ANOMALY DETECTION ALGORITHMS FOR HOME ACTIVITIES DATA

POH SOON CHANG

MASTER OF ENGINEERING SCIENCE

MULTIMEDIA UNIVERSITY

OCTOBER 2019

ANOMALY DETECTION ALGORITHMS FOR HOME ACTIVITIES DATA

BY

POH SOON CHANG

B.Eng. (Hons) Electronics, Multimedia University, Malaysia

THESIS SUBMITTED IN FULFILMENT OF THE

REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING SCIENCE

(by Research)

in the

Faculty of Engineering

MULTIMEDIA UNIVERSITY

MALAYSIA

October 2019

DECLARATION

I hereby declare that the work has been done by myself and no portion of the work contained in this Thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institution of learning.

_____

Poh Soon Chang

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Tan Yi Fei and my co-supervisor, Mr. Cheong Soon Nyean for their guidance, advice and considerable encouragements to complete this thesis.

Besides that, I would like to thank Dr. Ooi Chee Pun and Dr. Tan Wooi Haw for their support throughout my study and research. I am grateful to Dr. Guo Xiaoning for her help with my conference paper preparation.

Next, I would like to express my appreciation and gratitude to my family for their support and love throughout my study.

Finally, I would like to thank Telekom Malaysia Research and Development (TM R&D) as this work was supported by their research grant.

ABSTRACT

The population of elderly around the world is projected to have continuous growth. It is speculated that the number of solitude elderly is on the rise. Research finding shown that elderly living alone has a higher mortality rate. Thus, there is a need to continuously monitor elderly condition. Hiring caregivers is not affordable for some young adults due to its relatively expensive cost. Researchers proposed using activity recognition to monitor elderly daily routine. This activity recognition system generates historical dataset of home activities for the elderly subject. The health condition of a person is closely related to that person's life pattern. Changes in behaviour or home activities pattern may indicate illness. Therefore, anomaly detection on home activities data is important.

In view of detecting home activities anomaly detection, study on past related researches were carried out. It was learnt that there are two types of home activities anomalies. The first type is sequential anomaly which refers to out-of-sequence activities. For example, dementia patients forget and repeat activities that were carried out before. The second type of anomaly is time anomaly. Time anomaly refers to change in time routine of a person's home activities. For example, prolonged sleeping activity which was due to elderly contracted an illness. When conducting literature review, several sequential anomaly detection algorithms and time anomaly detection algorithms were investigated. Several strengths and weaknesses of these algorithms were identified. Therefore, there is a need to develop new anomaly detection methods to combat these weaknesses.

In this work, two sequential anomaly detection algorithms which are Long Short-Term Memory (LSTM) method and database method were designed to overcome the identified weaknesses. The sequential anomaly detection algorithm using LSTM was inspired and modified from language model to deal with the identified weaknesses. The experiment to evaluate these two algorithms was carried out using a publicly available dataset. The results demonstrated that LSTM method performs well on an unseen dataset with an accuracy of 85.53%. On the other hand,

the database method has an equivalently excellent test accuracy of 84.21%. Another experiment was carried out to compare performance of LSTM method and database method with language model and Hidden Markov Model (HMM). Both of these algorithms are better in detecting sequential anomalies when compared to language model and HMM method in terms of test accuracy.

Besides, a time anomaly detection algorithm using database was designed to detect time anomalies. One benefit of the algorithm is that it only needs to be trained once using entire sequences of different home activities collectively unlike the sequential algorithms in past works which train one or more model for each type of activity separately. The experimental results demonstrated that this algorithm performs well on test set with a high accuracy of 93.57%. This performance is comparable to combination method which used *K*-means clustering and Gaussian distribution in past work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Due to improved healthcare and lifestyle, the elderly population is on the rise. The United Nations predicted that the global population of elderly people with age of 60 years old and over will outnumber the youth in 2050 to reach nearly 2.1 billion (United Nations, 2017). In Malaysia, 20% of the whole population was predicted to be elderly by 2040 (Yuen, 2017). On the other hand, there is an increasing number of solitude elderly living alone (Yuen, 2017). Therefore, it is conjectured that there would be a significantly large population of solitude elderly in the future. Research findings shown that solitude elderly has a higher rate of mortality (Ng, et al., 2015). Without constant care and monitoring of their health conditions, solitude elderly's health may continue to deteriorate. Young adults can hire caregivers or sending elderly parents to nursing home. A downside of hiring caregivers or sending elderly parents to nursing home is the expensive cost especially for long-term care which is unaffordable for some family (Gurnon, 2017). In addition, hired caregivers just spend limited time with the elderly. Therefore, it is difficult to keep an eye on elderly all the time.

Researchers had suggested using activity recognition technology to monitor elderly's activities at home. There are two approaches to collect input data for activity recognition. The first approach involves using input data of readings of different types of sensors such as accelerometer and gyroscope (Bao & Intille, 2004; Huang, et al., 2016; Tian, et al., 2017; Kaytaran & Bayindir, 2018; Weiss, et al., 2019; Barshan & Yurtman, 2020). As for vision-based method, the input data are video collected using cameras (Bux, et al., 2016; Babiker, et al., 2018; Zhang, et al., 2017; Poppe, 2010; Ke, et al., 2013). Most past researches on activity recognition used machine learning algorithms for activity classification. Recently, deep learning algorithm become more commonly used for activity recognition (Chen & Xue, 2015; Friday, et al., 2018; Nweke, et al., 2018; Shi, et al., 2018; Ramanathan, et al., 2014). Irrespective of the

classification algorithms, activity recognition can be used to collect records of elderly home activities with timestamp. The dataset used in this research is a publicly available dataset released by Washington State University (Cook, 2010).

The health condition of a person is closely associated with his or her lifestyle. Behavioural changes in terms of pattern of home activities might be indicator of health decline. For example, increased toilet visit especially at night is a symptom of diabetes (Weiss & Blaivas, 2000). Studies also shown that nocturia is associated deteriorated health, drop of sleep quality and worsening of life quality (Asplund, 2002). This health condition can change home activities pattern. Anomaly detection algorithm trained using historical home activities data can be used to detect anomalous pattern of home activities.

1.2 Problem Statements

In this research, two main types of anomalies were identified which are sequential anomaly and time anomaly. Several anomaly detection algorithms were developed to detect these two types of anomalies in dataset of home activities. However, there are a few limitations with the dataset and anomaly detection algorithms. The main problem with dataset is the limitation of data size. For sequential anomaly detection algorithm, each data point is for a collection of activities in a day. In other words, a sequence of activities for a day is a single piece of data point. Machine learning requires to learn from a large enough dataset before it can be used. For example, to collect 1000 data points, a time period of 1000 days are needed. It does not make sense to collect data for years just to build an anomaly detection model.

Existing anomaly detection algorithms have some weaknesses or limitations. The weakness includes the need of human expertise for hyperparameter settings, poor performance due to limitation of the data size and hefty training procedure. Some of the algorithms in past research require human expertise for hyperparameter settings. For example, one of the hyperparameter is the number of clusters of *K*-means clustering algorithm denoted by *K*. In addition, the training procedure for all the algorithms is repeated for each type of activities separately (Hoque, et al., 2015;

Forkan, et al., 2015; Zhao, et al., 2014). This means that one or more model is trained for each type of activity.

1.3 Objectives

a) To develop sequential anomaly detection algorithm and time anomaly detection algorithm that can detect anomalies in home activities and overcome problems which include limitation of algorithm's accuracy due to small dataset and hefty training process of multiple models per activity type.

b) To compare the proposed algorithms with existing anomaly detection algorithms on a same publicly available benchmarking dataset called Aruba (Cook, 2010).

1.4 Contribution of the Research

In this work, several anomaly detection algorithms which do not require human expertise for hyperparameter settings, can be trained using tiny dataset and a simpler and more straightforward training procedure are presented. The contributions of this research work can be summarized as follows:

- Two algorithms for sequential anomaly detection algorithms were proposed namely Long Short-Term Memory (LSTM) method and database method. A rule of thumb is that learning algorithms require training set size around few thousands per class label (Goodfellow, et al., 2016). Therefore, these two algorithms were designed with the consideration that they can perform well even though the training set size is relatively small. In past research, Hidden Markov Model (HMM) for sequential anomaly detection was developed for sequential anomaly detection and evaluated on an artificial dataset (Forkan, et al., 2015). The HMM method was reproduced on for sequential anomaly detection using a real public dataset demonstrated average test accuracy of 50%. The experimental results demonstrated that the two algorithms which are

LSTM method and database method outperformed HMM on a public real-life dataset.

- An algorithm for time anomaly detection was proposed called database method. Some of the algorithms in past research require human expertise for hyperparameter settings. The proposed database method train only one model using entire sequences consisting different types of activities in one go instead of separately training one or more models for each type of home activity.

1.4 Thesis Organization

The thesis comprises of five chapters. The first chapter is on the introduction to the thesis. The rest of the thesis is organized into Chapter 2, Chapter 3, Chapter 4 and Chapter 5. Chapter 2 consists of the literature review. It touches on the home activities dataset, types of home activities anomalies and several related anomaly detection algorithms. In addition, several machine learning concepts such as neural network and language model which lead to design of a LSTM method for sequential anomaly detection are discussed.

Chapter 3 presents two alternative sequential anomaly detection algorithms that were developed to detect sequential anomalies in home activities. These two algorithms are LSTM method and database method. In this chapter, methodology for building good sequential anomaly detection model is presented. Then, several experiments were carried out to validate the methods. In addition, the performance comparison between both of these algorithms with HMM method and language model method are also presented.

Chapter 4 presents database method for time anomaly detection. The methodology and performance comparison of this algorithm with past work which used combination method of *K*-means clustering and Gaussian distribution for time anomaly detection are presented.

Chapter 5 presents the conclusion of this research work. It summarizes and discusses the research findings. Several recommendations for future work are also included in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

There are two types of home activities anomalies to be detected namely sequential anomaly and time anomaly. Details of the home activities datasets and the types of anomalies are discussed in section 2.2 and section 2.3 respectively. Section 2.4 presents the existing sequential anomaly detection algorithms. The strengths and weaknesses of these algorithms are also given. Section 2.5 presents the existing time anomaly detection algorithms. In addition, the strengths and weaknesses of each sequential anomaly detection algorithm are also presented. Finally, a discussion on neural network, recurrent neural network, LSTM and language model which inspired LSTM method are presented in section 2.6.

2.2 Datasets of Home Activities

Anomalies are anomalous data points which have different characteristics or patterns when compared to normal data points (Zimek & Schubert, 2017). This type of points is also known as outliers. Figure 2.1 shows an example of an anomalous data point for a dataset with two features or variables labelled with *x* and *y*. The anomalous data point or outliers locates far from the normal data points which were visualized as four pointed stars.

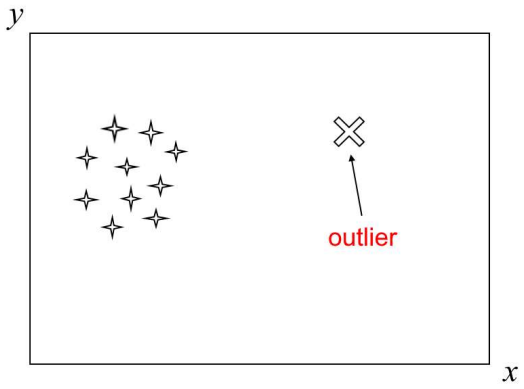

Figure 2.1: An Illustration Depicting an Outlier for a 2-dimensional Dataset

Dataset of home activities is historical record of a person's home activities. Figure 2.2 illustrates the Aruba dataset (Cook, 2010) which is an example of historical home activities dataset. Visually, a sequential pattern can be observed. For example, the sleeping activities (purple data points) for different days can be clustered into certain range of time. In addition, almost all sleeping activities (purple data points) are followed by meal preparation activities (dark blue data points).

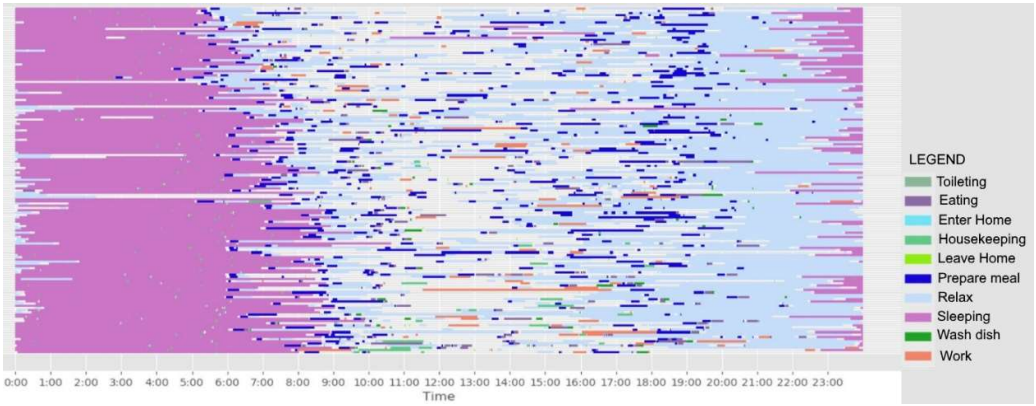

Figure 2.2: Visualization of Aruba Dataset

For anomaly detection task, there are two class labels namely normal and abnormal. If a given data instance is classified as normal, that means it has similar characteristics to usual pattern of home activities. Conversely if a data point is classified as abnormal, then that means it is an outlier. The normal data points can be used to build a model which characterizes normal behaviour of a person.

There are two main types of normal dataset. The first type of normal dataset is the ground truth dataset of an activity recognition research which is annotated with date, time and activity type (Cook, 2010; Hoque, et al., 2015). The second type of normal dataset used by researchers is artificially generated dataset (Forkan, et al., 2015).

For ground truth dataset of activity recognition, there are two data collection methods. The first data collection method involves a volunteer living in a smart home installed with sensors (Cook, 2010). The activity annotation is carried out by asking the volunteer to keep a time diary. The second type of data collection method is created

by using a script of home activities which is specified by the researchers where a graduate student will carry out home activities according to the script (Hoque, et al., 2015). Table 2.1 shows a list of publicly available normal home activities dataset which include Ordonez (Ordonez, et al., 2013), Kasteren (van Kasteren, et al., 2010) and Aruba (Cook, 2010). These datasets consist of historical records of home activities which can be used in activity recognition researches. In this research, the Aruba dataset with highest total number of days of 220 was used because it is the largest dataset.

Table 2.1: Publicly Available Normal Home Activities Dataset

Dataset name	Number of days	Number of activities
Ordonez A	14	10
Ordonez B	21	10
Kasteren A	25	13
Kasteren B	14	13
Kasteren C	19	16
Aruba	220	11

On the other hand, abnormal home activities datasets were artificially generated by introducing abnormalities into normal home activities dataset (Forkan, et al., 2015).

2.3 Types of Anomalies

It was found that there are two types of anomalies that can be detected in home activities data. They are sequential anomaly and time anomaly. Sequential anomaly is out-of-sequence or change in order of occurrence of activities (Forkan, et al., 2015; Hoque, et al., 2015). Detecting these sequential anomalies are essential as they might be linked to certain illnesses. For example, elderly with diabetes have nocturia symptom which means increased toileting activities during sleeping hours (Weiss & Blaivas, 2000).

Time anomaly refers to change in routine of an activity (Forkan, et al., 2015; Hoque, et al., 2015). For example, a person which usually takes lunch between 12 pm and 1 pm suddenly takes lunch at 2 pm.

2.4 Sequential Anomaly Detection Methods

Subsection 2.4.1 discusses Zhao's work (Zhao, et al., 2014) which uses Markov process to detect sequential anomalies in location sequence dataset. Subsection 2.4.2 discusses HMM and its implementation to detect sequential anomalies (Forkan, et al., 2015). Subsection 2.4.3 discusses data mining algorithms and their usage for sequential anomaly detection (Hoque, et al., 2015). Subsection 2.4.4 discusses strength and weakness of each algorithm for sequential anomaly detection on home activities data.

2.4.1 Discrete Markov Processes

Real-world processes such as coin tossing generate outputs which are observable and can be recorded (Rabiner L. , 1989). The outputs are physically observable events. These outputs are either discrete or continuous in nature. An example of discrete outputs are head and tail which are produced by tossing a coin. On the other hand, continuous output exists in the form of continuous signal such as audio signals. Home activities are discrete events. Hence, the focus is on discrete Markov processes.

In probability theory and statistics, the observable outputs are known as states (Gagniuc, 2017). For a discrete case, a real-world process generates sequences of states. A Markov process can statistically model a real-world process. Modelling a real-world process means learning the characteristics of that process. Since the process produce sequences of states, the most important piece of information about that process is the pattern of occurrences of consecutive states in the sequences. In other words, the sequential correlations between states. Conditional probability is a tool which can be used to quantify the sequential correlations between states.

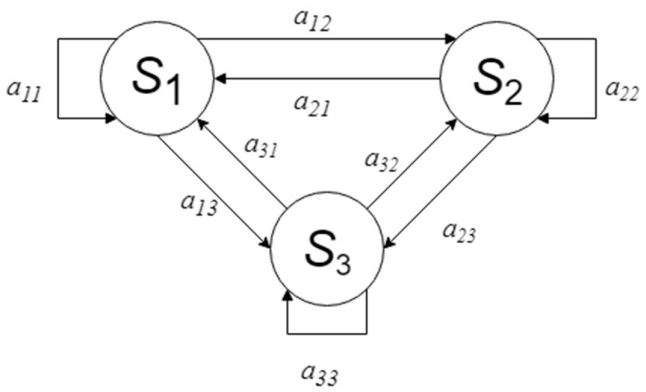

Figure 2.3: A Markov Process with Three States (S_1, S_2, S_3)

To formalize, the total number of distinct states is denoted by N. The N distinct states are denoted by S_1 , S_2 , ⋯, S_N (Revuz, 2008). For example, a Markov process with a set of three states as shown in Figure 2.3 has $N = 3$. For a discrete Markov process, the system goes through a transition of state. The transition of state includes staying at the same state and transferring to another state. In addition, the time for Markov process is discrete and is denoted using $t = 1, 2, \cdots T$. The transition of state follows a set of conditional probabilities known as transition probabilities. The state transition probability is the conditional probability of a state ends up transitioning to another state or itself given the predecessor state (Norris, 1998). For a first order discrete Markov process, the assumption is that current state is only dependent on a predecessor state (Freedman, 2012). The state transition probability a_{ij} for transition of state from S_i to S_j can be expressed as in (2.1). All the possible state transition probabilities of a Markov chain can be arranged in the form of matrix. For example, the state transition probability matrix of the three state Markov process in Figure 2.3 is given in (2.2).

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i) \quad \text{where } 1 \leq i, j \leq N \tag{2.1}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{2.2}$$

The transition probability must follow basic probability axioms such as non-negativity (2.3) and the sum of all probabilities is one (2.4).

$$a_{ij} \geq 0 \tag{2.3}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{2.4}$$

Markov process can be used for anomaly detection on location sequence (Zhao, et al., 2014). Location is similar to activity as it is discrete. In the case of location sequence, the set of states is the set of different locations of a smart home which includes bedroom, bathroom, kitchen, dining room and parlour. For example, a sequence of [*bedroom*, *kitchen*, *dining room*] means that the person went to kitchen from bedroom followed by dining room.

For anomaly detection, two different Markov processes were trained. One of the Markov process models the normal sequences whereas the other Markov process models the abnormal sequences. For each Markov process, the transition probability for transition from state a to b can be calculated using (2.5). Equation (2.5) is the total count of state b occurs after state a denoted by c_{ab} over the sum of all the total count for the adjacent states denoted by S_i and S_j where $1 \leq i, j \leq N$. In other words, the state transition probability is the frequency of a type of transition over all occurred transitions.

$$a_{ab} = \frac{c_{ab}}{\sum c_{ij}} \tag{2.5}$$

At time instant t, the state can be denoted by q_t. Given a Markov process and a sequence of states $Q = q_1, q_2, q_3 \cdots q_T$, the conditional probability $P(Q \mid model)$ is high if that sequence has a similar pattern as the sequences used to obtain the transition probability. Conversely, $P(Q \mid model)$ is low if the given sequence is dissimilar to the sequences modelled by the Markov process. $P(Q \mid model)$ is calculated using equation (2.6).

$$P\,(Q \mid model) = P(q_1)\,P\,(q_2 \mid q_1)\,\ldots\,P\,(q_t \mid q_{t\text{-}1}) \tag{2.6}$$

Then, a score called Log Odds Ratio (LOR) was used to quantify the abnormality of a given sequence. LOR is the log of the ratio of two conditional probabilities: the probability of the sequence given the abnormal model and the probability of the sequence given the normal model. The LOR equation is given in (2.7). If the LOR of a sequence is more than zero, it means that the sequence is more likely to match the pattern of the abnormal sequences. This method achieved a detection ratio of 92.539% for location sequential anomaly detection using a 125 days dataset of an elderly location.

$$\mathrm{LOR} = \log \frac{P(Q \mid abnormal\ model)}{P(Q \mid normal\ model)} \tag{2.7}$$

2.4.2 Hidden Markov Model

Each of the states of a Markov process correspond to an observable physical event (Rabiner & Juang, 1986). This formalism restricts Markov process to be applied to more complex problems. In some problems, certain process is influenced by some other underlying and unseen process. A toy example of this scenario is the urn and ball model (Rabiner L. , 1989). Urn is a kind of container and it consists of balls with *M* distinct colours. In addition, there are *N* urns. In this scenario, there are two processes. The process which produce observable results which is the process of a human retrieving a ball from the given urn. The other process is the underlying or hidden process which involves an imaginary genie selecting an urn out of all the urns. The genie is not observable to human. The human selects ball from the urn chosen by the genie. Markov process is not applicable for this type of problems where there exists another underlying process influencing the observable process.

