An Investigation of Machine Learning and Clustering Algorithms in Human Activity Recognition and Behavioural Prediction using MHEALTH Data

By

Jamie O'Halloran

M.Sc. Computer Science (Data Analytics)

12458152

Supervisor

Ed Curry

Computer Science (Data Analytics)

College of Engineering and Informatics

National University of Ireland, Galway

August 2019

Declaration of Originality

I declare that this thesis is my original work except where stated.

Signed: _____

Date: _____

Acknowledgment

I would like to express my gratitude to my supervisor Ed Curry for guidance and continuous support throughout the research. His advice has been invaluable, facilitating me to overcome many obstacles throughout the completion of the research. I am grateful for the continuous supervision, guidance and motivation that made the journey that little bit easier.

I would like to thank my program director, Enda Howley, for his guidance, encouragement and knowledge throughout the year.

Finally, I would like to thank my family and friends for their support throughout my studies.

Abstract

The purpose of this research was to conduct an investigation of deep learning and dimensionality reduction techniques in human activity recognition and behavioural prediction using MHEALTH data. These techniques combined with multiple sensor data aim to classify daily activities. Previous work in HAR has focused on using multiple accelerometers placed on different parts of the body, with more recent work focused on sensors embedded in smartphones to classify activities. This research classifies activities from utilising the data from the following sensors – accelerometer, gyroscope and magnetometer.

The aim of this research was to conduct an investigation and compare six different machine and deep learning algorithms with each other to evaluate which network best suits the MHEALTH dataset. Evaluation is carried out by comparing each networks accuracy, precision, recall and f1-score. In addition to comparing these evaluation metrics, a comparison of each networks confusion matrix, feature importance and multisensory fusion analysis is performed – in order to evaluate which network best suits the data and successfully classifies the daily activities in question. Another intriguing aim of this research is to compare two data clustering techniques for visualising the MHEALTH dataset. This research aims to present the best visualisation technique by conducting a comparative study on the two-visualisation techniques.

The result of this research found that all six-machine learning classification algorithms consistently outperformed State-of-the-Art baselines. XGBoost (89.97% accuracy) and MLP (90.55%) accuracy performed excellently on the data, with very little misclassified instances. All six-classification algorithms produced more insightful, predictive results than existing baselines, while DBSCAN successfully clustered and visualised the data. The results show that each algorithm is suited to the MHEALTH dataset, with high performance results achieved throughout.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Continuous growth in the health sector has led to astronomical advancements in the field of medicine. Due to this continuous growth, the quality of life has greatly increased when compared to one hundred years ago. Everything from life expectancy, physical health, education, safety and freedom has vastly improved. This rise in health sector growth has led to an increase in healthcare costs. Increased healthcare costs in monitoring patients with chronic illness, monitoring the elderly along with many more instances has led to drastic cost-cutting measures employed by healthcare providers worldwide.

Technological advancements in healthcare can contribute unquestionably in reducing healthcare costs by ensuring clinicians, doctors and other medical staff operate and conduct their daily activities more efficiently in the hospital vicinity. Since the turn of the 21st century, Human Activity Recognition (HAR) has undergone significant research in the healthcare domain. Human activity recognition utilised with powerful technologies can potentially benefit remote patient monitoring, the elderly, patients suffering from chronic illness and ambient assisted living.

Simple activities such as cycling, running and jogging have been successfully recognised and classified to date. Complex activities are proving increasingly difficult to monitor, with continuous active research conducted in this area of HAR. The main goal of HAR is to predict common activities in real-life surroundings. Researchers are exploring pattern recognition and human-computer relationships due to its applicability in the real world, such as a Human Activity Recognition Healthcare Framework. Successfully classifying human activities through wearable sensors generates endless individual information, which provides insight about the individuals' functional ability, lifestyle and health.

In this research, the MHEALTH dataset is analysed using a variety of deep learning models. These models aim to classify activities performed by volunteers based on data gathered from on-body inertial sensors. Exploratory analysis distinguishes differences and similarities between these deep learning models throughout this research. The overall aim is to identify which algorithm best suits the data, while discovering which algorithm best classifies each individual body movement of each person based on vital signs recordings. Two data clustering algorithms analysis identify relationships between feature attributes and pleasantly visualise the data.

1.1 Motivation

Human activity recognition has shown to be effective in benefiting clinicians in the treatment and remote monitoring of patients. This field is not only vital for diagnosis and treatment, but also an assessment of how likely a medical patient will fall ill or die from certain diseases or health problems. To show the great importance of activity recognition in the health sector, analytically driving an improvement in accuracy in classifying patients' activities improves the relationship of patients and clinicians as well as reducing the possibility of a fatality.

This project revolves around the topic of using deep learning to benefit the healthcare industry. One aspect that deep learning could benefit is Remote Patient Monitoring (RPM). The sufficient monitoring of remote persons' activities in real-time can yield great benefits in medical environments. Doctors, nurses and clinicians can build strong relationships with, and improve the experience of their patients, by analysing data send to them via RPM technologies. As seen in [1], the data sent to them via RPM can develop a personalised care plan and to engage in joint decision-making to foster better outcomes. Wearable sensors, generating this data, can feed data to a clinician in real-time, leading to a significant reduction in continuous patient monitoring.

This system could be beneficial to the elderly, those suffering from chronic illness and those who are prone to heart attacks (or serious medical conditions). According to [2], chronic heart failure (CHF) is the most common cause of readmission for patients in the USA. It is estimated that up to 84% of readmissions within a 7-day period were considered preventable, while 76% of 30-day readmission were also considered preventable [2].

1.2 Motivational Scenario

The best way to provide protection to patients that are prone to chronic heart failure, chronic illness, disease spreading as well as aiding remote patient monitoring and providing a quick response to fall detection is a Human Activity Recognition Health Framework. This would motivate the healthcare industry to implement this framework to contain and tackle the above problems.

In regards to preventing disease spreading, the primary task of the framework should be the early detection and prevention of the disease as oppose to recommending preventative measures to cure the diagnosis. The framework could provide accurate and timely measures ensuring the disease does not come to surface.

The framework can provide care to the elderly living alone. The framework could provide benefits to remote patient monitoring as a part of the intervention could be the early detection and prevention of an elderly person falling, signalling them to control certain movements and be more aware of surroundings.

This leads to monitoring patients who are suffering from chronic illness. Similar to preventing the spreading of disease, the framework could monitor and control the illness to ensure it does not take hold of the patient. The framework suggests preventative measures if the patient is in critical condition.

Prevention and control of the above issues can be a difficult test. It is necessary that advanced healthcare technology is available to monitor and predict these challenges. A reliable, precise prediction can save lives worldwide, ensuring the world is a healthier place, with piece of mind given to patients if they find themselves needing to contact a healthcare provider. Health workers, clinicians and doctors can measure, soften and control the severe, widespread effect of chronic health failure, chronic illness, disease spreading and fall detection with this Human Activity Recognition Health Framework.

1.3 Research Objective

The purpose of this research is to utilise the MHEALTH dataset to develop predictive models, which provides a forecast for human activity recognition. This research aims to present the best predictive analytics model for human activity recognition by conducting a comparative study between six predictive models evaluated on the MHEALTH dataset. The results are the outcome of six different comparative studies based on evaluating the MHEALTH dataset with six different predictive models.

Another purpose of this research is to evaluate and compare two different dimensionality reduction techniques for visualising this high, dimensional data. This research aims to present the best visualisation technique by conducting a comparative study on the two-visualisation models. The two models are DBSCAN and t-SNE.

1.4 Research Methodology

The purpose of this project is to classify health data from on-body inertial sensors - gyroscopes, accelerometers and magnetometers, with the outcome of classifying the volunteers' activities. This research creates an analytical platform to analyse health data and lies in the field of Human Activity Recognition, the Internet of Things and Deep Learning. This research focuses precisely on the machine learning and deep learning aspect of analysing health data generated from on-body inertial signals.

This project details a deep learning comparative study of the:

- Convolutional neural network model.
- Long short-term memory (Recurrent neural network) model.
- Convolutional long short-term memory (ConvLSTM) model.
- Multilayer Perceptron model.
- Extreme Gradient Boosting (XGBoost) model.
- Autoencoder by Random Forest model.

All six models employed in this research are interchangeable to any other relevant data-oriented predictions. The contributions of this research is to present six different classification models when evaluated on the MHEALTH dataset. The top two performing models are XGBoost and Multilayer Perceptron. The XGBoost model outperforms the MLP model when evaluated on the dataset by a minute margin.

Another contribution of this research is to present two different data-clustering algorithms, DBSCAN and t-SNE, when evaluated on the MHEALTH dataset. Due to the large size of the data and computational power of the computer used for research, the two clustering algorithms analyse each volunteer individually. DBSCAN outperformed t-SNE when evaluated on the dataset, clusters were precisely visualised giving a key insight into the relationship between the attributes.

1.5 Research Questions

To the best of the author's knowledge, previous studies have not addressed the following.

- The XGBoost machine learning algorithm is not implemented on the MHEALTH dataset.
- The data-clustering algorithm, DBSCAN, has not been implemented on the MHEALTH dataset.

This research poses a number of goals and research questions:

- To what extent can deep learning algorithms recognise physical human activity with on-body sensor data?
- Conduct a comparative study on different machine learning models to see which best fits the data.
- Does the data-clustering algorithm, DBSCAN, successfully visualise data clusters in the MHEALTH dataset?
- Does the machine learning algorithm, XGBoost, successfully classify activities in the MHEALTH dataset? Does it outperform existing state of the art baselines?

This thesis uses data extracted from accelerometers, barometers and magnetometers to classify activities based on vital signs recordings extracted from the classification of patients activities.

All of the research questions face significance overlap throughout this research. The final chapter discusses each question along with specific results and conclusions.

1.6 MHEALTH Dataset

The following figures are examples of different activities each volunteer performed. Each volunteer is of diverse profile and portrays contrasting characteristics. The Shimmer2 [BUR10] wearable sensors are visible on the volunteers left ankle, right wrist and chest. The sensors are attached using elastic straps. These sensors measure the motion of each body part such as acceleration, rate of turn and magnetic field orientation. The chest sensor captures 2-lead ECG measurements. Multiple wearable sensors lead to better capturing of body dynamics as well as sufficient capturing of multiple signals in real-time.

Figures 1-9: The following figure depicts a volunteer 'Standing still', 'Frontal elevation of arms' and 'Jogging.



The following figure depicts a volunteer 'Lying down', 'Cycling' and 'Waist bends forward'.



The following figure depicts a volunteer 'Knees bending (crouching), 'Running' and 'Sitting and relaxing'.

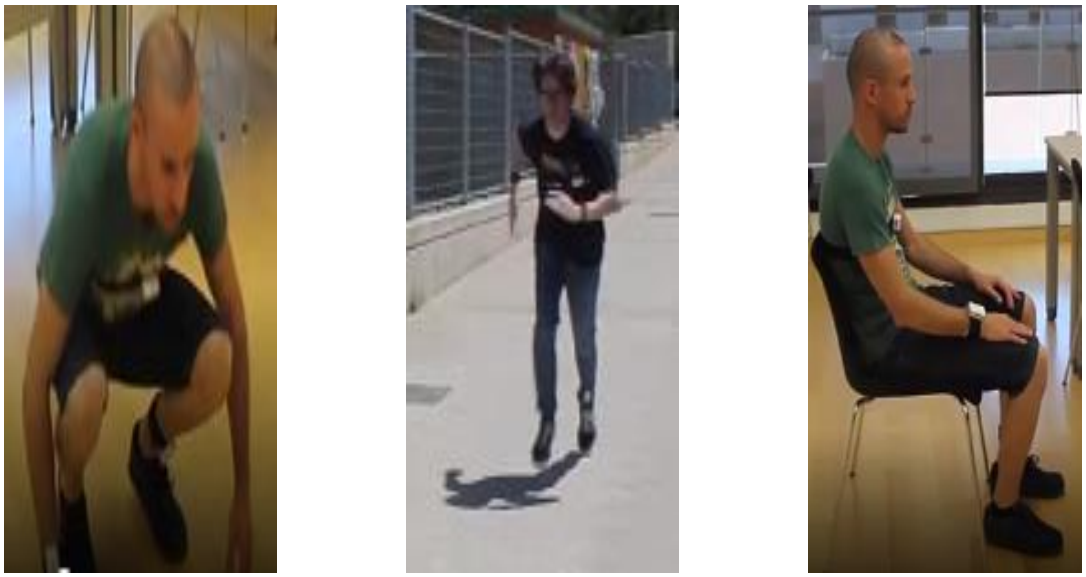

Figure 1 mhealth data activities [3]

1.7 Thesis Structure

This thesis has ten different chapters.

Chapter 2 presents an overview of Human Activity Recognition, the Internet of Medical Things, HAR Applications and wearable sensors.

Chapter 3 presents a detailed account of the State-of-the-Art of HAR and an in-depth analysis of machine learning algorithms.

Chapter 4 presents a detailed account of the MHEALTH dataset, outlining the experimental setup and attribute information.

Chapter 5 presents a discussion of each model in a step-by-step manner. This chapter presents an account of the each models data flow and network methodology process as well as tools and techniques used.

Chapter 6 presents the high-level approach of each model.

Chapter 7 outlines each models individual experiment, focusing on network architecture, hyperparameter setting and performance comparison.

Chapter 8 presents the results of each model, including accuracy and loss plots, confusion matrices, feature importance and a comparative analysis.

Chapter 9 evaluates the results presented in chapter 8.

Chapter 10 presents overall conclusion, key findings/contributions and results, limitations of work, and future research directions.

1.8 Contributions

The personal contributions entailed by this research are:

A comparative study of machine learning classification and data clustering approaches for sensor-based (human) activity recognition based on the MHEALTH dataset. Figures 25-39 compare the clustering approaches while Figures 40-56 compare the classification approaches, Chapter 8 presents these results.

Demonstrate that these approaches are applicable for activity recognition from on-body sensor data by the activity set presented by the MHEALTH dataset. Chapter 8 supports this given the high accuracy, precision, recall, F1-Score and Confusion Matrix values.

Demonstrate that accelerometer, gyroscope, magnetometer and electrocardiogram data fused together benefit the approaches. Feature importance analysis (Section 8.6) and Multisensor fusion analysis (Section 8.7) supports this from figures 57-67.

Demonstrate each approach is implemented on the raw, unstructured sensor data with limited pre-processing, eliminating the concept of engineering bias.

Exploratory analysis for data visualisation using T-distributed Stochastic Neighbour Embedding (t-SNE) and DBSCAN. Chapter 8, figures 25-39, present this exploratory analysis.

Performance of the approaches achieves greater performance measures than existing state-of-the-art baselines on the MHEALTH dataset. Section 8.8, tables 15-22, present a comparison against existing State-of-the-Art.

Detailed results, analysis and discussion section outlines key hyperparameters, which greatly influenced performance, which features prevailed as significant and which sensors proved the greatest impact. Chapter 9 presents detailed discussions.

The classification approaches are:

- XGBoost.
- Multilayer Perceptron.
- Convolutional Neural Network.
- Long Short-Term Memory (Recurrent Neural Network).
- ConvLSTM (CNN + LSTM Hybrid).
- Autoencoder by Random Forest.

The data clustering approaches are:

- DBSCAN.
- t-SNE.

2 Human Activity Recognition Background

The task of Human Activity Recognition dates back to the 1990s when researchers became intrigued in classifying body gesture and motion along with aiming to predict states of action or behaviour. It all began with the use of the accelerometer in the early 1990s when research involving advanced technologies opened up the gateway to wearing small-scale, light, cost-effective Micro-Electro-Mechanical Systems (MEMS) inertial sensors. Research into the field of Human Activity Recognition proceeded to increase greatly.

Outline of the Background and State-of-the-Art Review Chapters:

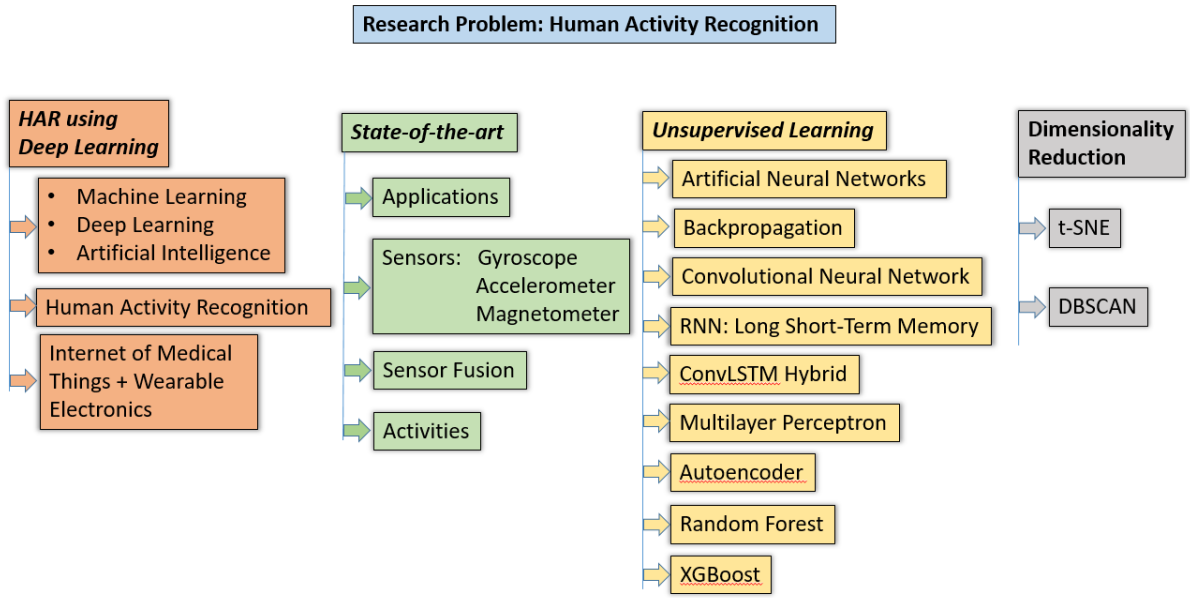

Figure 2 Human Activity Recognition Research Problem Overview

The following chapter provides a detailed account of background knowledge related to Human Activity Recognition. The chapter begins by outlining a brief introduction to artificial intelligence, machine learning and deep learning. It then proceeds to examine HAR along with the Internet of Medical Things and Wearable electronics. The chapter concludes by providing an overview of the state-of-the-art in terms of HAR applications, commonly used sensors in HAR, sensor fusion and common HAR activities.

This Chapter structure is as follows:

- Section 2.1: Artificial Intelligence, Machine Learning and Deep Learning.
- Section 2.2: Human Activity Recognition.
- Section 2.3: Internet of Medical Things (IoMT) + Wearable Electronics.
- Section 2.4: Application of Human Activity Recognition.

- Section 2.5: Wearable Sensors and Sensor Fusion.
- Section 2.6: Activities in HAR.

2.1 Artificial Intelligence

Artificial intelligence is one of the most talked about topics in the world today. Machine learning is a core topic in the artificial intelligence field. As seen in [4], countries are integrating AI and ML into everyday activities to drive economic growth. The benefits of AI along with the problems it addresses are endless. The term AI is not definitive as the field is so diverse.

AI aims to mimic the human brain. It aims to mimic how us humans think, act, move and speak. AI researchers have not discovered AI that can act rationally and process information that a living conscious can. The future of AI will evolve around creating a structure that is similar to the human reasoning, human acting and task performance processes. With technological advancements continuously occurring on a daily basis, it is only a matter of time before this happens.

Machine Learning

Machine learning is the ability of a computer or machine to think and act like a human. ML implementation can solve a variety of problems. As demonstrated in [5], it eliminates the need for recurrent needless testing and repeated evaluation procedures as the 'machine' consistently learns. The solutions are often insightful and a human alone would not be able to come to certain conclusions that machine learning would prevail. Machine learning can predict specific problems such as human activity recognition, stock market prediction and facial recognition. Machine learning has the ability to conduct analyse on data and learn from the data while making decisions (predictions) based on the learning process.

Deep learning

In recent years, deep learning has emerged as a new area of machine learning research [6]. Deep learnings ability to process raw data, no matter how complex it is, is much faster and efficient. For example, in image classification, the purpose is to distinguish between a penguin and a polar bear. As demonstrated in [7], using machine learning, the network would begin by extracting pixel values from all the relevant pictures. These pixel values would need to be processed and stored in a vector. The vector feeds into the machine learning network. The process is long, complex and takes time.

Deep learning significantly reduces process complexity and processing time. The raw-unstructured data for distinguishing between a polar bear and penguin feeds directly

into the deep learning network, extracts the relevant pixel values from each image and performs classification.

Deep Learning Architecture

A deep learning network consists of a group of connected neurons. A deep learning network consists of three layers, which are the input layer, the output layer and the hidden layers. These three layers have their own function in a deep learning network environment. The main question when building a DL architecture is, 'what do you want to do?'. This greatly affects the number of neurons applicable in each layer, the DL structure along with specific network requirements. A deep learning network is 'deep' because of the number of hidden layers it obtains. The 'deeper' the network, the more hidden layers. More layers can often lead to better performance, but not every time. A DL network allows data to feed into the network in order to try to predict a task, or classify a task for the user.

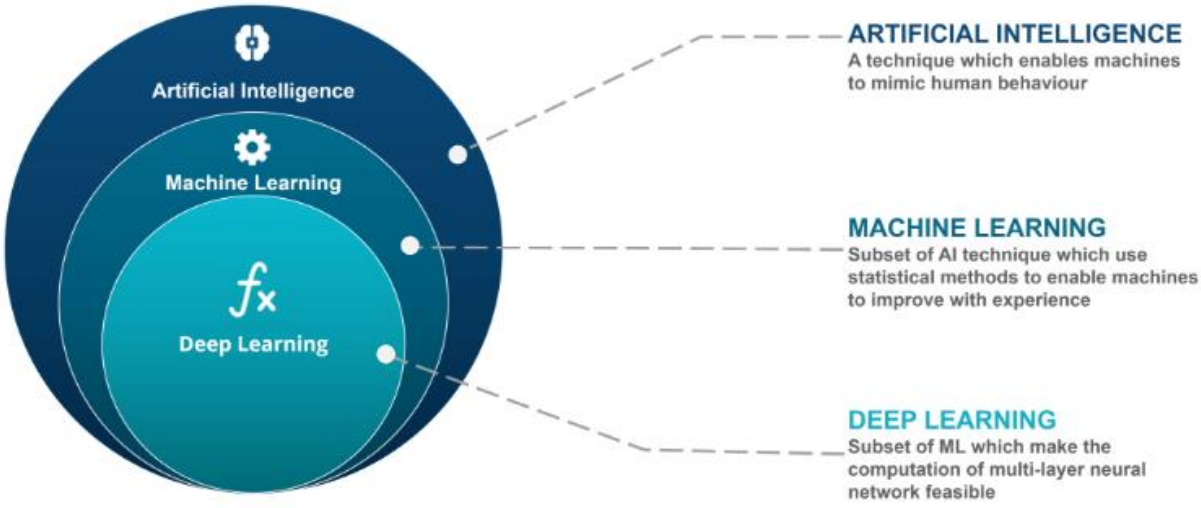

Figure 3 Artificial Intelligence [8]

2.2 Human Activity Recognition

Human Activity Recognition (HAR) using wearable electronics entails recognising volunteers physical movements (activities) by analysing data generated from on-body wearable sensors. These activities identify as Activities of Daily Living (ADL). As mentioned in [9], ADL's involves one's self and body, with specific emphasis on mobility. For example, walking, running, jogging or standing still are examples of ADL's.

Fully developed HAR systems continuously monitor human actions. The implementation of a HAR system in medical, entertainment and manufacturing settings can yield great benefits. Areas such as remote patient monitoring in medical settings

and sporting injury prevention have witnessed how this continuous monitoring can lead HAR systems to change lives.

Increased computational ability along with the development of Deep Learning, Artificial Intelligence, Data Analytics and Machine Learning has led to a surge in interest in HAR systems in recent years. The classification of movements using wearable sensors is becoming a vital part in reducing the cost of companies, organisations, manufacturers and healthcare.

Figure 4 Human Activity Recognition [10]

HAR systems present psychological benefits to individuals also. The reassurance of knowing that your heart rate, blood pressure and level of oxygen is normal leads to monitored individuals feeling calm and a peace of mind. It reduces stress levels. The wearable devices widely used by consumers today are Fitbits, Microsoft Bands and Apple Watches. These devices actively monitor individuals' health and exercise habits to ensure the wearer is healthy. These HAR systems monitor mobility, while leading to psychological benefits.

2.3 Internet of Medical Things (IoMT)

The internet of Things (IoT) is a term used to describe a connected network of physical devices. These devices utilise built in sensors that gather data about the connected devices and output the data to a central, cloud-based computing resource. Kevin Ashton is a consumer sensor expert who first coined the phrase 'the Internet of Things'. Bilal [11] describes it as a complex network that connects objects in the physical world to the Internet via sensors. Connectivity and sensor size has contributed greatly to the rise of IoT.

The Internet of Things (IoT) is essentially a large array of connected devices. These connected devices produce a subsequent array of data from the sensors they contain. Sensor data can develop new services, leading to improved efficiency thus forming an overall better, more reliable business model. One thing that we know for sure, before the world slowly evolves around the IoT, is that all of these connected devices will dramatically change how we interact with computers, nothing will be the same again. Wireless data connectivity allows the immediate transfer of data over a network. We are going to be living in a world where our physical and virtual worlds are constantly connected; they will eventually merge into one environment. The billion-node network that is the Internet today will be a shadow of what is to come in the future.

Sensor fabrication methods have led to the reduced weight, size and cost of the sensor component and system. Sensor size is continuously decreasing and they are still in their early growth development stage. Technological advancements such as the development of microelectromechanical systems (MEMS) have developed sensors so small that they can be placed anywhere on the body [12].

The widespread use and popularity of wearable electronics offer a large variety of applications in the healthcare arena. With continuous monitoring of body activity and vital signs, wearables could possibly be lifesaving. These types of applications would significantly improve patients' lives and open up possibilities for alternative treatments.

IoT will have a profound effect on everyday life, the economy and electronic devices in the coming years. Organisations and research institutions have projected the effect IoT will have on the technological world. Machina Research [13] reported in April 2015 that the value of 'the Internet of Things was around $900 billion in 2014, and they predicted it will rise to $4.3 trillion by 2024'. Frontier Economics [14] predict that' 10% rise in IoT investment would result in an increase in GDP of $370bn in Germany and $2.26trn in the US over a 15-year period (2018-2032)'. These estimations are an accurate prediction of the connected, data-oriented world that we will live in. The rise of devices connected to networks is so large it is out of our control to monitor. With large volumes of data produced every day, IoT is a significant contributor to the era of big data.

2.4 Sensors

Human activity recognition is applicable to a range of different types of sensors. These sensors come in various shapes and sizes while the data each one generates in terms of magnitude differs greatly. Accelerometers, magnetometers, gyroscopes, audio sensors, RFID tags and smartphone sensors are the most commonly used sensors in this domain.

Accelerometers

An accelerometer is an electromechanical device that measures changes in a gravitational pull. It measures sheer acceleration forces. The force of gravity can be static or dynamic. A moving object, falling downwards, accelerates to due gravity. Acceleration is essentially a reading of an increase in speed and range of motion.

Accelerometers allow for the calculation of inertial measurements in a fast and easy manner. Accelerometers characteristics make them highly suitable for human activity recognition. They are precise, highly accurate, are small and they are suitable for harsh environments ranging from cold to hot temperatures.

As seen in [15], Accelerometers can be uniaxial, biaxial or triaxial. Uniaxial is concerned with the x-axis. Biaxial is concerned with x and y-axis while triaxial is concerned with x, y and z-axis. Due to the range of motion involved in HAR, triaxial accelerometers are the most sought after accelerometers in HAR. Triaxial accelerometers measure the range of motion on three levels (x,y,z). They allow the sufficient storage of data for analysis. Biaxial and Uniaxial accelerometers measure movement of one and two planes respectively; therefore, triaxial accelerometers measure more energy, movements and generate more data. As demonstrated in [16], these instruments can distinguish differences in activity levels between individuals to assess the effect of interventions on physical activity within individuals.

Accelerometers are the most popular sensor due to the continuous output it produces. Garcia-Ceja, Brena, Carrasco-Jimenez and Garrido [47] reports on how the data generated from accelerometers based from a wristwatch can classify long-term activities such as running or jogging for a period. Accelerometers are preferred as they lead to precise results in classifying physical activities most of the time. In obtrusive environment conditions, accelerometer data is rarely affected. Gupta and Dallas [18] present another application in which accelerometer data is used in classifying activities. This paper presents a highly precise physical activity recognition procedure that incorporates feature selection and utilises the diverse data generated from a waist mounted triaxial accelerometer. Salarian, Russmann, Vingerhoets, Burkhard, and Aminian [19] present a framework for monitoring patients with Parkinson's disease. The framework analyses accelerometer data extracted from three inertial sensors attached to different parts of the volunteers' body.

Gyroscopes

A Gyroscope is a device that detects the rotation around a particular axis of an object. Gyroscopes contain a rotational wheel or a rotational beam of light to enable this detection to occur. A gyroscope has a built in guidance system that allows it to gravitationally rotate while detecting any movement or range of motion as this gravitational rotation occurs.

A gyroscope also measure the range of motion on three levels: the x, y and z-axis. As mentioned before, this is particularly effect in HAR as more data generated leads to more analysis. This leads to concise, definitive results allowing more variables to factor into the equation.

Another approach for analysing sensor data for activity recognition is analysing gyroscope data. Hassan, Uddin, Mohamed and Almogren [20] present a sensor-based approach for classifying activities by extracting data from a gyroscope and accelerometer simultaneously. The Feature extraction method extracts the important features from the raw, unstructured data in order to classify the activity that the human performs. Ronao and Cho [21] reports how data generated from smartphone using accelerometers and gyroscopes can lead to the successful classification of activities while simultaneously extracting highly effective features from the extracted, unstructured data. Mekruksavanich, Hnoohom and Jitpattanakul [22] conduct a study to identify if accelerometer and gyroscope data combined leads to an increase in HAR accuracy. Results proved conclusive. The results reveal that the accelerometer and gyroscope are complementary in their application and using them in combination thus improves the accuracy of recognition.

Magnetometers

Bai and Bai [23] defines a magnetometer as a scientific instrument used to measure the strength and direction of the magnetic field near the instrument. The earth has a giant magnetic field around it, magnetism varies along this magnetic field and a magnetometer has the ability to measure this magnetic field while generating Earth-magnetic field data.

It is essential when conducting research on HAR that a tri-axis magnetometer gathers data due to the sheer volume of data it generates. The magnetometer arrays that it produces when measuring the magnetic field in the vicinity is comparable to the precise HAR classified movement in question.

Altun and Barshan [24] reports on classifying physical activities that volunteers' perform using magnetic sensors. A triaxial gyroscope, a triaxial accelerometer and a triaxial magnetometer gathers mobile data about the volunteer. The eight volunteers performed daily and sports activities to while the data was gathered, deep learning techniques performed as well as cross-validation techniques.

Accelerometers combined with gyroscopes has been studied at [25], [26] and [27] while Accelerometer, Gyroscope and Magnetometer data combined to conduct HAR can be seen at [28] and [29]. The three combined together are excellent tools for measuring body dynamics

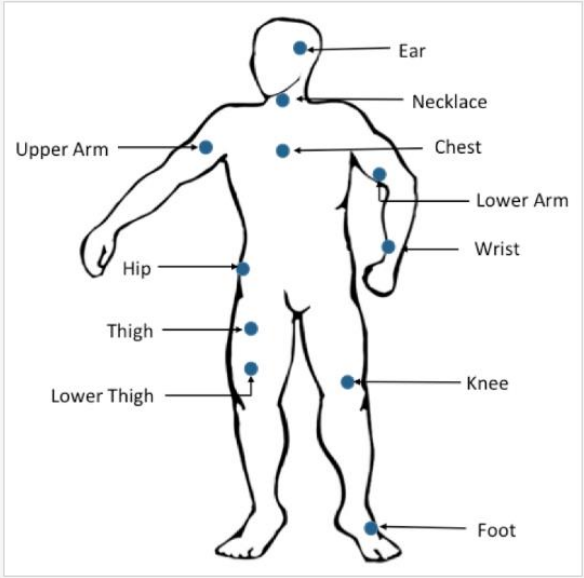

Figure 5 Graphical representation of wearable sensor placement [30]

Electrocardiogram

An electrocardiogram measures the speed at which your heart is racing. It identifies if it is regular or irregular and it is widely used to identify heart problems and hearty diseases. Small 'tickers' allow for the recording of electromagnetic waves coming to and from the heart once attached to the outer skin of your chest, arms and legs.

The Electrocardiogram is effective for: [31]

- Heart monitoring.
- Checking for various arrhythmias on the heart.
- Looking at the effects of exercise on the heart.

2.5 Senor Data Fusion

As seen in [32], Sensor fusion is the process of merging data from multiple sensors such that to reduce the amount of uncertainty that may be involved in a robot navigation motion or task performing.

It is the combination of multiple sensors to create resulting data leading to distinctive results and precise conclusions. There is a wide range of benefits associate with sensor fusion such as reliability, accuracy, completeness and the applicability of two or more sensor types. It saves time, money and effort.

Firstly, you begin with your specific environment. For example, in a room conducting activities for the research surrounding HAR. Secondly, three sensors placed on specific areas on the human body.

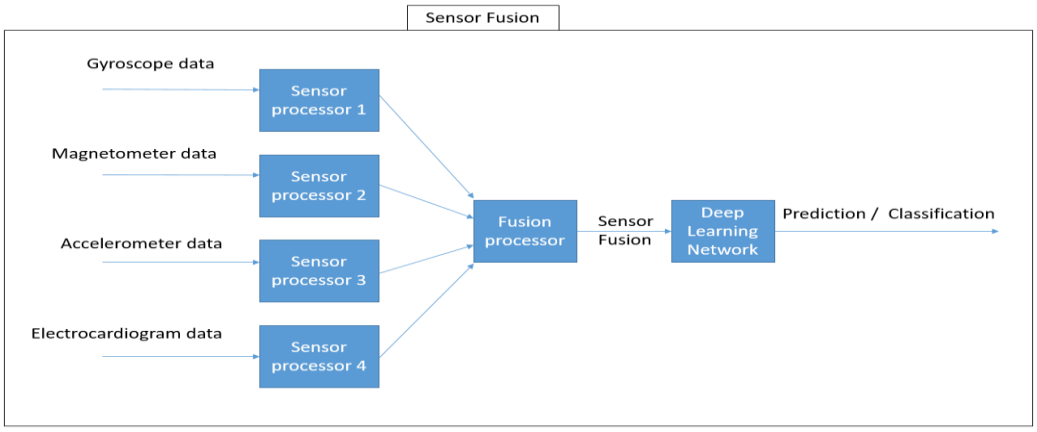

Figure 6 Sensor Fusion Architecture

Next, the fusion occurs. There are three types of sensor fusion; competitive fusion, complementary fusion and cooperative fusion. Competitive fusion, one that aims to distinguish and outshine others. A real-world example of this would be a presidential election. Fusion measurements represent a contrasting instance such as candidates competing in an election. As seen in [33], they both deliver independent measurements of the same property.

Complementary fusion is associated with two sensors essentially helping each other out in order to generate data and identify meaningful insights and relationships between the sensors. They are not solely dependent on one another but combining the two sensors gives a broader, more definitive conclusion rather than operating independently. A real-world example would be a footballer kicking a football; this would generate a range of data ranging from strength of the kick to range of motion of the football.

Cooperative fusion entails two sensors working together to solve the problem at hand and generate data. They would not have the power to generate data independently, these two sensors need to fuse together in order to derive information. They cannot operate alone.

Due to these three sensors fusing together competitively, complementary and cooperatively, the resulting data generated allows thorough data analysis to generate insights about relationships between the sensors.

2.6 Human Activity Recognition Applications

This section outlines different settings in which HAR can influence using wearable devices. This section begins with detailing how activity recognition has benefited healthcare and remote patient monitoring in the past. Other areas that activity recognition has benefited in todays' world include industrial applications and applications for entertainment and gaming.

Healthcare and remote patient monitoring

Activity recognition is a highly active research area in todays' society. It is becoming a particularly massive part of healthcare complementing remote patient monitoring. In the past one hundred years, the average life expectancy has seen a complete turnaround as it has never been higher. This has led to a huge increase in the percentage of the population over the age of 75. Healthcare systems are feeling the pressure as the amount of elderly people needing assistance, as they get older, is rapidly increasing. Technological advancements, along with improvements in healthcare, is the only solution to this problem. It is vital that technology advances with this fast, paced changing environment to ensure the elderly live sustainable lives and limit the need for constant monitoring.

Fall detection using wearable sensors is an area that studies remotely monitoring the elderly. Elderly people are prone to falling; it can be harmful to their life and can limit their ability to call for help. Fall detection software can enhance the security and safety of elderly people. It can give them peace of mind when home alone. Foroughi, Aski, and Pourezza [34] proposes a fall detection system that successfully identifies if a person 'backwards fall', 'frontwards fall', 'sideways fall' or 'lies down'. Creating such a system is vital in monitoring elderly while ensuring they are safe at all times. Williams, Xie, and Ou [35] propose an approach where multiple sensors on the elderly persons' body analyse and determine if there is abnormal activity, leading to trigger an alarm in the fall detection application. Sposaro and Tyson [36] further described a fall monitoring application detecting fall monitoring and response.

Fall detection using video surveillance is another widely discussed topic in activity recognition. The only concern with video surveillance is privacy, overcoming this obstacle ensures health comes first and ensure health threats are eliminated. Nait-Charif and. McKenna [37] present a fall detection application by combining multiple video datasets to predict the type of fall occurring. It detects the type of fall of the elderly person in their home environment. Nait-Charif et al. [37] further discusses a camera based surveillance system. Unusual activity, such as a fall occurring, triggers an alarm in their home environment.

Alzheimer's is one of the most common age-related conditions. Successful identification of an elderly person developing symptoms of Alzheimer's can prevent those performing actions that could prove fatal. Alzheimer's is a condition, which worsens over time. Building an application to detect early symptoms of Alzheimer would be very difficult. Research, detection and the generation of data would take years. This is why elderly people should become familiar with the application relatively early to ensure the application integrates with the persons' daily living, activity habits and eating habits. Cheng and Zhuanga [38] present a Bluetooth enabled in-home patient monitoring system, which aims to detect Alzheimer's in elderly people. Lopez et al.

[39] presents a framework, which aims to detect Alzheimer's at an early stage, but stresses that this field is a work in progress and will take time to perfect.

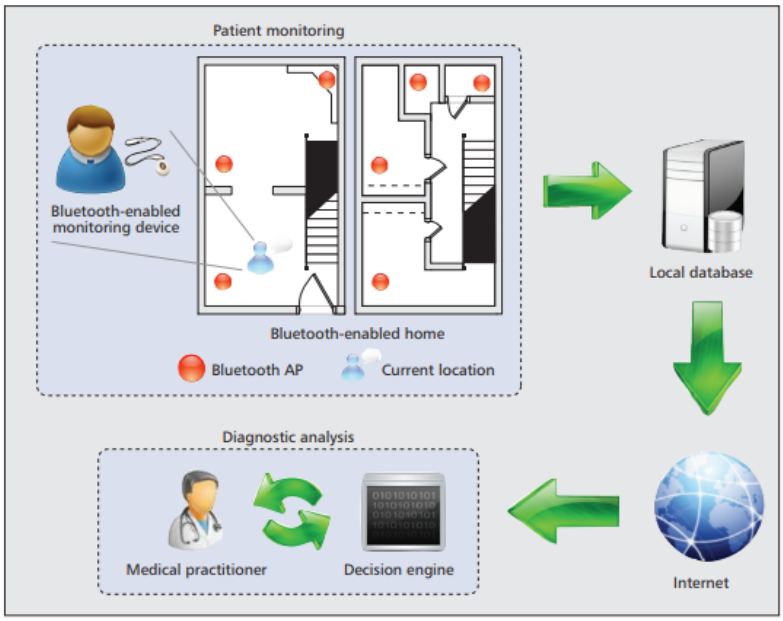

Figure 7 Typical Remote Patient Monitoring System [40]

Modern Working Environments

Activity recognition applications can increase productivity, assist workers completing assignments and prevent dangerous accidents occurring in an industrial setting. Activity recognition applications supporting wearable sensors can increase productivity by enhancing communication along with providing access to data/procedures that will reduce processing time. These applications provide safety in modern environments due to successful communication between several parties.

Modern environments can benefit greatly from the continuous monitoring of workers current activities. The data generated from sensors in these activity recognition applications allow information to be analysed and pro-actively used. Modern applications, such as in the aircraft industry, yield these benefits. Aircraft maintenance procedures with the aid of activity recognition applications using wearable sensors continue to develop. Nicolai, Sindt, Witt, Reimerdes, and Kenn [41] employs a wearable computing system with the aim of reducing aircraft maintenance. The system incorporates management techniques such as visual assisted maintenance techniques, deployment of electronic logbook as well as aircraft industry manuals. Lampe, Strassner and Fleisch [42] present a ubiquitous computing environment for Aircraft Maintenance to ensure maintenance time is minimised and resources such as staff, tools and techniques are efficiently used.

Maurtua, Kirisci, Stiefmeier, Sbodio, Witt [43] present an activity recognition prototype using wearable electronics that aids training activities in automotive production. Results showed that the prototype was highly effective in allowing automotive production to be flexible. Workers guided by the prototype, ensure automotive production is maximised compared to operating alone. Monitoring the workers procedures and actions led the prototype guided the worker in performing appropriate tasks for error handling.

Aleksy and Rissanen [44] reports on how wearable electronics can yield benefits in modern working environments. A detailed account of how efficiency improves in various sectors with constant interaction between the user and the device is given. The study presents a case study where processes in an industrial plant aided by wearable technology to analyse if productivity increases.

Entertainment and Games

Wearable electronics using physical activity recognition are widely used in the entertainment sector. The development of this area has increased greatly in recent years mainly because there is little privacy concerns and classification accuracy does not have to be exact, for example as in remote patient monitoring.

Cheok et al. [45] presents a wearable electronic entertainment system, which is a simulator evolving around a human Pacman entertainment system. The system is a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. Cheok et al. [45] builds an architecture that is capable of monitoring human motion with the data generated from body worn inertial sensors. Building an entertainment system that clearly mimics the real and virtual world is difficult, [45] successfully accomplishes this.

Tobita and Kuzi [46] presents an unusual entertainment application. The wig-based wearable computing device enhances communication and provides entertainment to users. This obscure, natural looking device employs wearable sensors to allow to parties to communicate with each other. SmartWig is one of the first wearable electronic applications for effective communication in the entertainment sector.

Cheok et al. [45] presents a wearable electronics entertainment system called 'Game-City', an almost virtual reality computer environment, which combines an interactive and physical mixed reality game-space. Real-time sensor tracking creates the 'Game-City' where social encounters occur and players can roam freely throughout the computer reality.

2.7 Activities

When presented with a human activity recognition problem, choosing the correct sensor to complement the working environment or application is important. The amount of different types of sensors to choose from is endless, with advanced sensors developed

each year. Continuous research in the activity recognition field has led more companies trying to reap benefits of predicting activities to improve communication and improve productivity. This section discusses the different types of activities performed for activity recognition.

Zhu, Xu, Guo, Liu, and Wu [48] classify physical exercise activities such as running, jogging, standing still and powerwalking using body-worn inertial sensors in related work. These activities, when performed, generate a specific type of range in body motion with the acceleration measured being relatively similar when individuals of different characteristics perform the activity.

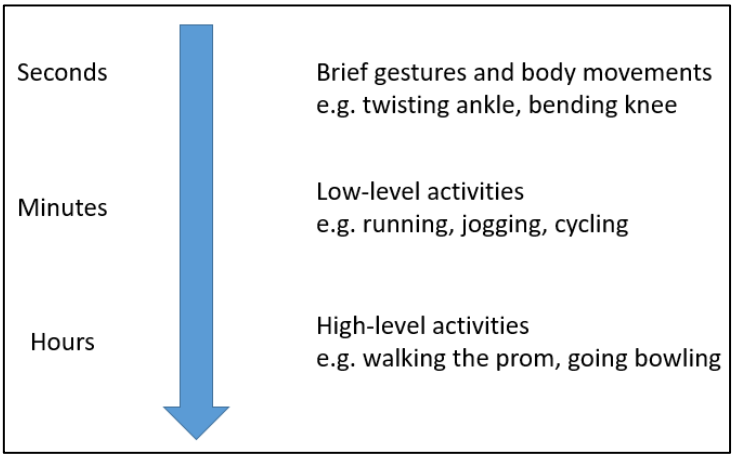

Figure 8 High to Low Level Activities

The related work present in [49] classifies a high number of physical exercises. Regular human physical activities such as walking, sitting down, lying down, walking upstairs or standing still are studied. ADL's, which are activities of daily living, such as dressing yourself, taking a bath or going to the toilet were performed with an aim to be classified. Weekly meal preparation, using your laptop or going to the shop are instrumental activities of daily living (IADL). Sports activities such as cycling, running and rowing mentioned in this related work along with activities that take little time to perform such as opening a closet.

