raw signals with deep learning features, they can now be directly compared with weak supervision.

For weak supervision, the G-Means based algorithm, which was proposed and evaluated in Chapter 3, will be used. The same strictness value of 4 will be used from those chapters as it was found to give the best results. In the case of this experiment and the following experiments within this chapter that use weak supervision, the 30-second bagging window will be used. The additional weak supervision methods mentioned in Chapters 4, 5 and 6 will also not be used in order to control the variables. These

additional techniques would add additional variables to the experiment, making it difficult to understand the exact effect weak supervision is having on the system.

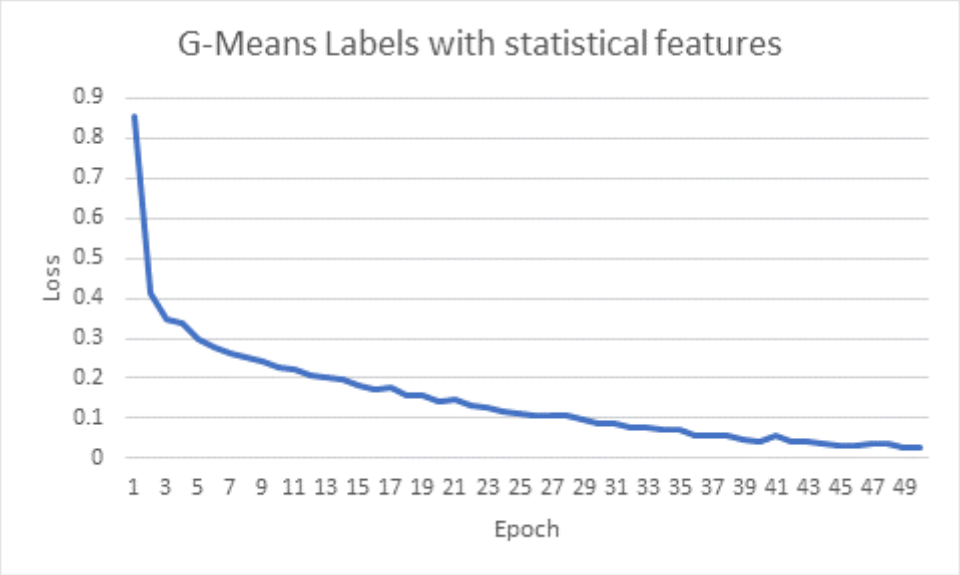

Figure 29- G-Means Loss Profile with statistical features

Figure 29 shows the learning profile of the deep learning system when given normal statistical features, the loss has incrementally reduced until the last epoch has been reached.

Table 21 - Full Supervision Vs Weak Label Performance

Full Supervision		**Generated Weak Labels**
Accuracy	**Split**	**Accuracy**
93.6%	1	89.0%
93.3%	2	85.1%
96.0%	3	95.7%
96.3%	4	86.1%
95.7%	5	92.1%
Mean: 94.98% Std: 1.42	**Average**	**Mean: 89.6% Std: 4.37**

Results of the experiment, presented in Table 21, show a reduction in performance of the weakly supervised deep learning model when compared to the fully supervised deep learning model. The reduction in performance is seen across all folds with an overall average reduction in performance of 5.38%. This is a significant reduction in performance with a *p*-value of 0.3.

7.2.3. Experiment 3: Weak supervision with deep learning-based features

Baseline metrics have been established for the performance of deep learning using (1) statistical feature vectors vs raw signals and (2) fully supervised labels vs weakly supervised labels. Thus, we now evaluate the overall performance of the proposed deep learning methodology of G-Means weak supervision system with deep learned features.

Figure 30 shows the loss over the 50 epochs when the network is being trained with deep features and weak labels. As before, the loss is a result of an average across the five folds.

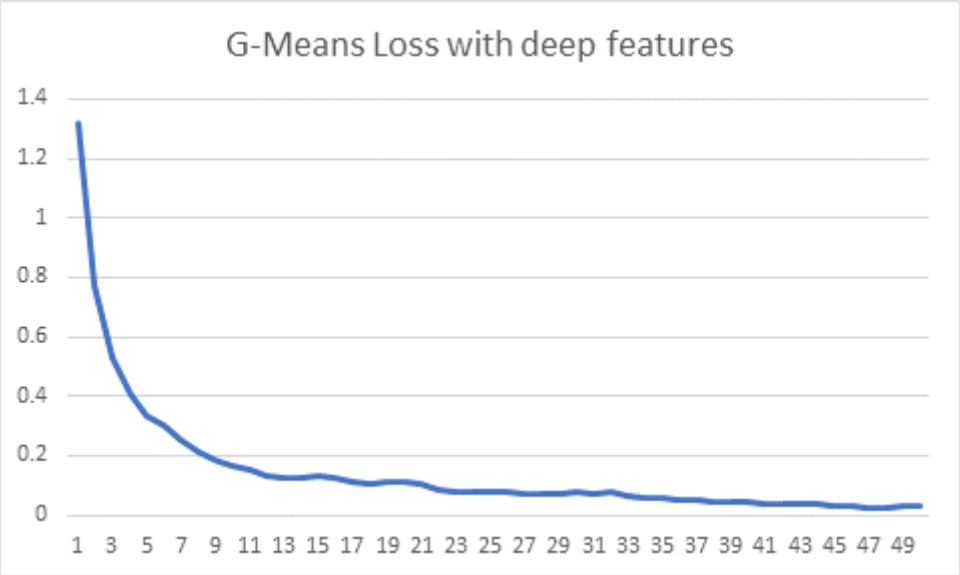

Figure 30 - G-Means loss with deep features

The methodology used for preparing raw signals for the input layer is the same as described in experiment 1. Similarly, the methodology used to process weak labels is the same as described in experiment 2, including the same strictness value of 4 for the G-means based weak supervision algorithm.

Table 22 - Weak Labels with Deep Learning Feature Performance

Split	Accuracy
1	69.1%
2	70.1%
3	72.0%
4	76.4%
5	77.2%
Average	**72.96%**

Table 22 presents classification performance results for the deep learning model. Results show an overall classification accuracy of 72.96%. The performance has dropped by 22.02% when compared to expertly derived features and fully supervised labels. However, relative to the overall reduction in burden such a system requires, the overall performance can still be deemed good.

7.2.4. Discussion

Within this chapter, several tests have been performed to evaluate the efficacy of deep learning for both feature extraction and classification for activity recognition. The first experiment was performed to ensure that deep-learning-based classification could in-fact perform with comparable results seen in other classifiers and previous chapters, such as an SVM.

