Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Feature Extraction for Motion Data

Ganesamanian Kolappan

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fulfillment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Dr. Anastassia Küstenmacher

January 2021

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Ganesamanian Kolappan

Abstract

Motion data is the readings from the sensors used to detect movement. These sensor readings are time-series data. Time-series data is discrete values ordered sequentially according to the time; one example of this, the stock price of Amazon. Smartphones and smartwatches used everyday consist of sensors such as accelerometers and gyroscopes to detect movement. The time-series data from these sensors collectively called motion data and is used for Human Activity Recognition or Classification (HAR/HAC).

Motion data is challenging to interpret due to its higher dimensions and complexity. Extracting features from the motion data, then utilizing these features for HAR/HAC overcomes the challenges. Current State-Of-The-Art (SOTA) methods defined in the literature either transform the time-series data to images or a different domain. Besides, some methods are neither reusable nor time-efficient as regards processing extensive data.

Motion data can be broken down into many small data series, and the features from those small series can be independently extracted. This research work uses a sliding window for the disintegration process and autoencoders for feature extraction. Additionally, feature-based methods from SOTA are separately implemented to validate the autoencoders' performance. The feature-based methods consist of a Fast Fourier Transform (FFT) and several statistical features. The extracted features from autoencoders and feature-based methods are individually provided as input to the Support Vector Machine (SVM) for HAC. The SVMs' accuracy indicates the significance of the method used to extract features from motion data.

This proposed method has experimented with three annotated HAC datasets, namely Motionsense (not the sensor manufacturer), Wireless Sensor Data Mining (WISDM), and Wireless Sensor Data Mining version 1.1 (WISDM_v_1.1) datasets. The accuracy of the feature-based methods on the Motionsense dataset is 99.38%, the WISDM dataset is 99.65%, and the WISDM_v_1.1 is 85.50%. The autoencoders' accuracy is 98.64%, 99.99%, and 70.46%, respectively. The autoencoders' performance is competitive with feature-based methods. Thus, this research has proven that autoencoders can extract features from motion data without transforming the raw data. In addition to this, the autoencoders have the further advantage of supporting code reusability and processing the extensive datasets.

Acknowledgements

At first, I thank God for keeping me healthy and strong during this pandemic situation.

I thank my first supervisor, Prof. Dr. Paul G. Plöger, for providing the opportunity to work on this Research and Development project. Also, I thank my second supervisor, Dr. Anastassia Küstenmacher, for being the backbone of this project, guiding me throughout the project and motivating me at tough times. I extend my gratitude to Mrs. Iman Awaad for the strong support throughout my journey in this master's course and the constructive criticism on this report.

A special thanks to Mrs. Jill Yates-Wolff, for her valuable inputs on literature writing to script this report.

I appreciate my colleagues, Deepan Chakravarthi Padmanabhan, Santosh Reddy, and Madhumetha Ramesh for their constant support, constructive criticism and motivation.

Finally, I extend my love and dedicate this work to my late father, Mr. Kolappan Perumal and my grandmother Mrs. Kangathanu Perumal.

Contents

List of Abbreviations

AR	Auto Regressive
ARIMA	Auto Regressive Integrated Moving Average
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Wrapping
DWT	Discrete Wavelet Transform
FFT	Fast Fourier Transform
GAN	Generative Adversarial Networks
GPU	Graphical Processing Unit
HAC	Human Activity Classification
HAR	Human Activity Recognition
IMU	Inertial Measurement Unit
IoT	Internet of Things
LSTM	Long Short Term Memory
LSTM-FCN	LSTM-Fully Convolutional Network
MA	Moving Average
MAD	Mean Absolute Deviation
ML	Machine Learning
NLP	Natural Language Processing

PCA	Principal Component Analysis
RAM	Random Access Memory
RBF	Radial Basis Functions
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SOTA	State-Of-The-Art
SVM	Support Vector Machine
SVR	Support Vector Regression
t-SNE	t-Distributed Stochastic Neighbor Embedding
UCI	University of California, Irvine Machine learning repository
WISDM	Wireless Sensor Data Mining

List of Figures

List of Tables

1

Introduction

Smartphone games, such as car racing, have two modes of operation; one using the soft keys, the other orientations of the phone. The smartphone recognizes the orientation from the motion data. Motion data is the collection of readings from multiple sensors such as gyroscope, altimeter, and accelerometer used to recognize the movement [140]. These sensor readings are recorded along with the respective timestamps. Sensor readings gathered according to the time order sequentially is known as time-series data [79]. One example is weather forecasting; the graph consists of temperature values in the increasing order of days, months, or years [46].

The time-series data is challenging to interpret due to their complex patterns and high dimensions [32] as represented in Figure 1.1. To overcome these challenges, the optimal way is to extract the features before further processing, instead of proceeding with the raw data itself [48]. The features are the attributes of the raw data in low dimension space, retaining its central properties, making it easier to interpret and visualize [133]. Some of the handcrafted features for any time-series data are frequency, phase, amplitude, data history, wavelet pattern, mean, standard deviation, correlation, entropy, linearity trend, and the maximum as well as the minimum value of the peak [10]. Hand engineering the features is not effortless due to different patterns, value ranges, and high dimensions. Only certain features can represent raw data effectively. In contrast, other features are inadequate to represent the raw data. The significance of each feature is not obvious before the extraction. Earlier research work [14] shows handcrafting the features requires highly-skilled labor, a detailed description of the application, proper organization of data. This requirement, in turn, increases the cost, time and error for feature extraction.

The features are crucial as they influence the performance of the application. Automatically extracting the features reduces the challenges imposed by handcrafting the features. Deep learning methods can automatically extract features [46]. Deep learning methods are of great success for many problems across different fields and applications

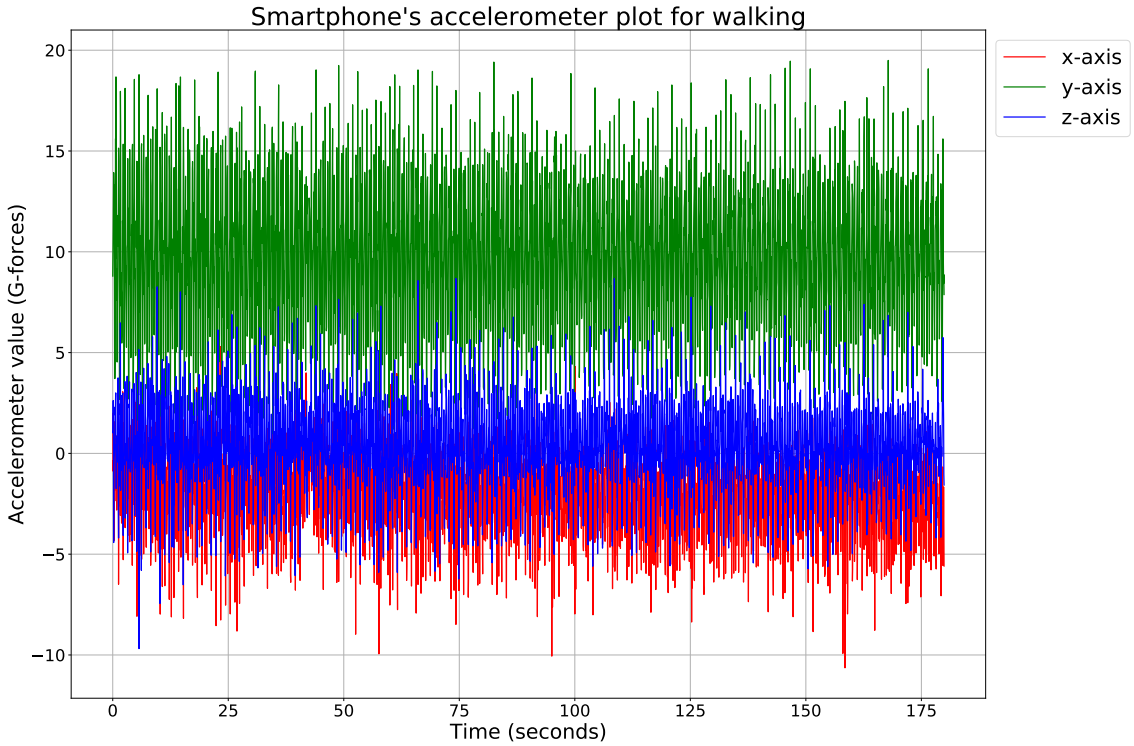

Figure 1.1: *Motion data from smartphone triaxial accelerometer. The sampling frequency of the sensor is 20 Hz. So for every one second, the sensor outputs twenty readings. The plot consists of 3,500 readings for each axis x, y, z of accelerometer spread in 3 minutes for a single activity "walking". Data from [47].*

[52][39][66]. Therefore, the deep learning methods are a better option for hand engineering methods [49]. The efficiency of the deep learning methods increases with the quantity of data. Large quantities of data are not problematic due to the well-equipped network, and wide usage of the Internet of Things (IoT) products [32].

This research work concentrates on automatically extracting features using autoencoders from motion data. The working of the proposed method is demonstrated using the Human Activity Recognition or Classification (HAR/HAC) datasets from the University of California, Irvine Machine learning (UCI) [113] and Kaggle repository [112]. These datasets include readings from the accelerometer and gyroscope in the smartwatch as well as the smartphone.

1.1 Motivation

Motion data is perplexing since it collectively consists of readings from more than one sensor, used to detect the motion of an object or human. Sensors such as accelerometer,

gyroscope, magnetometer, Inertial Measurement Unit (IMU), and pulse oximeter are commonly embedded in a smartphone, and smartwatch [89][140]. A sufficient amount of data to track and predict human activity can be acquired from these electronic gadgets [73]. HAR is one of the dynamic research areas which intend to improvise the well-being of the human [151]. Efficient HAR helps detect the abnormalities beforehand and reduce the chance of being critical due to the invasion of new diseases from time to time [25]. Healthcare research focuses on HAR for symptoms tracking [124][54] and evaluating exercise performance of the patients to recover [141].

Smartwatch manufacturers like Apple, Fitbit, Samsung as well as Fossil compete to provide better HAR using their in-built sensors [33]. These smartwatch manufacturers provide HAR based on the motion data but use different algorithms [33]. Apple provides comparatively better results since there is a shell integration where the smartphone utilizes the smartwatch data to provide the HAR [5].

Apart from HAR, motion data also marked their presence in robotics for adaptive learning (concentrated on failure situation) [91], fault detection and analysis [45], in manufacturing sectors for production line management [31], functional and crash prediction for the automobiles [27].

The majority of the application using motion data is either classification or forecasting task [46]. A classification or forecasting can be estimated with the help of characteristics, also called features from the data. The features can be extracted in two approaches; one is manually using transformation methods like Fourier transform [14]. Another is instance-based, which converts the graph as depicted in Figure 1.1 into several segmented images then process using machine learning methods [48]. These approaches are not sufficient due to the substantial feature extraction time, demand domain knowledge, redundancy in the features, and consume more memory.

Deep learning methods have been a replacement for classical methods in different disciplines [49] [39] [66] [53]. When it is reviewed [132], the deep learning methods are extremely skewed towards Natural Language Processing (NLP) [60][39] and computer vision [129][66]. Incorporating the deep learning method from the NLP domain for feature extraction from motion data is a standalone approach. This approach does not require much domain knowledge, comparatively fast, and consumes less memory. However, it is likely to produce redundant features, but this can be easily traced and avoided [39].

This research work aims to use the potentiality of the deep learning method to extract the features from the motion data for robust HAR. These features are then used for the HAC. A classification task validates the proposed method in order to hypothesize the competence of the features extracted by the deep learning method.

1.2 Problem Statement

Motion data consists of either distinguishable or non-distinguishable patterns (readings) from each dimension. The features extracted from the motion data should represent these patterns uniquely in the low dimension space. Extracting features manually or by instance-based approaches like clustering methods [11][49], and Random Forest (RF) [38], or deep learning method like Convolutional Neural Network (CNN) induces difficulties such as:

1. Handcrafted features are extracted by trial and error method. Since there are no specific rules or conditions, that particular feature can effectively represent the motion data. This process is more time-consuming [79].

2. Domain knowledge is required to handcraft the features, especially while having features from transformation methods like Fourier transform [14].

3. Transformation methods can effectively extract features, yet it requires additional statistical features like mean and variance for the effective representation of raw data [117]. The number of additional statistical features required varies with the complexity of the raw data.

4. Instance-based methods segments the graph presented in Figure 1.1 into several images. Then it extracts different instances from those images and groups them into different classes based on similarity [11]. Thus, the instance-based methods consume an abundance of memory and time for extracting instances as well as processing [38].

5. Redundant features are likely to appear causes overfitting, but the major difficulty is, it is untraceable [79].

6. Time-series data should be converted to an image/matrices format when processing using CNN [44].

Hence, automatically extracting the features reduces the difficulties induced by the handcrafted or instance-based methods. Additionally, automatically extracted features should be competitive to the handcrafted features. The Research Questions (RQ) answered in this research work are as follows:

RQ1 What are the available methods to extract features?

RQ2 What are the minimum number of sensors required for HAC?

RQ3 What are the minimum number of sensor placements required for HAC?

RQ4 How the raw data supports improving the performance?

RQ5 How proposed method is performing on the three datasets, namely Motionsense (not the sensor manufacturer), Wireless Sensor Data Mining (WISDM), and Wireless Sensor Data Mining version 1.1 (WISDM_v_1.1) datasets?

1.3 Objectives of This Research

The primary objectives of this research work are:

1. Extracting the features using the feature-based methods on the HAR dataset.
2. Extracting the features using autoencoders on the same HAR dataset.

Extracting the features alone does not suffice the goal of this project. The features extracted from both methods are used separately for the HAC executed by Support Vector Machines (SVM). SVM performs equivalently to certain deep learning methods [84]. Besides, SVM does not impose hyperparameter tuning compared to deep learning methods [52]. The classification accuracy of the SVM learned from the features extracted by the autoencoder is compared with the classification accuracy of the SVM learned from the features extracted by feature-based methods for validation.

The research hypothesis is that the deep learning method can coherently and automatically extract the features for motion/time-series data. The proposed method is generic and suitable for any task such as forecasting, classification, and anomaly detection. As a first step, this research work is being demonstrated in the classification task since time-series classification is regarded to be one of the challenging tasks in the data mining field [146]. The robustness and performance of any task depend on the uncorrelated features representing the raw data, which is the core of this research work.

1.4 Report Outline

This report is composed of six chapters; Chapter 1 provides the introduction followed by the motivation for this research work, an overview of the research proceedings, and a report outline. Chapter 2 enlightens the necessary knowledge about the concepts used throughout this research work. Chapter 3 provides various State-Of-The-Art (SOTA) methodology used for time-series feature extraction along with its deficits. Chapter 4 states the proposed method for the feature extraction from the motion data. Chapter 5 presents the working and results of the proposed methodology on three different datasets. Chapter 6 includes the summary, future direction, and lessons learned from this research work.

2

Background

This chapter provides a general overview of time-series, their types, and challenges, along with the methods to analyze and pre-process. This research chiefly focuses on extracting features using autoencoders, followed by a classification task to validate the method. Therefore, the machine learning method used in this research work is discussed. This chapter helps to provide the concept useful to understand the later chapters. Readers can skip if the topics are familiar.

2.1 Time-series

Time-series data is the variables recorded in the chronological order of time [79]. A smartwatch measures the number of steps continuously based on its accelerometer reading [140]. The same smartwatch measures the number of steps with the respective time of measuring (timestamp) in a regular interval, known as time-series data. The interval of recording can range from seconds to years depending on the application, yet a lesser interval provides better results [49]. Some examples of time-series data are the annual profit of the company, monthly wages of the employee, daily price of the petrol, hourly temperature reading, mortality count for every minute, and electricity fluctuation for every second. Time-series are used in weather forecasting [81], earthquake prediction [90], statistics [68], econometrics [26], finance [55], and many other applications [152][59][114]. Time-series data can be represented as

$$T_d = \{X_0, X_1, ..., X_{N-1}, X_N\} \tag{2.1}$$

$$T_d = \{X_i\}_{i=0}^{N} \tag{2.2}$$

where, X is the variable recorded, i is the particular instance usually starts from zero since the start time is the ground value, by default has the value zero, N is the total number of observations inversely proportional to the time interval, less interval more observation. In the above example of a smartwatch, number of steps is the variable X.

2.1.1 Types

Based on the number of variables recorded at an instance, time-series are of three types [121][135].

1. **Univariate:** At an instance only one variable is recorded.

$$T_d = \{X_i\}_{i=0}^{N} \tag{2.3}$$

   For example, monthly electricity consumption of Bonn for the year 2020, the unit of electricity consumed by Bonn city is the variable X, N is the total number of observations.

2. **Bivariate:** At an instance, two variables are recorded.

$$T_d = \{X_{1i}, X_{2i}\}_{i=0}^{N} \tag{2.4}$$

   For example, monthly sales-demand of iphone 11 pro for the year 2020, the sales and demand values are the variables X_1 and X_2 respectively, N is the total number of observations same for both the variables.

3. **Multivariate:** At an instance, more than two variables are recorded.

$$T_d = \{X_{1i}, X_{2i}, ., X_{Mi}\}_{i=0}^{N} \tag{2.5}$$

   where, M is the number of variables which is always greater than or equal to 3, N is the total number of observations same for all the variables. For example, a monthly weather forecasting of Bonn for the year 2020. The weather data includes variables like temperature, humidity, wind, dew point, cloud cover as X_1, X_2, X_3, X_4, and X_5, respectively. Chapter 1, Figure 1.1 is a multivariate time-series data.

Note: For simplicity, N is considered the same for all the variables. N can vary concerning the individual variable.

2.1.2 Pattern

Time-series data is the discrete values acquired from sensors along with the timestamps [133]. According to the timestamp, time-series data is plotted by joining the discrete values instead of scatter plot [116]. Figure 2.1 renders the scatter plot for the smartphone's

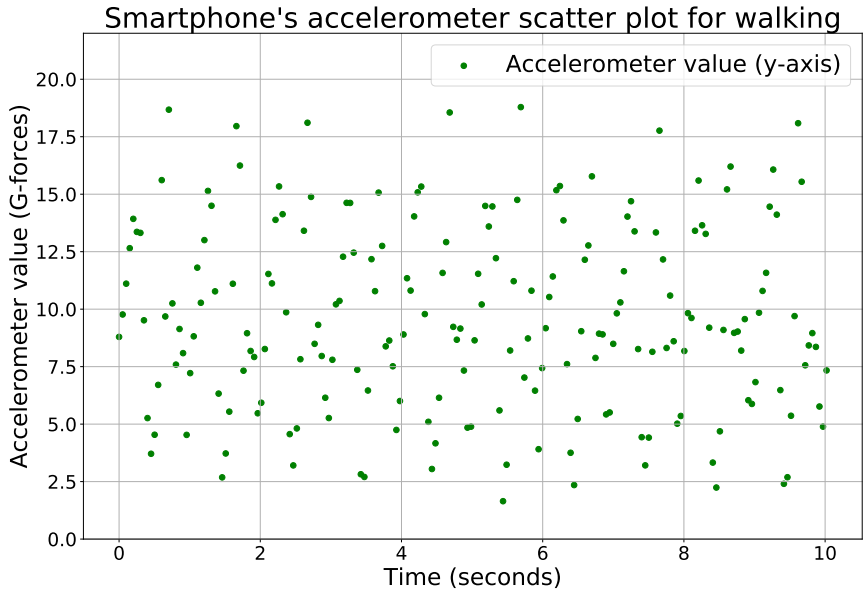

Figure 2.1: *Time-series scatter plot. The plot represents smartphone's accelerometer (y-axis) readings spread in ten seconds for the walking activity. Data from [47].*

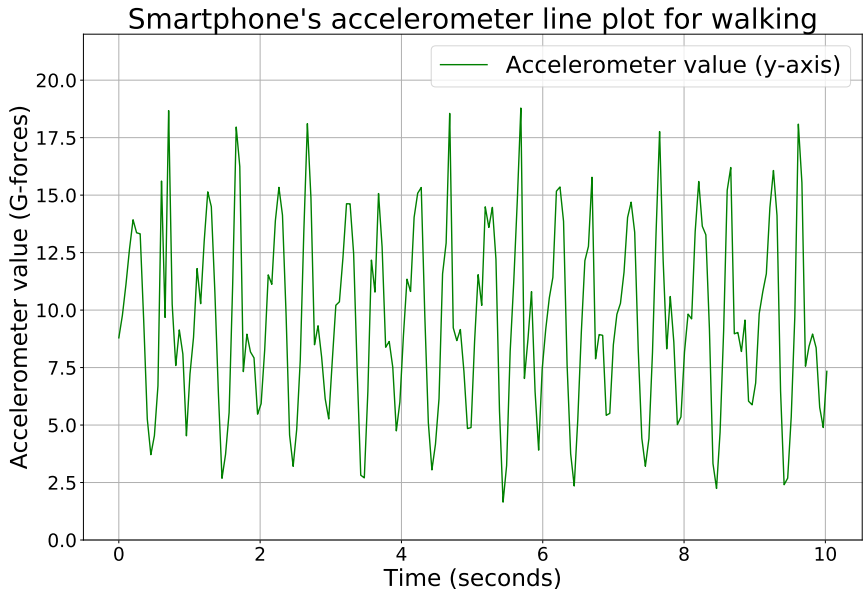

Figure 2.2: *Time-series line plot. The plot represents smartphone's accelerometer (y-axis) readings spread in ten seconds for the walking activity. Data from [47].*

accelerometer (y-axis) readings spread in ten seconds for the walking activity. The scatter plot would be uninterruptible or misleading to predict. The same values are joined in

chronological order, as presented in Figure 2.2. The later plot conveys that the walking activity is cyclic with slight irregularities in between the cycles. This information holds since walking is a repetitive task, and those irregularities may be due to the variations in the ground level. Therefore, the line plot for the time-series data provides better visualization. This visualization helps extract the information from the raw data through the pattern it depicts. Also, to understand the relationships between the variables in multivariate time-series data [116].

Time-series data embeds any one of the following patterns, also called components of the time-series data [133][111] as shown in Figure 2.3.

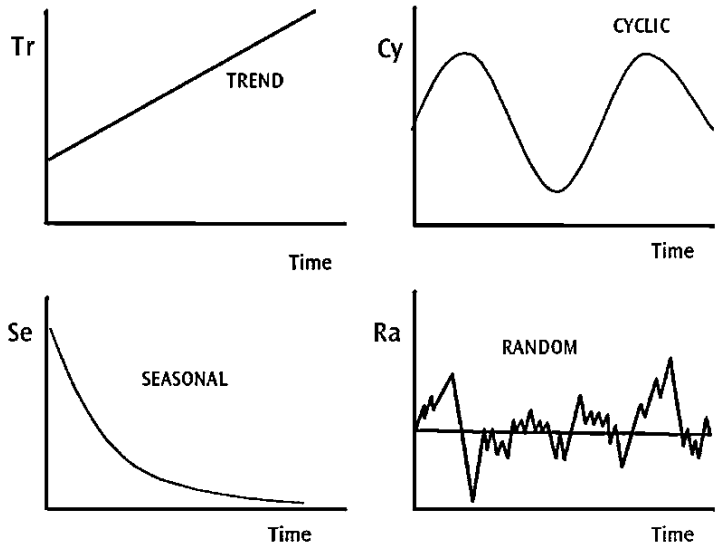

Figure 2.3: *Time-series components. The components are of four types namely trend, cyclic, seasonal, and random (from left to right). Trend depicts certrain pattern throughout the graph, cyclic is pure repetitive like the sine function, seasonal exhibit particular pattern at certain time and random is irregularities that does not fit in earlier components. Reproduced from [118].*

1. **Trend:** The graph shows a clear trend of the data such as linear, non-linear, increasing as well as decreasing over the long period. The trend is also called the "change of direction". The bottom left graph in Figure 2.4 exhibits an increasing trend for Australian monthly electricity production for over thirty years. The observation of the trend should be for a long period. Whereas the top right graph in Figure 2.4 implies a decreasing trend, the period is very less to conclude about the trend.

2. **Cyclic:** The graph shows a continuous oscillating pattern, but the oscillation period varies. The top right graph in Figure 2.4 depicts a US treasury bill contract for a hundred consecutive days, is a part of a cycle. If the graph has been provided for more consecutive days, then could have witnessed the complete cycle. Figure 2.2 shows that the walking activity exhibits a cyclic pattern.

3. **Seasonal:** The graph repeats the pattern at every particular interval due to seasonal behavior or location, or human nature. The frequency is predictable and fixed. The seasonal pattern is different from the cyclic pattern, where the frequency is not fixed and has a repetitive pattern at regular intervals. The top left graph in Figure 2.4 illustrates a seasonal pattern for every year where the monthly housing sales fall as the new year approaches. Additionally, the graph depicts a cyclic pattern for every ten years.

4. **Random:** The graph does not depict any pattern. It consists of irregularities and sudden variations. The irregularities could appear in the cyclic (as in Figure 2.2) or seasonal, yet the overall pattern does not appeal irregular. The bottom right graph in Figure 2.4 portrays a random pattern that is very difficult to infer.

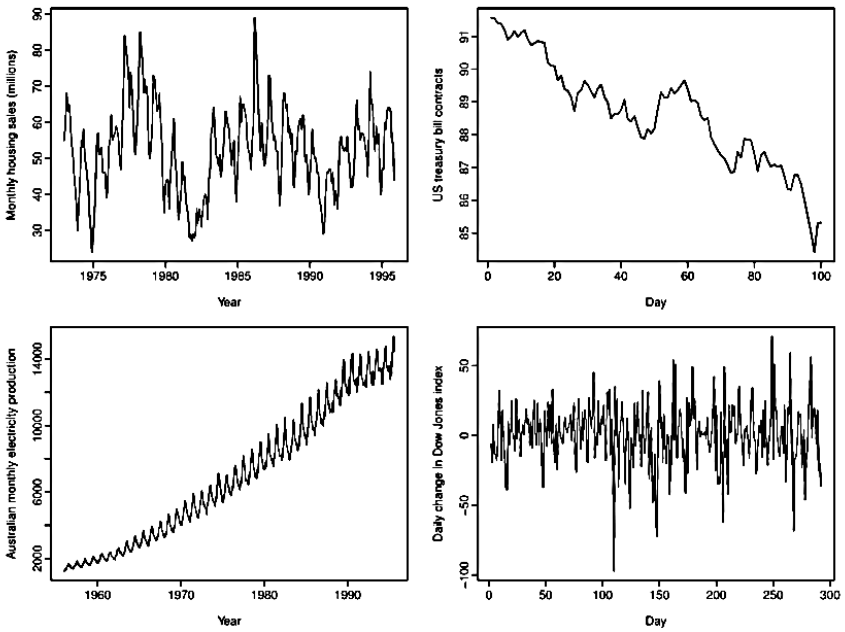

Figure 2.4: *Different time-series patterns. The patterns are namely seasonal, cyclic, trend, and random (from left to right). Reproduced from [131].*

2.1.3 Challenges

There are six challenges in working with time-series data; at least any two could be present irrespective of the application or field.

1. **Stream of data:** Mostly, time-series data is the output of sensors [65]. These outputs are a continuous flow of readings. Depending on the application, the streaming(recording) interval of the reading varies [88]. In Figure 2.5, there are 1,412,864 readings from the z-axis of the accelerometer in the smartwatch spread in hours, which means approximately one reading is recorded for every 0.1 seconds. The stream of data produces two difficulties; one is to stream such a large amount of data the hardware should be robust. Another is storing and organizing those readings, which is the crucial part of the data mining [67]. The difficulties can be addressed by frequent maintenance of the hardware, and the usage of cloud storage or other alternatives [65].

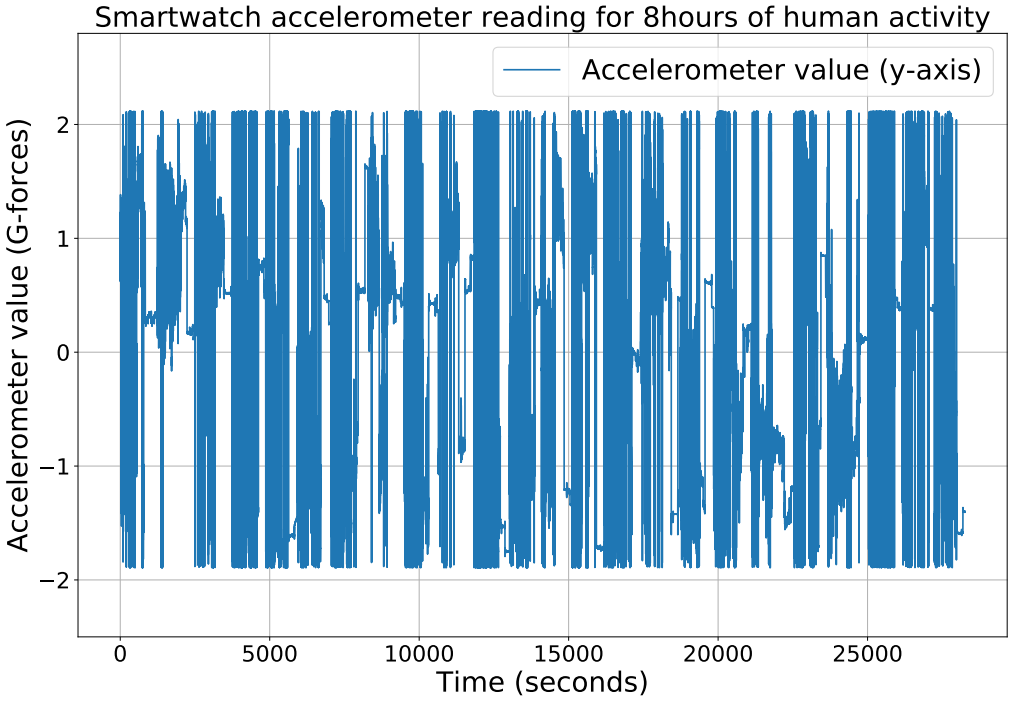

Figure 2.5: *Streaming of time-series data. The plot represents z-axis readings of accelerometer from the smartwatch. The sampling frequency of the sensor is 50 Hz, for every one second the sensor outputs fifty readings. The plot consists of 1,412,864 readings from z axis of accelerometer spread in 8 hours. Data from [94].*

2. **High dimension:** This challenge is mentioned earlier in the Chapter 1 but was not elaborated. The complication occurs when the time-series data is multivariate [37]. This complication increases when there is a concatenation of two or more multivariate data. For example, IMU consists of multiple triaxial sensors such as accelerometer, gyroscope, and magnetometer on board, resulting in nine dimensions. Therefore, in the Equation 2.5 substituting M=3 for IMU yields,

$$T_{IMU} = \{X_{1i}, X_{2i}, X_{3i}\}_{i=0}^{N} \tag{2.6}$$

$$\{X_{1i}\}_{i=1}^{N} = \{X_{1xi}, X_{1yi}, X_{1zi}\}_{i=0}^{N} \tag{2.7}$$

where, X_{1i}, X_{2i}, X_{3i} are triaxial accelerometer, gyroscope, magnetometer respectively, X_{1xi}, X_{1yi}, X_{1zi} denotes the x, y, z axis of accelerometer, N represents number of observations. The complete representation is

$$T_{IMU} = \{X_{1xi}, X_{1yi}, X_{1zi}, X_{2xi}, X_{2yi}, X_{2zi}, X_{3xi}, X_{3yi}, X_{3zi}\}_{i=0}^{N} \tag{2.8}$$

There can be a mixture of multivariate and univariate data. The dimensions can be calculated by the sum of the number of axis corresponding to each sensor [109].

$$D = \sum_{i=1}^{M} A_{X_i} \tag{2.9}$$

where, M is the number of a variable/sensor, A_{X_i} is the number of axis corresponding to the particular variable/sensor. The popularly used method to reduce the time-series data dimensions is Principal Component Analysis (PCA) [51]. PCA projects the raw data to a low dimensional space retaining its central properties [141].