To include this special case, Hidden Markov Model (HMM) extends the original Markov model such that the observation is a probabilistic function of the state (Bengio, 1999; Murphy, 2013). As such, HMM consists of two connected processes. One of the process is hidden and can only be observed through another process. The

observable process produces the sequence of observations (Symth, Heckerman, & Jordan, 1997). To formalize HMM, the number of distinct observation symbol per state is denoted with M and the number of states of is denoted with N. The individual states are denoted by $S = \{S_1, S_2, \ldots, S_N\}$. In addition, the individual observation symbols are denoted by $V = \{V_1, V_2, \ldots, V_M\}$. For example, in the urn and ball model, the states correspond to the urns. Whereas, the observation symbols correspond to the colours of the balls retrieved from the urns.

Like Markov process, HMM has a set of probabilities called state transition probabilities $A = \{a_{ij}\}$which define the conditional probabilities of a state given predecessor state as given in (2.1). In addition, there exists a set of emission probabilities or output probabilities $B = \{b_j(k)\}$ which give the conditional probability of an observation symbol given a hidden state as follows:

$$b_j(k) = P(V_k \text{ at } t \mid q_t = S_j) \tag{2.8}$$

where $1 \leq j \leq N$ and $1 \leq k \leq M$. Lastly, HMM also consists of the initial state distribution $\pi = \{\pi_i\}$ such that:

$$\pi_i = P(q_t = S_i) \qquad 1 \leq i \leq N \tag{2.9}$$

To make HMM useful in real-world applications, there are three problems to be solved which include evaluation, finding hidden states and training. The evaluation problem is about calculating the probability of a sequence given a trained HMM model. Given a trained model denoted with λ and an observation sequence O, the probability that the sequence was generated by the model, $P(O \mid \lambda)$ can be computed. The brute force method to compute $P(O \mid \lambda)$ is to consider every combinations of sequence of states Q with length T as given in equation (2.10). The more efficient method is Forward Algorithm which uses recursive properties to simplify this calculation.

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q, \lambda)\, P(Q|\lambda) \tag{2.10}$$

For the problem of uncovering hidden part of the HMM model, there is no correct and exact state sequence. An algorithm called Viterbi algorithm (Rabiner L., 1989) can be used to find the state sequence which fulfils the optimality criterion as best as possible. As for the training problem, the aim is to optimize the model parameters such as state transmission matrix and emission probabilities so that the HMM model matches the training set of observation sequences. During training, the Baum-Welch method (Rabiner L., 1989) can be used to iteratively adjust the parameters of the HMM model.

Forkan and colleagues used HMM for sequential anomaly detection on both home activities and location (Forkan, et al., 2015). For the case of home activities, each observation sequence is the home activities sequence. For each sequence, a log-likelihood score log [$P(O \mid \lambda)$] can be computed. If the model matches the sequence, the computed log likelihood score should be high. Conversely if the model does not match the sequence, the log likelihood score should be low. This property is useful for anomaly detection task. A threshold τ can be used to separate high probabilities of normal sequences and low probabilities of abnormal sequences as given in (2.11).

$$\begin{aligned} &\text{if } \log P(O|\lambda) \geq \tau, \text{ then normal} \\ &\text{else } \log P(O|\lambda) < \tau, \text{ then abnormal} \end{aligned} \tag{2.11}$$

Table 2.2: Different HMM Models

Model	Training set
1	Only normal observations
2	Only abnormal observations
3	Both normal and abnormal observations

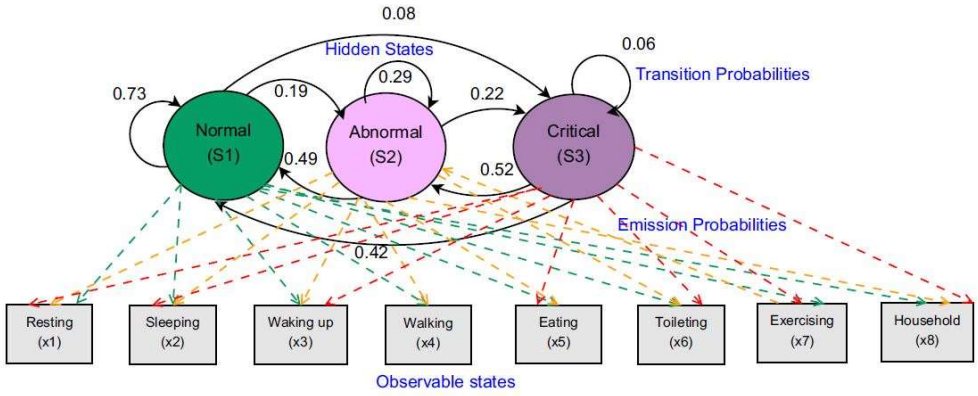

Figure 2.4: Model 3 for Identifying Abnormal Activity in the Activity Sequence

Forkan and partners generated an artificial dataset of 3 months for 10 subjects using MATLAB (Forkan, et al., 2015). In the experiment, they developed and evaluated three different HMM model using this artificial dataset. The details of each of the HMM model is given in Table 2.2. Model 1 and Model 2 were built for sequential anomaly detection, to tell whether a sequence is anomalous or not. On the other hand, Model 3 was built to pinpoint which activity in the anomalous sequence detected by Model 1 or Model 2 contributed to the abnormalities. Figure 2.4 shows Model 3 built with a set of three hidden states, S = {normal, abnormal, critical} where each state represents different severity. Given an activity sequence, Viterbi algorithm can be used to generate a state sequence. By matching and comparing the generated state sequence and observation sequence, the activities in the sequence which caused abnormalities can be identified.

The evaluation results in the research paper shown that Model 1 and Model 2 were excellent at classifying whether a given sequence is normal or anomalous with accuracies which range from 87.50% to 100%. Whereas, the Model 3 was good at locating activities which caused abnormalities in the activity sequence with accuracies varying from 88.5% to 97.10%.

2.4.3 Data Pattern Mining

In Hoque's work, they called sequential relationship between activities as temporal correlations among activities (Hoque, et al., 2015). According to their research, temporal correlation can be sequential or non-sequential. Sequential temporal correlation is about two or more activities that happen one after another and the order of the activities matters. On the other hand, non-sequential temporal correlation consists of few activities that happen together but in no specific order and regardless of time gap between them.

To model sequential temporal correlation, they used a sequential pattern mining algorithm called PrefixSpan (Pei, et al., 2001) and probability distribution. First of all, they broke down the activity sequences of the dataset into smaller segments. Then, they ran the PrefixSpan algorithm on these shorter segments to identify segments which occur frequently and retrieve them. The number of occurrences of a segment must exceeds a threshold to be considered as frequent. After all the frequent segments were obtained, the time interval between successive activities of each segment was modelled using Gaussian distribution. This step enables anomaly detection because the anomaly would be an outlier with respect to the distribution.

For non-sequential temporal correlation, Hoque and researchers used a type of data mining algorithm called itemset mining algorithm to recover frequent group of activities that happen collectively without any correlation sequentially or itemsets (Hoque, et al., 2015). Specifically, they used the Apriori algorithm (Agrawal & Srikant, 1994) to collect frequent itemsets from the segments of activity sequences. Similarly, there is also a threshold for this algorithm. If the occurrence of an itemset exceeds that threshold, then only it will be considered as frequent.

2.4.4 Strength and Weakness of Sequential Anomaly Detection Methods

Markov process and HMM are better than data mining algorithms in terms of data usage. This is because both Markov process and HMM used entire sequence for training. Whereas, the data mining algorithms only extract frequent segments of

activities from the training sequences. It only used a small fraction of the dataset for building the anomaly detection model. Therefore, the model can only detect the types of anomalies that had happened before. It did not learn the usual pattern of home activities and therefore cannot detect unseen anomalies that does not conform to the pattern.

In terms of model complexity, HMM is an extension of Markov process as HMM can model two processes where one is observable and the other is hidden. Using Viterbi algorithm, HMM can even be used to locate the activities that is causing anomaly in the sequence. Other existing methods such as Markov process and data mining algorithms cannot do this. However, experiment of HMM on the Aruba dataset has only an average accuracy of 50%. This means that HMM may not be suitable for realistic dataset of home activities. The comparison and weakness of each sequential anomaly detection algorithms is listed in Table 2.3.

Table 2.3: Summary of Sequential Anomaly Detection Algorithms

Algorithms	Descriptions
Markov Process	• Use entire data
HMM	• Extension of Markov Process • Use entire data • Can be used to detect which activity caused anomalies in the sequence • Achieved an average accuracy of 50% on Aruba dataset
Data Mining	• Only uses a small portion of dataset • Oblivious to unseen anomalies

2.5 Time Anomaly Detection Methods

Subsection 2.5.1 discusses about Forkan's work on using Gaussian distribution for time anomaly detection (Forkan, et al., 2015). Subsection 2.5.2 presents Zhao's anomaly detection method using *K*-means clustering and Gaussian distribution for time anomaly detection (Zhao, et al., 2014). Whereas, subsection 2.5.3 discusses about

clustering, DBSCAN and Hoque's implementation for time anomaly detection (Hoque, et al., 2015). In subsection 2.5.4, the strength and weakness of each method to detect time anomaly are presented.

2.5.1 Time Anomaly Detection Using Gaussian Distribution

Probability distribution is a function which describes the probability of occurrences of all possible outcomes of a random variable (Ross, 2009). There are two types of probability distribution namely discrete and continuous probability distribution (Dekking, et al., 2006). For the discrete probability distribution, the outcomes are discrete in nature. An example of events which can be described using discrete probability distribution is coin tossing with two possible outcomes namely head or tail. On the other hand, outcome of continuous probability distribution takes an interval of values.

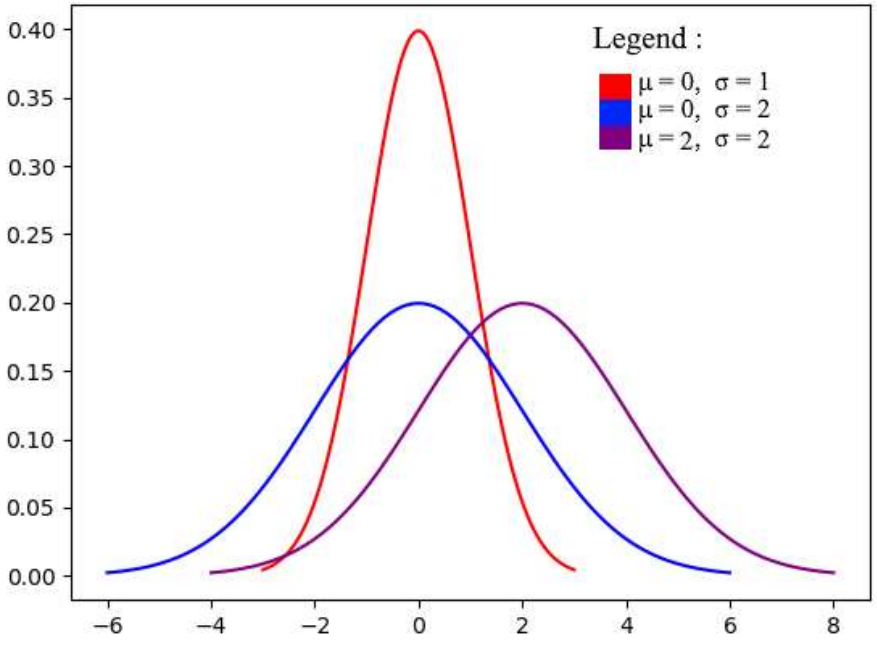

Figure 2.5: Gaussian Distribution with Varying μ and σ

Gaussian distribution or normal distribution is a common continuous probability distribution. Many common attributes such as height, weight and test scores follow normal distribution. Given a large enough sample, most of the points of the sample are centred around the mean forming a peak and there are few points scattering at the lower and higher ends. Because of its curve shape as shown in Figure 2.5, it is called "bell curve". The probability density function of Gaussian distribution is given in equation (2.12) where x can be any value in the range of $(-\infty, +\infty)$. The shape characteristics of the Gaussian distribution is governed by the mean and standard deviation. A random variable that is normally distributed can be written as $N(\mu, \sigma)$. In this convention, μ is the mean and σ is the standard deviation which is square root of variance σ^2. Mean can be calculated using equation (2.13) and variance can be calculated using equation (2.14).

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.12}$$

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \tag{2.13}$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2 \tag{2.14}$$

In Figure 2.5, three different Gaussian distributions are presented. The blue Gaussian distribution with $\mu = 0$ has peak centres around $x = 0$. On the other hand, the purple Gaussian distribution with $\mu = 2$ is the blue Gaussian distribution shifted to the right by two. Standard deviation is an indicator of the degree by which Gaussian distribution plot is spread out from the mean (Rohatgi & Ehsanes Saleh, 2011). In Figure 2.5, red Gaussian distribution with $\sigma = 1$ is narrower and taller than blue Gaussian distribution with $\sigma = 2$.

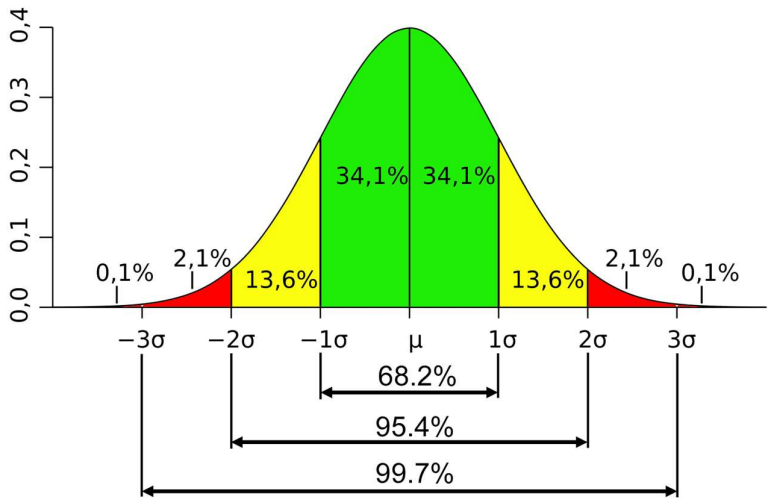

Figure 2.6: Gaussian Distribution

As mentioned earlier, most of the values centres around the mean μ as shown in Figure 2.6 where approximately 68.2% of the points of the distribution is in the range $[\mu - \sigma, \mu + \sigma]$, 95.4% of the points in the range $[\mu - 2\sigma, \mu + 2\sigma]$ and 99.7% of the points in the range $[\mu - 3\sigma, \mu + 3\sigma]$ (Mendenhall, et al., 2012). This characteristic of Gaussian distribution can be used for anomaly detection where a point which is far from the mean can be detected as an outlier if it is not within the defined range.

Forkan and colleagues (2015) used Gaussian distribution to detect anomaly in time and duration of home activities. For each type of home activities, they divided the Gaussian distribution into different ranges of severity. They defined three ranges which are normal region, alert region and critical region range as listed in Table 2.4. Forkan and colleagues also listed the classification results such as the number of data points detected for each class label without any evaluation metrics such as accuracy.

Table 2.4: Ranges for Time Anomaly Detection of Home Activities

Name	Range
Normal range	$[\mu - \sigma, \mu + \sigma]$
Alert region range	$[\mu - \delta_1\sigma, \mu + \delta_1\sigma]$
Critical region range	$[\mu - \delta_2\sigma, \mu + \delta_2\sigma]$

2.5.2 A Combination Method of *K*-means Clustering and Gaussian Distribution

Clustering is a category of unsupervised learning algorithm which divides unlabelled data points into groups such that similar data points are grouped together (Wu, 2012). Each data point x can be n dimensional vector where each dimension is a unique attribute or feature.

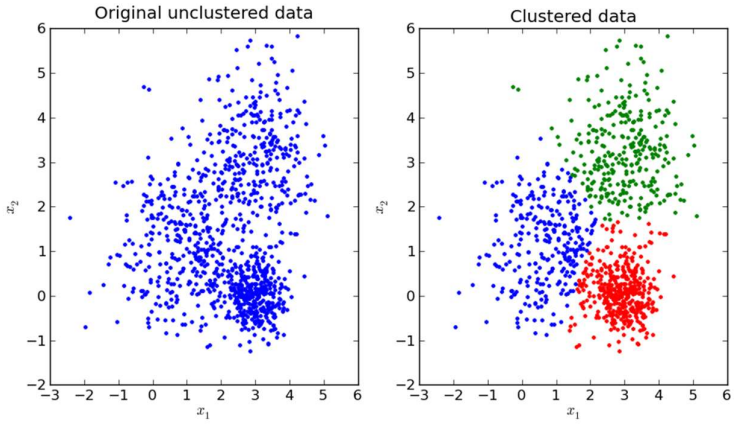

Figure 2.7: *K*-means Clustering for 2-dimensional Data Points

K-means clustering algorithm groups data by separating data points into K different clusters as shown in Figure 2.7 (Wu, 2012; Natingga, 2017; Madhavan, 2015). The number of clusters denoted by K is a hyperparameter that the user has to define. Each cluster consists of a cluster centroid μ_j. Cluster centroid is the mean calculated for all the points assigned to that cluster j where $j = 1,2, \ldots, K$ (VanderPlas, 2016). The clustering algorithm consists of three steps. First of all, K data points are randomly selected as cluster centroid. Secondly, find the nearest centroid μ_j for each of the point in the dataset which minimizes the within-cluster sum (WCSS) of squared criterion in equation (2.15). Each data point is then assigned to the cluster j with centroid μ_j which is the nearest to it.

$$\text{WCSS} = \sum_{i=1}^{n} min_{\mu_j \in C} (\| x_i - \mu_j \|^2) \tag{2.15}$$

where C refer to the set of all cluster centroids.

Thirdly, the centroid of each cluster is replaced with the computed mean of the data points assigned to that particular cluster in the previous step. Step two and Step three are repeated until convergence where the cluster centroids do not move significantly. Given enough time, *K*-means clustering will always converge but sometimes to a local minimum. The convergence is highly dependent on centroid initialization. Therefore, the steps are often repeated several times, each time with different centroid initialization.

Zhao and colleagues (2014) demonstrated using *K*-means clustering and Gaussian distribution for time anomaly detection of location. They collected a dataset of an elderly's location at home with timestamp. There are two types of time features namely starting time of staying at a location and its duration. Here, each data point is a tripleton of location, starting time and duration. They used *K*-means clustering to group a subject's location based on starting time and duration. For example, they set $K = 3$ for kitchen and $K = 4$ for bedroom.

$$P(d < d_\alpha) = \int_{-\infty}^{d_\alpha} f(x)\, dx = \alpha \tag{2.16}$$

Then, they approximated each cluster using Gaussian distribution. For each cluster of a location at home, anomalies due to duration can be detected using α-quantile. α is a value to be set by user. With α, the threshold d_α can be obtained as given in (2.16). If the duration of staying at a location is greater than d_α, then it is classified as abnormal. Conversely if the duration of staying at a location is less than d_α, then it is classified as normal.

As for detecting starting time anomaly, a dominant range $[\mu - 2\sigma, \mu + 2\sigma]$ which consists of 95.4% of the distribution is defined. Any data point with starting time outside this range is classified as outlier. This combined method with *K*-means clustering and Gaussian distribution was demonstrated to have an excellent accuracy of 92.80%.

2.5.3 Time Anomaly Detection Using DBSCAN

As opposed to *K*-means, the Density-based Spatial Clustering of Applications with Noise (DBSCAN) algorithm can handle noise in a given dataset and user does not have to specify number of clusters, *K*. In short, it is a density-based clustering algorithm where closely packed points in some space are grouped together and other points located on lower density region are marked as outliers (Ester, et al., 1996). For DBSCAN, there are two important hyperparameters *min_pts* and *eps* which define the density. The details about each hyperparameter is given in Table 2.5.

Table 2.5 Hyperparameters of DBSCAN Clustering Algorithm

Hyperparameters	Description
min_pts	Minimum number of data points within the neighbourhood needed for a data point to be regarded as a core point
eps	Radius of neighbourhood with respect to a data point

For DBSCAN clustering, there are three types of data points namely core point, border point and noise point. Firstly, a data point is a core point if there are at least *min_pts* data points (including itself) within a neighbourhood with radius *eps* from itself. Secondly, a data point is a border point if there are fewer than *min_pts* data points within a neighbourhood with radius *eps* from itself. But at the same time, there should be at least a core point within *eps*. Border points are also called density-reachable points because they are directly reachable from at least a core point. Lastly, a data point is a noise point or outlier if it does not fulfil the condition of a core point or a border point.

For DBSCAN clustering, each cluster needs to have at least a core point. Border points form the cluster's edge because there are no other points that the border point can reach within *eps*. Figure 2.8 visualizes DBSCAN clustering on a 2-dimensional dataset. The clustering results in 3 clusters where each cluster is represented with a unique colour. The gigantic coloured circles are core points, smaller coloured circles are border points and the smallest black circles are noise points.