The first experiment found the accuracy of the deep-learning classifier to be approximately 95% on statistical features, showing performance which is comparable with the SVM. In the second experiment, deep learning features were tested, and results showed there is a statistically significant drop in performance of 7.82% However, at approximately 87% it can still be considered acceptable.

The next steps of experiments set out to evaluate the performance of a deep-learning model trained on a dataset which had been generated using our proposed weakly supervised methods, including the proposed weak supervision algorithm which uses G-Means. The deep learning model, trained using the weakly supervised training set, achieved an accuracy of 90% using the statistical features. This is an excellent result considering that it has significantly less labelling information than the normal fully-labelled training set.

The final experiment, and the focus of this chapter, is how well the weakly-supervised training set will perform when it uses deep-learning features. From the previous experiments, we know that we have had an approximate drop of 7% accuracy when we switch to deep-learning features and approximately a 5% drop when using weak supervision. The results show a 22% drop in performance, from 95% to 73% when using deep-learning features and weak supervision. While a performance drop of 22% is significant, we still consider it an promising result considering how little data the algorithms have to use and how little human input was required to achieve the result.

There is, however, still improvements needed in order keep human input to a minimum while also achieving results comparable to fully human supervised classifiers modelled on expert human engineered features. There are two main speculative reasons why this drop in performance happens. The first is that weak supervision gives the opportunity for a system to be provided with a higher ratio of incorrect to correct labels. More incorrect labels will negatively affect the classifier learning process

and may cause the deep-learning features to be less accurate. The second reason is that there are simply fewer overall data points to learn from and therefore less chance for the system to generate good features for discrimination. Drops in classification performance are acceptable for most activity recognition purposes.

7.3. Conclusion

At the beginning of this chapter, some ideas were laid out for what we could expect when classifying a weak training set with automatically generated deep features. Having the ability to automatically derive features would provide substantial benefit over the current approach as it removes the requirement to have an expert pick features for certain activities. A system of this style may be able to add new activities, and potentially learn increasingly better features to discriminate activities as more data is added.

> *Firstly, how well does deep learning perform on statistical features? If it is unable to classify activities with performance like other classifiers from other chapters, such as SVM, then there is no point in investigating further.*

It was found during the experimentation that the features generated by the deep-learning algorithm perform close enough in performance to permit the substitution of statistical features with deep-learned features.

> *The second problem is how well do deep learning features work when classified? If the performance is initially low in comparison to statistical features then it is likely that providing even weaker information, such as weak labels is going to reduce performance to unreasonably low levels.*

It was found that performance did drop when the deep-learning classifier was switched to using weaker labels, however, it did not drop by very much and would still be usable.

> *Finally, if the performance of both classification and feature extraction is comparable to typical methods, then how well will deeply learned features fair when provided with weaker labelling information.*

In this final test, the deep-learning classifier is combined with weak labels, only this time it generated its own features. From this, we have seen a substantial performance drop. The performance drop exceeded what was expected but classification performance did remain at a level which could be considered reasonable.

To conclude, the features generated via deep-learning do appear to be comparable to those normally generated by an expert. However, it is possible with further tuning and development of the deep-learning architecture that performance could be increased. Unfortunately, no definite methods of tuning or designing these types of networks currently exist, and certainly none exist for the purposes shown in this chapter. When combined with weak labels, performance does suffer somewhat, however, it is still at a level which could be considered acceptable from a real-world standpoint. In these experiments, the experience sampling window was kept short in order to prove the concept. In future experimentation, it could be a starting point to test longer windows.

8. Conclusion

This chapter will conclude, summarize and evaluate the results shown in the previous chapters. The central idea of this thesis was to take advantage of the current smartphone revolution and allow activity recognition systems to move away from laboratory-based data collection experiments and bring them towards free-living environments. To achieve this goal, a literature review was performed in Chapter 2 to discover the state of the art in activity recognition and weak supervision. The literature review found that there are limitations in how the experiments are performed, these rigid laboratory-based experiments produce data which may not be indicative of the way activities are performed when participants are not being monitored. There are also concerns that collecting data for a small number of people may not provide enough data for accurate classification across the full population. The review also provided guidelines on what activities to capture, with locomotive and ADL identified as useful. Information was also provided on potential weak supervision and machine learning technologies to investigate. The findings of this review were used to create of the original research objective:

> *"To prove that within a controlled environment, a recognition system such as this could work and achieve results similar to those seen in traditional systems without the intrusive experiments."*

From this research objective, three research questions were derived:

1. *Can weak supervision techniques successfully recognize activities when given significantly weaker labelling information? And if so, are there any limitations on the type of activities which can be successfully recognized?*
2. *Are there techniques we can employ to further increase the feasibility of these experiments?*
3. *Can we detect activities without requiring a domain expert to analyze the data?*

Chapters 3, 4 and 5 of the dissertation, documents the research work carried out with the aim of solving the problems presented in the research questions. The overarching theme was to provide a method which breaks away from the traditional medical questionnaire and intrusive data-collection experiments by allowing the user to provide their own labels. These types of activity recognition systems have legitimate uses within medical and sports fields, and the ability to tune the system to match individual people's movement style and allow them to capture and annotate their own activities could both improve accuracy and usability.

For such a system to work, there would first need to be a method of capturing labels from the user in the least intrusive way. Unfortunately, when asking for user input about the activities they are performing there is always going to be some intrusion. As a result of the literature review, a method known as experience sampling, which asks the user to input information at certain time intervals was identified for deployment. This was combined with weak supervision methods which can take advantage of mixtures of unlabeled and labelled data to improve the classification performance of a classifier.

Concluding Summary of chapters

A brief summary of each of the chapters is now presented highlighting key methods deployed and results obtained.

Chapter 2 – Literature Review

The purpose of the literature review in this thesis was to find any research into both weak supervision and activity recognition and any combination of the two. Experience sampling had been shown to provide efficacy for activity recognition, and it is a method which if designed correctly can balance the intrusion of user-request systems with the need for additional data. The review also targeted different types of sensors and how the data would be used, and what meaningful features could be extracted from the data to aid classification.

Different types of activities are also investigated in this section. The first of which are locomotive activities which encompass basic movements, such as walking, standing, sitting, laying and walking up or down stairs. The other main group of activities are the activities of daily living, these activities are more complex and are used to quantify an individual's ability to live independently.