Instrumental activities of daily living (IADLs) proposed by [50], consist of bathing or showering, using the telephone, preparing meals, doing housekeeping, doing laundry and managing medications. The sensor fusion approach of combining the data generated from gyroscopes, accelerometers and magnetometers proposed in related work [50], prove that recognising these instrumental activities of daily living from sensors is a difficult task. Doing laundry and doing housekeeping is difficult to detect as separate activities as these activities could generate the same range of motion. The speed of performing the activity and the ability of how exact the volunteer also has an impact on how the activity is classified. For example, if they volunteer mixed up putting clothes

in the dryer instead of the washing machine, would the sensors detect this. Quality of performance is a huge factor in activity recognition.

Research in human activity recognition ranges from analysing low-level activities to high-level activities. In each diverse environment ranging from remote patient monitoring to clinicians, gathering data and analysing it is the most important aspects. The next section will focus on unsupervised learning for activity recognition in previous work.

2.8 Conclusion

This chapter has presented an overview of Human Activity Recognition, the Internet of Medical Things, common sensors used, common activities performed and real-life applications of HAR. The information above divides into seven different sections to give the reader a broad understanding of how diverse the HAR field is.

The first section deals with Artificial intelligence, Machine learning and deep learning, all of which revolve about the concept of classifying Human Activity Recognition. The second section deals with Human Activity Recognition; it provides an overview along with informing the reader about the benefits of activity recognition. The third section dives into the Internet of Medical Things, explaining how it is rapidly growing along with some key facts about how the IoMT can benefit todays' economy. The fourth section presents an account of common sensors used in HAR such as gyroscopes, accelerometers and magnetometers, all of which are implemented on the MHEALTH dataset. The process of sensor fusion is then further discussed. Section 5 discusses common HAR applications, detailing its profound impact in the health sector. Section 6 finalises the chapter providing an overview of common HAR activities.

This Chapter provides a solid foundation for the next section of this research. The next chapter leads into the State-of-the-Art of machine learning algorithms in Human Activity Recognition. The next chapter gives an in-depth analysis of Convolutional Neural Networks, Recurrent Neural Networks, CNN and LSTM Hybrid, Multilayer Perceptron, Random Forests, Autoencoders and Extreme Gradient Boosting along with their applicability and related work in activity recognition.

3 State-of-the-Art Review

The following section gives an account of machine learning methods and deep learning methods that have been previously been studied for human activity recognition. The mathematical concept, layers, hyperparameters, architecture and related work in the field of HAR is presented for each algorithm. The chapter concludes with an overview of dimensionality reduction, providing an overview of data clustering along with two commonly used algorithms in HAR, DBSCAN and t-SNE.

Nearly every machine learning and deep learning method has been implemented for activity recognition. It is difficult to find a standard activity recognition approach to suit all HAR datasets. State-of-the-Art proposed approaches usually fit the dataset perfectly, leading to outstanding results. If you were to use 'Approach A' on 'Dataset B', the approach may not work as expected as different features are present, different data recorded along with different activities.

This section provides a summary of the most well-known machine and deep learning techniques. Related work in Human Activity Recognition complements each technique.

- Section 3.1 provides a summary of Artificial Neural Networks.
- Section 3.2 provides a summary of Backpropagation.
- Section 3.3 provides a summary of Convolutional Neural Networks.
- Section 3.4 provides a summary of Recurrent Neural Networks and LSTMs.
- Section 3.5 provides a summary of. ConvLSTM.
- Section 3.6 provides a summary of. Multilayer Perceptron
- Section 3.7 provides a summary of Autoencoders.
- Section 3.8 provides a summary of Random Forests.
- Section 3.9 provides a summary of XGBoost.
- Section 3.10 provides a summary of Dimensionality Reduction.
- Section 3.11 provides a summary of Density-based spatial clustering of applications with noise (DBSCAN).
- Section 3.12 provides a summary of t-Distributed Stochastic Neighbour Embedding (t-SNE).

3.1 Artificial Neural Networks

An Artificial Neural Network is a sort of computation, mathematical model with the same functions and characteristics of the biological neural network. As seen in [51], ANNs aim to create a model of a system without any predefined relation between input and output. A neural network usually consists of three layers: input layer, hidden layer and output layer. A neural network can have many hidden layers in order to learn more

about the data. Neural networks are excellent mathematical tools for measuring relationships between pieces of data. It enables one to explore patterns in data leading to insightful discoveries between inputs and outputs.

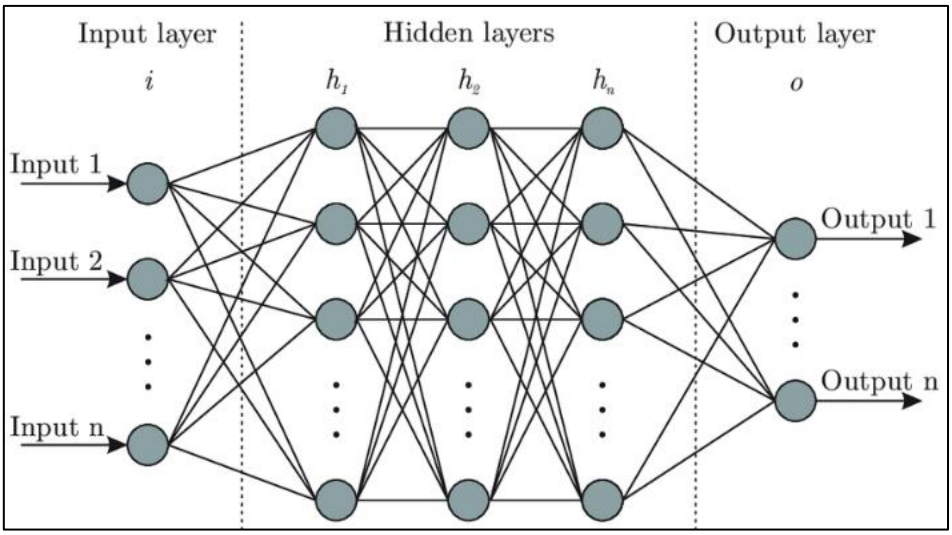

Figure 9 Architecture of a Neural Network [52]

The problems that neural networks have the ability to solve are quite complex. They have the ability to solve the most complex of problems that simple computer systems would not be able to solve. Neural Networks have quite a complex algorithm and it takes time to learn, but once understood they possess a simple structure and self-organising nature, which allows them to be applied to almost any data-solving problem.

As demonstrated in [53] and [54], online shops such as EBay and ASOS train neural networks on customer behaviour data to generate insights about customer behaviour. Theses online stores aim to predict what their customers will buy next. They often have a specific section on the users' current page named 'Similar things to X' or 'Things to go with X'. It aims to suggest complementary goods. An Artificial Neural Networks contains a range of neurons or nodes. These neurons contain weighted connections and aim to learn from other neurons in the network, in a learning process. Activation functions define the output value of each neuron, based on the value of the input value.

Firstly, the input layer receives the data entry. This piece of data is unique so it has its own unique range of characteristics or attribute values. The output layer produces the output of the network. The sum of all the incoming neurons combines to generate the value of the input neuron. The input neuron multiplies with weights on the interconnected relationship between the respective nodes.

There are two types of Neural Networks: Feedforward Networks and Recurrent Networks.

Feedforward neural networks are any type of network that run straight through from the input to output layer. It does not gain feedback and cannot back track. The flow from input to output does not stop.

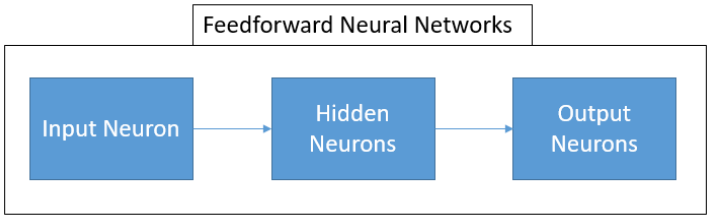

Figure 10 Feedforward Neural Network

Recurrent Networks are any type of network that allow the network to generate feedback. RNN's are great for the learning process as they enable neurons to learn and back track as they pass through the networks. They can back track as far as the earliest stages.

3.2 Backpropagation

Backpropagation in neural networks entails allowing for the readjustment of weights in the network backwards through the neural net due to an error occurring. When the network makes a prediction about a set of input values and the prediction is incorrect, the output value and prediction value is compared leading to the error being calculated. This error allows for the readjustment of the weights that link to the output neurons of the network. The error crawls through the network, adjusting weights as it goes along. The next step is to train the neural network. Parameters must be set to allow for an efficient learning process:

The learning rate identifies the speed of the learning process for each neuron. The learning rates value can be computed between 0 and 1. This value multiplies with the error that each outputted value produces. In other words, a learning rate of 0 would lead to no adaptation at all. The learning process will not move forward and benefit the network unless there is careful consideration to set the learning rate. If the learning rate is set too high, it complicates the process leading to a poor model with no optimal values. If the value is too low, there will not be enough muscle mass to generate the network to form a new optimisation. As depicted in [55], this can lead the weights stuck in local maxima. Momentum is another vital neural network parameter. It is utilised by evening out the optimisation process. It uses a portion of the previous weight change and adds it to the new current weight change. The combination of momentum and learning rate is the key to a successful learning process for the networks neurons.

3.3 Convolutional Neural Network

In a neural network, neurons learn from each other as they are fully connected. Neurons in convolutional neural networks connect to a fraction of the neurons that are in the previous layer. This layer is the *receptive field* as seen in related work [56]. Neurons in convolutional neural networks have three dimensions. These dimensions are width, height and depth.

Architecture

CNNs have a unique architecture. CNNs contain many sequential layers such as the convolutional layer, pooling layer, Rectified Linear Unit layer, normalisation layer and fully connected layer.

Convolutional Layer

The convolutional layer is the focal point of a CNN. The convolutional layers main objective is to extract high-level features about the data.

MaxPooling Layer

Pooling layers allow for the reduction in number of parameters in the neural network. It reduces the number of descriptive parameters used to explain the structure of the neural network. It essentially avoids overfitting as it reduces the spatial size of the network. Training a neural network takes a great amount of time. Pooling ensures the amount of computations needed to train the network are minimised. It ensures the classification task runs smoothly.

ReLU Layer

Relations in data is often non-linear. The ReLU layer ensures there is an increase in non-linearity. It applies the following element wise non-saturating activation function to ensure that a neural network can build the non-linear relation between data points. If there were no ReLU layer, a neural network would not be able to classify non-linear data points. When compared to tanh and sigmoid, the ReLU layer prevails in terms of speed, accuracy and precision. The width, height and depth of the neural network, also known as the spatial size, is left unchanged.

Dropout

Overfitting is a common problem neural networks face when training data. The Dropout regularisation technique successfully prevents overfitting. Fully connected layers in a neural network have many variations. Dropout identifies the nodes in a specific layer and removes them. A definitive probability, p, is then applied to the layer. The training process removes nodes linked with the removed layer. As seen in [57], after training, these nodes are placed back into the neural network and assigned their original value (weight). This, in turn, boosts the performance of the neural network. The validation training set benefits the most from this during the deployment of the model.

Optimiser: Adam

Adam is a gradient-based optimiser. It is straightforward, simple to implement and is computationally inexpensive. It is suited to solving classification problems related to human activity recognition. The data involved in HAR is normally relatively large, leading to Adam to be a perfect fit. The hyper-parameters require little or no tuning, which is why Adam is the most common optimiser in convolutional neural networks.

Softmax Activation Function

The softmax activation function usually set in the output layer and loss layer. This is usually the final layer in the neural network, before the output layer presents the result. The following equation is a detailed representation of the softmax activation function. The layers described above make up the full architecture of a convolutional neural network.

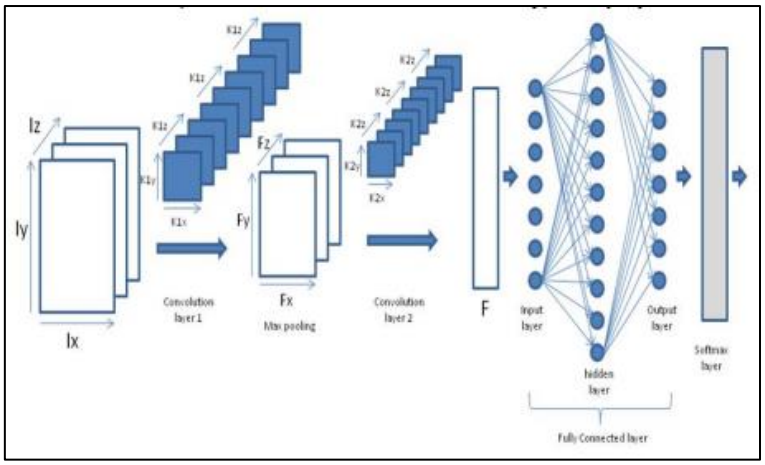

Figure 11 Typical architecture of a CNN [58]

The equation below incorporates all the layers and functions mentioned above to represent the typical architecture of a CNN.

$$Input \rightarrow [[Conv \rightarrow ReLU] * N \rightarrow Pool?] * M \rightarrow [FC \rightarrow ReLU] * K \rightarrow FC$$

Figure 12 CNN Equation

Convolutional Neural Networks in HAR:

Jiang and Yin [59] compares multiple deep convolutional neural network (DCNN) architectures using accelerometer and gyroscope data in classifying activities. Jiang et al. [59] performs analysis on the UCI MHEALTH dataset (UCI) [60], USC-SIPI Human Activity Dataset (USC) [61] and a dataset compiled by the fusion of Smartphone Motion Sensors (SHO) [62]. Jiang et al. [59] compared performances to identify the architecture, which achieved highest accuracy, recall and precision along with low computational cost. Jiang et al. [59] found that SHO achieved the highest accuracy, followed by USC, while UCI performing slightly lower.

Hammerla, Halloran2 and Plotz [63] compare multiple convolutional and recurrent approaches when using wearable sensors in classifying activities. Hammerla et al. [63] performs analysis on three datasets: The Opportunity dataset [64], Pamap2 dataset [65] and Daphneit Gair (DG) dataset. Hammerla et al. [63] conducts thousands of experiments to identify the substantial effect of altering hyperparameters. Performance evaluation indicated that the approaches achieved the highest accuracy on DG, lowest Root mean squared error and highest F1-Score, followed by Opportunity, while performance scores on Pamap2 were slightly lower.

Kim and Moon [66] compares the use of deep convolutional neural networks (DCNNs) for activity recognition in classifying activities. Kim et al. [66] also compare DCNNs for human detection. A Doppler radar gathers data, which produces velocity data when placed on a human or near a human. Kim et al. [66] found that the DCNN achieved an accuracy as high as 97.6% for human detection. They also found that human activity classification accuracy reached heights of up to 90.9%.

3.4 Recurrent Neural Network

Traditional Neural Networks have a major disadvantage of not being able to take into consideration the impact of previous events or previous information when analysing decisions for prediction of future outcomes. Recurrent Neural Networks counter this problem with the creation of Long Short-Term Memory cells. RNNs are neural networks that contain loops; these loops allow information to feed back through the

loop. These loops enable information to pass through stages of the neural network. Hence, the name recurrent. A key characteristic of an RNN is their ability to process information created in the past.

Vanishing Gradient Problem

The main problem associated with RNNs is the vanishing gradient problem. This problem arises due to a complication in the backpropagation algorithm during the training of the neural net. As demonstrated in [67], weights and biases update throughout each training iteration, they move in the direction of the gradient of the weight/bias values with respect to the loss function. In few DNNs, the gradient shrinks as we move backwards through more hidden layers. This leads to the neurons further back in these hidden layers struggling to learn. This is the vanishing gradient problem. If gradient is absent from the network, no weights are adjusted in the network. This leads to a reduction in error while the network is refusing to learn.

Long Short-Term Memory (LSTM)

LSTMs intended on tackling the vanishing gradient problem. The main difference between LSTMs and RNNs is LSTMs use of memory cells. Memory cells allow for the sufficient storage and sequential processing of data. Time is not restricted and the data does not disappear back into the network. It enables the development of relationships in the data, leading to insightful knowledge regarding the output to be analysed. Gating is at the core of LSTMs. Gating regarding LSTMs involves component wise multiplication of the input as seen in related work [68]. This leads to consistent updates in each data cell, due to the gating calculation that applies to each cell. The data must encounter the write, read and reset gates to process data correctly. The write gate is the input gate. The read gate is the output gate while the reset gate is the forget gate. LSTMs contain information in a gated cell, which is the key idea of these networks. LSTMs can add or delete information to the cell through the gates. These gates are composed of a sigmoid neural network layer and a pointwise multiplicative operator. Overall, an LSTM has three of these gates to control and protect the cell state information as seen in related work [69].

Architecture

The following diagram details the basic architecture of an LSTM network. The diagram explains the data flow through the memory cells in the network.

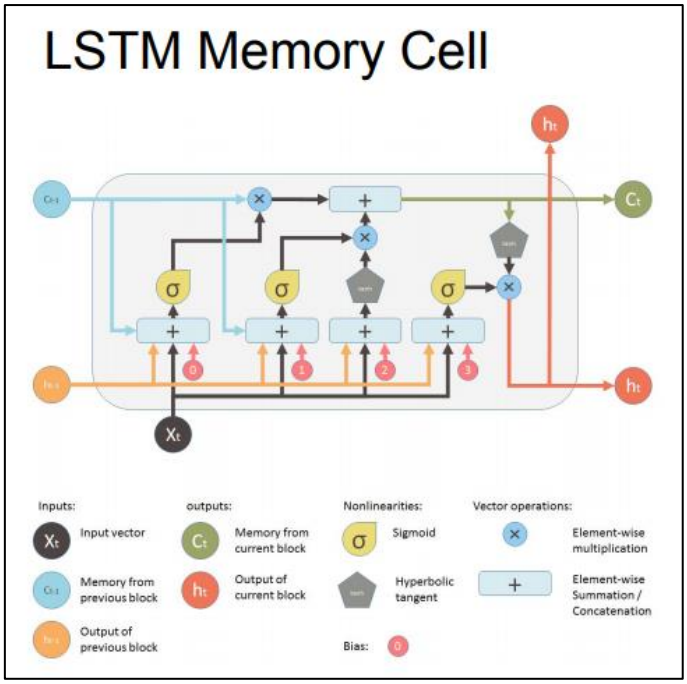

Figure 13 LSTM Memory Cell [70]

The above diagram provides a detailed account of an LSTM Memory cell.

The current input and previous state feeds into the cell as well as each of the three gates, which handles the flow of the input further.

The input shape is the input to the model we build which has the following parameters.

- Batch size.
- Number of time steps.
- Hidden layers.

The input feeds into the stacked LSTM cell layers while the output contains a SoftMax activation function applied on it. As seen in [71], the SoftMax function calculates the probability distribution of an event over 'n' different events e.g. this function will calculate the probabilities of each target class over all possible target classes. The output of this network is compared with the training data values and the respective error and gradient back propagation is performed, as demonstrated in related work [72].

Human Activity Recognition applications of Recurrent Neural Networks (LSTMs)

M.Inoue1, S.Inoue1and Nishida [73] compare various deep recurrent neural network architectures using raw accelerometer data to classify activities. Inouel [73] discusses the benefits of DRNN due to their compact, small architecture. DRNNs ability to

analyse data in real-time leads it to recognising a high percentage of activities. The best achieving recognition rate performed by the DRNN was 95.42%.

Du, W.Wang, and L.Wang [74] compare five deep RNN architectures in classifying human actions. The following datasets were analysed: the MSR Action3D Dataset [75], the Berkeley MHAD Dataset [76] and the Motion Capture Dataset HDM05 Dataset [77]. Du et al. [74] created a deep architecture from scratch to compare performance against the five deep RNN architectures. The proposed approach is a hierarchical RNN, for skeleton-based activity classification. The proposed method outperformed all five Deep RNN architectures. It achieved state-of-the-art performance with high computational efficiency. Du [74] discusses issues such as overfitting and computational efficiency.

Li et al. [78] also conducts experiments on 3D skeleton data. Li [78] compares deep Long Short-Term Memory (LSTM) networks to better capturing body dynamics, leading to successful classification of activity recognition. The datasets are in question during this paper: They created their own unique data by placing wearable sensors on volunteers and the public G3D dataset. Both datasets generated excellent performances. Performance was evaluated using accuracy, precision, recall, f1 score and confusion matrices.

3.5 CNN + LSTM Hybrid (ConvLSTM)

A convolutional LSTM (ConvLSTM) is a hybrid model combining the architecture of a CNN and LSTM together. ConvLSTMs are widely used to solve classification problems in [79] [80] [81]. A CNN allows sequences of data to pass through its convolutional layers. The number of convolutional layers present in a CNN can vary. The outputted result is essentially a dense network. This resulting network is a ConvLSTM that is fully connected.

Convolutional LSTM cells

The data passes forward through each convolutional layer in the model. The outputted result is a one-dimensional array. This flattened array leads the model to extract features regarding the data. Without this flatted array, it is not possible to extract features regarding the data. Every row of data in the specified period performs this process. ConvLSTMs operate at their best on time-series data. These extracted features is the LSTM layer input.

Keras Layer

The format of Keras's ConvLSTM's layer is rows, columns and channels.

Layer Input

As previously explained, the features extracted is the LSTM layer input. This layer can come in many forms. As seen in [82], it is usually a 3D tensor with a pre-defined shape. The shape consists of a number of samples (this can be set), time steps (this can be set also) and features (previously extracted from the flattened array) as seen in related work [83].

Layer Output

The output of the ConvLSTM greatly depends on the attributes obtained for the input layer. Each input produces an output. CNNs and LSTMs evolved around producing a sequence of sequential information over a time-period. The output is in the same format as the input layer; samples, time steps and features. If you were to merge the output of a CNN and LSTM, it would be exactly like the outputted result of a ConvLSTM.

Additional (optional) parameters

- filters: Dimensionality of the output space
- kernel_size: Length of convolutional window.
- padding: Ensure output length = input length.
- data_format: Dimensions of input match input shape. Ensure smooth running.
- activation: Activation function. a(x) = x is the default is the linear function

LSTM Layer optional parameters

- recurrent_activation: Applies activation function for time steps.
- return_sequences: Specify the following: Return last output in outputted result or full sequence result.

Human Activity Recognition applications of ConvLSTM

Wang et al. [84] compares Convolutional Recurrent Neural Networks using accelerometer data in classifying human activities. The proposed approach revolves around a personalised recurrent neural network, which contains convolutional layers. The dataset used to evaluate performance is the WISDM Dataset. The data was captured in controlled laboratory settings. This unique algorithm from scratch achieved state-of-the-art results with high computational efficiency. The model identified six different physical activities. Accuracy as high as 96.44% was achieved.

Saeedi, Norgaard, and Gebremedhin [85] creates a deep neural network that utilises both convolutional and Long-Short Term Memory (LSTM) layers while using data generated from sporting individuals in fitness applications with applied wearable sensors. The Opportunity activity recognition dataset aided this experiment. Activity learning parameters and feature extraction were two key aspects in ensuring this experiment yielded excellent results. Evaluation metrics used were precision, recall and F1-Score. This ConvLSTM network achieved an accuracy of 90% on 20% of unlabelled data.

Sudhakaran and Lanz [86] presents a deep learning ConvLSTM to perform encoding on short time interval data generated from RGB images. Feature learning and extraction is performed using convolutional long short-term memory cells. Results proved conclusive as performance evaluation surpassed endless amounts of state-of-the-art results. All previous results yielded less than 20% accuracy than this unique ConvLSTM approach. Accuracy achieved was 79.6%, given the complex task of performing encoding on short time interval data generated from RGB images.

3.6 MultiLayer Perceptron

MultiLayer Perceptron is an advanced enhancement based on the perceptron algorithm. The perceptron algorithm evolves around one layer. It is simple and straightforward; neurons multiply with weights while adding bias to the weights. Weights update when there is a misclassification error.

To update the weights, the following equation is used:

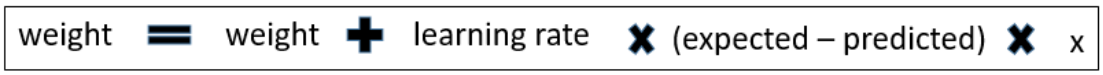

Figure 14 MLP Learning Rate

MultiLayer Perceptron is a *Feed-Forward Neural Network.* A simple neural network, as mentioned before, consists of three layers; input layer, hidden layer and output layer. Data feeds into the input layer and processes through the hidden layer. The output layer generates a result. More hidden layers leads to a more complex model.

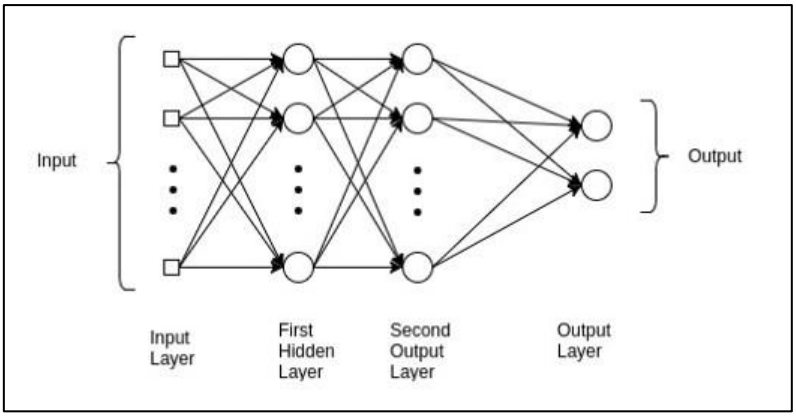

Figure 15 MLP Architecture [87]

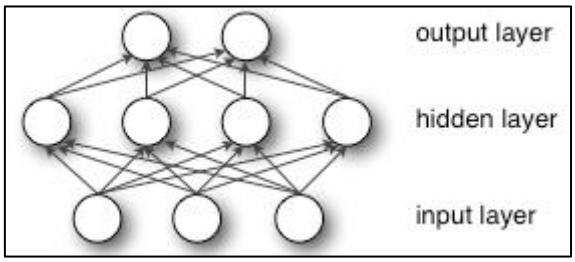

Figure 16 MLP with one hidden layer [87]

MLP is a typical NN model. The resulting goal is usually to approximate a function such as f (). Let us take for example the following classifier;

$y = f^*(x)$

As you can see, the input class x must be closely related or used to identify the output class y. MLP intends to identify the best/closest approximation to the above equation. In order to do this, the networks must map

$y = f(x; \theta)$

Identifying and learning the most appropriate parameters for θ is the next step.

The functions in each layer of a MLP network link together. The network obtains a linear sum of inputs to identify what needs to be classified. Each layer subsequently performs an affine transformation on a linear sum of inputs. Weisstein [88] defines an affine transformation as 'any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation)'.

A MLP network is a typical neural network as all the layers are fully connected. Each layer connects to the previous layer. This enables each layer to learn from one another. Weights associated with units in a fully connected layer are unique and view as independent to all other units in the layer.

The loss function measures performance in a MLP. The main aim is to ensure loss is minimised. Low loss leads to significant correspondence to the true class. High loss leads to insignificant correspondence to the true class. Before training, identifying an optimisation technique as well as a pre-defined loss function is required. An optimiser also increases computability and performance. Optimisation procedures identify network attribute values for each weight, in order to minimise the loss function, leading to greater correspondence to the true class.

Random values assigned to weights while changing them at regular intervals lead to a lower loss. This change in weight leads the network to progress in the direction associated with the gradient of the loss function. The learning rate must be set after every training iteration, to visualise that the loss function is moving in the right direction.

Activation function

Activation function explains how non-linear the model is. It details the relationship between the input and output variables; this is processed in a non-linear way. Examples of activation functions are ReLu, Tanh and Sigmoid.

The following are the three steps in training a MLP model.

- Forward pass; pass along the input while multiplying it by weight, add bias, produce output.
- Calculate error or loss; this is essentially expected output minus predicted output. Backpropagate loss is the next step.
- Backward pass; backpropagate loss, update weights by using gradient and analysing gradient flow.

Human Activity Recognition applications of Multilayer Perceptron

Mo, Li, Zhu, and Huang [89] compares convolutional neural networks and multilayer perceptron performance on the classification of activities based on the CAD-60 Dataset. The CAD-60 Dataset [90] contains RGB-D video sequences of volunteers performing activities. The Microsoft Kinect sensor records sensor signals. This research focuses on data pre-processing along with feature extraction to generate highly accurate performance results. The model presented combines CNN and MLP by using CNN for feature extraction and using MLP for the classification of the activity. It proved highly successful with the model achieving 81.8% accuracy across twelve different types of activities.

Catal, Tufekci, Pirmit and Kocabag [91] compares performance of a model integrating aspects of Decision Tree, Multilayer Perceptron and Logistic Regression.

Accelerometer data is analysed in order to classify activities. Related work [92] performs analysis on the WISDM Dataset (Wireless Sensor Data Mining) which contains information from 36 volunteers performing activities as seen in related work [92]. The proposed model achieved state-of-the-art results while achieving a superior performance when compared to a Multilayer Perceptron approach in related work. Results prove that integrating an ensemble of classifier that yield outstanding results in the activity recognition domain.

Talukdar and Mehta [93] built a Multilayer Perceptron network to classify physical human activities through the automated analysis of video data. The volunteer performed six activities 25 different times wearing a variation of different clothes each time. The activities performed were; walking, jogging, running, boxing, hand waving and hand clapping. Talukdar et al. [93] presents an MLP network that trains the data through a recurrent neural network that led to a vast reduction in learning time for the features and labels. The model achieved an overall accuracy of 92%.

3.7 AutoEncoder

Autoencoders are an example of a feedforward neural network. Most of the time, the input is identical to the resulting output. Firstly, the input compresses into a latent-space representation. As seen in [94], a latent-space representation is summarisation of the input, it compresses to identify the key values and attributes.

An Autoencoder consists of three *components*:

- Encoder
- Code
- Decoder

The encoder takes the input and compresses it, upon producing the code. The decoder then takes this code and aims to reconstruct the input. It aims to summarise the input like previously explained.

Before building an Autoencoder, it is essential that the network have a pre-defined idea of how to encode the input and decode the output. These methods are essential in the efficient running on an Autoencoder network. Building a loss function to compare predicted output with expected output is also essential.

The *important features* in the Autoencoder compression algorithm are as follows:

- Dimensionality Reduction; ensure compression is made.
- Data-specific; specific summary containing key insights.
- Loss function; identify appropriate loss function for comparison.
- Unsupervised; generate own labels in respect to training data.

Autoencoder Architecture

- Fully connected Artificial Neural Network.
- Input passed through encoder.
- Code is produced.
- Decoder produces output based on the code; aims to mimic the input.
- **Goal:** Output identical to Input.

Hyperparameters

The following hyperparameters must be set before training the Autoencoder.

- Code size; number of neurons in middle layer.
- Number of layers; how deep the Autoencoder will be.
- Number of neurons per layer
- Loss function; mse or binary crossentropy

Autoencoders are trained using backpropagation, previously mentioned in section 3.2.

Autoencoders in HAR

Wang [95] provides a continuous Autoencoder (CAE) neural network for recognising physical human actions using wearable sensors. A fast stochastic gradient descent (FSGD) algorithm proposes to update the gradient weights in the continuous Autoencoder NN. Wang [95] conducts analysis on the Swiss Roll Dataset [96] and the Human Activities Dataset [97]. The human activities dataset contains human action (e.g. walking upstairs, walking downstairs) data by using body-worn inertial and magnetic sensors. The Swiss Roll Dataset contains 2000 data points. The CAE experiments with different epochs, applied feature reduction and a time and frequency domain feature extract (TFFE) to achieve high accuracy and correct differentiation rate. The CAR achieved a correct differentiation rate of 99.3%.

Zou [98] proposes DeepSense, a device-free HAR that can successfully detect activity recognition for multiple activities using WiFi-enabled IoT devices. The framework built on DeepSense contains an Autoencoder Long-Term Recurrent Convolutional Network (AE-LRCN). It combines an AE module, RNN module and a CNN module to build a framework capable of performing such a complex task. The dataset consists of common activities such as running, jogging, standing still and walking downstairs. All activities were performed in an indoor environment. DeepSense achieved a high accuracy of 97.4%.

Almaslukh, AlMuhtadi, and Artoli [99] proposes a Stacked Autoencoder (SAE) using activity recognition data generated from a smartphone. High accuracy and low computational costs are needed conducting HAR experiments using smartphones.

Stacked Autoencoders are renowned for achieving high accuracy and successfully classifying activities. The proposed method yielded excellent results with the classification accuracy increasing 1.1% from previous studies. It increased from 96.4% to 97.5%. The average time for each testing sample to be recognised decreased from 0.2724ms to 0.0375ms

3.8 Random Forest

Decision trees are basic building blocks for creating random forests. Decision trees map out all possible ways of deriving a decision. Given a problem and a solution, a decision tree enables the simple visualisation of all possible routes from been given the problem to deriving the potential solution. It is a detailed flowchart of questions and answers.

Decision trees allow you to dissect data perfectly in order to weigh up possible options to come to a specific conclusion. It weighs up all possible routes while ranking each routes importance, risk and loss. It allows for the split in possibilities in a way that differentiates each possibility with one another.

The Random Forest Classifier

A random forest is a group of individual decision trees. They merge to form an ensemble and operate together. The random forest produces a range of predictions and possible outcomes, each prediction compares with the models actual prediction to identify and rank each predicted values specific importance.

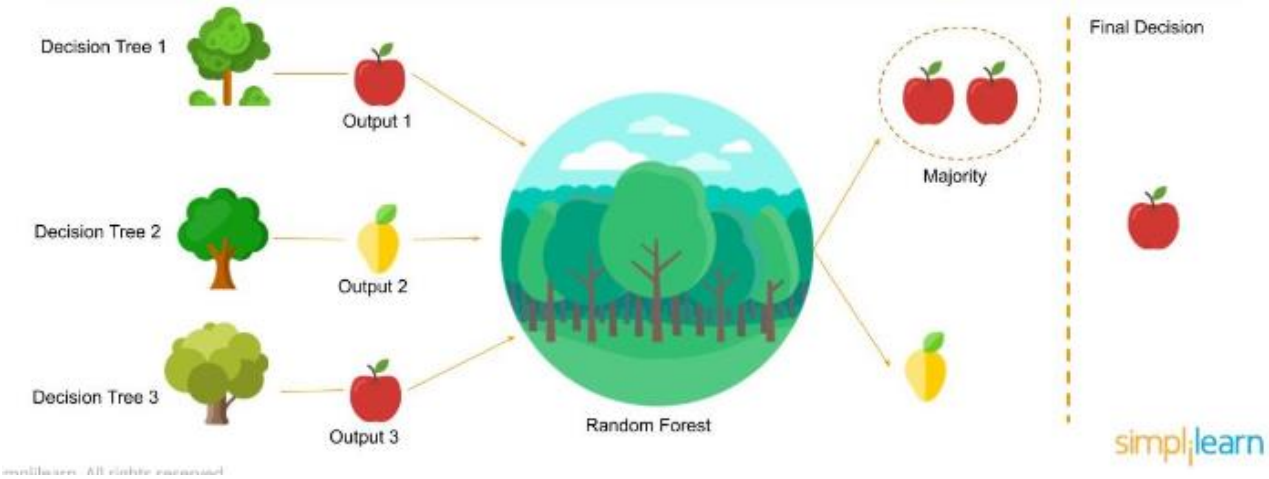

Figure 17 Random Forest [143]

Random forests excel in performance due to the fact that;

A group (2 or more) of dissimilar (but, slightly similar, in a sense that the same input is used) trees working together as an integrated model will yield excellent performance results rather that an individual tree operating alone.

Low correlation between models (trees) is important to produce the desired output. It ensures predictions are precise in comparison to a single trees prediction. A key insight from Random Forest prediction is that trees protect each other. If one tree fails in one aspect (category), another tree will protect it and ensure the overall models performs well. As depicted in [100], in times where a model is going astray, a tree will lead it in the right direction.

In order for a random forest to perform well:

1. Features must be utilised correctly.
2. There must be low correlation between trees.

Human Activity Recognition applications of Random Forests

A. Ignatov [101] compares performance of a 1D Convolutional Neural Network and Random Forest approach for classifying human actions. An accelerometer sensor gathers the data from a smartphone. The data generated from the accelerometer is transformed into x, y and z acceleration data which converts to magnitude data. The magnitude data enables the 1D CNN to learn the features more efficiently than the random forest. The 1D CNN outperforms the Random Forest by achieving 92.71% accuracy, while the Random Forest achieves 89.10% accuracy.

Hu, Chen, Hu and Peng [102] proposes an effective class incremental learning method, called class incremental random forests (CIRF). This method aims to modify existing activity recognition frameworks by detecting new activities added to the existing framework. The random forest hones in on two strategies while aiming to modify the framework. The two strategies are adopting Gini index and using information gain. Their goal is to split leaves in the decision tree of the random forest. The DSADS [103], HARUDS [104] and OPPORTUNITY Dataset [105] performs analysis while an accuracy of 98%, 97% and 89% is achieved on the incremental approach respectively.

Zubair, Song and Yoon proposes a Random forest and decision tree with AdaBoost to classify physical activities. The public domain HAR dataset [107] is analysed. The proposed method focuses on selecting the correct features to perform analysis on, while removing and excluding redundant features. The classifier trains on 90% of the training data while testing on 10%. The classifier achieved 99.8% and 99.9% accuracy for five different activities, which were sitting, sitting down, standing, standing up and walking. The RF with AdaBoost method achieved accuracy that outshone other research in previous work.

3.9 Extreme Gradient Boosting

Gradient boosting machines are associated with a distinctive type of machine learning branch called Ensemble Learning. The objective of ensemble learning is to train and predict a variety of models at the same time, while each model aims to outperform each other with respect to their output. Take for example, the route from Paris to Berlin. There are many alternative travel options. As you proceed to take each route, you begin to learn which route is faster and more efficient, leading to the 'superior' route. Taking time to learn each model has led to the conclusion that X is the superior route. Ensemble learning simply implies this strategy.

XGBoost implements boosting. Boosting aims to convert weak learners to strong learners. During boosting, iterations lead to the weights of weak learners to adjust accordingly. Bias reduces allowing for an increase in performance. Accuracy, Precision and Recall benefit from the implementation of boosting greatly, as well as a range of evaluation techniques. XGBoost (Extreme Gradient Boosting) is the most best performing boosting algorithm.

XGBoost is a decision-tree-based algorithm that utilises the use of gradient boosting and ensemble learning. XGBoost performs so well on data due to its ability to transform weak learners into strong learners. It utilises boosting within the gradient descent architecture. It allows the framework to develop dramatically with its quick and easy to learn optimisation techniques and parameter enhancements.

XGBoost performance

XGBoost and Gradient Boosting Machines are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

Optimisation Techniques

- **Parallelised implementation**: With parallelised implementation, consecutive trees building at consecutive times occurs. A group of trees identifying as base learners make up the base layer. The outer layer contains the leaf neurons associated with each specific tree; the second inner layer calculates attributes regarding each tree. Run time decreases due to the parallel interchangeable actions of the base layer, the second inner layer and the outer layer. Threads identify with trees run parallel with each other and interchange.
- **Tree Pruning:** This allows for the splitting of trees within the gradient descent architecture. Tree pruning utilises the parameter 'max_depth'. It specifies at which length to begin pruning the tree in a backward direction.

Hyperparameters

- **Regularisation**: Helps prevent overfitting.
- **Sparse features**: XGBoost allows the inputs to 'learn' the best missing value based on accuracy and loss. It handles sparse features well.
- **Iterations**: The number of trees utilised. Using too many trees leads to overfitting on the training data, while not enough trees can lead to underfitting of a model that is too simple.
- **Tree depth**: Tree depth is similar to iterations as too many interactions leads to overfitting while a small depth can lead to underfitting.

Human Activity Recognition applications of XGBoost

Ayumi [108] investigates if Extreme Gradient Boosting is superior in classifying activities in the HAR domain over classical techniques such as Support Vector Machine (SVM) and Naïve Bayes (NB). The UTKinect-Action3D Dataset [109], the Badminton Sports Action Dataset [110] and the Bali Dance Motion Dataset conducts analysis. XGBoost takes more computational time to run as opposed to the other two methods, but prevails as the best method with higher accuracy, precision, recall and F1-Score.

Zhang, Zhao and Li [111] propose an XGBoost method to recognise activities on their own dataset, which they created themselves. The dataset consists of 40 volunteers performing multiple activities contained in an indoor facil0ity. XGBoost outperforms other ensemble classifiers with a higher recognition rate in accuracy, F1-Score and precision. F-score reached heights of up to 84.41% while accuracy surpassed previous studies achieving a rate of 84.19%.

Nguyen, Fernandez, Nguyen and Bagheri [112] present an XGBoost model using wrist-worn accelerometer data, RGB-D camera data and environmental sensor data to classify activities. This unique approach achieved an increased improvement of 38% accuracy in comparison to previous studies. A Brier score of 0.1346 was also achieved which means that 90% of the time, it predicts the correct activity.

3.10 Dimensionality Reduction

Dimensionality reduction is a branch of machine learning which incorporates several ML and statistical techniques in order to reduce the number of features (variables) considered in a dataset. Classification problems in ML sometimes factors in too many variables while deploying the model. The higher the number of variables, the more difficult it is to visualise results. Dimensionality reduction aims to focus on essential variables while cutting out unnecessary variables.

Feature selection and *feature extraction* are two important aspects of dimensionality reduction:

- Feature selection refers to selecting a subset of features (from the original set of features) for model deployment.
- Feature extraction refers to forming a new, modified set of features (from the original set of features).

The Curse of Dimensionality

R. Bellman describes the Curse of Dimensionality in his book 'Dynamic Programming' in 1953. Bellman describes the curse of dimensionality as the issue where error increases as the number of variables factored into the problem increases. The number of variables taken into account enlarges the whole scale of the problem, more data is being involved, analysed and processed. Bellman explains why he thinks this has astronomical effect on the Euclidean distance, leading to a vast increase in run time. The increased amount of data in question allows material to be gathered. Finding the optimal point, desired output, in this increase in gathered material becomes more difficult.

Clustering

Clustering aims to assign objects to clusters (groups) while ensuring these groups are all dissimilar. The goal of clustering algorithms is to generate insight into relationships between features. It aims to uncover these hidden relationships while dissecting these objects structure to gain in-depth knowledge about the object.

Clustering is the assignment of objects to homogeneous groups (called clusters) while making sure that objects in different groups are not similar. Clustering is an unsupervised task as it aims to describe the hidden structure of the objects. A set of characters called features DESCRIBES EACH OBJECT. The first step of dividing objects into clusters is to define the distance between the different objects. Defining an adequate distance measure is crucial for the success of the clustering process.

Methods

Two dimensionality reduction techniques for visualising high-dimensional data are:

- Density-based spatial clustering of applications with noise (DBSCAN).
- T-Distributed Stochastic Neighbour Embedding (t-SNE).

Dimensionality Reduction with Human Activity Recognition

Erol and Amin [113] proposes a Radar Data Cube Processing for Human Activity Recognition Using Multi Subspace Learning. This proposed technique aims to combine

multilinear principal component analysis and linear discriminant analysis along with convolutional neural networks to minimise dimensionality reduction and maximise classification accuracy. The technique was implemented on two datasets, which was collected by Erol and Amin themselves in indoor environments. BMPCA achieved the highest test accuracy of 97.2%.

Hassan, Huda, Uddin, Almogre and Alrubaian [114] propose a HAR system using Deep learning by focusing on [114]:

- Gathering sensor data for body worn sensors.
- Conducting feature extraction on par with dimensionality reduction.
- Recognising the human motion.

Their proposed method revolves around Deep Belief Networks, linear discriminant analysis and principal component analysis. Deep belief networks achieved the highest accuracy 97.5%.

El Moudden, Jouhari, El Bernoussi and Ouzir [115] deploy a model for human activity recognition using dimensionality reduction. Figure 18 is a summary of the framework results. High accuracy was widely achieved which shows the beneficial solutions of implementing feature extractors such as PCA and a Kernel PCA.

Features (p)	Features Extraction Model	Features (k)	Pattern Recognition Model	Classification Accuracy %
561	PCA	26	PCA-KNN	94.83
		15		93.32
		8		91.36
		26	PCA-C5.0	94.89
		15		93.28
		8		90.37
	Kernel PCA	26	Kernel PCA-KNN	97.15
		15		96.45
		8		92.50
		26	Kernel PCA-C5.0	96.79
		15		95.87
		8		92.02
	Pristine		KNN	99.19
			C5.0	99.22

Figure 18 E.Moudden et al. [115] results

3.11 Density-based spatial clustering of applications with noise (DBSCAN)

This density-based algorithm is an excellent visualisation model for visualising high-dimensional data. DBSCAN visualises highly clustered dense regions identified in a dataset.