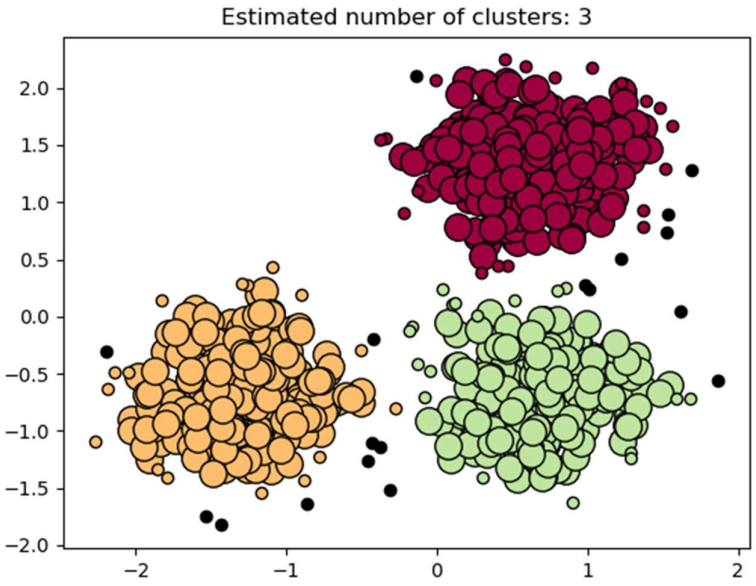

Figure 2.8: DBSCAN Clustering for 2-dimensional Data Points

The DBSCAN algorithm performs clustering by following several steps. In short, the algorithm will look recursively for core points that form a cluster. Firstly, the algorithm looks at each point in the dataset and tries to determine whether that point is a core point or not. If that point is a core point, then the algorithm will investigate the neighbouring points within *eps* of the first core points with the aim to check whether those points are core points or not. If they are indeed core points, these points will be labelled as such. Then, the algorithm will repeat this step for these secondarily detected core points to find more core points within *eps* of detected core points. The recursive search for core points will continue until there is no core point left within *eps* of previous core points. These core points form a new cluster. After that, the algorithm looks at non-core points that are within *eps* of these detected core points to classify them as border point or noise point. Border points will be added to the cluster whereas noise point will be labelled as outlier. Note that these noise points might belong to another cluster when the algorithm repeats the entire process on other unchecked data points.

The DBSCAN algorithm has several advantages and disadvantages. The first advantage would be that it does not require specification of number of clusters as opposed to *K*-means clustering (Ali , et al., 2010). Instead, DBSCAN requires specification of density-based hyperparameters such as *eps* and *min_pts*. These hyperparameters are more intuitive and can be set by domain experts because they understand the nature of the dataset. Thirdly, it can be used to find arbitrarily shaped clusters since it relies on density. Furthermore, the algorithm can identify noise in the dataset and is robust to outliers. Unfortunately, all these features are what *K*-means clustering algorithm lacks, making DBSCAN a more powerful algorithm than *K*-means.

There are two main disadvantages for DBSCAN clustering algorithm. Firstly, DBSCAN is not suitable for datasets with substantial difference in densities. This is due to *min_pts* and *eps* are used to specify a single density throughout whole dataset (Kriegel, et al., 2011). They cannot be set to accommodate more than one density. Therefore, the requirement of DBSCAN is the dataset must have a similar density. Secondly, the assignment of a border point to certain cluster is not deterministic because the assignment is dependent on the order the data points is processed (Schubert, et al., 2015). This means if the order of processing the data points is changed, border point that was previously assigned to a cluster might be assigned to another cluster.

In Hoque's research (2015), the data points are clustered based on two features which are starting time and duration collectively by using DBSCAN. After acquiring a number of clusters for each type of home activity, agglomerative clustering algorithm was used to merge clusters which are closest to each other. For each cluster, the mean μ_i and standard deviation σ_i for each feature i of the training data points belonging to that cluster were calculated. For anomaly detection, Mahalanobis Distance, d (De Maesschalck, et al., 2000) can be calculated using equation (2.17) where k is the number of features per data point. If a new data point is not within two standard deviations from the centre of the cluster, it is classified as anomalous. They evaluated

this anomaly detection method using a 6 months dataset. The result shown that this method increases precision by at least 17% and recall by at least 6%.

$$d = \sqrt{\sum_{i=1}^{k} \frac{(x_i - \mu_i)^2}{\sigma_i^2}} \tag{2.17}$$

2.5.4 Strength and Weakness of the Sequential Anomaly Detection Methods

Each of the existing sequential anomaly detection algorithms that are discussed earlier have strength and weakness. The strength and weakness of each method are discussed in the following subsection.

2.5.4.1 Strength and Weakness of Gaussian Distribution Method

The strength of Forkan's Gaussian distribution method is speed. It is the "simplest" method out of 3 related works presented for time anomaly detection. "Simplest" means that the computation and detection is the fastest.

The weakness of only using Gaussian distribution for anomaly detection is that it is too "simple". An activity type can have more than one cluster of starting time and duration. The Gaussian distribution centres around the largest cluster while ignoring other smaller clusters. Those smaller but frequent occurring clusters might be far from the main distribution. If a data point is not located within the largest cluster, it will be classified as abnormal.

2.5.4.2 Strength and Weakness of *K*-means Clustering and Gaussian Distribution Method

Using *K*-means clustering and Gaussian distribution collectively for anomaly detection is better than using only Gaussian distribution alone. As mentioned previously, a single activity type can have more than one clusters of starting time and duration. This method involves first using the *K*-means clustering to group data points of each activity based on starting time and duration. After that, each of the clusters can

be modelled using Gaussian distribution. This avoid a data point belonging to a smaller and less frequent occurring cluster to be misclassified as abnormal.

The weakness of this algorithm is its increased complexity when compared to the first method which used only Gaussian distribution. *K*-means clustering requires human expertise for setting of *K*. Of course, the selection of *K* can be automated by using model selection method such as elbow method. However, this anomaly detection method used both *K*-means clustering and Gaussian distribution collectively, this makes the automated training and model selection process complicated. There is at least one model for every activity type. The automated training and model selection process repeats for every activity type. All in all, this makes the modelling process hefty and complicated.

2.5.4.3 Strength and Weakness of DBSCAN

The strength of DBSCAN algorithm is that it can cluster the data points without specifying the number of clusters *K*. The weakness of DBSCAN algorithm is that the hyperparameters such as *min_pts* and *eps* allow only one type of density. In the real case, the activities may have more than one density where DBSCAN does not allow this. This means that for each type of activity, human expertise will be needed for hyperparameter settings. For this method, there is one or more model per type of activity which makes the training a hefty process. The details of each algorithm are listed Table 2.6.

Table 2.6: Summary of Time Anomaly Detection Algorithms

Algorithms	Description
Gaussian Distribution Method	• Simplistic
K-means Clustering and Gaussian Distribution Method	• More advanced out of three methods • Requires human expertise in setting hyperparameters • Training and model selection process is complicated

	• At least one models per activity type
DBSCAN	• Requires human expertise in setting hyperparameters • At least one models per activity type

2.6 Neural networks

LSTM is a specialized recurrent neural network which may be suitable for small dataset and as an alternative of HMM. In subsection 2.6.1, neural network is introduced. Subsection 2.6.2 presents the gradient descent which is an optimization method to train the neural networks. Subsection 2.6.3 talks about the recurrent neural network which is a more specialized neural network for sequence data. Subsection 2.6.4 presents the LSTM neural network which is a variant of recurrent neural network. Subsection 2.6.5 covers the language model which inspired the LSTM method. This LSTM method is described later in Chapter 3 for sequential anomaly detection.

2.6.1 Introduction to Neural Network

As shown in Figure 2.9, a biological neuron receives electrical impulses from neighbouring neurons via tree branch like structure called dendrites (Morris, 2004). Each line of input is controlled by a synaptic weight. Synaptic weight is the strength of a connection between two nodes. Based on these weighted inputs, the neuron decides whether to fire or send electrical impulse to neighbouring neurons.

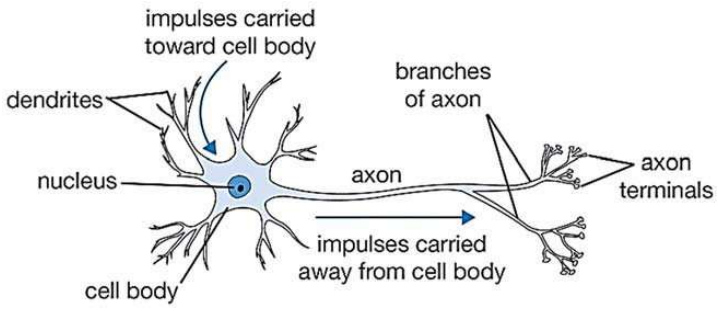

Figure 2.9: Biological Neurons

Neural network is the most basic form of deep learning algorithm which was loosely inspired by biological neurons found in living things (Goodfellow, et al., 2016). There are two inspirations gained from biological neurons. The first is the synaptic weights which influence the strength of the signals transmitted from one neuron to another. In other words, synaptic weights are the weightages of importance for different signals. A neuron decides whether to fire or send an electric impulse based on these weighted input signals. In addition, these synaptic weights adapt and change over time so that the brain learn to perform useful computations for certain task such as recognizing object. Artificial neural network learns by adjusting the parameters which include weight and bias using an optimization technique known as gradient descent. The second inspiration is the deep (many layers) connection of many neurons. It was discovered that neural networks can learn increasingly complex features and concepts as the number of layers increases as shown in Figure 2.10 (Ng & Katanforoosh, 2018) which takes face recognition as an example, earlier layers of a neural network learn to detect edges and simple shapes. Building up from these edge and shape detectors, the intermediate layers of the neural network learn to distinguish various parts of face such as eyes, noses and ears in the intermediate layers. In the final layer, the neural network can differentiate faces of different person based on all earlier features such as eyes, noses and etc.

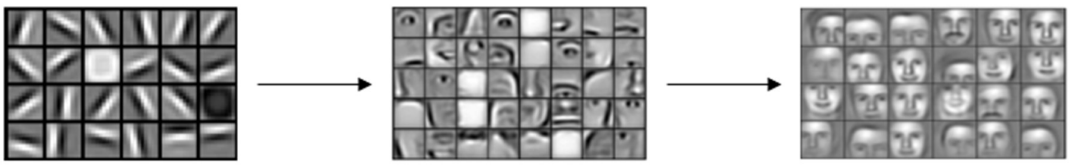

Figure 2.10: Visualization of Features Learnt at Different Layers for Face Recognition

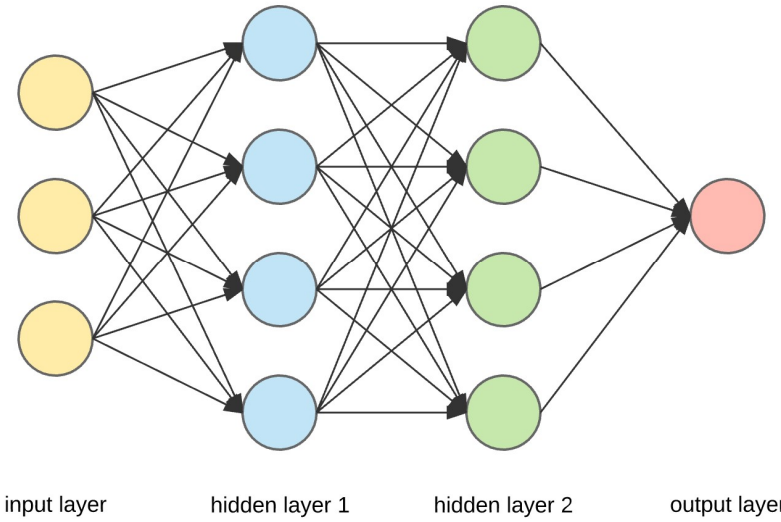

Figure 2.11: A Vanilla Neural Network with 2 Hidden Layers

Figure 2.11 shows a vanilla neural network with 2 hidden layers (blue and green). Vanilla is the term used in literature to describe the plainest version of a deep learning algorithm. In addition, the circles in Figure 2.11 are known as nodes. A node can either be input node, hidden unit or output node depending on the layer. In Figure 2.11, there is a line arrow connecting one node from one layer to another node at the next layer. That line indicates a connection between each pair of nodes such that the successor node uses the output value of previous node to compute its output value. Each connection is associated with a weight which scales the output value from previous node. The scaling has a multiplying or diminishing effect depends on the magnitude of the weight.

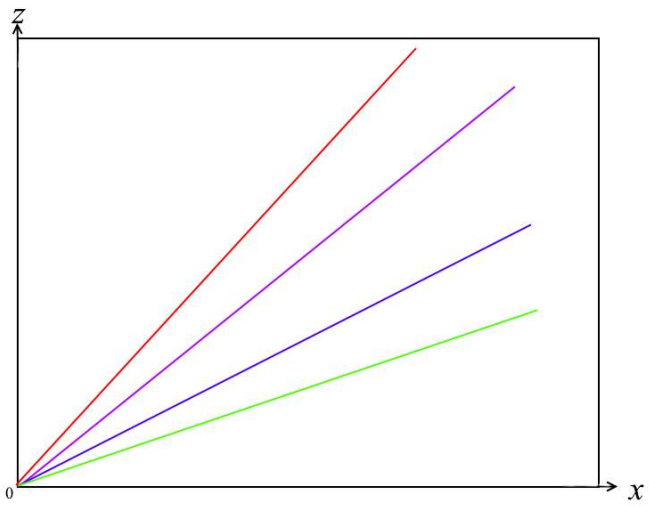

Figure 2.12: Graph of z for Different w and No Bias ($b = 0$)

For simplicity, consider a single node which takes in input x and outputs an activation value a. The computation of activation can be divided into 2 steps as shown by equation (2.18) and equation (2.19).

$$z = w\,x + b \tag{2.18}$$

$$a = g(z) \tag{2.19}$$

There are two parameters in the computation namely the weight w and bias b (Ng & Katanforoosh, 2018). For each connection between two nodes in the neural network, there exists a weight. Equation (2.18) shows that the weight w can be interpreted as the gradient of the function z in terms of x. The weight w scales the input. On the other hand, each hidden unit consists of a bias b. The bias b is added to improve the model's flexibility. Without b, the neuron can only learn different linear functions. Visually, they are straight line passing through the origin as shown in Figure 2.12. Whereas with bias term, the line can be shifted in z-axis (Anthony & Bartlett, 2009). In this setup, the neuron can be more flexible in forming different linear functions by adjusting the weight and bias.

In equation (2.20), a non-linear function is applied to z to limit the output value of a node called activation a. Imagine a neural network with 10 hidden layers and ignores the bias b and each activation computation reduces to $z = w \cdot a$ and the resulting activation at the final layer would be $z = (w^{10})\, a$. If the activation a of the first hidden layer is an integer greater than 1, the resulting z at the final layer will be very large. This large number cannot be represented in digital computers because it overflows the limited memory. Conversely, if a is small value, the resulting value at the final layer z will underflow. Therefore, a non-linear activation function $g(\cdot)$ is needed to squish or limit the z to a range of values. Some common activation functions are sigmoid, tanh and rectified linear (ReLU) activation function as given in equations (2.20), (2.21) and (2.22) (Goodfellow, et al., 2016).

$$\text{sigmoid: } g(z) = \frac{1}{1+e^{-z}} \tag{2.20}$$

$$\tanh: g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.21}$$

$$\text{ReLU: } g(z) = \max(z, 0) \tag{2.22}$$

Generalizing to a multilayer neural network given in Figure 2.10, the input is a group of input features x_1, x_2, x_3 together known as the input layer. The intermediate layers are called hidden layers (Gurney, 1997). Each node of the hidden layer is called a hidden unit. The term neuron is also used to refer to hidden unit interchangeably. The final layer with a single node is known as output layer. The layers of a neural network are indexed with $l = 0, 1, 2, \ldots, L$ where the input layer is layer 0 and the output layer is layer L. For the first hidden layer, each hidden unit takes x_1, x_2 and x_3 from previous layer as input and outputs a value called activation. Similarly, the units in the second layer takes activations from the first hidden layer as input and outputs their activation.

To generalize the notation to a multilayer vanilla neural network like that shown in Figure 2.11, let $a_j^{[l]}$ denotes the activation which is an output of the j^{th} unit in layer l. Note the total number of hidden units in layer l is denoted by $n^{[l]}$. The weights

for the connection between layer l and l+1 are arranged in a matrix $W^{[l+1]}$ with size $n^{[l+1]} \times n^{[l]}$. In addition, the activations for layer l are arranged in a vector denoted as $A^{[l]} = \{a_1^{[l]}, a_2^{[l]}, \ldots, a_{n^{[l]}}^{[l]}\}$. The input features can be arranged as a vector which is denoted as $A^{[0]}$. The activation $A^{[l]}$ is calculated using equation (2.23) and (2.24). To calculate the activations for a layer l denoted by $A^{[l]}$, the vector of input activations from previous layer denoted by $A^{[l-1]}$ is first used to compute $z^{[l]}$ vector using equation (2.23). After that by using equation (2.24), the activation function $g(\cdot)$ is applied element-wise to the $z^{[l]}$ vector. For a neural network with L layers, equations (2.23) and (2.24) are used in every layer from the first layer until final layer L to compute the output $\hat{y} = A^{[L]}$ as depicted in Figure 2.13. This process of using previous activations and current weights and biases to compute current activations and repeating it for the next layer is known as forward propagation (Rojas, 2013).

$$\underset{z^{[l]} \in \mathrm{R}^{n^{[l]} \times 1}}{\begin{bmatrix} z_1^{[l]} \\ \vdots \\ z_{n^{[l]}}^{[l]} \end{bmatrix}} = \underset{W^{[l]} \in \mathrm{R}^{n^{[l]} \times n^{[l-1]}}}{\begin{bmatrix} w_{11}^{[l]} & \cdots & w_{1n^{[l-1]}}^{[l]} \\ \vdots & \ddots & \vdots \\ w_{n^{[l]}1}^{[l]} & \cdots & \mathrm{w}_{n^{[l]}n^{[l-1]}}^{[l]} \end{bmatrix}} \underset{A^{[l-1]} \in \mathrm{R}^{n^{[l-1]} \times 1}}{\begin{bmatrix} a_1^{[l-1]} \\ \vdots \\ a_{n^{[l-1]}}^{[l-1]} \end{bmatrix}} + \underset{\boldsymbol{b}^{[l]} \in \mathrm{R}^{n^{[l]} \times 1}}{\begin{bmatrix} b_1^{[l]} \\ \vdots \\ b_{n^{[l]}}^{[l]} \end{bmatrix}} \tag{2.23}$$

$$A^{[l]} = g(z^{[l]}) \tag{2.24}$$

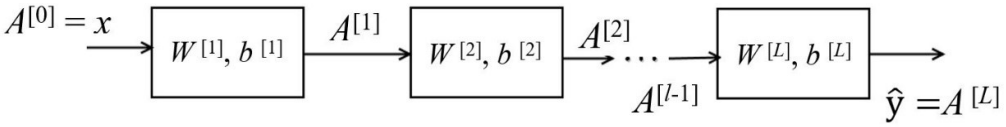

Figure 2.13: Illustration of Forward Propagation

Neural networks can be used for binary classification and multiple class classification. The output values of output layer denoted by $\hat{y}$ and can be interpreted as conditional probabilities of the being in one of the class given the input denoted by x.