Several different methods of classification are investigated, including supervised, unsupervised and weak supervision. Weak supervision differs from the others in that there is a mixture of labelled and unlabeled data, and the weak classifier attempts to gain additional knowledge from the unlabeled data.

Chapter 3 – Experience sampling for label collection

This chapter set out to answer questions regarding the abilities of experience sampling to provide enough information for reasonable levels of activity classification. It used a simulated experience sampling method on two different datasets, one which contained locomotive activities and another which contained activities of daily living.

One major issue which is discovered is, if we gather labels from experience sampling and ambiently collect sensor data, how do we correctly translate the labels to the correct feature vectors. There are two main options, the first option being to use clustering and appropriate experience sampling requests to estimate the most probable feature vectors the labels apply to. The second option is to use the feature vector which occurred at the same time as the request. The latter has a higher chance of getting a correct label but limits you to only a small number of labels. However, the first option has a higher chance of getting more overall labels correct at the cost of accuracy.

Both methodologies are tested, and it is found that a clustering methodology, known as G-Means, was able to take the weak labels gathered from experience sampling and apply them to relevant feature vectors with higher levels of accuracy. This provided a substantial reduction in overall labels, compared to a fully labelled dataset with only minor reductions in the classification performance.

The next challenge was to find out what activities could be recognized using this methodology. It was found that the G-Means system was capable of making classification at near the standards of a fully supervised training set. While we do see reductions in performance, this is a trade-off when considering a system which can be utilized in real-world conditions.

The final challenge was to find how weak the labels could be made before we ran into serious classification performance problems. In these experiments, the optimal windows were seen at 1 to 1.5 minutes, which is quite short and will require future work.

Chapter 4 -Reducing the overall number of user-requests

The challenge here was to reduce the intrusion of a system which used user-requests. Any system, activity recognition, or not, runs the risk of being rejected by the user if excessive requests are made. So, any method which can be employed to reduce the number of these requests and label automatically would be beneficial. In order to do this, a system which was able to make predictions on the activities the user is currently performing is required, however, a problem is that how do we know if this prediction is correct? What about the cases when an activity the system doesn't recognize is being performed?

To solve this problem, the chapter used classifier confidence in its prediction as an indicator of the chances that the prediction would be correct. While several classifiers could have been chosen, e.g. SVM, the chapter uses Random Forest as it provided the best results. A Random Forest works by building a series of Decision Trees and each of these trees is provided with a sub-set of the original training set. When presented with new data each of the trees will make a prediction and the Mode of these predictions becomes the output prediction of the Random Forest. By looking at the votes made

we can get an idea of the confidence the forest has in its output prediction, for example, for a new data point, if the each of the trees had made predictions of several different classes then we could say that the confidence was low, but if for a new data point the trees mostly picked the same class then we could infer that the confidence was high.

Using this principle, we can decide if a prediction should be accepted and the user-request skipped or if the confidence is low then we can push a request to the user. However, in order for such a system to work, we need some initial training data, and this was achieved by having a short training phase before the system began to make predictions.

Two different methods, which use the same principles are put forward in this sub-chapter and both of which showed that they can reduce the number of requests when compared to a normal time-based user-requesting system. However, the main drawback to this style is they are online systems, meaning the classifier will continuously be in the process of re-training with any new data that's received, this can have significant performance implications and it something which would require additional work to ensure that it operates as efficiently as possible if put into operation on a smart device.

Chapter 5 – Detecting the transitions between activities

The challenge here was detecting the transitions that occurred between activities, for example, detecting the period when an individual transition from the sitting activity to the walking activity. This can be useful as there may be certain medical significance to these transitions, however, in the case of this thesis the transitions could allow the segregation of moments of activity.

The activity transition was achieved in a manner like how the user-request reduction algorithm from the previous chapter 4. It used a Random Forest to again create an array of Decision Trees and the votes of these trees are analyzed. The way this algorithm differed, however, is that it looked at how many classes are voted for. If the range of classes outputted from the Decision Trees was equal to that of the number of classes which existed in the training set, then it is seen as extremely low confidence and therefore is either an unknown class or a transitional phase between activities.

When tested on the HAPT dataset, it achieved excellent results in detecting the transitions between these classes, managing to detect 84% of transitions without being provided with any transitional labels.

Chapter 6 - Label propagation for increasing classifier performance

This chapter was based around attempting to improve the performance of a weakly labelled training set by propagating labels from labelled data points to unlabeled data points. The idea is that increasing the number of labelled data points should improve classifier performance.

Two algorithms are presented within this chapter, each of which uses distance metrics to measure the similarity between data points. While distance measures, such as pairwise Euclidean distance which are used in these algorithms can give a high likelihood that two close points are of the same class, they have the drawback that if the data points are extremely similar then it may not actually provide a classifier with much extra data for discrimination of data points. Another significant drawback is that there is always a risk of incorrectly labelled points, which has serious implications on classifier performance.

With the potential problems in mind, each of the algorithms are given threshold values to stop them overpopulating. The results from these algorithms showed a significant improvement over the labels generated from experience sampling alone, and its main enhancements are improving the performance of under-represented classes.

Chapter 7 – Reducing the requirement for expert feature analysis

This chapter focuses on how to further increase the usability of user-trained activity recognition systems by removing the requirement for a domain expert to extract features from the raw data.

The raw data used within this chapter was the HAPT dataset, and statistical features had been extracted so that a baseline of performance with these features could be created and compared with. Methodologies from Chapter 3 are used within this Chapter 5, as they will be providing the weak labels.

The automatically generated features were provided using deep learning and a convolutional neural network. The neural network used consisted of an input layer, two convolutional layers, a max-pooling layer and two fully connected layers. The output is generated using a softmax activation function and the other layers use a ReLu activation function.

For the results, the first discussed are the comparison of deep features with traditional features and it was found that there was only a small drop in performance seen in the deep features. This could potentially be improved through further reconfiguration and tuning of the deep network, but unfortunately, no distinct method of creating deep networks current exists.

The second set of results created baselines of a weakly labelled dataset with a fully labelled dataset using normal features. This is a similar experiment to the ones performed within Chapter 3 and is used as a baseline for the final set of results.

The final experiment ran weak supervision with deep features, and it was found that the performance does indeed drop when compared to weak supervision with traditional features. However, the performance could still be considered acceptable and improvements could be seen through the better configuration of the network as previously mentioned.

Thesis Contributions

The main contributions of this thesis are through the unique combination of weak supervision techniques with activity recognition systems. These methods will allow activity recognition experiments to be performed in free-living environments with either the user or someone monitoring the user providing labels to the system.