3. **Missing data:** This is a rarely faced critical challenge were some parts of the total readings are missed [86] as represented in Figure 2.6. Missing certain readings are due to power failure, system malfunction, human error, and power surge/sink. In Figure 2.6, the red line plot depicts the raw stream temperature missing values at certain timestamps. These missing values are determined by the Support Vector Regression (SVR) model [50] as indicated by the blue line plot. There are methods to calculate the missing values such as Mahalanobis distance [125], vector auto-regressive model [13], and few others [128][2]. These methods either predict the missing values or transform the data into a different space. However, there is

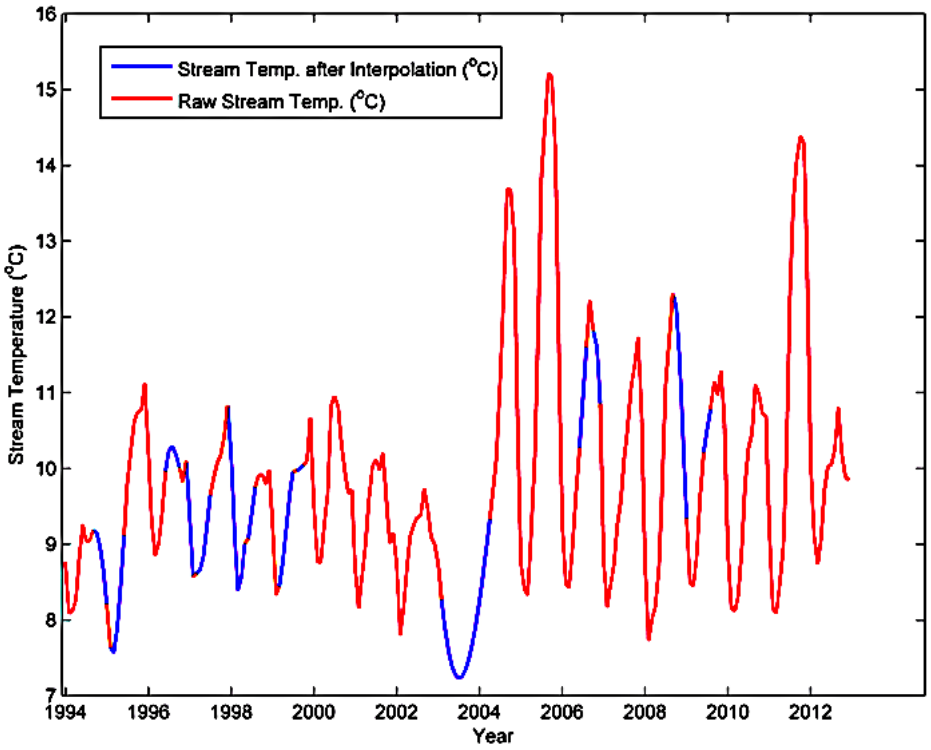

Figure 2.6: *Time-series with missing data. Red line plot is the raw data with missing values, blue line plot is the calculated values inspite of missing values. Reproduced from [50, p. 24].*

degradation, or sometimes a failure in the performance of the application since the calculated values are not accurate [86].

4. **Interpretation:** Time-series data is hard to interpret [37] as presented in Chapter 1, Figure 1.1 and Figure 2.5. If the time-series data have a random pattern with irregularities in a multivariate type, it is very tedious to extract any information. This challenge is addressed by various approaches such as changing to the frequency domain [14], segmented visualization [49] where only a particular part is interpreted rather than the whole data, downsampling using a moving average or sliding a window [65]. The multivariate time-series with the random patterns are often encountered in signal processing applications [150]. The multivariate time-series data is transformed mostly to the frequency domain to extract information [14].

5. **Varied length of data:** The number of discrete values for a particular variable can be more than the other variables [85]. For example, in Figure 2.7 the number

of discrete values for the Ireland population is 50% higher than the number of discrete values for the European population. The varied length is different from the missing data challenge since the timestamps for both variables are the same, but with different scales. Europe's population is measured on a hundred million scale due to which it failed to achieve in certain years. In contrast, Ireland's population is measured on a scale of millions. In some applications, even the timestamp length of the variables vary as depicted in Figure 2.8, the time to climb upstairs is longer than the time to climb downstairs. This challenge is not counteracted by padding since it will destroy the existing pattern; instead, the data is processed without any manipulation [144]. The reason is that the varied length aid as an additional feature [144].

$$T_{upstairs} = \{X_i\}_{i=0}^{N1} \tag{2.10}$$

$$T_{downstairs} = \{X_i\}_{i=0}^{N2} \tag{2.11}$$

where, N_1 and N_2 are length of observations for activity upstairs and downstairs respectively. The N_1 and N_2 are not padded to have a common length N.

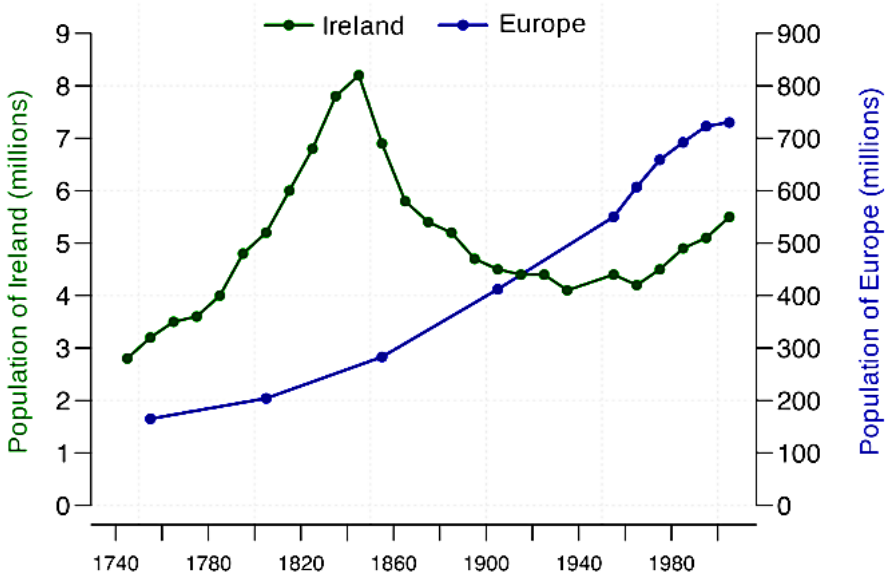

Figure 2.7: *Time-series with varied length of data. The scatter points of Europe's population in blue line plot is lesser than the Ireland's population in green line plot. Reproduced from [103].*

6. **Change point:** The time-series data consists of numerous change points [6]. The change point is the transition from one value range to another, which can be positive or negative [22]. In Figure 2.8 the initial accelerometer readings for downstairs were in the range [0.6,1.4]. As it approaches near 50 seconds, it suddenly falls to the value range [0.0,-1.8], which is the transition denoting the first change point at (40, 0.0). Change points occur due to various reasons like noise, system fault, environment change, and human behavior. Change points can be deceptive sometimes. To illustrate, in Figure 2.8, the upstairs has a spike after 200^{th} second, which is not a change point since the value range is not settled in a period [22]. Additionally, the value retains the previous value range immediately after the spike. The sudden spike might be due to some disruption while climbing upstairs. Change points are an important and influential parameter for the time-series data since it indicates the transition from one activity to the other [6]. There are a lot of change point detection methods [29][6][74] used to extract pattern or information from the time-series data. If the time-series data consists of change points more than 50% of total data, then that time-series data exhibits a random pattern [29].

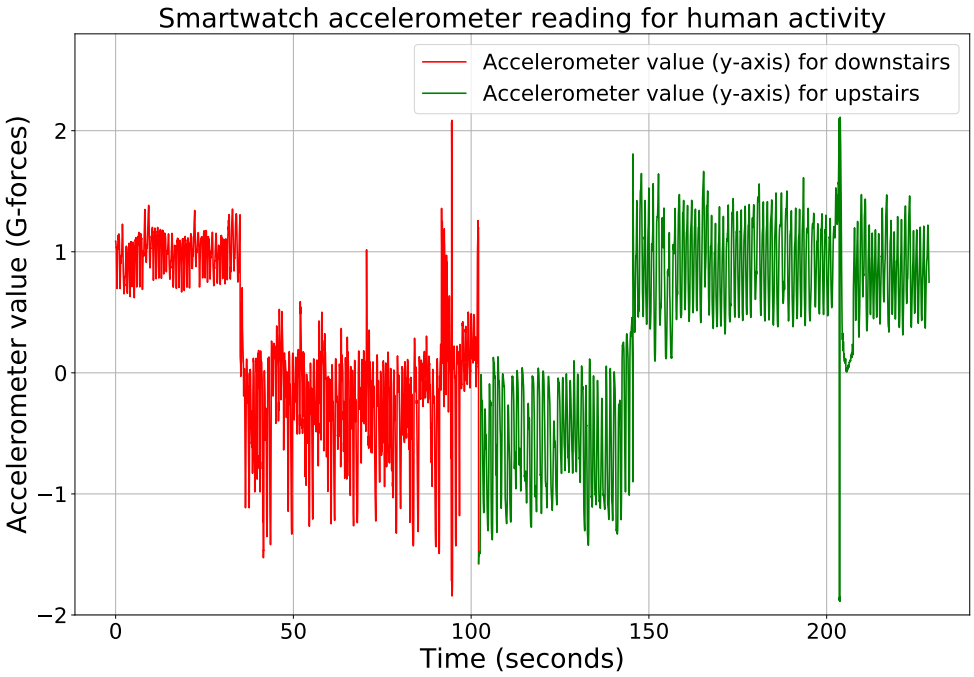

Figure 2.8: *Time-series data with change point. The graph consists of two activities upstairs and downstairs from a six continous activities namely walking, jogging, standing, sitting, upstairs and downstairs. Data from [94].*

2.1.4 Stationarity

Apart from the above challenges, there are situations where time-series data processing can not proceed further this situation is called stationarity of time-series [76]. Stationarity of time-series can be identified by three characteristics as follows: [76]

1. Constant mean.
2. Constant variance.
3. Constant correlation or uncorrelated between the variables (multivariate).

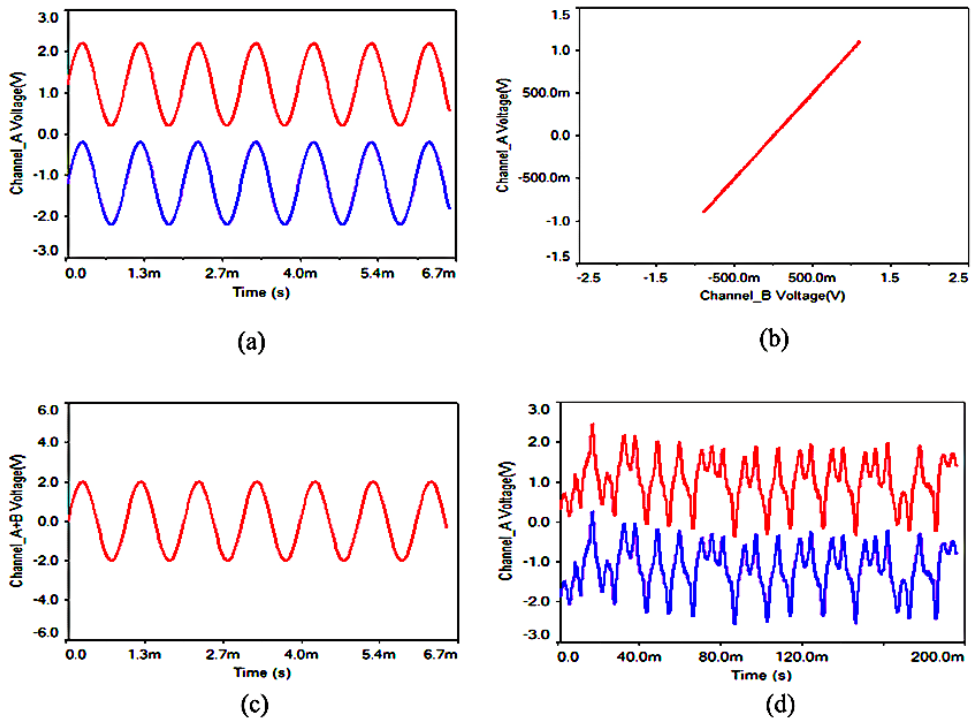

Figure 2.9: *Time-series stationarity. Plot (a) & (c) denotes a sine function, (b) denotes a linear function, and (d) denotes a cyclic function with constant mean. Reproduced from [143, p. 8].*

Figure 2.9 depicts different examples for stationarity. The stationarity is likely to appear if the time-series data can be represented as an existing function [95]. In Figure 2.9 (a) and (c) represents the sine function, (b) is a constant value linear function. These types of functions have solutions for any real value, so it is not feasible to process further. In

this situation, the time-series data is processed by taking the average of the whole series. This is also called the white noise method [95]. White noise method can be formulated as

$$E[T_d] = \frac{\sum_{i=0}^{N} X_i}{N} \tag{2.12}$$

where, $E[T_d]$ is the mean or expected value of time-series. The white noise is computed with the mean $E[T_d] = 0$, constant variance $V[T_d] = \sigma^2 = constant$ and no correlation in case of multivariate $E[X_1, X_2] = 0$. Extraction of information is not achievable from the sationarity condition of time-series, other than the average value [95].

2.2 Motion Data

Motion data consists of several time-series data from the sensors used to detect the movement [94]. The sensors are namely gyroscope, accelerometer, altimeter as well as a magnetometer. Sometimes these sensors are integrated into IMU.

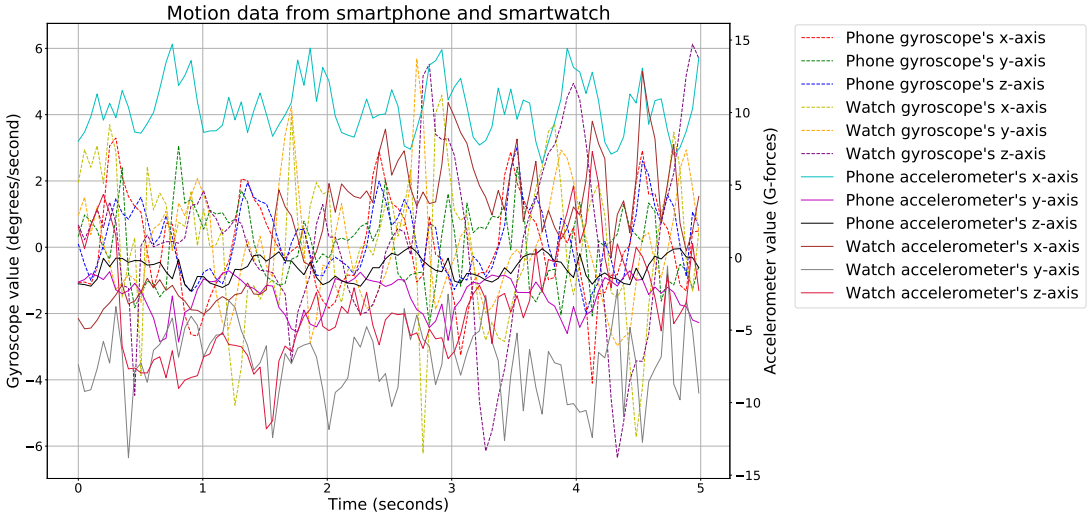

Figure 2.10: *Motion data from smartphone and smartwatch. Spread in 5 seconds for walking activity. Data from [47].*

Motion data suffers from all the challenges faced by the time-series data, especially the dimension and the interpretation challenges as represented in Figure 2.10. The motion data is a collection of multivariate time-series data since either univariate or single multivariate can not effectively detect the movement.

According to Equation 2.9 for a 4 triaxial sensor the dimension can be calculated as

$$D = \sum_{i=1}^{M} A_{X_i} = 3 + 3 + 3 + 3 = 12 \tag{2.13}$$

Thus motion data for $D = 12$ looks like below,

$$T_{md} = \{X_{a1xi}, X_{a1yi}, X_{a1zi}, X_{a2xi}, X_{a2yi}, X_{a2zi}, X_{g1xi}, X_{g1yi}, X_{g1zi}, X_{g2xi}, X_{g2yi}, X_{g2zi}\}_{i=0}^{N} \tag{2.14}$$

where, X_{a1}, X_{a2}, X_{g1}, X_{g2} represents the accelerometers and gyroscopes as sequentially. Motion data exhibits a cyclic pattern because the motion is a repetitive action as depicted in Figure 2.2. Stationarity check is not necessary for the motion data because the motion data mean is not constant. Additionally, the sensor system does not follow any function to record the values.

2.3 Sliding Window

The sliding window is a popular method used in signal processing, image processing as well as network protocols [12]. The sliding window consists of two parameters: one is window size, other is step size [12]. Window size (k) determines the length of the window, and step size (t) determines the number of steps the window should slide forward. Figure 2.11 presents a single 1-d array with the sliding window highlighted by the red box of size three ($k = 3$) and the step size one ($t = 1$).

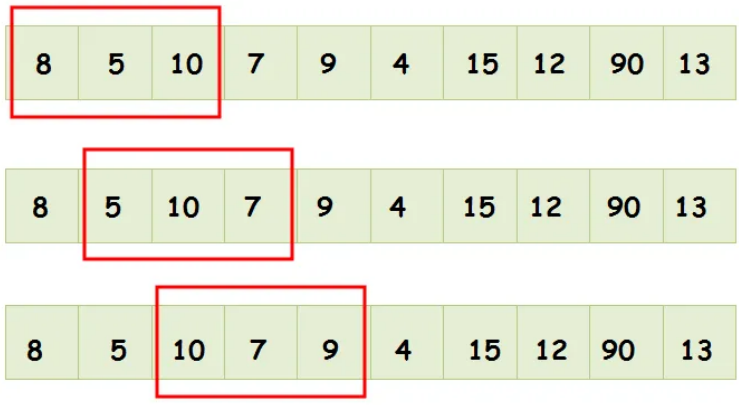

Figure 2.11: *Example for sliding window. The red box is the sliding window that move across the one dimensional array with the time step one. Reproduced from [80].*

At the first instance, the sliding window consists of three values: 8, 5, and 10. Then it slides a single value in the array forward, and at the second instance, it holds the next three values, 5, 10, and 7, and so on until the array ends. The sliding window is utilized differently based on the application. In the network protocol, the values captured by the sliding window are returned simultaneously for parallel processing. In image processing, the sliding window takes up the values and returns the sum or average. For example, the first instance of the sliding window takes the values 8, 5, and 10, then returns either 23 (sum) or 7.6 (average) instead of those three values. The sliding window is used to reduce the data complications as it provides several series of segmented data to process efficiently [24]. Additionally, the sliding window supports debugging the error in a short time [24].

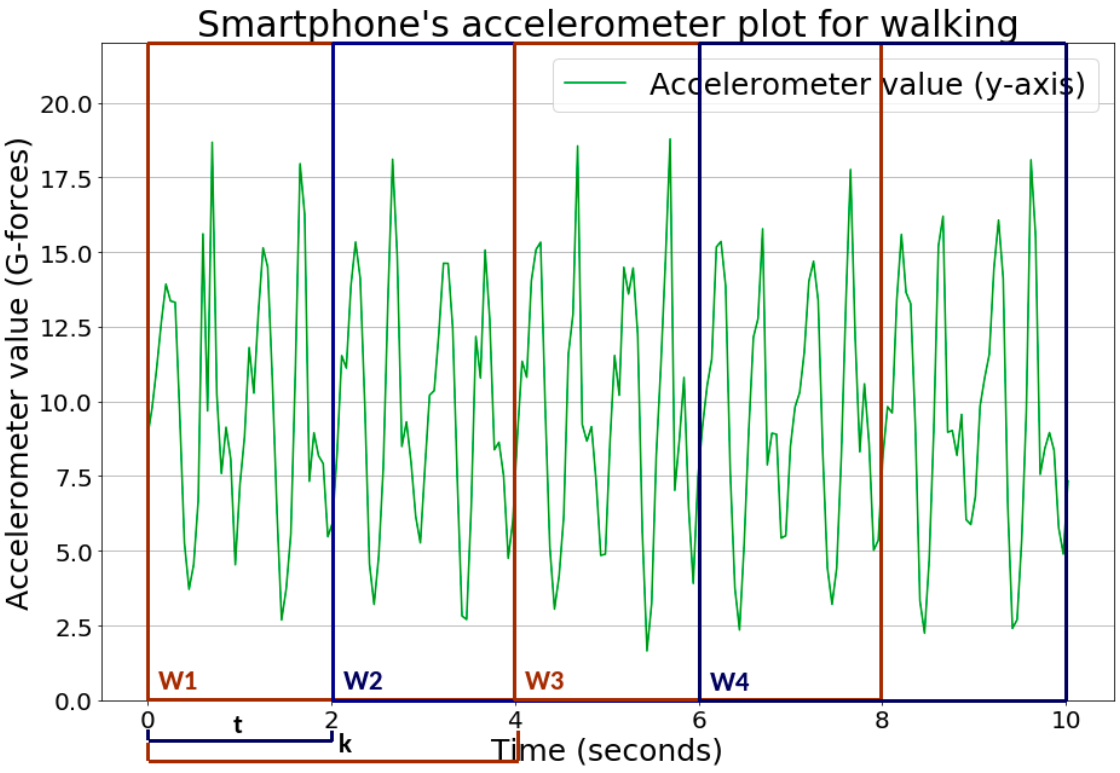

Figure 2.12: *Sliding window for time-series. The sliding window at consecutive positions are differentiated with alternate colors. Data from [47].*

Sliding window in time-series is used similar to network protocol [83]. The sliding window for a univariate time-series is explained below for simplicity, yet it applies the same for the multivariate. The sliding window with the same window size and step size

is called a non-overlapping window ($t = k$). The sliding window with a step size lesser than the window size is called an overlapping window ($t < k$). The sliding window with the step size greater than the window size is not advisable as it will ignore certain values [83].

In Figure 2.2, a sliding window with window size four seconds and step size two seconds has been applied as shown in Figure 2.12. These segmented data is retained instead of a single value; therefore, it increases the dimension by one.

In Equation 2.3 substituting N=200 since the accelerometers' sampling frequency is 20 Hz, for 10 seconds it generates 200 discrete values as shown in Figure 2.12. Before applying the sliding window, the shape of the data is (200, 1), where 200 discrete values for one variable (univariate).

$$T_d = \{X_i\}_{i=0}^{200} \tag{2.15}$$

k=4 seconds (80 values) and t=2 seconds (40 values) while processing, the sizes are taken in terms of values instead of seconds since the data is discrete values stored in an array.

$$T_d = \{\{X_i\}_{i}^{i+k}\}_{i=0}^{N-k} \tag{2.16}$$

where, $i = \{0, t, 2 * t, ...\}$, $\forall i \leq N - k$.

Therefore, substituting $i = \{0, 40, 80, 120\}$ in Equation 2.16,

$$T_d = \{\{X_0, ..., X_{79}\}, \{X_{40}, ..., X_{119}\}, \{X_{80}, ..., X_{159}\}, \{X_{120}, ..., X_{199}\}\} \tag{2.17}$$

The shape of the data after the sliding window is being applied is (4, 80, 1) where, 4 is the number of samples after the sliding window. Each samples consists of 80 discrete values recorded for one variable. This is an overlapping window therefore the total number of discrete values ($4 * 80 * 1 = 360; 360 > 200$) are increased compared to the raw data due to few repetitions in the values. The total number of increased values from the raw data can be calculated by $N + (N - k)$.

2.4 Feature Extraction

Tasks such as classification, prediction, and forecasting give better results based on the number of uncorrelated attributes of the raw data [31]. These tasks do not perform directly on the raw data but instead perform on the extracted attributes from those raw data [14]. These raw data attributes are called features. Features should represent the central properties of the raw data uniquely in low dimension [133]. Figure 2.13 depicts

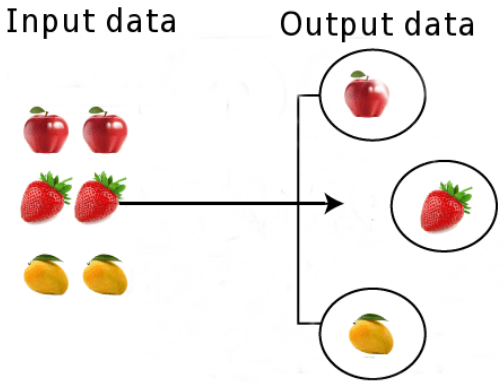

Figure 2.13: *Classification based on features. The possible features to yield this classification can be color, geometric shape, size, weight, texture. Inspired from [134].*

the classification task among apple, strawberry, and mango. Humans can classify these fruits with ease, but scientifically, the brain has learned the attributes for each fruit to classify. For example, color is an attribute; with color, apple, and mango can be differentiated as two separate fruits. Unfortunately, the color attribute can confuse while classifying apple and strawberry since both share the same color. A geometric shape is another attribute that can classify the apple from strawberry since the apple is round and the strawberry is a cone. Color and geometric shape can comfortably classify these three fruits. Additionally, attributes like size, weight as well as the texture can efficiently classify since the color and shape may vary due to different reasons. A good amount of independent features for the raw data reduces the dimension and assist in better results for any task [48]. Raw data presented in the form of images or signals serves no purpose after extracting the features [79]. The feature x can be denoted as $f : X_i \mapsto x$, a function maps the raw data to certain attribute in low dimension. Applying the former notation to the univariate time-series data in Equation 2.2 generates,

$$x = f(\{X_i\}_{i=0}^{N}) \tag{2.18}$$

The feature vector for multiple features for the same data yields of shape $1XK$ that can be represented as

$$\vec{x} = (f_1(\{X_i\}_{i=0}^{N}), f_2(\{X_i\}_{i=0}^{N}), .., f_K(\{X_i\}_{i=0}^{N})) \tag{2.19}$$

where, K is the number of features. A feature vector for multivariate data from the Equation 2.8 yields of shape $1XD$ can be represented as

$$\vec{x} = (f_1(\{X_{1xi}, X_{1yi}, X_{1zi}\}_{i=0}^{N}), f_1(\{X_{2xi}, X_{2yi}, X_{2zi}\}_{i=0}^{N}), f_1(\{X_{3xi}, X_{3yi}, X_{3zi}\}_{i=0}^{N})) \quad (2.20)$$

The multiple features for multivariate data is the combination of Equation 2.19 and 2.20 yielding the feature vector of shape $1XT$ where, $T = K * D$.

$$\begin{aligned}\vec{x} = (&f_1(\{X_{1xi}, X_{1yi}, X_{1zi}\}_{i=0}^{N}), f_1(\{X_{2xi}, X_{2yi}, X_{2zi}\}_{i=0}^{N}),\\ &f_1(\{X_{3xi}, X_{3yi}, X_{3zi}\}_{i=0}^{N}), f_2(\{X_{1xi}, X_{1yi}, X_{1zi}\}_{i=0}^{N}),\\ &f_2(\{X_{2xi}, X_{2yi}, X_{2zi}\}_{i=0}^{N}), f_2(\{X_{3xi}, X_{3yi}, X_{3zi}\}_{i=0}^{N}), ...,\\ &f_K(\{X_{1xi}, X_{1yi}, X_{1zi}\}_{i=0}^{N}), f_K(\{X_{2xi}, X_{2yi}, X_{2zi}\}_{i=0}^{N}),\\ &f_K(\{X_{3xi}, X_{3yi}, X_{3zi}\}_{i=0}^{N})) \quad (2.21)\end{aligned}$$

2.5 Machine Learning (ML) Models

The following ML models are used to perform feature extraction and classification in this research work:

2.5.1 Long Short Term Memory (LSTM) Networks

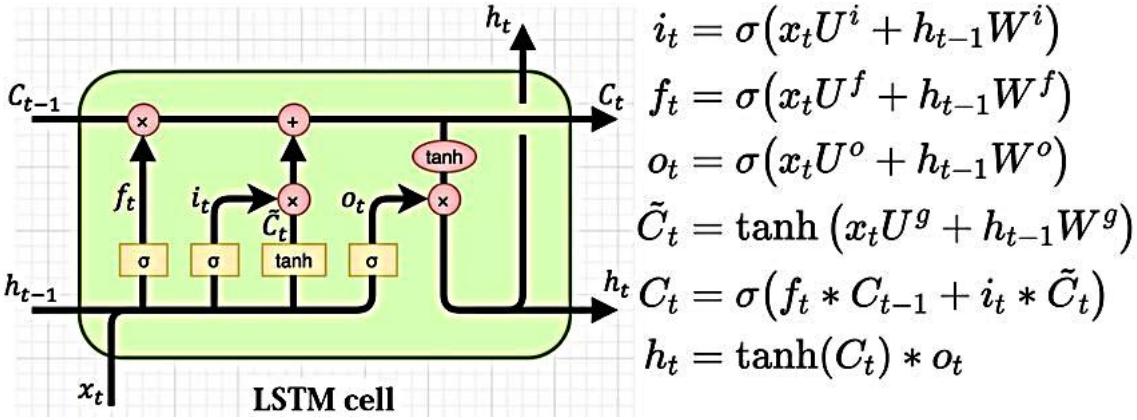

Figure 2.14: *LSTM cell. Reproduced from [136, p. 4].*

LSTM networks are derived from the Recurrent Neural Network (RNN) [52]. The LSTM uses the input data along with its temporal information [136]. Figure 2.14 shows

a single LSTM memory cell. The information inside this memory cell can be stored for a particular time; this ability gives this network its name [136]. Three gates control the flow of information, namely, the forget gate, input gate, and output gate [83]. Three gates get the input x_t with the previous hidden state h_{t-1} at the same instance. The working is explained [136] eliminating the weights in consideration for simplicity as follows:

1. The forgot gate f_t applies the sigmoid function on the received input data $[x_t + h_{t-1}]$ which regularises the input range $[0, 1]$. The "0" indicates the particular data to be removed, and "1" indicates to retain the particular data.