2.6.2 Gradient Descent

For training neural networks, many examples of input and target pairs of the classification task are needed and these examples are called training examples (Ng &

Katanforoosh, 2018). Target y is the value that the output $\hat{y}$ aim to become. In other words, the goal of training is to minimize the discrepancies between the output and target for each training examples. These discrepancies can be measured using loss function J. An example of loss function is the mean squared error. Using the single neuron neural network example with a weight w and bias b again, equation (2.25) shows the mean squared error for m training examples. The goal is to find w and b which minimizes $J(w, b)$. If the neuron only consists of a parameter w, its loss function graph is a bowl curve or convex as depicted in Figure 2.14. In other words, gradient descent is a method which enables the initialized point to go down the bowl shape hill to reach the minimum point.

$$J(w, b) = \frac{1}{m}\sum (y^{(i)} - \hat{y}^{(i)})^2 \quad \text{where } \hat{y}^{(i)} = wx^{(i)} + b \tag{2.25}$$

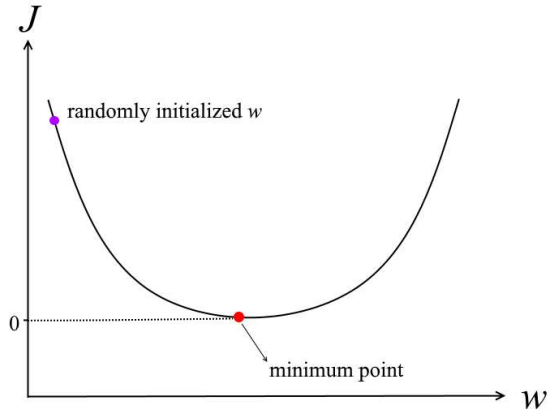

Figure 2.14: The Illustration of Loss Function *J* in terms of *w* with *b* = 0

The goal is already formalized. Subsequently, the gradient descent method used for minimizing loss function is presented. First, the weight w is initialized to some reasonable value. The loss can be reduced iteratively using equation (2.26). The weight needs to be updated to a value which corresponds to a smaller loss. The direction and magnitude of the update is given by the derivative $\frac{dJ}{dw}$ and the learning rate α scales the update. If α is too large, it will gloss over the minimum point. Conversely, if α is too small, it will take a long time before gradient descent converge to the minimum point. This is for the simplest case of a single neuron and a parameter. For a multilayer neural

network with multiple parameters, the loss function is not convex. It is similar to the function *f* illustrated in Figure 2.15 with a global minimum and multiple local minima (Goodfellow, et al., 2016). It is hard for gradient descent to bring the point to the global minimum when there are multiple local minima around the global minimum. Therefore, a local minimum which is close to the global minimum is an acceptable alternative.

$$w = w - \alpha \frac{dJ}{dw} \tag{2.26}$$

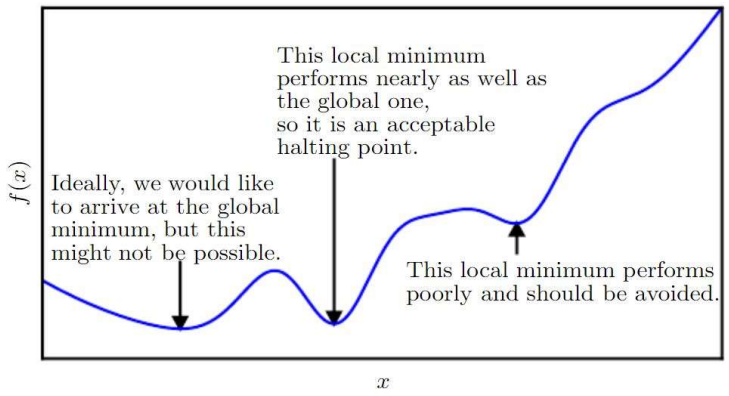

Figure 2.15: A Non-convex Function with Multiple Local Minima

As for minimizing the loss function of a multilayer vanilla neural network, the weights can be updated using equation (2.27) which is a vectorized version of equation (2.26) where $\frac{\partial J}{\partial W^{[l]}}$ is a vector containing partial derivatives of loss function with respect to weights in layer *l*. The biases can be updated in a similar fashion with the derivative $\frac{\partial J}{\partial b^{[l]}}$. The learning rate α is one of the hyperparameters. Hyperparameters are settings of a deep learning algorithm which is fixed before the training process starts. Returning to α, it scales the magnitude of steps.

$$\begin{aligned} W^{[l]} &= W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}} \\ b^{[l]} &= b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}} \end{aligned} \tag{2.27}$$

2.6.3 Recurrent Neural Network

Recurrent neural networks are a family of neural networks which are suited for time series or sequential data (Goodfellow, et al., 2016). Examples of time series data are English sentences, speech data and music. Given a time series data, a data point might be influenced by a data point earlier in the sequence. For example, when reading a sentence, humans do not just throw away everything he or she learn in previous words of the sentence and start understanding from scratch. A vanilla neural network does not have the trait to model sequence data as it cannot share features learned across different position of the sequence.

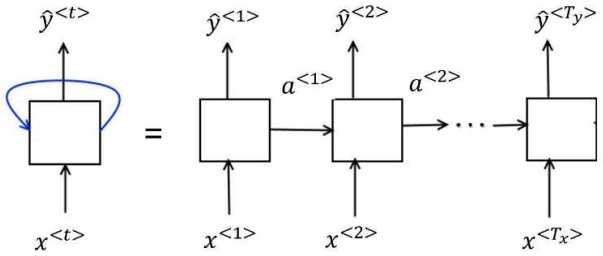

Figure 2.16: A Single Layer Vanilla Recurrent Neural Network

Figure 2.16 shows a single layer vanilla recurrent neural network with the square block representing a neural network. It uses a combination of input data for current time step, $x^{<t>}$ and activation from previous time step $a^{<t-1>}$ as input to compute activation for current time step $a^{<t>}$, which in turn is used to compute the prediction for current time step $\hat{y}^{<t>}$ with <·> indexing the time step. The equations are given in (2.28). The weights W_{aa}, W_{ax} and W_{ya} are not indexed with <t>. One can say that recurrent neural network is a neural network which share parameters which include weights and biases across time. Usually, the recurrent neural networks are not deep in terms of layers, because they are already deep in terms of time step.

$$\begin{aligned} a^{<t>} &= g(W_{aa}a^{<t-1>}+W_{ax}x^{<t>}+b_a) \\ \hat{y}^{<t>} &= g(W_{ya}a^{<t>}+b_y) \end{aligned} \tag{2.28}$$

On the left of Figure 2.16, it shows a neural network which takes in input and computes an output and there exists a loop feedback of information from previous time-step to be used in computation at current time-step. On the right of Figure 2.16, another view of recurrent neural network is multiple copies of the neural network interconnected with each copy passes information to the next. Each copy is for different time step <*t*>. This sequential chain-like structure of recurrent neural network shows that it is related to sequences and lists. It is useful for working with time series data.

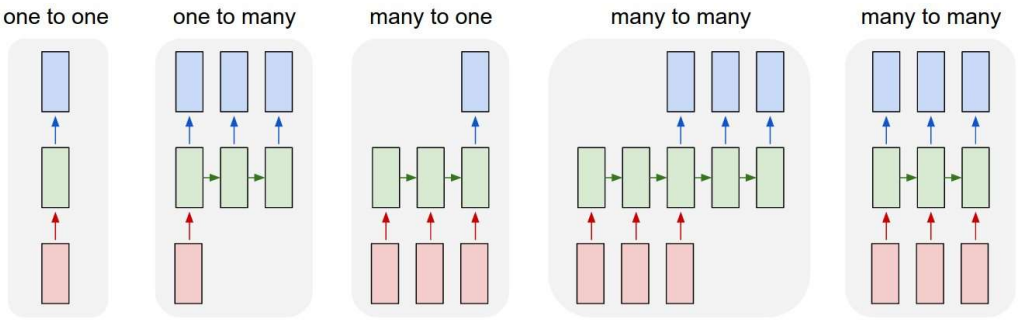

Figure 2.17: Types of RNN Architecture

Recurrent neural networks have many applications. For example, they are used for speech recognition, language modelling (Mikolov, et al., 2010; Peters, et al., 2018), machine translation (Cho, et al., 2014), diabetic glucose prediction (Dong, et al., 2019) and text generation (Abujar, et al., 2019). There are four common types of recurrent neural networks as displayed in Figure 2.17. Each architecture is used depending on the nature of the problem. There are one-to-many, many-to-one and two different many-to-many recurrent neural networks. On the leftmost of Figure 2.17, the one-to-one recurrent neural network is really a standard vanilla neural network. In Chapter 3, the LSTM method uses many-to-one architecture. Before going into that in Chapter 3, a more advanced variant of recurrent neural networks called Long Short-Term Memory (LSTM) is presented. LSTM which brought about many successes and works better than standard vanilla recurrent neural network.

2.6.4 Long Short-Term Memory neural network

Recurrent neural networks enable relevant past information to be connected to the present task. For example, in activity recognition, actions are continuous and relevant information from past video frames are used for the understanding of the present frame. LSTM (Olah, 2015; Hochreiter & Schmidhuber, 1997) has more specialized components which allow it to do this better than a vanilla recurrent neural network.

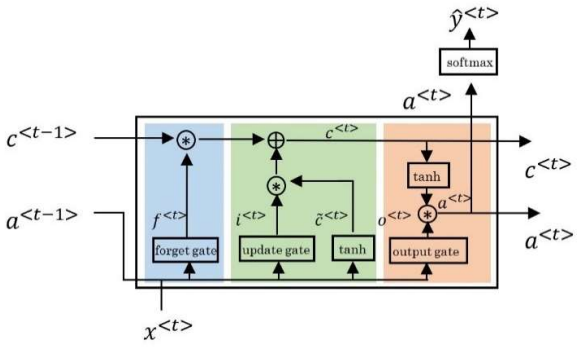

Figure 2.18: A Unit of LSTM

Figure 2.18 shows a unit of LSTM to replace the square block in Figure 2.16. It takes $c^{<t-1>}$, $a^{<t-1>}$ and $x^{<t>}$ as input to compute $c^{<t-1>}$, $a^{<t>}$ and $y^{<t>}$. To simplify the explanation, the LSTM unit is divided into 3 regions: blue, green and orange as shown in Figure 2.18.

In the blue region, the forget gate Γ_f decides information to be discarded from the previous time step cell state $c^{<t-1>}$ which is used to store past information. The forget gate can be computed using equation (2.29). The size of the vector of forget gate Γ_f and past cell state $c^{<t-1>}$ is equivalent. The forget gate takes on values between 0 and 1 where a 0 represents forget everything and a 1 represents keep everything.

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}]+b_f) \tag{2.29}$$

In the green region, there are two main equations including equation (2.30) and equation (2.31). Equation (2.30) is about computation of candidate values $\tilde{c}^{<t>}$ that

might be added to the cell state. Equation (2.31) is about calculating the update gate which is denoted using Γ_u. It takes values between 0 and 1 and controls the updates to the cell state with a 0 represents no update at all and a 1 represents an update using the candidate values. Then, the forget gate Γ_f and update gate Γ_u are used together to decides what to keep from the past cell state and what to update using the candidate values to get the next cell state $c^{<t>}$ as defined in equation (2.32).

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}]+b_c) \tag{2.30}$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}]+b_u) \tag{2.31}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{2.32}$$

In orange region, the output gate Γ_o acts as a filter which decides the parts of the cell state that are going to be outputted as defined in equation (2.33).

$$a^{<t>} = \Gamma_o * \tanh c^{<t>} \tag{2.33}$$

There are other variants of LSTM which has additional specialized components such as LSTM with peephole connection and Gated Recurrent Unit (GRU) but the differences are minor. A research compared the performance of popular variants of LSTM and found that they work about the same (Greff, et al., 2017).

2.6.5 Language Model

The LSTM method described in Chapter 3 is inspired by some ideas of language modelling. A language model consists of the probability distribution over sentences of words (Karpathy, 2015). It is used to compute the probability of likeliness of a sentence. For example, a speech recognition system interpreted a speech signal into 2 sentences where the first sentence $S1$ is "The apple and ***pair*** salad" and the second sentence $S2$ is "The apple and ***pear*** salad". An English speaker would know right away that $S1$ does not make sense and $S2$ is much more likely to be a correct interpretation of that speech signal. An appropriate language model will estimate that $P(S1) < P(S2)$. To be clearer, a language model is used to compute the joint probability

of the individual words. For example: $P(S1) = P$ (“The”, “apple”, “and”, “pair”, “salad”).

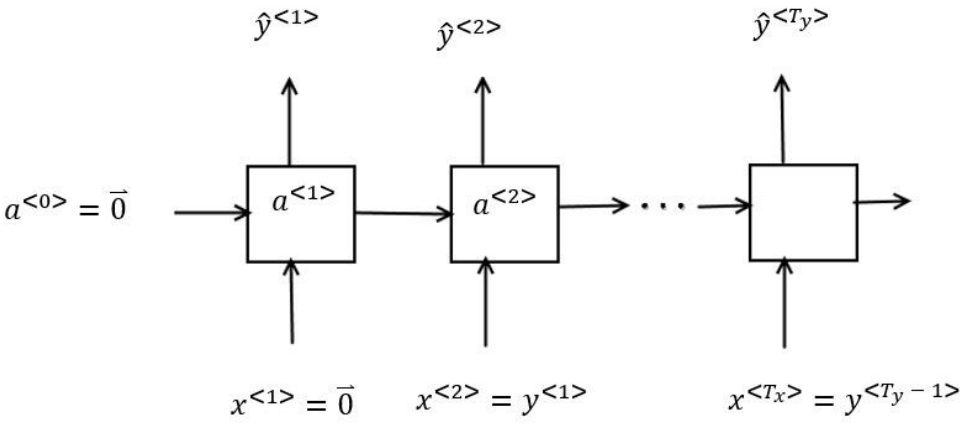

Figure 2.19: A Many-to-many RNN for Language Modelling

The language model is built using a many-to-many recurrent neural network architecture as shown in Figure 2.19. It uses a training set of large text corpus of the target language. Words are categorical not numerical and computers cannot understand them. To overcome this, each of the words is encoded into a unique one-hot vector using one-hot representation. This step is known as tokenization. There are additional tokens such as end of sentence <EOS> which represents end of sentence and unknown word <UNK>. If there are *w* words, each one-hot vector would have length of *w*. After tokenizing *w* words, the one-hot vectors are packed into a vocabulary matrix which has a size of $w \times w$. Note that there are more advanced tokenizing methods, one-hot representation is used for ease of explanation. In addition, the term token is used interchangeably with word because the vocabulary contains some non-word tokens such as <UNK> and <EOS>.

Given a sentence “Dinosaurs are extinct”, each word of the sentence is tokenized into respective one-hot vectors and <EOS> is added. They are denoted such that $y^{<1>}$ for “Dinosaurs”, $y^{<2>}$ for “are”, $y^{<3>}$ for “extinct” and $y^{<4>}$ for “<EOS>. The initial inputs to the recurrent neural network are $x^{[0]}$ and $a^{[0]}$. Each of them is a vector of zeroes. During training at each time step *t*, the recurrent neural network is used to

compute probability of each token in the vocabulary being the next token given the previous words. At each time step, the LSTM outputs a vector of size w where each element of the vector is a conditional probability for each unique token and can be formalized as stated in equation (2.34). In short, the aim of training is to use previous words to predict future words for each training examples and minimizes the discrepancies between prediction $\hat{y}$ and target y.

$$\hat{y}^{<t>} = P(\text{token } i \mid y^{<1>}, y^{<2>}, \cdots, y^{<t-1>}) \quad \text{for } i = 1,2, \ldots, w \tag{2.34}$$

After successfully training a recurrent neural network language model, the probability of a given sentence which is the same as the joint probability of each word sequentially in a sentence can be computed using equation (2.35).

$$P(y^{<1>}, y^{<2>}, \ldots, y^{<T_y>}) = P(y^{<1>}) \prod_{t=2}^{T_y} P(y^{<t>} \mid y^{<1>}, \cdots, y^{<t-1>}) \tag{2.35}$$

Basically, equation (2.35) consists of an initial probability for the first word, $P(y^{<1>})$ and other conditional probabilities based on previous words. Using the "Dinosaurs are extinct <EOS>" as an example, the probability of this sentence computed using equation (2.35) is given in (2.36).

$$\begin{aligned} &P(y^{<1>}, y^{<2>}, y^{<3>}, y^{<4>}) = \\ &\quad P(y^{<1>}) \times P(y^{<2>} \mid y^{<1>}) \times P(y^{<3>} \mid y^{<1>}, y^{<2>}) \times P(y^{<4>} \mid y^{<1>}, y^{<2>}, y^{<3>}) \\ &\text{where:} \quad y^{<1>} = \text{“Dinosaurs”}, y^{<2>} = \text{“are”}, y^{<3>} = \text{“extinct”}, \\ &\qquad\quad y^{<4>} = \text{“<EOS>”} \end{aligned} \tag{2.36}$$

2.7 Summary

One of the problems is that the dataset size of home activities for anomaly detection is relatively much smaller when compared to other machine learning dataset size. After reviewing different datasets as presented in Section 2.2, it was decided that the Aruba dataset is most suitable to be used in this research because it has largest data size. There are two categories of anomaly for home activities data namely sequential anomaly and time anomaly. For sequential anomaly detection, there are three existing

algorithms namely Markov Process, HMM and data mining algorithm. Out of these three algorithms, the best method is HMM. Our experimental results on Aruba dataset shown that HMM performed badly with an average accuracy of 50%. Data size limits the performance of the HMM algorithm. LSTM was investigated as an alternative of HMM because LSTM has more specialized components and may work better for small dataset. For time anomaly detection, there are three different existing algorithms namely Gaussian distribution, DBSCAN and combinational method of *K*-means clustering and Gaussian distribution. Gaussian distribution is not suitable because it cannot model an activity type with more than one cluster. Both DBSCAN and *K*-means are clustering algorithm that require specification of hyperparameters. For all three methods, each activity type is modelled separately. For DBSCAN and combinational method, each activity type may have more than one model. Therefore, it makes training procedure very cumbersome.

CHAPTER 3

SEQUENTIAL ANOMALY DETECTION

3.1 Introduction

In this chapter, two sequential anomaly detection methods are proposed to detect sequential anomalies in home activities data. They are LSTM method and database method. The objective of this research is to develop accurate and reliable anomaly detection algorithms which overcome the weaknesses of sequential anomaly detection algorithms mentioned in subsection 2.4.4. Figure 3.1 shows the research methodology framework for sequential anomaly detection. Section 3.2 covers the step of data collection. Section 3.3 covers the data preparation which include noise removal, data processing and data partitioning. Section 3.4 describes the algorithms of LSTM method and database method. In addition, a method to identify the activity which is causing the sequential anomaly is also given. In section 3.5, the details of model selection are presented. In section 3.6, the model evaluation step is discussed. Section 3.7 presents the experimental results and discussion. Lastly, section 3.8 summarizes this chapter.

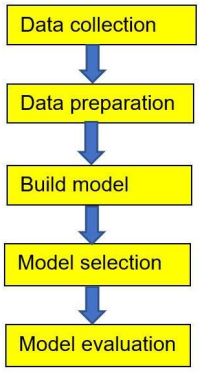

Figure 3.1: Methodology Framework for Sequential Anomaly Detection

3.2 Data Collection

Anomaly detection is a machine learning task, machine learning algorithm's performance usually improve with increases of data size. Hence, it is best to use the biggest dataset available to evaluate the anomaly detection model. The Aruba dataset was chosen out of all the datasets listed in Table 2.1 because it has largest data size.

In anomaly detection, there are two class labels namely normal and abnormal. The normal data points used are from the dataset. It consists of 220 days historical records of a volunteer's home activities. There are 11 types of home activities for this dataset such as sleeping, eating, enter home, leave home, housekeeping, toileting, prepare meal, wash dish, relax, work and respirate.

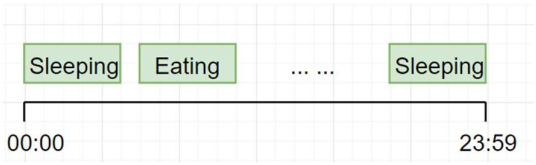

Figure 3.2: An Illustration of a Data Point

Each data point is a collection of activities happened in a day from 0:00 to 23:59 in the form of a sequence as illustrated in Figure 3.2. Thus, the size of data that can be extracted from the Aruba dataset is equivalent to the total number of days, which is 220. In this chapter, the terms data point and activity sequence are used interchangeably.

In this research, all the data points taken from Aruba dataset are categorized as normal points. The abnormal data points need to be artificially generated (Forkan, et al., 2015). The two methods used to generate abnormal data points are injection of activities and random shuffling. For injection of activities, normal sequences were injected with multiple toileting activities during sleeping hours to simulate nocturia symptom of diabetes. For random shuffling, the normal sequences were randomly shuffled using a function called *numpy.random.shuffle* of Python programming language. This simulates events of dementia patients randomized activities. Figure 3.3

shows a visualization of 12 normal data points dated 4 to 15 November 2010. Twelve abnormal data points were artificially generated from these 12 normal data points and is visualized in Figure 3.4.

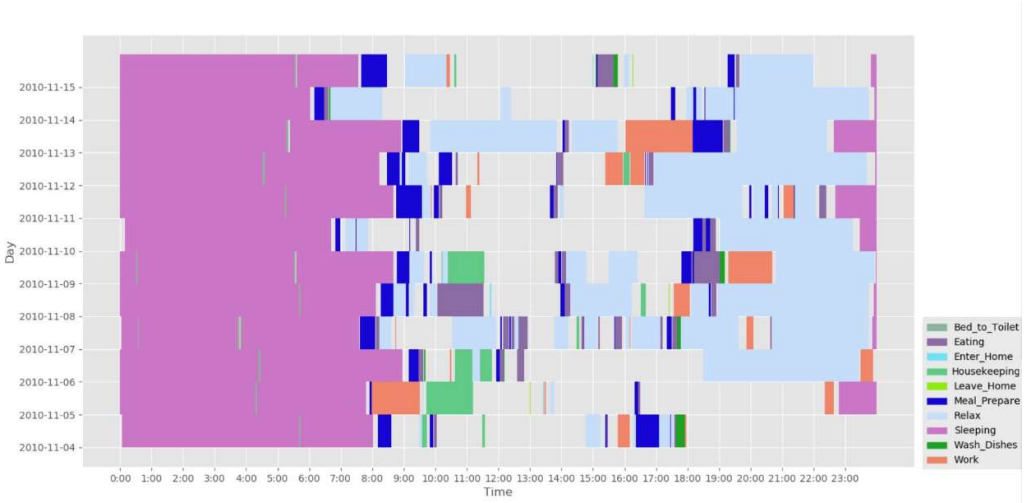

Figure 3.3: Visualization of Normal Dataset from 4 November to 15 November 2010

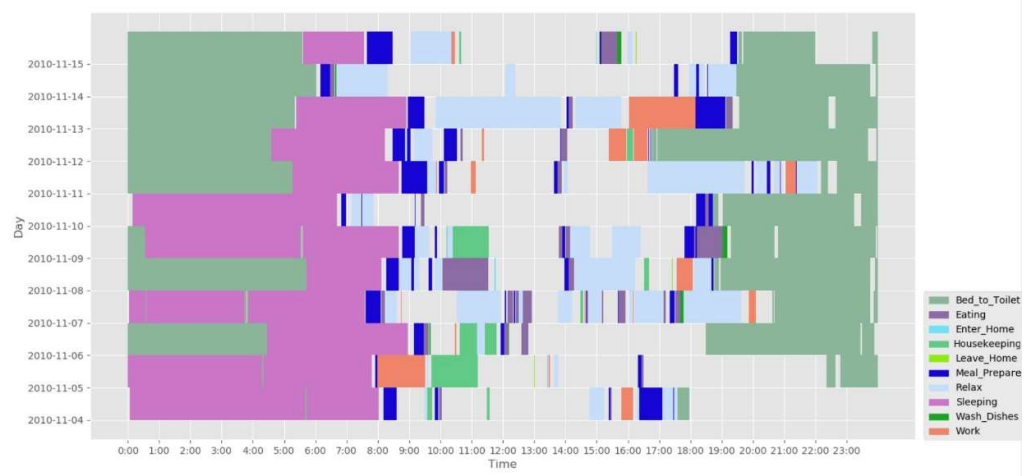

Figure 3.4: Visualization of Abnormal Dataset with Sequential Anomalies from 4 November to 15 November 2010

3.3 Data Preparation

Data preparation consists of three steps namely noise removal, data processing and data partitioning which are presented in subsection 3.3.1, 3.3.2 and 3.3.3 respectively.