The main contributions and methods provided are:

6. A novel method of collecting and applying labels to activity data using techniques derived from weak supervision and clustering. (Chapter 3)
7. Two novel methods of reducing the number of user-requests required when collecting labels from users for activity recognition. (Chapter 4)
8. A novel method of recognizing the transitions between activities (Chapter 5)
9. Two novel methods of propagating activity labels from known labelled points to unlabeled points (Chapter 6)
10. A novel method of automatically generating feature vectors from raw activity data and weak labels (Chapter 7)

Future work

During this research, solutions to challenging problems associated with the work under investigation have been put forward and tested in realistic settings. However, the results of the work have highlighted additional challenges and other related areas of research worthy of investigation. For example, while experience sampling does provide us with an interesting method of gathering labels, it does have some drawbacks.

The first drawback is that whenever a user is polled for data, they will be required to make movements in order to annotate the activity. For instance, if a user receives a notification from the recognition application, they are going to have to raise the phone up (if they are not already looking at it), unlock

the phone, potentially navigate to the recognition application and type in the activity label. All these movements will be tracked on the internal gyroscope and accelerometer and be incorrectly added to the currently performed activities data. This could be conflicting for a classifier. Potential solutions to this problem could include stopping the sensors from recording after a request has been made and only resuming once they have provided the label. Or alternatively, ask what activity a user was performing a fixed time interval ago, so for example, asking the user “what activity were you performing 30 seconds ago”.

Experience sampling also has the issue of potentially being a burden on the user, for example, through excessive requests for input. Solutions to this problem have been discussed within this thesis, however, other ideas have not yet been tested. One potential example is to remove the requirement for fixed time intervals between requests or by making cost-benefit calculations on the next data. For example, high benefit data could be if the recognition system has low confidence in the data being received and cost could be inferred from certain markers on the phone e.g. do not disturb mode or if a request has been made very recently. Feature engineering could also be used to infer the user’s tolerance to being interrupted, for example, by looking at the time they take to respond when performing certain activities.

Another consideration is performance, if such a system is intended to be run on smart-devices, which are mostly battery powered with limited computing power. Then careful consideration needs to be taken of the overheads of this recognition systems. Re-training classifiers can be intense computationally, however, the burden of this could be reduced by only doing it, for example, when the device is plugged in and charging.

9. References

[1] C. Wong, Z. Q. Zhang, B. Lo, and G. Z. Yang, “Wearable Sensing for Solid Biomechanics: A Review,” *IEEE Sensors Journal*, vol. 15, no. 5. pp. 2747–2760, 2015, doi: 10.1109/JSEN.2015.2393883.

[2] M. Stikic, D. Larlus, S. Ebert, and B. Schiele, “Weakly supervised recognition of daily life activities with wearable sensors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2521–2537, 2011, doi: 10.1109/TPAMI.2011.36.

[3] W. Y. Wong, M. S. Wong, and K. H. Lo, “Clinical applications of sensors for human posture and movement analysis: a review.,” *Prosthet. Orthot. Int.*, vol. 31, no. 1, pp. 62–75, 2007, doi: 10.1080/03093640600983949.

[4] Y. L. Zheng *et al.*, “Unobtrusive sensing and wearable devices for health informatics,” *IEEE Trans. Biomed. Eng.*, vol. 61, no. 5, pp. 1538–1554, 2014, doi: 10.1109/TBME.2014.2309951.

[5] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2004.

[6] M. Stikic, K. Van Laerhoven, and B. Schiele, “Exploring semi-supervised and active learning for activity recognition,” *2008 12th IEEE Int. Symp. Wearable Comput.*, pp. 81–88, 2008, doi: 10.1109/ISWC.2008.4911590.

[7] J. Hernández-González, I. Inza, and J. A. Lozano, “Weak supervision and other non-standard classification problems: A taxonomy,” *Pattern Recognit. Lett.*, vol. 69, pp. 49–55, 2016, doi: 10.1016/j.patrec.2015.10.008.

[8] A. Kapoor and E. Horvitz, “Experience sampling for building predictive user models,” *Proceeding twenty-sixth Annu. CHI Conf. Hum. factors Comput. Syst. - CHI ’08*, pp. 657–666, 2008, doi: 10.1145/1357054.1357159.

[9] D. Roggen *et al.*, “Collecting complex activity datasets in highly rich networked sensor environments,” in *INSS 2010 - 7th International Conference on Networked Sensing Systems*, 2010, pp. 233–240, doi: 10.1109/INSS.2010.5573462.

[10] L. Bao and S. S. Intille, “Activity Recognition from User-Annotated Acceleration Data,” *Pervasive Comput.*, pp. 1–17, 2004, doi: 10.1007/b96922.

[11] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga, “Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey,” in *23th International Conference on Architecture of Computing Systems 2010*, 2010, pp. 1–10.

[12] M. Panwar *et al.*, “CNN based approach for activity recognition using a wrist-worn accelerometer,” 2017, doi: 10.1109/EMBC.2017.8037349.

[13] L. Atallah, B. Lo, R. King, and G. Z. Yang, “Sensor positioning for activity recognition using wearable accelerometers,” 2011, doi: 10.1109/TBCAS.2011.2160540.

[14] E. J. W. Van Someren *et al.*, “A new actigraph for long-term registration of the duration and intensity of tremor and movement,” *IEEE Trans. Biomed. Eng.*, vol. 45, no. 3, pp. 386–395, 1998, doi: 10.1109/10.661163.

[15] L. Cao, Y. Wang, B. Zhang, Q. Jin, and A. V. Vasilakos, “GCHAR: An efficient Group-based Context—aware human activity recognition on smartphone,” *J. Parallel Distrib. Comput.*, vol. 118, pp. 67–80, 2018, doi: 10.1016/j.jpdc.2017.05.007.

[16] S. Jiang *et al.*, “CareNet: An Integrated Wireless Sensor Networking Environment for Remote Healthcare,” 2008, doi: 10.4108/ICST.BODYNETS2008.2965.

[17] D. Kelly and B. Caulfield, “An investigation into non-invasive physical activity recognition using smartphones.,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2012, pp. 3340–3, 2012, doi: 10.1109/EMBC.2012.6346680.

[18] J. C. Hou *et al.*, “PAS: A wireless-enabled, sensor-integrated personal assistance system for independent and assisted living,” in *Proceedings - 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, HCMDSS/MDPnP 2007*, 2007, pp. 64–75, doi: 10.1109/HCMDSS-MDPnP.2007.13.