2. The input gate i_t applies the sigmoid function on the received input data $[x_t + h_{t-1}]$ which regularises the input range $[0, 1]$. In parallel, intermediate cell state $\vec{C}_t$ applies tanh function on input data $[x_t + h_{t-1}]$ which regularises the input range $[-1, 1]$. The output of the input gate is the product of both regularised outputs from the input gate as well as the forgot gate.

3. The cell states can be calculated from the output of forget gate and input gate. The product of previous cell state C_{t-1} and forgot gate output is added with the input gate output to generate the current cell state C_t

4. The output gate o_t applies the sigmoid function on the received input data$[x_t + h_{t-1}]$ which regularises the input range $[0, 1]$. In parallel, the tanh function is applied to the current cell state, which regularises the cell state range $[-1, 1]$. The current hidden state h_t is given by the product of the regularised current cell state and the result of the output gate.

Due to these abilities, LSTMs are popularly used in NLP [106][120] for the tasks such as speech synthesis, text generation, and automatic caption writing as well as computer vision tasks like handwriting recognition and image classification. Additionally, LSTMs are combined with other ML models such as CNN [21] and SVM [84] for a hybrid structure to improvise the performance.

2.5.2 Support Vector Machines (SVM)

SVM can be used for both classifications as well as regression tasks [20]. This research work uses SVM for classification, so the SVM is explained from a classification perspective. As illustrated in Figure 2.15, SVM fits an optimal decision hyperplane (in this illustration, it is a line) that separates different classes with maximum margin [82]. Margin is the shortest distance between the observation (samples in each class) and the hyperplane.

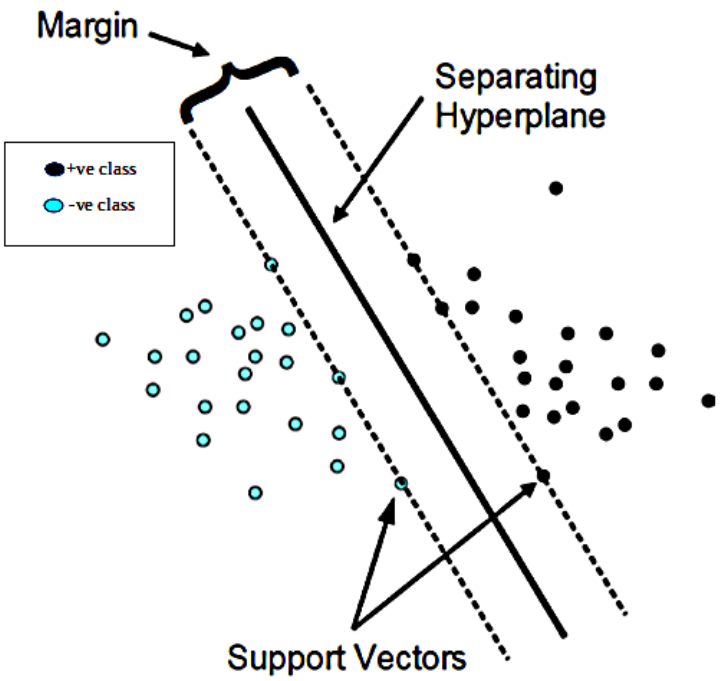

Figure 2.15: *SVM classifier. Reproduced from [36].*

The objective is to maximize the margin so that the new observations can be fitted with more confidence. The observations near the hyperplane on either side are called support vectors. Support vectors influence the margin length, position, and orientation of the hyperplane. The above Figure 2.15 is illustrated for the linearly separable data. However, SVM fits the hyperplane for the linearly non-separable data by manipulating it to linearly separable data by projecting from low dimension to higher dimension [84]. Projecting to a higher dimension is computationally expensive. Hence, the SVM uses a kernel trick that sidesteps the expensive computation. The kernel trick is computing the dot product instead of projecting it to the higher dimension. Linear, non-linear, Radial Basis Functions (RBF), polynomial, and sigmoid are popular kernel functions [20]. This kernel trick is not associated with SVM; it can be used with other machine learning algorithms like logistic regression [20].

3

Related Work

There are methods varying from hand engineering to deep learning [79][133][14][38] for extracting features from the time-series data. These methods can be classified into three categories, namely feature-based methods, instance-based methods, and deep learning methods.

3.1 Feature-based Methods

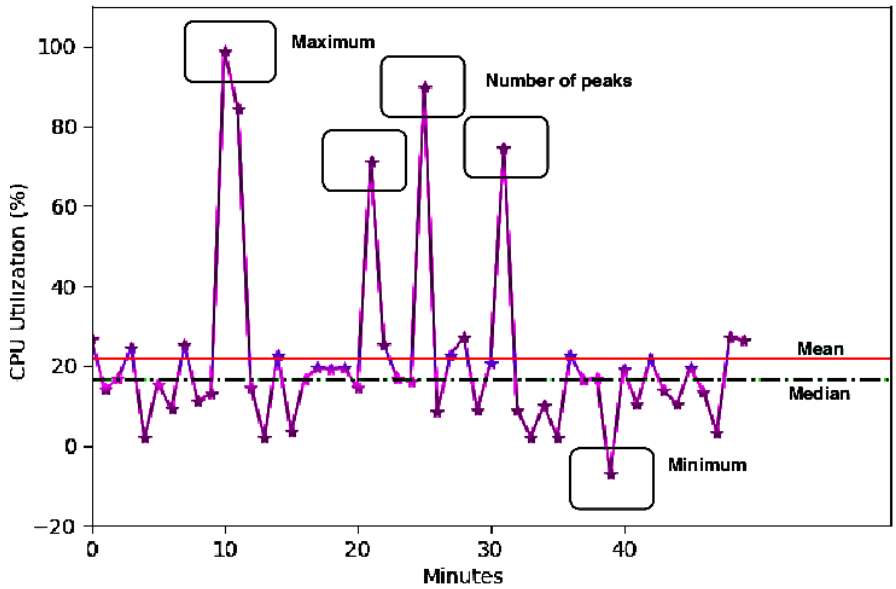

Figure 3.1: *Handcrafted features for motion data. Presented are statistical features that works for any time-series data. Reproduced from [8, p. 1686].*

Feature-based methods are handcrafted, where each feature is penned with domain expertise. The feature-based methods use the properties of the raw data to transform the temporal information into static [32] as illustrated in Figure 3.1.

3.1.1 Transformation Methods

Transformation methods remodel the data from a time domain to frequency domain [108] as depicted in Figure 3.2. The transformation method is also called spectral analysis. In the frequency domain, the complexity imposed on the data is reduced, which aids in interpreting the data [122]. The sum of the functions in the frequency domain generates the time domain values. If the time-series data is disintegrated to the number of sine functions in the frequency domain, then the sum of these sine functions produces that time-series data. Transformation methods include a Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT), and Discrete Wavelet Transform (DWT). The features extracted using the transformation methods are amplitude, phase, and frequency [108].

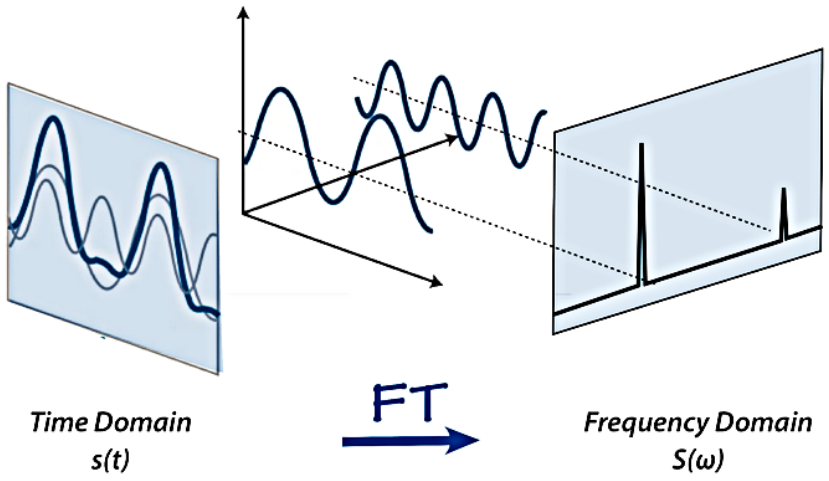

Figure 3.2: *Time domain to frequency domain. Portrays the visualization of the time-series data decomposed into two sine functions in the frequency domain. Reproduced from [119].*

3.1.1.1 Discrete Fourier Transform (DFT)

DFT transforms the time-series data to frequency domain using the sine and cosine functions [77]. DFT assumes the time-series data to be periodic at the point of truncation [14], where the previous sliding window $(k-1)$ end and current sliding window start (k)

as shown in Figure 3.3. DFT can be expressed as

$$X(f) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} T_d e^{-j2\pi f(i)/N} \tag{3.1}$$

$$X(f) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} T_d cos(2\pi f(i)/N) - j \sum_{i=0}^{N} T_d sin(2\pi f(i)/N) \tag{3.2}$$

$$X(f) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} T_d W_N^{f(i)} \tag{3.3}$$

where, $X(f)$ is the DFT coefficient, T_d is the time-series data as in the Equation 2.2, N is the number of observations, f is the frequency, $W_N^{f(i)}$ is called twiddle factor, represented as

$$W_N^{f(i)} = e^{-j2\pi f(i)/N} = cos(2\pi f(i)/N) - jsin(2\pi f(i)/N) \tag{3.4}$$

DFT has the time complexity $O(N^2)$ due to the real and the imaginary part of the twiddle factor in Equation 3.4.

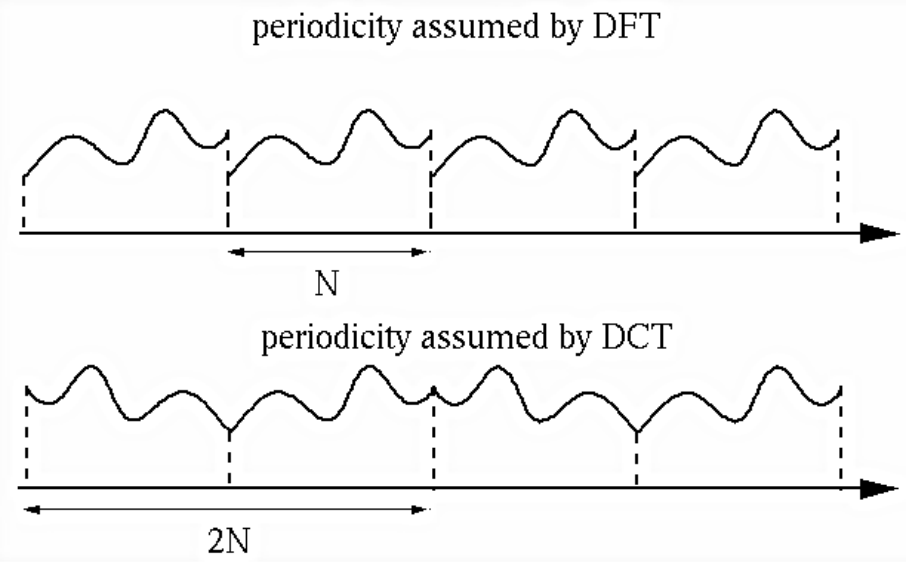

Figure 3.3: *Periodicity of DFT and DCT. Highlighting the frequency leak using DFT compared to DCT. Reproduced from [34].*

3.1.1.2 Discrete Cosine Transform (DCT)

DCT transforms the time-series data to frequency domain using only the cosine function [14]. DCT is popularly used for audio signal processing, image compression as well as video compression. DCT assumes the time-series data to be even symmetric at the point of truncation [14]. The time complexity of the DCT is $O(N)$. DCT can be expressed as

$$X(f) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} T_d cos(2\pi f(i)/N) \tag{3.5}$$

$$X(f) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N} T_d W_N^{f(i)} \tag{3.6}$$

$$W_N^{f(i)} = cos(2\pi f(i)/N) \tag{3.7}$$

DCT is preferred over DFT due to the following properties [14]:

1. DFT accounts for both real and imaginary coefficients whereas DCT examine only real coefficients which is sufficient to process time-series data.
2. DFT can not deal with the frequency leaks whereas DCT can process the frequency leaks as shown in Figure 3.3.
3. DFT performs inadequately when the variables are highly correlated compared to DCT.
4. DCT can perform equivalently to FFT with the time complexity $O(NlogN)$, which is called fast DCT.

3.1.1.3 Fast Fourier Transform (FFT)

FFT transforms the time-series data to the frequency domain, applying both the sine and cosine functions [77]. FFT does not suffer from frequency leakage like DFT because it converts the discrete data to a continuous signal [18]. FFT is the collection of Fourier transform that implements DFT with the time complexity $O(Nlog(N))$ by the use of the Cooley-Tukey algorithm [122]. In FFT, the data is computed by the divide and conquer method, which recursively computes a lot of mini DFT instead of one large matrix multiplications as given in the Equation 3.3 [77]. FFT has been successful in many applications [77] such as filters (low pass or high pass filter) in image processing, image compression and signal processing that includes audio, sonar, and radar. DCT is competitive to FFT, yet FFT results are slightly better than DCT [96].

3.1.1.4 Discrete Wavelet Transform (DWT)

The DCT and DFT assume that the signal is periodic [18]. This can not model the interpretation aptly from the time domain to the frequency domain. To illustrate, if there is a sudden spike for a short span of time, then the spike is difficult to infer in the frequency domain [72]. The Fourier transform is applied to the number of disintegrated signals from the single raw signal to counteract the prior concern. The raw signal is disintegrated using the sliding window, as stated in Section 2.3. Discretizing the raw signal using a sliding window produces two challenges as follows:

1. Narrow window produces good time resolution and poor frequency resolution, which captures high-frequency components.

2. Wide window produces good frequency resolution and poor time resolution, which captures low-frequency components.

The above problem is addressed by wavelet transform that analyzes the signal in diverse frequency with diverse resolutions [72]. DWT transforms the data into a combination of the time-frequency domains [14]. Unlike the above transformation methods, DWT chooses a particular wavelet as the basis function instead of sine and cosine functions. This wavelet is used as a window within which the Fourier transform is calculated [70]. The wavelet can be shrunk or stretched by adjusting the frequency, which addresses both low and high frequencies. DWT can be expressed as

$$X(\zeta, s) = \frac{1}{\sqrt{|s_i|}} \sum_{i=0}^{N} T_d \psi \left(\frac{t - \zeta}{s_i} \right) \tag{3.8}$$

where, s is the wavelet scale, $s_i = \frac{1}{f(i)}$ is used to stretch and shrink the wavelet, $\psi(t)$ is the mother wavelet or referenced wavelet, ζ defines the sliding of the wavelet across the total data. $\psi\left(\frac{t-\zeta}{s_i}\right)$ provides the translation of the wavelet across the whole signal. DWT is widely used in digital communication [3], image processing [4], multiplexing of sensors [123], and data compression [4]. The time complexity of DWT is linear $O(N)$, which is faster compared to the above transformation methods. DWT and FFT differ in performance based on the application [72]. DWT and FFT are highly competitive, so FFT is used where the frequencies are to be analyzed [70]. DWT is used where the frequencies are to be analyzed along with their instance of occurrence. There are different types of wavelet methods [4] such as Haar wavelet, Chriplet wavelet, Morlet wavelet,

Mexican hat wavelet, Gabor wavelet, triangular wavelet and Daubenchies or orthogonal wavelet.

3.1.2 Auto Regressive Moving Average (ARIMA)

ARIMA is a successful method for time-series forecasting [142][107][105]. ARIMA combines Auto Regressive (AR) model and Moving Average (MA) model by an integration factor (I) [1].

3.1.2.1 Auto Regressive (AR) Model

According to the AR model, the current value depends upon the previous values [101]. This can be represented by

$$X_i = f\{X_{i-1}, X_{i-2}, X_{i-3}, .., \epsilon_i\} \tag{3.9}$$

where, ϵ_i is the error term $\epsilon_i = \vec{X_i} - X_i$, $\vec{X_i}$ is the true value. The estimation based on the past values is called lags denoted by $p = 1, 2, ...N - 1$. The AR model is represented in terms of the weighted sum for X_i as

$$AR(p) = \beta_0 + \sum_{j=1}^{p} \beta_j X_{i-j} + \epsilon_i \tag{3.10}$$

$$AR(1) = \beta_0 + \beta_1 X_0 + \epsilon_1 \tag{3.11}$$

where, β_0 is the bias term and β_i is the weight factor. The AR model computes a real-valued integer throughout the series based on the lags with which the future values are forecasted [56]. The AR model fails to consider the cumulative error generated over the series [101].

3.1.2.2 Moving Average (MA) Model

The MA model includes the cumulative error factor. According to the MA model, the current value is influenced by the previous random error like the white noise process [101]. This can be represented by

$$X_i = f\{\epsilon_i, \epsilon_{i-1}, \epsilon_{i-2}, ...\} \tag{3.12}$$

The MA model can be represented similar to the AR model as

$$MA(q) = \beta_0 + \sum_{j=1}^{q} \epsilon_j X_{i-j} + \epsilon_i \tag{3.13}$$

where, the ϵ_i is assumed to be white noise (2.1.4) with zero mean and constant variance, $q = 1, 2, ...N - 1$ is MA coefficient. The MA model fails to consider the previous state value [101].

3.1.2.3 Auto Regressive (AR) and Moving Average (MA) Model

ARIMA integrates the AR model with the MA model by the factor d [1]. The $d = 1, 2, ...N - 1$ is the differencing term that accounts for the number of raw data to be differenced [56].

$$X_i(d = 1) = X_i - X_{i-1} \tag{3.14}$$

$$X_i(d = 2) = (X_i - X_{i-1}) - (X_{i-1} - X_{i-2}) \tag{3.15}$$

Therefore, the ARIMA model is expressed as [56]

$$ARIMA(p, d, q) = AR(p) + d - MA(q) \tag{3.16}$$

The particular instance value is calculated based on the parameter values (p, d, q). The calculated value depends on the previous values and compensated for the error [107]. ARIMA method downsamples the time-series data into the respective static values, which can be used as a feature [105].

3.1.3 Statistical Features

The statistical-based features work well on any data, namely images, signals and time-series [8]. As illustrated in Figure 3.1, the statistical features represent the physical attributes of the time-series data. Statistical features include mean, median, variance, standard deviation, correlation, entropy, energy, homogeneity, Euler number, and minimum as well as maximum peak. The statistical features are combined with the transformation methods to provide improved results [117].

3.2 Instance-based Methods

The time-series data is disintegrated into many instances by a sliding window. The similar instances are grouped into one instance [154], as shown in Figure 3.4. These

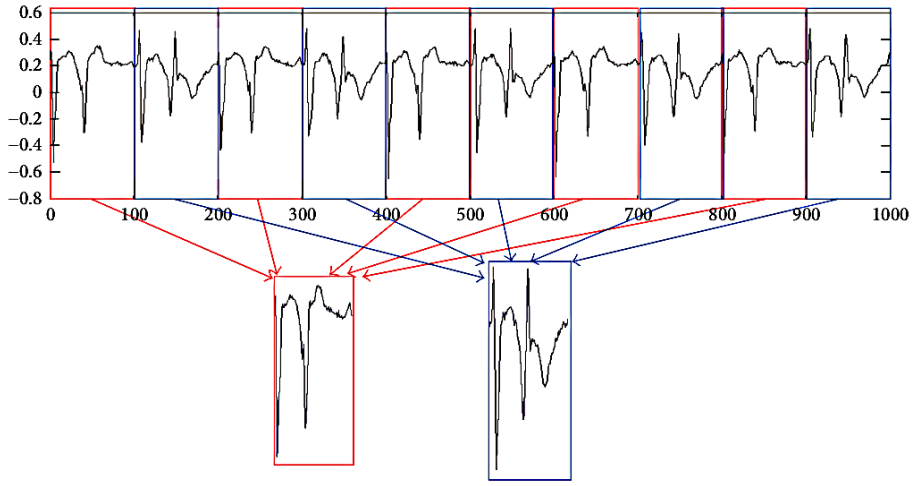

Figure 3.4: *Instances for a univariate time-series data. Depicts the two repetitive instances from the provided time-series. Reproduced from [154, p. 3].*

instances are matched across the new incoming data to predict the next instances or classify based on the matched instances. The instance-based method performs well for the classification task compared to the forecasting [137]. The reason is that the forecasting data tend to have a seasonal pattern, which is difficult to predict using the instances [154]. The instance-based method is widely used by clustering methods like K-means [78] for classification, and RF [23] for regression as well as a classification task.

3.2.1 Clustering Methods

Clustering is a popular unsupervised machine learning method to group the data based on similarity [149]. Time-series clustering works the same way based on the discrete wavelets or instances from the whole data [92]. The sliding window is applied to the time-series data from which particular instances are preferred. These instances should be repetitive among the data or should have a particular trend such as increasing, decreasing, shifting upward as well as downward [81]. Once the instances are collected, these instances are grouped hierarchically that represents different classes. A class can contain only a particular instance or can have multiple instances in some sequences. These classes/clusters are the features collected from the current data. When a new time-series data is available, it is divided into many instances by the sliding window. The distance between each new instance and each feature is calculated using Dynamic Time Wrapping (DTW) [138][147]. As represented in Figure 3.5 (a) the two wavelet appears to

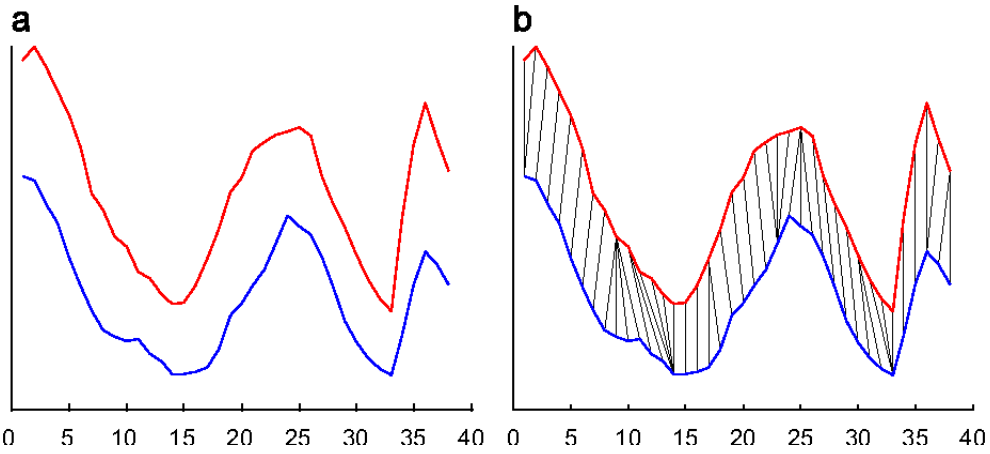

Figure 3.5: *Example for DTW. Portrays the DWT calculating the similarity from the temporal sequence. Reproduced from [64, p. 2].*

have few dissimilarities due to phase shift, yet both are same[64]. The normal distance method or instance matching could indicate that these wavelets are different. However, DWT matches one wavelet's current indexes with another wavelet's previous indexes as illustrated in Figure 3.5 (b) with the thin line connecting both wavelets. DWT calculates the similarity in the temporal sequence with varying shifts or speeds. The similarity value of the DWT supports the grouping of new time-series data, based on the clustered features [147].

3.2.2 Random Forests (RF)

RF algorithm is an ensemble-based bagging method that trains several decision trees on different subsets of the same data (replacements are allowed) [115]. The RF contains a collection of decision trees that extract the features from the data provided [97]. The RF creates many small trees at random to avoid overfitting that extracts sufficient features from the data. These small trees are joined together hierarchically as illustrated in Figure 3.6. RF is successful for regression as well as classification[62]. Since this research work is on classification, the RF used for time-series classification is focused. Initially, the time-series data is segregated as many small instances using a sliding window. These instances (as images) and their respective labels are fed to the RF. The RF uses many decision trees to learn the features for the particular label. Each decision tree learns differently for the same instance. Finally, the class is predicted by the maximum voting from the decision trees, which act as a classification layer. Hence, features extracted by the RF is available at the leaf nodes, which is the last layer before the classification

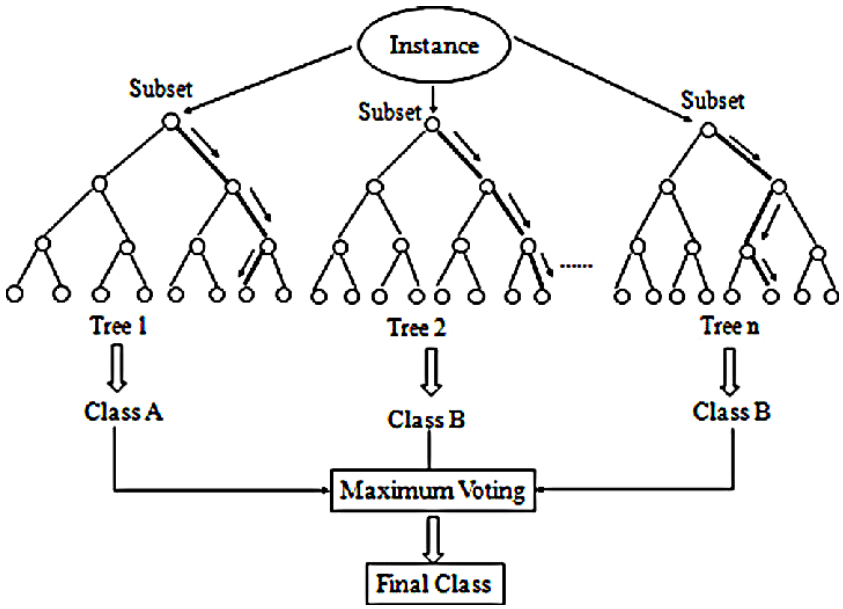

Figure 3.6: *Random forest for time-series data. Working of random forest in summarized in a single image. Reproduced from [100, p. 5].*

layer [115] [15]. Instead of RF, other methods like AdaBoost [43], the neural networks [43] and other machine learning methods [130][110] can also be used. The RF, which is an ensemble bagging method, generates better results compared to other methods [28]. Hence, the RF method is used popularly to process the time-series data.

3.3 Deep Learning Methods

Deep learning methods are widely used in different fields, and applications [52][39][66]. Deep learning methods are neural networks with deep architecture [99]. The deep architecture consists of one input layer to receive the input, one output layer to provide the result, and one or more hidden layers between those two layers [102]. The initial hidden layers are used to extract the features from the data. Whereas the last one or two layers and the output layer are used to learn the features for the prediction. Hence, a single method is used for feature extraction and tasks such as classification or regression. The last layers are designed according to the task requirement. There are different types of deep learning methods [49] such as CNN, Generative Adversarial Networks (GAN), Boltzmann machines, Recurrent Neural Networks (RNN) and Autoencoders.

3.3.1 Convolutional Neural Networks (CNN)

CNNs are proved to be the robust method in the field of computer vision tasks such as image classification, and recognition [61]. CNN is intended to work on the images or matrix of real numbers (digital images) [44]. In CNN, the hidden layers are called a convolutional layer, as shown in Figure 3.7. Each convolutional layer has filters. The filters slide across each pixel and detect the pattern such as geometric shapes, edges, corners as well as area. As it is deep architecture, down the layers, it extracts the features to detect the objects in the image. Subsampling or pooling layer mostly befalls after every convolutional layer. The pooling layer lowers the dimension of the image, determine the filter size and stride pattern. The filter is also a sub-matrix of shape (2k+1)x(2k+1), where, k is the filter or kernel or mask size having values from one. Stride pattern represents particular filter values (values inside the filter sub-matrix) that detect specific patterns such as vertical line, horizontal line, and circle. Alternate stacking of convolution and pooling layer with different stride pattern extracts the possible features from the images effectively [35]. The extracted features are organized in a hierarchy graph internally, which results in a particular geometric shape or object at the leaf nodes of the graph [139]. These leaf nodes are integrated with a fully connected structure to output the prediction [35].