The DBSCAN algorithm finds relationships and little networks in the data that are difficult to find manually. This is excellent for identifying patterns and predicting associations between data points.

Practical Application of DBSCAN Algorithm

Let's say we have an e-commerce website, named 'Vell'. Vell wants to improve online customer service by getting to know customers better, identifying purchasing habits while building a relationship. Vells goal is to soon predict what each customer wishes to buy and provide a personalised 'Suitable for X' section which entails items that are relevant to the customer. Vell could apply DBSCAN to their dataset extracted from their online database. DBSCAN can learn specific data patterns in items that customers have bough, and find clusters between these items. Vell can learn these patterns and then build the personalised recommender system.

DBSCAN dissects data by partitioning it in two ways. Firstly, the 'Dense' data points are extracted while the 'Non-Dense' data points are left in the background. The space where these 'non-dense' data points are left is classified as the sparse background. The data points left in the sparse background are classified as 'Noise'. These, so-called, outliers are low-density data points from low-density regions.

Parameters

The DBSCAN algorithm requires two parameters:

eps: Epsilon choses how close data points in a cluster are to each data point. Epsilon aims to find which data points are neighbours with each other. If the so called, Epsilon, is less than or equal to the set 'eps' value, then these two data points are considered neighbours and patterns/relationships can be learnt leading to predicting associations between points.

minPoints: minPoints or minSamples is classified as the minimum amount of data points needed in a cluster of points to be so called a dense region. For example a region of densely related data points that has 25 points in it would need a minPoints = 25 to be assigned.

Setting Parameters

It is essential to take careful consideration when setting the Epsilon value and the minimum number of points to find a dense region. Setting a small epsilon value will ignore and not factor into account a vast majority of the data. Too high an epsilon will merge all the cluster points in the data into one so it will be hard to distinguish and learn associations. Setting the minPoints relates to how big the dataset is. The larger the dataset, the larger the number of minimum points.

Human Activity Recognition Applications of DBSCAN

Kwon, Kang and Bae [116] propose a human activity recognition model using smartphone sensors to classify activities 'walking', 'running', 'sitting', 'standing' and

'lying down'. DBSCAN achieved accuracy of at least 90% when classifying these activities. Figure 19 shows how Gaussian and k-means techniques selected how many features (e.g. MinPts = 6, 8, 10, 12, 14 and 16) were proposed for different iterations. These feature selection and extraction techniques led DBSCAN to successfully visualise the data and achieve high accuracy.

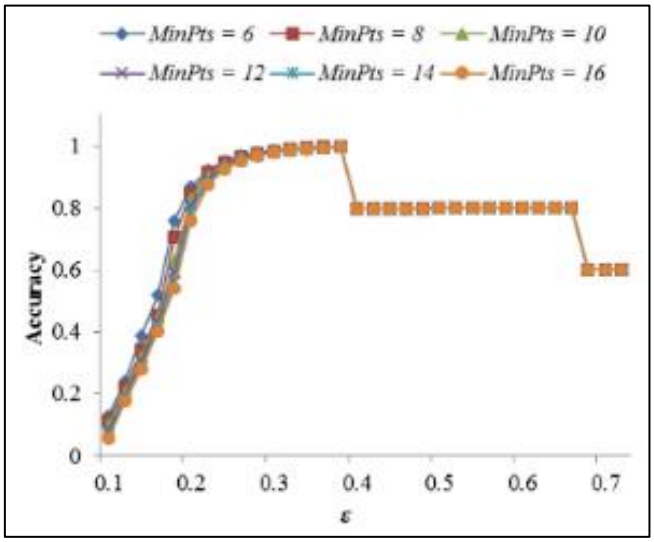

Figure 19 Kwon et al. [116] Results

Guo et al [117] proposes a human activity recognition model using smartphone data in a medical setting. The smartphone data collected from an accelerometer, gyroscope, magnetometer, gravity accelerometer and a linear accelerometer classifies physical human activities. The DBSCAN algorithm classifies unknown activities such as 'run' and 'upstairs' while the clustering result is depicted in figure 20. DBSCAN result for unknown activities 'run' and 'upstairs'.

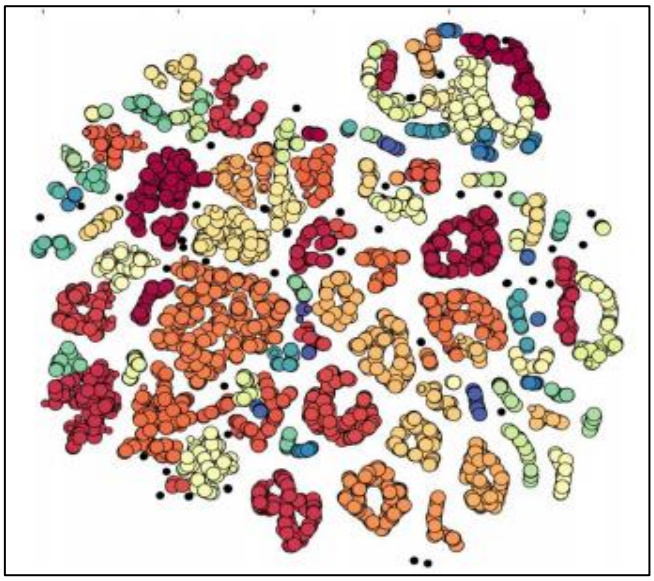

Figure 20 Guo et al [117] Results

3.12 t-Distributed Stochastic Neighbour Embedding (t-SNE)

t-SNE is another non-linear dimensionality reduction technique for visualising high-dimensional data. It is widely used in activity recognition for analysing correlation between features that are relevant in classifying a specific activity.

t-SNE Algorithm

1. Calculate probability of similarity in data points in high-dimensional space.
2. Calculate probability of similarity in data points in low-dimensional space.
3. Similarity calculation based on the similarity of points 1 and 2 identifies if they are neighbours. Similarity calculation factors in Gaussian distribution (as mentioned above in DBSCAN HAR section 3.11) of point 1 relative to point 2.
4. The conditional probability between the two points is calculated.
5. The difference between the conditional probability of all data points aims to be minimised. Kullback-Leibler divergence is an excellent gradient descent method for interpreting the minimisation of conditional probability. This technique is performed before the final step.
6. The goal of t-SNE is to map high-dimensional data points in a low-dimensional space. As conditional probability is calculated and Kullback-Leibler divergence is minimised, mapping the data occurs.

Pattern identification along with analysing correlation between features on the clusters lead to identifying similarities between data points.

Setting Parameters

Perplexity: Referred to as the number of nearest neighbours.

Learning rate: The learning rate sets how much the model will adjust and change in responding to the error produced.

Number of iterations: Refers to how many times algorithm is performed on the model.

Human Activity Recognition applications of t-SNE

Reunanen et al [118] proposes multiple t-SNE visualisation graph approaches for classifying eight physical activities (walking, Nordic walking, running, soccer, rowing, cycling and lying down). The data utilised is from the Palantir Context Data Library in which triaxial wrist-worn accelerometers generated the data. Their proposed technique is compared with Logistic regression, Random forest, decision tree, Adaptive boosting and linear support vector machine. Their technique outperformed all of these techniques as well as state-of-the-art baseline models achieving an accuracy as high as 91.22%.

Figure 21 depicts t-SNE visualisation of the eight activities.

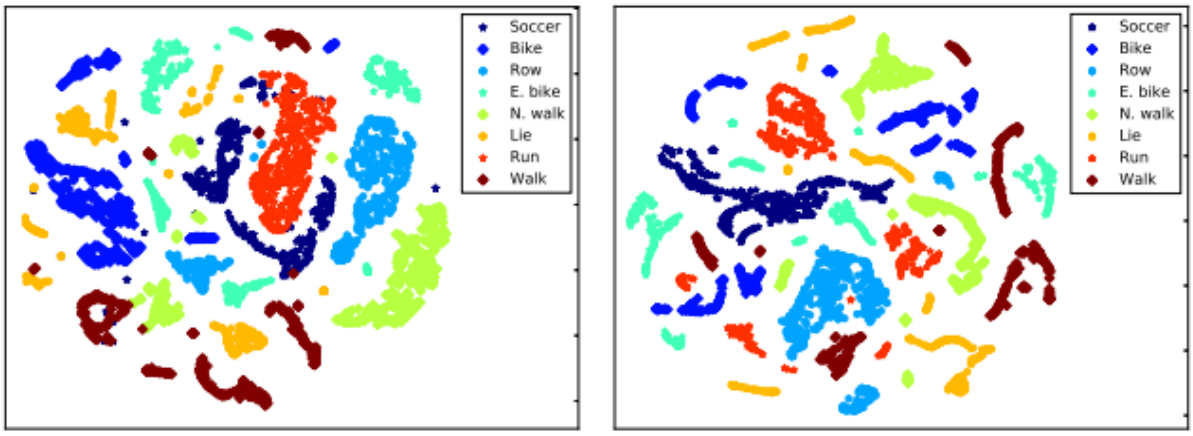

Figure 21 Reunanen et al [118] Results

Brophy, Veiga, Wang, Smeaton and Ward [119] propose a HAR model using Photo plethysmograph Sensor Data. T-SNE visualises the sensor data to produce classification accuracies on the activities performed so users from outside the data science field can easily interpret and understand the idea. Classification accuracy reached an average of 92.3% while producing multiple t-SNE visualisations. Figure 22 is an excellent visualisation of the activities performed. Figure 22 visualises the Convolutional Neural Network nodes [119].

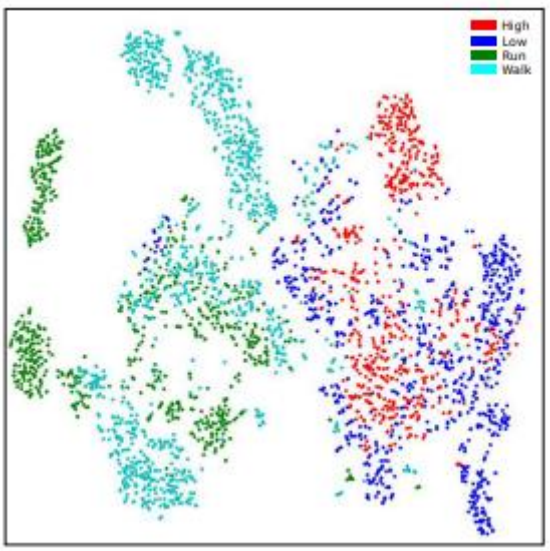

Figure 22 Brophy et al. [119] Results

3.13 The NULL Class Problem

Human activity recognition systems contain a vast amount of streaming data. Only a certain percentage of this streaming data is significant in the performance of the HAR system. There is a slight imbalance between the portion of significant data and insignificant data. This leads to some of the activities to be easily confused with

activities that have similar range of motion patterns and are irrelevant in predicting the activity in question. For example, jogging is often mistaken for running and cycling is often mistaken for running upstairs. These easily confused activities are the so-called NULL class.

Detecting, monitoring and modelling the NULL class is a tough task. The NULL class often represents a massive portion of the dataset. As seen in [120], the NULL class represents 72.28% of the whole dataset. It is good practice to remove the Null class if there is a skewed pattern in the dataset. If the datasets attribute information and labels differ substantially from the correctly classified activities, then the NULL class problem may be identified and appropriate action or precautions taken.

The NULL class is not a huge problem and can be dealt with accordingly when analysing and evaluating datasets. At most, it leads to minor confusion when classifying activities in HAR systems. As seen in [122], applying self-learning can reap benefits of the NULL class. The studies in [122] present a performance comparison of self-learning activity spotters to show the benefits of this proposed approach. Results yielded an increase of 15% in performance, which outlines that the NULL class if managed accordingly can generate great model performance.

3.14 State-of-the-Art Summary / Conclusion

This chapter has presented an over of the State-of-the-Art of Machine Learning algorithms used in Human Activity Recognition. The chapter has been organised into eight principle sections, with each section detailing different machine learning techniques. Within each section, algorithm explanations give an outline of network architecture and appropriate layers. Human Activity Recognition applications of all of the above algorithms give the reader a feel of what is to come in the next section.

As described by the application of HAR section of each algorithm, Human Activity Recognition is a continuously growing field with research on the rise, leading to an increase in the number of applications it can benefit in the real world. The State-of-the-Art presented many approaches in classifying physical human activities. Activity Recognition researchers aim to tweak and improve existing baseline models to make their approach unique and outshine their favourite classifiers.

However, the No Free Lunch theorem states that there is no specific machine learning algorithm that is statistically superior to all other machine learning algorithms for all learning problems (Classification and Regression) [123]. There is no 'Number one' or 'superior' algorithm. The No Free Lunch theorem concludes that researchers conduct research implementing various machine and deep learning models to analyse, which models best suits the data.

The next step in this research is to evaluate all eight algorithms presented above to classify and visualise the MHEALTH dataset with appropriate evaluation techniques.

The research aims to discuss the significant impact modelling machine and deep learning to health data has on human activity recognition. The algorithms presented in this research are as follows:

Convolutional neural network, long short-term memory network, ConvLSTM, Multilayer Perceptron, XGBoost, Autoencoder by Random Forest, DBSCAN and t-SNE.

4 Dataset

The following chapter discusses the MHEALTH dataset [31], experimental setup and attribute information.

MHEALTH Dataset

The MHEALTH dataset consists of body motion and vital signs recordings. Ten volunteers conducted the experiment, each of different characteristics. The subjects' task is to perform 12 different types of activities. The accelerometer, gyroscope and magnetometer placed on the subjects' body measure acceleration, rate of turn and magnetic field orientation. These sensors measures the range of motion experienced by each body parts. The electrocardiogram sensor positioned on the chest also provides 2-lead ECG measurements. The ECG can basic heart monitoring, checking for various arrhythmias or looking at the effects of exercise on the ECG.

Experimental Setup

The collected dataset comprises body motion and vital recordings of the ten volunteers performing the physical activities as stated above.

Shimmer2 [BUR10] wearable sensors were used for the recordings.

Elastic straps complement the sensors on the subjects' chest, right wrist and left ankle.

The sensors capture the range of motion of each body part during the activity.

The sensors measures acceleration, rate of turn and the magnetic field orientation.

The sensor positioned on the chest also provides 2-lead ECG measurements, which are not used for the development of the recognition model but rather collected for future work purposes. This information aids basic heart monitoring and checks for various arrhythmias or looking at the effects of exercise on the ECG.

All sensing modalities record at a sampling rate of 50 Hz. 50 Hz is sufficient for capturing human activity. A video camera recorded each session for each subject.

Each subject executed the activities to the best of their abilities. There were no constraints. The experiment was recorded in an out-of-lab environment.

MHEALTH Dataset Overview

Table 1 outlines key dataset characteristics.

MHEALTH Dataset Overview	
Dataset Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification
Number of Instances:	120
Number of Attributes:	23
Missing Values:	N/A
Data Donated:	07/12/2014

Table 1 mhealth dataset characteristics

Activities

Table 2 outlines the activity set.

Activity Set	
Activity 1	Standing still (1 min)
Activity 2	Sitting and relaxing (1 min)
Activity 3	Lying down (1 min)
Activity 4	Walking (1 min)
Activity 5	Climbing stairs (1 min)
Activity 6	Waist bends forward (20x)
Activity 7	Frontal elevation of arms (20x)
Activity 8	Knees bending (crouching) (20x)
Activity 9	Cycling (1 min)
Activity 10	Jogging (1 min)
Activity 11	Running (1 min)
Activity 12	Jump front & back (20x)
Min = duration of minutes for each exercise	
20x = Number of repetitions of each exercise	

Table 2 mhealth activity set

Attribute Information:

The data collected for each subject is stored in a different log file: 'mHealth_subject.log'. Each file contains the samples (by rows) recorded for all sensors (by columns). The labels used to identify the activities are similar to the abovementioned (e.g., the label for walking is '4').

Attribute Information	
Column 1:	acceleration from the chest sensor (X axis)
Column 2:	acceleration from the chest sensor (Y axis)
Column 3:	acceleration from the chest sensor (Z axis)
Column 4:	electrocardiogram signal (lead 1)
Column 5:	electrocardiogram signal (lead 2)
Column 6:	acceleration from the left-ankle sensor (X axis)
Column 7:	acceleration from the left-ankle sensor (Y axis)
Column 8:	acceleration from the left-ankle sensor (Z axis)
Column 9:	gyro from the left-ankle sensor (X axis)
Column 10:	gyro from the left-ankle sensor (Y axis)
Column 11:	gyro from the left-ankle sensor (Z axis)
Column 13:	magnetometer from the left-ankle sensor (X axis)
Column 13:	magnetometer from the left-ankle sensor (Y axis)
Column 14:	magnetometer from the left-ankle sensor (Z axis)
Column 15:	acceleration from the right-lower-arm sensor (X axis)
Column 16:	acceleration from the right-lower-arm sensor (Y axis)
Column 17:	acceleration from the right-lower-arm sensor (Z axis)
Column 18:	gyro from the right-lower-arm sensor (X axis)
Column 19:	gyro from the right-lower-arm sensor (Y axis)
Column 20:	gyro from the right-lower-arm sensor (Z axis)
Column 21:	magnetometer from the right-lower-arm sensor (X axis)
Column 22:	magnetometer from the right-lower-arm sensor (Y axis)
Column 23:	magnetometer from the right-lower-arm sensor (Z axis)
Column 24:	Label (0 for the null class)
Units: Acceleration (m/s^2), gyroscope (deg/s), magnetic field (local), ecg (mV)	

Table 3 mhealth attribute information

5 Approach

Six classification algorithms, Multilayer Perceptron, Convolutional Neural Network, Long Short-Term Memory, ConvLSTM, XGBoost and Autoencoder with Random Forest and two data clustering algorithms, DBSCAN and t-SNE are proposed in this thesis to analyse the MHEALTH dataset. This chapter outlines the approach for each proposed algorithm, revolving around the CRISP-DM methodology process.

5.1 Data Flow Diagram for Classification

Figure 23 shows the data flow and approach implemented to classify the activities detailed in chapter 4. All three sensors data (gyroscope, accelerometer, magnetometer and electrocardiogram) are classified together while predictions are generated from each individual classification model. A classification model including all six classification predictions is generated, followed by an in-depth evaluation procedure. The best overall prediction model is then identified.

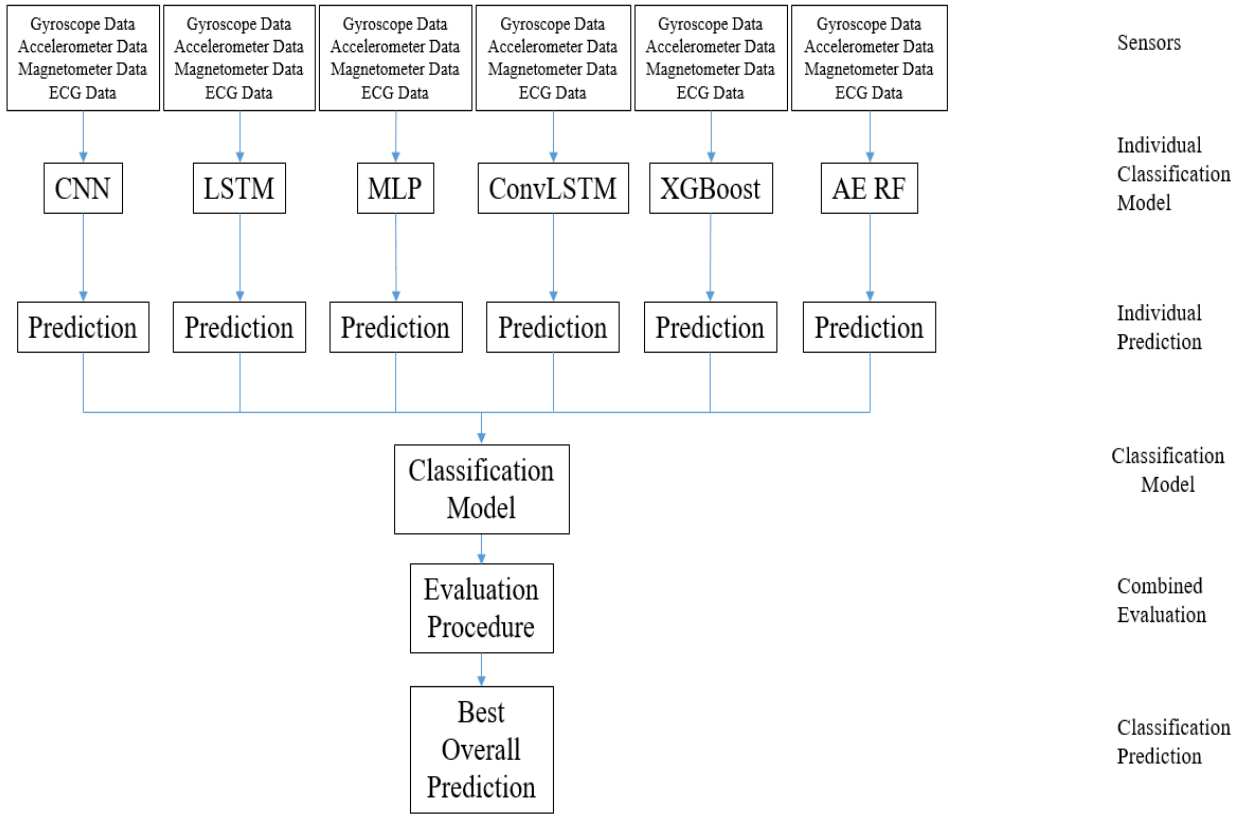

Figure 23 Data Flow Diagram for Classification

5.2 Modelling – CRIPS-DM Cycle

The CRISP-DM methodology is a well-structured process that is applicable to any data analytics project. This methodology involves a list of steps to approach and successfully complete any analytics related project. Given the large scope of this research, this architecture details the continuous flow of the research throughout. Figure 24 presents the CRISP-DM Cycle of this research.

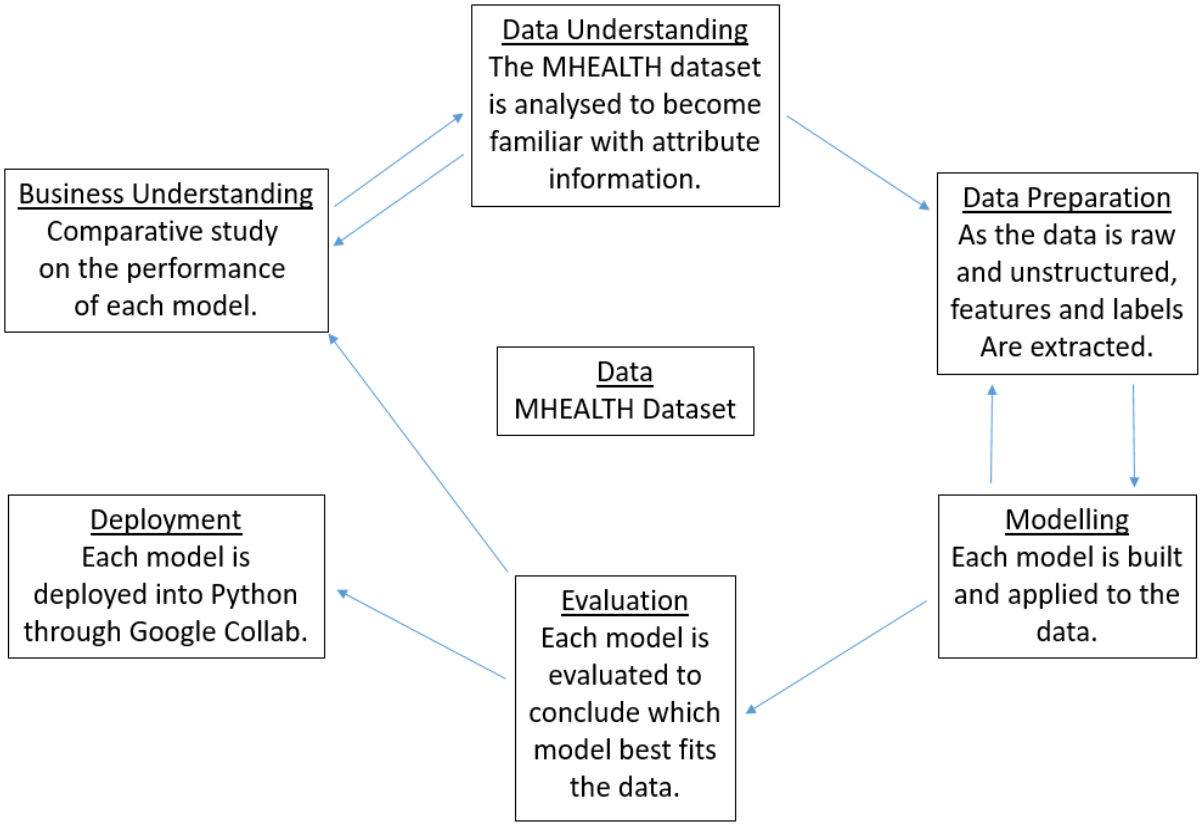

Figure 24 CRISP-DM Cycle

The six stages of the CRISP-DM Cycle for this research are:

Stage 1: Business Understanding

The first goal of this research is to build six different type of prediction models and conduct a comparative study on the performance of each model to identify which model yields the greatest results in terms of classifying human activities. The six models are CNN, LSTM, ConvLSTM, MLP, XGBoost and AE with RF.

The second goal of this research is to conduct analysis on two different dimensionality reduction techniques to identify which technique best visualises the data. Since high dimensional data is used, DBSCAN and t-SNE are the two techniques employed. The goal is to identify which model best visualises the data in terms of clusters.

Stage 2: Data Understanding

For this research, The MHEALTH dataset is analysed. The dataset includes body motion and vital signs recordings for ten volunteers of differing characteristics while performing several physical activities. Ten volunteers perform twelve different activities with three sensors attached to their body (gyroscope, accelerometer and magnetometer). Each activity leads the three sensors to record twenty-three signals. There are no missing values in this classification task. The goal is to predict each activity given the twenty-three signals recorded. The data collected for each volunteer is stored in individual log files: 'mHealth_subject.log'. There are ten log files altogether. Each file contains the data samples recorded for all three sensors.

Stage 3: Data Preparation

Data preparation involves feature extraction, encoding labels to one-hot form, converting the raw data into the right shape for input into the model, normalising the data and finally splitting the data into training and testing.

Each volunteers' data was analysed by DBSCAN and t-SNE separately for various reasons, as discussed in sections 5.6 and 5.7.

Stage 4: Modelling

Building each model involved extracting the features and labels, hyperparameter setting, compiling the model and model evaluation, as previously discussed in chapter 3.

- CNN: The process for building the CNN model is discussed in section 6.1.
- LSTM: The process for building the LSTM model is discussed in section 6.5.
- ConvLSTM: The process for building the ConvLSTM model is discussed in section 6.2.
- MLP: The process for building the MLP model is discussed in section 6.3.
- XGBoost: The process for building the XGBoost model is discussed in section 6.4.
- AE with RF: The process for building the AE with RF model is discussed in section 6.6.
- DBSCAN: The process for building the DBSCAN model is discussed in section 6.8.
- t-SNE: The process for building the t-SNE model is discussed in section 6.7.

Stage 5: Evaluation

This stage involves evaluating each model. Each classification model is evaluated using accuracy, precision, recall, F1-Score, feature importance and confusion matrix. All six-classification models are evaluated based on their categorical cross entropy values. The model, which best suits the data is chosen based on the evaluation procedures.

DBSCAN and t-SNE are evaluated based on how well each one visualises clusters in this high-dimensional dataset.

Stage 6: Deployment

The final stage involves the deployment of each model into Python to analyse the MHEALTH dataset. Each models code is not only applicable to the MHEALTH dataset, but can be applied to any health prediction dataset, as each model is built on multivariate analysis. For example, the classification models could be utilised for disease prediction, while the two-visualisation models could be utilised to detect disease spreading by analysing clusters. Each models experimental results are analysed and discussed in chapter 7 and 8.

5.3 Network Methodology Process

The network methodology process for the classification models is as follows:

Step 1: Data Preparation

Data preparation involves feature extraction, encoding labels to one-hot form, converting the raw data into the right shape for input into the model, normalising the data and finally splitting the data into training and testing.

Step 2: Feature Extraction

The next step is feature extraction. The MHEALTH dataset consist of 10 log files, with each log file corresponding to each of the ten subjects. In order to extract the features (signal attributes) and labels (activities) of each log file, the following feature extraction method is used.

1. Each file is opened by reading it into the Python Google colab framework.
2. Each line in the file is then processed and read in until the final line in the file is reached.
3. Each line is split to extract the labels for each line.
4. 'x' is the line.
5. Once the label is found, it is split and removes the word. If there is a space or comma, it detects it.
6. A sub list is created store all the values of each line.
7. Each line is split into values.

8. An array is then created from the sublist.

This loop successfully extracts all the features and labels of each subjects log file.

Step 3: One-hot encoding

The third step involves encoding the labels to one-hot form. One-hot encoding ensures the categorical variables in the data are converted into a 'form' that is appropriate for DL algorithms to analyse and generate valid, precise predictions. It ensures all values are of numerical relevance. It involves converting the raw, unstructured input data from the log files into the correct shape required for inputting into Tensorflow. Deep learning models are very sensitive to the correct shape of the input data especially when used with the desired activation function. Normalising the data is highly important.

Step 4: **Train/Test Split**

For each of the DL classification models, the data is split into training and testing. The ratio is 80:20.

Step 5: Hyperparameter Setting

Define the input parameters. Hyperparameters such as batch size, number of epochs, learning rate, number of hidden layers, type of hidden layers, shape of input, shape of output and the number of parameters are set. Setting the hyperparameters is important to the successful running of the model in question.

Step 6: Model compilation

Step 6 involves compiling the model, ensuring it is ready to be fitted. It is necessary to structure each model into organised layers. Once the hyperparameters are tuned accordingly, as outlined in step 5, compiling the model can begin. The compiled model is then fitted to the training data in order to classify the volunteers' activities.

Step 7: Model Evaluation

The final step is model training and evaluation. When the model is compiled, fitted on the training data, it is evaluated against both the training data and the testing data. The models predicted output is compared with the true output. Loss is measured using categorical cross entropy. Accuracy, precision, recall, f1 score, confusion matrix, accuracy and loss plots and feature importance measures the models performance.

The methodology process outlined above is performed in Python on the following models:

1. Convolutional Neural Network
2. LSTM (Recurrent Neural Network)
3. ConvLSTM
4. Multilayer Perceptron
5. Autoencoder by Random Forest

5.4 XGBoost Network Methodology Process

The XGBoost process is similar to the methodology process outlined in the previous section.

The Extreme Gradient Boosting methodology process is as follows:

Step 1: Data Preparation

Data preparation involves feature extraction and splitting the data into training and testing.

Step 2: Feature Extraction

The next step is feature extraction. In order to extract the features (signal attributes) and labels (activities) of each log file corresponding to each subject, the following feature extraction method is used.

1. Each file is opened by reading it into the Python Google colab framework.
2. Each line in the file is then processed and read in until the final line in the file.
3. Each line is then split to extract the labels for each line.
4. 'x' is the line.
5. Once the label is found, it is splitting and removing the word. If there is a space or comma, it detects it.
6. A sub list is created store all the values of each line.
7. Each line is split into values.
8. An array is then created from the sublist.

This loop successfully extracts all the features and labels of each subjects log file.

Step 3: Hyperparamater setting

This involves defining the input parameters. Hyperparameters such as maximum depth, number of parallel threads, number of classes, evaluation metric and multiclass classification metric is set. Setting these parameters correctly is vital in ensuring XGBoost produces good classification results. Maximum depth of a tree ensures the model does not overfit. The multiclass classification error rate is calculated as the number of wrong cases over the total number of cases, 'merror' is the metric used. The softmax objective is then used to perform multiclass classification.

Step 4: **Train/Test Split**

For the XGBoost model, the data is split into training and testing in a ratio 0f 80:20.

Step 4: Model compilation

Compiling the model involves setting the number of rounds to be run. The XGBoost model is then trained based on parameter settings, number of rounds and evaluation

settings. Once training is complete, compiling the model can begin. The compiled model is then fitted to the training data in order to classify the volunteers' activities.

Step 5: Model Evaluation

The final step is modelling training and evaluation. When the model is compiled, fitted on the training data, it is evaluated against both the training data and the testing data. The models predicted output is compared with the true output. Loss is measured using categorical cross entropy. Accuracy, precision, recall, f1 score, confusion matrix, accuracy and loss plots and feature importance measures the models performance.

5.5 DBSCAN Network Methodology Process

Due to the MHEALTH dataset being so large and the computational ability of the computer used to perform the experiment, the DBSCAN algorithm was run separately for each subject.

An experiment was conducted on the full dataset but the computer failed to execute the plots as the RAM and memory failed to handle the volume of the output.

The DBSCAN experiment is run on each subjects text file separately: mHealth_subject1, mHealth_subject2, mHealth_subject3, mHealth_subject4, mHealth_subject5, mHealth_subject6, mHealth_subject7, mHealth_subject8, mHealth_subject9 and mHealth_subject10.

The DBSCAN network methodology process is as follows:

Step 1: Data Preparation

Preparing the model doesn't require any pre-processing. DBSCAN has the built in capacity to process, analyse and fix any complications.

Step 2: Feature Extraction

The next step is feature extraction. The MHEALTH dataset consist of 10 log files, with each log file corresponding to each of the ten subjects. In order to extract the features (signal attributes) and labels (activities) of each log file, the following feature extraction method is used.

1. Each file is opened by reading it into the Python Google colab framework.
2. Each line in the file is then processed and read in until the final line in the file.
3. Each line is then split to extract the labels for each line.
4. 'x' is the line.
5. Once the label is found, it's splitting and removing the word. If there is a space or comma, it detects it.

6. A sub list is created store all the values of each line.
7. Each line is split into values.
8. An array is then created from the sublist.

This loop successfully extracts all the features and labels of each subjects log file.

Step 3: Hyperparameter Setting

Define the input parameters. Hyperparameters such as Epsilon, the number of jobs and the minimum amount of points is set. Once the parameters are correctly set, clustering can begin.

Step 4: Model compilation

Step 6 involves compiling the model, ensuring clustering can begin. Once the hyperparameters are tuned accordingly, as outlined in step 3, clustering can begin.

Step 5: Model Evaluation

The final step is evaluating the clustered data. The following measures are used to evaluate the clusters:

- Estimated number of clusters
- Estimated number of noise points
- Homogeneity
- Completeness
- V-measure
- Adjusted Rand Index
- Adjusted Mutual Information
- Silhouette Coefficient

The estimated number of clusters is plotted against the noise points for an easily comparable, visually pleasing plot.

5.6 t-Distributed Stochastic Neighbour Embedding (t-SNE)

The following section presents an account of the extended AE with RF algorithm methodology using t-SNE.

Due to the MHEALTH dataset being so large and the computational ability of the computer used to perform the experiment, the t-SNE algorithm was run separately for each subject.

Similar to DBSCAN, an experiment was conducted on the full dataset but the computer failed to execute the plots as the RAM and memory failed to handle the volume of the output.

The t-SNE experiment is run on each subjects text file separately: mHealth_subject1, mHealth_subject2, mHealth_subject3, mHealth_subject4, mHealth_subject5, mHealth_subject6, mHealth_subject7, mHealth_subject8, mHealth_subject9 and mHealth_subject10.

Step 1: Data Preparation

Data preparation involves feature extraction, encoding labels to one-hot form, converting the raw data into the right shape for input into the model, normalising the data and finally splitting the data into training and testing.

Step 2: Feature Extraction

The next step is feature extraction. In order to extract the features (signal attributes) and labels (activities) of each log file corresponding to each subject, the following feature extraction method is used.

1. Each file is opened by reading it into the Python Google collab framework.
2. Each line in the file is then processed and read in until the final line in the file.
3. Each line is then split to extract the labels for each line.
4. 'x' is the line.
5. Once the label is found, it's splitting and removing the word. If there is a space or comma, it detects it.
6. A sub list is created store all the values of each line.
7. Each line is split into values.
8. An array is then created from the sublist.

This loop successfully extracts all the features and labels of each subjects log file.

Step 3: One-hot encoding

The third step involves encoding the labels to one-hot form. One-hot encoding ensures the categorical variables in the data are converted into a 'form' that is appropriate for DL algorithms to analyse and generate valid, precise predictions. It ensures all values are of numerical relevance. It involves converting the raw, unstructured input data from the log files into the correct shape required for inputting into Tensorflow. Deep learning models are very sensitive to the correct shape of the input data especially when used with the desired activation function. Normalising the data is highly important.

Step 4: **Train/Test Split**

For each of the DL models, the data is split into training and testing. The ratio is 80:20.

Step 5: Hyperparameter Setting

Define the input parameters. Hyperparameters such as test size, number of jobs, number of estimators, random state, perplexity, learning rate, number of iterations and number of components. Setting the hyperparameters is important to the successful running of the t-SNE model.

Step 6: Model compilation

Step 6 involves compiling the model. Once the hyperparameters are tuned accordingly, and Autoencoder by Random Forest is performed, compiling the t-SNE model can begin. Data clustering then begins.

Step 7: Model Evaluation

The final step is evaluating the clustered data. The following measures are used to evaluate the clusters:

- Number of clusters.
- Compare clusters to DBSCAN.
- KL-divergence.

5.7 Tools and Techniques

The code for this thesis was implemented through Python 3 Notebooks on Google Collab. As this research involves statistical analysis, deep learning and complex visualisation methods, Python is the perfect platform due to its built-in libraries, which compliments all the models that are used, leading to the completion of effective predictive modelling frameworks.

Python

Python is a high-level programming language that was invented in 1991. Guido van Rossum created the highly complex object-oriented programming language, Python [124]. Python is essential in conducting analysis in the Data Science and Data Analytics domain. Pythons easy to learn framework and built-in packages along with its applicability to almost any problem make it one of the most sought after programming skills.

Research was conducted through Python 3 Notebooks on Google Collab. Tensorflow backend is used to create each deep learning model.

The following libraries are run on top of Tensorflow:

- Keras: DL library that is compatible for conducting analysis on NN's [125].
- Pandas: Aids data structure and data analysis [126].
- Pickle: Enables saving and loading of objects [127].
- Scikit-learn: Easy to use data analysis [128].

- Numpy: High-level mathematical functions [129].
- Matplotlib: Generate plots [130].
- Collections: High performance container datatypes [131].
- Tqdm: Create high-performing loops [132].
- Seaborn: Visualise statistical data extracted from data analysis [133].
- Xgboost; Extreme gradient boosting framework. Useful for extracting feature importance [134].

Highcharts.js

Highcharts.js is a charting library that is written in pure JavaScript [135].

Highcharts provides the following benefits:

- Setup and implementation is easy.
- Can generate various chart types: pie charts, bar charts, scatter plots and many more.
- It is free while support is provided 24/7 with minimal response time.
- Simple configuration syntax.
- Allows sufficient storage of external data.
- Setting up account takes seconds.
- Creating of interactive 3D charts.
- Mobile and desktop friendly.

5.8 Approach Summary

Chapter 5 presented the CRISP-DM methodology process, which was utilised continuously throughout this research. The methodology process of all six classification models and the two data clustering models are discussed. An in-depth account of the business understanding, data understanding, data preparation, modelling, evaluation and deployment regarding each model is also presented. The chapter concludes by discussing the tools and techniques implemented throughout this research.

6 High-level Approach

This chapter presents a detailed account of each algorithms approach along with how the approach is implemented. The building of each model is outlined as well as each approaches network architecture, network layers, optimiser, loss function and hyperparameter setting is discussed.

6.1 Convolutional Neural Network

The CNN network builds on the 'Convolutional Neural Network' outlined in section 3.3.

Network Architecture

This section explains the network architecture of the CNN that aims to classify the human physical activity that each volunteer performs.

- The first layer of each model provides the 'input_shape' for the model. It is a tuple of integers, containing 23 features.
- The CNN model has 23 inputs, 2 1D convolution layers, 2 'MaxPooling1D' layers, 2 Dropout layers, 3 hidden layers and an output layer with 1 output.
- There are 64 neurons in the first convolution layer and 128 neurons in the second convolution layer. 'MaxPooling1D' operation is set to the temporal data in the two convolution layers, acting as feature extractors.
- The 'Flatten' layer is the next layer. It acts as a connector between the two convolution layers and the dense layer. It allows for interpretation of features.
- The first hidden layer has 128 neurons.
- The second hidden layer has 256 neurons.
- The third hidden layer has 512 neurons.
- Rectified linear activation function is set in each of the three hidden layers and the 2 1D Convolution layers.
- Dropout regularisation is set on the first and second hidden layers to prevent/reduce overfitting. 'Dropout' is set to 0.4.
- There are two 1D-convolution layers in the model, e.g. temporal convolutions. These layers forms a convolution kernel that is folded over with the layer input that is defined above. This occurs over a temporal dimension. The resulting output is a tensor of outputs.

- The kernel size is the size of the filter matrix that is applied to our previous two convolution layers. The kernel size is set to 3, resulting in a 3x3 filter matrix.
- The output layer is a dense layer that produces the output. There is 13 neurons in the output layer. It is set to 13 to account for each possible outcome (0-12 activities). The activation function set in the final dense layer is 'softmax'. Softmax ensures the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

Network layers, optimiser and loss function

This section contains an overview of the CNNs network layers, optimiser used and loss function implemented.

- The CNN model is then created that includes an input layer, 2 1D convolution layers, 2 'MaxPooling1D', 2 Dropout layers and four dense layers.
- Compiling the model takes three parameters: optimiser, loss function and metrics. The Adam algorithm best fits the data, it is implemented as the optimiser for this convolutional neural network.
- 'Categorical crossentropy' is the loss function which fits this task the best. The classification task is a multi-class classification, which is highly suited to 'Categorical crossentropy'.
- 'Accuracy' is the metric used to measure performance on the validation set upon training the model.

6.2 ConvLSTM

The ConvLSTM Hybrid network builds on 'Long Short-Term Memory: Recurrent Neural Networks' outlined in section 3.5.

Network Architecture

This section explains the network architecture of the ConvLSTM that aims to classify the human physical activity that each volunteer performs.

- The first layer of each model provides the 'input_shape' for the model. It is a tuple of integers, containing 23 features.
- The Hybrid model includes an input layer, 2 1D-convolution layers, 2 'MaxPooling1D', an LSTM layer, 2 Dropout layers and four dense layers.

- There are 64 neurons in the first convolution layer and 128 neurons in the second convolution layer. 'MaxPooling1D' operation is set to the temporal data in the two convolution layers, acting as feature extractors.
- The next layer of the model is the LSTM layer. 'MaxPooling1D' operation is set here.
- The first hidden layer has 128 neurons.
- The second hidden layer has 256 neurons.
- The third hidden layer has 512 neurons.
- Rectified linear activation function is set in each of the three hidden layers and the 2 1D Convolution layers.
- Dropout regularisation is set on the first and second hidden layers to prevent/reduce overfitting. 'Dropout' is set to 0.4.
- The output layer is a dense layer that produces the output. There is 13 neurons in the output layer. It is set to 13 to account for each possible outcome (0-12). The activation function set in the final dense layer is 'softmax'. Softmax ensures the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

Network layers, optimiser and loss function

This section contains an overview of the ConvLSTM network layers, optimiser used and loss function implemented.

- The Hybrid model is then created that includes an input layer, 2 1D convolution layers, 2 'MaxPooling1D', an LSTM layer, 2 Dropout layers and four dense layers.
- Compiling the model takes three parameters: optimiser, loss function and metrics. The Adam algorithm best fits the data, it is implemented as the optimiser for this convolutional neural network.
- 'Categorical crossentropy' is the loss function, which fits this task the best. The classification task is a multi-class classification, which is highly suited to 'Categorical crossentropy'.
- 'Accuracy' is the metric used to measure performance on the validation set upon training the model.

6.3 Multilayer Perceptron

The MLP network builds on the Multilayer Perceptron outlined in section 3.6.

Network architecture

This section explains the network architecture of the MLP network that aims to classify the human physical activity that each volunteer performs.

- The first layer of each model provides the 'input_shape' for the model. It is a tuple of integers, containing 23 features.
- The MLP model has 23 inputs, 4 hidden layers and an output layer with 1 output.
- The first hidden layer has 128 neurons.
- The second hidden layer has 256 neurons.
- The third hidden layer has 512 neurons.
- The fourth hidden layer has 1024 neurons.
- Rectified linear activation function is set in each of the four hidden layers.
- Dropout regularisation is set on the third and fourth hidden layers to prevent/reduce overfitting. 'Dropout' is set to 0.4.
- The output layer is a dense layer that produces the output. There is 13 neurons in the output layer. It is set to 13 to account for each possible outcome (0-12). The activation function used in the final dense layer is 'softmax'. Softmax ensures the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.
- A 'softmax' activation function is set in the output layer.