[19] M. Ermes, J. Pärkkä, J. Mäntyjärvi, and I. Korhonen, “Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions,” *IEEE Trans. Inf. Technol. Biomed.*, 2008, doi: 10.1109/TITB.2007.899496.

[20] C. J. Caspersen, K. E. Powell, and G. M. Christenson, “Physical activity, exercise, and physical fitness: definitions and distinctions for health-related research.,” *Public Health Rep.*, vol. 100, no. 2, pp. 126–131, 1985.

[21] J. L. Rodrigues, N. Gonçalves, S. Costa, and F. Soares, “Stereotyped movement recognition in children with ASD,” *Sensors Actuators, A Phys.*, vol. 202, pp. 162–169, 2013, doi: 10.1016/j.sna.2013.04.019.

[22] M. Reiner, C. Niermann, D. Jekauc, and A. Woll, “Long-term health benefits of physical activity - A systematic review of longitudinal studies,” *BMC Public Health*. 2013, doi: 10.1186/1471-2458-13-813.

[23] X. Su, H. Tong, and P. Ji, “Activity recognition with smartphone sensors,” *Tsinghua Sci. Technol.*, 2014, doi: 10.1109/TST.2014.6838194.

[24] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, “Simple and complex activity recognition through smart phones,” 2012, doi: 10.1109/IE.2012.39.

[25] J. M. Wiener, R. J. Hanley, R. Clark, and J. F. Van Nostrand, “Measuring the activities of daily living: comparisons across national surveys.,” *J. Gerontol.*, vol. 45, no. 6, pp. S229–S237, 1990, doi: 10.1093/geronj/45.6.S229.

[26] V. S. Thomas, K. Rockwood, and I. McDowell, “Multidimensionality in instrumental and basic activities of daily living,” *J. Clin. Epidemiol.*, vol. 51, no. 4, pp. 315–321, 1998, doi: 10.1016/S0895-4356(97)00292-8.

[27] P. Casale, O. Pujol, and P. Radeva, “Human activity recognition from accelerometer data using a wearable device,” *Pattern Recognit. Image Anal.*, pp. 289–296, 2011, doi: 10.1007/978-3-642-21257-4.

[28] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, “Activity recognition from accelerometer data,” 2005.

[29] D. Kelly, G. F. Coughlan, B. S. Green, and B. Caulfield, “Automatic detection of collisions in elite level rugby union using a wearable sensing device,” *Sport. Eng.*, 2012, doi: 10.1007/s12283-012-0088-5.

[30] N. Shahar, N. F. Ghazali, M. A. As’ari, T. S. Tan, and M. F. Ibrahim, “Investigation of Different Time-Series Segmented Windows from Inertial Sensor for Field Hockey Activity Recognition,” 2020.

[31] M. O’Reilly, B. Caulfield, T. Ward, W. Johnston, and C. Doherty, “Wearable Inertial Sensor Systems for Lower Limb Exercise Detection and Evaluation: A Systematic Review,” *Sports Medicine*. 2018, doi: 10.1007/s40279-018-0878-4.

[32] M. A. O’Reilly, D. F. Whelan, T. E. Ward, E. Delahunt, and B. M. Caulfield, “Classification of deadlift biomechanics with wearable inertial measurement units,” *J. Biomech.*, 2017, doi: 10.1016/j.jbiomech.2017.04.028.

[33] S. Thiemjarus, “A device-orientation independent method for activity recognition,” 2010, doi: 10.1109/BSN.2010.55.

[34] J. Lester, T. Choudhury, and G. Borriello, “A practical approach to recognizing physical activities,” 2006.

[35] C. V. C. Bouten, K. T. M. Koekkoek, M. Verduin, R. Kodde, and J. D. Janssen, “A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity,” *IEEE Trans. Biomed. Eng.*, 1997, doi: 10.1109/10.554760.

[36] M. J. Mathie, A. C. F. Coster, N. H. Lovell, and B. G. Celler, “Accelerometry: Providing an integrated, practical method for long-term, ambulatory monitoring of human movement,” *Physiological Measurement*. 2004, doi: 10.1088/0967-3334/25/2/R01.

[37] A. Yurtman and B. Barshan, “Activity recognition invariant to sensor orientation with wearable motion sensors,” *Sensors (Switzerland)*, vol. 17, no. 8, 2017, doi: 10.3390/s17081838.

[38] A. Bayat, M. Pomplun, and D. A. Tran, “A study on human activity recognition using accelerometer data from smartphones,” 2014, doi: 10.1016/j.procs.2014.07.009.

[39] E. Horvitz, P. Koch, R. Sarin, and J. Apacible, “Bayesphone : Precomputation of Context-Sensitive Policies for Inquiry and Action in Mobile Devices,” *User Model. 2005*, pp. 251–260, 2005, doi: 10.1007/11527886_33.

[40] L. Dabbish, R. Kraut, S. Fussell, and S. Kiesler, “Understanding email use: predicting action on a message,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 691–700, doi: 10.1086/300104.

[41] D. Avrahami and S. E. Hudson, “Responsiveness in Instant Messaging: Predictive Models Supporting Inter-Personal Communication,” in *CHI 2006 Proceedings*, 2006, pp. 731–740, doi: 10.1145/1124772.1124881.

[42] S. Liu, R. Gao, D. John, J. Staudenmayer, and P. Freedson, “Multi-Sensor Data Fusion for Physical Activity Assessment,” *IEEE Trans. Biomed. Eng.*, vol. PP, no. 99, pp. 1–1, 2011, doi: 10.1109/TBME.2011.2178070.

[43] N. Selvaraj, A. Jaryal, J. Santhosh, K. K. Deepak, and S. Anand, “Assessment of heart rate variability derived from finger-tip photoplethysmography as compared to electrocardiography,” *J. Med. Eng. Technol.*, vol. 32, no. 6, pp. 479–484, 2008, doi: 10.1080/03091900701781317.

[44] G. Fortino and V. Giampà, “PPG-based methods for non invasive and continuous blood pressure measurement: an overview and development issues in body sensor networks,” *Med. Meas. Appl. Proc. (MeMeA), 2010 IEEE Int. Work.*, pp. 1–4, 2010, doi: 10.1109/MEMEA.2010.5480201.

[45] F. Lamonaca, D. L. Carni, D. Grimaldi, A. Nastro, M. Riccio, and V. Spagnolo, “Blood oxygen saturation measurement by smartphone camera,” in *2015 IEEE International Symposium on Medical Measurements and Applications, MeMeA 2015 - Proceedings*, 2015, pp. 359–364, doi: 10.1109/MeMeA.2015.7145228.