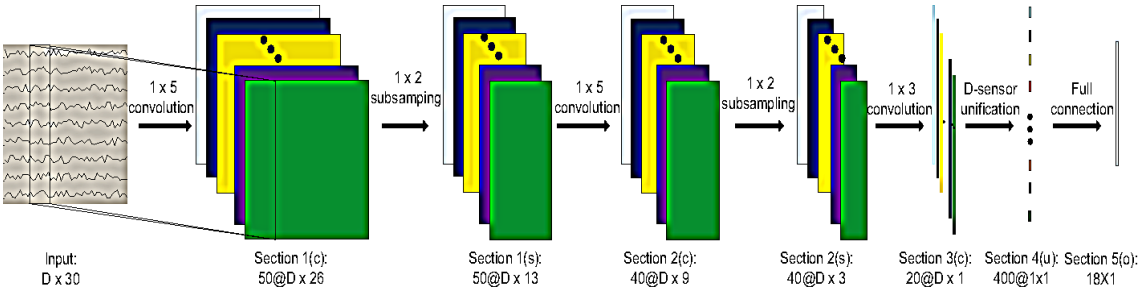

Figure 3.7: *Time-series feature extraction using CNN. Time-series data is converted as image and processed using CNN. Reproduced from [145, p. 3].*

To extract features from time-series data using CNN, the time-series data is converted to images by the following steps [44]:

1. The time-series data may vary in length, as mentioned in Section 2.1.3 but the images should have the same dimension. Hence, the time-series data is padded with zeros to produce all the classes of the same length.

2. The pixel in the image hold values in the range [0,255] (intensity values). The time-series data is normalized to be in the range []0,255].
3. Formulating the matrix structure for each pixel in the image.
4. The image formed is scaled to the RGB intensities to have three layers.
5. Sometimes, along with step 1, the following steps are performed to eliminate step 4 [35]:
   (a) **Smoothing:** Removing high frequency components considering to be noisy.
   (b) **Downsampling:** Applying transformation method to stack the data in the different domain as layers similar to RGB layers.

3.3.2 Transfer Learning

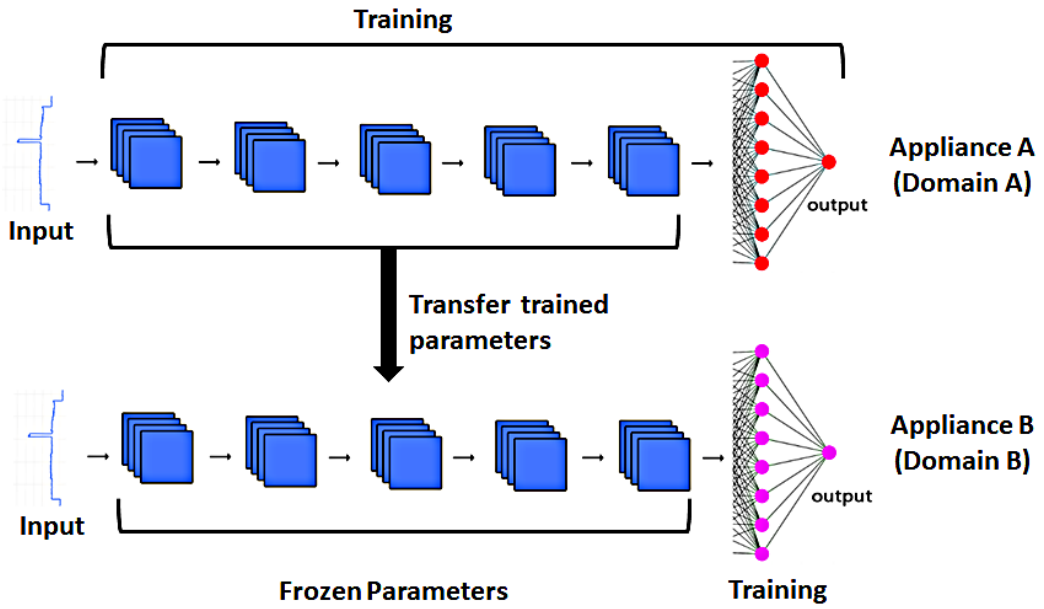

Figure 3.8: *Transfer learning. Depicts the transfer of knowledge of the model from Domain A to Domain B. Reproduced from [41, p. 5].*

Transfer learning is a technique where the trained model is used for the new task [98]. In Figure 3.8, a particular model is being trained on a domain A dataset (source set). The model learned the weights for the source set. This trained model can be used for the other domain B dataset (target set). The first few layers and weights are frozen

(unchanged). The weights of the last layers are newly learned based on the new dataset called fine-tuning. The transfer learning enables the reusability of the model, along with its parameters. The popularly used models in transfer learning are CNNs [63], GANs [16], RNNs [16], and Autoencoders [132].

3.4 Summary

The feature extraction methods for time-series data can be classified into three types: the feature-based method, the instance-based method, and the deep learning method. The feature-based method is a handcrafted method that requires more manual work. The feature-based method includes transformation methods like FFT, DCT, DWT and ARIMA. It concentrates on the physical attributes of the data and utilizes the existing functions. The instance-based method works based on the physical structure of the data. Comparatively does not demand much manual work. The instance-based method includes K-means and random forests. K-means is a clustering method that uses DTW to cluster similar data. Random forest is an ensemble method that follows the bagging approach on the wavelets of data. The random forest can be replaced by other methods such as Adaboost and neural network. Deep learning methods automatically extract features with minimal effort. Deep learning methods include CNNs and transfer learning. CNN works great on the image data; therefore, the time-series data is converted to images for CNN to process. Transfer learning is a concept of using the knowledge of the model on the source data to target data with a fine-tuning based on the target data. Models like CNN and GAN can be used in transfer learning. The feature-based and instance-based methods are hand engineering methods that require domain knowledge, consume more time as well as space compare to the deep learning methods. Additionally, the hand engineered features tend to be redundant and cause overfitting, difficult to trace. Table 3.1 provides the limitations of each method.

Type	**Method**	**Limitations**
Feature-based method	DFT	1. Assumes the time-series to be periodic. 2. Suffers from frequency leaks. 3. The complexity increases with the dimension of the data.
Feature-based method	DCT FFT ARIMA	1. The time-complexity increases exponentially with the dimension of the data. 2. Should have additional statistical features for better result
Feature-based method	DWT	1. Choosing the particular wavelets and time period is tedious in higher dimension. 2. Redundancy occurs which is difficult to trace.
Feature-based method	Statistical features	1. Among the statistical features best ones are picked by trial and error method. 2. Computation time increases as the dimension and number of samples of the data increases.
Instance-based method	Clustering method	1. Complexity and storage area increases with the dimension of the data. 2. Number of clusters is difficult to decide.
Instance-based method	Random forests	1. Consumes more time for single multivariate time-series. 2. Number of decision tree is chosen by trial and error.
Deep learning method	CNNs	1. Does not process varied length since continuous data can not be padded. 2. Should convert time-series data to images.
Deep learning method	Transfer learning	1. Most of the transfer learning methods are image based. 2. The source dataset should be larger than the target dataset.

Table 3.1: *Limitations of related work*

4

Methodology

This chapter focus on the methodology implemented to extract the features from the motion data automatically. This methodology is generic that can be applied to any motion or time-series data. This research also uses feature-based methods, which serve as a benchmark to validate the proposed method. Hence, the features extracted by hand engineering are discussed before the methodology.

4.1 Feature-based Methods

According to the prior research work [117], few statistical features are combined with the transformation method. The transformation method used in this research work is FFT as it is robust, effortless as well as fast. The feature-based methods are applied to the data produced from the sliding window explained in Section 2.3.

4.1.1 Fast Fourier Transform (FFT)

The key idea of FFT is decomposition, as demonstrated in Figure 4.1. The time-series data can be decomposed into several functions of cosine and sine using FFT, as stated in Section 3.1.1.3. These functions have amplitude, frequency and a phase, which are the features of FFT. The number of decomposition is proportional to the rigorous feature extraction, as illustrated in Appendix B. In this research work, the amplitude and frequency are considered, whereas the phase is disregarded as it tends to be redundant. The phase value of more than two activities is similar since it is cyclic in nature. The redundant features either deviate from the result or do not contribute to the improvement of the result [79].

In Figure 4.1, the smartwatch's accelerometer reading (blue plot) for the walking activity is decomposed into three readings (magenta, orange, green). The decomposed signal1 (magenta plot) has the highest amplitude and shortest time period compare to

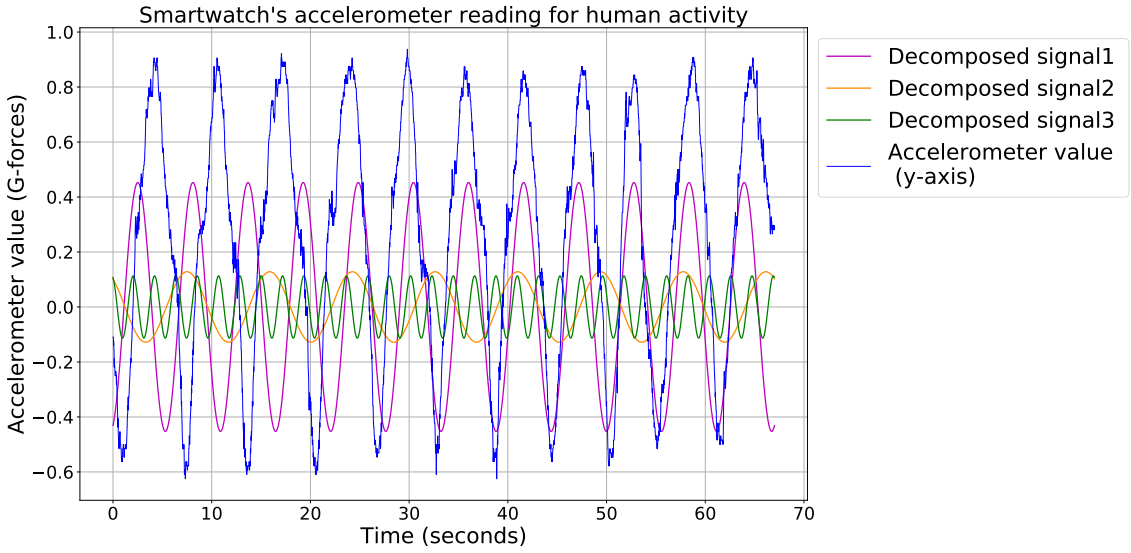

Figure 4.1: *FFT decomposition for motion data (The blue plot represents the smartwatch's yaw accelerometer reading for the human activity "walking" for 70 seconds the extended graph of Figure 2.2. The other three represents the decomposed signal from the original blue plot). Data from [47].*

the other decomposed signal. In contrast, the decomposed signal3 (green plot) has the highest time period and shortest amplitude. As the number of decomposition increases, the amplitude reduces, and the time period increases. Besides, the amplitude value for different classes becomes similar since the higher and lower amplitude components get reduced to the same range. Hence, it is sufficient to take the first two decomposition's amplitude and frequency to avoid redundancy. One interesting observation is the phase of the last two decomposed signals (green and orange) is the same. This observation justified the exclusion of the phase.

4.1.2 Statistical Features

Statistical features can represent any data [117]. Popularly used statistical features for time-series data [8] are incorporated in this research work. The other statistical features, such as energy, homogeneity, and Euler number, are not incorporated since the below eight features provide satisfactory results. Besides, as we append the features, the significant change in the performance attains saturation (explained in Chapter 5, Figure 5.4). Hence, adding features beyond the saturation point either causes overfitting or does not influence the performance [87].

4.1.2.1 Mean

Average of all the values is called mean, also known as the expected value since it is considered as a central value of the data.

$$E[M_{dj}] = \mu = \frac{\sum_{i=0}^{N} X_{ji}}{N_j} \tag{4.1}$$

where, $j = \{1, 2, ..., n\}$, n is the number of sliding window, $N_j$is the number of observations in the particular sliding window, X_{ji} is the value corresponding to the i^{th} instance of the j^{th} window, M_d is motion data similar to T_d in Equation 2.2.

4.1.2.2 Variance

Variance is the expectation of the squared difference from the expected value.

$$Var[M_{dj}] = \sigma^2 = E[M_{dj}^2] - (E[M_{dj}])^2 \tag{4.2}$$

4.1.2.3 Standard Deviation

Standard deviation measures the distance for each group of discrete values from the expected value.

$$sd[M_{dj}] = \sigma = \sqrt{Var[M_{dj}]} \tag{4.3}$$

It might appeal that the standard deviation or variance is redundant. This is not the case since the standard deviation represents the data in the lower dimension, and variance represents the data comparatively in a higher dimension. The linearly non-separable features by standard deviation will be separable features combined with variance.

4.1.2.4 Mean Absolute Deviation (MAD)

MAD calculates the average of distance between each data point to the central value. This is similar to the standard deviation but more robust to the outliers since it calculates the average of the deviations.

$$MAD[M_{dj}] = \frac{\sum_{i=0}^{N} X_{ji} - E[M_{dj}]}{N_j} \tag{4.4}$$

4.1.2.5 Maximum Peak

As the name indicates, it gives the maximum value from the series of data, illustrated in Chapter 3, Figure 3.1.

$$Max[M_{dj}] = Max\{X_{ji}\}_{i=0}^{N} \tag{4.5}$$

4.1.2.6 Maximum Peak

As the name indicates, it gives the minimum value from the series of data, illustrated in Chapter 3, Figure 3.1.

$$Min[M_{dj}] = Min\{X_{ji}\}_{i=0}^{N} \tag{4.6}$$

4.1.2.7 Entropy

Entropy is the measure of information; for time-series, it gives the complexity of the data, portraying the distribution of the discrete values.

$$Entropy_j = -\sum_{i=i}^{n} p_{ji} log_2 p_{ji} \tag{4.7}$$

$$p_{ji} = \frac{d_{ji}}{\sum_i d_{ji}} \tag{4.8}$$

$$d_{ji} = \frac{{M_{dj}}_i^2}{N} \tag{4.9}$$

where, M_{dji} is the motion data value in i^{th} instance, d_{ji} is the normalized point of i^{th} instance also called as density, p_{ji} is the normalized density function (probability density function).

4.1.2.8 Correlation

Correlation provides the measure of dependency between two distribution or variables. The correlation generates the value in range [-1, 1] where the sign indicates the direction. If there is no correlation the value is 0.

$$correlation[M_{dj}, M_{dj+1}] = r = \frac{E[M_{dj}, M_{dj+1}] - E[M_{dj}]E[M_{dj+1}]]}{\sigma_{M_{dj}} \sigma_{M_{dj+1}}} \tag{4.10}$$

where, r is the correlation coefficient, $Cov[M_{dj}, M_{dj+1}] = E[M_{dj}, M_{dj+1}] - E[M_{dj}]E[M_{dj+1}]]$ is the covariance between the motion data M_{dj}, M_{dj+1}. The correlation is computed between the x, y, z axis of each sensors in the dataset.

4.2 Autoencoders

As stated, earlier features are the low level representation of the raw data retaining their crucial properties. Also, these features should be able to reconstruct the raw data. Autoencoder is an unsupervised machine learning approach that performs the above-stated functionality [127]. Autoencoders are usually used for dimension reduction. PCA is one of the popularly used dimension reduction methods, but autoencoder offers the following advantages over PCA [126]:

1. **Non-linear transformation:** PCA is a linear transformation of data, whereas autoencoder can be both linear and non-linear based on the activation function.

2. **Subset of the data:** Autoencoder can transform many subsets of the data in parallel using convolutional layers and filters. In contrast, PCA can process sequentially.

3. **Efficiency:** Efficiency of the autoencoders can be improved by adding more layers. PCA efficiency depends upon the preprocessing of the data.

4. **Multiple output:** Autoencoders can provide features of different dimension at each stage in a single execution. PCA has to be executed again for different configurations.

Pre-trained autoencoders can be used on the new dataset as transfer learning, but the limitations are stated in Chapter 3, Table 3.1.

Autoencoder is the concatenation of two neural networks [127]; one is an encoder. The other is the decoder, joined by latent space as represented in Figure 4.2. The decoder is the encoder's mirror image, forming a bottleneck at the middle called latent space. The encoder transforms the input data (X) into a low dimension at latent space ($e(X)$). Data(Z) in the latent space is the features representing the input. The decoder generates the input data ($\vec{X}$) from the latent space. Hence, both input and output of the autoencoders are similar, yet the output is lossy ($\vec{X} = d(Z)$). The objective of the autoencoder is to have a minimal loss ($||X - \vec{X}||^2 = 0$). The loss is the distance between input and output, as stated in Figure 4.2. Autoencoder consists of four hyperparameters that influence performance as follows [148]:

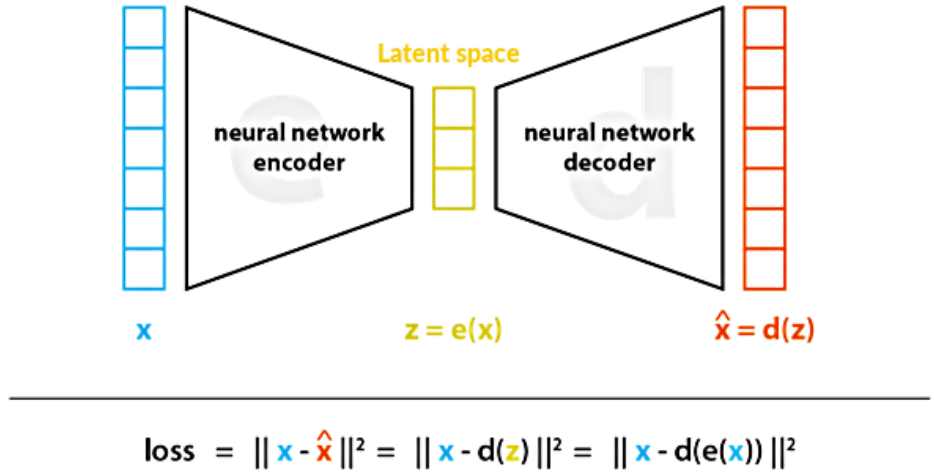

Figure 4.2: *Autoencoder and its loss. X is the input provided, compressed into latent space as $Z = e(X)$ using encoder. The latent space is used to reconstruct the input $\vec{X} = d(Z)$ using decoder. Reproduced from [47].*

1. **Number of layers:** The encoder and decoder are neural networks with several number layers; usually, they are symmetrical. Symmetry is not mandatory, but non-symmetry may introduce complexities. As the layers increase, the performance increases. The number of layers should be proportional to the number of samples; otherwise, this may overfit the data. This research work consists of 2 layers each.

2. **Dimension of the latent space:** The dimension of the latent space is determined by the loss value. The higher loss indicates that the dimension of latent space is not sufficient to reconstruct the input data or vice versa. The latent space dimension is increased until the loss attains a low saturated value. However, increasing the dimension beyond the saturation point results in degradation of the performance (overfitting). In this research work, the latent space dimension of about 180 to 200 gives satisfactory performance.

3. **Loss function:** Mean squared error or binary cross-entropy is used as a loss function based on the datatype. In this research work, the mean squared error is used after comparing the performance of both the loss function.

4. **Number of nodes in layers:** The number of nodes in the layers takes the value between the input layer and latent space dimension. In this research work, the nodes in the layers are double the nodes in the latent space. All the layers consist of LSTM cells as nodes.

The prior research work [132] used autoencoder related to computer vision techniques for time-series feature extraction. This research work uses the NLP approach of autoencoder for motion data feature extraction. Autoencoders are popularly used for speech recognition, machine translation, caption generation, and summarization [60][39]. In these applications, the autoencoder uses an embedding layer prior to the encoder. The neural network can not process strings, so the embedding layer converts the words to a corresponding numerical value in vector form. The compression of embedded data is done sequentially based on their time of occurrence into the latent space. This latent space is available at each time step of the decoder for reconstructing the input data.

In this research work, a similar technique is incorporated except the embedding layer since the time-series data is already a numeric value in chronological order. The autoencoder is trained with the number of samples that resulted from the sliding window. The decoder is trained only to compute the loss function. The trained model is bifurcated at the latent space, and these latent space data (features) are provided as input for the SVM. SVM classifies based on the features provided by the latent space. This trained model can be used for transfer learning for feature extraction and classification as one unit, such as auto-SVM.

4.3 Evaluation

The feature extraction is validated by the classification task executed by SVM. The classification performed by SVM should be evaluated to understand the efficiency of autoencoders. In addition, it supports the performance comparison of autoencoders with the feature-based methods. Since the datasets are annotated, it is possible to produce a confusion matrix. Following are the two evaluation metrics used in this research work.

4.3.1 Accuracy

The measure of closeness of the predicted value to the true value is called accuracy. This provides information about the performance of the method [140].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.11}$$

where, TP is True Positive, TN is True Negative, FN is False Negative, and FP is False Positive.

4.3.2 F1-score

F1-score gives information about the enhancement in the performance of the method [141]. In the classification task, there are two main objectives; one is to minimize the incorrect classification (FP), which maximizes the precision, other is to minimize the incorrect missing(FN) to classify it correctly, which maximizes the recall. These two parameters provide the direction of enhancement.

$$Precision = \frac{TP}{TP + FP} \tag{4.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.13}$$

$$F1\text{-}score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.14}$$

4.4 Workflow

The previous sections provided insight into the methods. According to the workflow, organizing these methods is illustrated in Figure 4.3 and detailed below.

1. **Data collection:** Motion data used for this research work is collected from the publicly available repositories such as UCI and Kaggle as stated in Chapter 1. This research work is more focused on the dataset from [94]. The reason is that the dataset is a real-time recording of human activities in outdoor conditions. Due to time constraints, the dataset for this research work is not generated.

2. **Preprocessing:** The dataset collected in the previous steps are organized in a particular order. This order can be based on activity, subject, sensor, and random. This order is read sequentially and composed in an array format chronologically. Then checked for missing values, null values, datatype and scaled the values using a standard deviation normalizer. The sliding window is applied to this array of data, as stated in Section 2.3.

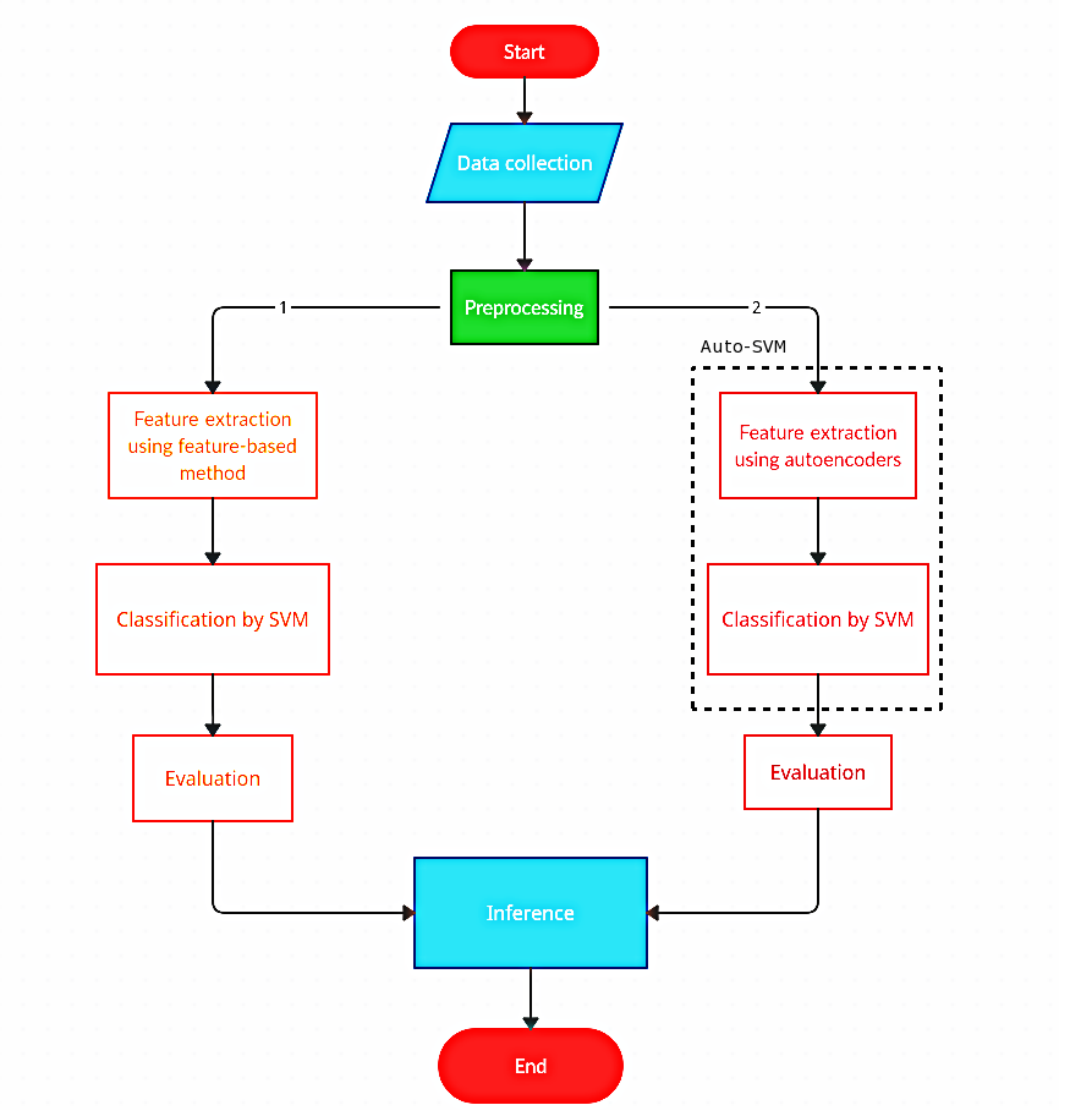

Figure 4.3: *Workflow of this research work. Red color indicates the start and end of the work, blue indicates the manual collection of information, data, and analysis, green denotes the process of data cleaning and organizing, two methods which are independent runs parallel from the preprocessing. Feature-based methods are denoted by orange, autoencoders by maroon. Differentiation in color is provided to indicate the individuality of the process.*

3. **Feature extraction using feature-based method:** As explained in the Section 4.1, features such as FFT, mean, standard deviation, variance, mean absolute deviation, correlation, entropy, and maximum as well as the minimum peak of the

time-series data is extracted individually. These extracted features are concatenated together as given in the Equation 2.21.

4. **Classification by SVM:** The concatenated features, along with their labels, are provided as input to the SVM. The SVM learns and predicts based on the features provided. The performance of the features is evaluated using accuracy and f1-score.

5. **Auto-SVM:** As described in Section 4.2, autoencoders extracts the features automatically in their latent space. These features are fed as input to the SVM for classification. The performance of the features is evaluated using accuracy and f1-score.

6. **Inference:** This is a concrete step of this research work. The results of step-3 are perceived to be a benchmark with which step-5 is being analyzed. Additionally, the influence of each hyperparameter, the nature of the data, the number of dimensions, and the number of samples on feature extraction using autoencoder is investigated.

5

Experiments and Results

This chapter describes, firstly, about different datasets used in this research work. Secondly, the performance and analysis of the proposed methodology both feature-based methods and autoencoders on those datasets. Finally, a comprehensive comparison of the methodology.

5.1 Dataset Description

This research work is experimented with three different datasets, namely Motionsense (does not mean the sensor manufacturer) [94], Wireless Sensor Data Mining version 1.1 (WISDM_v_1.1)[140], and Wireless Sensor Data Mining (WISDM) [140]. All three datasets consist of cyclic and multivariate time-series data. These datasets differ in the situations that have been recorded, the number of sensors, the number of participants, and the streaming of data (explained in Section 2.1.3). A comprehensive description of each dataset is provided below.

5.1.1 Motionsense Dataset

This dataset is available in the kaggle repository [112] and maintained in the Github of the author [93]. The dataset includes discrete values from the triaxial accelerometer and gyroscope from the smartphone and a smartwatch that totals up to 12 dimensions. The data is collected from 24 participants after recording their physical attributes like gender, age, weight as well as height. Each participant is provided with an iPhone 6s smartphone in their front trouser pocket and a smartwatch at their wrist. The participants performed 6 activities, namely walking, sitting, standing, jogging, downstairs, and upstairs around the Queen Mary University of London Mille End campus to replicate the common everyday activities. A detailed map of the activities performed is illustrated in Figure 5.1. The data is collected through the Crowdsense application available at the Apple

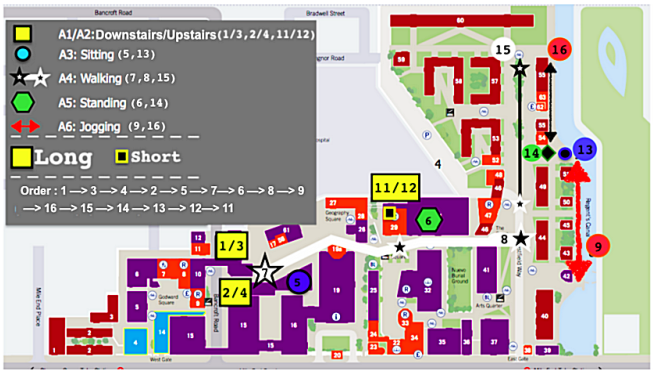

Figure 5.1: *Motionsense trails. Reproduced from [94].*

store. The participants start the Crowdsense application before performing their tasks and stop the application at the end of their tasks. The time of task switching is recorded by creating an interval in the application supported for annotation. In Figure 5.1, there are 15 trails of activities that each participant performed in the same order. The 15 trails' order is downstairs, upstairs, upstairs, downstairs, sitting, walking, standing, walking, jogging, jogging, walking, standing, sitting, upstairs, and downstairs. The first 9 trails consume 2 to 3 minutes, and the last 6 trails consume 0.5 to 1 minute. The attributes of the dataset are provided in Table 5.1.