3.3.1 Noise Removal

Noise refers to portion of data that has unwanted characteristics when compared to the rest. They are outliers and have different behaviour from normal data points. Noise affects the performance of the model. If these data points are used to train the model, the model which should only characterizes normal data points will be polluted with outliers. This may result in some real anomalies to be marked as normal. Therefore, it is important to remove noise from training set to ensure the performance of the model.

In the Aruba dataset, there are three types of noise. The first type is activity sandwiching where an activity happened in the duration of another activity between begin and ending of another activity. As an example, the sequence {*Prepare meal begin*, *relax begin*, *relax end*, *prepare meal end*} where the relax activity happens during meal preparation. This might be due mistake when annotating the dataset for activity recognition research. Regardless, it is rare and only appears in minority of the dataset.

The second type of noise is due to rare occurrences of the activity “Respirate”. It only occurred 6 times in the 220 days dataset as given in Table 3.1 which shows the number of occurrences of each type of activities in the Aruba dataset. Since the occurrences of “Respirate” is only 0.093% of total activity occurrences, it was deleted from the dataset.

Table 3.1: Types of Activities and Number of Occurrences

Type of activity	Number of occurrences
Prepare meal	1606
Relax	2910
Eating	257
Work	171
Sleeping	401
Dish washing	65
Toileting	157
Enter home	431
Leave home	431
Housekeeping	33
Respirate	6

Thirdly, the length of sequence or number of activities per daily sequence varies greatly. Some sequence has much more or much less activities when compared to the rest. Figure 3.5 illustrates the histogram of the frequency for different sequence lengths. It is noticed that most of the data points have sequence length in the range with minimum length of 10 and maximum length of 30. Therefore, during noise removal, any daily sequence with less than 10 activities or more than 30 activities were removed. After noise removal, the size of the remaining data is 189.

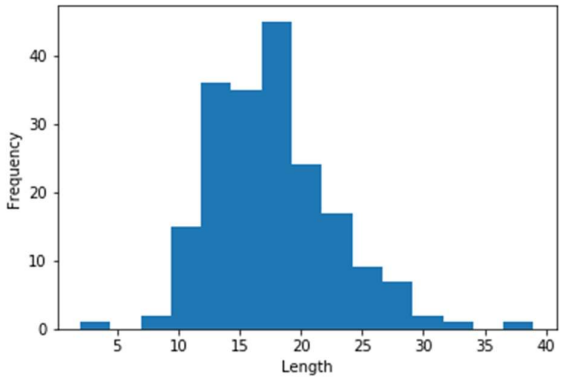

Figure 3.5: A Histogram of Frequency for Different Sequence Lengths

3.3.2 Data Processing

Data processing converts the data into format required by the anomaly detection algorithm. There are two types of sequential anomaly detection algorithms namely LSTM method and database method. The data processing for each algorithm is different. The data processing step of LSTM method is given in subsection 3.3.2.1. On the other hand, the data processing step of database method is presented in subsection 3.3.2.2.

3.3.2.1 Data Processing for LSTM Method

In language model, a <EOS> token is added to sentences to represent end of sentence. Similarly, for LSTM method, an <EOS> is added to the end of each data point to represent end of sequence. In this research, the term object is used to refer to the different types of activities and also <EOS>. In total, there are 11 distinct objects including ten different types of activity and <EOS>.

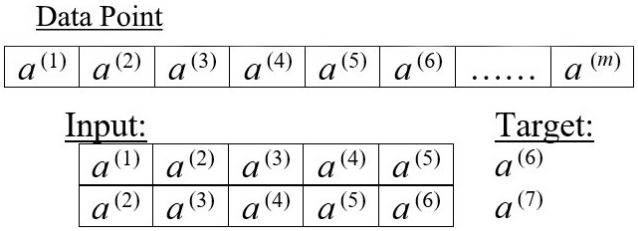

Figure 3.6: Illustration of Data Processing of a Data Point for LTSM with Window Size of 5

The next step of data processing is to process the data points into pairs of input and target to be used for training and inference. Inference means deploying the trained classifier into work. The first step is segmenting each data points into shorter segments using sliding window with the size denoted with *winsize* and step size of 1. The window size *winsize* is a hyperparameter. During the experiment, various choices in the {3, 5, 7, 9} were attempted and evaluated. The inputs would be the segments of *winsize* activities and the targets are activity or <EOS> next to each segment. Figure 3.6 illustrates the segmentation of a data point consists of *m* activities using a sliding

window with *winsize* = 5 as an example for explanation. The first input is the first 5 activities $\{a^{(1)}, a^{(2)}, a^{(3)}, a^{(4)}, a^{(5)}\}$ and the target is the sixth activity, $a^{(6)}$. This segmentation and processing into pairs of input and target step was repeated on every data point.

The input and target pairs consist of objects are categorical data which cannot be understand by computer. They have to be tokenized into numbers. The input data were tokenized using embedding layer and will be discussed in section 3.4. As for the target, the one-hot encoding was used to represent each type of object with a unique one-hot vector of length 11. Therefore, the number of possible predictions is 11.

3.3.2.2 Data Processing for Database Method

For database method, the data points were segmented into shorter segments using a sliding window of window size *winsize* and step size 1. Segmentation is to keep the input data with equal number of activities because each data point of Aruba dataset has different length. Similarly, the window size *winsize* is a hyperparameter and various choices were attempted during the experiment. For example, given a daily sequence of activities which consists of 5 activities {*sleeping*, *prepare meal*, *eating*, *wash dish*, *relax*} and using a window size *winsize* = 3, it can be split into 3 segments which are {*sleeping*, *prepare meal*, *eating*}, {*prepare meal*, *eating*, *wash dish*} and {*eating*, *wash dish*, *relax*}.

3.3.3 Data Partitioning

The final step of data preparation is data partitioning. During data partitioning, the dataset was partitioned into sets of different size for different purpose. These sets are training set, validation set and test set. They are useful at different stage of methodology.

The training set is used for building the anomaly detection model which captures the behaviour of normal data points. Therefore, the training set only consists of only normal data points. The normal data points used here are from Aruba dataset.

The validation set is used during model selection to select the model's hyperparameters. Each anomaly detection algorithm has different types of hyperparameters. The best setting of these hyperparameters is unknown. The systematic way would be to test a few models with different hyperparameters and chose the best one. The validation set is used to evaluate these models with different setting of hyperparameters. The details of model selection are given in section 3.5.

Last but not least, the test set is used during model evaluation. Model selection is the step which selects a model with the best settings of hyperparameters. The test set is used as a pool of data unseen to the algorithm to evaluate the performance consistency of the selected model. This is because the selected model may overfit to the data points in the validation set and cannot perform well on unseen dataset. Testing serves as a final evaluation to ensure the trained model works both on seen and unseen dataset.

In this research, the 189 normal data points were split into training set, validation set and test set with a ratio of 6:2:2. The number of abnormal data points is equivalent to the number of normal data points for both validations set and test set. The details of the partitioning are given in Table 3.2.

Table 3.2: Number of Normal and Abnormal Data Points for Training Set, Validation Set and Test Set

	Number of normal data points	Number of abnormal data points
Training set	113	–
Validation set	38	38
Test set	38	38

3.4 Building Model

In this section, two sequential anomaly detection algorithms for detecting sequential anomalies are presented. Subsection 3.4.1 discusses about the LSTM method and subsection 3.4.2 presents the database method.

Anomaly detection algorithms work by first building a model which characterizes normal data points. There is a unique score metric which can be used to measure the level of abnormalities of the data point. When given a data point, compute a score of that data point. Then, there is a threshold associated with that model which is used as a cutting point which separates score of normal data points and abnormal data points. The details about selecting suitable threshold is given in section 3.5. In each subsection, each method is described and the associated score metric for classification is also given.

3.4.1 Sequential Anomaly Detection using LSTM

The model architecture of the sequential anomaly detection method was inspired by language model. In subsection 3.4.1.1, the concepts of language model that inspired this anomaly detection method are presented. Preliminary results revealed using language model's many-to-many model architecture yield an unsatisfactory performance due to small training set size. Therefore, a modified model architecture was designed to deal with this problem. The new model architecture designed specifically for anomaly detection task is presented in subsection 3.4.1.2. In subsection 3.4.1.3, the score metric of this new model architecture is presented. In subsection 3.4.1.4, the method for identifying activities causing anomalous sequence is given.

3.4.1.1 Inspirations and Design Considerations

There are two main ideas from language model that had inspired this anomaly detection algorithm. The first idea is the end of sequence <EOS> token of language model which is appended to the sentence to represent end of sentence. This concept is also applied in this anomaly detection algorithm to represent end of sequence. During data processing, a <EOS> token was added to the end of every activity sequence. The purpose is to ensure the LSTM can learn when and how the sequence should end. The second idea originated from language model is the probability of sentence. The probability of a sentence measures the likelihood of that sentence. Similarly, probability of an activity sequence can be computed and used as a score metric which measures the likelihood of that sequence. This is useful for anomaly detection. If the

pattern of a given sequence is similar to the pattern of training data points, the computed probability of sequence should be high. Conversely, if the pattern of a given sequence is anomalous or dissimilar to the pattern of training data points, the computed probability of sequence should be low.

Unfortunately, using the model architecture of language model directly for anomaly detection does not work because the relatively small training set size. Preliminary experiment result demonstrated that the average accuracy of anomaly detection model trained using whole sequences as inputs without any segmentation with sliding window is 59.87%. Therefore, the main design consideration would be to reduce the reliance of the performance on the training set size. This can be done by reducing the complexity of the problem.

One of the two devised strategies is segmenting data points into shorter segments with fixed length. The main difference with language model is that the training set consists of segmented sequences instead of using whole sequence of a data point as input. This is due to the relatively small training set size and variation in length of data points. First of all, segmentation increases the data size several fold. The magnitude of increases depending on *winsize* used. Segmentation into shorter sequences also reduces the complexity of the problem. Secondly, fixing input length reduces the complexity of the problem because it is easier to learn the pattern from a fixed shorter segment of activities. Variation in length of original data points result in much more possible outcomes in output. For example, let's say there are three different length denoted with $T1$, $T2$ and $T3$. If a many-to-many LSTM architecture like that of a language model is used, the number of possible predictions would be $11^{T1} + 11^{T2} + 11^{T3}$.

Due to this modification in data format, the model architecture of the LSTM used for anomaly detection is different from the language model. The details about the modified LSTM model architecture is given in subsection 3.4.1.2.

For anomaly detection, a score metric is needed to measure the level of abnormalities of a given data point. The initial idea was to use a score similar to probability of sentence of language model. The probability of sentence measures how likely is a given sequence of activities. However, preliminary result using a many-to-many architecture similar to language model demonstrated poor performance due the small data size. A LSTM model architecture modified from the language model therefore developed to counter this problem. As a result, the score metric of language model cannot be used. A new score metric was devised and presented in subsection 3.4.1.3.

3.4.1.2 Model Architecture

The flow chart on the left of Figure 3.7 illustrates each layer of the model architecture of the sequential anomaly detection using LSTM which consists of embedding layer, LSTM layer and vanilla neural network layer. The block diagram on the right of Figure 3.7 illustrates the multilayer many-to-one architecture of LSTM layer connected to other layers. In this section, each of the layers of the anomaly detection model is described. This model was implemented Google's Keras deep learning library in Python programming language which used Tensorflow as its backend (Chollet, 2017; Summerfield, 2008; Galea & Capelo, 2018).

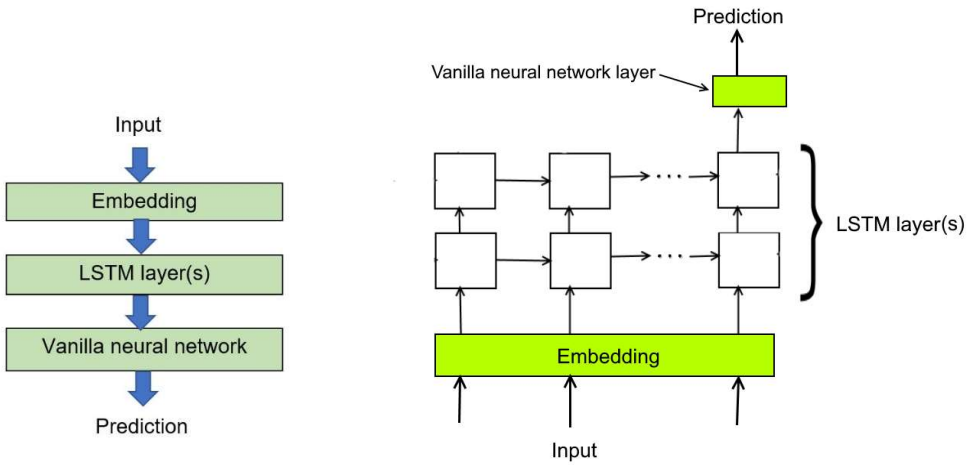

Figure 3.7: Illustration of Each Layer of Model Architecture of the Sequential Anomaly Detection using LSTM

The first layer is embedding layer. Embedding layer converts non-numerical input into vectors of real numbers. The data points consist of objects which include the 10 distinct types of activity and <EOS> which are categorical. Computers cannot understand categorical data directly. The one-hot encoding discussed in Chapter 2 is a type of tokenizing technique which represent each category with a one-hot vector of one and zeroes. Embedding is a more advanced tokenizing technique used in natural language processing which maps distinct words to vectors of real numbers. There are multiple embedding algorithms such as *Word2vec* (Mikolov, Chen, Corrado, & Dean, 2013) and *n-gram* (Franz & Brants, 2006). The embedding layer will generate a unique embedding vector of length E for each type of object (activities or token). In this research, the embedding was implemented in Python using *keras.layers.Embedding* function. The input data to the embedding layer is a vector with length *winsize*, where each element of that vector is an activity. The embedding layer converts each activity to its respective embedding vector of length E. This results in an output matrix with size of ($winsize \times E$) where E denotes the length of the embedding vector. Several choices of E were attempted. It was discovered that choice of E does not affect end result significantly for $E = 5$ and $E = 10$. However, the results may differ if larger choice of E is used. The choice of $E = 10$ was selected to present research findings.

The LSTM layer is the most important layer of LSTM method since it is the layer that learns the pattern of sequences. In this research, the LSTM layer was implemented using *keras.layers.CuDNNLSTM* function. LSTM learn the pattern of the sequences by using *winsize* activities to predict the next object. It consists of one or more many-to-one LSTM layer. The number of LSTM layer is a hyperparameter. The total number of time step is equivalent to the sliding window size used during data processing and is denoted with *winsize*. Instead of using many-to-many architecture like language model, a many-to-one architecture was selected. This is due to the modification of data formats. Using many-to-one architecture, the LSTM uses several previous activities to predict only one next activity at the final time step instead of predicting at every time step. For many-to-many LSTM architecture, the number of possible predictions is 11^T where T denotes the length of output. For many-to-one

LSTM architecture, the number of possible predictions is 11. This drastic reduction in number of possible outcomes reduces the complexity of the problem.

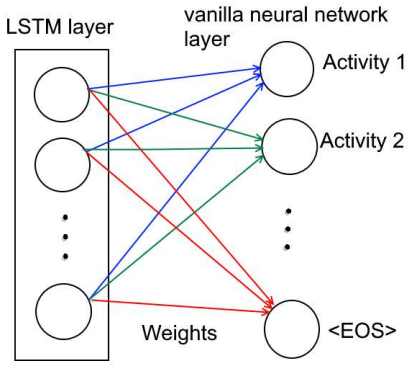

Figure 3.8: Illustration of Connection Between Last Time Step of LSTM Layer and the Vanilla Neural Network Layer

The next layer of LSTM method is a vanilla neural network layer with 11 nodes connected to final time step of the LSTM layer as shown in Figure 3.8. The purpose of vanilla neural network is predicting the next object. The neural network layer takes activations from the final time step of LSTM layer as input and outputs a prediction vector of length 11 denoted with p. The prediction task was framed as a 11-class classification problem. Each element of the prediction vector p can be interpreted as the conditional probability since the target data are one-hot vectors. It was implemented in Python using *keras.layers.Dense* function with a Softmax activation function.

LSTM method has several hyperparameters which include LSTM architecture, sliding window size used for segmentation and threshold choice. The LSTM architecture refers to number of LSTM layers and number of hidden units per LSTM layer. The choices of number of layers of LSTM tried out are one and two. The number of hidden units tried out are 32 and 64. The number of layers and number of hidden units because it can cause the model to overfit to the small dataset. This results in 4 different LSTM architecture to be evaluated as listed in Table 3.3 labelled with $A1$, $A2$, $A3$ and $A4$. For example, architecture $A3$ consists of 2 LSTM layers and there are 64

hidden units per LSTM layer. The different sliding window sizes attempted are 3, 5, 7 and 9. The sliding window size cannot be larger than 10 because that is the minimum length of data points. It does not make sense to segment a data point with length smaller than the window size. The way to sample threshold is given in section 3.5.

Table 3.3: Different LSTM Architectures as Hyperparameter

Architecture	Number of LSTM layers	Number of hidden units
*A*1	1	32
*A*2	1	64
*A*3	2	32
*A*4	2	64

3.4.1.3 Score Metric

Score metric is used to measure the level of abnormalities of a given data point. Given a data point, first segment into input and target pairs. For each pair of input x and target y, the trained LSTM model takes in the input and outputs a prediction vector p. Then, a difference vector d can be subtracting the prediction vector p from the target vector y as given in equation (3.1). Compute the difference vector for every pair of input and target of the data point. After that, a score is calculated by taking L^1 norm of every difference vectors, sum them up and normalize the summed value with $\frac{1}{n}$ as given in equation (3.2). The lowercase n denotes the number of pairs of input and target that can be obtained from the given data point.

$$d = y - p \tag{3.1}$$

$$\text{score} = \frac{1}{n}\sum L^1(d) \tag{3.2}$$

$$\begin{aligned} &\text{if score} > \textit{threshold} \text{ then abnormal} \\ &\text{else if: score} \leq \textit{threshold} \text{ then normal} \end{aligned} \tag{3.3}$$

The score metric can be interpreted as a function which measure the prediction error. The idea is that if the given data point is similar to the sequences in the training set, then the computed score should be small. Conversely, if the score of a data point is big, this means the data point has very different pattern from the training sequences.

A threshold is used for anomaly detection as given in (3.3). If the score of the data point is greater than the threshold, it is classified as abnormal. Conversely, if the score is less than or equal to threshold, it is classified as normal.

3.4.1.4 Identifying Activity which Caused Anomalous Sequence

Each data point refers to a daily sequence of activities from 0:00 to 23:59. The proposed LSTM method and its score metric can be used to classify a given data point as normal or anomalous. During data processing, the data points with less than 10 activities or more than 30 activities were deleted. Therefore, each data point has about 10 to 30 activities. When a data point is classified as anomalous, it does not mean that every activity of that sequence is out-of-sequence. The source causing the anomalies should be only some of the activities of that sequence instead of every activity of that sequence. It is useful if there is a way to track which activities out of entire sequence is causing the anomaly.

The procedure of identifying out-of-sequence activities within the sequence causing the anomaly consists of 3 steps. First step is to process a given data point into pairs of input and target. The second step is to obtain the difference vector *d* of length 11 for each input and target pair using equation (3.1) which subtract the prediction vector from the target vector. Each value in the difference vector is converted to absolute value. Then, a mean is calculated for every absolute difference vector. The segment of *winsize* activities with the largest mean is the source of anomaly. The idea behind this is that segment which causes anomaly should have relatively larger prediction error. Each difference vector is actually the prediction error of a pair of input and target of a data point. If the segment of activities used as input is out-of-sequence, the prediction error should be large. As a result, the mean of that difference vector should be large as well.