[46] M. A. A. H. Khan, N. Roy, and A. Misra, “Scaling Human Activity Recognition via Deep Learning-based Domain Adaptation,” 2018, doi: 10.1109/PERCOM.2018.8444585.

[47] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, “Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 156–167, 2006, doi: 10.1109/TITB.2005.856864.

[48] D. S??nchez Morillo, J. L. R. Ojeda, L. F. C. Foix, and A. L. Jim??nez, “An accelerometer-based device for sleep apnea screening,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 491–499, 2010, doi: 10.1109/TITB.2009.2027231.

[49] H. Gjoreski and M. Gams, “Activity/posture recognition using wearable sensors placed on different body locations,” 2011, doi: 10.2316/P.2011.716-067.

[50] B. Delporte, L. Perroton, T. Grandpierre, and J. Trichet, “Accelerometer and magnetometer based gyroscope emulation on smart sensor for a virtual reality application,” *Sensors and Transducers*, 2012.

[51] L. L. Chen, J. Zhang, J. Z. Zou, C. J. Zhao, and G. S. Wang, “A framework on wavelet-based nonlinear features and extreme learning machine for epileptic seizure detection,” *Biomed. Signal Process. Control*, 2014, doi: 10.1016/j.bspc.2013.11.010.

[52] A. J. Cook, G. D. Gargiulo, T. Lehmann, and T. J. Hamilton, “Open platform, eight-channel, portable bio-potential and activity data logger for wearable medical device development,” *Electron. Lett.*, 2015, doi: 10.1049/el.2015.2764.

[53] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, “Using mobile phones to determine transportation modes,” *ACM Trans. Sens. Networks*, 2010, doi: 10.1145/1689239.1689243.

[54] K. Zhan, F. Ramos, and S. Faux, “Activity recognition from a wearable camera,” 2012, doi: 10.1109/ICARCV.2012.6485186.

[55] J. L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-Aware Human Activity Recognition Using Smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016, doi: 10.1016/j.neucom.2015.07.085.

[56] N. Yala, B. Fergani, and A. Fleury, “Feature extraction for human activity recognition on streaming data,” in *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on*, 2015, pp. 1–6, doi: 10.1109/INISTA.2015.7276759.

[57] O. Banos, J. M. Galvez, M. Damas, H. Pomares, and I. Rojas, “Window size impact in human activity recognition,” *Sensors (Switzerland)*, 2014, doi: 10.3390/s140406474.

[58] S. Pirttikangas, K. Fujinami, and T. Nakajima, “Feature selection and activity recognition from wearable sensors,” 2006.

[59] N. Kern, B. Schiele, and A. Schmidt, “Multi-sensor activity context detection for wearable computing,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2003.

[60] K. Van Laerhoven and O. Cakmakci, “What shall we teach our pants?,” *Int. Symp. Wearable Comput. Dig. Pap.*, 2000.

[61] T. Huynh and B. Schiele, “Analyzing features for activity recognition,” 2005, doi: 10.1145/1107548.1107591.

[62] J. A. Ward, P. Lukowicz, and G. Tröster, “Gesture spotting using wrist worn microphone and 3-axis accelerometer,” 2005, doi: 10.1145/1107548.1107578.

[63] Z. Y. He and L. W. Jin, “Activity recognition from acceleration data using AR model representation and SVM,” 2008, doi: 10.1109/ICMLC.2008.4620779.

[64] A. Krause, A. Smailagic, D. P. Siewiorek, and J. Farringdon, “Unsupervised, dynamic identification of physiological and activity context in wearable computing,” 2003.

[65] J. Y. Yang, J. S. Wang, and Y. P. Chen, “Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers,” *Pattern Recognit. Lett.*, 2008, doi: 10.1016/j.patrec.2008.08.002.

[66] M. Khan, S. I. Ahamed, M. Rahman, and R. O. Smith, “A feature extraction method for realtime human activity recognition on cell phones,” *Proc. 3rd Int. Symp. Qual. Life Technol.*

(isQoLT 2011). Toronto, Canada, 2011.

[67] Y. Z. Lin, Z. H. Nie, and H. W. Ma, "Structural Damage Detection with Automatic Feature-Extraction through Deep Learning," *Comput. Civ. Infrastruct. Eng.*, 2017, doi: 10.1111/mice.12313.

[68] G. L. Santos, P. T. Endo, K. H. de C. Monteiro, E. da S. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors (Switzerland)*, 2019, doi: 10.3390/s19071644.

[69] A. Ignatov, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks," *Appl. Soft Comput. J.*, 2018, doi: 10.1016/j.asoc.2017.09.027.

[70] M. Mathie, "Monitoring and interpreting human movement patterns using a triaxial accelerometer," *Adv. Cancer Res.*, 2003.

[71] M. J. Mathie, B. G. Celler, N. H. Lovell, and A. C. F. Coster, "Classification of basic daily movements using a triaxial accelerometer," *Med. Biol. Eng. Comput.*, 2004, doi: 10.1007/BF02347551.

[72] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*. 2018, doi: 10.1016/j.eswa.2018.03.056.

[73] P. Nurmi and P. Floréen, "Online feature selection for contextual time series data," *SLSFS*, pp. 1–6, 2005, [Online]. Available: http://www.cs.helsinki.fi/u/ptnurmi/papers/slsfs05.pdf.

[74] I. P. Machado, A. Luísa Gomes, H. Gamboa, V. Paixão, and R. M. Costa, "Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization," *Inf. Process. Manag.*, vol. 51, no. 2, pp. 201–214, 2015, doi: 10.1016/j.ipm.2014.07.008.

[75] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1473–1488, 2008, doi: 10.1109/TCSVT.2008.2005594.

[76] O. D. Lara and M. a. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013, doi: 10.1109/SURV.2012.110112.00192.

[77] X. Goldberg, “Introduction to semi-supervised learning,” *Synth. Lect. Artif. Intell. Mach. Learn.*, 2009, doi: 10.2200/S00196ED1V01Y200906AIM006.

[78] P. Mehta *et al.*, “A high-bias, low-variance introduction to Machine Learning for physicists,” *Physics Reports*. 2019, doi: 10.1016/j.physrep.2019.03.001.

[79] R. Bellman, “The Theory of Dynamic Programming,” *Bull. Am. Math. Soc.*, 1954, doi: 10.1090/S0002-9904-1954-09848-8.