Attributes	**Value / type**
Number of participants	24
Number of activities/labels	6
Annotated	Yes
Sampling frequency of the sensors	50 Hz
Types of sensors	Triaxial accelerometer and triaxial gyroscope
Number of device/sensor placements	2 (wrist, waist)
Type of devices	smartphone (iphone 6s) and smartwatch (Apple)
Continous activities	Yes
Recorded environment	Outdoor condition
Number of dimensions	12
Total discrete values	1,412,865
Missing values	None
Varied length	Yes
Dataset file extension	.csv

Table 5.1: *Attributes of Motionsense dataset.*

5.1.2 Wireless Sensor Data Mining version 1.1 (WISDM_v_1.1) Dataset

This dataset is available in the UCI repository [47]. The dataset is collected by the WISDM Lab, Department of Computer and Information Science, Fordham University. This dataset is named version 1.1 since the Lab provided another dataset publicly, appending further activities and sensor data. The dataset includes discrete values from the triaxial accelerometer and gyroscope from the smartphone that sums up to 6 dimensions. The data is collected from 36 participants. Each participant is provided with a Samsung S2 smartphone in their waist bag, as demonstrated in the video [104]. The participants performed 6 activities: walking, sitting, standing, lying, downstairs, and upstairs inside the WISDM Lab. The data is collected through some custom-made application running on the smartphone (the name is not specified). The application is started by the researcher and provided to the participants before performing their tasks. The attributes of the dataset are provided in Table 5.2. This dataset has fewer dimensions and samples as well as recorded in the Lab condition compared to the Motionsense dataset. Recording in the Lab condition does not depict the real circumstances where the ground floor is not even, different floor types, and the obstacles in the trails.

Attributes	**Value / type**
Number of participants	36
Number of activities/labels	6
Annotated	Yes
Sampling frequency of the sensors	20 Hz
Types of sensors	Triaxial accelerometer and triaxial gyroscope
Number of device/sensor placements	1 (waist)
Type of devices	smartphone (Samsung Galaxy S2)
Continous activities	Yes
Recorded environment	Lab condition
Number of dimensions	6
Total discrete values	1,098,207
Missing values	None
Varied length	Yes
Dataset file extension	.txt

Table 5.2: *Attributes of WISDM_v_1.1 dataset.*

5.1.3 Wireless Sensor Data Mining (WISDM) Dataset

This dataset is available in the UCI repository [47]. This dataset is the recent version from the WISDM Lab. The dataset includes discrete values from the triaxial accelerometers and gyroscopes in the smartphone and a smartwatch that scores up to 12 dimensions. The data is gathered from 51 participants. Each participant is provided with a Google Nexus 5x smartphone in their front trouser pocket and an LG smartwatch at their wrist. The participants performed 18 activities, namely walking, sitting, standing, jogging, stairs, typing, brushing teeth, eating soup, eating chips, eating pasta, drinking from the cup, eating the sandwich, kicking, playing tennis, basketball, writing, clapping and folding clothes to replicate most of the everyday activities. The data is collected through some custom-made application running on the smartphone (the name is not specified). The activities are recorded discretely, not similar to the prior datasets. In prior datasets, after performing all the activity, the application is forced to stop recording, but in this dataset, after each activity, the application is restarted again. The attributes of the dataset are provided in Table 5.3. This dataset consists of higher activities and samples compared to the former datasets.

Attributes	**Value / type**
Number of participants	51
Number of activities/labels	18
Annotated	Yes
Sampling frequency of the sensors	20 Hz
Types of sensors	Triaxial accelerometer and triaxial gyroscope
Number of device/sensor placements	2 (wrist, waist)
Type of devices	smartphone (Google Nexus 5x) and smartwatch (LG G watch)
Continous activities	No
Recorded environment	Lab/indoor condition
Number of dimensions	12
Total discrete values	15,630,426
Missing values	None
Varied length	No
Dataset file extension	.txt

Table 5.3: *Attributes of WISDM dataset.*

5.2 Machine Configuration

All three datasets are large, as described in the prior sections. Processing these large datasets requires good computation power and memory capacity. The experiment is conducted on the Platform for Scientific Computing at Bonn-Rhein-Sieg University [57]. The server consists of hardware threads of 5,300 bridging 48,000 Graphical Processing Unit (GPU) cores with a collective memory of 460 TeraByte (TB). A subset of this configuration is made available for students' research work. The details are below in Table 5.4.

Components	**Description**
Operating system	Nitrogen, Linux 7.8
Central Processing Unit (CPU)	50 nodes with 2x Xeon processors having 16 cores
Graphical Processing Unit (GPU)	Nvidia Tesla V100
Memory	50 GB for computation and 100 GB for storage

Table 5.4: *Machine configuration at Bonn-Rhein-Sieg University [57].*

A good practice is to execute a small portion or subset of the code to check for error-free before executing on the University Computing Platform. A detailed software prerequisite is explained in Appendix D.

5.3 Evaluation of Feature-based Methods

This section deals with the experiment and analysis of the feature extraction performed by both FFT and statistical features.

5.3.1 Experiment

As described in Section 4.4, the Motionsense dataset is extracted from the .csv files as a single data frame (table/array). This data frame is examined for null or missing values, normalized using a standard deviation scale to have a range [-1, 1]. Normalizing the values improves accuracy. The sliding window is implemented on the normalized data frame and labels retaining the array structure. The former steps are common for both feature-based methods and autoencoders. Hence, the array after the sliding window implementation is stored as .pkl format for the autoencoders. The features are extracted from this array and stored as a large feature array with their respective labels. The features such as mean, standard deviation, variance, entropy, maximum as well as minimum peak are

calculated using the predefined functions. The detailed software prerequisite and usage are mentioned in Appendix D. The feature array along with the labels are provided as inputs to the SVM for classification. The data is split as 80-20% for training and testing, respectively. The SVM classification results from the features based on the FFT and other statistics are provided in Figure 5.3. The evaluation result is presented in Table 5.6, and these results are affected by hyperparameters. The hyperparameters and their values for the feature-based methods are presented in Table 5.5. These values influence the change points presented in the motion data (explained in Section 2.1.3). The values in Table 5.5 are obtained after several executions on different combinations of the time step and window size as presented in Figure 5.2.

Hyperparameters	**Value**
Time step	50 milliseconds
Window size	10 seconds

Table 5.5: *Hyperparameters for feature-based method.*

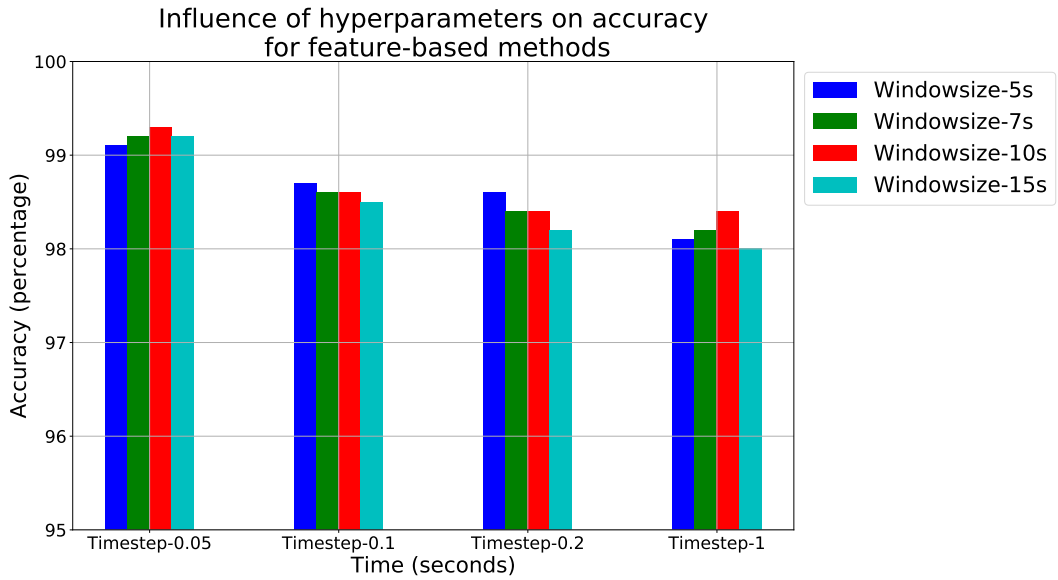

Figure 5.2: *Influence of hyperparameter on accuracy for feature-based methods(y axis scale is shrunken to 95-100 for better visualization since all the results where above 98%).*

According to the window size and time step values, the change point can appear at the start of the window or anywhere within the window. There are many change points in each activity. The change point at the transition of the activities (the point where one activity ends and another activity starts) is targeted at the start of the sliding window.

This can be achieved by holding the time step equal to the reciprocal of the sampling frequency of the sensor. The Motionsense dataset has a sampling frequency of 20 Hz, so the reciprocal is 1 millisecond. Time steps of 1 millisecond provide an equal amount of samples to the raw data, which results in out of memory error while extracting the features (computation overloading). Hence, the minimum possible time step value is 50 milliseconds. The value of 50 milliseconds can be deviating from the target. Therefore 3 more values, such as 100 milliseconds, 200 milliseconds, and 1 second are preferred for comparison. Considering the trail time from the dataset description, each activity could take 5 seconds to 15 seconds (6 activities in 30 seconds and 9 activities in 2 minutes). Besides, based on the results of the prior research work [140], four values such as 5, 7, 10, and 15 seconds for window size are preferred. These values contribute about 16 combinations that are experimented with and presented in Figure 5.2. The time step of 50 milliseconds performed comparatively well for any window size; one reason is that it samples more data than the other three values. The window size of 5 seconds performed well with the time step of 100 milliseconds and 200 milliseconds. The window size 10 seconds performed well with the time step 50 milliseconds and 1 second. The time step of 50 milliseconds and window size 10 seconds samples more data collectively. Besides, these values performed well among other combinations, so it is preferred throughout the experiments and other datasets. The accuracy is 99.38% from the feature-based methods for Motionsense data with the hyperparameter's value as in Table 5.5. However, the research work [94] has achieved 95% for the same dataset with the time step 100 milliseconds and window size 1 second. Hence, higher samples provide better results yet increase the computational load.

5.3.2 Analysis of the Results

Table 5.6 is the classification report for the feature-based methods. The activities jogging, sitting, and standing is classified aptly with very negligible error as the f1-score indicates 100%. The activities sitting and standing are identical, as there is no physical movement. The different orientations of the arm could have caused the SVM effortless to distinguish correctly. In contrast, the upstairs and downstairs are often misclassified with walking or vice versa, as presented in Figure 5.3. This misclassification is possible since the walking, upstairs, and downstairs are similar activities, as the readings are analogous to one another in the video [104]. A triaxial altimeter that measures the altitude will be a beneficial addition to differentiate the activities downstairs and upstairs from walking. The upstairs and downstairs also can be differentiated with the direction

parameter. Hence, a combination of triaxial accelerometer, gyroscope as well as altimeter will improve the current performance.

Accuracy score:	99.38			
	precision	**recall**	**f1-score**	**support**
Walking	1.00	0.99	0.99	34648
Jogging	1.00	1.00	1.00	13512
Sitting	1.00	1.00	1.00	33649
Standing	1.00	1.00	1.00	30428
Upstairs	0.98	0.99	0.98	15863
Downstairs	0.99	0.99	0.99	13162
accuracy			0.99	141262
macro avg	0.99	0.99	0.99	141262
weighted avg	0.99	0.99	0.99	141262

Table 5.6: *Evaluation report for Motionsense dataset based on the features from feature-based methods (the report is replicated in table form. Support refers to number of samples).*

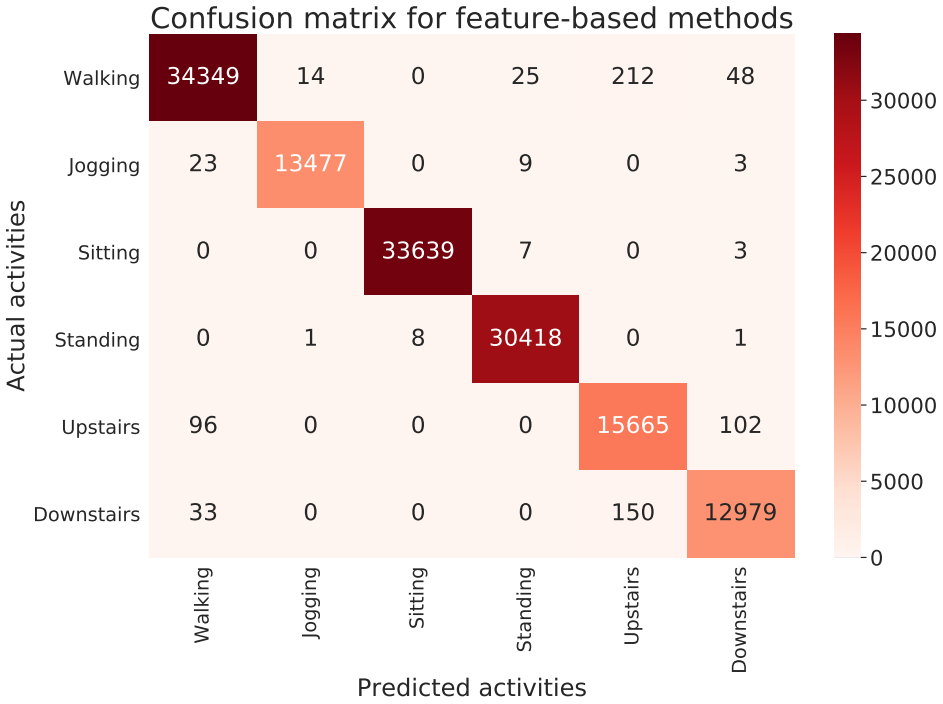

Figure 5.3: *Confusion matrix for Motionsense dataset based on the features from feature-based methods (activities jogging, sitting, and standing are classified comparatively well to the activities upstairs and downstairs).*

All the features have contributed to the accuracy of 99.38%. As described in Section 4.1, the features are sequentially added and tested individually for classification since few features can degrade the performance. In this experiment, the entropy feature generated an accuracy lower to 10%. This low accuracy indicates that the feature is biased towards a certain class. Hence, the entropy is excluded from the feature array. At the same time, the other features performed comparatively well. Figure 5.4 illustrates each features classification performance in descending order. Transformation methods, either DCT or FFT, performed slightly lesser than the statistical features such as mean, maximum, and minimum peak. The mean feature sound to be one of the powerful features.

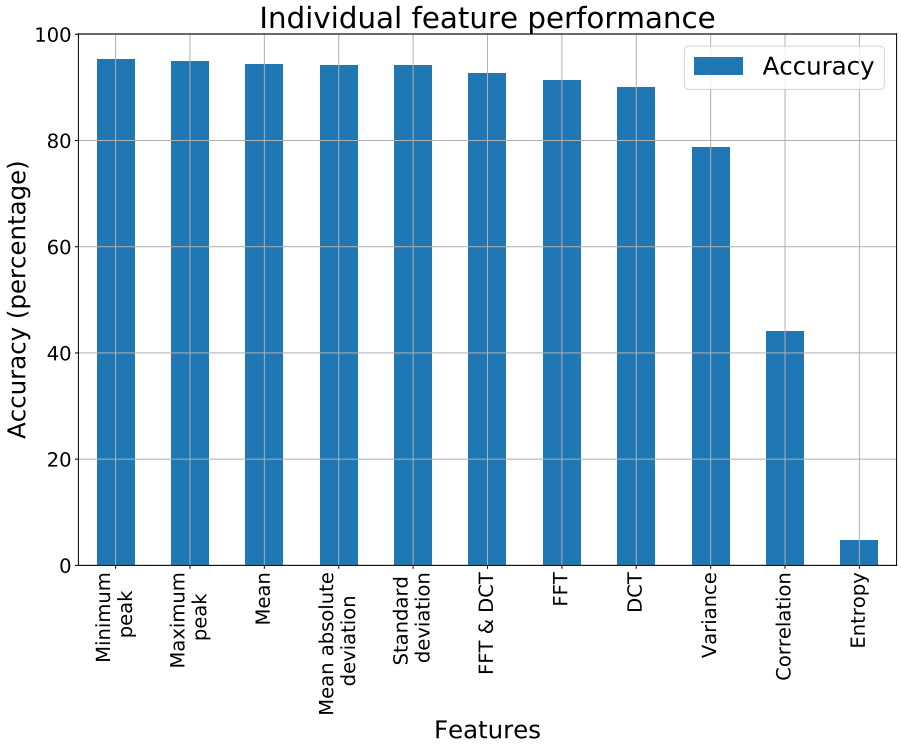

Figure 5.4: *Performance of each features for feature-based method (bars are arranged in the descending order).*

According to the research work, [117], few statistical features are included. Figure 5.5 illustrates the contribution of the statistical features that improve overall performance. DCT is implemented to highlight the significance of statistical features. The transformation methods, either DCT or FFT solely, can yield a good classification. However, the transformation methods combined with statistical features provide efficient classification, which is near accurate. In Figure 5.5, the accuracy of the DCT is less than the FFT. In

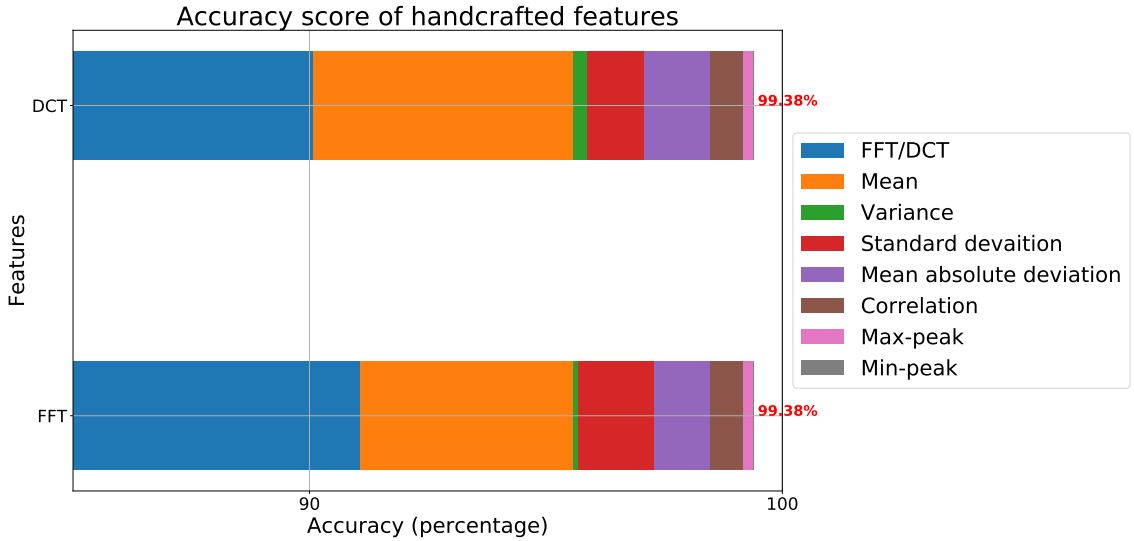

Figure 5.5: *Contribution of each features from feature-based method (each feature contribution is visible except Min-peak since it does not provide any new information to classify based on it. So this is a redundant feature, or the accuracy attains the saturation point).*

contrast, the DCT and FFT, with the statistical features, produce equivalent performance. Each feature strengthens the performance when it is totally dealt with. The minimum peak feature resembles redundant even though the individual performance was valuable. Since there is no additional information provided by the minimum peak for the classifier to learn. Even the order of the features influences the intermediate results, yet the overall performance is identical. Including additional features may or may not enhance the results, surpassing the scope of this research work. As the hyperparameters and useful features are known, implementing the same to the other datasets yields the following results.

Table 5.7 provides the evaluation report for the WISDM_v_1.1 dataset. The performance of the feature-based method on the WISDM_v_1.1 is less to the Motionsense dataset. The f1-score for the activities downstairs and upstairs is less than 80%, indicating more misclassification of these activities. Figure 5.6 depicts that all the activities have misclassifications. Activities sitting and standing are relatively classified well. However, other activities are misclassified highly among one another. The number of samples in this dataset is 50% higher than the Motionsense dataset, yet there is an underperformance. This is due to the inadequate number of sensors, and sensor placements, which is the only distinction from the Motionsense dataset. Hence, the number of samples with an adequate number of sensors can generate better performance. The results for the WISDM

Accuracy score: 85.50

	precision	recall	f1-score	support
Walking	0.87	0.88	0.87	85022
Jogging	0.89	0.89	0.89	68489
Sitting	0.89	0.88	0.89	11911
Standing	0.87	0.89	0.88	9677
Upstairs	0.77	0.75	0.76	24626
Downstairs	0.75	0.75	0.75	19876
accuracy			0.86	219601
macro avg	0.84	0.84	0.84	219601
weighted avg	0.85	0.86	0.85	219601

Table 5.7: *Evaluation report for WISDM_v_1.1 dataset based on the features from feature-based methods (the report is replicated in table form. Support refers to number of samples).*

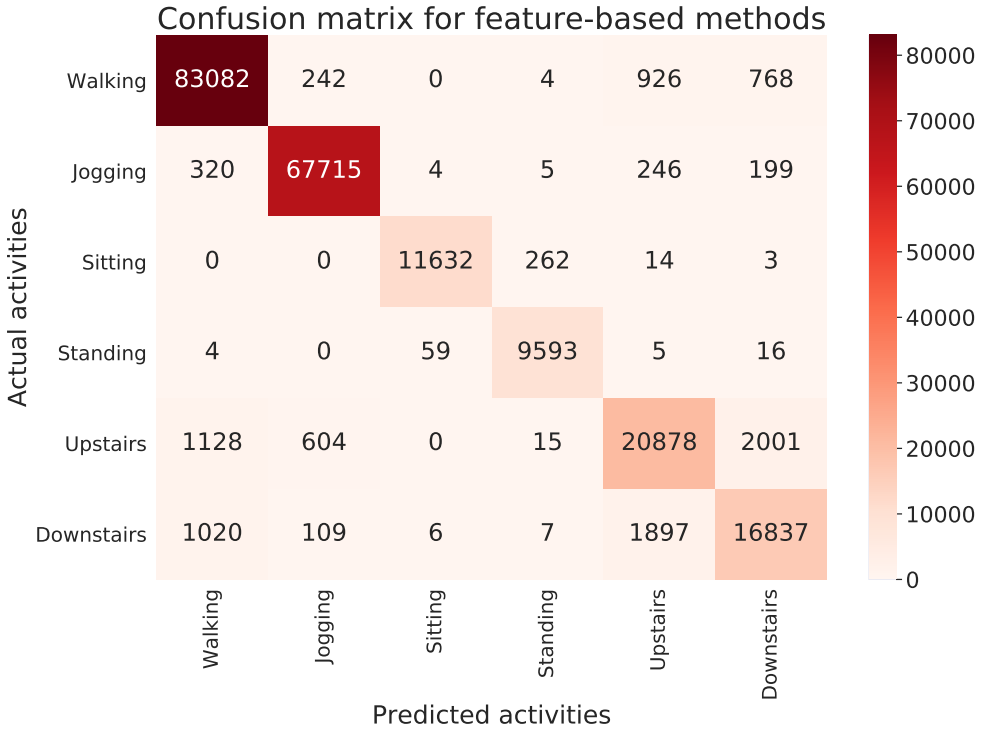

Figure 5.6: *Confusion matrix for WISDM_v_1.1 dataset based on the features from feature-based methods (all the activities are misclassified, among these sitting holds good).*

dataset are provided below.

From Table 5.8 it is obvious that the performance of the feature-based methods on the WISDM dataset is high compared to the other two datasets. The f1-score for the

Accuracy score:	99.65			
	precision	**recall**	**f1-score**	**support**
Walking	1.00	1.00	1.00	7153
Jogging	1.00	1.00	1.00	7140
Climbing Stairs	1.00	1.00	1.00	7149
Sitting	1.00	1.00	1.00	7134
Standing	1.00	1.00	1.00	7080
Typing	1.00	1.00	1.00	7148
Brushing Teeth	1.00	1.00	1.00	7086
Eating Soup	1.00	1.00	1.00	7278
Eating Chips	1.00	1.00	1.00	7144
Eating Pasta	1.00	1.00	1.00	7185
Drinking from cup	1.00	0.89	0.94	7134
Eating Sandwich	0.91	1.00	0.95	7129
Kicking	1.00	1.00	1.00	6933
Playing Tennis	1.00	1.00	1.00	7104
Basketball	1.00	1.00	1.00	7137
Writing	1.00	1.00	1.00	7289
Clapping	1.00	1.00	1.00	7070
Folding Clothes	1.00	1.00	1.00	7227
accuracy			0.99	128520
macro avg	1.00	0.99	0.99	128520
weighted avg	0.99	0.99	0.99	128520

Table 5.8: *Evaluation report for WISDM dataset based on the features from feature-based methods (the report is replicated in table form. Support refers to number of samples).*

activities drinking from cup and eating sandwich has recorded around 95%. Drinking from cup and eating sandwich are quite similar activities. Besides, Figure 5.7 shows misclassification for those two activities. Whereas the other activities are classified very accurately. Since the dataset has trivial varying lengths between each activity and the start as well as the end of the activity are known due to the discrete recording. Overall, the feature-based methods performed well on the three datasets. The results enforce the following:

1. The triaxial accelerometer and gyroscope are minimum required sensors for HAC.

2. The wrist and waist are the minimum required sensor placements for HAC.

3. Efficient change point detection at the activity transition.

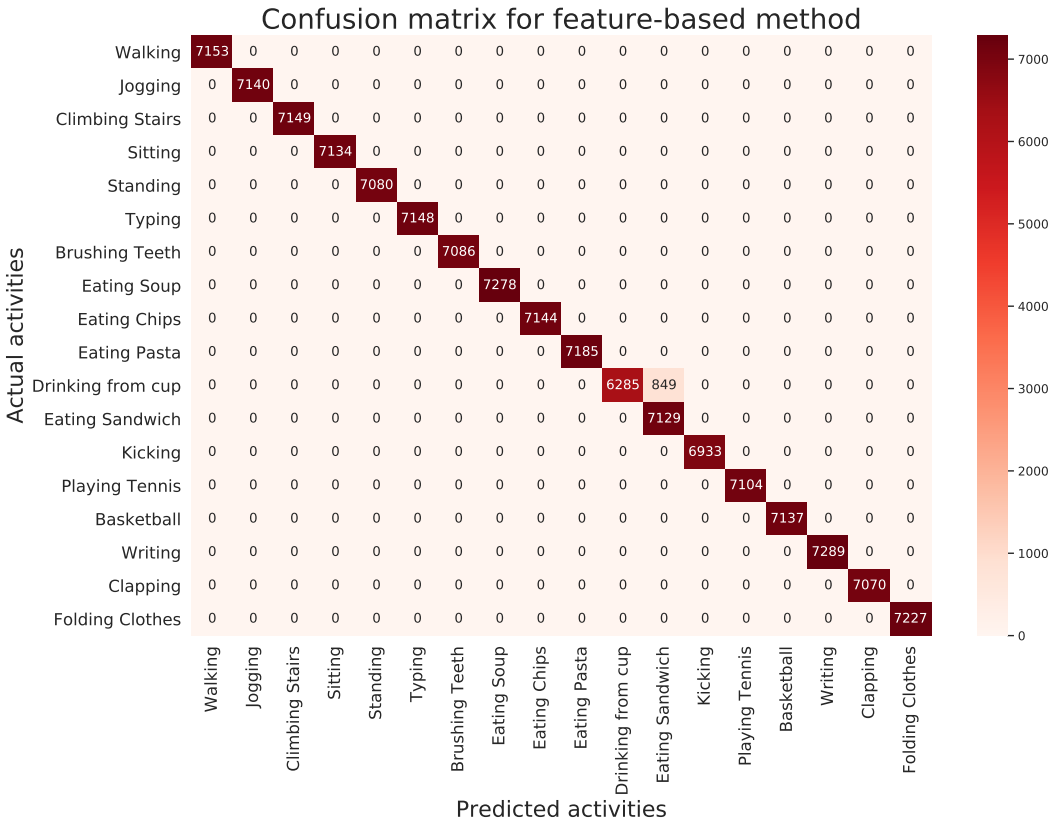

Figure 5.7: *Confusion matrix for WISDM dataset based on the features from feature-based methods (the only misclassification; Drinking cup is predicted as Eating sandwich).*

5.4 Evaluation of Autoencoders

This section deals with the experiment and analysis of the feature extraction performed by autoencoders.

5.4.1 Experiment

As specified in Section 5.3.1, the data and labels are restored from the .pkl file. There is no additional preprocessing performed for autoencoders. The restored data is provided as an input to the autoencoders. The autoencoders are modeled using Keras supported by the TensorFlow framework. The autoencoder is trained on 80% of the data, and the other 20% of data is used for validation. After the model is trained on the data, the learned parameters of the encoder network are stored in a .h5 file format along with the data in the latent space. The decoder network is used to compute the loss between the

Hyperparameters	**Value**
Time step	50 milliseconds
Window size	10 seconds
Learning method	Adam
Initial learning rate	$1X10^{-3}$
Weight initialization	Xaviar (he_uniform)
Activation function	Sigmoid
Batch size	4
Epochs	20
Loss	Mean squared error
Latent space dimension	180
Number of layers	4

Table 5.9: *Hyperparameters for autoencoders.*

original input and reconstructed input, as presented in Figure 4.2. The decoder network is not required after the model is trained. The saved model can extract features from the data, outcomes from the sliding window with different time steps, and the same window size. The autoencoder's extracted features are by default an array, which is an input to the SVM for classification. The splitting of data is the same as the previous method, which is 80-20%. The SVM classification based on autoencoders' features is exhibited in Figure 5.9 with the classification report, provided in Table 5.10. These results are influenced by various hyperparameters presented in Table 5.9. The time step and window size are common to both the methods, which are constant. The other parameters are obtained from a few test runs based on previous research work [132] as outlined in detail below:

1. **Learning method:** Adam is a commonly used learning method that combines the momentum and RMSProp [75], which results in an optimum solution compared to other learning methods. The prior research work [132] also uses Adam.

2. **Initial learning rate:** The learning rate is a sensitive and crucial parameter. If the initial value of the learning rate is less, then the weight update is very slow. If the initial value is high, then there is a prospect of ignoring the optimal solution. The default value is $1X10^{-3}$, which is not modified according to the research work [132]. The weight update will be slow, yet it provides an optimum solution.