3.4.2 Sequential Anomaly Detection using Database

Figure 3.9 illustrates each component of the anomaly detection model which includes database and anomaly detector. This method is called database method

because it consists of a database. The database models a person's sequence pattern of home activities by saving all the segmented sequences of training set in a CSV file. During data processing, each data point was broken down into shorter segments of fixed length where each segment consists of *winsize* consecutive activities where *winsize* is the window size of the sliding window. Window size will affect the performance of this method. This is because this method memorizes every segment of the training data points. If larger window size is used, there will be more possible variation in possible combination of activities. For example, window size 5 will result in 5^{10} possible combinations of 5 consecutive activities. Power 10 because there are 10 types of activities. As a result, there will be more combinations for the database method to memorize and results in much poorer performance.

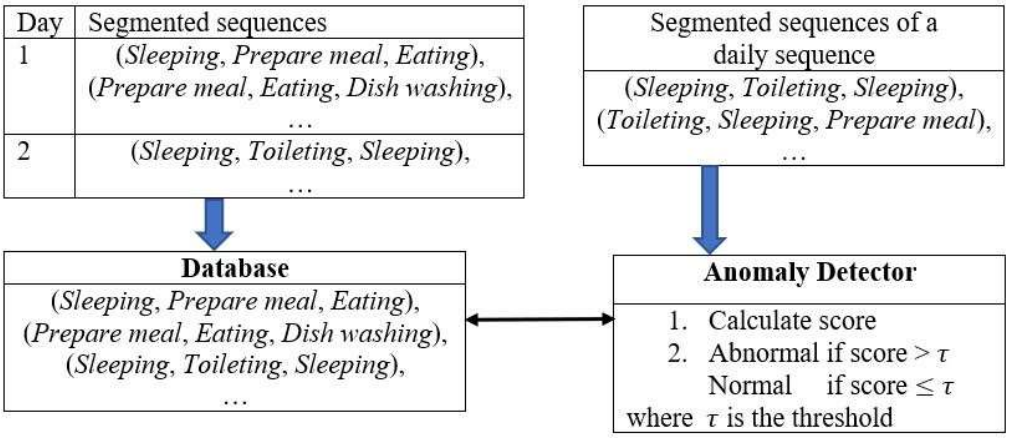

Figure 3.9: Illustration of the Components of the Sequential Anomaly Detection Model using Database

The anomaly detect decides whether a given data point is anomalous or not. Given a data instance, it is first segmented using sliding window. Then, there is a checking process to find out whether each segment of the data point exists in the database or not. If a segment is not saved in the database, then it is considered an anomalous segment. There is a counter which keep track of the number of abnormal segments. After checking process ends, a score is calculated for that data point using equation (3.4). The score metric is the percentage of abnormal segments out of all segments available for that data point. The anomaly detection is also using a threshold

to divide between scores of normal data points and anomalous data points. Similar to classification used by LSTM method, the classification of data points of this method follows (3.3).

$$\text{score} = \frac{\text{Number of abnormal segment}}{\text{Total number of segments}} \times 100\% \tag{3.4}$$

3.5 Model Selection

Each anomaly detection algorithms have different hyperparameters. For example, one of the hyperparameters for Hidden Markov Model (Forkan, et al., 2015) is the number of hidden states. The rest of the hyperparameters of the sequential anomaly detection algorithms are detailed in Table 3.4. The best settings of these parameters are not known because there is no analytical method to solve them. As mentioned in section 3.3, the data points in the validation set are used to measure performance of different models, each built with different hyperparameters initialization. The proper way of initializing hyperparameters are by sampling a few choices, build these different models, evaluate them using validation set and choose the best performing model.

Table 3.4: The Hyperparameters of Sequential Anomaly Detection Algorithms

Algorithms	Hyperparameters
HMM (Forkan, et al., 2015)	Number of hidden states and threshold
LSTM	Sliding window size, number of units per LSTM layer, number of LSTM layers, threshold
Database	Sliding window size and threshold

The threshold is a hyperparameter common for different sequential anomaly detection algorithms. Each algorithm has a specific score metric used to measure the degree of abnormalities of a given data point. For example, the log-likelihood score of HMM is one such score. Threshold is the cutting point which separates scores of normal data points and abnormal data points. The threshold sampling consists of 2 steps. First, calculate a score for each data points in the validation set. Secondly, sample T values linearly from the range with minimum and maximum of the range

being the minimum and maximum of the calculated scores. In the experiment, the number of threshold choices, T is 100.

Table 3.5: Confusion Matrix

		Predicted	
		Positive	*Negative*
Actual	*Positive*	True positive, *TP*	False negative, *FN*
	Negative	False positive, *FP*	True negative, *TN*

$$\text{precision} = \frac{\text{number of } TP}{\text{number of } TP + \text{number of } FP} \tag{3.5}$$

$$\text{recall} = \frac{\text{number of } TP}{\text{number of } TP + \text{number of } FN} \tag{3.6}$$

The performance metric used for model selection is $F1$ score. It is derived from the confusion matrix as shown in Table 3.5. For anomaly detection, positive refers to the abnormal class label whereas negative refers to normal class label. The confusion matrix consists of 4 values including true positive (TP), false positive (FP), false negative (FN) and true negative (TN). TP means actual positive data points that were correctly classified as positive. FP means actual negative data points that were misclassified as positive. FN means actual positive data points that were misclassified as negative. TN means actual negative data points that were correctly classified as negative. From the confusion matrix, precision and recall can be derived as shown in equation (3.5) and equation (3.6). Precision is a ratio of the number of correctly classified positive data points to the total number of data points classified as positive. On the other hand, recall is a ratio of the number of correctly classified positive data points to the total number of actual positive data points.

A good model does well on precision and recall. Hence, the $F1$ score (Sasaki, 2007) which is a balanced measure of precision and recall is chosen. The $F1$ score can be calculated using equation (3.7). For each model, the $F1$ score is calculated. The model with the highest $F1$ score is selected as the best model.

$$F1 \text{ score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 100\% \quad (3.7)$$

3.6 Model Evaluation

Model evaluation evaluates the performance of the model selected during model selection using an unseen test set to ensure consistency in performance. There are cases where a model works very well on validation data but performs badly on unseen data when it is deployed. This type of problem is called overfitting. It is desirable to have a model which performs excellent not only on validation set but can also have similar performance on unseen data points. In other words, the performance of the model should generalize well on the test set. To measure the performance, the performance metric used is accuracy. Accuracy is a percentage of correctly classified data points out of all data points as given equation (3.8).

$$\text{Accuracy} = \frac{\text{Number of correctly classified data points}}{\text{Total number of data points}} \times 100\% \quad (3.8)$$

3.7 Experimental Results and Discussion

This section includes the experiments carried out to validate the performance of the LSTM method and database method. The LSTM method used for sequential anomaly detection was inspired by language model. One of the inspirations is addition of <EOS> to the end of the sequences. Besides that, a score metric was designed for the modified LSTM algorithm. Evaluations of the necessity of adding <EOS> and the suitability of the score metric are given in subsection 3.7.1. One of the hyperparameter of data processing for LSTM method is the window size. In subsection 3.7.2, the experimental results on various choices of window size for LSTM method are presented. In subsection 3.7.3, the performance of LSTM method using different LSTM architectures is presented and discussed. In subsection 3.7.4, experimental results on various choices of window size for database method are presented. In subsection 3.7.5, the performance of database method is given. Then, the comparison between LSTM method, database method and HMM method is presented in subsection

3.7.6. In subsection 3.7.7, the performance of the method to identify activities which caused the anomalies using LSTM method is discussed.

3.7.1 Suitability of Design of LSTM Method

A. Suitability of using <EOS> Token

One of the key features of LSTM anomaly detection method is adding an end of sequence token <EOS> to each sequence in the training set to represent end of sequence. Figure 3.10 shows the scores of the normal and abnormal data points in validation set for two models with and without using <EOS>. There are two models: the first model with architecture *A*4 and window size 3, the second with architecture *A*4 and window size 5. Figure 3.10 (a) shows the scores computed using the first model without using <EOS>. Figure 3.10 (b) shows the scores for the first model using <EOS>. Figure 3.10 (c) shows the scores for the second model without using <EOS>. Figure 3.10 (d) shows the scores for the second model using <EOS>.

Models which were trained with data points consisting of <EOS> can distinguish abnormal data points from normal data points better than those do not use <EOS>. This is evidenced by the clearer gap between scores of normal data points and scores of abnormal data points in Figure 3.10 (b) and Figure 3.10 (d) compared with their counterpart in Figure 3.10 (a) and Figure 3.10 (c) which did not use <EOS>. Visibly, there is a clear gap and a threshold can act as a straight line to divide these two types of scores. On the other hand, the models trained with data points without using <EOS> resulted in scores of normal data points and abnormal data points overlapping with each other. There is no straight line which can divide them well. This concludes that using <EOS> can enhance the performance of the LSTM anomaly detection model.

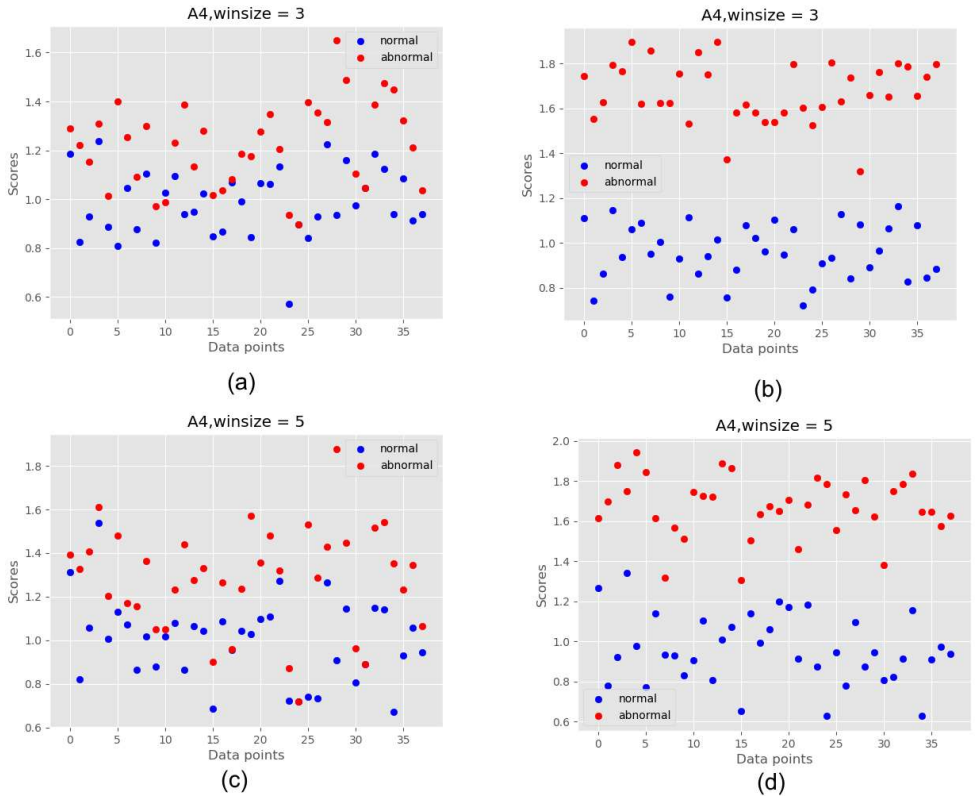

Figure 3.10: Scores of Normal Data Points and Abnormal Data Points of Validation Set (a) Model *A*4 and *winsize* = 3 Without Using <EOS>, (b) Model *A*4 and *winsize* = 3 Using <EOS>, (c) Model *A*4 and *winsize* = 5 Without Using <EOS>, (d) Model *A*4 and *winsize* = 5 Using <EOS>

B. Suitability of Score Metric

Figure 3.11 and Figure 3.12 consists of graphs of scores calculated for data points in validation set for 16 models with one of the combinations of hyperparameters (Architecture: *A*1, *A*2, *A*3, *A*4 as listed in Table 3.3 and window size: 3, 5, 7, 9). The normal scores and abnormal scores form 2 separate clusters with some points overlapping. For all of the graphs, these two clusters are at a far enough distance apart one another which allows for a threshold to be used for classifying normal and abnormal scores. In conclusion, the score metric devised for LSTM method is an effective metric to measure the level of abnormalities of data points.

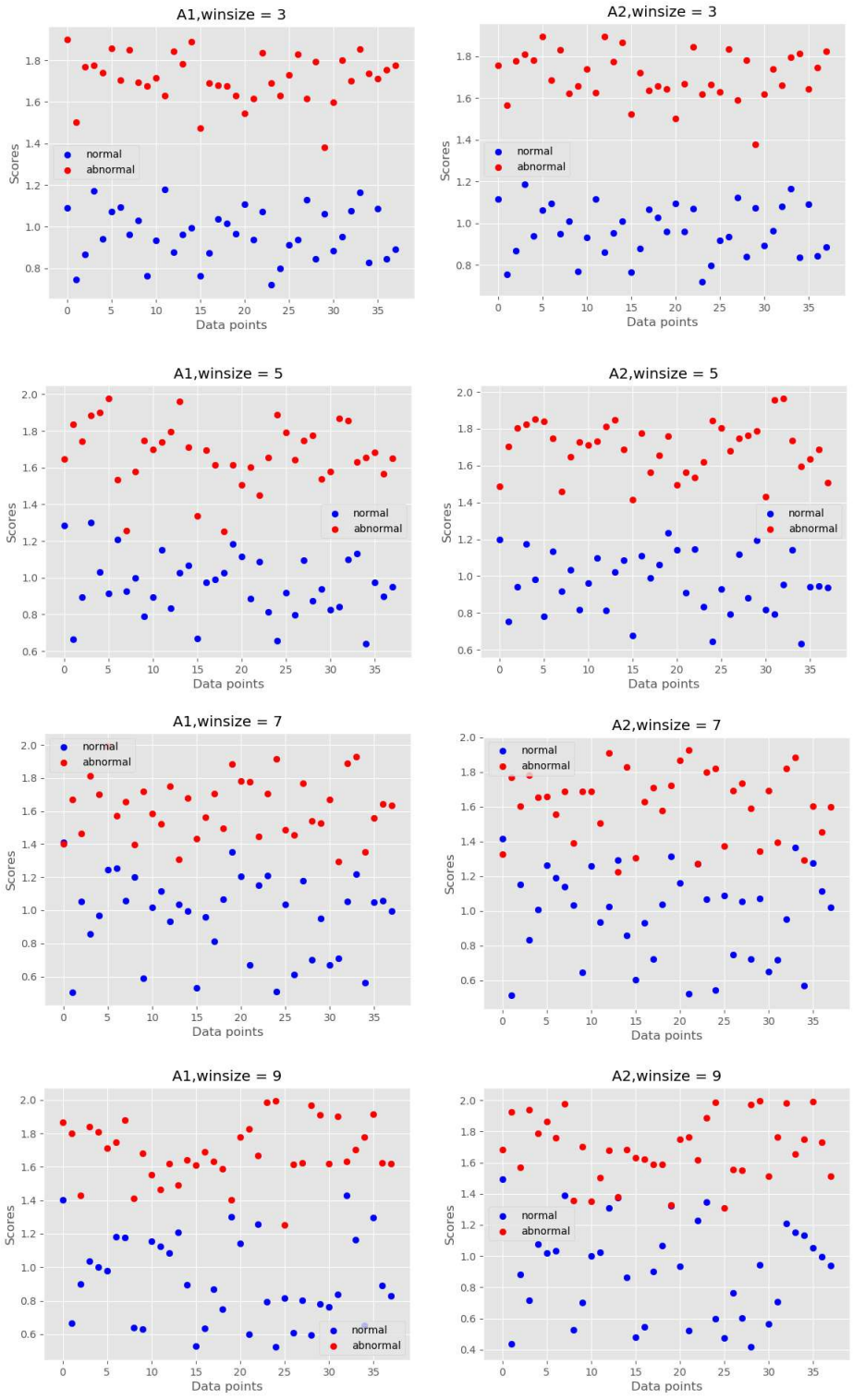

Figure 3.11: Scores of Data Points in Validation Set for LSTM Models with Different Hyperparameters (Architecture: *A*1, *A*2; Window Size: 3, 5, 7, 9)

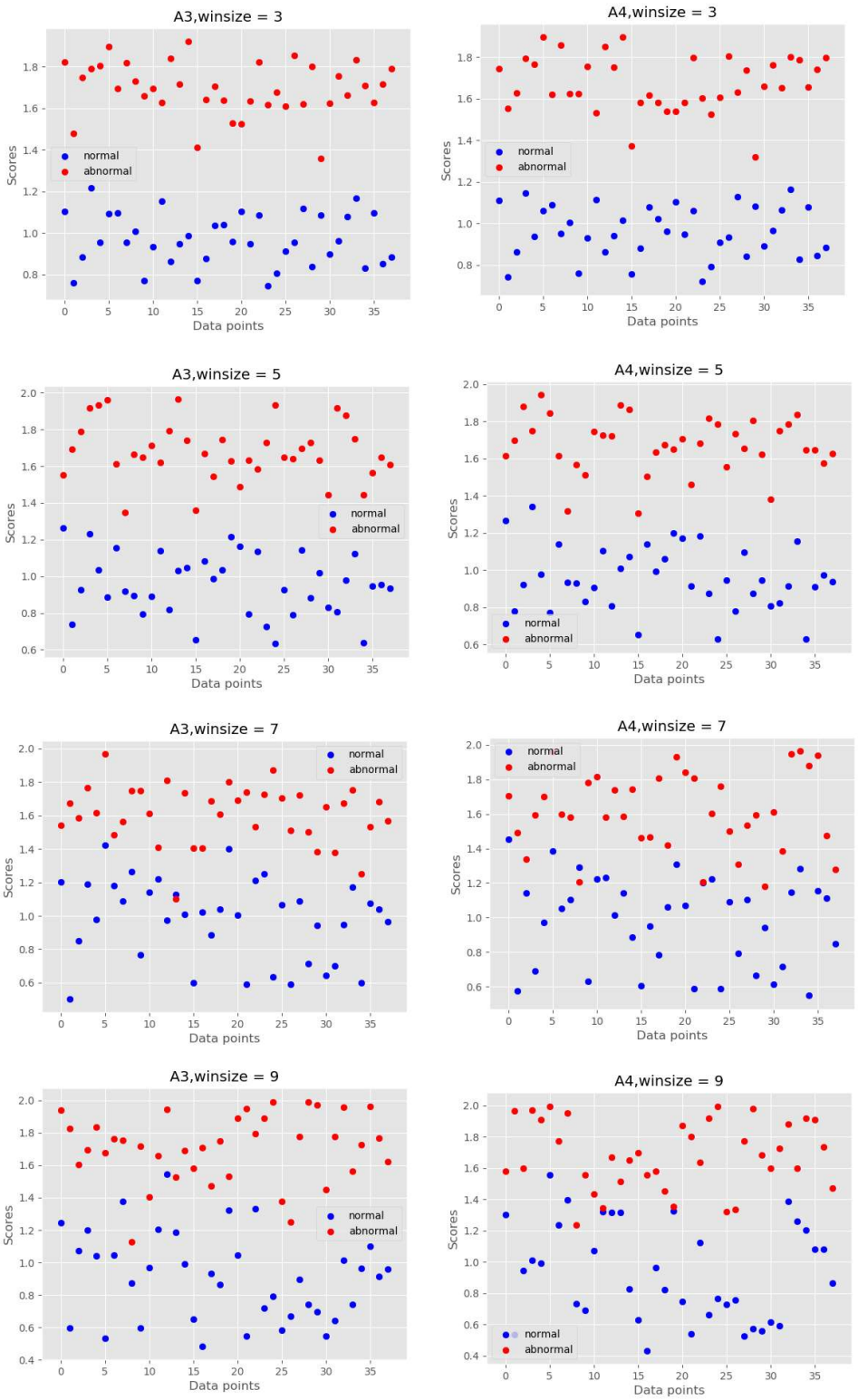

Figure 3.12: Scores of Data Points in Validation Set for LSTM Models with Different Hyperparameters (Architecture: *A*3, *A*4; Window Size: 3, 5, 7, 9)

3.7.2 Choice of Window Size on Performance of LSTM Method

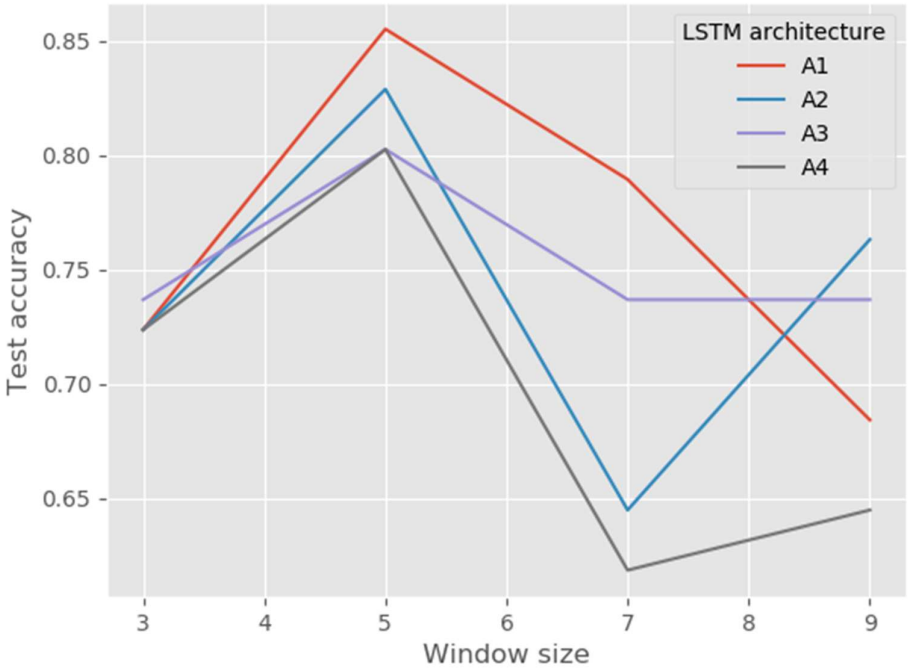

Figure 3.13: Graph of Test Accuracy vs. Window Size for Different LSTM Architectures

Window size of sliding window used for segmentation during data processing is also a hyperparameter. There is no way of knowing which choice of window size would work best prior to evaluation. Four window sizes including 3, 5, 7 and 9 were evaluated. Figure 3.13 graphs the effects of window size on test accuracy for different LSTM architectures $\{A1, A2, A3, A4\}$ where each architecture has a unique colour. The graph shows that window size of 5 is the best choice for window size because the test accuracy for different architecture is highest when window size is 5.