Network layers, optimiser and loss function

This section contains an overview of the MLPs network layers, optimiser used and loss function implemented.

- The MLP model consists of an input layer, two dropout layers and five dense layers.
- Compiling the model takes three parameters: optimiser, loss function and metrics. The Adam algorithm best fits the data, it is implemented as the optimiser for this convolutional neural network.
- 'Categorical crossentropy' is the loss function, which fits this task the best. The classification task is a multi-class classification, which is highly suited to 'Categorical crossentropy'.
- 'Accuracy' is the metric used to measure performance on the validation set upon training the model.

6.4 XGBoost

The XGBoost network builds on 'XGBoost' outlined in section 3.9.

Setting Parameters

This section details the parameters that are set for successful implementation of the XGBoost network.

- The maximum depth of the tree used in the model is set to 10. Increasing the value more than 10 will make the model more complex, leading to overfitting.
- The number of parallel threads used to run XGBoost in this instance is 4.
- The number of classes is set. In regards to the MHEALTH dataset, it is 13.
- The evaluation metric is set to 'merror'. This is multiclass classification error rate. It is the percentage of wrong cases over percentage of all cases.
- The softmax objective is set for XGBoost model, as it is a multiclass classification task.

Network architecture

This section explains the network architecture of the XGBoost network that aims to classify the human physical activity that each volunteer performs.

- First, a dense matrix is formed for the testing data.
- Secondly, a dense matrix is formed for the training data.
- An evaluation list is then formed to compare training and testing outputted results.
- The model is then run taking into account the following parameters 'param', 'dtrain', 'num_round' and 'evallist'.

6.5 LSTM (Recurrent Neural Network)

The LSTM network builds on 'LSTM (Recurrent Neural Network)' discussed in section 3.5.

Network architecture

This section explains the network architecture of the LSTM RNN network that aims to classify the human physical activity that each volunteer performs.

- The first layer of each model provides the 'input_shape' for the model. It is a tuple of integers, containing 23 features.
- The LSTM model has 23 inputs, 2 LSTM layers, 3 hidden layers and an output layer with 1 output.
- The two LSTM layers have 8 and 16 neurons in each one respectively.
- The last output in the output sequence is returned in the two LSTM layers.
- The first hidden layer has 128 neurons.
- The second hidden layer has 256 neurons.

- The third hidden layer has 512 neurons.
- Rectified linear activation functions are used in each of the three hidden layers.
- Dropout regularisation is used on the third and fourth hidden layers to prevent/reduce overfitting. 'Dropout' is set to 0.4.
- The output layer is a dense layer that produces the output. There is 13 neurons in the output layer. It is set to 13 to account for each possible outcome (0-12). The activation function used in the final dense layer is 'softmax'. Softmax e the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.
- A 'softmax' activation function is set in the output layer.

Network layers, optimiser and loss function

This section contains an overview of the LSTMs network layers, optimiser used and loss function implemented.

- The LSTM model is then created that includes an input layer, two LSTM layers, two dropout layers and four dense layers.
- Compiling the model takes three parameters: optimiser, loss function and metrics. The Adam algorithm best fits the data, it is implemented as the optimiser for this convolutional neural network.
- 'Categorical crossentropy' is the loss function, which fits this task the best. The classification task is a multi-class classification, which is highly suited to 'Categorical crossentropy'.
- 'Accuracy' is the metric used to measure performance on the validation set upon training the model.

6.6 Autoencoder by Random Forest

The AE by Random Forest network builds on 'Autoencoder by Random Forest' outlined in section 3.7 and 3.8.

Building the model

This section explains the network architecture of the AE that aims to classify the human physical activity that each volunteer performs using random forests.

- The first layer of each model provides the 'input_shape' for the model. It is a tuple of integers, containing 23 features.

- The AE model has 23 inputs, 5 hidden layers and an output layer with 1 output.
- The first hidden layer has 128 neurons.
- The second hidden layer has 64 neurons.
- The third hidden layer has 8 neurons. This hidden layer that is squashed between two consecutive hidden layers either side. It is an 'encoding' layer as it learns a representation of the data here.
- The fourth hidden layer has 64 neurons.
- The fifth hidden layer has 128 neurons.
- Rectified linear activation functions are used in each of the five hidden layers.
- The output layer is a dense layer that produces the output. There is 23 neurons in the output layer. It is set to 13 to account for each possible outcome (0-22). The output layer implies a linear activation function
- A 'linear' activation function is set in the output layer.

Network layers, optimiser and loss function

This section contains an overview of the AE with RF network layers, optimiser used and loss function implemented.

- The AE model created includes an input layer and six dense layers.
- Compiling the model takes three parameters: optimiser, loss function and metrics. The Adam algorithm best fits the data, it is implemented as the optimiser for this convolutional neural network.
- 'Categorical crossentropy' is the loss function, which fits this task the best. The classification task is a multi-class classification, which is highly suited to 'Categorical crossentropy'.
- 'Accuracy' is the metric used to measure performance on the validation set upon training the model.

6.7 t-Distributed Stochastic Neighbor Embedding (t-SNE)

The t-SNE model builds on the dimensionality reduction technique 't-Distributed Stochastic Neighbor Embedding (t-SNE)' discussed in section 3.12.

For this non-linear dimensionality reduction technique, t-SNE accelerates using the previous Autoencoder with Random Forest model. The Autoencoder with Random Forest algorithm learns the t-SNE embeddings.

Building the AE with Random Forest model

The Network architecture for the AE with RF is the same as previously discussed in section 6.6.

AE with RF Network layers, optimiser and loss function

The network layers, optimiser and loss function for the AE with RF is the same as previously discussed in section 6.6.

Building the t-SNE Model

This section explains the network architecture of the t-SNE model build on the AE with RF.

Step 1: Load previous Autoencoder Random Forest model

Step 2: Learn a representation (encoding) for a set of data.

Step 3: Build an intermediate layer model by getting the models input and output layers.

Step 4: Convert the intermediate layer models predicted features to an array.

Step 5: Split the training and testing data based on these new features.

Step 6: Dump the new features to the autoencoder features.

Step 7: Run t-SNE with defined parameter settings.

Step 8: Fit the t-SNE into the embedded space.

Step 9: Dump the embedded features to the t-SNE features.

6.8 Density-based spatial clustering of applications with noise (DBSCAN)

The DBSCAN model builds on the dimensionality reduction technique 'Density-based spatial clustering of applications with noise (DBSCAN)' discussed in section 3.11.

DBSCAN is an excellent tool for visualising clusters of data. Clusters of data are made up of data points, which hold spatial or temporal values. Upon forming a cluster of data, these spatial and temporal values are assigned a designated data location point within the cluster.

Hyperparameter setting

As mentioned in section 3.11, two values must be set before the algorithm is run:

- *Epsilon = 3*
- *Minimum Points = 60*

DBSCAN Methodology:

In order for DBSCAN to begin clustering, Epsilon and Minimum Points should be set. After setting these hyperparameters, the clustering begins by visiting a point, for example 'Point A'. The Epsilon value aims to find out which data points are neighbours with this 'Point A'. If the data points in the neighbourhood match the Epsilon value and associations are found, clustering begins. Once a point in the neighbourhood is visited, it is marked as visited. Noise plays a role in DBSCAN clustering. Noise outlines which data points are irrelevant in the clustering process. If a point is visited and the relationship isn't satisfactory, isn't less than or equal to Epsilon value, the point is classified as 'Noise'. If a data point is part of a data cluster, its Epsilon neighbour of data points becomes part of the data cluster. This process is repeated until all data points are assigned a cluster or classified as Noise. Evaluating the DBSCAN clusters can begin once all points have been visited.

Evaluating DBSCAN

The *Silhouette Coefficient* measures the accuracy of the DBSCAN Clustering. It measures accuracy based on the tightness and separation of its cluster based on other clusters. Satisfactory clustering is achieved if Silhouette Coefficient lies between -1 and +1.

$$\text{Silhoutte Coefficient} = \frac{b-a}{\max(a,b)}$$

6.9 Approach Conclusion

Chapter 6, approach, discusses in detail the building of each model. Building each model revolves around these six aspects:

- Network Architecture.
- Identifying Network Layers.
- Choosing an Optimiser.
- Choosing the Loss Function.
- Hyperparameter setting.

The next chapter, experiment and results, discusses how each models individual experiment was conducted which includes training the model, hyperparameter setting. A performance comparison of each model is then presented.

7 Experiments

The following chapter outlines the experiments completed to identify the best network architecture and parameters for the classification models as well as their results. The experiments for the two clustering algorithms, DBSCAN and t-SNE, are also discussed to find the superior clustering algorithm. Network architecture, hyperparamter setting, training the model and performance comparison is discussed for each individual model throughout.

Experiments preparation

Analysis is run in: Python, TensorFlow and Keras.

7.1 Convolutional Neural Network

Feature extraction, encoding the labels to one-hot form along with splitting the training and testing data is performed before the model is initialised.

Network architecture

CNN activity recognition overview:

- One input layer containing 23 features.
- Two separable convolution 1D layers with max pooling.
- Three hidden layers that were big enough to train the data well.
- Two dropout layers that yielded positive results.
- Quite good performance, moderately slow.
- One output layer of 12 results (labels).

The neural network contained in the CNN model utilises the data values given for each of the 23 signals recorded in order to classify (predict) our class variable, which is the movement that each subject performs.

Training the model / Hyperparameter setting

This section gives an overview of training the model and hyperparameter setting.

- In the training process, the 'fit()' function is used to train the CNN model.
- 'X_train' represents the training data.
- 'y_train' refers to the target data.
- 'X_test,y_test' represent the validation data.
- The model is trained on a total of 245584 parameters.

While training the model:

- The Learning rate is set as 0.0005.
- Batch size is set as 32.
- The training process is run for 20 epochs.

Algorithm Speed

CNN performed excellently on this classification problem. It processed the data relatively fast. Taking into account the speed of the other deep learning algorithms and considering the performance it achieved. The time the model took to train the data was 242 minutes 18 seconds.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following table depicts a detailed evaluation summarisation of each class, the figures are normalised to the percentage of data.

Activity	Precision	Recall	F1-Score	Support
Null class	89	91	90	174531
Standing still	68	97	80	6081
Sitting and relaxing	72	100	83	6116
Lying down	75	100	86	6195
Walking	75	46	57	6081
Climbing stairs	70	31	43	6168
Waist bends forward	73	63	67	5699
Frontal elevation of arms	77	76	76	5829
Knees bending	81	40	54	5871
Cycling	73	69	71	6109
Jogging	64	71	67	6265
Running	71	75	73	6116
Jump front and back	49	9	16	2088
Accuracy			84	243149
Macro avg	72	67	66	243149
Weighted avg	84	84	83	243149

Table 4 CNN performance comparison

The CNN models performance measures are as follows:

- Accuracy of 84%.
- Precision of 84%.

- Recall of 84%.
- F1-Score of 83%.

7.2 ConvLSTM (Convolutional Neural Network + LSTM)

Feature extraction, encoding the labels to one-hot form along with splitting the training and testing data is performed before the model is initialised.

Network architecture

ConvLSTM activity recognition overview:

- One input layer containing 23 features.
- Two 1D-convolution layers.
- Two 'MaxPooling1D'.
- One LSTM layer.
- Two Dropout layers.
- Four dense layers.
- One output layer of 12 results (labels).

The neural network contained in the ConvLSTM model utilises the data values given for each of the 23 signals recorded in order to classify (predict) our class variable, which is the movement that each subject performs.

Training the model / Hyperparameter setting

This section gives an overview of training the model and hyperparameter setting.

- In the training process, the 'fit()' function is used to train the CNN model.
- 'X_train' represents the training data.
- 'y_train' refers to the target data.
- 'X_test,y_test' represent the validation data.
- The model is trained on 191376 parameters.

While training the model:

- The Learning rate is set as 0.0005.
- Batch size is set as 32.
- The training process is run for 20 epochs.

Algorithm Speed

The ConvLSTM model performed well on the classification problem. The total amount of time the model took to process the data was four hours, around 12 minutes for each epoch. Considering the level of performance and speed of processing, the Hybrid model performed mediocrely well on the data. The CNN and ConvLSTM model yielded relatively similar results.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following table depicts a detailed evaluation summarisation of each class, the figures are normalised to the percentage of data.

Activity	Precision	Recall	F1-Score	Support
Null class	88	89	89	174577
Standing still	68	95	79	6131
Sitting and relaxing	71	100	83	6033
Lying down	75	100	86	6140
Walking	78	25	37	6285
Climbing stairs	73	11	19	6198
Waist bends forward	65	73	69	5640
Frontal elevation of arms	70	83	76	5859
Knees bending	68	58	62	5895
Cycling	68	76	72	6080
Jogging	66	52	58	6074
Running	57	85	68	6173
Jump front and back	0	0	0	2064
Accuracy			83	243149
Macro avg	65	65	61	243149
Weighted avg	83	83	81	243149

Table 5 ConvLSTM performance comparison

The ConvLSTM models performance measures are as follows:

- Accuracy of 83%.
- Precision of 83%.
- Recall of 83%.
- F1-Score of 81%.

7.3 Multilayer Perceptron

Feature extraction, encoding the labels to one-hot form along with splitting the training and testing data is performed before the model is initialised.

Network architecture

MLP activity recognition overview:

- One input layer containing 23 features.

- Four hidden layers that were big enough to train the data well.
- Two dropout layers that benefited the results greatly.
- Top class algorithm that based around the concept of gradient descent.
- One output layer of 12 results (labels).

The neural network contained in the MLP model utilises the data values given for each of the 23 signals recorded in order to classify (predict) our class variable, which is the movement that each subject performs.

Training the model / Hyperparameter setting

This section gives an overview of training the model and hyperparameter setting.

- In the training process, the 'fit()' function is used to train the CNN model.
- 'X_train' represetns the training data.
- 'y_train' refers to the target data.
- 'X_test,y_test' represent the validation data.
- The model is trained on 706317 parameters.

While training the model:

- The Learning rate is set as 0.0005.
- Batch size is set as 32.
- Training process is run for 20 epochs.

Algorithm Speed

MLP performed excellently on this classification problem. It processed the data relatively fast. Taking into account the speed of the other deep learning algorithms and considering the performance it achieved. The time the model took to train the data was 86 minutes 40 seconds.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following table depicts a detailed evaluation summarisation of each class, the figures are normalised to the percentage of data.

Activity	Precision	Recall	F1-Score	Support
Null class	97	90	93	174852
Standing still	79	96	87	6172
Sitting and relaxing	76	97	85	6101
Lying down	79	100	88	6152
Walking	79	93	85	6219
Climbing stairs	78	86	81	6014
Waist bends forward	75	97	84	5558
Frontal elevation of arms	77	96	86	5782
Knees bending	80	94	86	5866
Cycling	76	96	85	6184
Jogging	74	98	84	6128
Running	82	80	81	6077
Jump front and back	62	66	64	2044
Accuracy			91	243149
Macro avg	78	91	84	243149
Weighted avg	92	91	91	243149

Table 6 MLP performance comparison

The MLP models performance measures are as follows:

- Accuracy of 91%.
- Precision of 92%.
- Recall of 91%.
- F1-Score of 91%.

7.4 XGBoost

Feature extraction and splitting the training and testing data is performed before the model is initialised.

Network architecture

XGBoost activity recognition overview:

- Multiclass classification using the softmax activation function.
- Evaluation: multiclass classification error rate.
- Learning rate is set to 0.05.
- Trained on 161959 parameters.

The XGBoost LSTM model utilises the data values given for each of the 24 signals recorded in order to classify (predict) our class variable, which is the movement that each subject performs.

Setting Parameters

The parameters regarding the XGBoost model are discussed in section 6.4.

Training

- The model is trained using the parameter list and the training data.
- The model is trained for 10 rounds.

Early Stopping

- The learning rate of the model is 0.05 while the number of estimators is set to 1000.
- Early stopping is used in the validation set to identify the appropriate amount of boosting rounds. This is usually the optimal number of boosting rounds needed. It is set to stop within 5 rounds. It will train until 'validation_0-merror' hasn't improved in 5 rounds.
- The XGBoost model will train until the 'validation_0-merror' hasn't improved anymore. The 'validation_0-merror' must continue to decrease in order for the 'early_stopping_rounds' to continue to train the XGBoost model.

Algorithm Speed

XGBoost performed very well on this classification problem. It processed the data relatively fast. Taking into account the speed of the other deep learning algorithms and considering the performance it achieved, XGBoost was the second best performing algorithm.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following table depicts a detailed evaluation summarisation of each class, the figures are normalised to the percentage of data.

Activity	Precision	Recall	F1-Score	Support
Null class	93	93	93	174530
Standing still	78	95	85	6298
Sitting and relaxing	79	96	86	6090
Lying down	83	99	90	6090
Walking	82	78	80	6138
Climbing stairs	88	47	61	6194
Waist bends forward	82	78	80	5510
Frontal elevation of arms	82	85	83	5843
Knees bending	84	68	75	6008
Cycling	81	91	86	6081
Jogging	82	90	86	6067
Running	84	91	87	6134
Jump front and back	76	63	69	2166
Accuracy			90	243149
Macro avg	83	83	82	243149
Weighted avg	90	90	90	243149

Table 7 XGBoost performance comparison

The XGBoost models performance measures are as follows:

- Accuracy of 90%.
- Precision of 90%.
- Recall of 90%.
- F1-Score of 90%.

7.5 Long Short-Term Memory

Feature extraction, encoding the labels to one-hot form along with splitting the training and testing data is performed before the model is initialised.

Network architecture

LSTM activity recognition overview:

- One input layer containing 23 features.
- Two LSTM layers.
- Two dropout layers.
- Three hidden layers and one output layer.
- One output layer of 12 results (labels).

The recurrent neural network contained in the LSTM model utilises the data values given for each of the 23 signals recorded in order to classify (predict) our class variable which is the movement that each subject performs.

Training the model / Hyperparameter setting

This section gives an overview of training the model and hyperparameter setting.

- In the training process, the 'fit()' function is used to train the CNN model.
- 'X_train' represetns the training data.
- 'y_train' refers to the target data.
- 'X_test,y_test' represent the validation data.
- The model is trained on 175373 parameters.

While training the model:

- The Learning rate is set as 0.0005.
- Batch size is set as 32.
- Training process is run for 20 epochs.

Algorithm Speed

LSTM performed very poorly on the classification problem. It processed the data very slowly. Upon evaluating all six algorithms, LSTM is the poorest performing algorithm and processes data ten times slower than the other algorithms. The total amount of time the model took to process the data was 10 hours. At first, the LSTM model was set to run for 20 epochs. The hard drive used to conduct each experiment was not strong enough to process data for such a significant amount of time. The RAM kept crashing after 5 hours of processing. The number of epochs was reduced to 4 to ensure the model ran to full capacity on each subject, and all the data was processed (not just a subset). Considering the level of performance and speed of processing, CNN was the poorest algorithm applied to the MHEALTH dataset.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following table depicts a detailed evaluation summarisation of each class, the figures are normalised to the percentage of data.

Activity	Precision	Recall	F1-Score	Support
Null class	84	88	86	174531
Standing still	54	70	61	6081
Sitting and relaxing	68	98	80	6116
Lying down	74	100	85	6195
Walking	69	0	1	6081
Climbing stairs	58	0	1	6168
Waist bends forward	57	58	58	5699
Frontal elevation of arms	64	65	65	5829
Knees bending	60	32	42	5871
Cycling	60	54	57	6109
Jogging	41	25	31	6265
Running	43	86	58	6116
Jump front and back	47	1	2	2088
Accuracy			78	243149
Macro avg	60	52	48	243149
Weighted avg	77	78	75	243149

Table 8 LSTM performance comparison

The LSTM models performance measures are as follows:

- Accuracy of 90%.
- Precision of 90%.
- Recall of 90%.
- F1-Score of 90%.

7.6 Autoencoder by Random Forest

Feature extraction, encoding the labels to one-hot form along with splitting the training and testing data is performed before the model is initialised.

Network architecture

AE with Random Forest activity recognition overview:

- One input layer containing 23 features.
- Four hidden layers that were big enough to train the data well.
- An encoding layer that learns a representation of the data.
- The model is trained on 23,711 parameters.
- One output layer of 12 results (labels).

The neural network contained in the Autoencoder model utilises the data values given for each of the 23 signals recorded in order to classify (predict) our class variable, which is the movement that each subject performs.

Training the model / Hyperparameter setting

This section gives an overview of training the model and hyperparameter setting.

- In the training process, the 'fit()' function is used to train the AE model.
- 'X_train' represents the training data while
- 'y_train' refers to the target data.
- 'X_test,y_test' represent the validation data.
- The model is trained on 23711 parameters.

While training the model:

- The Learning rate is set as 0.0005.
- Batch size is set as 32.
- Training process is run for 20 epochs.

Call features from AE model to apply a Random Forest.

The Autoencoder model produces features that it classifies as correct. The RF model converts the AE model to an array to identify the key features. The predicted new features are then split into training and testing, in a ratio of 80:20. A pickled representation of the new features is wrote to the AE features. A Random Forest classifier is then built. The number of trees is set to '40' while the number of 'processors (n_jobs)' allowed to use is set to '4'. The 'fit()' function is again used to fit the training data. Upon training the data with an Autoencoder with Random Forest, the accuracy is printed.

Algorithm Speed

Autoencoder by random forest performed moderately well on this classification problem. It processed the data relatively fast. Processing time equalled the amount of time that the ConvLSTM model and CNN took. The time the model took to process the data was roughly 45 minutes.

Performance comparison

The generalisation between each subjects activity performed was measured in performance. Performance is measured through precision, recall, f1 score and accuracy. The following figure depicts a detailed evaluation summarisation of each class.

Activity	Precision	Recall	F1-Score	Support
Null class	85	94	89	174531
Standing still	73	61	67	6081
Sitting and relaxing	77	91	83	6116
Lying down	81	95	87	6195
Walking	77	27	40	6081
Climbing stairs	80	22	35	6168
Waist bends forward	79	42	55	5699
Frontal elevation of arms	78	64	70	5829
Knees bending	80	46	58	5871
Cycling	68	47	56	6109
Jogging	73	67	70	6265
Running	76	74	75	6116
Jump front and back	69	13	22	2088
Accuracy			83	243149
Macro avg	77	57	62	243149
Weighted avg	83	83	82	243149

Table 9 Autoencoder by Random Forest performance comparison

7.7 t-Distributed Stochastic Neighbor Embedding (t-SNE)

The following section presents an account of the extended AE with RF algorithm using t-SNE.

Due to the MHEALTH dataset being so large and the computational ability of the computer used to perform the experiment, the t-SNE algorithm was run separately for each subject.

An experiment was conducted on the full dataset but the computer failed to execute the plots as the RAM and memory failed to handle the volume of the output.

The t-SNE experiment is run on each subjects text file separately: mHealth_subject1, mHealth_subject2, mHealth_subject3, mHealth_subject4, mHealth_subject5, mHealth_subject6, mHealth_subject7, mHealth_subject8, mHealth_subject9 and mHealth_subject10.

The following section explains the method in which t-SNE is executed.

Autoencoder with Random Forest

The Autoencoder with Random Forest algorithm learns the t-SNE embedding's, so the network architecture, training the model, setting the hyperparameters for the AE with

RF along with calling features from the AE model to apply a Random Forest is the exact same as previously discussed in section 7.6.

Building the t-SNE Model

This section explains the network architecture of the t-SNE model build on the AE with RF.

- Step 1: Load previous Autoencoder Random Forest model
- Step 2: Learn a representation (encoding) for a set of data.
- Step 3: Build an intermediate layer model by getting the models input and output layers.
- Step 4: Convert the intermediate layer models predicted features to an array.
- Step 5: Split the training and testing data based on these new features.
- Step 6: Dump the new features to the autoencoder features.
- Step 7: Run t-SNE with defined parameter settings.
- Step 8: Fit the t-SNE into the embedded space.
- Step 9: Dump the embedded features to the t-SNE features.

Hyperparameter Setting for t-SNE algorithm

- Test size is set to 0.2.
- Random state is set to 42.
- The number of components is set to 2.
- Perplexity is set to 30.
- The learning rate is set to 150.
- The number of iterations is set to 500.
- The number of iterations without progress is set to 200.

7.8 DBSCAN

The following section outlines the DBSCAN hyperparameter setting and DBSCAN algorithm implementation. Code from [145] and [146] aided in the visualisation of DBSCAN.

Hyperparameter Setting

The following values must be set before the algorithm is run:

- *Epsilon = 3*
- *Minimum Points = 60*

The *Silhouette Coefficient* measures the accuracy of the DBSCAN Clustering. It measures accuracy based on the tightness and separation of its cluster based on other

clusters. Satisfactory clustering is achieved if Silhouette Coefficient lies between -1 and +1 [136].

DBSCAN Algorithm: [137]

Step 1: Epsilon and Minimum Points are set before DBSCAN clustering begins.

Step 2: Begin by visiting a point (point A) which has never been visited.

Step 3: Epsilon aims to find which data points are neighbours with this point A.

Step 4: If the data points in the neighbourhood match the Epsilon value and associations are found, clustering begins.

Step 5: If a point in the neighbourhood is visited, it is marked as visited. If a point is visited and the relationship isn't satisfactory, isn't less than or equal to Epsilon value, the point is classified as 'Noise'.

Step 6: If a data point is part of a data cluster, its Epsilon neighbour of data points becomes part of the data cluster.

Step 7: Step 2 – 6 is repeatedly initiated until all points are assigned a cluster or classified as Noise.

Step 8: All data points in question at the end of DBSCAN will be marked as visited.

Step 9: Silhouette Coefficient is calculated for each cluster. Equation 1 presents the silhouette coefficient formula.

$$\text{Silhoutte Coefficient} = \frac{b-a}{\max(a,b)}$$

Formula 1 Silhouette Coefficient

7.9 Evaluation

The following techniques are used for evaluation.

A list of some common terminology:

- True positives (TP): Predicted positive. Actual result = positive.
- False positives (FP): Predicted positive. Actual result = negative.
- True negatives (TN): Predicted negative. Actual result = negative.
- False negatives (FN): Predicted negative. Actual result = positive.

Accuracy

Accuracy is the ratio of correctly classified predictions over the total amount of input samples.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

Formula 2 Accuracy

Precision

Precision is the total number of positive predictions over the total number of predicted positive instances. Aims to clarify what instance of positive predictions was actually correct.

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

Formula 3 Precision

Recall

Recall is the amount of correctly classified predictions over the total amount of actual positive instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Formula 4 Recall

F1 score

F1 Score is the balance between precision and recall. It takes in account both terms. The higher the F1 score, the higher amount of correctly classified instances. A model with high F1 score will have a high number of positive predictions that are classified as positive, and doesn't exclude how important recall is.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Formula 5 F1 Score

Confusion Matrix

Confusion matrices are excellent for mapping out the performance of a classification problem. Allows for comparison between 'true label' and 'predicted label'. [148] guided writing the confusion matrix code.

Feature Importance

Extracting important feature attribute information about each model presents the following benefits:

1. Assigns a score for each attribute, score is based on the attributes F-Score.
2. Gives key insights into which features influence the model.
3. Gives key insights into which features have little or no impact on the model.
4. Shows which types of sensors have the most effect on the model evaluation.
5. Evaluates performance of each model.
6. Highly compressed graph showing reason for models behaviour.

The following parameters were set to extract the important feature attributes. The number of estimators is set to 100 for the Extreme Gradient Boosting classifier. Training and testing data was split in an 80:20 ratio. When fitting the model, the number of early stopping rounds without progress was set to 5. [147] guided writing the feature importance code.

Next, the plotting of the important feature attributes occurs. Firstly, the model is fitted to the training data. Secondly, the fitted model is converted to a pandas dataframe. Thirdly, the 'feature' and 'importance' columns are set to sort in descending order for plotting purposes. Finally, feature importance is plotted.

Highcharts.js

As mentioned in section 5.8, Highcharts.js is an excellent tool for visualising important feature attributes. Seeing as there are 23 attributes recorded, Highcharts visualises the data more prominently than Python. The following figure is an example of a feature importance graph generated in Python.

The following steps were followed to convert Python output to Highcharts.js:

1. Perform XGBClassifier in Python to extract important feature attributes.
2. Generate Python visualisation as depicted in section 8.3.
3. Repeat steps 1 and 2 for each model: CNN, LSTM, ConvLSTM, XGBoost, MLP and AE with Random Forest.
4. Python output: feature importance for each attribute evaluated by F-Score.
5. Hard code Python output into Highcharts.js to generate visualisations.

8 Results

8.1 DBSCAN Results and Analysis

A total of 1215745 data points were analysed using DBSCAN for all ten subjects. Analysis of data clusters are presented in figures 1-10 below. The figures (1-10) show detected hotspots of activity recognition (movement) in each subject individually.

Subject 1

Figure 25 shows clusters for Subject 1. A total of 161280 data points were reported from Subject 1 and its neighbouring areas. After DBSCAN was performed, 10 clusters are visualised which contained 142976 points in total. 18304 data points were classified as noise. The Silhouette Coefficient is 0.073.

88.65% of the data points generated by Subject 1 can only be used for effective visualisations and possibly in classifying the activities in question.

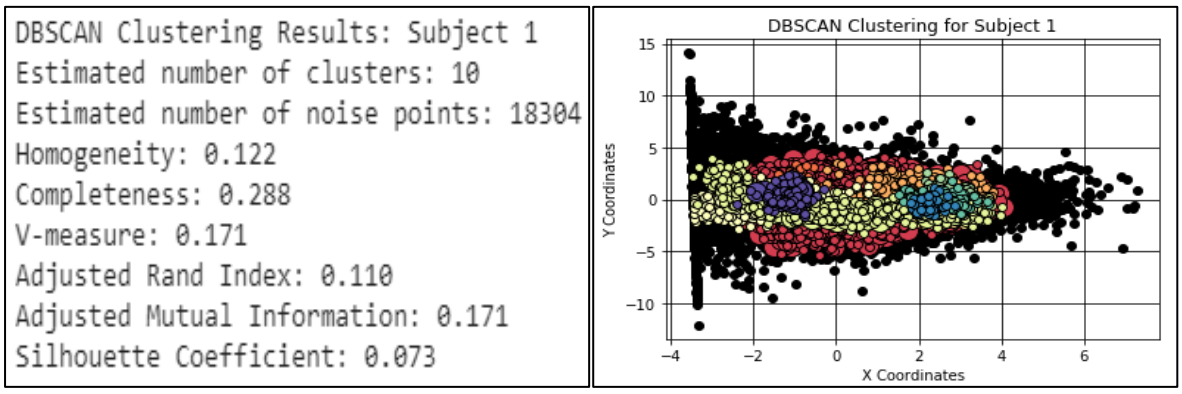

Figure 25 DBSCAN clustering subject 1

Subject 2

Figure 26 shows clusters for Subject 2. A total of 130561 data points where reported from Subject 2 and its neighbouring areas. After DBSCAN was performed, 11 clusters are visualised which contained 118921 points in total. 11640 data points were classified as noise. The Silhouette Coefficient is 0.087.

90.21% of the data points generated by Subject 1 can only be used for effective visualisations and possibly in classifying the activities in question.

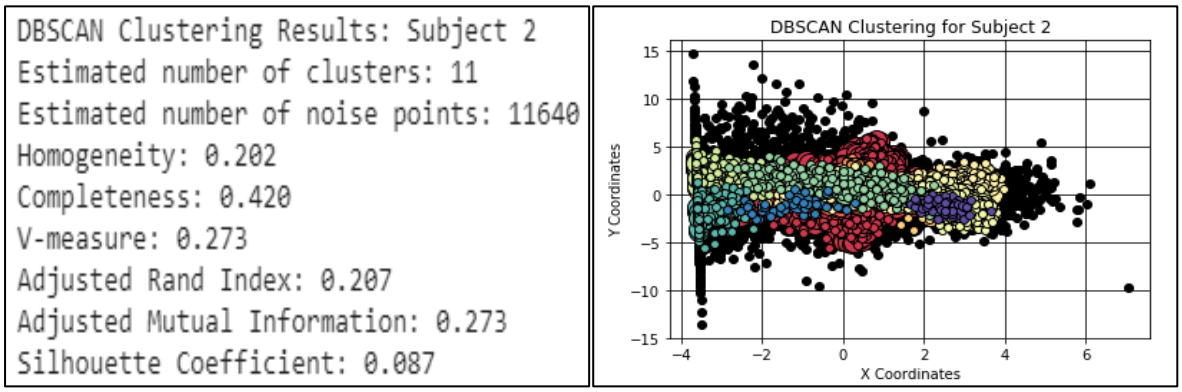

Figure 26 DBSCAN clustering subject 2

Subject 3

Figure 27 shows clusters for Subject 3. A total of 122112 data points where reported from Subject 3 and its neighbouring areas. After DBSCAN was performed, 3 clusters are visualised which contained 106931 points in total. 15181 data points were classified as noise. The Silhouette Coefficient is 0.255.

87.56% of the data points generated by Subject 3 can only be used for effective visualisations and possibly in classifying the activities in question.

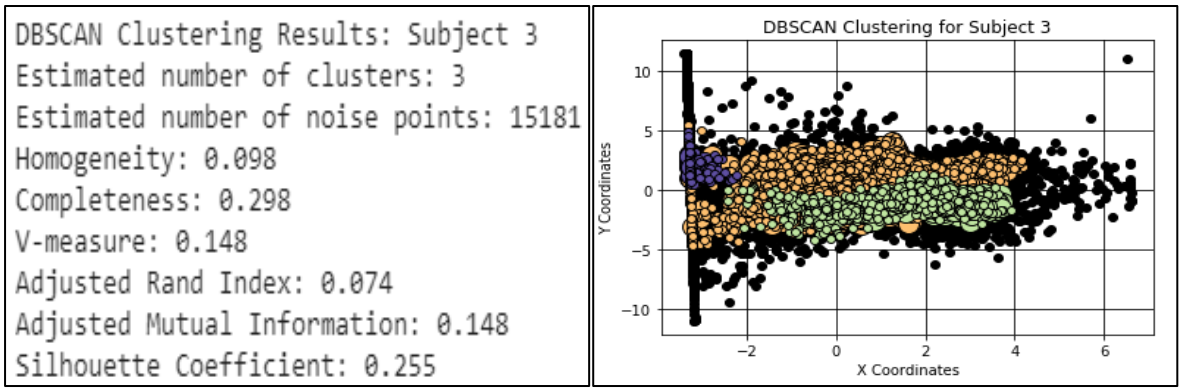

Figure 27 DBSCAN clustering subject 3

Subject 4

Figure 28 shows clusters for Subject 4. A total of 116736 data points where reported from Subject 4 and its neighbouring areas. After DBSCAN was performed, 10 clusters are visualised which contained 103238 points in total. 13498 data points were classified as noise. The Silhouette Coefficient is 0.255.

88.43% of the data points generated by Subject 4 can only be used for effective visualisations and possibly in classifying the activities in question.

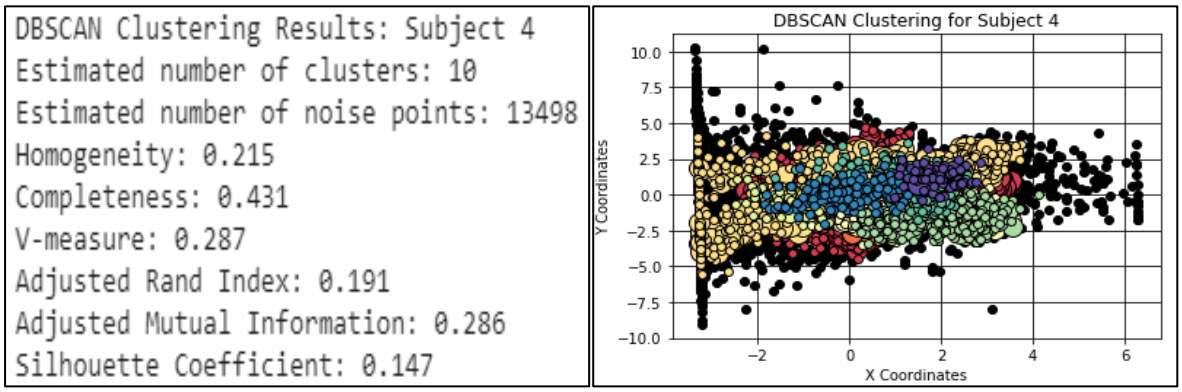

Figure 28 DBSCAN clustering subject 4

Subject 5

Figure 29 shows clusters for Subject 5. A total of 119808 data points where reported from Subject 5 and its neighbouring areas. After DBSCAN was performed, 5 clusters are visualised which contained 109054 points in total. 10754 data points were classified as noise. The Silhouette Coefficient is 0.189.

91% of the data points generated by Subject 5 can only be used for effective visualisations and possibly in classifying the activities in question.

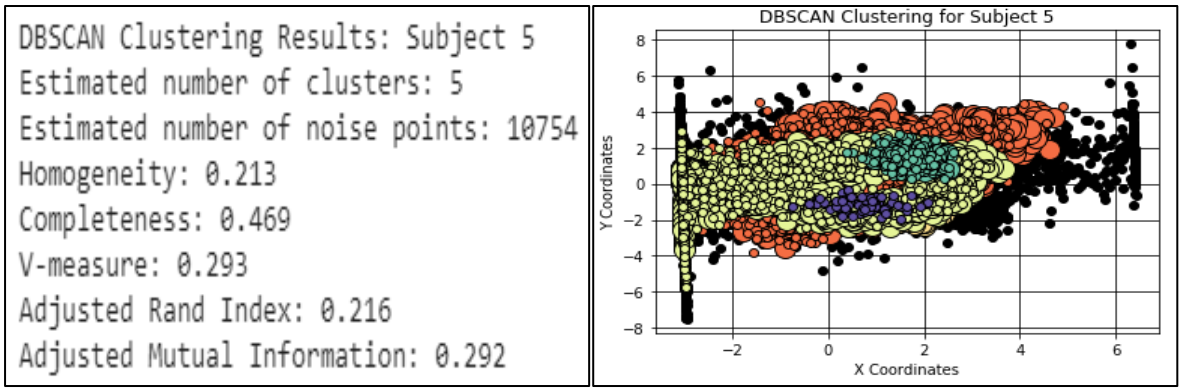

Figure 29 DBSCAN clustering subject 5

Subject 6

Figure 30 shows clusters for Subject 6. A total of 98304 data points where reported from Subject 6 and its neighbouring areas. After DBSCAN was performed, 6 clusters are visualised which contained 87761 points in total. 10543 data points were classified as noise. The Silhouette Coefficient is 0.229.

89.27% of the data points generated by Subject 6 can only be used for effective visualisations and possibly in classifying the activities in question.

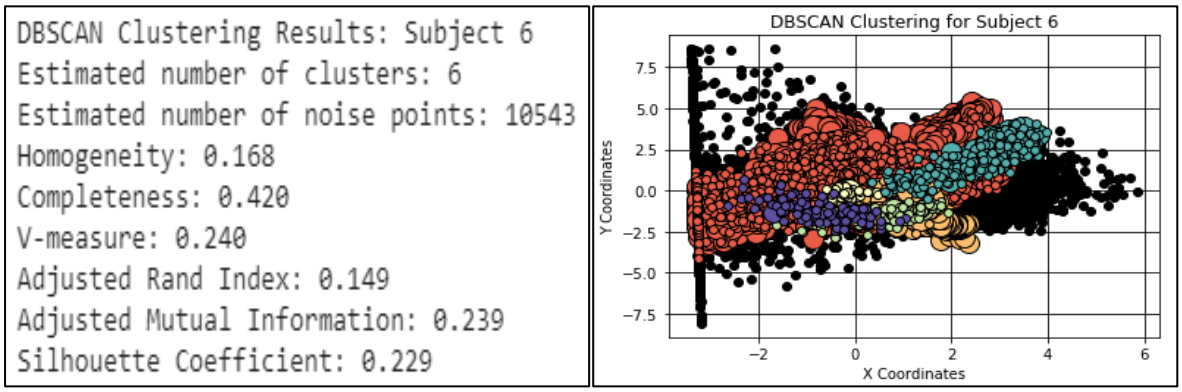

Figure 30 DBSCAN clustering subject 6

Subject 7

Figure 31 shows clusters for Subject 7. A total of 104448 data points where reported from Subject 7 and its neighbouring areas. After DBSCAN was performed, 7 clusters are visualised which contained 95486 points in total. 8962 data points were classified as noise. The Silhouette Coefficient is 0.265.

91.41% of the data points generated by Subject 7 can only be used for effective visualisations and possibly in classifying the activities in question.

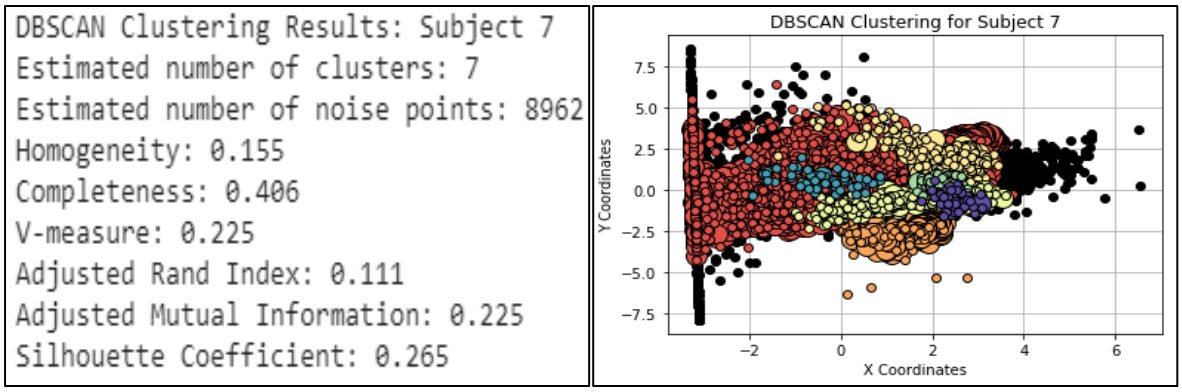

Figure 31 DBSCAN clustering subject 7

Subject 8

Figure 32 shows clusters for Subject 8. A total of 129024 data points where reported from Subject 8 and its neighbouring areas. After DBSCAN was performed, 7 clusters are visualised which contained 114888 points in total. 14136 data points were classified as noise. The Silhouette Coefficient is 0.306.

89.04% of the data points generated by Subject 8 can only be used for effective visualisations and possibly in classifying the activities in question.

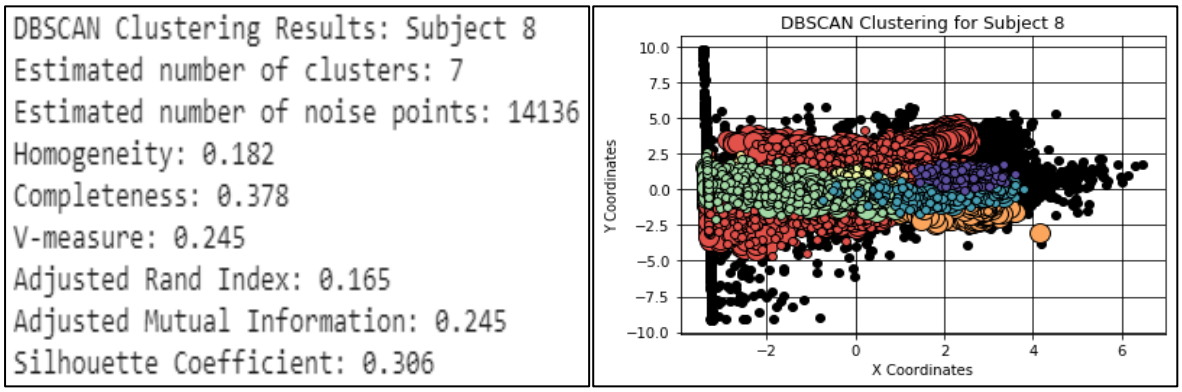

Figure 32 DBSCAN clustering subject 8

Subject 9

Figure 33 shows clusters for Subject 9. A total of 135168 data points where reported from Subject 9 and its neighbouring areas. After DBSCAN was performed, 5 clusters are visualised which contained 119866 points in total. 15302 data points were classified as noise. The Silhouette Coefficient is 0.213.

88.67% of the data points generated by Subject 9 can only be used for effective visualisations and possibly in classifying the activities in question.