3. **Weight initialization:** The Xaviar weight initialization consists of two types such as Glorot and He. This research work adopts He as the Figure 5.8 illustrates, He performs better than Glorot.

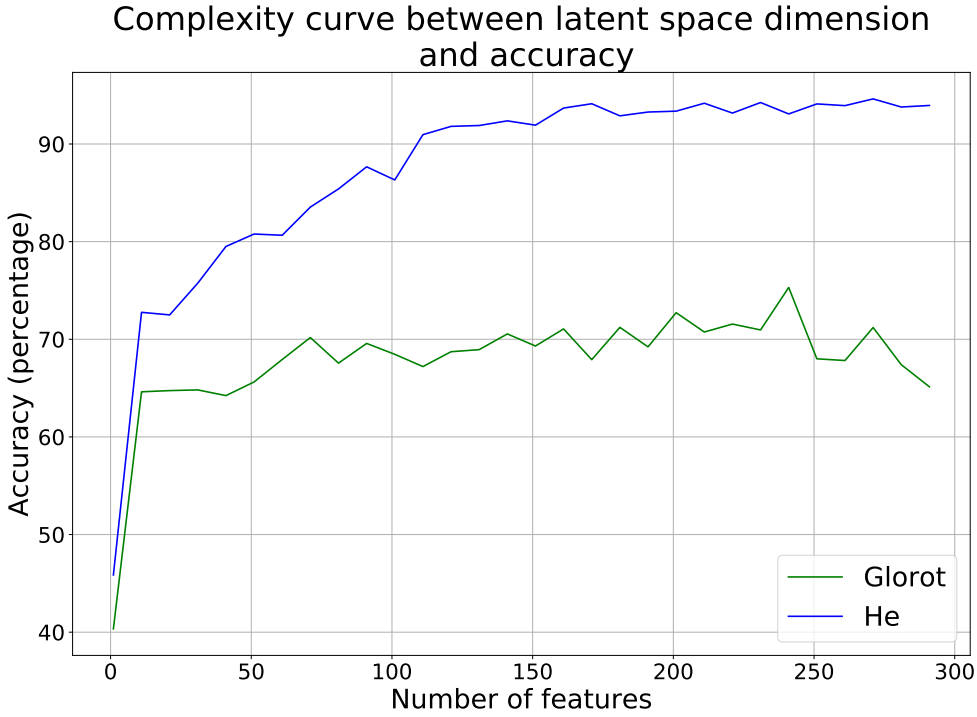

Figure 5.8: *Complexity curve between latent space dimension and accuracy (the Glorot appears to be decreasing after 270 features, whereas He appears to be almost saturated).*

4. **Activation function:** Rectified Linear Unit (ReLU) is regularly used in NLP [106]. The sigmoid function gives an improved performance compared to ReLU and tanh function in this research work.

5. **Loss:** Mean squared error and binary cross-entropy are the widely used loss functions for autoencoders [52][120]. The mean squared error gives an accuracy of around 1% higher than the binary cross-entropy for a similar configuration.

6. **Latent space dimension:** The value for this parameter is found by the complexity curve. Figure 5.8 illustrates the complexity curve between the number of features and accuracy for two different weight initialization types. The performance of the features is consistent from 180, which could be the optimum value to decide. As the epoch increases, a higher number of features may tend to overfit the data.

7. **Number of layers:** The number of layers can be modified based on the number of samples. This research work adopted a basic structure of an autoencoder used for generating image caption [120], which is 4 layers.

8. **Batch size and epoch:** These two parameters always have a trade-off between the computational load and accuracy. The epoch is monitored based on a checkpoint that increases until the required accuracy is met or the performance is consistent over a period. The average epoch was 20. The batch size initially was set to 128. As the batch size decreases, the performance increases. However, the computation load was very high, as the batch size was 4 or lesser. Hence, the batch size is set to 8.

5.4.2 Analysis of the Results

Table 5.10 presents the evaluation report based on the features extracted by the autoencoders. The only activity that has been classified perfectly is sitting. In contrast, other activities are misclassified as a pair, such as walking, often misclassified with jogging or vice versa. Activity upstairs is misclassified with downstairs or vice versa, as demonstrated in Figure 5.9. This pattern indicates that there should be an additional sensor placement distinguishing this pair of activities. According to the feature-based methods, there should be an altimeter to distinguish the activity upstairs and downstairs. Besides, the autoencoder results indicate a need for another sensor placement at the leg to differentiate jogging and walking. Hence, including two sensors, such as an altimeter at the current sensor placements and a triaxial accelerometer as well as a gyroscope at the leg, could improvise the current performance.

Accuracy score:	98.64			
	precision	**recall**	**f1-score**	**support**
Walking	0.98	0.99	0.99	34648
Jogging	0.99	0.99	0.99	13512
Sitting	1.00	1.00	1.00	33649
Standing	0.99	1.00	0.99	30428
Upstairs	0.96	0.96	0.96	15863
Downstairs	0.97	0.96	0.96	13162
accuracy			0.99	141262
macro avg	0.98	0.98	0.99	141262
weighted avg	0.99	0.99	0.99	141262

Table 5.10: *Evaluation report for Motionsense dataset based on the features from autoencoders (the report is replicated in table form. Support refers to number of samples).*

All the autoencoders' latent space features have contributed to the accuracy of 98.64% as described in Section 4.2. Unlike the feature-based methods, the features from the

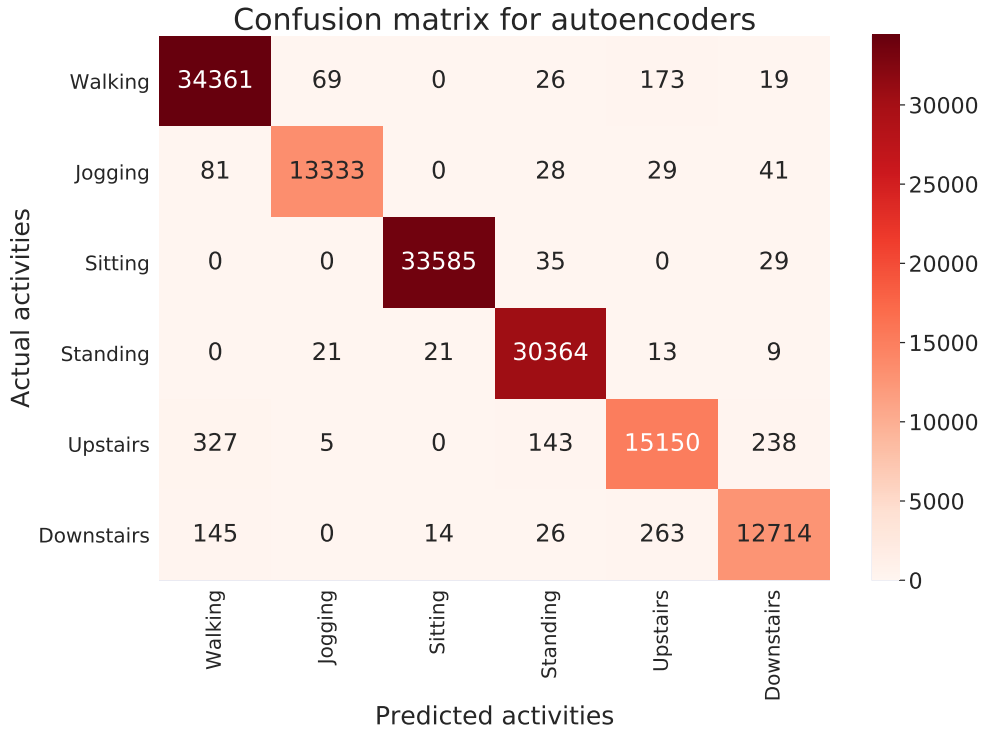

Figure 5.9: *Confusion matrix for Motionsense dataset based on the features from autoencoders (all the activities are misclassified, among these sitting and standing appears to be better classified).*

autoencoders are not sequentially appended and tested. However, analyzing in such a way tend to provide intense insight into the features extracted. Additionally, there is no proper understanding of the type of features extracted as in the feature-based methods. The raw data is of 12 dimensions due to 4 triaxial sensors. Thus, 180 features are considered 15 features each of 12 dimensions as per the feature-based methods. Figure 5.10 depicts these 15 features classification performance individually. All the features are satisfactory since each feature has performed above 50%. There is no strong feature such as mean, FFT, and DCT that contributed individually to 90% of total accuracy.

Figure 5.11 illustrates the contribution of each feature from the autoencoders. Similar to the feature-based methods, there exists a pattern. The first feature contributed the most, up to 86%, the contribution of each features decreases as the number of features increases. According to Figure 5.11, the contribution of Feature-14 is not visible, yet removing it degrades the performance by 0.5%. Hence, there are no redundant features like feature-based methods. Besides, the number of features selected as 180 is sensible. Since these hyperparameters provide satisfactory results, implementing the same to the

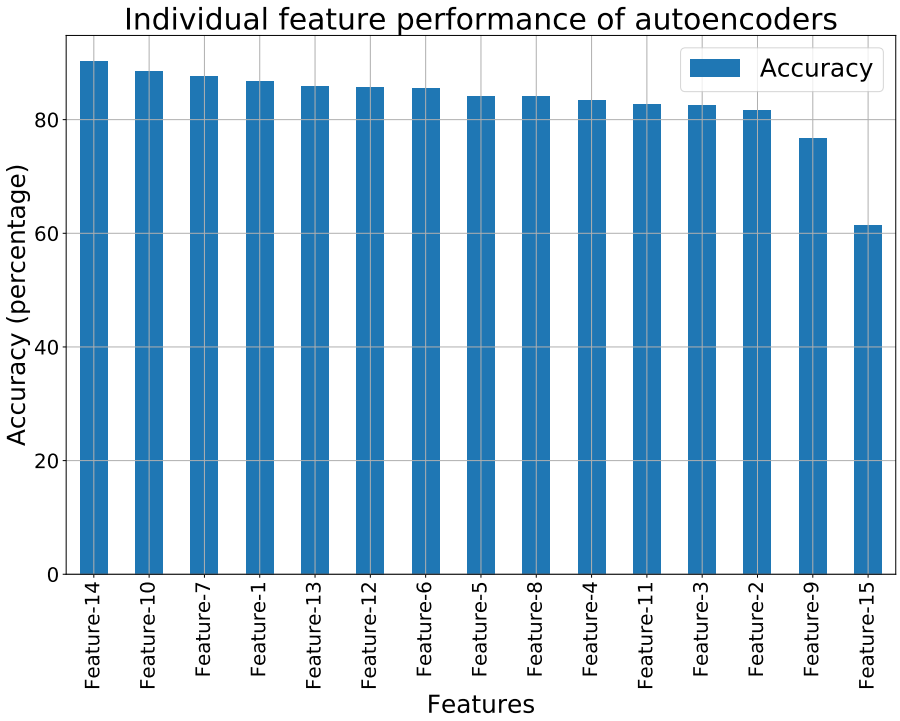

Figure 5.10: *Performance of each features for autoencoders (all the features has performed well generating more than 60% of accuracy individually).*

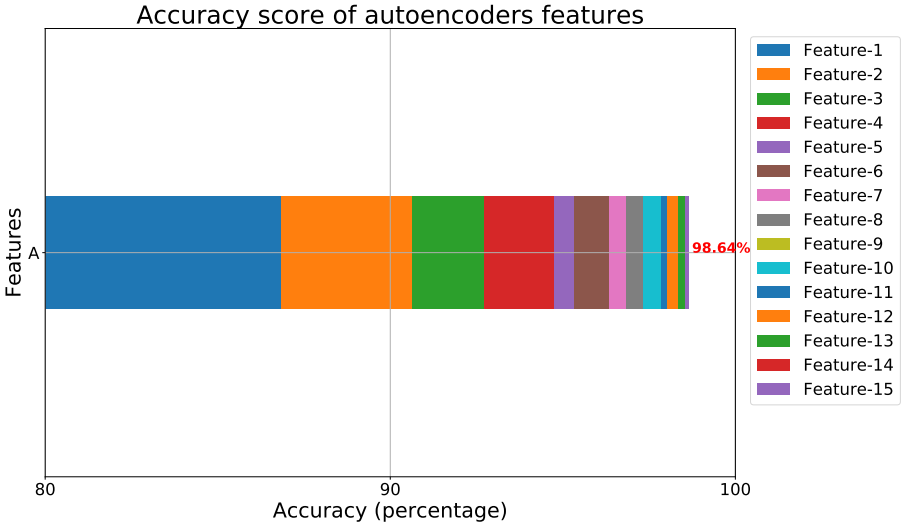

Figure 5.11: *Contribution of each features from autoencoders (contribution of Feature-14 is not visible yet when removed the accuracy drops by 0.5%).*

other two datasets generate the following results.

Accuracy score:	70.46			
	precision	**recall**	**f1-score**	**support**
Walking	0.61	0.97	0.75	85022
Jogging	0.85	0.88	0.87	68489
Sitting	0.99	0.90	0.94	11911
Standing	1.00	0.00	0.00	9677
Upstairs	0.38	0.05	0.09	24626
Downstairs	0.37	0.02	0.04	19876
accuracy			0.70	219601
macro avg	0.70	0.47	0.45	219601
weighted avg	0.68	0.70	0.62	219601

Table 5.11: *Evaluation report for WISDM_v_1.1 dataset based on the features from autoencoders (the report is replicated in table form. Support refers to number of samples).*

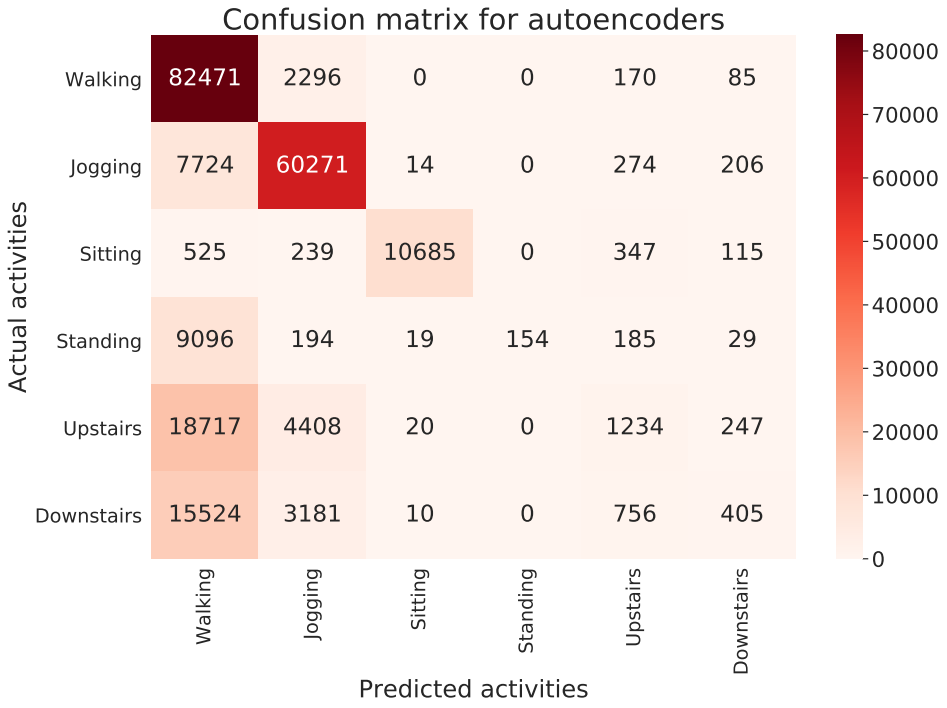

Figure 5.12: *Confusion matrix for WISDM_v_1.1 dataset based on the features from autoencoders.*

Table 5.11 provides the evaluation report based on extracted features from autoencoders for the WISDM_v_1.1 dataset. The autoencoder has not performed strongly on the

WISDM_v_1.1 dataset. The f1-score for the activities standing, downstairs, and upstairs is less than 10%, indicating a complete misclassification of these activities. According to Figure 5.12 classification of walking, jogging and sitting are better than the other three activities. The f1-score for the standing is zero since extensively it is predicted as walking. Besides, upstairs and downstairs are predicted as walking or jogging. The inadequate amount of sensors and dimensions made the autoencoder to underperform. The results indicate that 4 triaxial sensors at two different places are the minimum requirement for better performance. The feature-based methods performed well compared to the autoencoders. Hence, the feature-based methods are preferable for the dataset with few samples and fewer dimensions. The results for the WISDM dataset are provided below.

Accuracy score:	99.99			
	precision	**recall**	**f1-score**	**support**
Walking	1.00	1.00	1.00	7153
Jogging	1.00	1.00	1.00	7140
Climbing Stairs	1.00	1.00	1.00	7149
Sitting	1.00	1.00	1.00	7134
Standing	1.00	1.00	1.00	7080
Typing	1.00	0.98	0.99	7148
Brushing Teeth	1.00	1.00	1.00	7086
Eating Soup	1.00	1.00	1.00	7278
Eating Chips	1.00	1.00	1.00	7144
Eating Pasta	1.00	1.00	1.00	7185
Drinking from cup	1.00	1.00	1.00	7134
Eating Sandwich	1.00	1.00	1.00	7129
Kicking	1.00	1.00	1.00	6933
Playing Tennis	1.00	1.00	1.00	7104
Basketball	1.00	1.00	1.00	7137
Writing	0.98	1.00	0.99	7289
Clapping	1.00	1.00	1.00	7070
Folding Clothes	1.00	1.00	1.00	7227
accuracy			0.99	128520
macro avg	1.00	0.99	0.99	128520
weighted avg	0.99	0.99	0.99	128520

Table 5.12: *Evaluation report for WISDM dataset based on the features from autoencoders (the report is replicated in table form. Support refers to number of samples).*

From Table 5.12 it is obvious that the autoencoders' performance on the WISDM dataset is near accurate and higher than the other two datasets. The f1-score for the

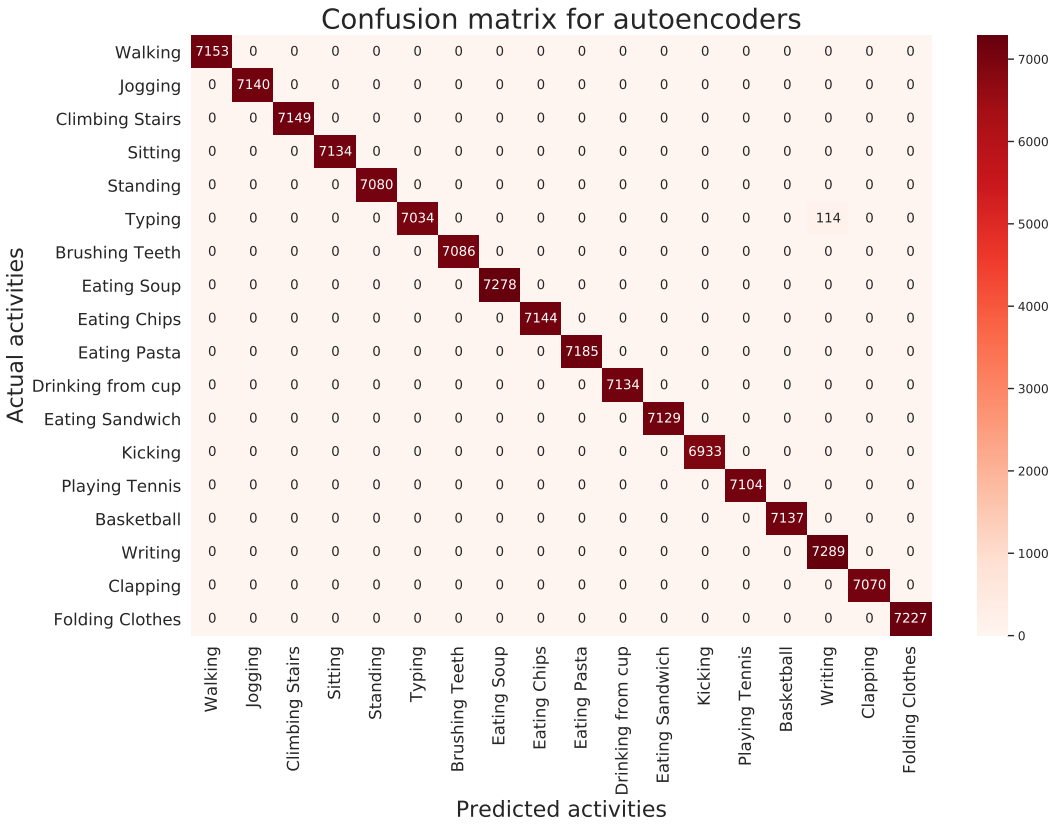

Figure 5.13: *Confusion matrix for WISDM dataset based on the features from autoencoders.*

activities typing and writing has recorded 99%. The same pattern was observed in the feature-based methods in a different set of activities. Figure 5.13 presents some of the typing activity is predicted as writing. However, other activities are classified correctly. As stated before, the dataset is quite discretely recorded and contains more or less the same amount of data for each activity compared to the other two datasets. The PCA visualization of the raw data and features extracted from autoencoders as well as feature-based methods are provided in Appendix E. Hence, the number of data for each activity is balanced; the autoencoders tend to learn efficiently. Overall, the autoencoders have performed well on the two datasets. The results enforce the following:

1. Required minimum 3 sensor placements, namely the wrist, waist, and leg.
2. Required more samples and dimensions, a minimum of 12 dimensions.
3. Balanced number of data in each activity will improve the results.

5.5 Comparison of Autoencoders with Feature-based Methods

The above two sections provided the evaluation of the feature-based methods and autoencoders individually. The performance entirely depends upon the number of uncorrelated features that the method generates. Figure 5.14 represents the low dimension features for the Motionsense dataset using t-Distributed Stochastic Neighbor Embedding (t-SNE), a non-linear dimension reduction method. Since the sensors are triaxial, it is sensible to reduce the dimensions to 3. The activities in Figure 5.14 are closely integrated. Figure 5.15 illustrates the t-SNE plot for the features from the feature-based methods. The activities are clustered properly without significant integration like the raw data. Whereas, Figure 5.16 depicts the activities in different clusters, yet there is no noticeable area between each cluster. These clusters appear to be bounded at the low dimension, which is the reason the autoencoders required a higher latent space. Following are the inference based on the t-SNE visualization:

1. **Raw data :** The activities are tightly bonded, which indicates that the activities are continuous and not linearly separable. Activities walking, jogging, upstairs as well as downstairs are firmly constrained at the core of the total data. These are similar activities where the contribution of legs is higher to the hands. Whereas sitting and standing are idle activities that encircle the preceding four activities.

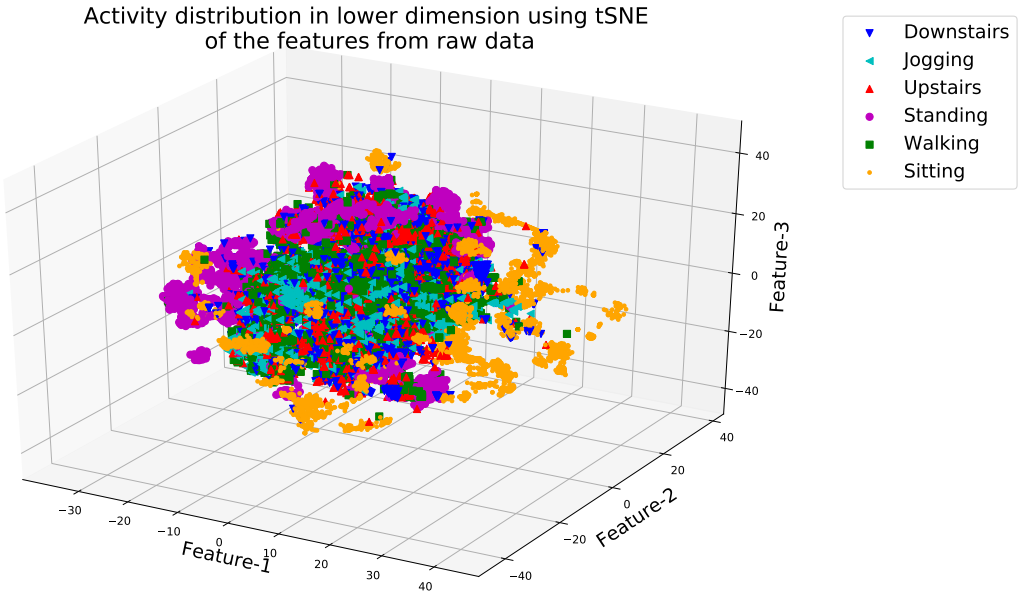

Figure 5.14: *tSNE plot for the Motionsense dataset*

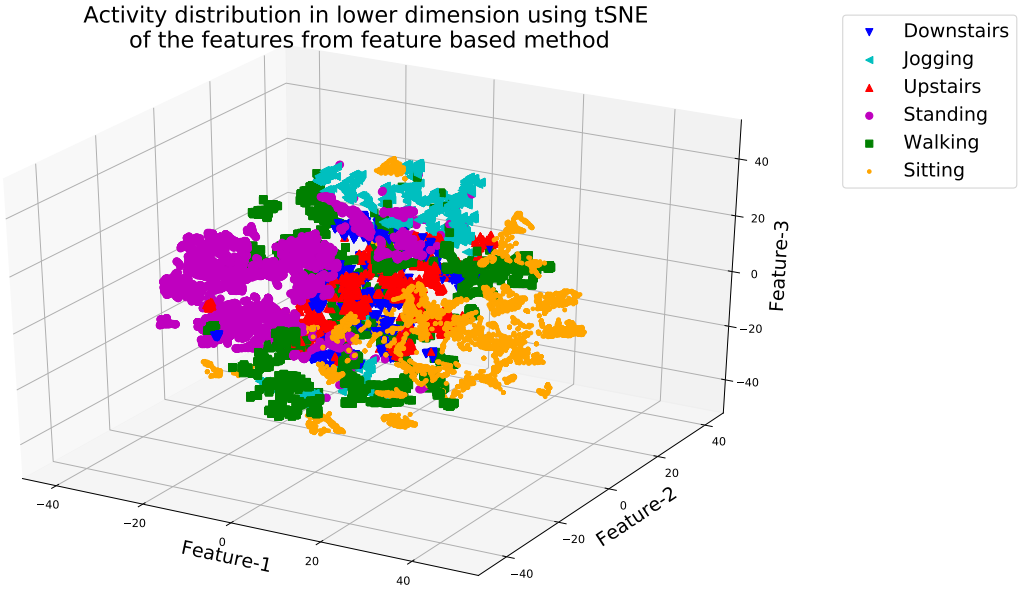

Figure 5.15: *tSNE plot for the features extracted by feature based method for Motionsense dataset*

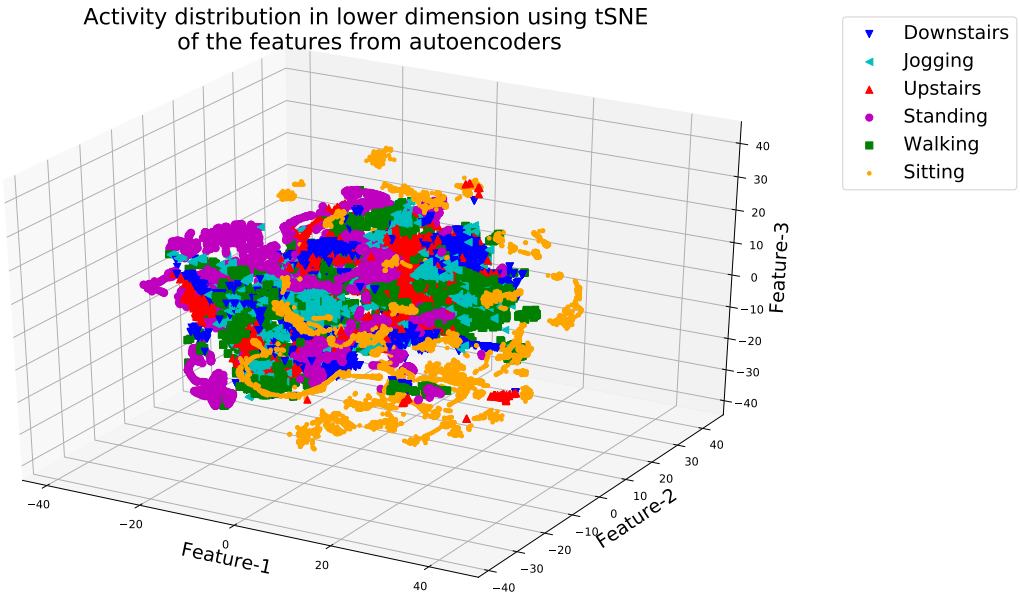

Figure 5.16: *tSNE plot for the features extracted by autoencoders for Motionsense dataset*

2. **Feature based methods:** Each activity is separated effectively. This indicates that the feature-based method can perform well even at the low dimension.

3. **Autoencoders:** Activities are clustered notably, yet the diversity does not appeal at the lower dimensions. The autoencoders are not suitable for the low dimension data. Autoencoders require more dimensions to make data distinguishable.