3.7.3 Performance of LSTM Method

To evaluate the performance of different LSTM architectures in the set of {$A1$, $A2$, $A3$, $A4$} as listed in Table 3.3, the window size is fixed to the best choice *winsize* = 5 and each model was trained with data points with <EOS>. During model selection, $F1$ score was used to select the best threshold choice for each architecture out of 100 threshold choices. Then, these chosen models were evaluated using data points in the test set. The accuracy metric was used to measure their performance on the test set.

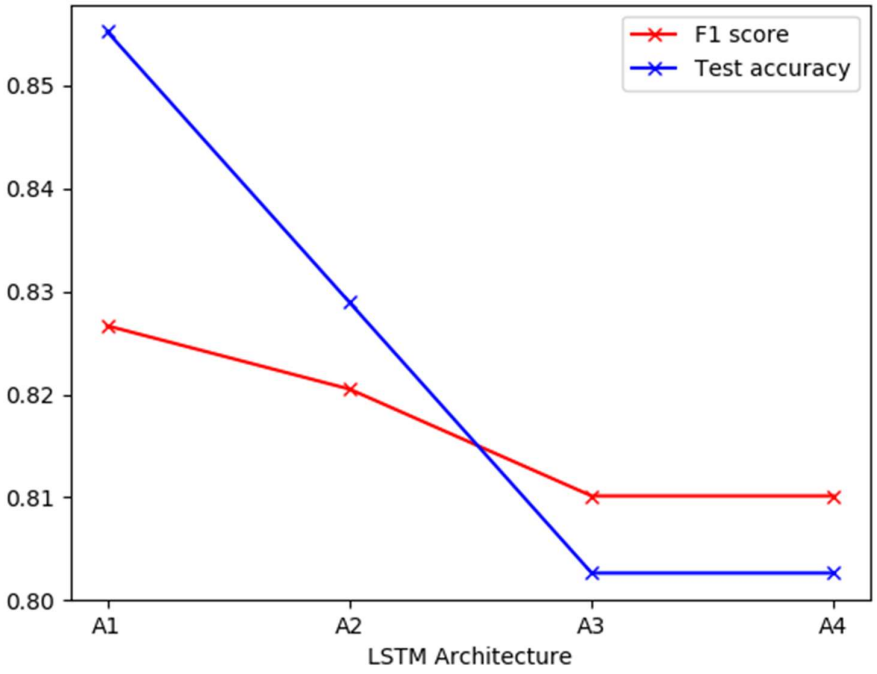

Figure 3.14: Graph of *F*1 Score and Test Accuracy vs. Different LSTM Architecture for *winsize* = 5

Table 3.6: Number of Trainable Parameters for Each LSTM Architecture

LSTM architecture	Number of parameters
$A1$	6095
$A2$	20271
$A3$	14543
$A4$	53551

Figure 3.14 consists of a graph of $F1$ score and test accuracy for best models of different LSTM architecture. The $F1$ score and test accuracy for all LSTM

architectures are in the range of 0.80 to 0.86, which is excellent for this task considering the limitation of training set size. The best performing LSTM architecture is $A1$, with highest $F1$ score of 82.67% and a highest test accuracy of 85.53%. LSTM model using $A1$ and window size of 5 is the best overall model for LSTM and its performance metrics on validation set and test set is given in Table 3.7. LSTM architecture $A1$ has the smallest number of trainable parameters out of all the LSTM architectures as given in Table 3.6. The results show that the performance is not dependent on the complexity of the LSTM architecture. In addition, the results demonstrated that the modified LSTM model has good performance in terms of test accuracy. This validates that the design of LSTM method can perform sequential anomaly detection despite trained with relatively small training set size.

Table 3.7: Performance Metrics for the Overall Best LSTM Model (*winsize* = 5, with <EOS> and Using *A*1 Architecture)

	Validation (%)	**Test (%)**
Precision	83.78	93.55
Recall	81.58	76.32
***F*1 score**	82.67	84.06
Accuracy	82.89	85.53

3.7.4 Choice of Window Size on Performance of Database Method

The choice of window size can affect the accuracy of the database method. An experiment was conducted using K-fold cross validation to study the window size effect. In this experiment, the total number of folds $K = 4$. This means there are four different combination training and validation set. The folds are indexed using $k = 1, 2, 3, 4$. Figure 3.15 shows the graph of test accuracy for different folds of dataset and window size choices $\{3, 5, 7, 9\}$. For the first fold ($k = 1$), the test accuracy is highest when window size 7 was used. For the second fold ($k = 2$), window size 3 and 5 has the same highest test accuracies. For the third and fourth fold, the window size of 3 has the highest accuracy. The best window size is 3 since it has the highest accuracies for $k = 2, 3, 4$.

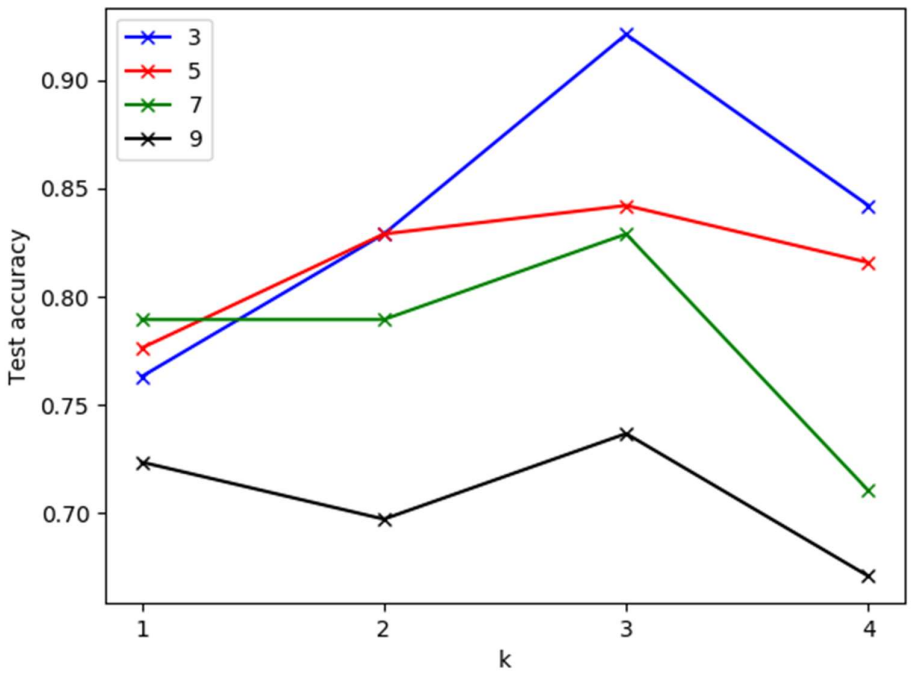

Figure 3.15: Test Accuracy for Varying Window Size and Fold *k*

3.7.5 Performance of Database Method

Figure 3.16 shows the scores of normal data points and abnormal data points in the validation set. Visually, there is clear gap between scores of normal data points and abnormal data points. This validates the score metric as being suitable to measure level of abnormalities of a given dataset.

Figure 3.17 shows the $F1$ scores of models using different choices of threshold. Visually, the $F1$ score is highest at the range from 0 to 10, from there onwards it drops as the choice of threshold increases in magnitude. The best performing model has a $F1$ score of 97.37% and used a threshold choice of 11.36%.

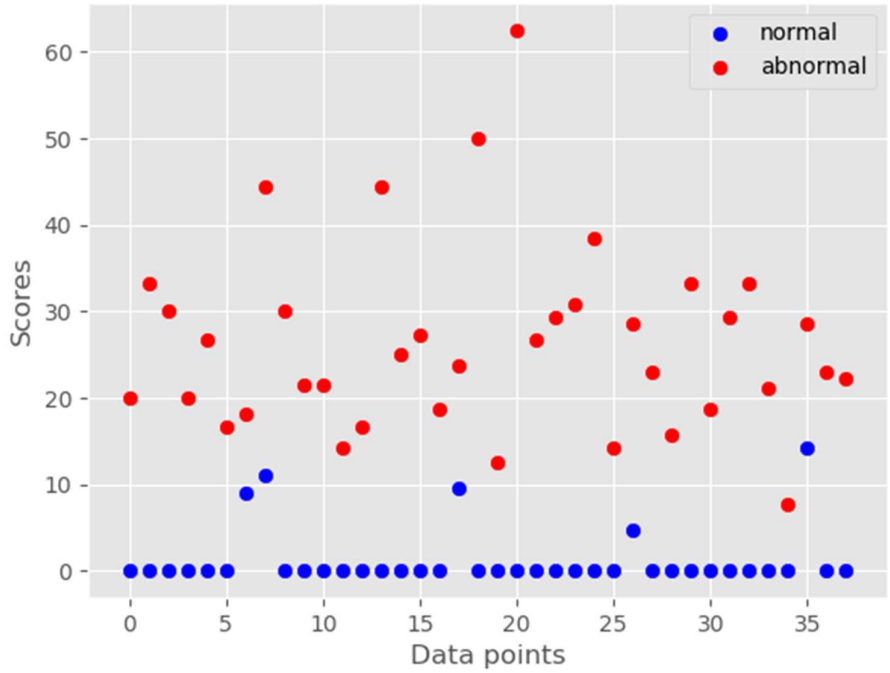

Figure 3.16: Scores of Normal Data Points and Abnormal Data Points in the Validation Set Computed Using Database Method

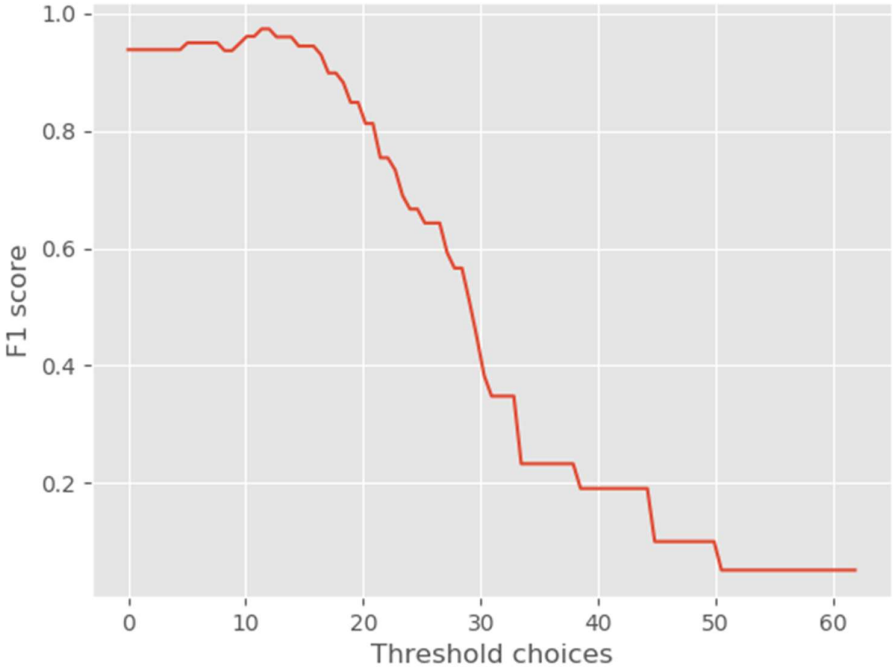

Figure 3.17: Graph of *F*1 Score vs. Threshold Choice for Database Method

Table 3.8: Performance of the Overall Best Model for Database Method

	Validation (%)	Test (%)
Precision	97.37	90.65
Recall	97.37	76.32
F1 score	97.37	82.86
Accuracy	97.37	84.21

Table 3.8 shows the performance metrics of the overall best model for database method on validation set and test set. There is a significant difference between accuracy of validation set and test accuracy. The much higher performance metrics of database method on validation set was probably due to it saves every possible normal segment in training set.

Table 3.9: Test Performance of Overall Best Model for LSTM Method and Overall Best Model for Database Method

	Database method	LSTM method	Difference
Precision	90.65	93.55	2.9
Recall	76.32	76.32	0
*F*1 score	82.86	84.06	1.2
Accuracy	84.21	85.53	1.32

Table 3.9 shows the performance of overall best LSTM model and overall best database model on test set. The difference in performance metrics of both methods on test set for both methods is small as given in the "Difference" column of Table 3.9. In conclusion, database method works with a comparable performance as the LSTM method in terms of precision, recall, *F*1 score and accuracy on test set.

3.7.6 Comparison of Sequential Anomaly Detection Algorithms

Finally, an experiment was conducted to compare the performance of four different sequential anomaly detection algorithms using *K*-fold cross validation. In this experiment, $K = 4$ was used. The algorithms under comparison include the LSTM method, database method, HMM method and LSTM language model. The LSTM method and database method are described in subsection 3.4.1 and 3.4.2 respectively. The HMM method described in the literature (Forkan, et al., 2015) was reproduced.

Whereas, the LSTM language model described in subsection 2.6.5. The difference between LSTM language model and LSTM method is that LSTM language model used data directly without segmentation. Whereas, the data points for LSTM method was segmented. Figure 3.18 shows that the test accuracies of LSTM method and database method are significantly higher than the test accuracies of HMM method and LSTM language model for all four different fold.

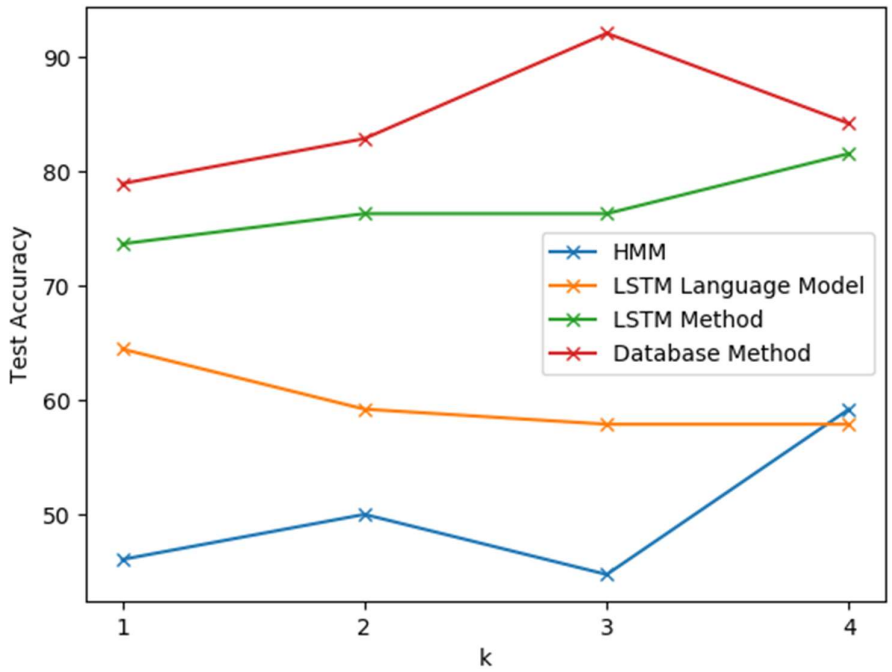

Figure 3.18: Comparison of Test Accuracies of Different Sequential Anomaly Detection Algorithms

The average test accuracy of LSTM method, database method, HMM method and LSTM language model are 76.98%, 84.54%, 50% and 59.87%. Using modifications described in subsection 3.4.1.2, LSTM method improved in terms of accuracy for 17.11% when compared to 59.21% of LSTM language model. For this experiment, the average test accuracy of LSTM method and database method differs about 7.56%.

3.7.7 Identify Abnormal Activities causing Anomaly using LSTM Method

To test the capability of the LSTM method to identify activities caused anomalies, 38 anomalous data points were artificially generated from the normal data points in the test set by randomly shuffling the order of final 5 activities. The detection criterion is if the input segment with the largest mean of difference vector consists of the shuffled activities, then the source of anomaly is correctly identified. The accuracy of using LSTM method to identify which activity in the anomalous sequence caused anomalies for 38 anomalous data points generated from test set is 89.47%.

3.8 Summary

In this chapter, two new sequential anomaly detection algorithms are introduced. The LSTM method was inspired by and modified from model architecture of language model to make it work well despite the relatively small training set size. Design choices such as addition of <EOS> token, using many-to-one LSTM architecture and segmentation during data processing were evidenced by experimental results to have reduce reliance on training set size. Addition of <EOS> enlarge the gap between normal scores and abnormal scores. Besides, the result shown the best choice of window size for LSTM method is 5. The comparison with LSTM language model shows that the test accuracies was raised by about 17.11% averagely.

The methodology for building reliable anomaly detection model using each of the algorithm for sequential anomaly detection is also presented. The experimental results demonstrated that both LSTM method and database method have similar performance. However, another comparison experiment using K-fold cross validation of K=4 shows that database method is higher in test accuracy about 7.56% averagely when compared to LSTM method. In addition, both of them have better performance than HMM method and LSTM language model in terms of test accuracy when evaluated on a real-life dataset. Lastly, a way to identify activity which caused anomaly in the sequence using LSTM method is presented. The experimental result showed a test accuracy of 89.47%.

CHAPTER 4

TIME ANOMALY DETECTION

4.1 Introduction

In this chapter, a time anomaly detection algorithm based on database is presented. It is also called database method. Time anomaly refers to change in time routine of activities (Forkan, et al., 2015). For example, an ill person might have a disrupted time routine such as longer sleeping activity than usual and skipping meals. Time anomaly detection enables this type of abnormal behaviour to be detected (Forkan, et al., 2015; Zhao, et al., 2015). The methodology for the time anomaly detection algorithm is similar to methodology of sequential anomaly detection algorithms in Chapter 3 with some differences. In this chapter, only the parts of methodology which are different are discussed. Whereas, the steps which are the same are skipped. In section 4.2, the data collection step for time anomaly detection is given. Section 4.3 details the data preparation which include the data processing step. In section 4.4, the database method is described. In section 4.5, the model selection step is given. In section 4.6, the model evaluation step is presented. Section 4.7 covers the experimental results and discussion. In addition, a comparison with a combination method of *K*-means clustering and Gaussian distribution (Zhao, et al., 2014) is presented. Section 4.8 provides a summary of this chapter.

4.2 Data Collection

Each data point refers to daily collection of activities where each activity of the data point comes with corresponding start time and end time. The normal data points are from Aruba dataset.

As for abnormal data points, there are no existing dataset. Therefore, abnormal data points were artificially generated by altering normal data points to simulate time anomaly. Each abnormal data point was created by shifting the start time and end time

of each activity of a normal data point by four hours. The shift of four hours is circular. This means any start time or end time shifted beyond 24:00 was deducted by 24 hours.

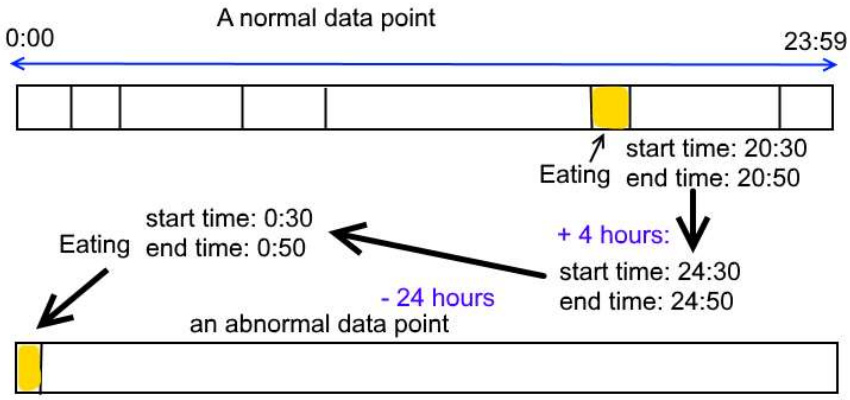

Figure 4.1: Illustration of four hours Circular Shifting of an Eating Activity

Figure 4.1 illustrates an example of circular shifting an eating activity by four hours. Originally, the eating activity of the normal data point has a start time 20:30 and end time 20:50. First, four hours were added to the start time and end time which resulted in a delayed start time 24:30 and delayed end time 24:50. Since both delayed start time and end time are beyond 24:00, 24 hours were deducted from both of them which results in new start time 0:30 and new end time 0:50. The eating activity which happened in the night was shifted to morning. Figure 3.3 visualizes normal data points dated from 4 November to 15 November 2010. Figure 4.2 visualizes abnormal data points generated by circular shifting 4 hours the normal data points in Figure 3.3.