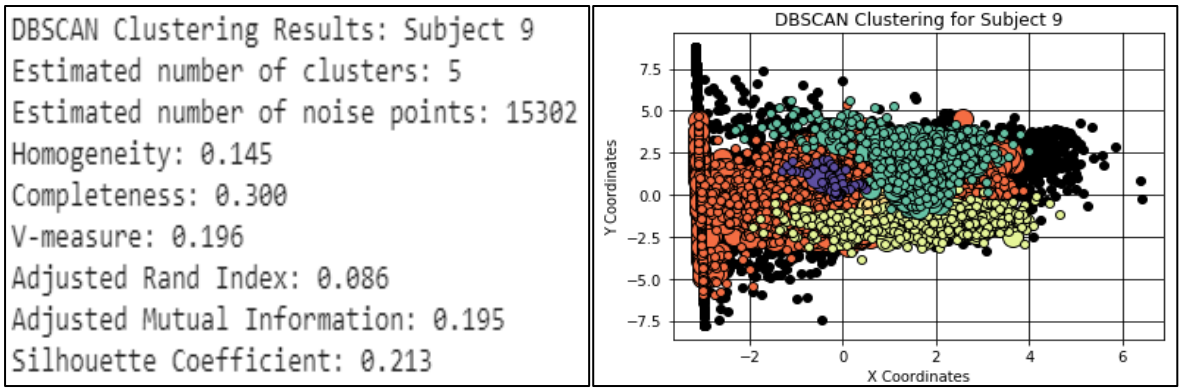

Figure 33 DBSCAN clustering subject 9

Subject 10

Figure 10. Shows clusters for Subject 10. A total of 98304 data points where reported from Subject 10 and its neighbouring areas. After DBSCAN was performed, 8 clusters are visualised which contained 90845 points in total. 7459 data points were classified as noise. The Silhouette Coefficient is 0.222.

92.41% of the data points generated by Subject 10 can only be used for effective visualisations and possibly in classifying the activities in question.

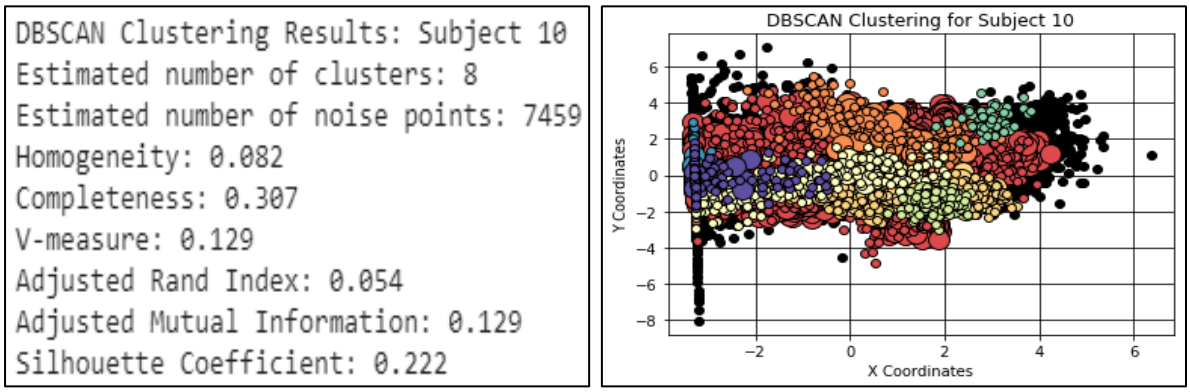

Figure 34 DBSCAN clustering subject 10

DBSCAN Key Points

Subject 8 achieves the maximum silhouette value, which is 0.306.

Subject 2 presents the highest number of clusters detected, which is 11.

The accuracy of the results are quite high with the minimum 87.56% and the maximum accuracy 92.41%. Parameter settings could be altered, perhaps changing the value of Epsilon and the minimum points could be presented as future work to try increase the accuracy even more. Results are very sensitive to changing parameter values so this will take time.

Table 10 presents the DBSCAN clustering outputs silhouette coefficient results.

Subject	Number of clusters	Silhouette Coefficient
Subject 1	10	0.073
Subject 2	11	0.087
Subject 3	3	0.255
Subject 4	10	0.147
Subject 5	5	0.292
Subject 6	6	0.229
Subject 7	7	0.265
Subject 8	7	0.306
Subject 9	5	0.213
Subject 10	8	0.222

Table 10 Silhouette results

8.2 t-SNE Results and Analysis

Table 11 outlines the number of data samples involved in the data-clustering algorithm, t-SNE, for each subject. Each subjects' accuracy after Autoencoder by Random Forest was performed is also given. As mentioned in section 7.7, t-SNE was conducted on each subject separately due to computational reasons. As seen by the table below, accuracy ranges from 83% to 86%, with all subjects achieving moderately similar accuracy. KL-divergence values range from 4.20 to 4.63.

Subject	Number of data samples	Autoencoder by Random Forest Accuracy	KL-divergence
1	161280	85.34%	4.49
2	130561	85.28%	4.63
3	122112	85.28%	4.6
4	116736	83.22%	4.57
5	119808	85.26%	4.51
6	98304	86.42%	4.24
7	10448	84.01%	4.26
8	129092	83.98%	4.56
9	135168	85.95%	4.62
10	98304	85.46%	4.2

Table 11 t-SNE Results

The following hyperparameters were set after Autoencoder by Random Forest was performed to prepare the dataset for the clustering process:

- Number of components is set as 2.
- Random state is set as 0.
- Perplexity is set as 30.
- Learning rate is set as 150.
- Number of iterations is set as 500.

t-SNE clustering for subject 1 and subject 2 are presented by figure 35.

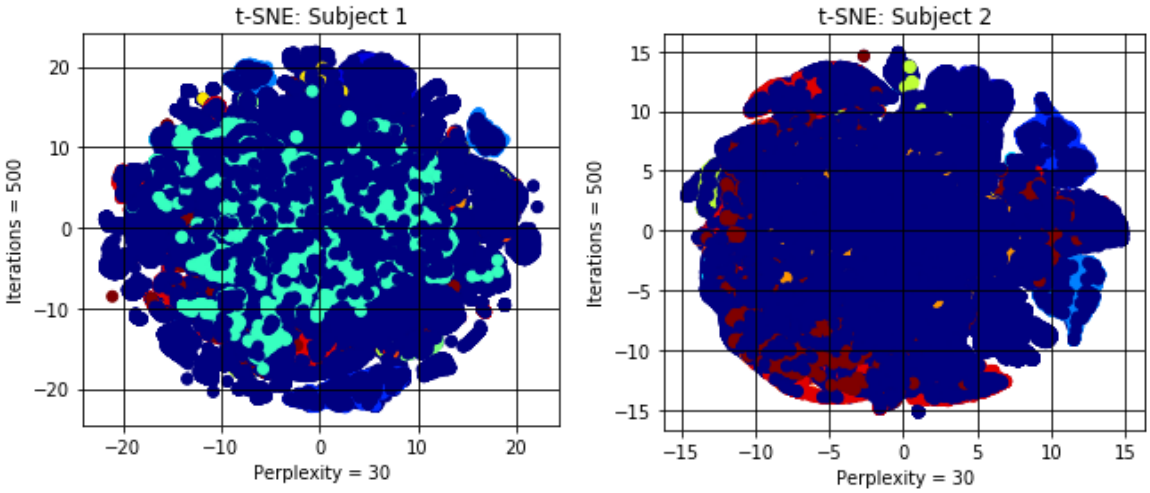

Figure 35 t-SNE subject 1 and subject 2

t-SNE clustering for subject 3 and subject 4 are presented by figure 36.

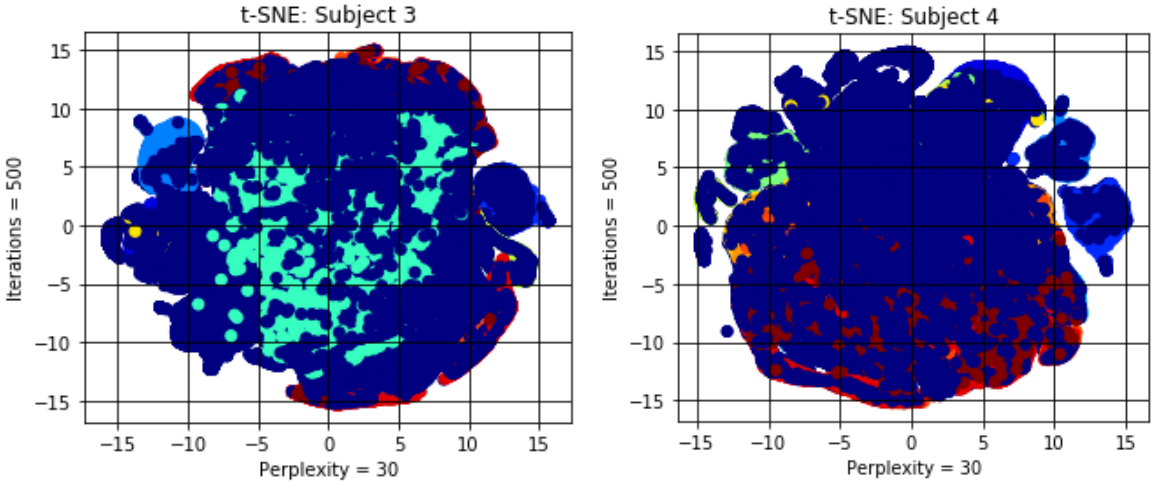

Figure 36 t-SNE subject 3 and subject 4

t-SNE clustering for subject 5 and subject 6 are presented by figure 37.

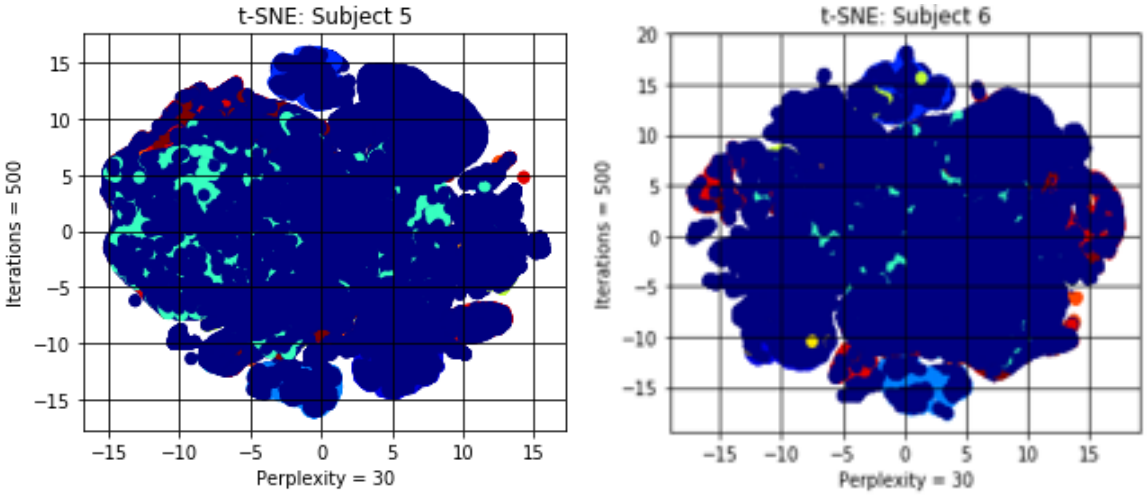

Figure 37 t-SNE subject 5 and subject 6

t-SNE clustering for subject 7 and subject 8 are presented by figure 38.

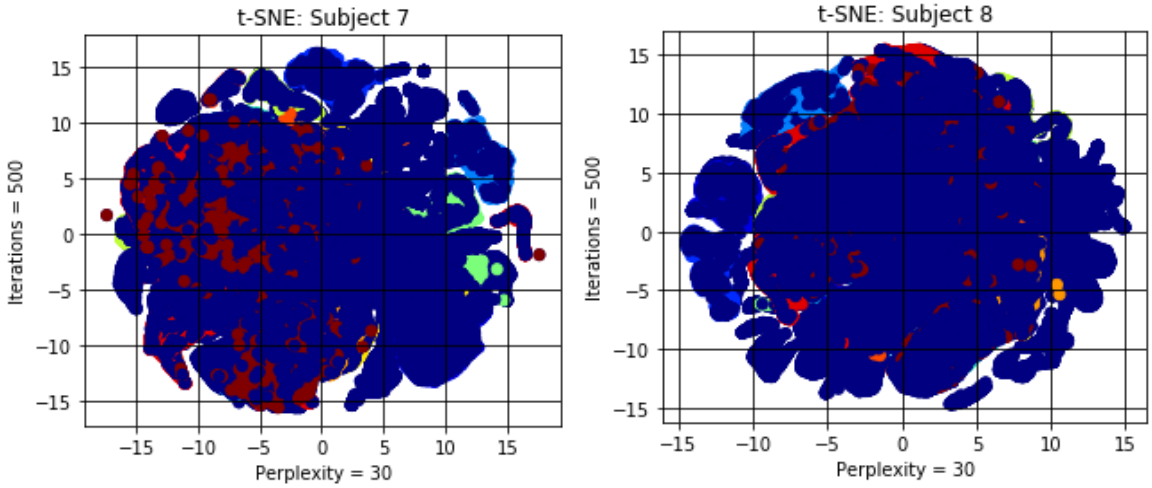

Figure 38 t-SNE subject 7 and subject 8

t-SNE clustering for subject 9 and subject 10 are presented by figure 39.

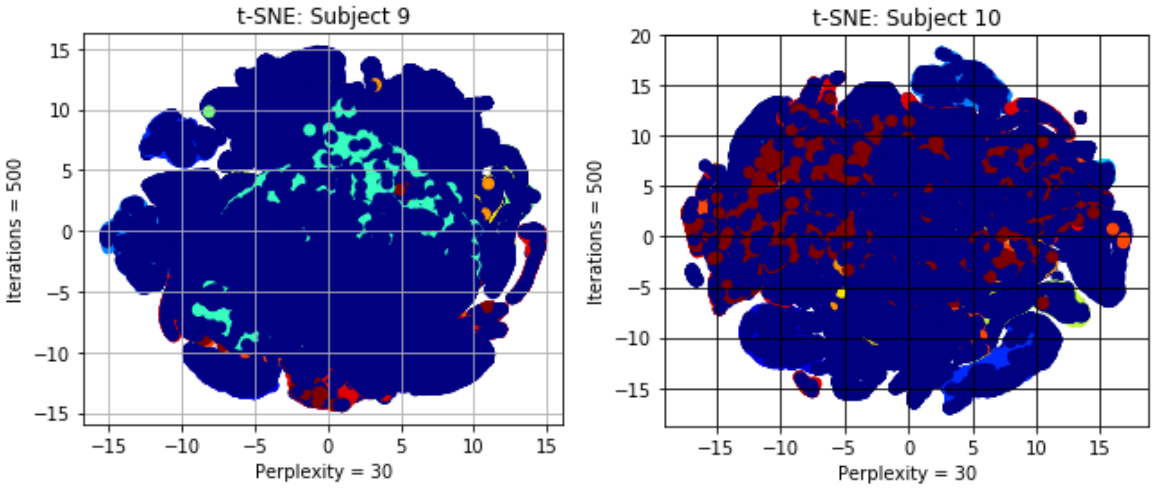

Figure 39 t-SNE subject 9 and subject 10

t-SNE Key Points

Each subjects experiment yields relatively similar results.

The t-SNE output for each subject is a strange 'ball' with uniformly distributed points. All of the points seem to overlap.

Typical value set for perplexity is between 5 and 50. The perplexity for this experiment was set at 30 given the sheer volume of the dataset.

As seen by all ten figures above for each subject, the data is condensed between -17.5 and +17.5.

This experiment concludes that data is clustered very tightly together, although it spreads across a wide range of space (-17.5, 17.5).

t-SNE is clearly not a good choice for data visualisation here as it is difficult to visualise the clusters.

The blue points on each graph seem to spread across the whole cluster, making it difficult to visualise the exact clusters of the variables in question. The 'Null Class' is having a huge effect on the t-SNE clustering output.

Kullback-Leibler divergence

The Kullback-Leibler divergence should have a distinct association to the t-distributed stochastic neighbour embedding. The KL-divergence measures the differences between probability distribution P and probability distribution Q. P is the true distribution while Q is the predicted distribution. For t-SNE to distinguish a good embedding, the KL-divergence must be minimised between probability distribution P and probability distribution Q. This minimised results conclude that the distance between P and Q in the embedded space and mapping space lead to identical probability. Therefore, the overall structure of the data in the embedded space is intact during the mapping of the data points.

As seen in table 12, the KL-divergence is quite high for all subjects. It should be closer to 0 to present a good embedding. This is why the data clusters for each subject are difficult to read. The distance between P and Q is too large, there is little or no identical probability. Therefore, the structure of the data in the embedded space is not intact during the mapping of the data points.

Subject	KL-divergence
1	4.49
2	4.63
3	4.6
4	4.57
5	4.51
6	4.24
7	4.26
8	4.56
9	4.62
10	4.2

Table 12 kullback-leibler divergence

t-SNE Hyperparameter Adjustments

For all ten experiments, hyperparameters such as perplexity, number of iterations and learning rate were adjusted. Hyperparameter adjustment leads to more insight into possible data clusters of the mhealth dataset and provide accurate results.

The following hyperparameters adjustments were made for each subjects experiment.

Adjustment 1:

- Perplexity = 5
- Number of iterations = 250
- Learning rate = 50

Adjustment 2:

- Perplexity = 30
- Number of iterations = 500
- Learning rate = 150

Adjustment 3:

- Perplexity = 50
- Number of iterations = 2500
- Learning rate = 250

Results are written up on experiments implementing adjustments for adjustments 2. Adjustments 1 and 3 did not seem to have a large effect. The data clusters visualised for these adjustments were relatively similar to adjustment 2. The KL-divergence for each adjustment was between 4.1 and 4.9, reiterating the fact that the data structure was not intact during the mapping of the data points.

For all 30 experiments, (10 experiments corresponding to each subject for each hyperparameter adjustment) the blue points dominated the outputted clusters. The blue points were consistently visualised across multiple iterations with these hyperparameter settings. It wasn't just a coincidence, why did this occur? One or many of the blue points are significantly close to every cluster, different coloured clusters are clearly shown across all ten subject experiments, but clearly inspecting them is impossible. A huge early exaggeration amplified this relationship with the other clusters. The Null Class played a huge role in amplifying this relationship, due to the vast amount of data points represented by the Null class. t-SNE output can be skewed, harmed and difficult to interpret if there is the presence of the Null Class in the dataset.

8.3 Performance Visualisations

Figure 40 compares the accuracy of the proposed machine learning approaches. MLP attains the highest accuracy with 90.6%, followed by XGBoost with 90%. CNN achieves 83.9%, ConvLSTM achieves 83.9%, AE w. RF 83.3% while LSTM achieves 78.1%

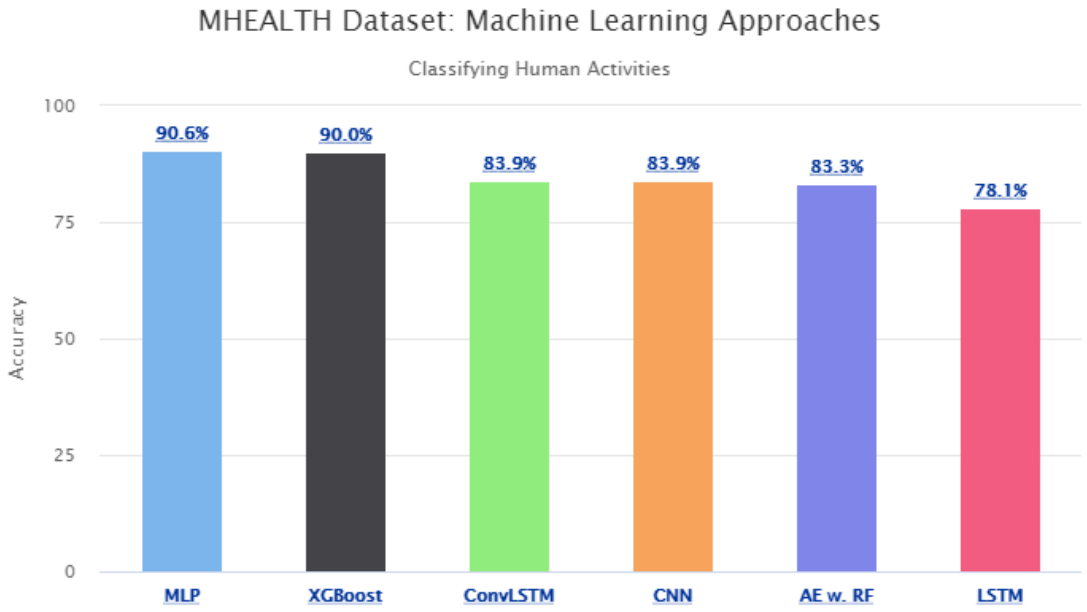

Figure 40 Models comparison

Figure 41 outlines the accuracy of the MLP, XGBoost and CNN machine learning models on the mhealth dataset. The figures are extracted from the models related confusion matrix. MLP attains the highest values, followed by MLP, then CNN.

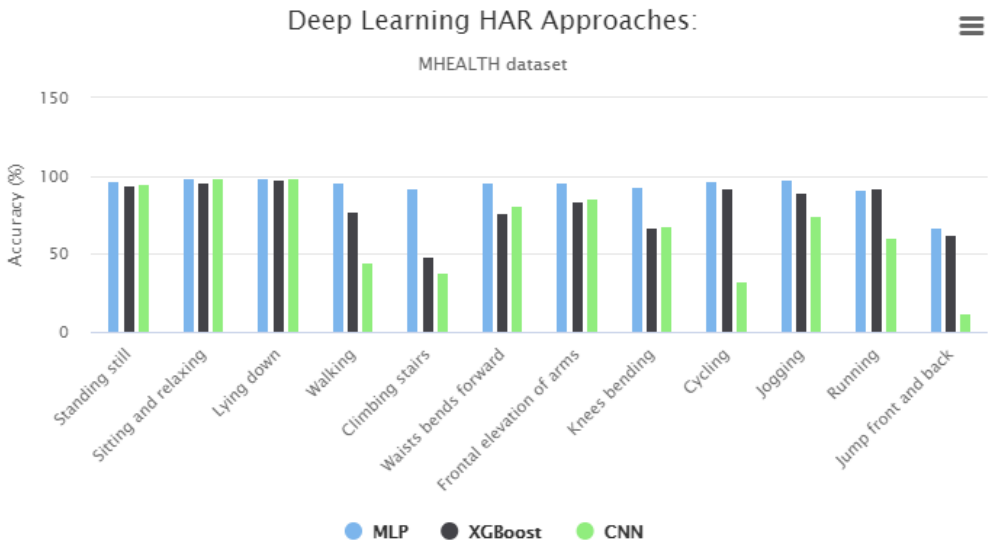

Figure 41 Accuracy comparison

Figure 42 compares the precision of the proposed machine learning approaches. MLP attains the highest precision with 91.7%, followed by XGBoost with 90.1%. CNN achieves 83.9%, ConvLSTM achieves 83.7%, AE w. RF 83.5% while LSTM achieves 74.9%.

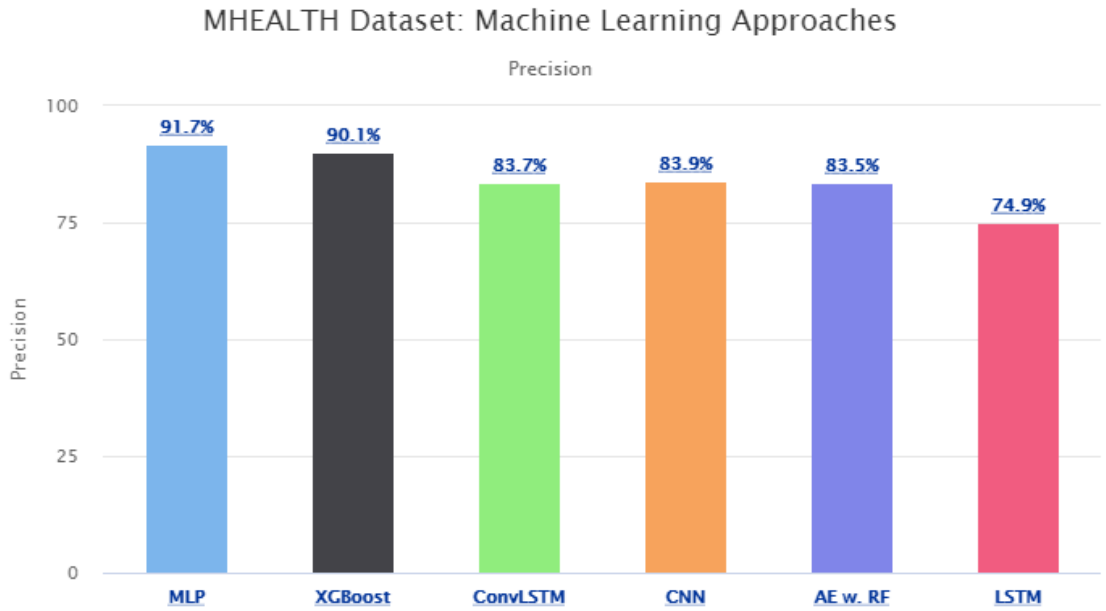

Figure 42 Precision comparison

Figure 43 compares the recall of the proposed machine learning approaches. MLP attains the highest recall with 90.6%, followed by XGBoost with 90%. CNN achieves 83.9%, ConvLSTM achieves 83.9%, AE w. RF 83.9% while LSTM achieves 78.1%

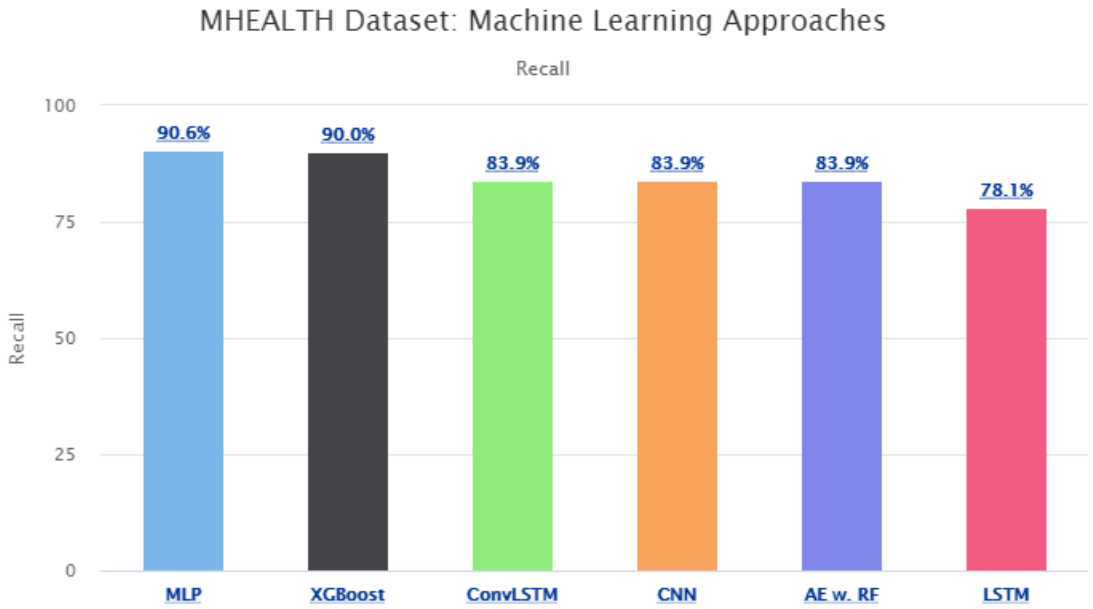

Figure 43 Recall comparison

Figure 44 compares the F1-Score of the proposed machine learning approaches. MLP attains the highest F1-Score with 90.7%, followed by XGBoost with 89.8%. CNN achieves 83%, ConvLSTM achieves 83.2%, AE w. RF 81.5% while LSTM achieves 75.6%

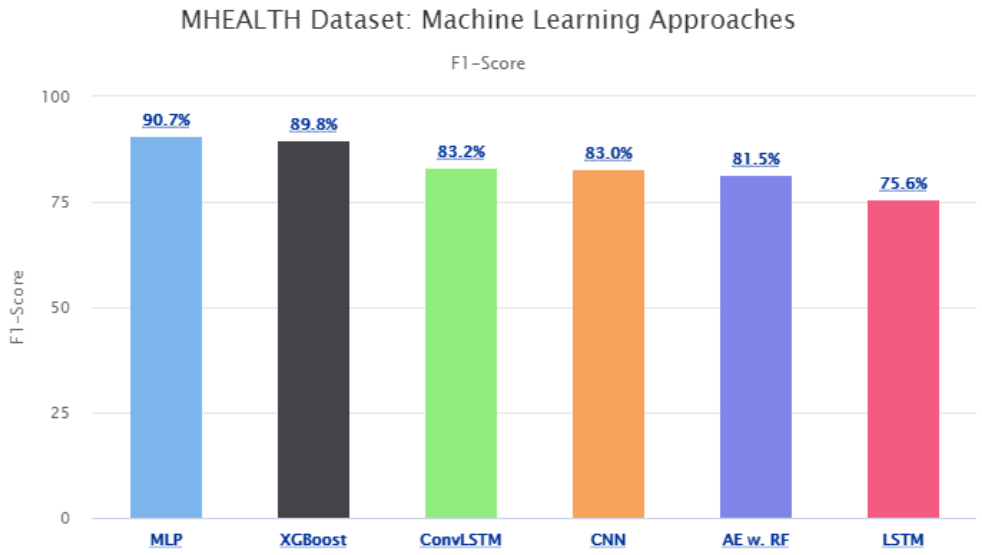

Figure 44 F1-Score comparison

Figure 45, compares the accuracy, precision, recall and F1-Score of the proposed machine learning approaches. MLP attains the highest percentage of the four performance comparison measures, achieving 90% or greater. XGBoost falls slightly short of the top spot but still achieving excellent results, achieving 89% or greater. ConvLSTM, CNN and AE w. RF achieve relatively similar results in the 82% to 84% range. LSTM achieves the poorest results.

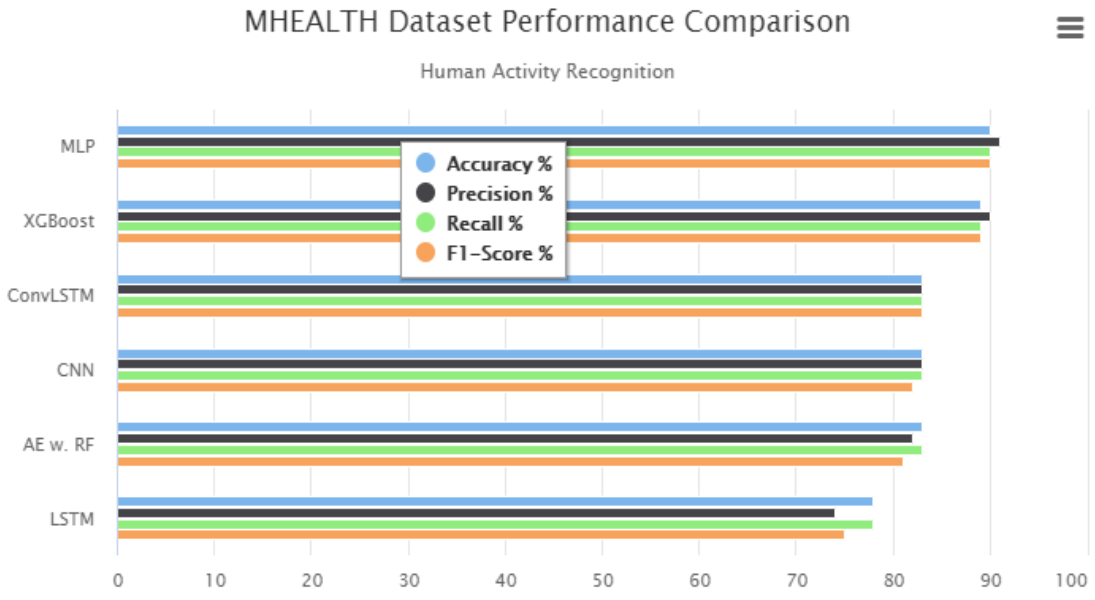

Figure 45 performance comparison

Confusion Matrix Performance of each classification model (%)

The following table examines the performance of each deep learning model on the classification task, the figures are taken from each approaches confusion matrix output. MLP performs the best in comparison to the other models. The CNN and LSTM models achieved an average performance of 66.2% and 48.9% respectively. The Hybrid and XGBoost model achieved an average performance of 70.6% and 81.25% respectively. For each subject, the MLP model attained the best performance obtaining an average performance of 93.5%. The table identifies the superiority of the MLP model over others by showing a significantly higher performance.

Activity	*CNN*	*LSTM*	*Hybrid*	*XGBoost*	*MLP*
Standing still	95	63	96	94	97
Sitting and relaxing	100	97	99	96	99
Lying down	100	100	99	98	100
Walking	45	0	66	77	96
Climbing stairs	38	0	32	48	92
Waist bends forward	81	53	77	76	96
Frontal elevation of arms	86	70	85	84	96
Knees bending	68	38	66	67	93
Cycling	33	42	74	92	97
Jogging	75	47	80	89	98
Running	61	68	70	92	91
Jump front and back	12	9	4	62	67
Average	**66.2**	**48.9**	**70.6**	**81.25**	**93.5**

Table 13 confusion matrix perfornance

Accuracy, Precision, Recall, f1 score of each DL architecture.

Table 14 presents accuracy, precision, recall and F1-Score of each proposed approach. MLP achieves the highest scores, which are highlighted in the table.

Architecture	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*f1 score*
MLP	**90.55**	**91.66**	**90.55**	**90.7**
CNN	83.91	83.47	83.91	82.98
LSTM	78.09	74.86	78.09	75.6
Hybrid	83.89	83.69	83.89	83.2
AE RF	83.27	82.59	83.25	81.54
XGB	89.97	90.09	89.97	89.78

Table 14 Performance Comparison

8.4 Accuracy and Loss Results and Analysis

The following section presents the accuracy and loss results for each approach along with discussion on the results.

Training

The evaluation techniques for each model are the accuracy metric and categorical cross entropy (loss) metric. Each model performed differently, with each model outperforming others in different aspects regarding classifying the data. The accuracy and loss plots for each model are displayed in figures 46-50. Accuracy give an insight into which models performed well while loss gives an insight into which models performed poorly. Each model was run for a total of 20 epochs. Twenty epochs was chosen as the correct amount to perform on the data as the accuracy and loss plots produced clearly show which models were able to converge after a certain amount of epochs. The learning rate was set to '0.05' for each experiment. ReLu activation function was implemented on the machine learning models. The batch size was set to 32 for all experiments.

Accuracy and Loss Results

Fine-tuning each model to correctly classify the data is significantly important to maximising accuracy while minimising loss. MLP, XGBoost, CNN, LSTM, Hybrid and AE with random forest are fine-tuned with hyperparameters respectively. The loss and accuracy of each model is visualised throughout this section.

After fine-tuning each model and using a total of 20 epochs, MLP prevailed as the network with the highest accuracy (91%) and minimal loss. The average accuracy for each model is relatively high. MLP and XGBoost performed better and more consistently than CNN, LSTM, Hybrid and AE with RF. MLP and XGBoost achieved accuracies of 91% and 89% respectively. MLP and XGBoost were able to converge far more easily as outlined in figure 46. Figure 46 depicts the Multilayer Perceptron model achieving 91% accuracy.

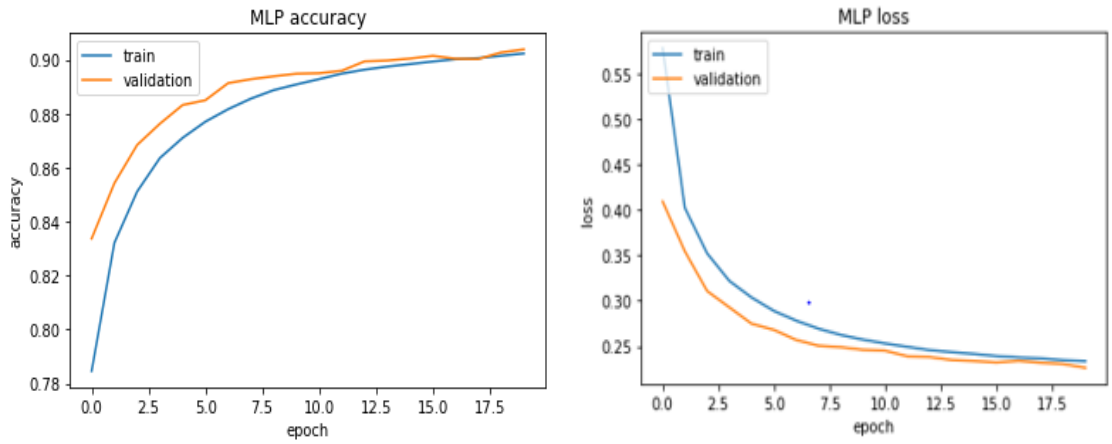

Figure 46 MLP accuracy and loss

The following two plots depict the Convolutional Neural Network model achieving 84% accuracy.

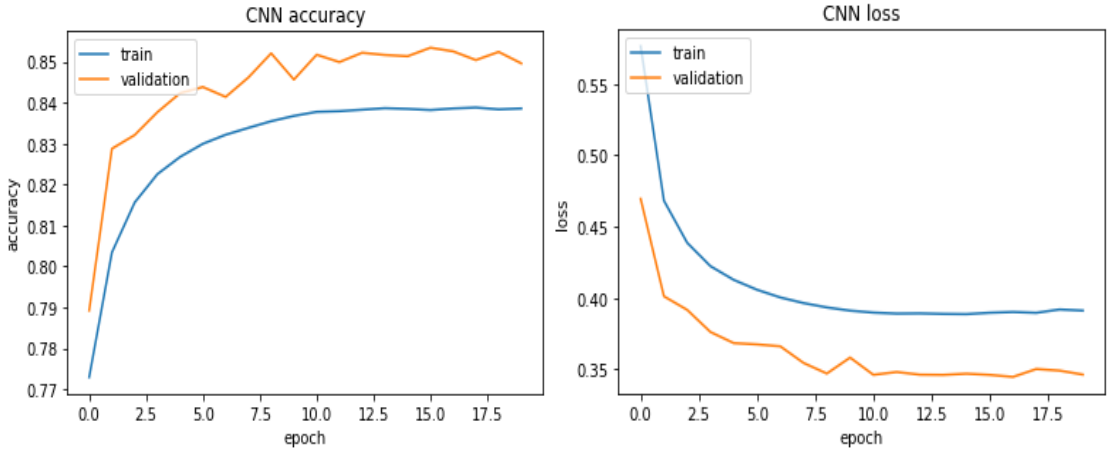

Figure 47 CNN accuracy and loss

CNN performed moderately well in terms of accuracy (84%) and loss. MLP and XGBoost increased their accuracy along with diminishing their loss as the number of training iterations increased. Overall, MLP outperformed the other networks with highest accuracy and lowest loss. The LSTM model misclassified too many instances, which lead it to becoming the poorest performing model with the least accuracy (78%) and highest loss.

LSTM and ConvLSTM networks performed poorly with 20 training iterations as depicted in Figures 48 and 49. They both achieved 78% and 84% accuracy respectively. The following two plots depict the LSTM model achieving 78% accuracy.

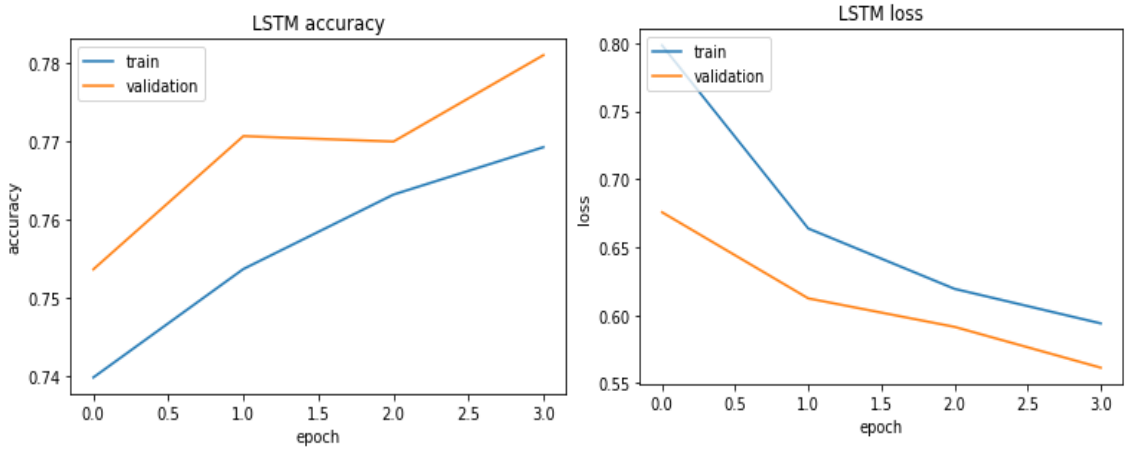

Figure 48 LSTM accuracy and loss

Figure 49 depicts the ConvLSTM model achieving 84% accuracy.

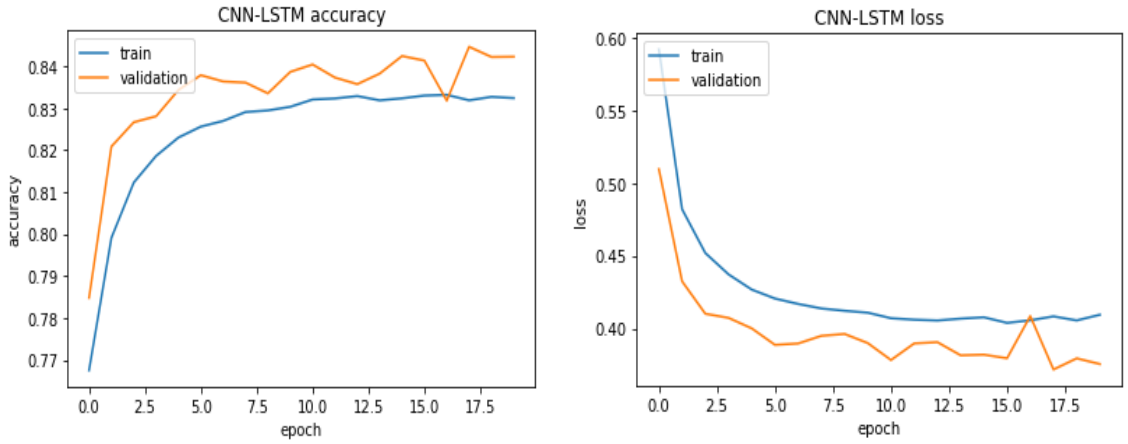

Figure 49 ConvLSTM accuracy and loss

Figure 50 depicts the AE with Random Forest model achieving 84% accuracy.

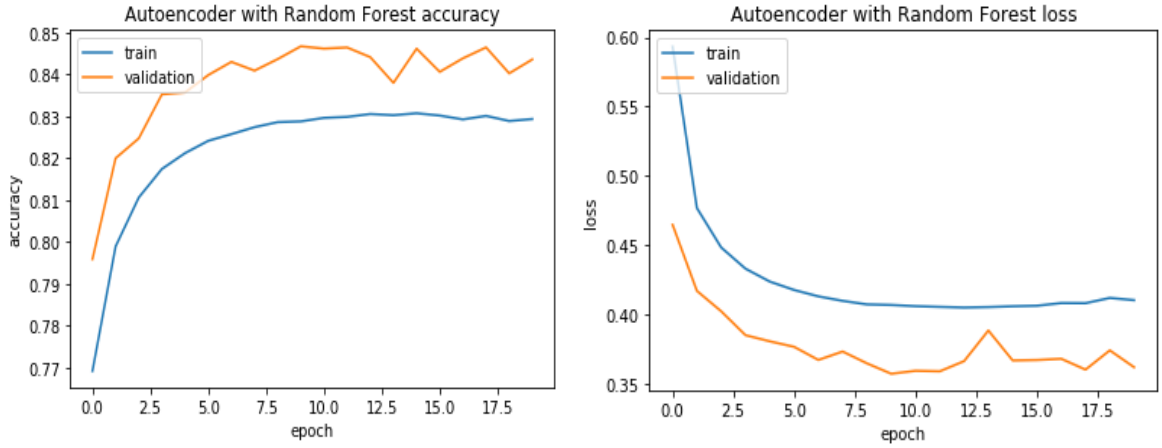

Figure 50 Autoencoder by Random Forest accuracy and loss

8.5 Confusion Matrix Analysis

The confusion matrixes on the MHEALTH dataset for the human activity recognition task are illustrated in figures 52-56. Figure 1 illustrates MLP while figure 2 illustrates XGBoost. Figures 3, 4 and 5 illustrate CNN, LSTM and Hybrid respectively. These confusion matrixes contain data regarding actual and predicted activity classification conducted by each deep learning model. The activities in the horizontal axis represent the models predicted activities while the activities in the vertical axis represent the models actual activities. The matrixes describe in detail what activities are misclassified, as well as the depth in which each misclassification occurred. A single cell represents the percentage that the activity in the row is predicted as the activity in

the column. There is a significant class imbalance due to the presence of the Null class. In order to conduct appropriate analysis of each confusion matrix, the null class will be ignored in order to gain a greater understanding of the data.