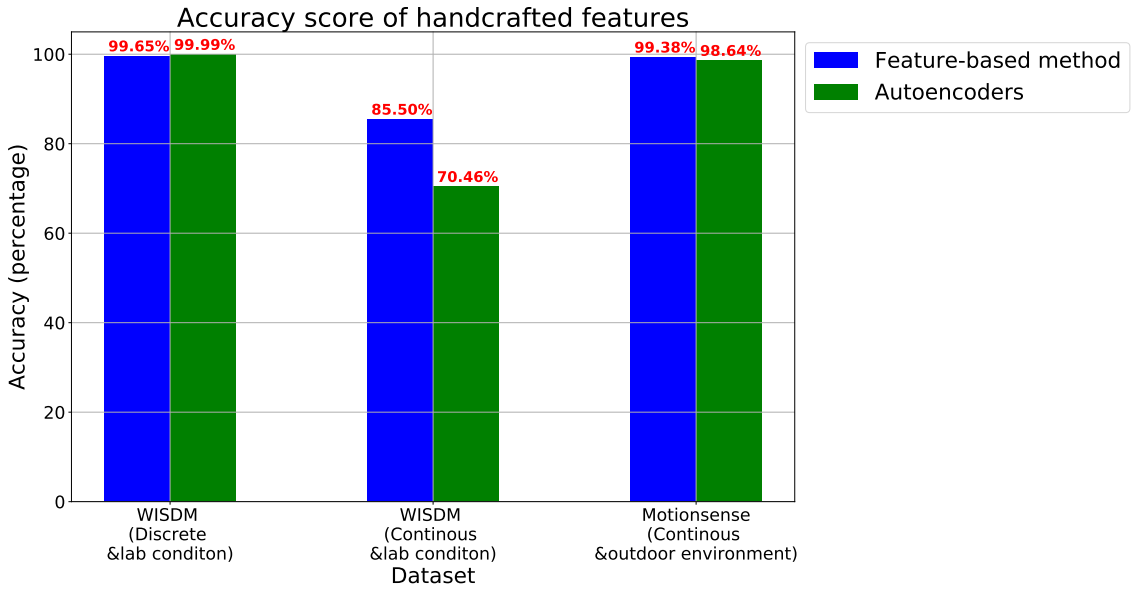

Figure 5.17: *Results for all the datasets*

Based on the performance of the methods with each dataset, the following are the inferences:

1. **Motionsense dataset:** Feature-based methods generated accuracy of 0.74% higher than the autoencoders. The feature-based methods results indicate that there should be an additional sensor to provide the appropriate distinction between upstairs and downstairs. However, the results of autoencoders marked a need for additional sensor placement. Both results are accountable for better performance. It is impossible to classify the activities with two sensor placements accurately, especially when there are more activities and an imbalanced number of samples in each activity.

2. **WISDM_v_1.1 dataset:** Both the methods underperformed compared to other datasets. This is due to the less number of sensors, which is evident compared to the previous results. The feature-based methods performed comparatively well. Hence, the feature-based methods are adequate for low dimension datasets.

3. **WISDM dataset:** Autoencoders performance exceeded the feature-based methods by 0.34%. The dataset is balanced among each activity, and the activities are not continuous. Hence, the autoencoders performed satisfactorily. Figure 5.17 illustrates the performance comparison.

The autoencoders equivalently performed to the feature-based methods provided the dimensions are at least 12 in the datasets. Autoencoders proposed in this research work use the discrete values without transforming into images or matrices or the other domain. The sliding window acted as the filter in the convolution layer. However, autoencoders require more amount of data. The great advantage of it is transfer learning. The trained model can be used for various time steps, unlike the feature-based method that has to be computed over. Thus, this research has proven that autoencoders can extract features from motion data without transforming the raw data. Besides, the autoencoders have the further advantage of supporting code reusability and processing the extensive datasets.

6

Conclusion

Motion data is difficult to interpret and adopt entirely. The sliding window aids in breaking the large series of discrete values into a sequence of several small discrete values chronologically. The sliding window is similar to the filters applied in the convolution layer of CNNs. The sliding window approach makes it effortless for any feature extraction method to extract features from the data. The autoencoder with LSTM layer is used to extract features from motion data in this research work, inspired by NLP techniques. The feature-based methods validate the autoencoder. The autoencoder has performed competitively, sometimes exceeding the performance of the feature-based methods. The performance of the methods is compared based on the classification task performed by SVM on the extracted features. Thus, the autoencoder can also be used to extract features from any time-series data as the motion data is a collection of time-series data. Besides, it aids code reusability and processing of extensive datasets.

This section enumerates the contributions, lessons learned, limitations, challenges and future directions of this research work.

6.1 Contributions

The contributions of this research work are:

1. **Literature review:** A detailed literature review is performed not only on the methods for feature extraction but also to understand the time-series data. The information gathered to support the analysis of motion data is presented in Chapter 2.

2. **Dataset analysis:** Due to time constraints, datasets are not generated as a part of this work. The publicly available datasets are used in this research. However, the research work has analyzed the attributes of the datasets and described their influence on the results compared to the other datasets.

3. **Importance of statistical features:** Initially, this research work decided to implement only FFT. To improvise the results, statistical features are included, which made to understand the importance of it. The statistical features' significance has been analyzed and presented in Section 5.3. Besides, the influence of each feature in the feature-based methods has been discussed that could pave the way for determining the features wisely.

4. **Automatic feature extraction:** Autoencoders with LSTM layers are used in this research to extract features automatically from the motion data. The process is not any more time consuming as the features does not want to be handcrafted or selected in a trial and error method. Also, the time-series data does not require to be converted into an image format. This approach can be adopted for any multivariate time-series data or motion data for feature extraction.

5. **Model for transfer learning:** The autoencoder model used in this research work can be used for transfer learning on a small motion dataset. Transfer learning does not require much domain knowledge since the model is trained. However, fine-tuning has to be done based on the new dataset.

6. **Method comparison:** A detailed comparison is provided between feature-based methods and autoencoders, which will enlighten the other researchers' knowledge.

6.2 Lessons Learned

The major part of the research work is learning and understanding. The lessons learned are summarised as follows:

1. **Understanding the time-series data:** This is the basis of this research work. Since the motion data is a collection of time-series data, it is necessary to understand the time-series data, their types, pattern, challenges as well as the process involved.

2. **Dataset preprocessing:** Preprocessing not only includes normalizing, extracting proper value/datatype, and organizing in a particular order. Besides, checking for unique or deviating values (out of normalized range), missing values as well as null values. The WISDM_v_1.1 contains one null value in a particular class. After removing the null value, the accuracy increased by 1%.

3. **Feature selection:** As feature extraction, feature selection is also important [32]. The influential features can be selected after intense analysis, as stated in Section 5.3.2. The proper analysis helped in dismissing the entropy feature that degraded the results.

4. **Hyperparameter tuning:** Hyperparameter tuning is the most sensitive and influential part of deep learning. While planning the research work timeline, there should be a dedicated time frame for hyperparameter tuning. This research work invested a good amount of time in hyperparameter tuning, which was not planned prior.

6.3 Limitations

This research work has the following drawbacks:

1. This research work aims to process different continuous activities in a real-time scenario like cycling, playing football, climbing a mountain, and other similar activities. In contrast, it was possible only for six basic activities from the Motionsense dataset.

2. Autoencoders are data-hungry [52], it needs more data to process. This induces computational load and larger hardware resources.

3. Features extracted by the autoencoders are not descriptive. Feature understanding has to be done to know the attributes of the raw data, which the autoencoders have learned.

4. The autoencoder model is generated in the Keras framework. The PyTorch framework could have been even better for debugging and analyzing the features extracted by the autoencoder [42]. However, Keras is extensively used compared to PyTorch [42].

6.4 Challenges

The followings are the challenges faced during this research work:

1. **Dataset:** This research work required a dataset consists of readings from a minimum of four triaxial sensors recorded in the outdoor environment for HAR. There are a good amount of datasets related to HAR [47] but the followings are the difficulties encountered:

(a) Datasets consists of readings only from the triaxial accelerometer.

(b) Datasets consists of the required amount of sensor readings, yet it is recorded either in the lab or indoor environment.

(c) Datasets consists of the required amount of sensor readings recorded in the outdoor condition, yet the number of samples or activities are less to train the network.

Few datasets meet the requirement; one such dataset from London Queen Mary University is being utilized [94]. Data from smartwatches like Fitbit, Apple is not publicly available.

2. **Benchmark:** Benchmark dataset [19][47] is available for HAR. However, the number of samples is inadequate to train the network [19]. Benchmarking is essential in order to validate the working of the proposed method. The dataset adopted in this research work is not a benchmark dataset. To address this concern, the features are handcrafted using the Fourier transform and a few statistical parameters. These features are regarded as a benchmark for the respective dataset to the automatically extracted features.

3. **Computational resource:** The datasets are huge, nearly one gigabyte consisting of 1,000,000 sample points for each axis (12 axes in this research work). The project requires a computational resource equivalent and above to the Nvidia GeForce GTX 1650 graphics card with 4GB of video Random Access Memory (RAM) for extracting the features.

6.5 Future Works

Following are the future work related to this research:

1. **Dataset creation:** The dataset will include activities described in the WISDM dataset and recorded continuously without any stoppage between the activities as in the Motionsense dataset. A triaxial accelerometer and gyroscope placements at four place in the body such as waist, wrist, leg and chest, as described in the research work [19]. These sensor placements provide more information about any activities and the type of ground it was performed, such as grass, water, wood, and terrain. Besides, this is to be made public so that the researchers can contribute.

2. **Benchmark:** There should be a benchmark with the motion data. So the existing motion data should be tested with all the SOTA described in Chapter 3 to validate the further methods. This research work has implemented a few from the feature-based methods, which should be extended.

3. **Multiple sliding window:** According to the research work [7], the multiple sliding window aids in better prediction of the future time data. The research work [71] uses multiple filters in the convolution layer for better classification. Hence, multiple sliding windows can be used for better change point detection, activity transition, future activity prediction, and improved results [153].

4. **Feature understanding for autoencoders:** The research work [132] has explored a few possible ways to interpret the features extracted by the deep learning methods. This work can be incorporated and extended for a better understanding of the features extracted by the autoencoders. Feature understanding provides better clarity on the features used for further tasks instead of treating it as a black box.

5. **Hyperparameter optimization:** In this research work, the hyperparameters are optimized manually based on the learning curve [30] and in comparison with the results of feature-based methods. These parameters are likely or unlikely to be optimal since the optimization is terminated once the required results are obtained. Also, these parameters are achieved manually, which is time-consuming. Hence, there should be an automatic optimization method to fine-tune the parameters that consume less time. Methods available for hyperparameter optimization are Bayesian Optimization (BO) [17], Gaussian Process (GP) [40], and Genetic Algorithm (GA) [58] can be adopted.

6. **LSTM-Fully Convolutional Networks (LSTM-FCNs):** According to the research work [69] the LSTM-FCNs are superior for the time-series classification. This method combines regular CNN and LSTM, where the LSTM converts univariate time-series to multivariate using a shuffling technique. Implementing this method can work well for a low dimensional dataset like WISDM_v_1.1.

A

Overfitting

Overfitting is the condition where the ML model performs well in the training data compared to the test data [87]. Overfitting occurs when the model fails to generalize the new data. Figure A.1 illustrates the overfitting due to model complexity. The model complexity applies to the more number of layers, more nodes at each layer, and higher epoch value. The model complexity should be equivalent to the number of samples. In the case of feature-based methods, more number features causes overfitting. The following methods can reduce overfitting [9]:

1. More number of samples with diversity
2. Dropout, ignoring certain node at different layers
3. Data augmentation in case of image datasets.

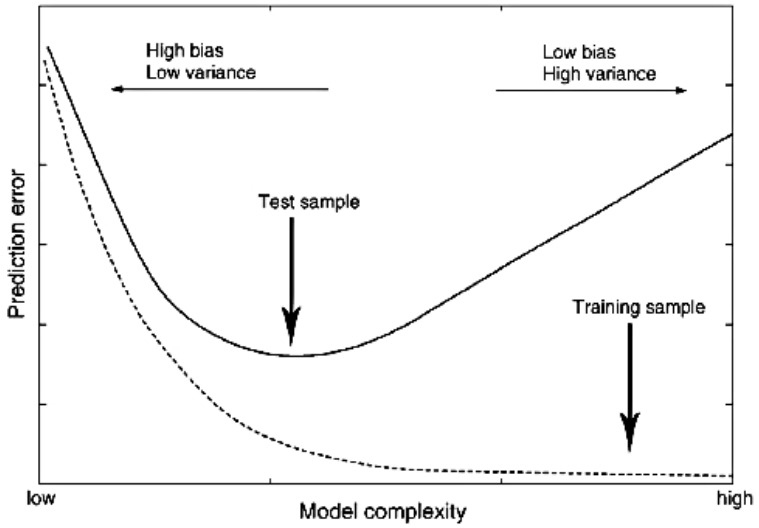

Figure A.1: *Overfitting. Reproduced from [9, p. 342].*

B

Fast Fourier Transform (FFT) Decomposition

FFT projects the data from the time domain to the frequency domain. The time domain data is represented by a series of functions (cosine and sine) in the frequency domain [18]. Summing up these functions reconstruct the data in the time domain. These functions are the decomposition of the original data, which is the key idea of FFT as illustrated in Figure 4.1. The number of decomposition is proportional to the accuracy of reconstruction of the original data, as illustrated in the graph below. Figure B.1 represents decomposition using 64 functions. When summed, it can reconstruct the original structure of the data, yet could not model the irregularities. As the decomposition is increased to 2048, as shown in Figure B.2, it perfectly reconstructed the original data with irregularities. In contrast, higher decompositions increase the computational load. Besides, it provides redundant features for the classification task as it compresses eventually to a very lower range. The higher decomposition is better suited for the forecasting application [77].

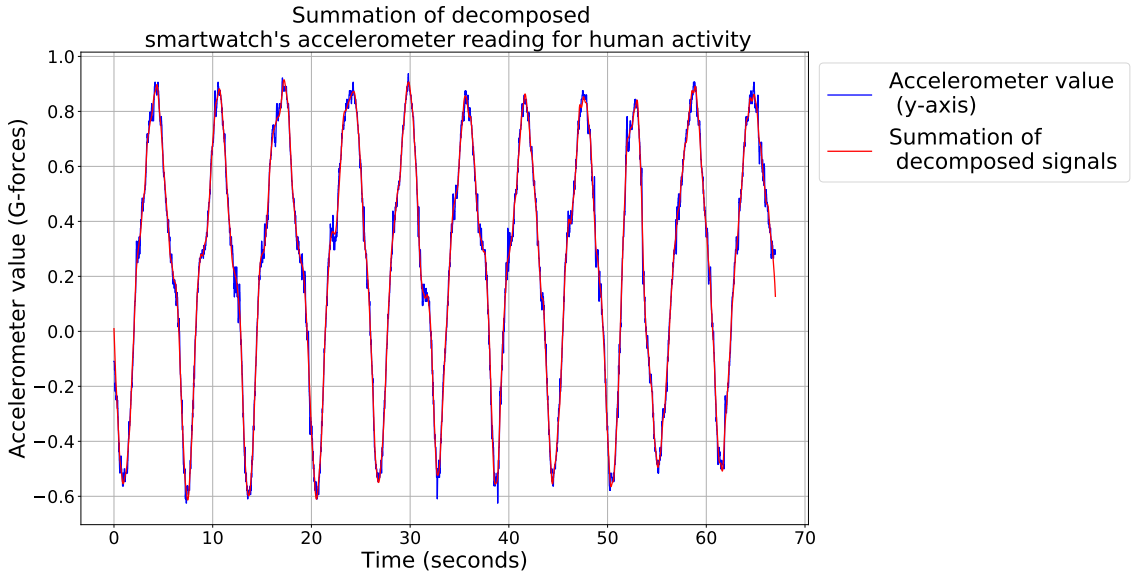

Figure B.1: *Summation of the decomposed data-64 (modelled the structure of the raw data yet could not capture the irregularities).*

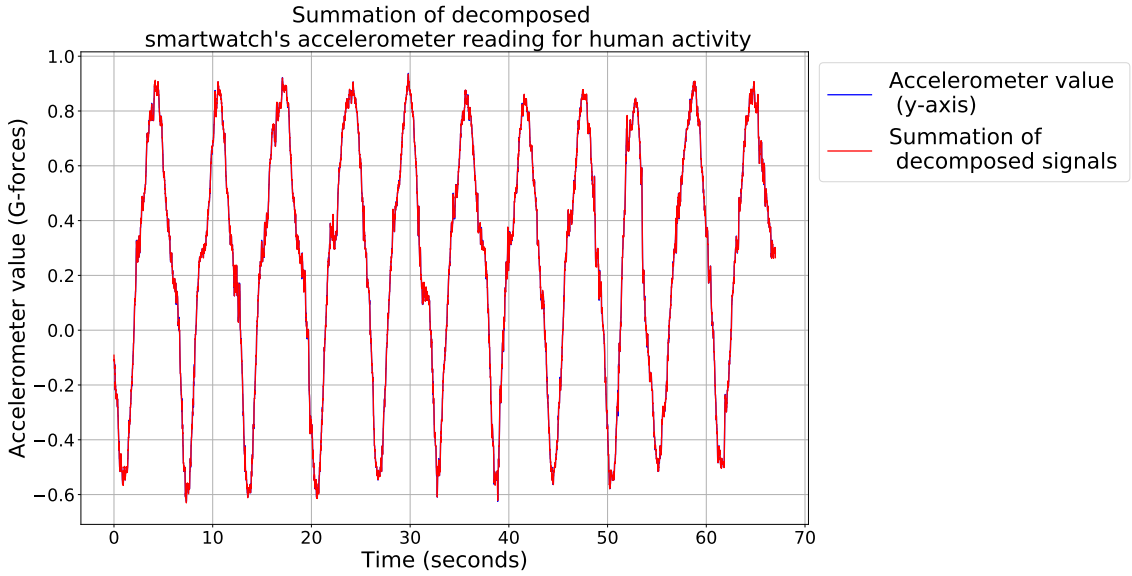

Figure B.2: *Summation of the decomposed data-2048 (accurately modelled the irregularities with the structure of the raw data).*

C

Features

The dataset used in this research work consists of a triaxial accelerometer and gyroscope from two devices such as smartphones and a smartwatch, as demonstrated in Figure C.1.

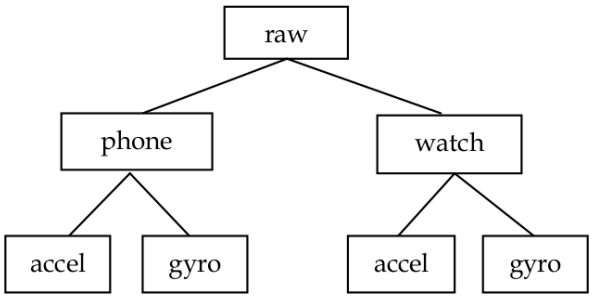

Figure C.1: *Dataset general structure. Reporduced from [140, p. 1].*

Dataset	**Device**	**Features**	**Dimensions**
Motionsense	Smartphone	1. Accelerometer - X, Y, Z 2. Gyroscope - X, Y, Z	12
	Smartwatch	1. Accelerometer - X, Y, Z 2. Gyroscope - X, Y, Z	
WISDM_v_1.1	Smartphone	1. Accelerometer - X, Y, Z 2. Gyroscope - X, Y, Z	6
WISDM	Smartphone	1. Accelerometer - X, Y, Z 2. Gyroscope - X, Y, Z	12
	Smartwatch	1. Accelerometer - X, Y, Z 2. Gyroscope - X, Y, Z	

Table C.1: *Features of the datasets (raw data).*

So the initial dimension is 12 according to the Equation 2.9. The handcrafted features are computed as a vector for each sensor as below

1. Mean - X, Y, Z
2. Variance - X, Y, Z
3. Standard deviation - X, Y, Z
4. Correlation XY, YZ, ZX
5. Max peak - X, Y, Z
6. Min peak - X, Y, Z
7. Entropy - X, Y, Z
8. Median absolute deviation - X, Y, Z
9. FFT - Amplitude (X, Y, Z), Frequency (X, Y, Z) Phase (X, Y, Z)

According to the Equation 2.21 it adds up to 132 dimension. The entropy feature is excluded, the remaining 7 features contribute to 82 added with the FFT which contributes to 48 (first two amplitude and corresponding frequency).

D

Software Prerequisites and Usage

To replicate this research work following software packages are necessary

1. Python3 >= 3.5.X
2. Numpy >= 1.19.X
3. Pandas >= 1.1.5
4. Seaborn >= 0.11.1
5. Pickle >= 5.0.0
6. Glob2 >= 0.7.X
7. Matplotlib >= 3.3.2
8. Scipy >= 1.5.2
9. Sklearn/ Scikit-learn >= 0.23.2
10. Tensorflow_gpu >= 2.3.X
11. Keras >= 2.4.3
12. Jupyter Notebook >= 5.4.1
13. Anaconda >= 3.0.0 (optional)

It is advisable to use Anaconda that makes a constraint environment. The script for this research work is written in Python following the Python3 format. Libraries like Numpy, Scipy are used for computation and feature extraction in the feature-based methods. Keras with TensorFlow framework is used to build the deep learning model and training. Seaborn provides better visualization along with Matplotlib. The feature

array is large (219601, 180), so it should be stored in a file for visualization and analysis. Pickle helps to store the data in a .pkl format with less consumption of memory. Glob2 reads all the file names in the specified directory and format. Pandas are utilized to read the data from the files stored by Glob2. Scikit learn is used to provide a classification report, train the SVM model as well as to divide the features for training and testing. The code and trained model can be downloaded using the following Github command

$ git clone git@github.com:Ganesamanian/Feature-extraction-for-motion-data-.git

The repository includes

1. Dataset directory contains the three datasets.
2. Script directory contains the script files to execute the code and load file to run the saved model.
3. Model directory contains the saved models.
4. Result directory contains experiment results and .pkl files.

Check the machine configuration before executing the code; further instructions are provided in the Github repository itself.

E

Visualization Using PCA

PCA is a popularly used dimension reduction method [51] but a linear transform. The features can contain non-linear relations; in this situation, the PCA is not suitable. The accuracy of the WISDM dataset is very close to 100%. To support the results, the PCA plot is provided below, which shows the clustering of activities.

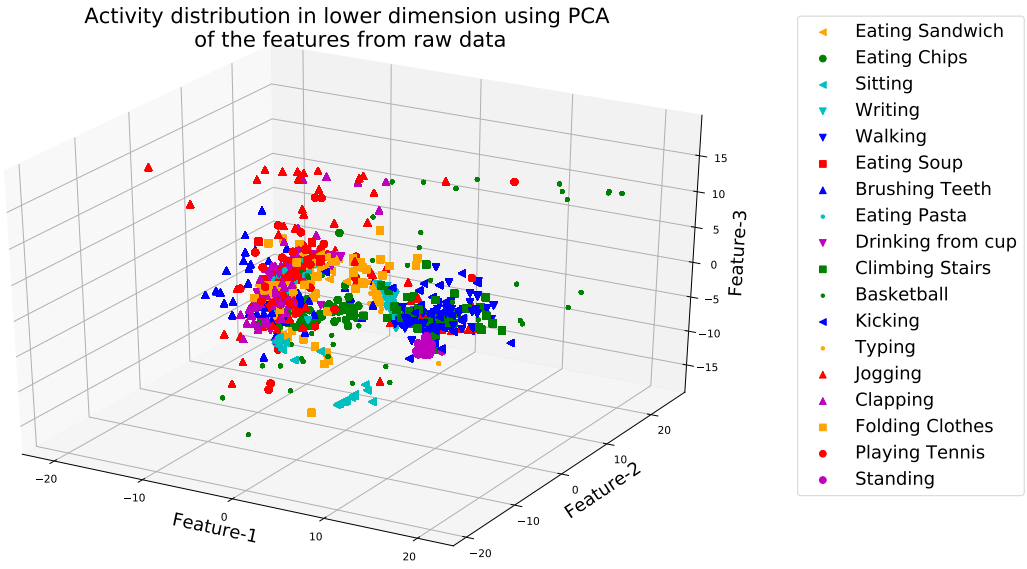

Figure E.1: *PCA plot for the raw data (WISDM).*

Figure E.1 represents the raw data of the WISDM dataset, which is not bonded tightly as in the Motionsense dataset represented in Figure 5.14. Figure E.2 represents the features spread out diversely by the feature-based method, yet Figure E.3 depicts appropriate classification between the activities. Hence the results are high compared to the other datasets.

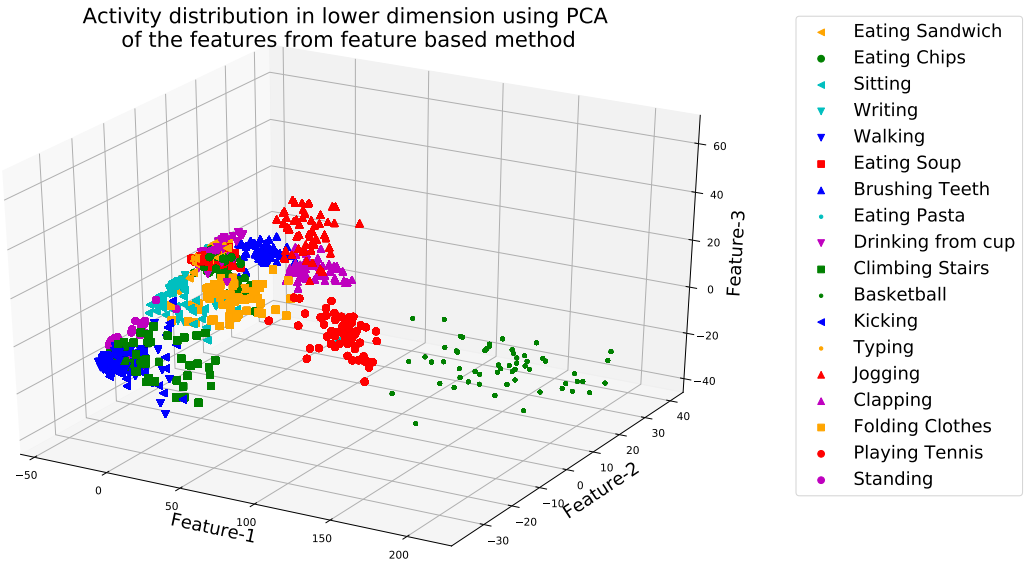

Figure E.2: *PCA plot for the feature-based method (WISDM).*

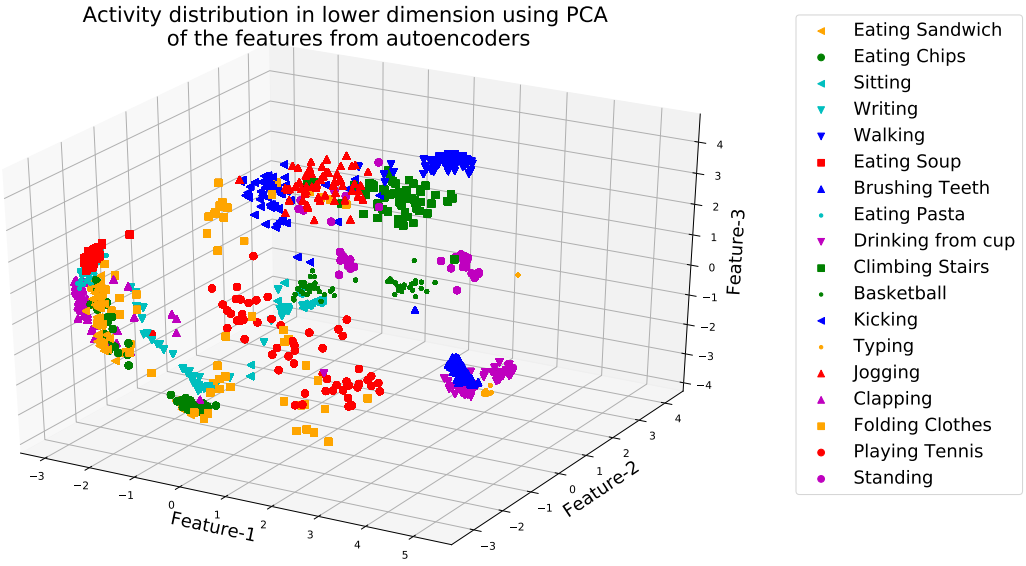

Figure E.3: *PCA plot for the autoencoders (WISDM).*

References

[1] I. A. S. Abu Amra and A. Y. A. Maghari. Forecasting Groundwater Production and Rain Amounts Using ARIMA-Hybrid ARIMA: Case Study of Deir El-Balah City in GAZA. In *2018 International Conference on Promising Electronic Technologies (ICPET)*, pages 135–140, 2018. doi: 10.1109/ICPET.2018.00031.

[2] Y. S. Afrianti, S. W. Indratno, and U. S. Pasaribu. Imputation algorithm based on copula for missing value in timeseries data. In *2014 2nd International Conference on Technology, Informatics, Management, Engineering Environment*, pages 252–257, 2014. doi: 10.1109/TIME-E.2014.7011627.

[3] A. N. Akansu, P. Duhamel, Xueming Lin, and M. de Courville. Orthogonal transmultiplexers in communication: a review. *IEEE Transactions on Signal Processing*, 46(4):979–995, 1998. doi: 10.1109/78.668551.

[4] Ali Akansu, Wouter Serdijn, and Ivan Selesnick. Full length article: Emerging applications of wavelets: A review. *Physical Communication*, 3:1–18, 03 2010. doi: 10.1016/j.phycom.2009.07.001.

[5] M. Al-Sharrah, A. Salman, and I. Ahmad. Watch Your Smartwatch. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*, pages 1–5, 2018. doi: 10.1109/ICCSE1.2018.8374228.

[6] Fabrizio Albertetti, Lionel Grossrieder, Olivier Ribaux, and Kilian Stoffel. Change points detection in crime-related time series: An on-line fuzzy approach based on a shape space representation. *Applied Soft Computing*, 40:441 – 454, 2016. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2015.12.004.

[7] S. Baek, D. Lee, G. Kim, and H. Bae. Summarization Method for Multiple Sliding Window Aggregate Queries. In *2009 Software Technologies for Future Dependable Distributed Systems*, pages 205–209, 2009. doi: 10.1109/STFSSD.2009.36.

[8] Shuja-ur-Rehman Baig, Waheed Iqbal, Josep Berral, Abdelkarim Erradi, and David Carrera. Adaptive Prediction Models for Data Center Resources Utilization Estimation. *IEEE Transactions on Network and Service Management*, 16, 07 2019. doi: 10.1109/TNSM.2019.2932840.

[9] Cristiano Ballabio. Spatial prediction of soil properties in temperate mountain regions using support vector regression. *Geoderma*, 151:338–350, 07 2009. doi: 10.1016/j.geoderma.2009.04.022.

[10] K. Bandara, C. Bergmeir, and Slawek Smyl. Forecasting Across Time Series Databases using Long Short-Term Memory Networks on Groups of Similar Series. *ArXiv*, abs/1710.03222, 2017.