[80] D. L. Olson and D. Delen, *Advanced data mining techniques*. 2008.

[81] M. V. Albert, S. Toledo, M. Shapiro, and K. Kording, “Using mobile phones for activity recognition in Parkinson’s patients,” *Front. Neurol.*, 2012, doi: 10.3389/fneur.2012.00158.

[82] C. E. Metz, “Basic principles of ROC analysis,” *Semin. Nucl. Med.*, 1978, doi: 10.1016/S0001-2998(78)80014-2.

[83] S. Chen, S. Gunn, and C. J. Harris, “Decision feedback equaliser design using support vector machines,” *IEE Proc. Vision, Image Signal Process.*, 2000, doi: 10.1049/ip-vis:20000360.

[84] A. Widodo and B.-S. Yang, “Support vector machine in machine condition monitoring and fault diagnosis,” *Mech. Syst. Signal Process.*, vol. 21, no. 6, pp. 2560–2574, 2007, doi: 10.1016/j.ymssp.2006.12.007.

[85] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7657 LNCS, pp. 216–223, doi: 10.1007/978-3-642-35395-6_30.

[86] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li, “Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations,” 2010, doi: 10.1007/978-3-642-16355-5_42.

[87] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “Energy efficient smartphone-based activity recognition using fixed-point arithmetic,” *J. Univers. Comput. Sci.*, 2013.

[88] A. Subasi, M. Radhwan, R. Kurdi, and K. Khateeb, “IoT based mobile healthcare system for human activity recognition,” 2018, doi: 10.1109/LT.2018.8368507.

[89] B. M. Abidine, L. Fergani, B. Fergani, and M. Oussalah, “The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition,” *Pattern*

Anal. Appl., 2018, doi: 10.1007/s10044-016-0570-y.

[90] A. K. Chowdhury, D. Tjondronegoro, V. Chandran, and S. G. Trost, "Physical activity recognition using posterior-adapted class-based fusion of multiaccelerometer data," *IEEE J. Biomed. Heal. Informatics*, 2018, doi: 10.1109/JBHI.2017.2705036.

[91] Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and SVM," 2009, doi: 10.1109/ICSMC.2009.5346042.

[92] J. R. Kwapisz, G. M. Weiss, and S. a. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newsl.*, vol. 12, no. 2, p. 74, 2011, doi: 10.1145/1964897.1964918.

[93] J. Mäntyjärvi, J. Himberg, and T. Seppänen, "Recognizing human motion with multiple acceleration sensors," *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2001, doi: 10.1109/ICSMC.2001.973004.

[94] S. K. Song, J. Jang, and S. Park, "A phone for human activity recognition using triaxial acceleration sensor," 2008, doi: 10.1109/ICCE.2008.4587903.

[95] C. A. Ronao and S. B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, 2016, doi: 10.1016/j.eswa.2016.04.032.

[96] W. Xu, Y. Pang, Y. Yang, and Y. Liu, "Human Activity Recognition Based On Convolutional Neural Network," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 165–170, doi: 10.1109/ICPR.2018.8545435.

[97] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," 2015, doi: 10.1145/2733373.2806333.

[98] A. Murad and J. Y. Pyun, "Deep recurrent neural networks for human activity recognition," *Sensors (Switzerland)*, 2017, doi: 10.3390/s17112556.

[99] M. Zeng *et al.*, "Convolutional Neural Networks for human activity recognition using mobile sensors," 2015, doi: 10.4108/icst.mobicase.2014.257786.

[100] A. Romero, C. Gatta, and G. Camps-Valls, "Unsupervised deep feature extraction for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, 2016, doi: 10.1109/TGRS.2015.2478379.

[101] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Futur. Gener. Comput.*

Syst., 2017, doi: 10.1016/j.future.2017.11.029.

[102] M. Kose, O. D. Incel, and C. Ersoy, “Online Human Activity Recognition on Smart Phones,” *Perform. Eval.*, 2012.

[103] K. Förster, S. Monteleone, A. Calatroni, D. Roggen, and G. Tröster, “Incremental kNN classifier exploiting correct-error teacher for activity recognition,” 2010, doi: 10.1109/ICMLA.2010.72.

[104] G. De Leonardis *et al.*, “Human Activity Recognition by Wearable Sensors : Comparison of different classifiers for real-time applications,” 2018, doi: 10.1109/MeMeA.2018.8438750.

[105] A. Wijekoon, N. Wiratunga, and S. Sani, “Zero-shot learning with matching networks for open-ended human activity recognition,” 2018.

[106] H. T. Cheng, M. Griss, P. Davis, J. Li, and D. You, “Towards zero-shot learning for human activity recognition using semantic attribute sequence model,” 2013, doi: 10.1145/2493432.2493511.

[107] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao, “Learning from weak and noisy labels for semantic segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, doi: 10.1109/TPAMI.2016.2552172.

[108] G. Papandreou, L. C. Chen, K. P. Murphy, and A. L. Yuille, “Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation,” 2015, doi: 10.1109/ICCV.2015.203.

[109] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, “Solving the multiple instance problem with axis-parallel rectangles,” *Artif. Intell.*, vol. 89, no. 1–2, pp. 31–71, 1997, doi: 10.1016/S0004-3702(96)00034-3.

[110] Z. Jorgensen, Y. Zhou, and M. Inge, “A Multiple Instance Learning Strategy for Combating Good Word Attacks on Spam Filters,” *J. Mach. Learn. Res.*, vol. 8, pp. 1115–1146, 2008, [Online]. Available: http://jmlr.csail.mit.edu/papers/volume9/jorgensen08a/jorgensen08a.pdf.

[111] Y. Chen and J. Z. Wang, “Image Categorization by Learning and Reasoning with Regions,” *J. Mach. Learn. Res.*, vol. 5, pp. 913–939, 2004, [Online]. Available: http://portal.acm.org/citation.cfm?id=1005332.1016789.

[112] M. Kim and F. De la Torre, “Multiple instance learning via Gaussian processes,” *Data Min. Knowl. Discov.*, vol. 28, no. 4, pp. 1078–1106, 2013, doi: 10.1007/s10618-013-0333-y.

[113] B. Zeisl, C. Leistner, A. Saffari, and H. Bischof, “On-line semi-supervised multiple-instance boosting,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 1879–1886, doi: 10.1109/CVPR.2010.5539860.