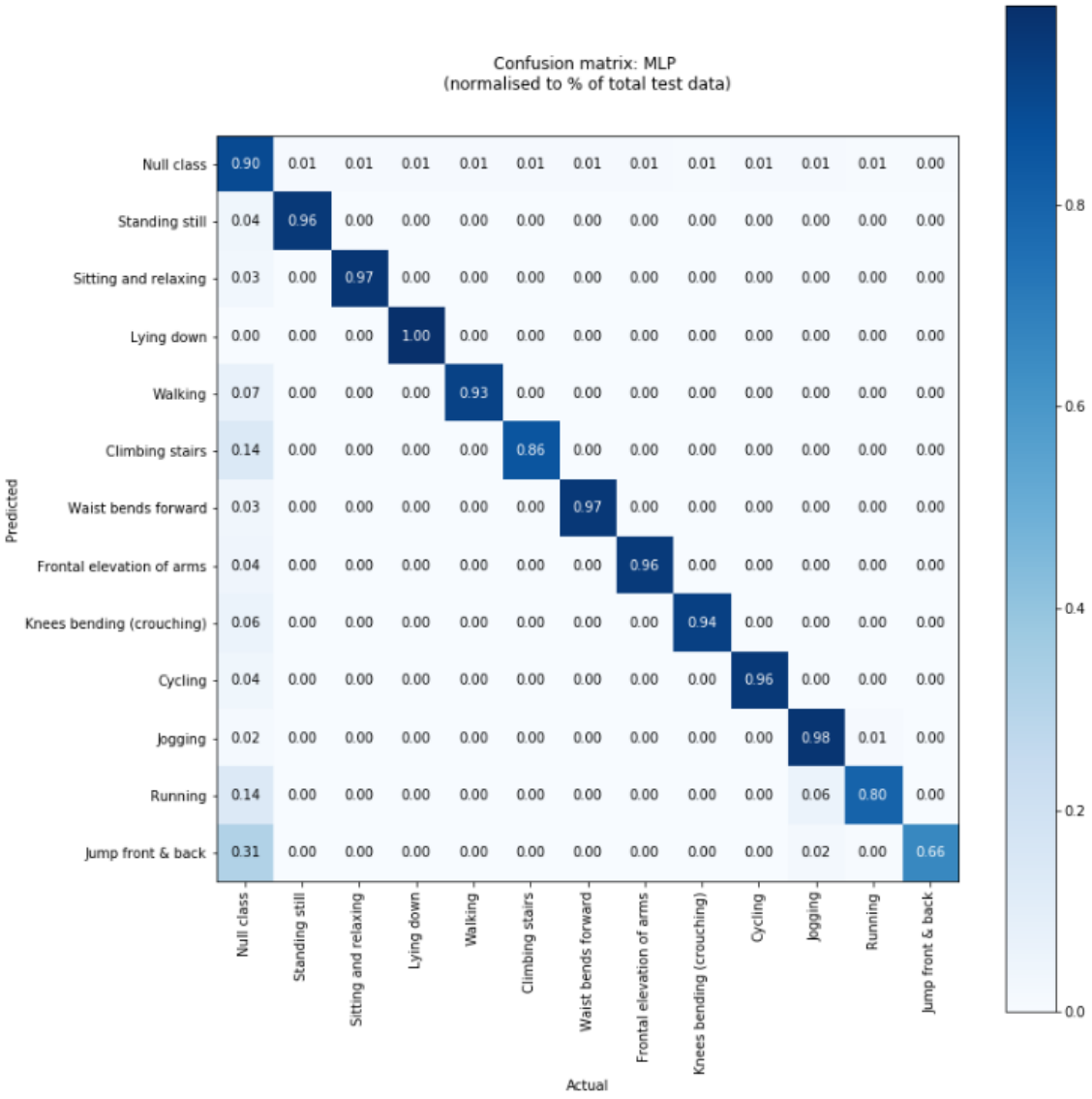

Figure 51 MLP confusion matrix

As seen by each confusion matrix, the null class has a significant contribution on the amount of false positives and false negatives detected. It accounts for a large portion of misclassified activities. Including the Null class in the analysis results leads to a high percentage of data processed as 'not an activity of significant interest' or 'not a classifiable activity' in terms of the labels (activities).

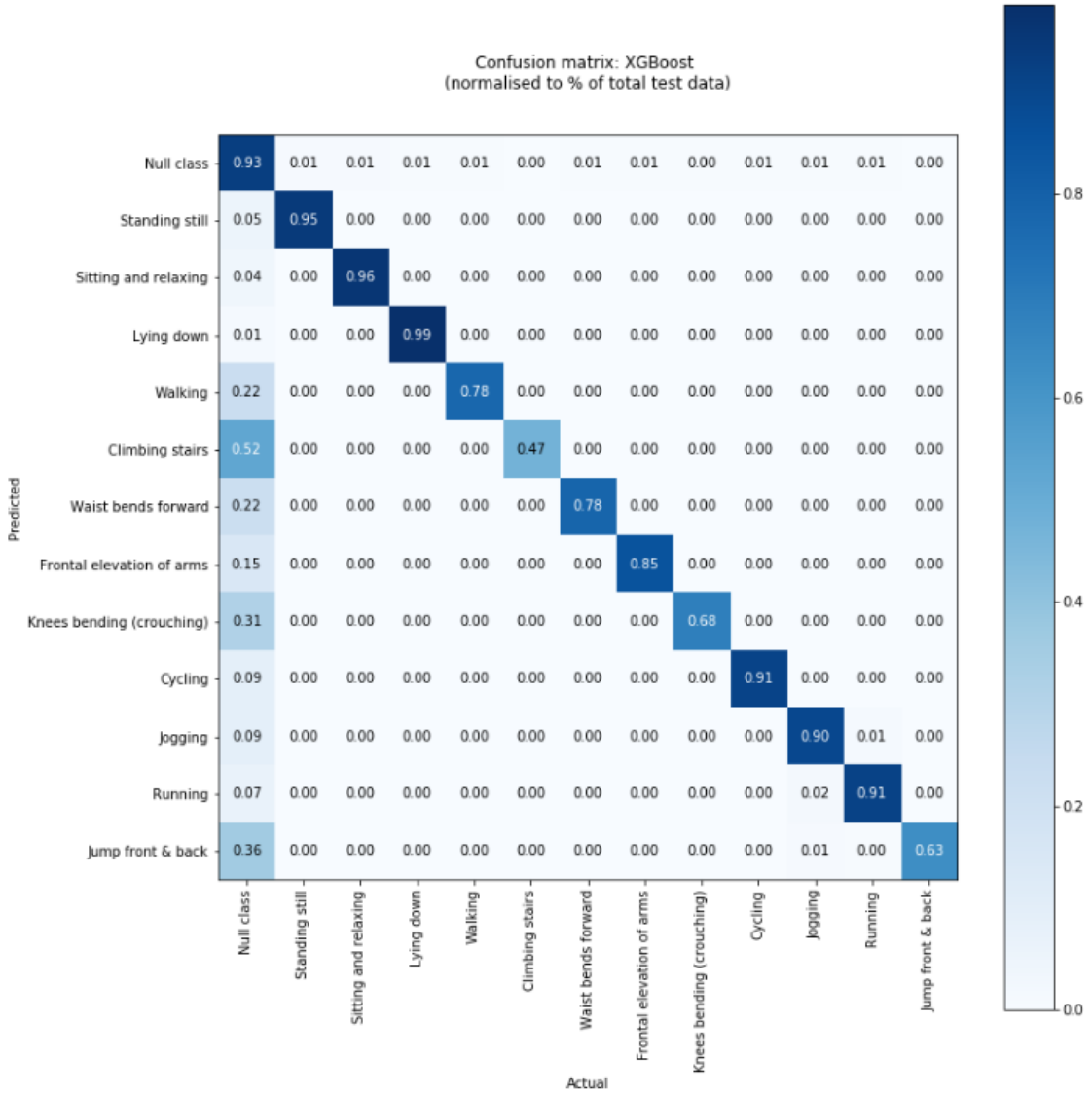

Figure 52 XGBoost confusion matrix

Excluding the null class allows for thorough analysis and appropriate evaluation of results. Every model misclassifies activities that are alike, such as 'running' and 'jogging'. This is due to the activation of similar sensors leading to the generation of similar signals being produced. The MLP model slightly struggles in distinguishing between the jogging activity and running activity with 7% of the activity misclassified. Autoencoder by Random forest misclassifies 7% of the jogging and running activities, misinterpreting both. XGBoost outperforms MLP in this aspect as it misclassifies 3% of the activities between jogging and running. A solution to this would be to place an extra sensor on the thigh in order to distinguish between both activities. This sensor would lead to detection in a greater acceleration and range of motion in the upper thigh area when the subject is running in comparison to jogging. The trick is to monitor when both activities activate the sensors, but with an uncommon sequentially.

In the classification of the activity 'jogging' and 'running', jogging is misclassified as running or vice versa 425 times by the MLP model, 186 times by the XGBoost model, 1006 times by the CNN model, 3221 times by the LSTM model and 2058 times by the Hybrid model.

The better performance of XGBoost in classifying these activities and producing fewer errors is due to the following reasons:

- Effective tree pruning.
- Regularisation.
- Parallel Processing.

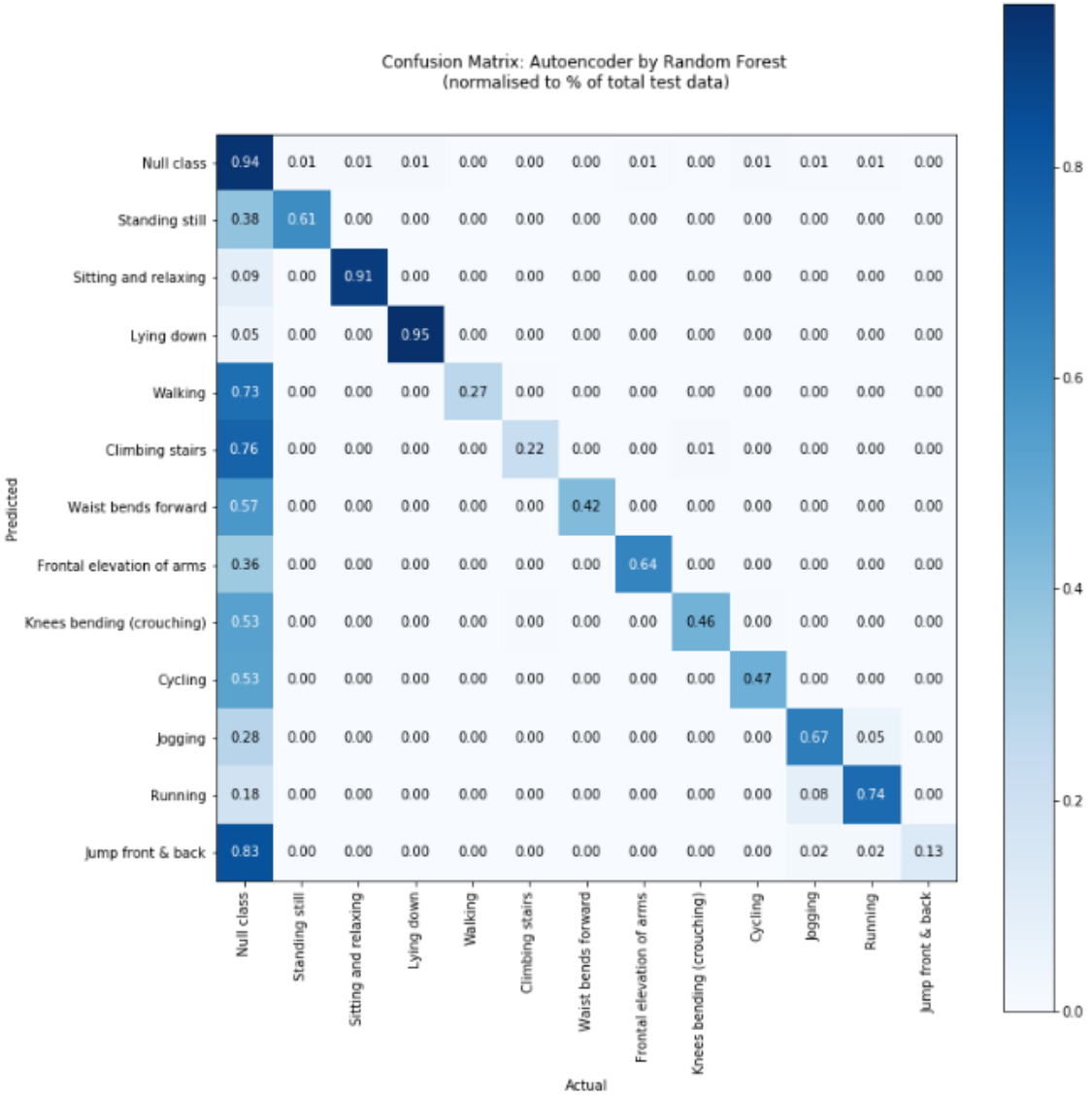

Figure 53 Autoencoder by random forest confusion matrix

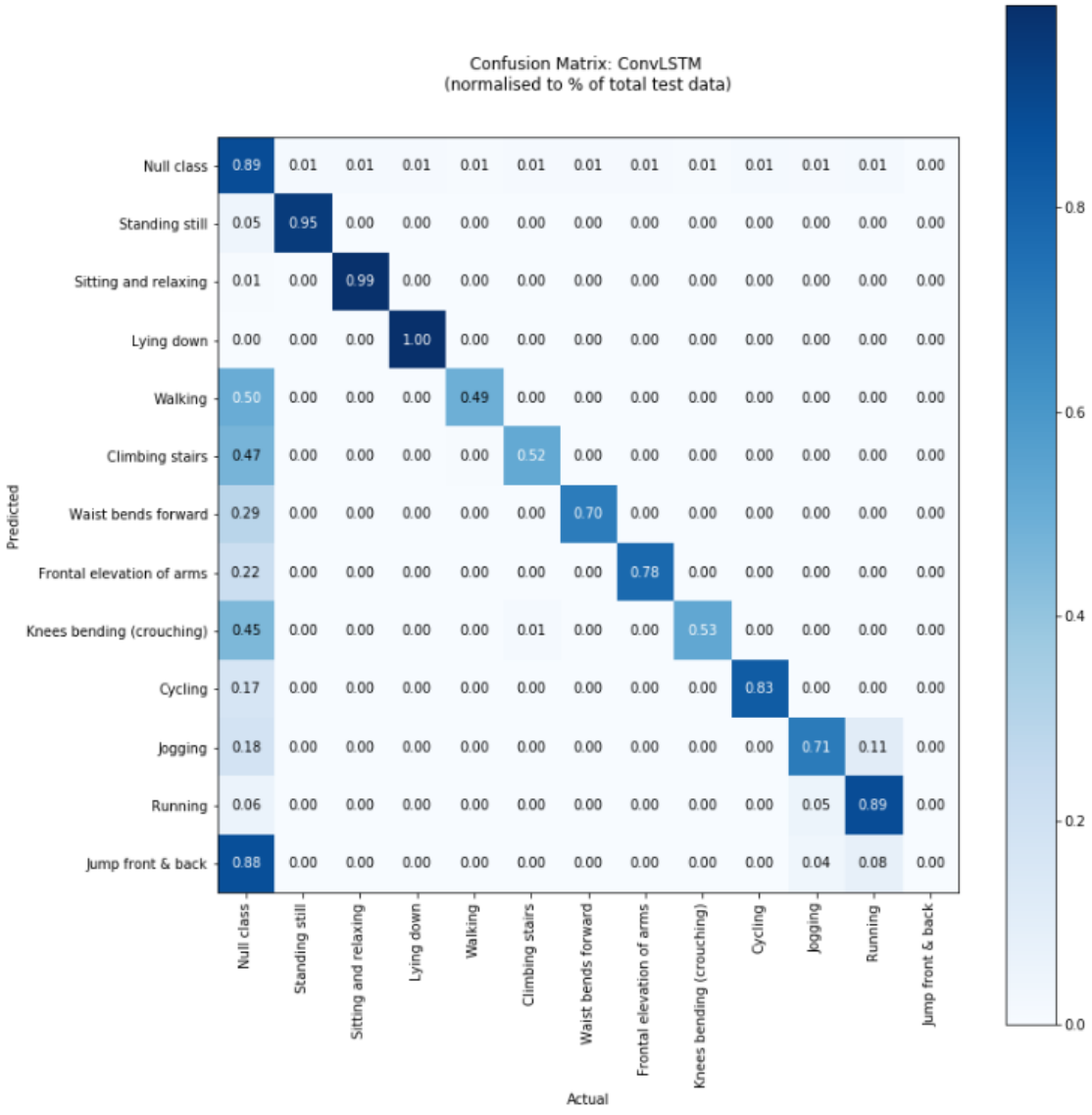

Figure 54 ConvLSTM confusion matrix

Similarly, for the activities 'jumping front and back' and jogging, XGBoost and MLP made little errors with 19 and 46 errors respectively. Autoencoder misclassified 2% of these activities, producing 86 errors. CNN made 102 errors while the LSTM model made 447 errors and Hybrid model made 81. The enhanced performance of MLP for these activities is down to the fact that MLP is very flexible when mapping input to output, it can correctly classify any type of data due to its ability to convert the data into a long row, which is suited to time series data. The activation of sensors makes it difficult to distinguish between jumping front and back and jogging here due to similar range of motion, acceleration and magnetic field orientation. Placing an extra sensor on the subjects' thigh is a valid solution here again. This sensor would detect a greater movement in the leg as jumping front and back leads to a greater acceleration and gravitational pull that jogging would not produce.

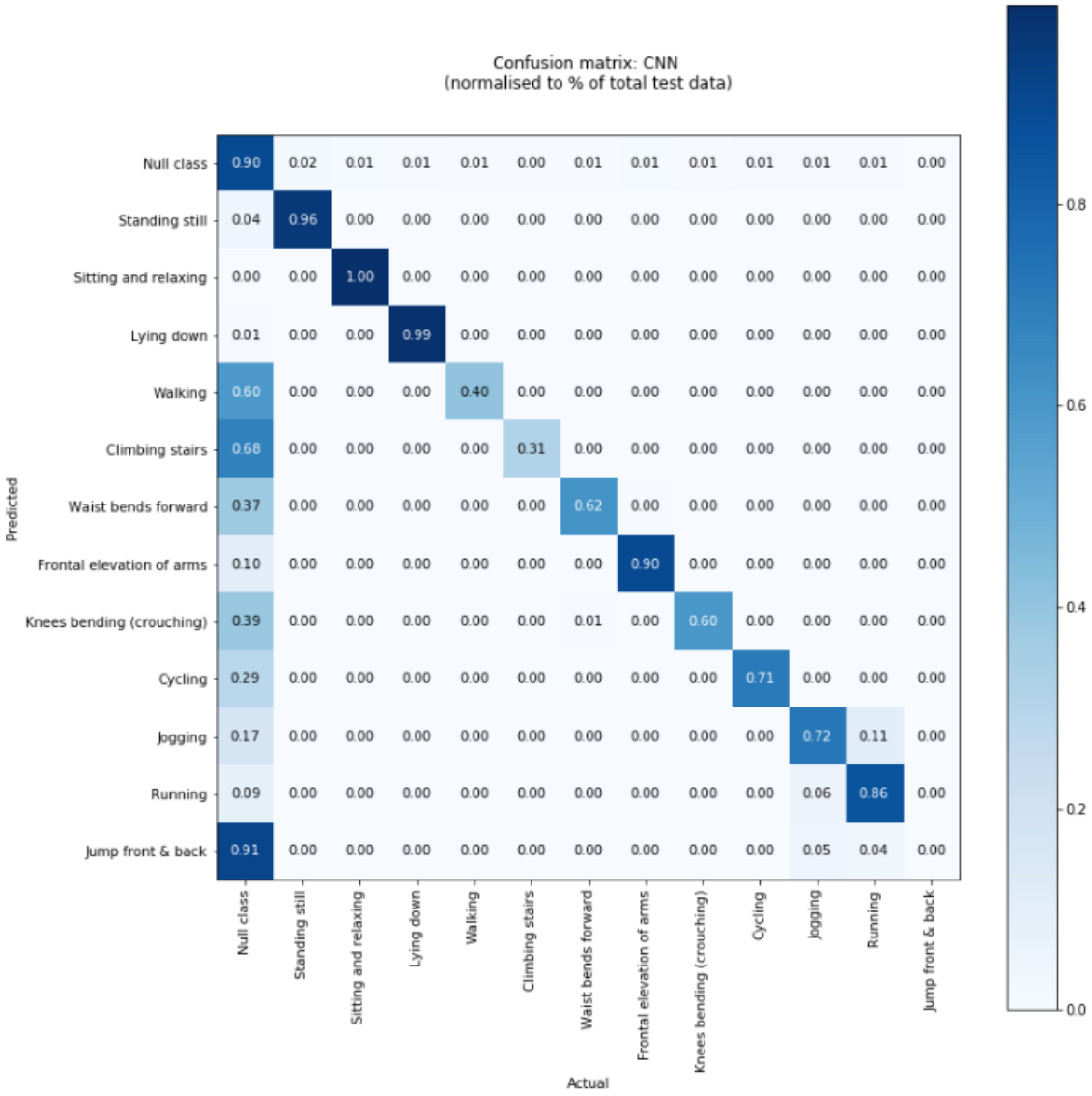

Figure 55 CNN confusion matrix

MLP and XGBoost yielded no more errors. 'Climbing stairs' and 'knees bending (crouching)' are similar activities that were misclassified in the CNN, LSTM, Autoencoder by Random forest and Hybrid models. In the case of the activities 'climbing stairs' and 'knees bending (crouching)' one is predicted as the other 34 times by the CNN model and 94 times by the LSTM model while the Hybrid model made 78 errors. The Autoencoder by Random Forest made 61 errors. Both activities involve similar bending of the knees. 'Climbing stairs' leads to a greater swing of motion in the left and right arm so a solution around this is viable. Placing an extra sensor on a certain part of the arm would lead to the activation of sensors that would be able to distinguish if the subject was climbing stairs or crouching. In a sense of crouching, this extra signal would produce little or no activation.

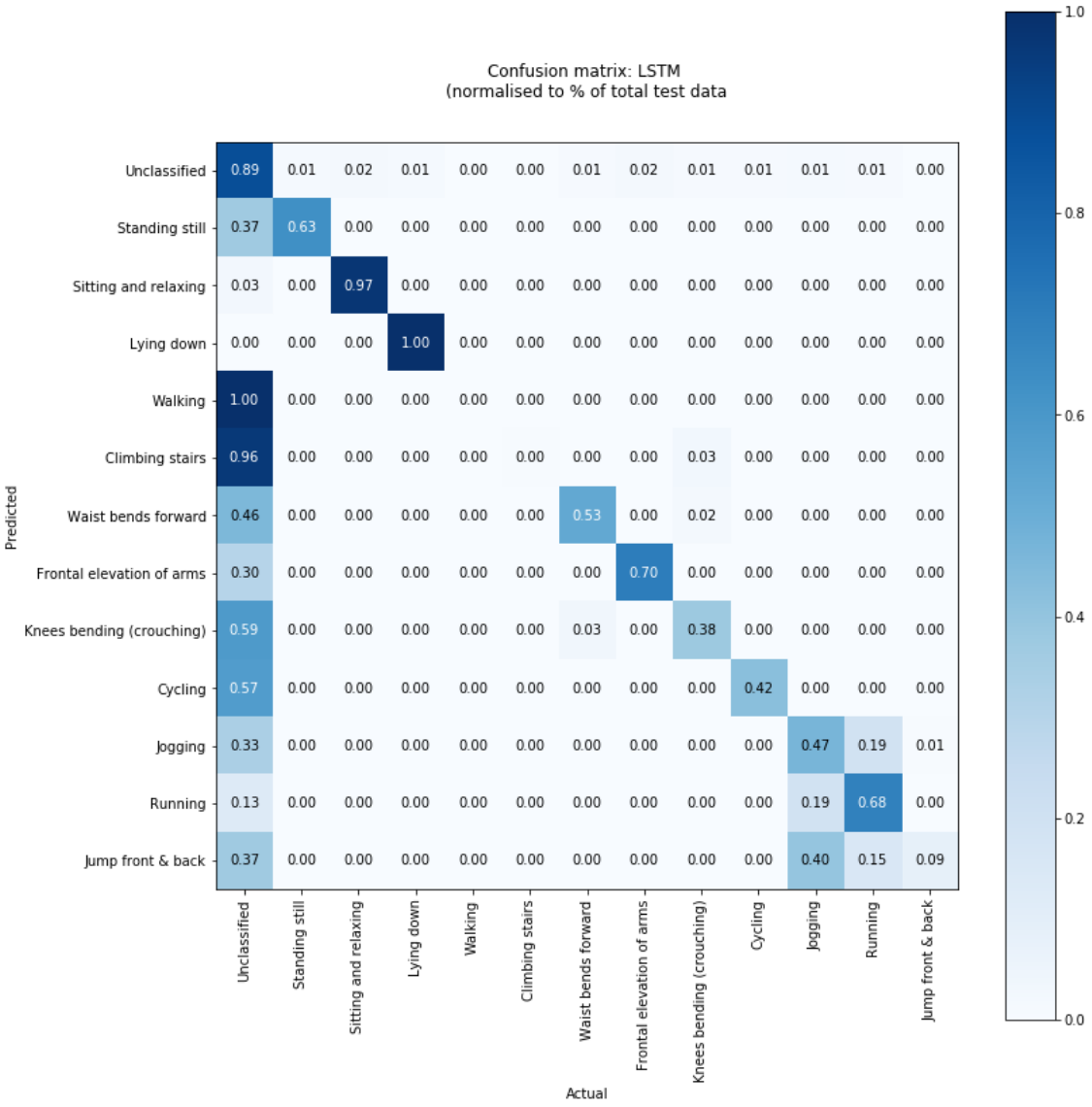

Figure 56 LSTM confusion matrix

LSTM and Hybrid had difficulty in classifying the activities 'waist bends forward' and 'knees bending (crouching)'. Both activities are similar in a sense that the waist in is a crouching motion when either activity is performed. In the case of 'waist bends forward' and 'knees bending (crouching)', one is misclassified as the other 216 times by the LSTM model, while the Hybrid model identifies 134 errors. Placing a sensor on the subjects' knee is a possible solution to distinguish between these activities. If the subject performed the 'waist bends forward' activity, there would be little or no activation of the signal. If the subject performed 'knees bending (crouching)', it would lead to a significant activation of the extra signal.

8.6 Feature Importance Analysis

Feature importance allows for the analysis of which features were the most important in each deep learning model in classifying each subject's activity. An extreme gradient boosting classifier was fitted to each models training and testing data to extract the features, which proved most decisive in correctly classifying the output. The number of estimators used for each model was 100. The XGBClassifier was run until the models 'merror' hadn't improved in 5 rounds. 'early_stopping_rounds' was set to 5 for each model. The F-Score was then extracted from the booster to give a detailed account of which features proved most important with the values sorted in terms of 'importance'.

The F score (f1-score) provides a detailed accuracy metric outlining the models accuracy. The F score is the weighted average of the models precision and recall.

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precisior}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precisior} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Formula 6 F1 score

Figure 57 depicts feature importance of the MLP network. It seems that the top five important features are:

1. Feature 8: acceleration from the left-ankle sensor (Z-axis).
2. Feature 18: gyro from the right-lower-arm sensor (X-axis).
3. Feature 15: acceleration from the right-lower-arm sensor (X-axis).
4. Feature 17: acceleration from the right-lower-arm sensor (Z-axis).
5. Feature 2: acceleration from the chest sensor (Y-axis).

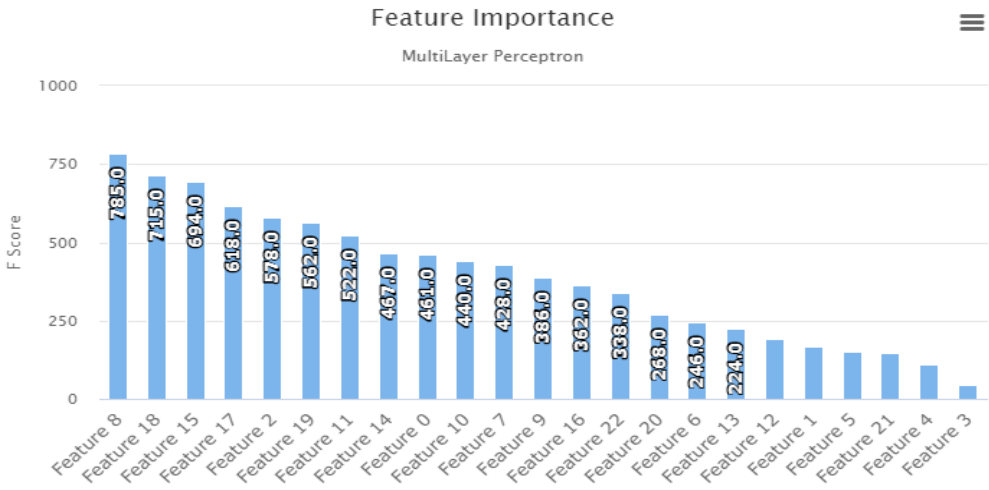

Figure 57 MLP feature importance

Figure 58 depicts feature importance of the CNN network. It seems that the top five important features are:

1. Feature 8: acceleration from the left-ankle sensor (Z-axis).
2. Feature 18: gyro from the right-lower-arm sensor (X-axis).
3. Feature 15: acceleration from the right-lower-arm sensor (X-axis).
4. Feature 17: acceleration from the right-lower-arm sensor (Z-axis).
5. Feature 2: acceleration from the chest sensor (Y-axis).

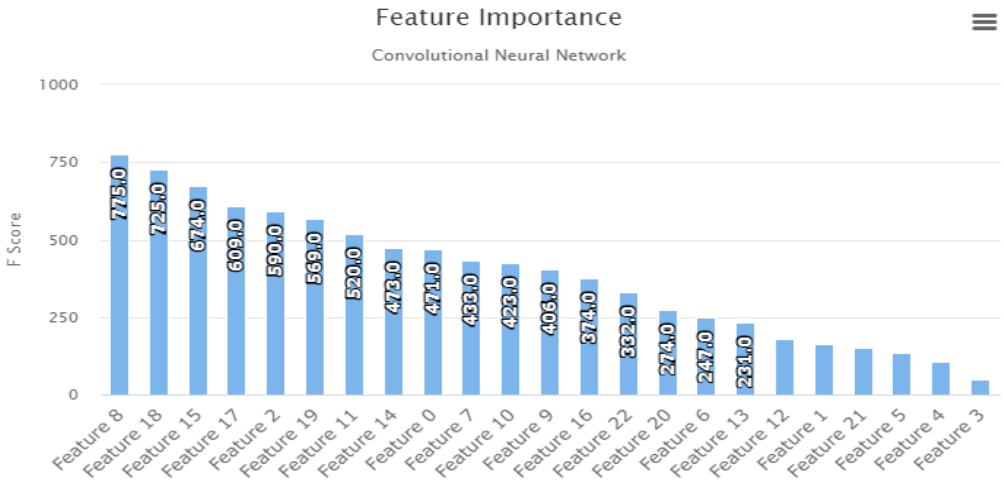

Figure 58 CNN feature importance

Figure 59 depicts feature importance of the XGBoost network. It seems that the top five important features are:

1. Feature 8: acceleration from the left-ankle sensor (Z-axis).
2. Feature 10: gyro from the left-ankle sensor (Y-axis).
3. Feature 18: gyro from the right-lower-arm sensor (X-axis).
4. Feature 19: gyro from the right-lower-arm sensor (Y-axis).
5. Feature 17: acceleration from the right-lower-arm sensor (Z-axis).

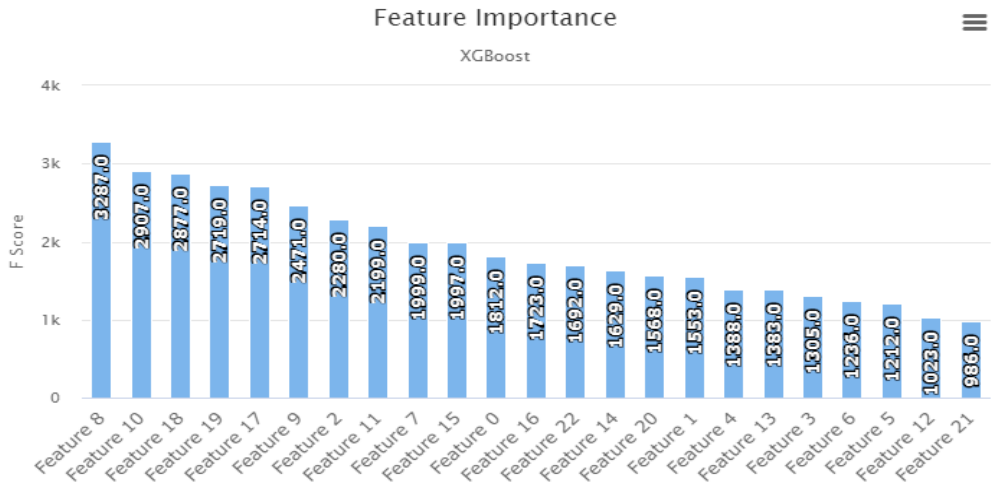

Figure 59 XGBoost feature importance

Figure 60 depicts feature importance of the Hybrid network. It seems that the top five important features are:

1. Feature 8: acceleration from the left-ankle sensor (Z-axis).
2. Feature 18: gyro from the right-lower-arm sensor (X-axis).
3. Feature 15: acceleration from the right-lower-arm sensor (X-axis).
4. Feature 17: acceleration from the right-lower-arm sensor (Z-axis).
5. Feature 2: acceleration from the chest sensor (Y-axis).

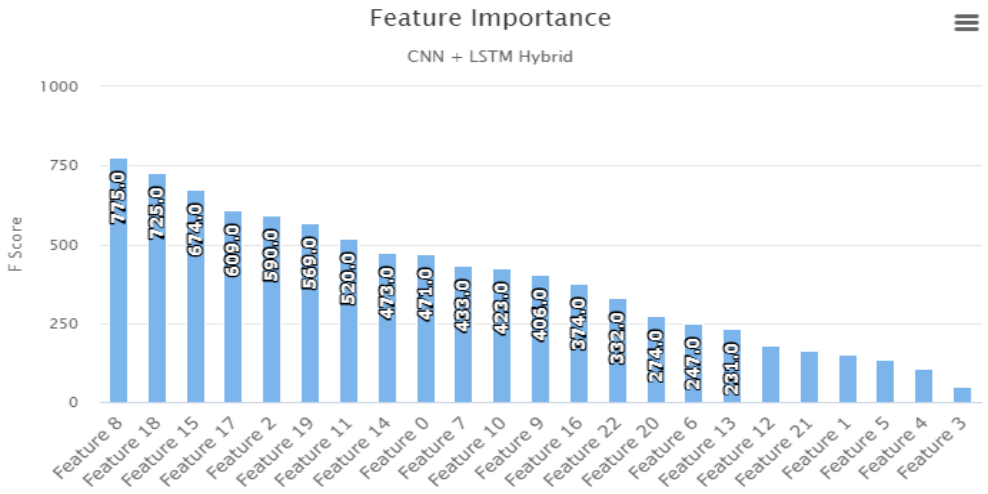

Figure 60 ConvLSTM feature importance

Figure 61 depicts feature importance of the Autoencoder by Random Forest network. It seems that the top five important features are:

1. Feature 8: acceleration from the left-ankle sensor (Z-axis).
2. Feature 18: gyro from the right-lower-arm sensor (X-axis).
3. Feature 15: acceleration from the right-lower-arm sensor (X-axis).
4. Feature 2: acceleration from the chest sensor (Y-axis).
5. Feature 17: acceleration from the right-lower-arm sensor (Z-axis).

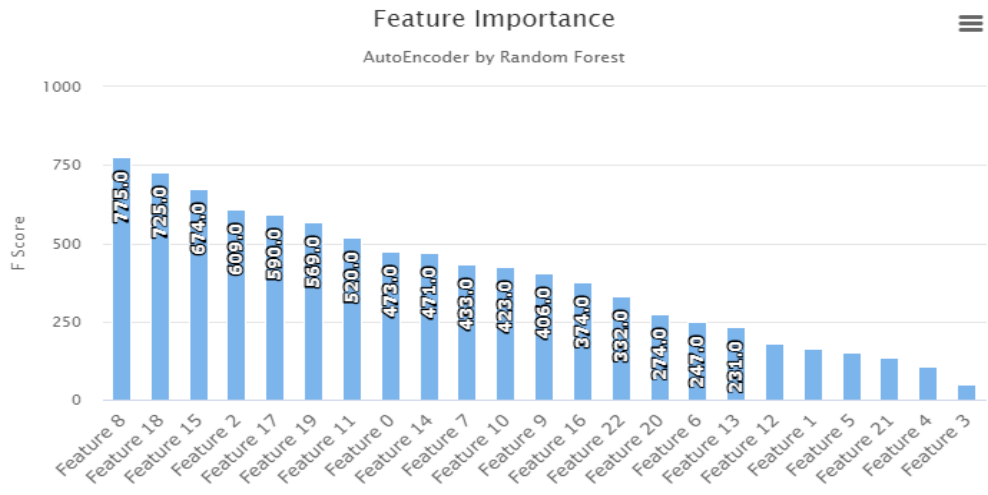

Figure 61 Autoencoder by Random forest feature importance

Upon evaluating the feature importance of all 6 models, it can be said that the following features are significant contributors to the performance of deep learning models in classifying the subjects activity.

- Feature 8: acceleration from the left-ankle sensor (Z-axis).
- Feature 18: gyro from the right-lower-arm sensor (X-axis).
- Feature 15: acceleration from the right-lower-arm sensor (X-axis).
- Feature 2: acceleration from the chest sensor (Y-axis).
- Feature 17: acceleration from the right-lower-arm sensor (Z-axis).
- Feature 10: gyro from the left-ankle sensor (Y-axis).
- Feature 19: gyro from the right-lower-arm sensor (Y-axis).

'Feature 8' prevailed as the most important feature in each of the 6 DL models, achieving an F score between 775 and 785 in each model. 'Feature 18' is depicted as the second most important feature across all 6 models, ranking as the second most important feature in 5 of the 6 models behind 'Feature 8'. 'Feature 15' is depicted as the third most important feature as it ranked as the third most important feature in 5 out of 6 models also. 'Feature 17', 'Feature 2', 'Feature 10' and 'Feature 19' were also significant contributors to each DL models performance achieving high F-Scores respectively.

'Feature 3' was the least important feature in the MLP, CNN, ConvLSTM and AE models. It had little or no contribution in classifying each subject's activities. 'Feature 4' was the second least important feature in the MLP, CNN, Hybrid and AE models also. However, 'Feature 3 and 4' were moderately important in classifying the activities for the XGBoost network. 'Feature 21' and 'Feature 5' also had minimum influence in predicting the subjects' activities.

8.7 Multisensor Fusion Analysis

Accelerometers, gyroscopes, magnetometers and electrocardiograms are four key sensors that are highly relevant for human activity recognition. One of these four sensors alone can predict activity recognition for human beings but studies have shown that the more sensors that are activated, processed and analysed lead to greater performance. Sensor fusion has yielded greater analysis results in recent studies.

Activity recognition is a complex problem domain with it being increasingly difficult to measure precise arm and leg orientation. This research has proved that combining accelerometers, gyroscopes, magnetometers and electrocardiograms sensors can be processed and applicable to activity recognition frameworks. Measuring precise limb orientation is a difficult task given different subjects (volunteers) weight and height, it is essential when building an activity recognition framework that multisensor fusion analysis is applied. This leads to greater performance results due to influential factors

such as size, weight, height and power being minimised as fusion analysis from many sensors is performed.

The MHEALTH data files consist of raw, unstructured data. It is essential that data pre-processing occurs, before a machine learning model is applied. Pre-processing is needed to extract the features and labels associated with the data. The labels, in this case, are each of the twelve different activities the subject performs, while the features are the 23 signals that are recorded from the accelerometers, gyroscopes, magnetometers and electrocardiograms when the subject performs an activity. Feature extraction is the first stage in fusion analysis. Before any specific pre-processing, analysis cannot be performed as the sensors data is raw and unstructured so it is difficult to feed it into the model.

Feature importance analysis depicts the necessity of multisensor fusion analysis.

MLP yielded the greatest performance. MLPs ability to integrate accelerometer, gyroscope, magnetometer and electrocardiogram signals led to it achieving the best results. Combining key features such as feature 8 (accelerometer with F score of 785), feature 18 (gyroscope with F score of 715) and feature 22 (magnetometer with F score 338) contributed greatly to MLP outshining the other models.

MLP Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3553.
- Gyroscope achieved a combined F-Score of 3506.
- Magnetometer achieved a combined F-Score of 1692.
- Electrocardiogram achieved a combined F-Score of 156.

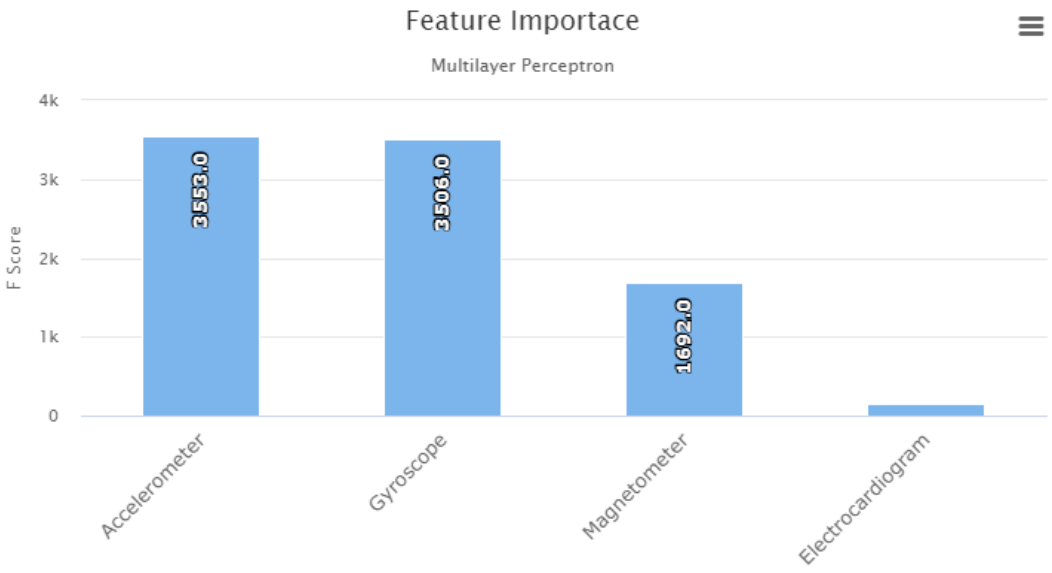

Figure 62 MLP sensor importance comparison

XGBoost Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3541.
- Gyroscope achieved a combined F-Score of 3465.
- Magnetometer achieved a combined F-Score of 1742.
- Electrocardiogram achieved a combined F-Score of 158.

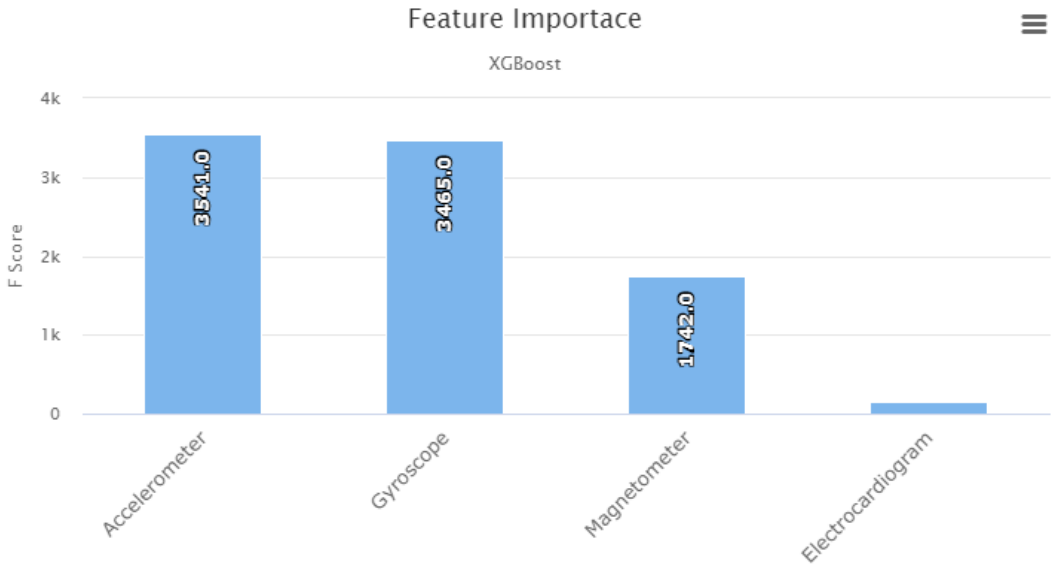

Figure 63 XGBoost sensor importance comparison

ConvLSTM Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3582.
- Gyroscope achieved a combined F-Score of 3454.
- Magnetometer achieved a combined F-Score of 1733.
- Electrocardiogram achieved a combined F-Score of 138.