[11] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140:112896, 2020.

[12] W. Bao-Jun and Z. Ying. A survey and performance evaluation on sliding window for data stream. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 654–657, 2011. doi: 10.1109/ICCSN.2011.6014977.

[13] F. Bashir and H. Wei. Handling Missing Data in Multivariate Time Series Using a Vector Autoregressive Model Based Imputation (VAR-IM) Algorithm. Part II: VAR-IM Algorithm Versus Modern Methods. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 459–463, 2016. doi: 10.1109/CSE-EUC-DCABES.2016.224.

[14] I. Batal and M. Hauskrecht. A Supervised Time Series Feature Extraction Technique Using DCT and DWT. In *2009 International Conference on Machine Learning and Applications*, pages 735–739, 2009. doi: 10.1109/ICMLA.2009.13.

[15] Z. Benbahria, I. Sebari, H. Hajji, and M. F. Smiej. Automatic Mapping of Irrigated Areas in Mediteranean Context Using Landsat 8 Time Series Images and Random Forest Algorithm. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 7986–7989, 2018. doi: 10.1109/IGARSS.2018.8517810.

[16] S. D. Bhattacharjee, W. J. Tolone, A. Mahabal, M. Elshambakey, I. Cho, A. a. Nayeem, J. Yuan, and G. Djorgovski. Multi-View, Generative, Transfer Learning for Distributed Time Series Classification. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5585–5594, 2019. doi: 10.1109/BigData47090.2019.9005452.

[17] R. J. Borgli, H. Kvale Stensland, M. A. Riegler, and P. Halvorsen. Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization. In *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, pages 1–6, 2019. doi: 10.1109/ISMICT.2019.8743779.

[18] E. Borgogno-Mondino and A. Lessio. A FFT-Based Approach to Explore Periodicity of Vines/Soil Properties in Vineyard from Time Series of Satellite-Derived Spectral Indices. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 9078–9081, 2018. doi: 10.1109/IGARSS.2018.8519437.

[19] Barbara Bruno, Fulvio Mastrogiovanni, and Antonio Sgorbissa. HOOD: a Real Environment Human Odometry Dataset for Wearable Sensor Placement Analysis. 10 2015. doi: 10.1109/IROS.2015.7354067.

[20] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. ISSN 1573-756X.

[21] N. Bölücü, D. Akgöl, and S. Tuç. Bidirectional LSTM-CNNs with Extended Features for Named Entity Recognition. In *2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT)*, pages 1–4, 2019. doi: 10.1109/EBBT.2019.8741631.

[22] C. Candemir and K. Oğuz. A Comparative Study on Parameter Selection and Outlier Removal for Change Point Detection in Time Series. In *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, pages 218–224, 2017. doi: 10.1109/EECS.2017.48.

[23] Zhaoyang Chai, H. Zhang, Xiong Xu, and L. Zhang. Garlic Mapping for Sentinel-2 Time-Series Data Using a Random Forest Classifier. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 7224–7227, 2019.

[24] C. Chang, Y. Wang, and S. Chen. Anomaly Detection Using Causal Sliding Windows. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(7):3260–3270, 2015. doi: 10.1109/JSTARS.2015.2422996.

[25] S. Chauhan and L. Vig. Anomaly detection in ECG time signals via deep long short-term memory networks. In *2015 IEEE International Conference on Data*

Science and Advanced Analytics (DSAA), pages 1–7, 2015. doi: 10.1109/DSAA.2015.7344872.

[26] C. Chen, W. Li, and D. Ma. Sleep Breathing Data Analysis Based on Econometrics. In *2019 6th International Conference on Systems and Informatics (ICSAI)*, pages 1413–1417, 2019. doi: 10.1109/ICSAI48974.2019.9010277.

[27] H. Chen, S. Feng, X. Pei, Z. Zhang, and D. Yao. Dangerous driving behavior recognition and prevention using an autoregressive time-series model. *Tsinghua Science and Technology*, 22(6):682–690, 2017. doi: 10.23919/TST.2017.8195350.

[28] C. Cheng, W. Xu, and J. Wang. A Comparison of Ensemble Methods in Financial Market Prediction. In *2012 Fifth International Joint Conference on Computational Sciences and Optimization*, pages 755–759, 2012. doi: 10.1109/CSO.2012.171.

[29] A. A. Chervova, G. F. Filaretov, and F. F. Pashchenko. A posteriori Fractal Characteristics Change Point Detecting Algorithm for Time Series. In *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–5, 2018. doi: 10.1109/ICAICT.2018.8747092.

[30] D. Choi, H. Cho, and W. Rhee. on the difficulty of dnn hyperparameter optimization using learning curve prediction. In *TENCON 2018 - 2018 IEEE Region 10 Conference*.

[31] M. Christ, A. Kempa-Liehr, and M. Feindt. Distributed and parallel time series feature extraction for industrial big data applications. *ArXiv*, abs/1610.07717, 2016.

[32] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72 – 77, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.03.067.

[33] Jiska Classen, Daniel Wegemer, Paul Patras, Tom Spink, and Matthias Hollick. Anatomy of a Vulnerable Fitness Tracking System: Dissecting the Fitbit Cloud, App, and Firmware. 2(1), 2018.

[34] Harvey Mudd College. Discrete Cosine Transform – E186 Handout, 2020. URL http://fourier.eng.hmc.edu/e161/lectures/dct/node1.html. Accessed on: 2020-12-12. [Online].

[35] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-Scale Convolutional Neural Networks for Time Series Classification, 2016.

[36] Towards data science. Demystifying Maths of SVM — Part 1, 2020. URL https://towardsdatascience.com/demystifying-maths-of-svm-13ccfe00091e. Accessed on: 2020-12-12. [Online].

[37] L. A. Demidova and M. A. Stepanov. Approach to the Analysis of the Multidimensional Time Series Based on the UMAP Algorithm in the Problems of the Complex Systems Proactive Maintenance. In *2020 International Conference on Information Technologies (InfoTech)*, pages 1–4, 2020. doi: 10.1109/InfoTech49733.2020.9211008.

[38] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[39] L. Deng and D. Yu. *Deep Learning: Methods and Applications*. 2014.

[40] F. Dernoncourt and J. Y. Lee. Optimizing neural network hyperparameters with Gaussian processes for dialog act classification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 406–413, 2016. doi: 10.1109/SLT.2016.7846296.

[41] Michele DIncecco, Stefano Squartini, and Mingjun Zhong. Transfer Learning for Non-Intrusive Load Monitoring, 2019.

[42] K. Dinghofer and F. Hartung. Analysis of Criteria for the Selection of Machine Learning Frameworks. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 373–377, 2020. doi: 10.1109/ICNC47757.2020.9049650.

[43] Y. Dong, J. Zhang, and J. M. Garibaldi. Neural networks and AdaBoost algorithm based ensemble models for enhanced forecasting of nonlinear time series. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 149–156, 2014. doi: 10.1109/IJCNN.2014.6889364.

[44] A. H. Fakhrulddin, X. Fei, and H. Li. Convolutional neural networks (CNN) based human fall detection on Body Sensor Networks (BSN) sensor data. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1461–1465, 2017. doi: 10.1109/ICSAI.2017.8248516.

[45] Spilios D Fassois and John S Sakellariou. Time-series methods for fault detection and identification in vibrating structures. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):411–448, 2007.

[46] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[47] UCI Machine Learning Repository Center for Machine Learning and Intelligent Systems. Time series dataset, 2020. URL https://archive.ics.uci.edu/ml/datasets.php?format=&task=&att=&area=comp&numAtt=10to100&numIns=&type=ts&sort=nameUp&view=table. Accessed on: 2020-11-26. [Online].

[48] B. D. Fulcher and N. S. Jones. Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12): 3026–3037, 2014. doi: 10.1109/TKDE.2014.2316504.

[49] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[50] Poulomi Ganguli, Devashish Kumar, and Auroop Ganguly. Water Stress on U.S. Power Production at Decadal Time Horizons. 09 2014.

[51] G. Gawde and J. Pawar. Shape based time series reduction using PCA. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–4, 2017. doi: 10.1109/ICIIECS.2017.8275897.

[52] A. Gensler, J. Henze, B. Sick, and N. Raabe. Deep Learning for solar power forecasting — An approach using AutoEncoder and LSTM Neural Networks. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002858–002865, 2016. doi: 10.1109/SMC.2016.7844673.

[53] M. Gheisari, G. Wang, and M. Z. A. Bhuiyan. A Survey on Deep Learning in Big Data. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 2, pages 173–180, 2017. doi: 10.1109/CSE-EUC.2017.215.

[54] Mark S Goldberg, Richard T Burnett, and David Stieb. A review of time-series studies used to evaluate the short-term effects of air pollution on human health. *Reviews on environmental health*, 18(4):269–303, 2003.

[55] C. Guo, L. Xu, H. Liu, L. Wang, X. Yu, and B. Han. The Financial Data of Anomaly Detection Research Based on Time Series. In *2015 International Conference on Computer Science and Applications (CSA)*, pages 86–89, 2015. doi: 10.1109/CSA.2015.42.

[56] A. Gupta and A. Kumar. Mid Term Daily Load Forecasting using ARIMA, Wavelet-ARIMA and Machine Learning. In *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–5, 2020. doi: 10.1109/EEEIC/ICPSEurope49358.2020.9160563.

[57] Hochschule Bonn-Rhein-Sieg (H-BRS). Platform for Scientific Computing at Bonn-Rhein-Sieg University, 2020. URL https://wr0.wr.inf.h-brs.de/wr/index.html. Accessed on: 2020-12-21. [Online].

[58] J. Han, D. Choi, S. Park, and S. Hong. Hyperparameter Optimization for Multi-Layer Data Input Using Genetic Algorithm. In *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 701–704, 2020. doi: 10.1109/ICIEA49774.2020.9101973.

[59] P. He, Y. Yuan, and G. Liu. Web Services Quality Prediction Based on Multivariate Time Series Analysis. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 881–884, 2018. doi: 10.1109/ICSESS.2018.8663771.

[60] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597.

[61] Elisabeth Hoppe, Gregor Körzdörfer, Tobias Würfl, Jens Wetzl, Felix Lugauer, Josef Pfeuffer, and Andreas Maier. Deep Learning for Magnetic Resonance Fingerprinting: A New Approach for Predicting Quantitative Parameter Values from Time Series. *Studies in health technology and informatics*, 243:202–206, 01 2017.

[62] Q. Hu, W. Wu, and M. A. Friedl. Mapping sub-pixel corn distribution using MODIS time-series data and a random forest regression model. In *2017 6th International Conference on Agro-Geoinformatics*, pages 1–5, 2017. doi: 10.1109/Agro-Geoinformatics.2017.8047051.

[63] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller. Transfer learning for time series classification. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1367–1376, 2018. doi: 10.1109/BigData.2018.8621990.

[64] Young-Seon Jeong, M. Jeong, and O. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognit.*, 44:2231–2240, 2011.

[65] Z. Jiang and K. Liu. Real time interpretation and optimization of time series data stream in big data. In *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 243–247, 2018. doi: 10.1109/ICCCBDA.2018.8386520.

[66] L. Jiao and J. Zhao. A Survey on the New Generation of Deep Learning in Image Processing. *IEEE Access*, 7:172231–172263, 2019. doi: 10.1109/ACCESS.2019.2956508.

[67] Jiayi Yao and Shuhui Kong. The application of stream data time-series pattern reliance mining in stock market analysis. In *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, volume 1, pages 159–163, 2008. doi: 10.1109/SOLI.2008.4686383.

[68] J. Kao and J. Jiang. Anomaly Detection for Univariate Time Series with Statistics and Deep Learning. In *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pages 404–407, 2019. doi: 10.1109/ECICE47484.2019.8942727.

[69] F. Karim, S. Majumdar, H. Darabi, and S. Chen. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*, 6:1662–1669, 2018. doi: 10.1109/ACCESS.2017.2779939.

[70] S. A. A. Karim, M. H. Kamarudin, B. A. Karim, M. K. Hasan, and J. Sulaiman. Wavelet Transform and Fast Fourier Transform for signal compression: A comparative study. In *2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, pages 280–285, 2011. doi: 10.1109/ICEDSA.2011.5959031.

[71] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff. ConvTimeNet: A Pre-trained Deep Convolutional Neural Network for Time Series Classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8852105.

[72] K. Kawagoe and T. Ueda. A similarity search method of time series data with combination of Fourier and wavelet transforms. In *Proceedings Ninth International Symposium on Temporal Representation and Reasoning*, pages 86–92, 2002. doi: 10.1109/TIME.2002.1027480.

[73] R. Khan, M. Abbas, R. Anjum, F. Waheed, S. Ahmed, and F. Bangash. Evaluating Machine Learning Techniques on Human Activity Recognition Using Accelerometer Data. In *2020 International Conference on UK-China Emerging Technologies (UCET)*, pages 1–6, 2020. doi: 10.1109/UCET51115.2020.9205376.

[74] S. Kihara, Y. Shimizu, N. Morikawa, and T. Hattori. An Improved Method of Sequential Probability Ratio Test for Change Point Detection in Time Series. In *2011 International Conference on Biometrics and Kansei Engineering*, pages 43–48, 2011. doi: 10.1109/ICBAKE.2011.48.

[75] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] Kiyoung Yang and Cyrus Shahabi. On the stationarity of multivariate time series for correlation-based data analysis. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, 2005. doi: 10.1109/ICDM.2005.109.

[77] G. Ganesh Kumar, Subhendu K. Sahoo, and Pramod Kumar Meher. 50 Years of FFT Algorithms and Applications. *Circuits, Systems, and Signal Processing*, 38 (12):5665–5698, 2019. ISSN 1531-5878.

[78] T. Lampert, B. Lafabregue, and P. Gançarski. Constrained Distance based K-Means Clustering for Satellite Image Time-Series. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 2419–2422, 2019. doi: 10.1109/IGARSS.2019.8900147.

[79] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[80] Techie Me learn with fun. Maximum Element Sliding Window, 2020. URL http://techieme.in/maximum-element-sliding-window/. Accessed on: 2020-12-12. [Online].

[81] C. Lee, Y. Su, Y. Lin, and S. Lee. Time series forecasting based on weighted clustering. In *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)*, pages 421–425, 2017. doi: 10.1109/CIAPP.2017.8167252.

[82] L. Li, Y. Wu, Y. Ou, Q. Li, Y. Zhou, and D. Chen. Research on machine learning algorithms and feature extraction for time series. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2017. doi: 10.1109/PIMRC.2017.8292668.

[83] L. Li, S. Dai, and Z. Cao. Deep Long Short-term Memory (LSTM) Network with Sliding-window Approach in Urban Thermal Analysis. In *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, pages 222–227, 2019. doi: 10.1109/ICCChinaW.2019.8849965.

[84] X. Li, X. Long, G. Sun, G. Yang, and H. Li. Overdue Prediction of Bank Loans Based on LSTM-SVM. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1859–1863, 2018. doi: 10.1109/SmartWorld.2018.00312.

[85] Y. Li, Jessica Lin, and T. Oates. Visualizing Variable-Length Time Series Motifs. In *SDM*, 2012.

[86] Z. Li, S. Wu, C. Li, and Y. Zhang. Research on methods of filling missing data for multivariate time series. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 382–385, 2017. doi: 10.1109/ICBDA.2017.8078845.

[87] Zeju Li, Konstantinos Kamnitsas, and Ben Glocker. Overfitting of neural nets under class imbalance: Analysis and improvements for segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 402–410. Springer, 2019.

[88] X. Lian and L. Chen. Efficient Similarity Join over Multiple Stream Time Series. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1544–1558, 2009. doi: 10.1109/TKDE.2009.27.

[89] Z. Liang and M. A. Chapa-Martell. Combining Resampling and Machine Learning to Improve Sleep-Wake Detection of Fitbit Wristbands. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–3, 2019. doi: 10.1109/ICHI.2019.8904753.

[90] B. Liu, Y. Luo, J. Zhang, L. Gong, W. Jiang, and L. Ren. PS-InSAR time series analysis for measuring surface deformation before the L'Aquila earthquake. In *2010 IEEE International Geoscience and Remote Sensing Symposium*, pages 4604–4607, 2010. doi: 10.1109/IGARSS.2010.5650971.

[91] E. Lughofer. Self-adaptive forecast models in predictive maintenance systems. In *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 8–8, 2018. doi: 10.1109/SYNASC.2018.00013.

[92] R. Ma and R. Angryk. Distance and Density Clustering for Time Series Data. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 25–32, 2017. doi: 10.1109/ICDMW.2017.11.

[93] Malekzadeh. Github repository of the author of motionsense dataset, 2020. URL https://github.com/mmalekzadeh/motion-sense. Accessed on: 2020-12-21. [Online].

[94] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile Sensor Data Anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, 2019. ISBN 978-1-4503-6283-2.

[95] M. C. Mallika, G. V. Drisya, K. S. Anil Kumar, and K. Satheesh Kumar. Effect of White Noise on the Structural Properties of Networks Arose from Time Series. In *2018 International CET Conference on Control, Communication, and Computing (IC4)*, pages 301–305, 2018. doi: 10.1109/CETIC4.2018.8530919.

[96] T. Mantoro and F. Alfiah. Comparison methods of DCT, DWT and FFT techniques approach on lossy image compression. In *2017 International Conference on Computing, Engineering, and Design (ICCED)*, pages 1–4, 2017. doi: 10.1109/CED.2017.8308126.

[97] Wenji Mao. *Cultural Modeling for Behavior Analysis and Prediction*, pages 91–102. 12 2012. ISBN 9780123972002. doi: 10.1016/B978-0-12-397200-2.00008-7.

[98] A. Meiseles and L. Rokach. Source Model Selection for Deep Learning in the Time Series Domain. *IEEE Access*, 8:6190–6200, 2020. doi: 10.1109/ACCESS.2019.2963742.

[99] Kevin M. Mendez, David I. Broadhurst, and Stacey N. Reinke. The application of artificial neural networks in metabolomics: a historical perspective. *Metabolomics*, 15(11):142–, 2019. ISSN 1573-3890.

[100] Somvir Nain, Dixit Garg, and Sanjeev Kumar. Performance evaluation of the WEDM process of aeronautics super alloy. *Materials and Manufacturing Processes*, 33:1–16, 05 2018. doi: 10.1080/10426914.2018.1476761.

[101] C. Narendra Babu and B. Eswara Reddy. Predictive data mining on Average Global Temperature using variants of ARIMA models. In *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, pages 256–260, 2012.

[102] Uchenna Odi and Thomas Nguyen. Geological Facies Prediction Using Computed Tomography in a Machine Learning and Deep Learning Environment. 07 2018. doi: 10.15530/urtec-2018-2901881.

[103] Humans of data. powered by atlan, 2020. URL https://humansofdata.atlan.com/2016/11/visualizing-time-series-data/. Accessed on: 2020-11-26. [Online].

[104] Jorge Luis Reyes Ortiz. Activity Recognition Experiment Using Smartphone Sensors, 2020. URL https://www.youtube.com/watch?v=XOEN9W05_4A&feature=youtu.be&ab_channel=JorgeLuisReyesOrtiz. Accessed on: 2020-12-21. [Online].

[105] J. C. Palomares-Salas, J. J. G. de la Rosa, J. G. Ramiro, J. Melgar, A. Aguera, and A. Moreno. ARIMA vs. Neural networks for wind speed forecasting. In *2009 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 129–133, 2009. doi: 10.1109/CIMSA.2009.5069932.

[106] Y. Pan, Q. Xu, and Y. Li. Food Recipe Alternation and Generation with Natural Language Processing Techniques. In *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, pages 94–97, 2020. doi: 10.1109/ICDEW49219.2020.000-1.

[107] A. Qonita, A. G. Pertiwi, and T. Widiyaningtyas. Prediction of rupiah against US dollar by using ARIMA. In *2017 4th International Conference on Electrical*

Engineering, Computer Science and Informatics (EECSI), pages 1–5, 2017. doi: 10.1109/EECSI.2017.8239205.

[108] B. G. Quinn. Estimation of frequency, amplitude, and phase from the DFT of a time series. *IEEE Transactions on Signal Processing*, 45(3):814–817, 1997. doi: 10.1109/78.558515.

[109] P. Radha and R. Divya. Multiple time series clinical data with frequency measurement and feature selection. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pages 250–254, 2016. doi: 10.1109/ICACA.2016.7887960.

[110] T. Radivilova, L. Kirichenko, and B. Vitalii. Comparative analysis of machine learning classification of time series with fractal properties. In *2019 IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL)*, pages 557–560, 2019. doi: 10.1109/CAOL46282.2019.9019416.

[111] S. S. Ratakonda and S. Sasi. Seasonal Trend Analysis on Multi-Variate Time Series Data. In *2018 International Conference on Data Science and Engineering (ICDSE)*, pages 1–6, 2018. doi: 10.1109/ICDSE.2018.8527804.

[112] Kaggle Repository. Datasets, 2020. URL https://www.kaggle.com/datasets. Accessed on: 2020-11-26. [Online].

[113] UCI Machine Learning Repository. Center for Machine Learning and Intelligent Systems, 2020. URL https://archive.ics.uci.edu/ml/index.php. Accessed on: 2020-11-26. [Online].

[114] N. Rikatsih and A. A. Supianto. Classification of Posture Reconstruction with Univariate Time Series Data Type. In *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, pages 322–325, 2018. doi: 10.1109/SIET.2018.8693174.

[115] M. A. A. Rodrigues, H. N. Bendini, A. R. Soares, T. S. Körting, and L. M. G. Fonseca. Remote Sensing Image Time Series Metrics For Distinction Between Pasture And Croplands Using The Random Forest Classifier. In *2020 IEEE Latin American GRSS ISPRS Remote Sensing Conference (LAGIRS)*, pages 149–154, 2020. doi: 10.1109/LAGIRS48042.2020.9165671.

[116] A. Salekin, M. M. Rahman, and S. H. Chowdhury. Pattern matching in time series using combination of neural network and rule based approach. In *2012 7th*

International Conference on Electrical and Computer Engineering, pages 478–481, 2012. doi: 10.1109/ICECE.2012.6471591.

[117] H. B. Sandya, P. H. Kumar, and S. B. Patil. Feature extraction, classification and forecasting of time series signal using fuzzy and garch techniques. In *National Conference on Challenges in Research Technology in the Coming Decades (CRT 2013)*, pages 1–7, 2013. doi: 10.1049/cp.2013.2508.

[118] SaturnCloud. Time Series Forecasting of Covid-19 Data with FBProphet, 2020. URL https://www.saturncloud.io/s/timeseriesforecasting/. Accessed on: 2020-12-12. [Online].

[119] Isaac's science blog. Fourier transform, 2020. URL https://isaacscienceblog.com/2017/08/13/fourier-transform/. Accessed on: 2020-12-12. [Online].

[120] S. Sehgal, J. Sharma, and N. Chaudhary. Generating Image Captions based on Deep Learning and Natural language Processing. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pages 165–169, 2020. doi: 10.1109/ICRITO48877.2020.9197977.

[121] J. K. Sethi and M. Mittal. Analysis of Air Quality using Univariate and Multivariate Time Series Models. In *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 823–827, 2020. doi: 10.1109/Confluence47617.2020.9058303.

[122] D. Shasha and Yunyue Zhu. High performance data mining in time series: techniques and case studies. 2004.

[123] F. A. Shirazi and M. J. Mahjoob. Application of Discrete Wavelet Transform (DWT) in Combustion Failure Detection of IC Engines. In *2007 5th International Symposium on Image and Signal Processing and Analysis*, pages 482–486, 2007. doi: 10.1109/ISPA.2007.4383741.

[124] S. Shukla, P. Singh, N. Neopane, and Rishabh. Health Care Management System Using Time Series Analysis. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 63–69, 2019. doi: 10.1109/ISCON47742.2019.9036150.

[125] D. Sitaram, A. Dalwani, A. Narang, M. Das, and P. Auradkar. A Measure of Similarity of Time Series Containing Missing Data Using the Mahalanobis

Distance. In *2015 Second International Conference on Advances in Computing and Communication Engineering*, pages 622–627, 2015. doi: 10.1109/ICACCE.2015.14.

[126] K. Siwek and S. Osowski. Autoencoder versus pca in face recognition. In *2017 18th International Conference on Computational Problems of Electrical Engineering (CPEE)*, pages 1–4, 2017. doi: 10.1109/CPEE.2017.8093043.

[127] Y. Sun, J. Li, W. Wang, A. Plaza, and Z. Chen. Active learning based autoencoder for hyperspectral imagery classification. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 469–472, 2016. doi: 10.1109/IGARSS.2016.7729116.

[128] S. P. Susanti and F. N. Azizah. Imputation of missing value using dynamic Bayesian network for multivariate time series data. In *2017 International Conference on Data and Software Engineering (ICoDSE)*, pages 1–5, 2017. doi: 10.1109/ICODSE.2017.8285864.

[129] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[130] P. Tan, L. Lymburner, N. Mueller, F. Li, M. Thankappan, and A. Lewis. Applying machine learning methods and time series analysis to create a National Dynamic Land Cover Dataset for Australia. In *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, pages 4289–4292, 2013. doi: 10.1109/IGARSS.2013.6723782.

[131] IDC Technologies. Specialists in engineering courses trianing, 2020. URL http://hydropedia.blogspot.in/search/label/Cyclic. Accessed on: 2020-11-26. [Online].

[132] Duy Khoi Tran. Feature extraction for time series classification. Master's thesis, Hochschule Bonn-Rhein-Sieg, Germany, 2020.

[133] Michele A Trovero and Michael J Leonard. Time series feature extraction. In *SAS*, pages 2020–2018, 2018.

[134] Tutorial and example. Understanding different types of machine learning, 2020. URL https://www.tutorialandexample.com/understanding-different-types-of-machine-learning/. Accessed on: 2020-12-12. [Online].

[135] A. Z. Ud Din, Y. Ayaz, M. Hasan, J. Khan, and M. Salman. Bivariate Short-term Electric Power Forecasting using LSTM Network. In *2019 International Conference on Robotics and Automation in Industry (ICRAI)*, pages 1–8, 2019. doi: 10.1109/ICRAI47710.2019.8967378.

[136] Savvas Varsamopoulos, Koen Bertels, and Carmen Almudever. Designing neural network based decoders for surface codes. 11 2018.

[137] D. Vihanga, M. Barlow, E. Lakshika, and K. Kasmarik. Weekly Seasonal Player Population Patterns in Online Games: A Time Series Clustering Approach. In *2019 IEEE Conference on Games (CoG)*, pages 1–8, 2019. doi: 10.1109/CIG.2019.8848108.

[138] W. Wang, G. Lyu, Y. Shi, and X. Liang. Time Series Clustering Based on Dynamic Time Warping. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pages 487–490, 2018. doi: 10.1109/ICSESS.2018.8663857.

[139] Zhiguang Wang and Tim Oates. Imaging Time-Series to Improve Classification and Imputation, 2015.

[140] G. M. Weiss, K. Yoneda, and T. Hayajneh. Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living. *IEEE Access*, 7:133190–133202, 2019. doi: 10.1109/ACCESS.2019.2940729.

[141] Anjana Wijekoon, N. Wiratunga, and K. Cooper. MEx: Multi-modal Exercises Dataset for Human Activity Recognition. *ArXiv*, abs/1908.08992, 2019.

[142] Xiaoguo Wang and Yuejing Liu. ARIMA time series application to employment forecasting. In *2009 4th International Conference on Computer Science Education*, pages 1124–1127, 2009. doi: 10.1109/ICCSE.2009.5228480.

[143] Li Xiong, Yan-Jun Lu, Yong-Fang Zhang, Xin-Guo Zhang, and Parag Gupta. Design and Hardware Implementation of a New Chaotic Secure Communication Technique. *PloS one*, 11:e0158348, 08 2016. doi: 10.1371/journal.pone.0158348.

[144] J. Xu, Y. Qiu, H. Zhang, M. Li, and M. Li. Large-scale time series data down-sampling based on Map-Reduce programming mode. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 409–413, 2017. doi: 10.1109/IAEAC.2017.8054047.

[145] Jianbo Yang, M. Nguyen, P. P. San, X. Li, and S. Krishnaswamy. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition. In *IJCAI*, 2015.

[146] Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

[147] Yao Zhao, Lei Lin, Wei Lu, and Yu Meng. Landsat time series clustering under modified Dynamic Time Warping. In *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, pages 62–66, 2016. doi: 10.1109/EORSA.2016.7552767.

[148] J. Zhai, S. Zhang, J. Chen, and Q. He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018. doi: 10.1109/SMC.2018.00080.

[149] Hui Zhang, Tu Ho, Yang Zhang, and Song Lin. Unsupervised Feature Extraction for Time Series Clustering Using Orthogonal Wavelet Transform. *Informatica (Slovenia)*, 30:305–319, 10 2006.

[150] Y. Zhang and J. Pan. Assessment of photoplethysmogram signal quality based on frequency domain and time series parameters. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2017. doi: 10.1109/CISP-BMEI.2017.8302279.

[151] Y. Zhang, H. An, H. Ma, Q. Wei, and J. Wang. Human Activity Recognition with Discrete Cosine Transform in Lower Extremity Exoskeleton. In *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, pages 309–312, 2018. doi: 10.1109/IISR.2018.8535705.

[152] Z. Zhao, Y. Zhang, X. Zhu, and J. Zuo. Research on Time Series Anomaly Detection Algorithm and Application. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 1, pages 16–20, 2019. doi: 10.1109/IAEAC47372.2019.8997819.

[153] H. Zhu, Y. Wang, and Z. Yu. Clustering of Evolving Data Stream with Multiple Adaptive Sliding Window. In *2010 International Conference on Data Storage and Data Engineering*, pages 95–100, 2010. doi: 10.1109/DSDE.2010.63.

[154] Seyedjamal Zolhavarieh, Saeed Aghabozorgi, and Ying Wah Teh. A Review of Subsequence Time Series Clustering. *The Scientific World Journal*, 2014:312521–, 2014. ISSN 2356-6140.