[114] G. Fung, M. Dundar, B. Krishnapuram, and R. B. Rao, “Multiple instance learning for computer aided diagnosis,” *Nips*, pp. 425–432, 2007, [Online]. Available: http://books.google.com/books?hl=en&lr=&id=Tbn1l9P1220C&oi=fnd&pg=PA425&dq=Multiple+Instance+Learning+for+Computer+Aided+Diagnosis&ots=V2kcFgrl0-&sig=oYqafE4kCNZQmpb_dUNGIq2OeWY.

[115] R. C. Bunescu and R. J. Mooney, “Multiple instance learning for sparse positive bags,” *Proc. 24th Int. Conf. Mach. Learn.*, vol. 79, no. June, pp. 105–112, 2007, doi: 10.1145/1273496.1273510.

[116] S. Andrews, T. Hofmann, and I. Tsochantaridis, “Multiple Instance Learning with Generalized Support Vector Machines,” *AAAI*, pp. 943–944, 2002.

[117] Z. Zhou and M. Zhang, “Neural Networks for Multi-Instance Learning,” *Int’l Conf. Intell. Inf. Technol.*, pp. 455–459, 2002.

[118] O. Maron and T. Lozano-Pérez, “A framework for multiple-instance learning,” *Adv. Neural Inf. Process. Syst.*, pp. 570–576, 1998, doi: 10.1613/jair.301.

[119] L. Jiang, Z. Cai, D. Wang, and H. Zhang, “Bayesian Citation-KNN with distance weighting,” *Int. J. Mach. Learn. Cybern.*, vol. 5, no. 2, pp. 193–199, 2014, doi: 10.1007/s13042-013-0152-x.

[120] B. Settles, “Active Learning Literature Survey,” *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 2010, doi: 10.1.1.167.4245.

[121] V. S. Sheng, F. Provost, and P. G. Ipeirotis, “Get another label? improving data quality and data mining using multiple, noisy labelers,” *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, pp. 614–622, 2008, doi: 10.1145/1401890.1401965.

[122] Z. Khan *et al.*, “Ensemble of optimal trees, random forest and random projection ensemble classification,” *Adv. Data Anal. Classif.*, 2019, doi: 10.1007/s11634-019-00364-9.

[123] W. Choi, K. Shahid, and S. Savarese, “Learning context for collective activity recognition,” 2011, doi: 10.1109/CVPR.2011.5995707.

[124] K. Ellis, J. Kerr, S. Godbole, G. Lanckriet, D. Wing, and S. Marshall, “A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip

accelerometers," *Physiol. Meas.*, 2014, doi: 10.1088/0967-3334/35/11/2191.

[125] C. Zhang and Y. Ma, *Ensemble machine learning: Methods and applications*. 2012.

[126] G. M. Weiss and J. W. Lockhart, "The impact of personalization on smartphone-based activity recognition," 2012.

[127] L. Wang, K. Murao, H. Gjoreski, T. Okita, and D. Roggen, "Summary of the Sussex-Huawei locomotion-transportation recognition challenge," 2018, doi: 10.1145/3267305.3267519.

[128] R. E. Schapire, "The Strength of Weak Learnability," *Mach. Learn.*, 1990, doi: 10.1023/A:1022648800760.

[129] B. Agarwal, A. Chakravorty, T. Wiktorski, and C. Rong, "Enrichment of machine learning based activity classification in smart homes using ensemble learning," 2016, doi: 10.1145/2996890.3007861.

[130] V. Janko *et al.*, "A new frontier for activity recognition - The Sussex-Huawei locomotion challenge," 2018, doi: 10.1145/3267305.3267518.

[131] H. Gong, K. Xing, and W. Du, "A user activity pattern mining system based on human activity recognition and location service," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 1–2, doi: 10.1109/INFCOMW.2018.8406918.

[132] Z. Peng and Y. Zhang, "Human motion classification based on inertial sensors with extreme gradient boosting," 2018, doi: 10.1117/12.2514563.

[133] K. Gusain, A. Gupta, and B. Popli, "Transition-aware human activity recognition using eXtreme gradient boosted decision trees," 2018, doi: 10.1007/978-981-10-4603-2_5.

[134] J. W. Lockhart and G. M. Weiss, "The Benefits of Personalized Smartphone-Based Activity Recognition Models," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, 2014, pp. 614–622.

[135] S. Bhattacharya and N. D. Lane, "From smart to deep: Robust activity recognition on smartwatches using deep learning," 2016, doi: 10.1109/PERCOMW.2016.7457169.

[136] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newsl.*, vol. 12, no. 2, p. 74, 2011, doi: 10.1145/1964897.1964918.

[137] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, “Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach,” 2012, doi: 10.1109/ISWC.2012.23.

[138] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R. Zaccaria, “Human motion modelling and recognition: A computational approach,” 2012, doi: 10.1109/CoASE.2012.6386410.

[139] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R. Zaccaria, “Analysis of human behavior recognition algorithms based on acceleration data,” 2013, doi: 10.1109/ICRA.2013.6630784.

[140] S. Ray and M. Craven, “Supervised versus multiple instance learning: An empirical comparison,” in *… 22nd international conference on Machine learning*, 2005, pp. 697–704, [Online]. Available: http://dl.acm.org/citation.cfm?id=1102439.

[141] I. Brown and C. Mues, “An experimental comparison of classification algorithms for imbalanced credit scoring data sets,” *Expert Syst. Appl.*, 2012, doi: 10.1016/j.eswa.2011.09.033.

[142] P. Casale, O. Pujol, and P. Radeva, “Human activity recognition from accelerometer data using a wearable device,” *Pattern Recognit. Image Anal.*, pp. 289–296, 2011, doi: 10.1007/978-3-642-21257-4.

[143] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga, “A Survey of Online Activity Recognition Using Mobile Phones,” *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015, doi: 10.3390/s150102059.

[144] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package),” *Neurocomputing*, 2018, doi: 10.1016/j.neucom.2018.03.067.

[145] A. W. Kempa-Liehr, J. Oram, A. Wong, M. Finch, and T. Besier, “Feature Engineering Workflow for Activity Recognition from Synchronized Inertial Measurement Units,” 2020, doi: 10.1007/978-981-15-3651-9_20.

[146] J. J. Dominguez Veiga, M. O’Reilly, D. Whelan, B. Caulfield, and T. E. Ward, “Feature-Free Activity Classification of Inertial Sensor Data With Machine Vision Techniques: Method, Development, and Evaluation,” *JMIR mHealth uHealth*, 2017, doi: 10.2196/mhealth.7521.