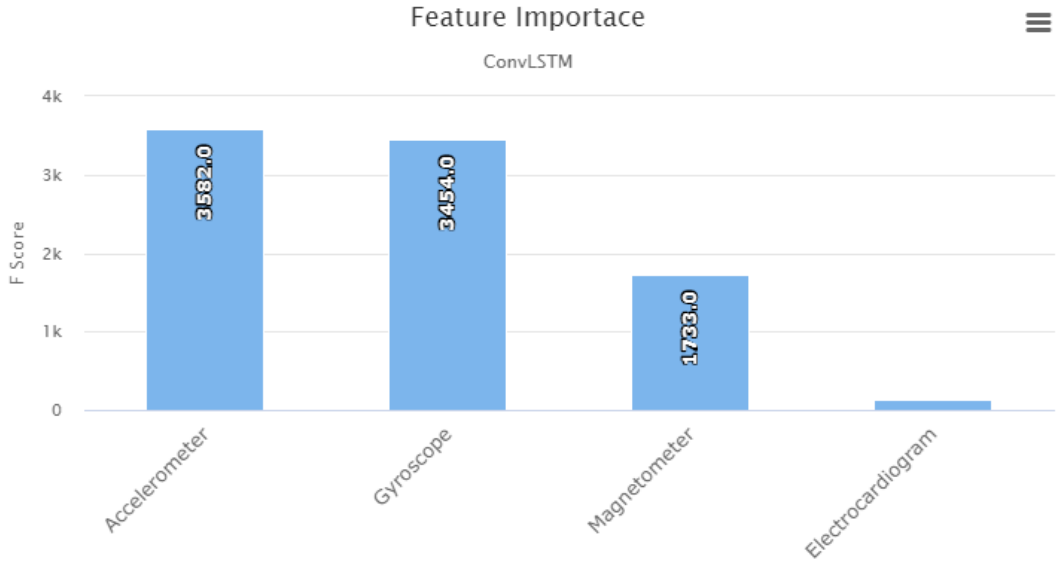

Figure 64 ConvLSTM sensor importance comparison

CNN Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3601.
- Gyroscope achieved a combined F-Score of 3448.
- Magnetometer achieved a combined F-Score of 1704.
- Electrocardiogram achieved a combined F-Score of 151.

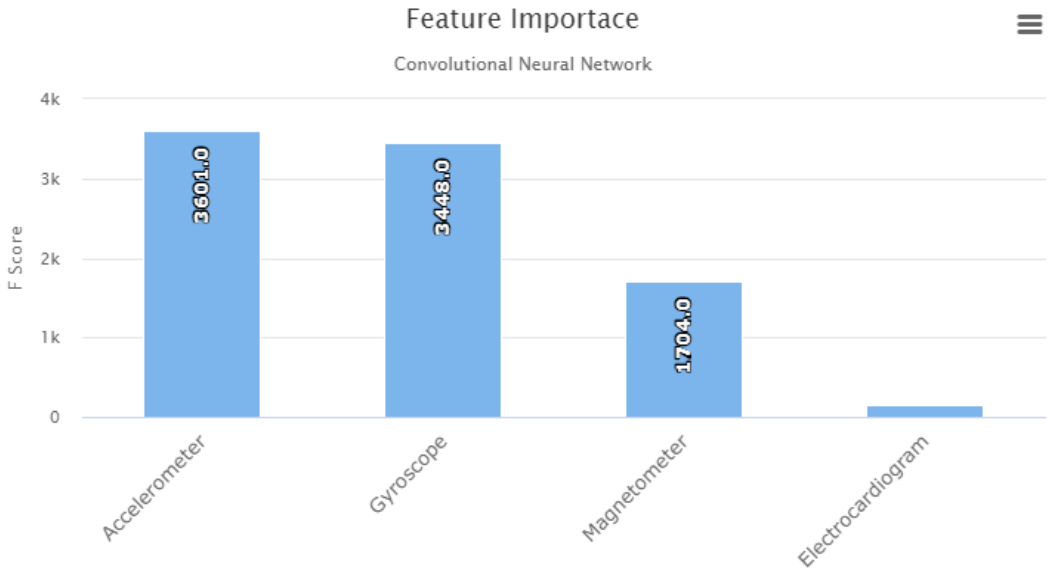

Figure 65 CNN sensor importance comparison

Autoencoder by Random Forest Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3589.
- Gyroscope achieved a combined F-Score of 3475.
- Magnetometer achieved a combined F-Score of 1710.
- Electrocardiogram achieved a combined F-Score of 136.

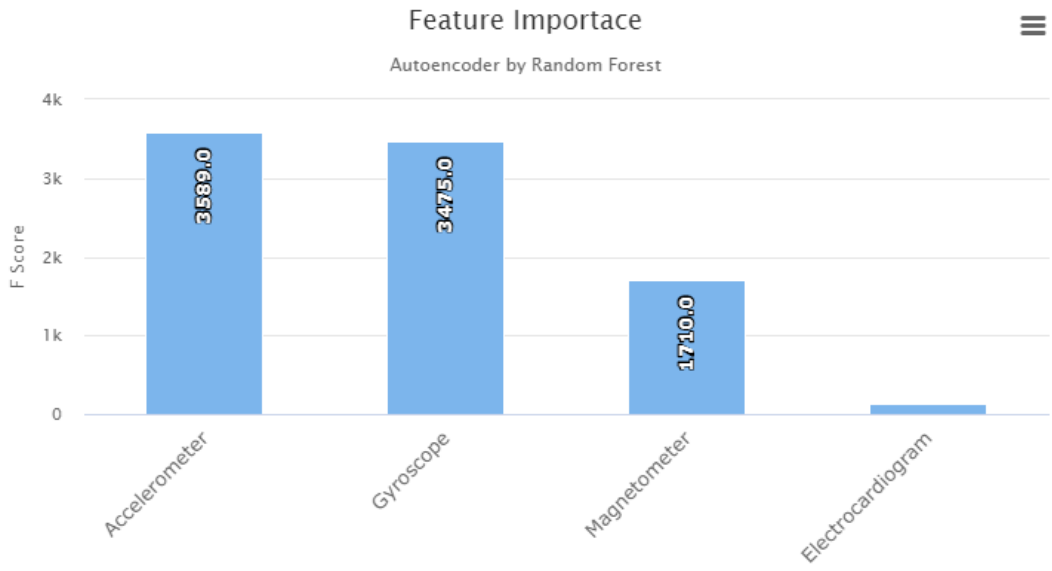

Figure 66 Autoencoder by random forest sensor importance comparison

LSTMs Multisensor Fusion Analysis Key Points:

- Accelerometer achieved a combined F-Score of 3582.
- Gyroscope achieved a combined F-Score of 3463.
- Magnetometer achieved a combined F-Score of 1704.
- Electrocardiogram achieved a combined F-Score of 140.

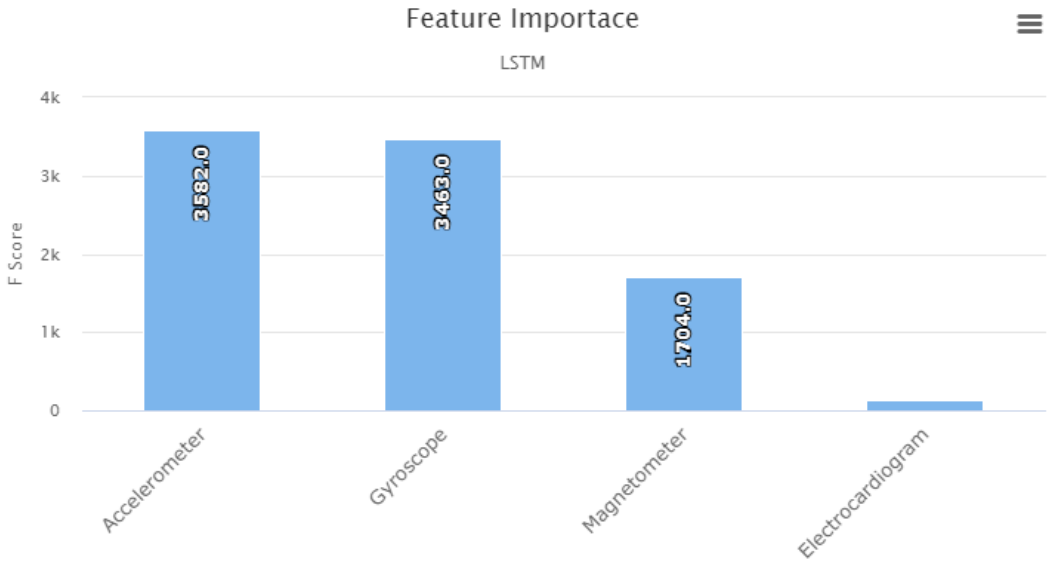

Figure 67 LSTM sensor importance comparison

Integrating fusion sensor analysis simultaneously contributes to greater performance as seen by the XGBoost model, while accelerometers, gyroscope and magnetometer having a significant influence in generating results. XGBoost and MLP were the two best performing models and benefited greatly from multisensor fusion analysis.

Multisensor fusion analysis benefited the CNN, LSTM and Hybrid models also as seen in Figure x, x and x. Each model fused accelerometer, gyroscope, magnetometer and electrocardiogram data together to improve model performance. Integrating the analysis of multiple sensor channels from various sensors leads to an increase in performance, efficiency and overall results. It has a significant effect on the amount of misclassified or unidentifiable activities also. These results below depict that implementing multisensor fusion analysis is essential when building an activity recognition framework.

8.8 Comparison with existing State-of-the-Art

Comparison 1

This section compares this research papers models with state-of-the-art methods.

Ha and Choi [138] present five models to conduct analysis on the mhealth dataset, these models include:

- Hidden Markov model (HMM)
- Support vector machine (SVM)
- Hidden conditional random fields (HCRF)
- Convolutional neural networks with 2D kernel (2D CNN)
- Convolutional neural networks with 1D kernel (1D CNN)

Table 15 presents an overview if Ha and Choi [138] approaches on the mhealth dataset

Classification Accuracy (%)	
Ha and Choi [1]	
HMM	66.62
SVM	65.4
HCRF	68.6
1DCNN	90.43
2DCNN	89.14

Table 15 Ha and Choi [138] mhealth performance

The following table presents the performance measures of the approaches implemented throughout this research. MLP achieves 90.55% accuracy, which is greater than each accuracy approach proposed in [138]. XGBoost outperforms four out of the five approaches, falling just 0.47% short of their top performing approach, which is 1DCNN. The CNN, LSTM, Hybrid and AE w.RF outperform Ha and Choi's HMM, SVM and HCRF models. Overall, the approaches presented in this research outshine the results presented in [138].

Architecture	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*f1 score*
MLP	**90.55**	**91.66**	**90.55**	**90.7**
CNN	83.91	83.47	83.91	82.98
LSTM	78.09	74.86	78.09	75.6
Hybrid	83.89	83.69	83.89	83.2
AE RF	83.27	82.59	83.25	81.54
XGB	89.97	90.09	89.97	89.78

Table 16 Performance Comparison

Comparison 2

Chen et al. [139] presents six classification models to conduct analyse on the mhealth dataset, these models include:

- Support vector machine.
- Random Forest
- KNN
- Decision Tree
- Single Neural Networks
- Unique approach.

The following table presents an overview of Chen et al. [139] approaches on the mhealth dataset.

Classification Accuracy (%)	
Chen et al. [2]	
SVM	68.7
Random Forest	82.5
KNN	86.1
Decision Tree	78.7
Single Neural Netowork	89.1
Unique approach	94

Table 17 Chen et al. [139] performance comparison

The following table presents the performance measures of the approaches implemented throughout this research. MLP achieves 90.55% accuracy, which is greater than the SVM, Random forest, KNN, decision tree and single neural network proposed in [139]. XGBoost outperforms five out of the six approaches, falling just 4% short of their top performing approach, which is a unique approach. The CNN, LSTM, Hybrid and AE w.RF outperform Chen et.al. [139] SVM approach. Overall, the approaches presented in this research outshine the results presented in [139] due to the fact that five out of the six classification accuracy's approaches in this research range between 83% and 91% while only three out of 6 approaches achieve a classification accuracy of 83% or greater in [139]. Time constraints led this research to not implement its own unique approach, if more time was given, achieving 94% accuracy would have been viable.

Architecture	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*f1 score*
MLP	**90.55**	**91.66**	**90.55**	**90.7**
CNN	83.91	83.47	83.91	82.98
LSTM	78.09	74.86	78.09	75.6
Hybrid	83.89	83.69	83.89	83.2
AE RF	83.27	82.59	83.25	81.54
XGB	89.97	90.09	89.97	89.78

Table 18 Performance Comparison

Comparison 3

Uddin and Hassan [140] present three models to conduct analysis on the mhealth dataset, these models include:

- Artificial Neural network
- Deep belief network
- Unique approach

The following table presents an overview of Ha and Choi Uddin and Hassan [140] approaches on the mhealth dataset:

Classification Accuracy (%)	
Uddin and Hassan [3]	
ANN	87.99
DBN	90.01
Unique approach	93.9

Table 19 Uddin and Hassan [140] performance comparison

The following table presents the performance measures of the approaches implemented throughout this research. MLP achieves 90.55% accuracy, which is greater than the ANN and DBN classification accuracy approach proposed in [140]. XGBoost outperforms the ANN model, falling just 0.04% short of the DBN model and 4% short of their top performing approach, which is a unique approach. The CNN, LSTM, Hybrid and AE w.RF fall short of the proposed three approaches Overall, the approaches presented in this research were very good, stressing the performance that a unique approach implementation can lead to very high classification accuracy.

Architecture	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*f1 score*
MLP	**90.55**	**91.66**	**90.55**	**90.7**
CNN	83.91	83.47	83.91	82.98
LSTM	78.09	74.86	78.09	75.6
Hybrid	83.89	83.69	83.89	83.2
AE RF	83.27	82.59	83.25	81.54
XGB	89.97	90.09	89.97	89.78

Table 20 Performance Comparison

Comparison 4

Kutlay and G.Palalic [141] present two models to conduct analysis on the mhealth dataset, these models include:

- Multilayer Perceptron
- Support vector machine

The following table presents an overview of Kutlay and G.Palalic [141] approaches on the mhealth dataset:

Performance Measures				
Kutlay and G.Palalic [4]				
	Accuracy	Precision	Recall	F1-Score
MLP	91.7	91.9	91.7	91.5
SVM	83.2	76.4	83.2	78.1

Table 21 Kutlay and G.Palalic [141] performance comparison

In terms of performance, the MLP model in this research achieves better accuracy, precision, recall and f1-score that the SVM model presented in [141]. XGBoost, Hybrid and CNN also outperform the SVM in terms of accuracy, precision, recall and f1-score. The MLP model presented in [141] achieves a greater performance measurement than the MLP presented in this research, the difference between the two in terms of the four performance measures are less than 1%.

Architecture	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*f1 score*
MLP	**90.55**	**91.66**	**90.55**	**90.7**
CNN	83.91	83.47	83.91	82.98
LSTM	78.09	74.86	78.09	75.6
Hybrid	83.89	83.69	83.89	83.2
AE RF	83.27	82.59	83.25	81.54
XGB	89.97	90.09	89.97	89.78

Table 22 Performance Comparison

Comparison Conclusion

In conclusion, the six classification models presented in this research perform well when compared to existing state-of-the-art baselines. MLP and XGBoost achieve excellent performance measures, challenging many research papers with an improved

accuracy, precision, recall and f1-score. The XGBoost model is the best performing model in terms of overall performance and is highly suited to mobile health data.

9 Discussion

9.1 Hyperparameter Evaluation

Hyperparameters are the most influential aspect in ensuring machine learning models perform. The following aspects had a significant influence in the development and deployment of each of following models: MLP, XGBoost, CNN, LSTM and Hybrid.

Setting up - Dropout regularisation, early stopping, normalising inputs, vanishing + exploding gradients, weight initialisation.

Optimisation algorithms - Adam optimisation: learning rate.

Hyperparameter tuning – learning rate, number of hidden layers, number of hidden units for different layers, batch size, epochs, Batch normalisation.

Setting up

Dropout regularisation benefits the DL models. A dropout core layer is applied to each of the models. The dropout rate is set to 0.4 as this is the probability of tuning each input to the subsequent layer to 0. Setting a dropout rate to 0.4 creates a dropout layer in the DL model with 40% chance of initialising inputs to 0. Dropout regularisation reduces overfitting and has been proven to be a very effective regularisation technique. It simplifies the model, allowing the model to learn minute details about the data while updating weights during gradient descent and producing excellent results. Dropout allows 0.4 (40%) of diverse sets of hidden layers to be 'dropped' as each epoch is initialised. Dropout is performed during the training phase, not testing.

Early stopping was used in each models extreme gradient boosting classifier. Early stopping allowed each models training and testing data to extract the features, which proved most decisive in correctly classifying the output. The number of estimators used for each model was 100. The XGBClassifier was run until the models 'merror' hadn't improved in 5 rounds. 'early_stopping_rounds' was set to 5 for each model. The XGBClassifier ran until the training error stopped continuously decreasing. Early stopping ensured the reduction of overfitting.

Normalising inputs enhances performances by reducing the amount of time the model takes to learn the data. It accelerates the training phase. Normalising the data allowed each data range (in each model) to be identical, leading to an increase in algorithm speed as the data becomes more symmetrical. Without normalising the data, algorithm speed

would be severely slow as scale will vary and the parameters for each signal would take too long to learn.

The use of the *ReLu activation function* led to a reduction in *vanishing / exploding gradients*. ReLu ensured that each DL network built non-linear relation between data points. If there was no ReLU layer, each network wouldn't be able to classify non-linear data points. ReLU significantly enhanced speed, accuracy and precision.

Optimisation algorithms - Adam optimisation: learning rate.

Adam optimisation leads to DL models training their network on the data extremely fast. Adam is a gradient-based optimiser, which is straightforward, simple to implement as is computationally inexpensive. The hyper-parameters require little or no tuning which is why Adam worked perfectly on the DL models. Learning rate was the hyperparameter used for Adam. The learning rate was fine-tuned to '0.0005' to enhance the speed of the learning process for each neuron. The learning rate was the key to a successful learning process for the networks neurons.

Hyperparameter tuning

The following hyperparameters were vital in effectively generating good performance results regarding each DL model: learning rate, number of hidden layers, number of hidden units for different layers, batch size and the number of epochs.

The *number of hidden layers* and the *number of hidden units* for different layers varied across each model. These two hyperparameters help the model to learn the training data. They ensure results are conclusive, relevant and maximised.

The *batch size* for each model was set to 32 while the number of *epochs* (*training iterations*) was 20. This set batch size and number of epochs proved best for each model, as proven by the results showing the minimum accuracy being 80%, which is quite satisfactory.

Batch normalisation ensured successful updates in data values across more than one layer in each DL model. Batch normalisation allowed each model to reparameterise after each subsequent layer, allowing for successful updates. Each DL model consisted of many layers and hyperparameters, leaving batch normalisation providing a key role in constant coordination and updates to ensure results provided accurate predictions in activity.

Epsilon and *Min Points* ensured DBSACN successfully clustered the mhealth data. Setting these hyperparameters ensured the clustering results provided insightful discussion points. Epsilon was set as 3 and Min Points was set as 60.

Perplexity, *learning rate* and *number of iterations* were set to process the t-SNE clustering algorithm. The output of t-SNE did not produce visualisations as expected. Tweaking the hyperparameters did not lead the results to vary, all three experiment adjustments yielded similar results.

Hyperparameter Evaluation Conclusion

- ✓ Regularisation is excellent in minimising overfitting for the mhealth dataset.
- ✓ Adam is the best optimisation algorithm that suits this data.
- ✓ Fine-tuning the hyperparameters to suit the subject data yields excellent, insightful results while speeding up training the model.
- ✓ Batch normalisation enhances training each DL model.

9.2 Machine Learning Models Discussion

The main conclusions from the comparison of Multilayer Perceptron, XGBoost, Convolutional Neural Network, Long short-term memory (Recurrent Neural Network), Hybrid (CNN+LSTM) and Autoencoder by Random Forest is that:

- MLP and XGBoost reaches a higher accuracy (90.55%, 89.97%), precision (91.66%, 90.09), recall (90.55%, 89.97%) and f1 score (90.7%, 89.78) respectively.
- MLP and XGBoost provide more conclusive segmentation characteristics as depicted by the analysis of feature importance, showing which features are most influential.
- They are significantly superior in their ability to distinguish between similar activities (e.g., 'jogging/running' and 'climbing stairs/knees bending (crouching)').
- To the authors' knowledge, XGBoost has not been implemented on the MHEALTH dataset in order to classify each subjects' activity. Findings suggest that XGBoost can be successfully applied to the MHEALTH dataset and be comparable to existing state of the art baselines.
- Deep learning algorithms can recognise physical human activity with on-body sensor data, upon multisensor fusion analysis.

These conclusions reinforce the hypothesis that an XGBoost model created and implemented on health data to predict human activities generates significant power to learn temporary feature activation dynamics and make decisive predictions in classifying the subjects predicted activity.

The MLP model is an 8 layer deep neural network. CNN, LSTM, Hybrid and AE using RF have 12, 6, 7 and 6 layers respectively. In the domain of activity recognition using deep learning, increasing the number of layers in a neural network is not always beneficial. Serious consideration is vital when given the opportunity to add successive

layers to a model for classification of a given task. As the number of layers increases, this leads to a greater time in training the data. This solely depends on the processing power and extent of the researchers GPU and hard drive, but in many instances, an increase in layers leads to an increase in training time. Time is an important factor when building DL models. DL architectures seem to be more applicable for application on a 'cloud infrastructure'. Therefore, performance and computational ability of hard drives and processers are key factors in selecting the number of layers in a DL model.

Many deep learning architectures implement convolutional, pooling and dropout layers successively, to increase model performance and reduce the degree of data complexity. However, implementing these layers are not strictly vital. XGBoost does not include convolutional, pooling or dropout layers as it processes data under the Gradient boosting framework. XGBoost is excellent for increasing performance and speed due to its ability to implement a variety of gradient boosted decision trees to analyse data and generate meaningful, decisive conclusions. XGBoosts results (accuracy 89.97%, precision 90.09%, recall 89.97%, f1-score 89.78%) proves it can generate excellent performance with a high degree of data complexity presented by the MHEALTH data. Convolutional, pooling and dropout layers also present many benefits. They are becoming significantly useful in analysing data spread across a more profound period. The 'deeper layers' that DL models create while utilising these layers have the ability to spread analysis across a significant period of time, while each unique convolutional layer performing on sensor data at different instances.

Fine-tuning each DL network architecture is a vital aspect of generating appropriate, conclusive results. Setting the correct hyperparameters is the most important aspect of creating a DL model. Without appropriate hyperparameters set, performance will diminish along with the 'number of incorrectly classified instances' increasing. Optimisation and fine-tuning of hyperparameters led to XGBoost and MLP performing excellently, while CNN, LSTM, Hybrid and AE with RF performed satisfactory also. These results indicate that setting learning rate, batch size, number of epochs and activation functions are key, necessary steps to ensure DL models conduct analysis sufficiently. Setting appropriate loss and accuracy metrics are important in visualising results also. Computational aspects such as the processing power of a hard drive and technical ability of a GPU is not always a key factor, with optimising hyperparameters prevailing as significant when building DL network architectures.

The architectures present in CNN, LSTM and ConvLSTM contain convolution procedures that are robust enough to be tested and applied to the raw sensor data that the MHEALTH dataset contains. CNN achieved accuracy, precision, recall and f1-score of 83.91%, 83.47%, 83.91% and 82.98% respectively. LSTM achieved accuracy, precision, recall and f1-score of 78.09 %, 74.86%, 78.09% and 75.6% respectively. ConvLSTM achieved accuracy, precision, recall and f1-score of 83.89 %, 83.69%, 83.89% and 83.20% respectively. These convolution procedures allow each model the ability to learn salient patterns, leading them to distinguish between features in order to

classify the subjects' activity. Neural networks are applicable in a wide range of domains for many reasons, the benefits NN's entail is endless. The most important one is that 'heuristic features' can be averted, therefore minimising the engineering bias as much as possible. Engineering bias is essentially hard coding the data in question to benefit the implemented model, almost 'cheating' the results. These results would not have been identified before the 'hard coding' was conducted, thus it is a false model and this is why the engineering bias should be minimised.

The MHEALTH dataset contains a vast amount of data. Each model is more than capable of generating excellent performances with the dataset. When an activity is performed, the activated sensors generated hundreds of thousands of signals. The dataset contains 1,215,745 instances. Despite this, each models performance was satisfactory. This outlines that even though machine learning and deep learning techniques are in many cases utilised with a significant amount of data in human activity recognition (millions of samples per millisecond), machine learning networks may be suited to specific problem domains where requiring data (which has already 'labels' applied) is too computationally expensive, such as in the health domain.

MLP, CNN, LSTM and Hybrid cells composed of a high number of values per cell. The overall amount of parameter values associated with the cells increases vastly as the number of layers increases. The MLP with the Null class is composed of 706,317 training parameters, while CNN contains 245,584 training parameters. LSTM and Hybrid are composed of 175,373 and 191,376 respectively. Autoencoder by random forest contains 23,711 training parameters. This variation in parameter size is linked to the associated connection between convolutional layers, pooling layers, dropout layers and dense layers. In a machine learning architecture, which is fully connected, the values in the dense layer must be linked with every parameter value of the last feature map, which is the previous layer. This leads to the formation of a weight matrix that is significantly large, it is vital in ensuring the parameters of the connection don't get out of proportion. XGBoost processes the data much differently than the models previously mentioned. XGBoost ensures that the amount of parameter values needed is minimised. The Gradient boosting framework built into XGBoost ensures this is correctly applied. XGBoost is said to me a more complex network, but it is formed of a reduced number of parameters, and is directly linked to the outstanding benefits it produce in respect to GPU memory and hard drive computational processing power. The amount of time XGBoost uses in comparison to the high number of values it processes is consistently satisfactory.

In terms of training time and classification of the correct activity, all the models yield generally good results results. XGBoost outperforms the rest in terms of processing speed and classification speed. MLP, CNN, LSTM and Hybrid are more complex than XGBoost. This is associated to the amount of units in the convolutional and dense layers needed to build the model in question. Training XGBoost on the MHEALTH dataset requires 60 minutes to assemble while MLP requires 90 minutes to assemble. Therefore,

the implementation of XGBoost, MLP, CNN, LSTM and a ConvLSTM model is suitable for analysing health data relating to human activity recognition.

9.3 DBSCAN and t-SNE Discussion

To gain insight into the formation of the MHEALTH dataset, and fully understand the relationships between the features of the data, two dimensionality reduction data clustering algorithms were applied: t-SNE and DBSCAN.

DBSCAN Conclusion

The purpose of DBSCAN is to analyse the spatial dynamics of activity recognition data over an area. DBSCAN analyses the data to identify patterns, trends and anomalies, noise points and identify appropriate actions. The main goal is to identify a data hotspot for specific activity recognition.

In terms of aiding remote patient monitoring and identifying unusual activity recognition data patterns in patients who are ill or battling diseases, fast and precise analysis is vitally important. A data hotspot would be used in detecting areas of significant risk if an abnormality occurs when comparing the real-time data hotspot (high-risk activity performed) with the normal data hotspot (safe activity performed).

The results from this research prove DBSCAN is successful in identifying data hotspots in MHEALTH data. DBSCAN successfully visualised 10 clusters for subject 1 with a silhouette coefficient of 0.073 and 11 clusters for subject 2 with a silhouette coefficient of 0.087. All ten subjects' results yielded excellent results, with the silhouette coefficient closer to zero than to 1, proving there is a beneficial relationship between clusters and reiterating the fact that DBSCAN is suited to MHEALTH data.

Showing the data hotspots would be a vital signal in:

- Safety control techniques
- Ensuring the patient doesn't become more ill.
- Ensuring disease doesn't spread.
- Ensure elderly patient is moving or performing activities in a correct, safe manner.

In human activity recognition research it is important to analyse whether data attributes are influencing a movement by chance or has a significant link to the activity performed. Data in clusters have a unique relationship in comparison to other data points excluded from the cluster. These clusters have a precise number of points with a specific shape and size. This isn't a coincidence. The cluster isn't associated with any other part of the data environment, with no other data points having an effective on its transformation. Therefore, analysing and evaluating cluster compositions from activity recognition can help understand:

- Prevalence of diseases.
- Identifying a disease.
- Identify unusual activity.
- Identify abnormal behaviour.
- Statistical confirmation or rejection.
- Suggesting possible clues for diagnostics.

This knowledge can be a driving force in benefiting remote patient monitoring as well as reaping great benefits for assisted ambient living and assisting the elderly.

t-SNE Conclusion

When applying t-SNE to the data the main outcome was to distinguish relationships between the variables and hopefully gain a better understanding of the overall variance in the MHEALTH dataset. As seen by previous studies, discussed in section 3.12, t-SNE can visualise complex relationships between data and map these relationships to non-linear paths. Conducting thorough analysis on these relationships and non-linear paths can lead to a greater understanding of the data.

For t-SNE to distinguish a good embedding, the KL-divergence should have been minimised leading the overall structure of the data in the embedded space to be intact during the mapping of the data points. This was not the case and t-SNE found it difficult to correctly visualise the data. Hyperparameters were adjusted to gain more insight into the relationships between the features. Adjustment experiment 1, 2 and 3 yielded relatively the same results after altering the learning rate, perplexity and number of components. The blue points throughout the three adjustment experiments were consistently visualised across multiple iterations with these hyperparameter settings. These blue points (noise) led to a huge early exaggeration; this further led to an amplified relationship with the other clusters. The Null Class played a huge role in amplifying this relationship, due to the vast amount of data points represented by the Null class.

The t-SNE experiments concludes that data is clustered very tightly together, although it spreads across a wide range of space (-17.5, 17.5). The t-SNE output for each subject is a strange 'ball' with uniformly distributed points. All of the points seem to overlap. t-SNE is clearly not a good choice for data visualisation here as it is difficult to visualise the clusters. The blue points on each graph seem to spread across the whole cluster, making it difficult to visualise the exact clusters of the variables in question. The 'Null Class' is having a huge effect on the t-SNE clustering output. The KL-divergence ranging from 4.1 to 4.9 across all ten subjects proves t-SNE is a poor choice for data visualisation on MHEALTH data.

Due to this, t-SNE found it difficult to distinguish between the MHEALTH activity set such as climbing stairs and walking, running and jogging, and waist bends forward and

knees bending (crouching). Each of these activities should have been visualised by a cluster, with each activity slightly or significantly overlapping. This would have led to a greater understanding of the dataset, showing which variables influenced certain movement. Visualising the activity set posed a problem for t-SNE, mainly due to the presence of the Null class.

If given more time to conduct experiments, eliminating the Null class would be step number one. Performing t-SNE on the data would lead to insightful analysis. An in-depth comparative study of DBSCAN vs t-SNE would prove interesting, comparing both types of cluster visualisations.

Similar to DBSCAN, t-SNE clustering aiding HAR in the healthcare sector in terms of analysing and evaluating the clustering compositions would help understand:

Prevalence of diseases, identifying a disease, Identify unusual activity, Identify abnormal behaviour, Statistical confirmation or rejection, Suggesting possible clues for diagnostics, Safety control techniques, Ensuring the patient doesn't become more ill, Ensuring disease doesn't spread, Ensure elderly patient is moving or performing activities in a correct, safe manner.

10 Conclusion

In this conclusion chapter, an overview is presented which looks back over the project, how it could have been improved, personal reflection, the contributions and achievements acquired during the project, human activity recognition limitations and the future work that could be implemented into the project.

10.1 Personal reflection

Upon reflecting back over the thesis and how each task was individually completed, there isn't any major flaw in the schedule, tasks were completed within the schedule and on-time.

Looking back if I were to start the thesis from scratch again, I would start getting to grips with Python earlier. I began learning how to code in Python in January, it took me quite some time to learn the basics and become comfortable with the syntax. If starting from scratch again, I would start becoming familiar with Python the previous September, or even earlier.

If I had perhaps became more comfortable with coding in Python, I could have wrote the Deep Learning models presented in this thesis from scratch. It would have taken a lot of time and a comparative study of six classification models would have proven very difficult, but classification evaluation measures could have possibly increased.

Assuming Python could be learned at a high-level in a short space of time was naïve because I spent a significant amount of time going back changing code and running the models differently, this time could have been utilised more efficiently focusing on other tasks. Coding the thesis in Java may have taken significantly less time to run the models also.

The last thing I would change if starting from scratch would be to create my own MHEALTH dataset. The MHEALTH dataset presented in this thesis was taken from the UCI Machine Learning Repository. If starting from scratch I would spend money on components such as cameras, sensors, GPUs, hard drives and multiple desktop monitors to run models simultaneously. Volunteers from my class would have performed the activities in question with sensors attached to them. I would also change the activities to more complex, challenging activities such as turning on the microwave, opening the fridge, hitting a sliotar with a hurl and kicking a football. I would ensure the volunteers performed different physical activities to create a complex human activity recognition model from scratch.

10.2 Machine Learning Conclusion

This research proves the strengths of deep learning architectures on MHEALTH (mobile health) data based on a comparative study of the following models:

- Multilayer Perceptron.
- XGBoost (Extreme Gradient Boosting).
- Convolutional Neural Network.
- Long Short-Term Memory (Recurrent Neural Network).
- ConvLSTM (CNN +LSTM Hybrid).
- Autoencoder by Random Forest.

Each model performs human activity recognition from wearable sensors such as gyroscopes, accelerometers, magnetometers and electrocardiogram. To the author's knowledge, for the MHEALTH dataset, XGBoost hasn't been performed to classify the activities in question. MLP prevailed as the best performing model achieving accuracy, precision, recall and F1 Score of 90.53%, 91.71%, 90.53% and 90.76% respectively. XGBoost was the next best performing model in a close second achieving accuracy, precision, recall and F1 Score of 89.98%, 90.14%, 89.98% and 89.78% respectively.

Although MLP outperformed XGBoost in terms of accuracy, precision, recall and F1 score, MLP misclassified 471 instances while XGBoost misclassified only 281. CNN, ConvLSTM and LSTM misclassified 1341, 2533 and 2742 instances respectively. In terms of overall accuracy, precision, recall, f1 score and number of correctly classified instances, XGBoost is the top performing mode. This details the known domain of appropriateness for the XGBoost framework, which has never been reported on the MHEALTH dataset using wearable sensors.

In terms of time requirements such as how long it takes to train the model, XGBoost performs excellently and presents the best trade-off in terms of classification performance and prediction time when compared to the MLP, CNN, LSTM, ConvLSTM and AE by Random Forest models. The XGBoost model performs classification recognition in roughly one hour, 50% faster than MLP and Autoencoder by Random Forest, and between 150-200% faster than CNN, LSTM and ConvLSTM.

This research determines that XGBoost is capable of recognising identical activities. Identical activities such jogging / running, jumping front and back / jogging, knees bending (crouching) / waist bends forward, and walking / climbing stairs are misclassified a small number of times. The MLP models performance on these activities wasn't as profound as XGBoost, generated two to three times more misclassified instances. CNN, LSTM, ConvLSTM and AE by Random Forest generated hundreds of more errors in respect to classifying these activities in comparison to XGBoost.

The XGBoost architecture offers much better analysis characteristics than the other five classification models. These characteristics include regularisation, tree pruning, tree depth and sparse features. XGBoost identifies the vital signs and range of motion of the

activities in question more accurately. All of these findings mentioned in this discussion section reiterate the hypothesis that XGBoost is the best performing model and is highly suited to analysing MHEALTH data.

From the feature importance and sensor fusion analysis results, each model significantly learned from the sensor signals generated from the gyroscope, accelerometer, magnetometer and electrocardiogram. Each model performed well on the data, with the lowest accuracy presented as 75% (LSTM), lowest precision as 75% (LSTM), lowest recall as 77% (LSTM) and lowest F1 Score as 74% (LSTM). Not much pre-processing was needed on the data, extracting features and labels, encoding to one-hot form and normalising the data was as complicated as it got before building and training the model began. This analysis presents a solution that sensor fusion can be performed where sensor data is generated from different devices, and it can undoubtedly be processed.

10.3 Data Clustering Conclusion

This research also presents the strengths of data clustering algorithms in analysing Mobile Health data. DBSCAN successfully the MHEALTH dataset in question by identifying clusters in the data which represented variables which have a strong relationship with one another. This details the known domain of appropriateness for the DBSCAN clustering algorithm, which has never been reported on the MHEALTH dataset using wearable sensors.

In terms of time requirements such as how long it takes to generate the clusters, DBSCAN performs excellently and presents the best trade-off in terms of the formation of clusters and clustering time when compared to the t-SNE. For each subject, data clustering for DBSCAN and t-SNE took approximately 30 minutes, so five hours in total for each algorithm to process all ten subjects.

This research determines that DBSCAN is more than capable of clustering MHEALTH data. The t-SNE models performance on clustering the data wasn't as effective of DBSCAN due to the presence of a vast amount of noise data points. It was difficult to visualise and analyse the t-SNE clusters correctly, leaving DBSCAN as the superior clustering algorithm. Each subject successfully visualised between 3 and 10 clusters with the Silhouette Coefficient ranging from 0.073 and 0.306, showing the consistency within the clusters of data.

The DBSCAN algorithm offers much better analysis characteristics than t-SNE also. These characteristics include:

- Estimated number of clusters.
- Estimated number of noise points.
- Homogeneity.
- Completeness.
- V-measure.

- Adjusted Rand Index.
- Adjusted Mutual Information.
- Silhouette Coefficient.

The t-SNE clustering algorithm can only be measured by KL-divergence, even this characteristic doesn't present much information as to why the clusters prevailed. The main two reasons for the poor performance of t-SNE is:

- Dimensionality Reduction.
- The cost function is not convex, many optimisation parameters need to be set.
- The KL-divergence for each adjustment was between 4.1 and 4.9, reiterating the fact that the data structure was not intact during the mapping of the data points.

These findings presented in the data clustering conclusion reiterate the hypothesis that DBSCAN outperforms t-SNE and is highly suited to analysing MHEALTH data.

DBSCAN and t-SNE also significantly learned from the sensor signals generated from the gyroscope, accelerometer, magnetometer and electrocardiogram. Extracting features and labels was the only pre-processing needed to feed the data into DBSCAN and t-SNE. This analysis presents a solution that sensor fusion can be performed where sensor data is generated from different devices, and it can undoubtedly be processed the data clustering algorithms DBSCAN and t-SNE. If given more time, t-SNE would have performed successfully on the MHEALTH dataset. The elimination of the noise points would have benefited the clustering output.

10.4 Achievements

The goal of this research was to build and implement various machine learning models to correctly classify activities from the MHEALTH dataset, while conducting a comparative study on all six classification algorithms. The implemented models included random forest, XGBoost and various neural network architectures.

Another goal of this research was to build and implement two data clustering algorithms to present patterns and relationships in the data, while conducting a comparative study on both clustering algorithms.

This research achieved all of its original goals through implementing each algorithm on the data. Successful evaluation procedures allowed for the comparison of all the models, leading to insights as to why certain models outperformed others. It can be safely said that all six algorithms can be applied to mobile health data, with;

- XGBoost and MLP achieving 89% and 90% evaluation measures respectively.
- CNN, ConvLSTM and Autoencoder by Random Forest achieving evaluation measures of 85%, 84% and 83% respectively.
- LSTM achieving evaluation measures of 78%.

As seen by the feature importance analysis and multisensory fusion analysis, accelerometer, gyroscope and magnetometer data fused together benefit all of the above approaches. Each sensor has a significant effect on the output of each model, without one of these sensors, accuracy along with other evaluation metrics would surely decrease. Confusion matrix values would also soar leaving a high number of misclassified instances.

All of the approaches achieves greater performance measures than existing state-of-the-art baselines on the MHEALTH dataset. XGBoost achieved an accuracy of 24% greater than the lowest performing model in comparison 1, 21% greater than the lowest performing model in comparison 2, 2% greater than the lowest performing model in comparison 3 and 6% greater than the lowest performing model in comparison 4. This shows the profound effect XGBoost has on MHEALTH data and it can successfully classify activities with minimal misclassified instances. MLP also achieved outstanding results to existing baselines achieving 25% greater than the lowest performing model in comparison 1, 22% greater than the lowest performing model in comparison 2, 3% greater than the lowest performing model in comparison 3 and 7% greater than the lowest performing model in comparison 4.

In terms of the data, there was no missing values and little or no pre processing needed. The models were demonstrated on the raw, unstructured data showing the strong capabilities each one has in classifying MHEALTH activities. Each approach is implemented on the raw, unstructured sensor data with limited pre-processing, eliminating the concept of engineering bias.

In a sense of personal achievement, I have become proficient in Python and am very confident in my coding abilities through the Python programming language now. I have benefited greatly from the completion of this research, gaining thorough knowledge of machine learning, deep learning, the Internet of Things, wearable electronics, mobile health data and how the mentioned above can benefit society in general if utilised properly.

10.5 Limitations of Human Activity Recognition

There are many limitations associated with Human Activity Recognition, which are further discussed in this section.

Challenges common to this research

The three main challenges presented by this research is unmeasurable uncertainty factors, activity similarity and the Null class problem.

Unmeasurable Uncertainty Factors

Building a Human Activity Recognition system that is diverse enough to deal with unmeasurable uncertainty factors is a difficult challenge. This confusion and uncertainty occurs in the system due to volunteers performing the activity in question differently. A number of factors influence this uncertainty such as tiredness, if the volunteer has an injury, jet lag, stress, emotional state (happy, sad, upset, confused etc.) and environmental state (walking up/down a hill/mountain in the snow, rain, sun etc.)

For example, the walking style may be different for a person 'A' walking in 50 degrees heat versus a person 'B' walking in minus 15 degrees or person 'C' walking in normal temperature such as 15 degrees. Person A may take significantly longer to walk a certain distance due to the extreme heat. This leads to a larger amount of data being captured by the activity recognition system, which may lead to unmeasurable uncertainty factors. The same can be said for a person strolling to work on a fresh summer morning with a cup of coffee versus the person walking home after a hard days' work. There is change in the movement due to many factors.

Unmeasurable uncertainty factors deals with a full-body human activity recognition model. To tailor for this uncertainty, models may revolve around interperson uncertainty. This model would be tailored to the volunteers needs taking into account variables which would affect the recognised activity. Models tailored to individuals rather than a generic model would tend to generate better performance results.

Activity Similarity

Another challenge presented by activity recognition is similar activities, which generate similar sensor data. For example, jogging and running, climbing stairs and cycling, turning on the washing machine and turning on the dryer. The only way to monitor this challenge is to create an individual activity recognition prediction model tailored to the individuals' needs. For distinguishing between turning on the washing machine and dryer, the model would need to classify that before turning on the washing machine that the clothes came from the wash basket, and before turning on the dryer that the clothes came from the washing machine. Additional features need to be captured by different sensor modalities or by continuously monitoring similar activities.

The NULL Class Problem

Human activity recognition systems contain a vast amount of streaming data. Only a certain percentage of this streaming data is significant in the performance of the HAR system. There is a slight imbalance between the portion of significant data and insignificant data. This leads to some of the activities to be easily confused with activities that have similar range of motion patterns and are irrelevant in predicting the

activity in question. For example, jogging is often mistaken for running and cycling is often mistaken for running upstairs. These easily confused activities are the so-called NULL class.

Detecting, monitoring and modelling the NULL class is a tough task. The NULL class often represents a massive portion of the dataset. As seen in [120], the NULL class represents 72.28% of the whole dataset. It is good practice to remove the Null class if there is a skewed pattern in the dataset. If the datasets attribute information and labels differ substantially from the correctly classified activities, then the NULL class problem may be identified and appropriate action or precautions taken.

The NULL class is not a huge problem and can be dealt with accordingly when analysing and evaluating datasets. At most, it leads to minor confusion when classifying activities in HAR systems. As seen in [122], applying self-learning can reap benefits of the NULL class. The studies in [122] present a performance comparison of self-learning activity spotters to show the benefits of this proposed approach. Results yielded an increase of 15% in performance, which outlines that the NULL class if managed accordingly can generate great model performance.

10.6 Repercussions of on-body inertial Sensor use in Healthcare

Schukat [142] describes the repercussions of wearable sensor use in healthcare. He identifies the four main unintended consequences as:

- Modifying and changing the actions of behaviour.
- Unforeseen creation of big data sets, leading to unintended use and misuse.
- Privacy and security issues revolving around sensor data.
- Unexpected challenges facing regulatory bodies. These bodies must regulate the security and safety of wearable sensors and relevant applications, which utilise the data in question.

The amount of unexpected consequences wearable sensors lead to is endless. It is near impossible to monitor, control and regulate all of these consequences. User actions and behaviour, information misuse, privacy and security downfalls and regulation challenges are the four main issues, although these issues are extremely broad and stretch across a continuous range of aspects.

Schukat [142] continues to discuss the repercussions of wearable sensor use in healthcare focusing on the creation of big data sets. Legislation laws will have a massive impact on the impact of wearable sensor use in the future. Whether it is a judge, lawyer, employer, the government or an insurer, discrimination towards an individual can occur due to the vast amount of data that can be created. There will be many debates about this topic in the future, with legal and political battles at the core. Although privacy and security is a huge concern, the good outweighs the bad if monitored correctly. As long as appropriate standardised laws are put in place to protect individuals and ensure the

data is only used for environmental and physical beneficial purposes, there won't be any drastic consequences. The future will only tell what will happen.

Hopefully, regulation is controlled to benefit the healthcare industry. Advancements in medical technology can save millions of lives per year. With the rapid increase in population in todays' society, medical technology along with related laws needs to keep advancing and updating at a rapid pace.

10.7 Future Applications

The following section presents an account of future work in Human activity recognition using Deep Learning. Below are a number of ways in which Human Activity Recognition Models using Deep Learning will benefit Healthcare:

- Remote Patient monitoring: clinician decision support
- Ambient assisted living / aiding the elderly.
- Drug discovery
- Developing regions whose healthcare services are limited. App with patient data, diagnostic abilities.
- Reduce need for electronic health records
- Creating more precise analytics for diagnosis
- Clinical decision making - decision support, risk scoring, early alerting
- Sepsis

Predictive analytics of medical devices in the ICU.

Human activity recognition models using deep learning could greatly benefit patients in the Intensive Care Unit (ICU). With the volume of patients around the world admitted to the ICU in hospitals every day, there would come times were clinicians will not always be in the same room as the patient to ensure they are doing ok. These HAR Deep learning models would detect vital signs signalling deterioration or detect if certain movements by the patient were becoming more difficult and leading to complications. This could save the patients' life and have a huge effect on their recovery process, ensuring that a fatality does not occur. It would improve healthcare quality in the inpatient hospital environment. Clinicians and doctors in the ICU would receive data regarding the patient and receive a detection signal if a deterioration or complication occurs. Doctors would provide aid to the ICU patient as quickly as possible.

Dependable, decisive risk predictor

The amount of patient data that a deep learning HAR system would generate is endless. Extracting, structuring and evaluating this data is a lengthy process. If the patient data was integrated into a dependable, decisive risk predictor, this would counteract any time factor as the result would benefit patients' health and save lives.

Managing the highly unstructured data on patients health is one of the stepping-stones needed to develop this risk predictor. As seen by the data in this research paper, data format is often raw, unstructured and hard to manipulate. An Artificial intelligence framework is needed to understand the data fully in order to make predictions leading to decision support. The framework needs to identify a time frame that builds up to the diagnosis while analysing the millions amounts of data points along the way. Dissecting these millions of data points while identifying significant irregular occurrences will lead to a successful risk predictor.

Health recommendations

The amount of patient-related data is continuously rising. Managing this data and gaining key individual insights can lead to medicine, exercise and diet recommendations. Implementing this through a HAR using deep learning framework can lead to an alternative viewpoint into a patients' health status, saving lives and reaping benefits in the long-term. Trusting these new, developed apps will take time for some patients but the more comfortable they get with sharing their health data with clinicians the more beneficial it will be for their health. It will make doctors and patients take better care of their patients by recommending certain exercises and dietary plans that will improve their overall well-being.

Developing regions

Developing regions, such as parts of Africa, lack certain healthcare facilities. There are shortages in qualified clinicians, medical assistants, nursing assistants, paramedics and dermatologists as well as many more healthcare and medical job titles. Having shortages in these roles limit the ability to provide sufficient healthcare in these developing regions. AI, DL and ML could counteract this shortage in developing regions by taking control of certain medical duties that trained healthcare providers perform.

For example, an AI tool could perform examining scans such as MRI's or ultrasounds. Chest, lung and abdomen scans could also be analysed, only to name a few. This would limit the need for trained clinicians on site in some areas, leading to patients being treated in a safe, appropriate manner. These tasks could be tackled with the creation of an app. Two main aspects the app would revolve around is individual patient data and diagnostic abilities.

10.8 Future Research Directions

This section focuses on discussing possible future research directions in the field of HAR, into which the research presented in this thesis can progress.

Long-term Monitoring

Given the time restraints and resources for this research, long-term monitoring of subjects' was not possible. Long-term monitoring using wearable sensors over a specific period under daily settings is a necessary step in modelling a human activity recognition framework to be deployed into the real world. For example, this monitoring could occur with patients in a medical facility. This type of extended study would generate useful sensor data for the analysis of long-term and high-level activities, some of which researchers are not studying in HAR context yet. It would discover many strengths and limitations that they are not familiar with yet. This type of research would require an extremely high-level approach, at PhD level or health organisation level, e.g. collaborating with hospitals, surgical centres', care clinics, dialysis centres, hospice homes, mental health and addiction treatment centres' and nursing homes.

Quality of Recognition and Performance

While monitoring patents' activity recognition in healthcare facilities, it is vital to know that an activity has been safely performed, but measuring the quality of performing the activity is even more important. Identifying and measuring the performance on an activity is testing, it is an open research question and a hot topic for discussion. There are a range of different factors that can affect the quality of recognition and performance such as stress, fatigue, hunger levels and environmental factors. Clinicians could provide patients an evaluation procedure; detailing dietary plans, rehabilitation exercises to fuel their body and generate energy.

Integration of Multiple Sensors

This research investigation revolved around the analysis of sensor data from accelerometers, gyroscopes and magnetometers to classifying activities. These sensors provide many benefits. Their characteristics make them highly suitable for human activity recognition. They are precise, highly accurate, are small and they are suitable for harsh environments ranging from cold to hot temperatures. These three sensors along may not be sufficient for the long-term monitoring of patients. Integrating these on-body sensors with other sensors such as in a smartphone may prove beneficial.

XGBoost Model Interpretability

An important addition to this project would be to focus more on the XGBoost implementation due to the successful performance it achieved. XGBoost is excellent for model interpretability, which is a huge aspect in machine and deep learning nowadays. Due to time constraints, analysing XGBoost shapley values wasn't feasible. Shapley values allows the XGBoost model to analyse a feature set and identify each feature's

marginal contribution to the overall classification prediction [144]. F. Santiago [144] defines interpreting of the Shapley value as;

'*The value of the feature A contributed X to the prediction of this particular instance compared to the average prediction for the dataset. More clearly, it is the contribution of the feature value to the difference between the actual prediction and the mean prediction*.'

Shapley values provide a very detailed account as which features greatly influenced the model. They offer transparency as well as global approximations. LIME (Local Interpretable Model-agnostic Explanations) is another technique, which would have benefited this research greatly. It also offers detailed account of model interpretability, detailing the highly important influential features. Lime offers local approximations while shapley offers global approximations. Upon extending this research, a comparison of both measures to improve model interpretability would greatly benefit the whole research. They both compile with General Data Protection Regulation (GDPR) law that all companies adhere by, unlike measures such as feature importance.

Bibliography

[1] *Heart.org*, 2019. [Online]. Available: https://www.heart.org/-/media/files/about-us/policy-research/policy-positions/clinical-care/remote-patient-monitoring-guidance-2019.pdf?la=en&hash=A98793D5A043AB9940424B8FB91D2E8D5A5B6BEB. [Accessed: 10- Jul- 2019].

[2] R. Chamberlain, J. Sond, K. Mahendraraj, C. Lau and B. Siracuse, "Determining 30-day readmission risk for heart failure patients: the Readmission After Heart Failure scale", *International Journal of General Medicine*, vol. 11, pp. 127-141, 2018. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5898587/ [Accessed 3 July 2019].

[3] Baños, O. (2014). *Datasets*. [online] Orestibanos.com. Available at: http://orestibanos.com/datasets.htm [Accessed 19 Jun. 2019].

[4] L. Milačić, S. Jović, T. Vujović and J. Miljković, "Application of artificial neural network with extreme learning machine for economic growth estimation", *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 285-288, 2017. Available: 10.1016/j.physa.2016.08.040 [Accessed 8 July 2019].

[5] D. MICHIE, ""Memo" Functions and Machine Learning", *Nature*, vol. 218, no. 5136, pp. 19-22, 1968. Available: https://stacks.stanford.edu/file/druid:jt687kv7146/jt687kv7146.pdf. [Accessed 14 June 2019].

[6] G. Hinton, S. Osindero and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006. Available at: https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf. [Accessed 16 July 2019].

[7] X. Wei, J. Zhu, S. Yuan and H. Su, "Sparse Adversarial Perturbations for Videos", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8973-8980, 2019. Available: https://arxiv.org/pdf/1809.00958.pdf. [Accessed 21 May 2019].

[8] "AI vs Machine Learning vs Deep Learning | Edureka", *Edureka*, 2019. [Online]. Available: https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/. [Accessed: 19- Jul- 2019].

[9] J. Wiener, R. Hanley, R. Clark and J. Van Nostrand, "Measuring the Activities of Daily Living: Comparisons Across National Surveys", *Journal of Gerontology*, vol. 45, no. 6, pp. S229-S237, 1990. Available: https://academic.oup.com/geronj/article/45/6/S229/706347. [Accessed 15 June 2019].

[10] "classification Archives |", *Semantive.com*, 2019. [Online]. Available: https://semantive.com/tag/classification/. [Accessed: 08- May- 2019].

[11] M. Bilal, "A Review of Internet of Things Architecture, Technologies and Analysis Smartphone-based Attacks Against 3D printers", 2019. Available: https://pdfs.semanticscholar.org/b81e/4ac8f04d9f94b947a4aa57bd5357251eb4bf.pdf. [Accessed 22 July 2019].

[12] T. Ya and C. Wenjie, "MEMS-based human activity recognition using smartphones", *Proceedings of the 35th Chinese Control Conference*, 2016. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7553975&tag=1. [Accessed 22 June 2019].

[13] *Intel.com*, 2019. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/images/iot/guide-to-iot-infographic.png. [Accessed: 20- Jul- 2019].

[14] *Frontier-economics.com*, 2019. [Online]. Available: https://www.frontier-economics.com/media/1167/201803_the-economic-impact-of-iot_frontier.pdf. [Accessed: 13- Jun- 2019].

[15] M. Mathie, N. Lovell, A. Coster and B. Celler, "Determining activity using a triaxial accelerometer", *Proceedings of the Second Joint EMBSBMES Conference*, 2002. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1053385. [Accessed 6 July 2019].

[16] H. Nait-Charif and S. McKenna, "Activity Summarisation and Fall Detection in a Supportive Home Environment", *Pattern Recognition, 2004.* ICPR 2004. Proceedings of the 17th International Conference on, Volume: 4. Available: http://staff.computing.dundee.ac.uk/stephen/icpr2004_2.pdf. [Accessed: 18- May- 2019].

[47] E. Garcia-Ceja, R. Brena, J. Carrasco-Jimenez and L. Garrido, "Long-Term Activity Recognition from Wristwatch Accelerometer Data", *Sensors*, vol. 14, no. 12, pp. 22500-22524, 2014. Available: 10.3390/s141222500 [Accessed 9 May 2019].

[18] P. Gupta and T. Dallas, "Feature Selection and Activity Recognition System Using a Single Triaxial Accelerometer", *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 6, pp. 1780-1786, 2014. Available: 10.1109/tbme.2014.2307069 [Accessed 29 June 2019].

[19] A. Salarian, H. Russmann, F. Vingerhoets, P. Burkhard and K. Aminian, "Ambulatory Monitoring of Physical Activities in Patients With Parkinson's Disease", *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 12, pp. 2296-2299, 2007. Available: 10.1109/tbme.2007.896591 [Accessed 2 June 2019].

[20] M. Hassan, M. Uddin, A. Mohamed and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning", *Future Generation Computer Systems*, vol. 81, pp. 307-313, 2018. Available: 10.1016/j.future.2017.11.029 [Accessed 6 June 2019].

[21] C. Ronao and S. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks", *Expert Systems with Applications*, vol. 59, pp. 235-244, 2016. Available: 10.1016/j.eswa.2016.04.032 [Accessed 31 July 2019].

[22] S. Mekruksavanich, N. Hnoohom and A. Jitpattanakul, "Smartwatch-based sitting detection with human activity recognition for office workers syndrome", *2018 International ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI-NCON)*, 2019. Available: https://ieeexplore.ieee.org/document/8378302. [Accessed 22 May 2019].

[23] Bai, Bai, "Magnetometer - an overview | ScienceDirect Topics", *Sciencedirect.com*, 2019. [Online]. Available: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/magnetometer. [Accessed: 20- May- 2019].

[24] K. Altun and B. Barshan, "Human Activity Recognition Using Inertial/Magnetic Sensor Units", *HBU*, 2019. [Online]. Available: http://repository.bilkent.edu.tr/bitstream/handle/11693/28535/bilkent-research-paper.pdf?sequence=1. [Accessed: 03- Jul- 2019].

[25] L. Milačić, S. Jović, T. Vujović and J. Miljković, "Application of artificial neural network with extreme learning machine for economic growth estimation", *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 285-288, 2017. Available: https://reader.elsevier.com/reader/sd/pii/S037843711630557X?token=3F6B99C1F4D1E27650EE91FB60CEA6F351BA3988D1C101361FED1DA4A0F53FC998A41B7C0005D112BEAD0A22551439AD. [Accessed 2 June 2019].

[26] M. Mathie, N. Lovell, A. Coster and B. Celler, "Determining activity using a triaxial accelerometer", *Proceedings of the Second Joint EMBSBMES Conference*, 2002. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1053385. [Accessed 6 July 2019].

[27] Y. Sun, H. Lv, X. Liu, P. Xu, Y. Huang and Y. Sun, Personalized recommendation for Weibo comic users. *2018 Wireless Telecommunications Symposium (WTS)*, 1-6. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8363939. [Accessed 10 July 2019].

[28] A. Castro-Lopez, R. Vazquez-Casielles and J. Puente, "How to manage the online experience concerning transactional and experimental customers: Case of e-fashion sector", *Journal of Business Economics and Management*, vol. 20, no. 3, 2019. Available: https://journals.vgtu.lt/index.php/JBEM/article/view/9860/8847. [Accessed 24 July 2019].

[29] N. Murata, S. Yoshizawa and S. Amari, "Network information criterion-determining the number of hidden units for an artificial neural network model", *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 865-872, 1994. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=329683 [Accessed 5 July 2019].

[30] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou and Y. Amirat, "Physical Human Activity Recognition Using Wearable Sensors", *Sensors*, vol. 15, no. 12, pp. 31314-31338, 2015. Available at: https://www.mdpi.com/1424-8220/15/12/29858/htm [Accessed 7 June 2019].

[31] O. Banos, R. Garcia and A. Saez, "UCI Machine Learning Repository: MHEALTH Dataset Data Set", *Archive.ics.uci.edu*, 2019. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/mhealth+dataset. [Accessed: 09- Jun- 2019].

[32] "Introduction to Mobile Robot Control | ScienceDirect", *Sciencedirect.com*, 2019. [Online]. Available: https://www.sciencedirect.com/book/9780124170490/introduction-to-mobile-robot-control. [Accessed: 11- Aug- 2019].

[33] "eMaintenance | ScienceDirect", *Sciencedirect.com*, 2019. [Online]. Available: https://www.sciencedirect.com/book/9780128111536/emaintenance. [Accessed: 22- May- 2019].

[34] H. Foroughi, B. Shakeri Aski and H. Pourezza, "Intelligent Video Surveillance for Monitoring Fall Detection of Elderly in Home Environments", *Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008)*, 2008. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4803020. [Accessed 10 July 2019].

[35] A. Williams, D. Xie and S. Ou, "Distributed Smart Cameras for Aging in Place", 2006. Available at: https://www-robotics.cs.umass.edu/uploads/Main/dsc06.pdf. [Accessed 7 August 2019].

[36] F. Sposaro and G. Tyson, "iFall: An Android Application for Fall Monitoring and Response", *31st Annual International Conference of the IEEE EMBS*, 2009. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5334912 [Accessed 11 June 2019].

[37] H. Nait-Charif and S. J. McKenna, "Activity Summarisation and Fall Detection in a Supportive Home Environment", *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, Volume: 4*. Available: http://staff.computing.dundee.ac.uk/stephen/icpr2004_2.pdf. [Accessed: 02- Jul- 2019].

[38] H. Ting Cheng and W. Zhuang, "Bluetooth-enabled in-home patient monitoring system: Early detection of Alzheimer's disease - IEEE Journals & Magazine", *IEEE Wireless Communications, 17, 2010*. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5416353. [Accessed: 13- Jun- 2019].

[39] M. López et al., "SVM-based CAD system for early detection of the Alzheimer's disease using kernel PCA and LDA", *Neuroscience Letters*, vol. 464, no. 3, pp. 233-

238, 2009. Available: https://reader.elsevier.com/reader/sd/pii/S0304394009011677?token=419633F14ABD8CB6E7340CB1CD6D6AE0539FE7455365CC6F3B8E0CF7CBD108C68610FAC456CD25DA2E06D48F845BF86F. [Accessed 15 July 2019].

[40] H. Ting Cheng and W. Zhuang, "Bluetooth-enabled in-home patient monitoring system: Early detection of Alzheimer's disease - IEEE Journals & Magazine", *Ieeexplore.ieee.org*, 2010. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5416353. [Accessed: 13- Jun- 2019].

[41] T. Nicolai, T. Sindt, H. Witt, J. Reimerdes and H. Kenn, "Wearable Computing for Aircraft Maintenance: Simplifying the User Interface", *Citeseerx.ist.psu.edu*, 2006. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.8520&rep=rep1&type=pdf. [Accessed: 18- Jun- 2019].

[42] M. Lampe, M. Strassner and E. Fleisch, "A Ubiquitous Computing Environment for Aircraft Maintenance", *Vs.inf.ethz.ch*, 2019. Available: http://www.vs.inf.ethz.ch/publ/papers/SAC2004_AircraftMaintenance.pdf. [Accessed: 19- Jun- 2019].

[43] I. Maurtua, P. T. Kirisci, T. Stiefmieir, M. Luco Sbodio and H. Witt, "A Wearable Computing Prototype for supporting training activities in Automotive Production - VDE Conference Publication", *Ieeexplore.ieee.org*, 2019. Available: https://ieeexplore.ieee.org/abstract/document/5760480. [Accessed: 22- Jun- 2019].

[44] M. Aleksy and M. Rissanen, "Utilizing wearable computing in industrial service applications", *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 4, pp. 443-454, 2012. Available: 10.1007/s12652-012-0114-2 [Accessed 4 July 2019].

[45] A. Cheok et al., "Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing", *Personal and Ubiquitous Computing*, vol. 8, no. 2, pp. 71-81, 2004. Available: 10.1007/s00779-004-0267-x [Accessed 17 July 2019].

[46] H. Tobita and T. Kuzi, "SmartWig: Wig-based Wearable Computing Device for Communication and Entertainment", 2012. [Online]. Available: http://delivery.acm.org/10.1145/2260000/2254613/p299-tobita.pdf?ip=140.203.12.9&id=2254613&acc=ACTIVE%20SERVICE&key=846C3111CE4A4710%2E5E11B48930E8A046%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1566465463_60c1670911ebaae8bf17d9e73a6000a6. [Accessed: 13- Jul- 2019].

[48] S. Zhu, Shangyue & Xu, Junhong & Guo, Hanqing & Liu, Qiwei & Wu, Shaoen & Wang, Honggang. (2018). Indoor Human Activity Recognition Based on Ambient Radar with Signal Processing and Machine Learning. 2018 IEEE International

Conference on Communications (ICC 2018). 1-6. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8422107&tag=1. [Accessed 20 July 2019].

[49] A. Salguero, M. Espinilla, P. Delatorre and J. Medina, "Using Ontologies for the Online Recognition of Activities of Daily Living", *Sensors*, vol. 18, no. 4, p. 1202, 2018. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5948724/. [Accessed 13 June 2019].

[50] J. Wu, Y. Feng and P. Sun, "Sensor Fusion for Recognition of Activities of Daily Living", *Sensors*, vol. 18, no. 11, p. 4029, 2018. Available: https://www.ncbi.nlm.nih.gov/pubmed/30463199. [Accessed 4 May 2019].

[51] N. Murata, S. Yoshizawa and S. Amari, "Network information criterion-determining the number of hidden units for an artificial neural network model", *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 865-872, 1994. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=329683. [Accessed 18 July 2019].

[52] Bre, Facundo & Gimenez, Juan & D. Fachinotti, Víctor. (2017). Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. *Energy and Buildings*. 158. Available: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051. [Accessed: 22- Jun- 2019].

[53] Sun, Yan Lindsay et al. "Personalized recommendation for Weibo comic users." *2018 Wireless Telecommunications Symposium (WTS)* (2018): 1-6. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8363939. [Accessed 10 July 2019].

[54] A. Castro-Lopez, R. Vazquez-Casielles and J. Puente, "How to manage the online experience concerning transactional and experimental customers: Case of e-fashion sector", *Journal of Business Economics and Management*, vol. 20, no. 3, 2019. Available: https://journals.vgtu.lt/index.php/JBEM/article/view/9860/8847. [Accessed 24 July 2019].

[55] L. Claesson and B. Hansson, "Deep Learning Methods and Applications Classification of Traffic Signs and Detection of Alzheimer's Disease from Images", 2017. Available: https://pdfs.semanticscholar.org/d532/777f2386766e7c93c0bf4257d0a359e91f6b.pdf. [Accessed 13 July 2019].

[56] W. Luo, Y. Li, R. Urtasun and R. Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks", 2019. Available: http://papers.nips.cc/paper/6203-understanding-the-effective-receptive-field-in-deep-convolutional-neural-networks.pdf. [Accessed 8 July 2019].

[57] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural Networks*, Vol 61, pp 85-117, Jan 2015. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135. [Accessed 30 June 2019].

[58] M. Mody, M. Mathew, S. Jagannathan, A. Redfern, J. Jones and T. Lorenzen, "CNN Inference: VLSI Architecture for Convolution Layer for 1.2 TOPS", *Ieeexplore.ieee.org*, 2017. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8226028. [Accessed: 10- Jun- 2019].

[59] W. Jiang and Z. Yin, "Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks", 2015. *Proc: the 23rd ACM international conference* 1307-1310.Available: http://web.mst.edu/~yinz/Papers/ACMMM2015_ActivityRecognition.pdf. [Accessed 17 July 2019].

[60] O. Banos, R. Garcia and A. Saez, "UCI Machine Learning Repository: MHEALTH Dataset Data Set", *Archive.ics.uci.edu*, 2019. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/mhealth+dataset. [Accessed: 09- Jun- 2019].

[61] M. Zhang and A. A. Sawchuk, "Human Activities Dataset", *Sipi.usc.edu*, 2012. [Online]. Available: http://sipi.usc.edu/had/. [Accessed: 18- Jul- 2019].

[62] M. Shoaib, S. Bosch, O. Durmaz Incel and H. Scholten, "Fusion of Smartphone Motion Sensors for Physical Activity Recognition", *Sensors, 2014*. Available: https://www.researchgate.net/publication/263015474_Fusion_of_Smartphone_Motion_Sensors_for_Physical_Activity_Recognition. [Accessed 9 July 2019].

[63] N. Y. Hammerla, S. Halloran and T. Plotz, "Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables", *IJCAI 2016*. Available: https://arxiv.org/pdf/1604.08880.pdf. [Accessed 28 June 2019].

[64] D. Roggen, A. Calatroni, L. Nguyen-Dinh, R. Chavarriaga, H. Sagha and S. Tejaswi Digumarti, "UCI Machine Learning Repository: OPPORTUNITY Activity Recognition Data Set", *Archive.ics.uci.edu*, 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition. [Accessed: 16- Jun- 2019].

[65] A. Reiss, "PAMAP2 Physical Activity Monitoring Data Set", 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring. [Accessed: 10- Jun- 2019].

[66] Kim, Youngwook & Moon, Taesup. (2015). Human Detection and Activity Classification Based on Micro-Doppler Signatures Using Deep Convolutional Neural Networks. *IEEE Geoscience and Remote Sensing Letters*. 13. 1-5.

10.1109/LGRS.2015.2491329. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7314905. [Accessed 3 June 2019].

[67] "The vanishing gradient problem and ReLUs - a TensorFlow investigation - Adventures in Machine Learning", *Adventures in Machine Learning*, 2019. [Online]. Available: http://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/. [Accessed: 05- Jun- 2019].

[68] F. Javier Ordóñez, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition", *Sensors 2016*, vol. 16, no. 115, 2016. Available: https://www.mdpi.com/1424-8220/16/1/115/htm. [Accessed 29 June 2019].

[69] N. K. Ahmed, A. F. Atiya, N. El Gayar and H. El-Shishiny, "An Empirical Comparison of Machine Learning Models for Time Series Forecasting", *Econometric Reviews*, vol. 295-6, pp. 594-621, 2010. [Accessed 17 June 2019].

[70] *Pages.cs.wisc.edu*, 2019. [Online]. Available: http://pages.cs.wisc.edu/~shavlik/cs638/lectureNotes/Long%20ShortTerm%20Memory%20Networks.pdf. [Accessed: 11- Jul- 2019].

[71] S. Polamuri, "Difference Between Softmax Function and Sigmoid Function", *Dataaspirant*, 2017. [Online]. Available: http://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/. [Accessed: 20- Jul- 2019].

[72] N. K. Ahmed, A. F. Atiya, N. El Gayar and H. El-Shishiny, "An Empirical Comparison of Machine Learning Models for Time Series Forecasting", *Econometric Reviews*, vol. 295-6, pp. 594-621, 2010. [Accessed 17 June 2019].

[73] M. Inoue, S. Inoue and T. Nishida, "Deep recurrent neural network for mobile human activity recognition with high throughput", *Artificial Life and Robotics*, no. 23, pp. 173–185, 2018. Available: https://link.springer.com/content/pdf/10.1007%2Fs10015-017-0422-x.pdf. [Accessed 5 July 2019].

[74] Y. Du, W. Wang and L. Wang, "Hierarchical Recurrent Neural Network for Skeleton Based Action Recognition", *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Available: http://openaccess.thecvf.com/content_cvpr_2015/papers/Du_Hierarchical_Recurrent_Neural_2015_CVPR_paper.pdf. [Accessed 4 July 2019].

[75] "Datasets", *Wangjiangb.github.io*, 2014. [Online]. Available: http://wangjiangb.github.io/my_data.html. [Accessed: 26- Jul- 2019].

[76] "Berkeley MHAD | Teleimmersion Lab", *Tele-immersion.citris-uc.org*, 2013. [Online]. Available: https://tele-immersion.citris-uc.org/berkeley_mhad. [Accessed: 12- Jul- 2019].

[77] "Motion Database HDM05", *Resources.mpi-inf.mpg.de*, 2007. [Online]. Available: http://resources.mpi-inf.mpg.de/HDM05/. [Accessed: 22- Aug- 2019].

[78] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan and J. Liu, "Online Human Action Detection Using Joint Classification-Regression Recurrent Neural Networks", *Computer Vision – ECCV 2016: 14th European Conference*, pp. Part VII (203-220), 2016. Available: https://arxiv.org/pdf/1604.05633.pdf. [Accessed 17 June 2019].

[79] S. Kim, S. Hong, M. Joh and S. Song, "DEEPRAIN: CONVLSTM NETWORK FOR PRECIPITATION PREDICTION USING MULTICHANNEL RADAR DATA", 2017. Available: https://arxiv.org/pdf/1711.02316.pdf. [Accessed 18 July 2019].

[80] Y. Liu, H. Zheng, X. Feng and Z. Chen, "Short-Term Traffic Flow Prediction with Conv-LSTM", *In Proceedings of the 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1-6, 2017. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8171119&tag=1. [Accessed 18 June 2019].

[81] Y. Guo, Z. Wu and Y. Ji, "A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction", *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, 2017. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8113070. [Accessed 19 June 2019].

[82] X. Shi, Z. Chen, H. Wang and D. Yeung, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting", *ArXiv*, vol. 150604214, 2015. Available: https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf. [Accessed 23 July 2019].

[83] X. Shi, Z. Chen, H. Wang and D. Yeung, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting", *ArXiv*, vol. 150604214, 2015. Available at : https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf [Accessed 23 July 2019].

[84] Wang, Xufei & Liao, Weixian & Guo, Yifan & Yu, Lixing & Wang, Qianlong & Pan, Miao & Li, Pan. (2019). PerRNN: Personalized Recurrent Neural Networks for Acceleration-Based Human Activity Recognition. 1-6. 10.1109/ICC.2019.8761931. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8761931. [Accessed 23 July 2019].

[85] Saeedi, Ramyar & Norgaard, Skyler & Gebremedhin, Assefaw. (2017). A Closed-loop Deep Learning Architecture for Robust Activity Recognition using Wearable Sensors. 10.1109/BigData.2017.8257960. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8257960. [Accessed 23 July 2019].

[86] Sudhakaran, Swathikiran & Lanz, Oswald. (2017). Convolutional Long Short-Term Memory Networks for Recognizing First Person Interactions. Available at: http://openaccess.thecvf.com/content_ICCV_2017_workshops/papers/w34/networks_sudhakaranfbk.eu_lanzfbk.eu_ICCV_2017_paper.pdf. [Accessed 18 June 2019].

[87] "Multilayer Perceptron — DeepLearning 0.1 documentation", *Deeplearning.net*, 2018. [Online]. Available: http://deeplearning.net/tutorial/mlp.html. [Accessed: 10- Jul- 2019].

[88] "Affine Transformation -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2019. [Online]. Available: http://mathworld.wolfram.com/AffineTransformation.html. [Accessed: 06- Jul- 2019].

[89] Mo, L., Li, F., Zhu, Y., & Huang, A. (2016). Human physical activity recognition based on computer vision with deep learning model. *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 1-6. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7520541&tag=1. [Accessed: 06- Jul- 2019].

[90] "Cornell Activity Datasets: CAD-60 & CAD-120 | re3data.org", *Re3data.org*, 2019. [Online]. Available: https://www.re3data.org/repository/r3d100012216. [Accessed: 10- Aug- 2019].

[91] C. Catal, S. Tufekci, E. Pirmit and G. Kocabag, "On the use of ensemble of classifiers for accelerometer-based activity recognition", *Applied Soft Computing*, vol. 37, pp. 1018-1022, 2015. Available: https://www.sciencedirect.com/science/article/pii/S1568494615000447. [Accessed 3 August 2019].

[92] [9]"WISDM Lab: Dataset", *Cis.fordham.edu*, 2012. [Online]. Available: http://www.cis.fordham.edu/wisdm/dataset.php. [Accessed: 25- Jul- 2019].

[93] J. Talukdar and B. Mehta, "Human Action Recognition System using Good Features and Multilayer Perceptron Network", *6th IEEE International Conference on Communication and Signal Processing (ICCSP), At India*, 2017. Available: https://arxiv.org/ftp/arxiv/papers/1708/1708.06794.pdf. [Accessed 17 July 2019].

[94] J. Cha, K. Soo Kim and S. Lee, "On the Transformation of Latent Space in Autoencoders", *ArXiv, 2009,* 2019. Available: https://arxiv.org/pdf/1901.08479.pdf. [Accessed 5 August 2019].

[95] L. Wang, "Recognition of Human Activities Using Continuous Autoencoders with Wearable Sensors", *Sensors*, vol. 16, no. 2, p. 189, 2016. Available: https://www.ncbi.nlm.nih.gov/pubmed/26861319. [Accessed 21 July 2019].

[96] D. Surendran, "a Swiss Roll Dataset", *People.cs.uchicago.edu*, 2019. [Online]. Available: http://people.cs.uchicago.edu/~dinoj/manifold/swissroll.html. [Accessed: 18- Jul- 2019].

[97] M. Zhang and A. A. Sawchuk, "Human Activities Dataset", *Sipi.usc.edu*, 2012. [Online]. Available: http://sipi.usc.edu/had/. [Accessed: 03- Jul- 2019].

[98] H. Zou, Y. Zhou, J. Yang, H. Jiang, L. Xie, C. J. Spanos (2018). DeepSense: Device-Free Human Activity Recognition via Autoencoder Long-Term Recurrent Convolutional Network. *2018 IEEE International Conference on Communications (ICC)*, 1-6. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8422895&tag=1. [Accessed: 01- Jul- 2019].

[99] B. Almaslukh, J. AlMuhtadi and A. Artoli, "An Effective Deep Autoencoder Approach for Online SmartphoneBased Human Activity Recognition", 2017. Available: https://pdfs.semanticscholar.org/d22c/6f4e21ffd87554fac39b9fb43b84f8dd8118.pdf. [Accessed 22 June 2019].

[100] L. Gigović, H. Pourghasemi, S. Drobnjak and S. Bai, "Testing a New Ensemble Model Based on SVM and Random Forest in Forest Fire Susceptibility Assessment and Its Mapping in Serbia's Tara National Park", *Forests*, vol. 10, no. 5, p. 408, 2019. Available: 10.3390/f10050408.

[101] A. Ignatov, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks", *Applied Soft Computing*, vol. 62, pp. 915-922, 2018. Available: https://www.sciencedirect.com/science/article/pii/S1568494617305665. [Accessed 19 July 2019].

[102] C. Hu, Y. Chen, L. Hu and X. Peng, "A novel random forests based class incremental learning method for activity recognition", *Pattern Recognition*, vol. 78, pp. 277-290, 2018. Available: https://www.sciencedirect.com/science/article/pii/S0031320318300360. [Accessed 8 July 2019].

[103] B. Barshan, "UCI Machine Learning Repository: Daily and Sports Activities Data Set", *Archive.ics.uci.edu*, 2013. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/daily+and+sports+activities. [Accessed: 21- Jul- 2019].

[104] J. L. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto and X. Parra, "UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set", *Archive.ics.uci.edu*, 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones. [Accessed: 13- Jul- 2019].

[105] D. Roggen, A. Calatroni, L. Nguyen-Dinh, R. Chavarriaga, H. Sagha and S. Tejaswi Digumarti, "UCI Machine Learning Repository: OPPORTUNITY Activity Recognition Data Set", *Archive.ics.uci.edu*, 2012. [Online]. Available:

https://archive.ics.uci.edu/ml/datasets/opportunity+activity+recognition. [Accessed: 16- Jun- 2019].

[107] J. L. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto and X. Parra, "UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set", *Archive.ics.uci.edu*, 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones. [Accessed: 13- Jul- 2019].

[108] V. Ayumi, "Pose-based Human Action Recognition with Extreme Gradient Boosting", *IEEE*, 2016. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7810099&tag=1. [Accessed 5 August 2019].

[109] L. Xia, C. Chen and J. Aggarwal, "View invariant human action recognition using histograms of 3D joints", *Cvrc.ece.utexas.edu*, 2012. [Online]. Available: http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html. [Accessed: 22- Jul- 2019].

[110] "ACASVA Downloads", *Cvssp.org*, 2010. [Online]. Available: https://www.cvssp.org/acasva/Downloads. [Accessed: 16- Jul- 2019].

[111] W. Zhang, X. Zhao and Z. Li, "A Comprehensive Study of Smartphone-Based Indoor Activity Recognition via Xgboost", *IEEE Access*, vol. 7, pp. 80027-80042, 2019. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8736849 [Accessed 18 July 2019].

[112] T. Nguyen, D. Fernandez, Q. Nguyen and E. Bagheri, "Location-Aware Human Activity Recognition", *Advanced Data Mining and Applications*, pp. 821-835, 2017. Available: https://link.springer.com/chapter/10.1007/978-3-319-69179-4_58. [Accessed 14 July 2019].

[113] B. Erol and M. Amin, "Radar Data Cube Processing for Human Activity Recognition Using Multi Subspace Learning", *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1-1, 2019. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8691492 [Accessed 2 July 2019].

[114] M. Hassan, S. Huda, M. Uddin, A. Almogren and M. Alrubaian, "Human Activity Recognition from Body Sensor Data using Deep Learning", *Journal of Medical Systems*, vol. 42, no. 6, 2018. Available: https://link.springer.com/content/pdf/10.1007%2Fs10916-018-0948-z.pdf. [Accessed 1 June 2019].

[115] I. El Moudden, H. Jouhari, S. El Bernoussi and M. Ouzir, "Learned Model for Human Activity Recognition Based on Dimensionality Reduction", *SSRN Electronic Journal*, 2018. Available:

https://www.researchgate.net/publication/323431511_Learned_Model_For_Human_Activity_Recognition_Based_On_Dimensionality_Reduction. [Accessed 17 July 2019].

[116] Y. Kwon, K. Kang and C. Bae, "Unsupervised learning for human activity recognition using smartphone sensors", *Expert Systems with Applications*, vol. 41, no. 14, pp. 6067-6074, 2014. Available: https://www.sciencedirect.com/science/article/pii/S0957417414002607. [Accessed 20 July 2019].

[117] J. Guo, X. Zhou, Y. Sun, G. Ping, G. Zhao and Z. Li, "Smartphone-Based Patients' Activity Recognition by Using a Self-Learning Scheme for Medical Monitoring", *Journal of Medical Systems*, vol. 40, no. 6, 2016. Available: https://link.springer.com/content/pdf/10.1007%2Fs10916-016-0497-2.pdf. [Accessed 5 August 2019].

[118] Figure N. Reunanen, V. Kononen, H. Halva, J. Mantyjarvi, A. Lamsa and J. Liikka, "Computational Graph Approach for Detection of Composite Human Activities", 2018. Available: https://arxiv.org/pdf/1812.01895.pdf. [Accessed 20 July 2019].

[119] Figure E. Brophy, J. Veiga, Z. Weng, A. Smeaton and T. Ward, "https://www.researchgate.net/publication/329387709_An_Interpretable_Machine_Vision_Approach_to_Human_Activity_Recognition_using_Photoplethysmograph_Sensor_Data", 2018. Available: https://arxiv.org/pdf/1812.00668.pdf. [Accessed 5 August 2019].

[120] F. Li, K. Shirahama, M. Nisar, L. Köping and M. Grzegorzek, "Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors", *Sensors*, vol. 18, no. 3, p. 679, 2018. Available: https://www.ncbi.nlm.nih.gov/pubmed/29495310. [Accessed 15 June 2019].

[122] O. Amft, "https://www.researchgate.net/publication/221240864_Self-Taught_Learning_for_Activity_Spotting_in_On-body_Motion_Sensor_Data", *ISWC*, pp. 83-86, 2011. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5959599&tag=1. [Accessed 29 July 2019].

[123] Y. Ho and D. Pepyne, "Simple Explanation of the No-Free-Lunch Theorem and Its Implications", *Journal of Optimization Theory and Applications*, vol. 115, no. 3, pp. 549-570, 2002. Available: https://link.springer.com/content/pdf/10.1023/A:1021251113462.pdf. [Accessed 29 July 2019].

[124] Ieeexplore.ieee.org. (2019). *Electronic Circuit and System Design using Python and VHDL - IEEE Conference Publication*. [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8620048 [Accessed 17 Jul. 2019].

[125] Keras.io. (2019). *Home - Keras Documentation*. [online] Available at: https://keras.io/ [Accessed 12 Jul. 2019].

[126] Pandas.pydata.org. (2019). *Python Data Analysis Library — pandas: Python Data Analysis Library*. [online] Available at: https://pandas.pydata.org/ [Accessed 6 Jul. 2019].

[127] Docs.python.org. (2019). *pickle — Python object serialization — Python 3.7.4 documentation*. [online] Available at: https://docs.python.org/3/library/pickle.html [Accessed 24 Jun. 2019].

[128] Scikit-learn.org. (2019). *scikit-learn: machine learning in Python — scikit-learn 0.21.3 documentation*. [online] Available at: https://scikit-learn.org/stable/ [Accessed 8 Jul. 2019].

[129] Numpy.org. (2019). *NumPy — NumPy*. [online] Available at: https://numpy.org/ [Accessed 21 Jun. 2019].

[130] Matplotlib.org. (2019). *Matplotlib: Python plotting — Matplotlib 3.1.1 documentation*. [online] Available at: https://matplotlib.org/ [Accessed 31 Jun. 2019].

[131] Docs.python.org. (2019). *8.3. collections — High-performance container datatypes — Python 2.7.16 documentation*. [online] Available at: https://docs.python.org/2/library/collections.html [Accessed 20 Jun. 2019].

[132] PyPI. (2019). *tqdm*. [online] Available at: https://pypi.org/project/tqdm/4.4.0/ [Accessed 22 Jul. 2019].

[133] Seaborn.pydata.org. (2019). *seaborn: statistical data visualization — seaborn 0.9.0 documentation*. [online] Available at: https://seaborn.pydata.org/ [Accessed 12 Jun. 2019].

[134] Xgboost.readthedocs.io. (2019). *XGBoost Documentation — xgboost 1.0.0-SNAPSHOT documentation*. [online] Available at: https://xgboost.readthedocs.io/en/latest/ [Accessed 30 Jul. 2019].

[135] Highcharts.com. (2019). *Interactive JavaScript charts for your webpage | Highcharts*. [online] Available at: https://www.highcharts.com/ [Accessed 28 Jun. 2019].

[136] Rousseeuw, P. (2019). *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*. [online] Available at: https://www.sciencedirect.com/science/article/pii/0377042787901257 [Accessed 22 Jul. 2019].

[137] Nandana, G., Mala, S. and Rawat, A. (2019). Hotspot Detection of Dengue Fever Outbreaks Using DBSCAN Algorithm. [online] Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8776916&tag=1 [Accessed 20 Jul. 2019].

[138] Ha, S. and Choi, S. (2016). Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. *2016 International Joint Conference on Neural Networks* (*IJCNN*). Available at: https://ieeexplore.ieee.org/document/7727224 [Accessed 29 Jul. 2019].

[139] Chen, K., Yao, L., Wang, X., Zhang, D., Gu, T., Yu, Z. and Yang, Z. (2018). interpretable parallel recurrent neural networks with convolutional attentions for multi-modality activity modeling. *arXiv*. Available at: https://arxiv.org/pdf/1805.07233.pdf [Accessed 26 Jun. 2019].

[140] Uddin, M. and Hassan, M. (2018). Activity Recognition for Cognitive Assistance Using Body Sensors Data and Deep Convolutional Neural Network. [online] Available at: https://ieeexplore.ieee.org/document/8469048 [Accessed 21 Jun. 2019].

[141] Kutlay, M. and Gagula-Palalic, S. (2016). Application Of Machine Learning In Healthcare: Analysis On MHEALTH Dataset. *Southeast Europe Journal of Soft Computing*, 4(2).

[142] McCaldin, D., Wang, K., Schreier, G., Lovell, N., Marschollek, M., Redmond, S. and Schukat, M. (2016). Unintended Consequences of Wearable Sensor Use in Healthcare. *Yearbook of Medical Informatics*, 25(01), pp.73-86.

[143] Simplilearn.com. (2019). *Random Forest Algorithm*. [online] Available at: https://www.simplilearn.com/random-forest-algorithm-article [Accessed 27 Jun. 2019].

[144] Medium. (2019). *Model interpretability — Making your model confess: Shapley values*. [online] Available at: https://medium.com/@santiagof/model-interpretability-making-your-model-confess-shapley-values-5fb95a10a624 [Accessed 25 Jul. 2019].

[145] Plot.ly. (2019). *DBSCAN Clustering Algorithm*. [online] Available at: https://plot.ly/scikit-learn/plot-dbscan/ [Accessed 17 May 2019].

[146] Scikit-learn.org. (2019). *Demo of DBSCAN clustering algorithm — scikit-learn 0.21.3 documentation*. [online] Available at: https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html [Accessed 13 May 2019].

[147] Oipapio.com. (2019). *python - Feature Importance with XGBClassifier - Oipapio- oipapio.com*. [online] Available at: https://www.oipapio.com/question-3935333 [Accessed 20 Jun. 2019].

[148] Raschka, S. (2019). *Confusion Matrix - mlxtend*. [online] Rasbt.github.io. Available at: http://rasbt.github.io/mlxtend/user_guide/plotting/plot_confusion_matrix/ [Accessed 23 May 2019].