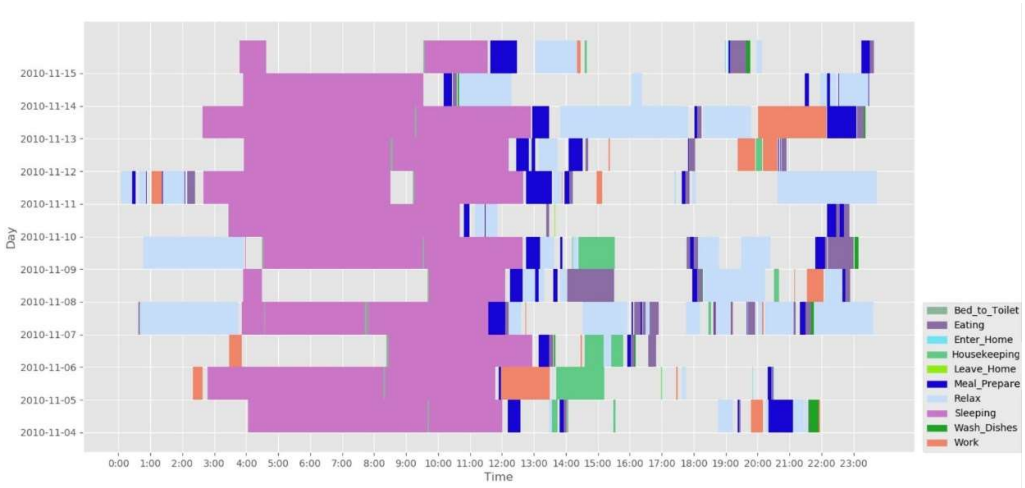

Figure 4.2: Visualization of Abnormal Dataset with Time Anomalies from 4 November to 15 November 2010

4.3 Data Preparation

Data preparation consists of three steps: noise removal, data processing and data partitioning. The noise removal step is same as described in subsection 3.3.1. After noise removal, the remaining normal data size is 189. Data processing and data partitioning steps are different.

4.3.1 Data Processing

Each activity of every data point consists of start time and end time. The timestamps of Aruba dataset are in [hour: minutes: seconds] format. During data processing, only hour and minutes of the time stamp are kept whereas the second of the timestamp is removed. For example, a timestamp of 00:03:50.209589 will be processed into 3:50. The reason for doing this is because the resolution of time up to seconds is adequate.

4.3.2 Data Partitioning

The data points were divided into training, validation and test set. Training set only consists of normal data points whereas both validation and test set consist of normal and abnormal data points. Figure 4.3 illustrates the data partitioning step. First of all, 20% (38 data points) out of total normal data points were allocated for test set.

Then, 38 abnormal data points were generated from these 38 normal data points for test set. Test set size is 76 and consists of equivalent amount of normal and abnormal data points.

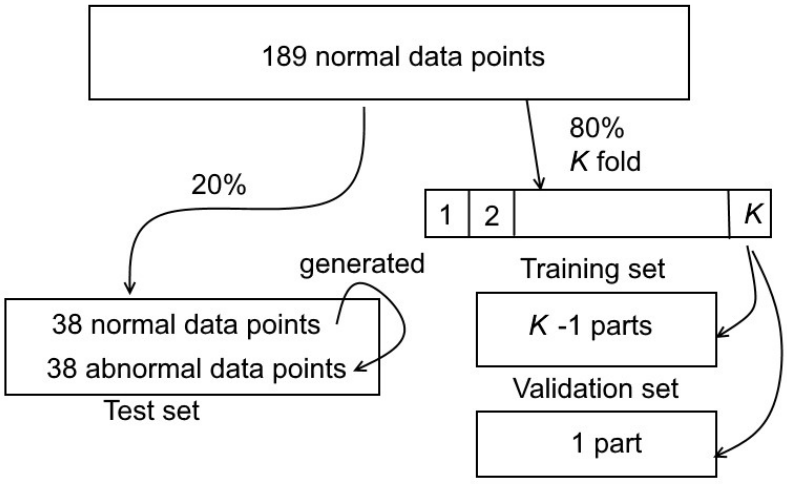

Figure 4.3: Illustration of Data Partitioning into Training, Validation and Test Set

As for training set and validation set, the normal data points were partitioned using *K* fold cross validation method. *K* fold cross validation randomly partitions data points into *K* equivalent parts (James , Witten, Hastie , & Tibshirani, 2013). Each part is used once as validation set while the remaining $K-1$ parts are used as training data (Norvig & Russell, 2009). This results in *K* combinations of training set and validation set. For each validation set, abnormal data points are generated from normal data points. There are *K* combination or fold of training set and validation set to be used. The folds are indexed using italic lowercase typeface *k* where $k = 1, 2, \ldots, K$.

4.4 Building Model

The time anomaly detection algorithm using database consists of two components namely database and anomaly detector. In short, the database is built by dividing the 24 hours duration of a day into smaller equivalent duration and saves each activity of the training data points into respective duration entry based on when it

happened. The number of duration entry of the database is dependent on the hyperparameter *period* and can be calculated using equation (4.1).

$$\text{number of duration entry} = \frac{1440 \text{ (in minutes)}}{\textit{period} \text{ (in minutes)}} \tag{4.1}$$

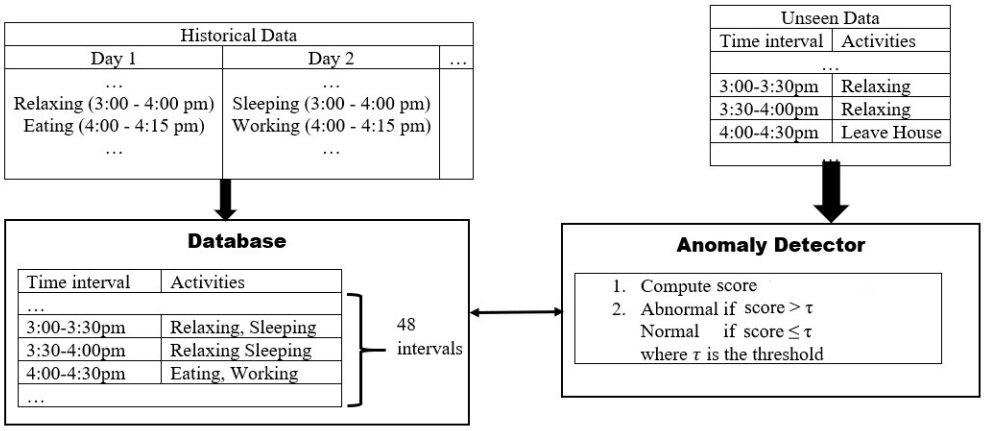

Figure 4.4: Illustration of Components of Time Anomaly Detection Algorithm using Database with Period = 30 minutes

For ease of understanding, the database building process is explained using an example in Figure 4.4. Figure 4.4 illustrates the process of building the database with *period* set to 30 minutes. The number of duration entry is 48 and was calculated using equation (4.1). During the database building process, each activity of the training data points is categorized based on duration when it happened and saved into the respective 30-minutes entry of the database. For example, the eating activity on day 1 (4:00 – 4:15 pm) and the working activity on day 2 (4:00 – 4:15 pm) in the historical record are saved into the duration entry specific for every activity happened from 4:00 – 4:30 pm. For the special case when an activity's duration spans across several smaller durations of the database, that activity is saved in every duration entry it spans across. For example, the relaxing activity on day 1 (3:00 – 4:00 pm) spans across two smaller duration (3:00 – 3:30 pm and 3:30 – 4:00 pm) is saved into both the duration entries for these two smaller durations.

The second component of the algorithm is the anomaly detector. Anomaly detector decides whether a given data point is normal or anomalous. A score metric is used to measure the level of abnormalities of the data point and can be computed using equation (4.2). The classification of a given data point is as given in Figure 4.4. If the score of the given data point is larger than the threshold, then that data point will be classified as anomalous and vice versa.

$$\text{score} = \frac{\text{number of abnormal activities}}{\text{Total number of activities for given data point}} \times 100\% \qquad (4.2)$$

For real time detection, the algorithm must be able to detect whether a given activity with timestamp is normal or anomalous. In this case, the detection is straightforward: if that given activity is not present in the respective duration entries of the database, then it will be classified as anomalous.

4.5 Model Selection

There are four hyperparameters for time anomaly detection algorithm using database method which includes K for K-fold cross validation, the fold k, threshold τ and time period of each database entry *period*. Table 4.1 shows the settings of hyperparameters which were evaluated during the experiment. The way of selecting hyperparameter value is same as in section 3.4. The performance metric used is $F1$ score.

Table 4.1: Hyperparameters of Database Method

Hyperparameter	Symbol	Values
Total number of folds for K fold cross validation	K	4, 6, 8
The fold of training and validation set	k	1,2, …., K
Threshold	τ	100 threshold choices linearly sampled from the range [minimum score, maximum score]
Time period	*period*	10, 15, 20, 25, 30 minutes

4.6 Model Evaluation

The purpose of model evaluation is to ensure of the performance of the selected model is consistent. The way for evaluation is same as the way described in section 3.6. The performance metric used is also accuracy.

4.7 Experimental Results and Discussion

In this section, the experiment results and discussion are presented. In subsection 4.7.1, the experiments conducted to validate the performance of the database method is given. In subsection 4.7.2, a comparison between the database method and combination method which used *K*-means clustering and Gaussian distribution is presented.

4.7.1 Performance of Database Method

An experiment was conducted to evaluate performance in terms of *F*1 scores of the proposed algorithm with different hyperparameter settings which include *period* and *K*. Since *T* is set to 100, 100 threshold choices were linearly sampled for each combination of hyperparameters *K* and *period* which resulted in *K*×100 models per combination. Figure 4.5 consists of a graph of *F*1 score for the best model for each combination of *K* and *period*. The best model is the model with highest *F*1 score. Based on Figure 4.5, the best choice for hyperparameter period is 25 minutes. This is because the *F*1 scores at 25 minutes are the highest for every choice of *K*. On the other hand, the best choice of hyperparameter *K* is 8 because it has the highest *F*1 score when compared to $K = 4$ and $K = 6$. Overall, the best hyperparameter setting is *period* = 25 minutes and $K = 8$.

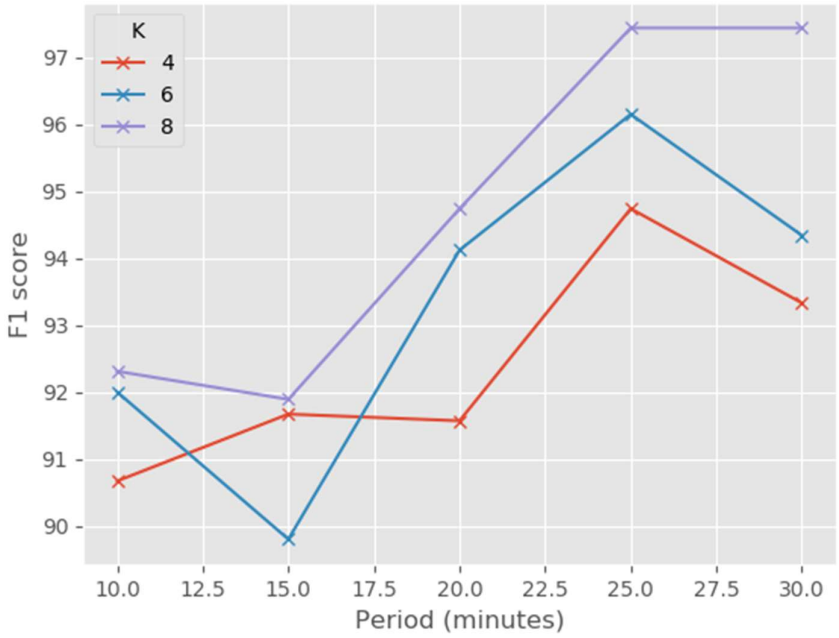

Figure 4.5: *F*1 Score vs. Period (minutes) for Different Choice of *K* = 4, 6, 8

Table 4.2 shows the performance of the selected model (*period* = 25 minutes and *K* = 8) on validation set and test set. The test accuracy and validation accuracy are the same. This means that the model is consistent on unseen dataset and it is a desirable trait. In conclusion, the experimental results evidenced that database method is an excellent time anomaly detection algorithm in terms of performance.

Table 4.2: Performance of the Overall Best Model for Database Method

	Validation (%)	Test (%)
Precision	95	95
Recall	100	100
F1 score	97.44	97.44
Accuracy	97.37	97.37

4.7.2 Comparison between Database Method and Combination Method using *K*-Means Clustering and Gaussian Distribution

In the literature, the combination method of *K*-means clustering and Gaussian distribution was used for time anomaly detection on location dataset instead of home activities dataset (Zhao, et al., 2014). For this experiment, it was applied to home activities data. Then, a performance comparison with the proposed database method was conducted.

Before going into the comparison, a brief explanation of the process of building the model which used combination of *K*-means clustering and Gaussian distribution for anomaly detection is given. The combination method can be used to detect time anomaly on a type of activity individually. This is different from the proposed database method which was trained on sequences consisting of different activities happened in a day collectively. Therefore, the comparison was carried out on each activity type separately.

For ease of explanation, the training methodology was explained using an activity type "sleeping" as an example. The methodology is the same for other activity type. There are two time-related features for the combinational method which include starting time and duration. Every sleeping activity in the test set was extracted along with their corresponding starting time and duration during data processing. Next, the starting time in *hours*: *minutes* format was converted into minutes only by multiplying the *hours* with 60 and added to *minutes* for every piece of sleeping data point. In addition, the duration of each sleeping data point was also converted to minutes only. Figure 4.6 illustrates the starting time and duration of each sleeping data point in the test set. *K*-means clustering with $K = 2$ was used to cluster the sleeping data points based on starting time and duration.

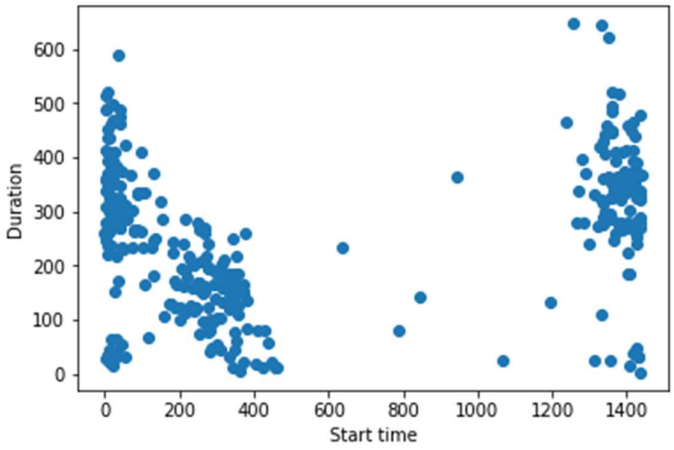

Figure 4.6: Start Time and Duration of Each Sleeping Activity in the Test Set

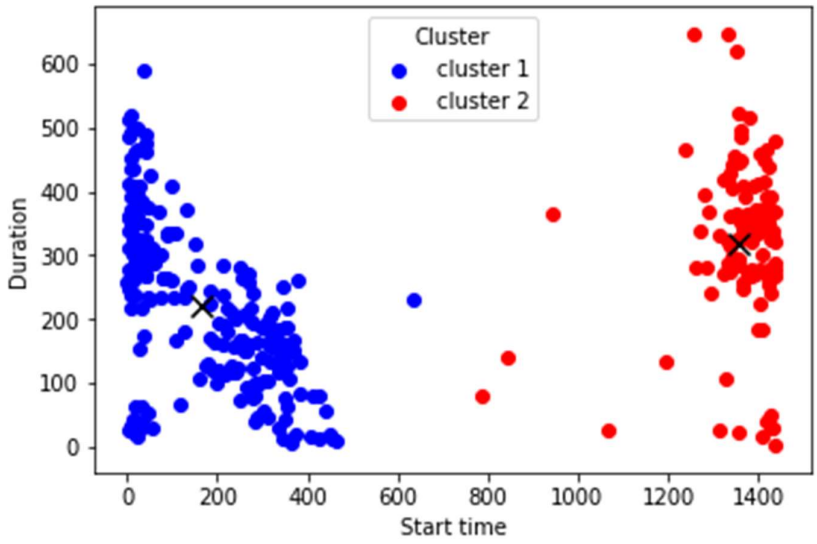

Figure 4.7: *K*-means Clustering of All Sleeping Activity into two Separate Clusters where $K = 2$

Figure 4.7 illustrates the *K*-means clustering result by coloring sleeping data points of cluster 1 with blue and data points of cluster 2 with red. The 'x' marks the data points which are the cluster centroids of respective clusters. The cluster centroid for cluster 1 is (166.67, 219.64) whereas the cluster centroid for cluster 2 is (143.56, 125.56).

As mentioned in Chapter 2, points which are within two standard deviation from the mean of the Gaussian distribution make up 95.4% of the distribution. For each cluster, the mean and standard deviation of the starting time and duration of sleeping data points belonging to that cluster were computed. Anomaly detection is a classification task and there are three class labels which include cluster 1, cluster 2 and anomalous. Classification involves several conditions as given in (4.3). If the start time and duration of a given point fulfills condition 1 and condition 2, then it is classified as belonging to cluster 1. If the start time and duration of a given point fulfills condition 3 and condition 4, then it is classified as belonging to cluster 2. If the sleeping data point is in cluster 1 or cluster 2, it is normal. If it does not fulfil any of the conditions (condition 1 to 4), then it is classified as anomalous. The combination method achieved an accuracy of 93.57%.

condition 1: $\mu_{s1} - 2\sigma_{s1} \leq$ start time $\leq \mu_{s1} + 2\sigma_{s1}$
condition 2: $\mu_{d1} - 2\sigma_{d1} \leq$duration $\leq \mu_{d1} + 2\sigma_{d1}$
condition 3: $\mu_{s2} - 2\sigma_{s2} \leq$ start time $\leq \mu_{s2} + 2\sigma_{s2}$
condition 4: $\mu_{d2} - 2\sigma_{d2} \leq$duration $\leq \mu_{d2} + 2\sigma_{d2}$
where:

μ_{s1} : mean of start time for cluster 1
σ_{s1} : standard deviation of start time for cluster 1
μ_{d1} : mean of duration for cluster 1
μ_{d1} : standard deviation of duration for cluster 1
μ_{s2} : mean of start time for cluster 2
σ_{s2} : standard deviation of start time for cluster 2
μ_{d2} : mean of duration for cluster 2
σ_{d2} : standard deviation of duration for cluster 2

(4.3)

The experiment on sleeping activity data was repeated on database method. As for database method, the overall best model using *period* = 25 minutes and K = 8 mentioned in subsection 4.7.1 was used for comparison. The classification for a single type of activity is easy: if that activity was not saved within the duration entries of database when it happened, then it is classified as anomalous. In total, 70 normal points and 70 abnormal points. Both database method and combination method using K-means clustering and Gaussian distribution achieved an accuracy of 93.57%.

This experiment was repeated on other activity types and the result is presented in Table 4.3. There is a total of ten types of activities. According to Table 4.3, the accuracies of five activity types which include Eating, Enter Home, Meal Preparation, Sleeping and Work using database method are at least the same as the accuracies achieved using combinational method. In short, database method has same or improved performance for 50% of the activities compared to combinational method. For Toileting, Leave Home and Relax, the accuracy of using database method is slightly lower than the accuracy when using combinational method. The difference in accuracy is less than 7% which is insignificant for a small dataset. The significantly worse are two activities which include Housekeeping and Dish Washing, the accuracy of using combinational method is significantly higher when compared to the accuracy of using database method. The degradation in performance is due to insufficiency of data because the Housekeeping and Dish Washing are the two least occurring activity types in the dataset as listed in Table 4.3. This reveals a weakness of the database method which is data insufficiency. Lack of data can affect the proposed database method's accuracy.

As for the activity type Relax, the occurrence is the highest but the accuracies of using combinational method and database method are 61.49% and 54.89% respectively. This is because Relax activity happens throughout the day and without fixed pattern as shown in Figure 4.8. Therefore, both methods performed badly in terms of accuracy for Relax.

Table 4.3: Comparison in terms of Accuracy between Combinational Method and Database Method for each Activity Type

Activity	Occurrence	Combinational Method Accuracy (%)	Database Method Accuracy (%)	Difference (%)
Toileting	157	96.67	93.33	3.34
Eating	257	82.22	92.22	10.00
Enter Home	431	87.80	98.17	10.37
Leave Home	431	95.12	93.29	1.83

Housekeeping	33	94.44	77.78	16.66
Meal Preparation	1606	80.42	96.39	15.97
Relax	2910	61.49	54.89	6.60
Sleeping	401	93.57	93.57	0.00
Dish Washing	65	95.00	65.00	30.00
Work	171	83.33	83.33	0.00

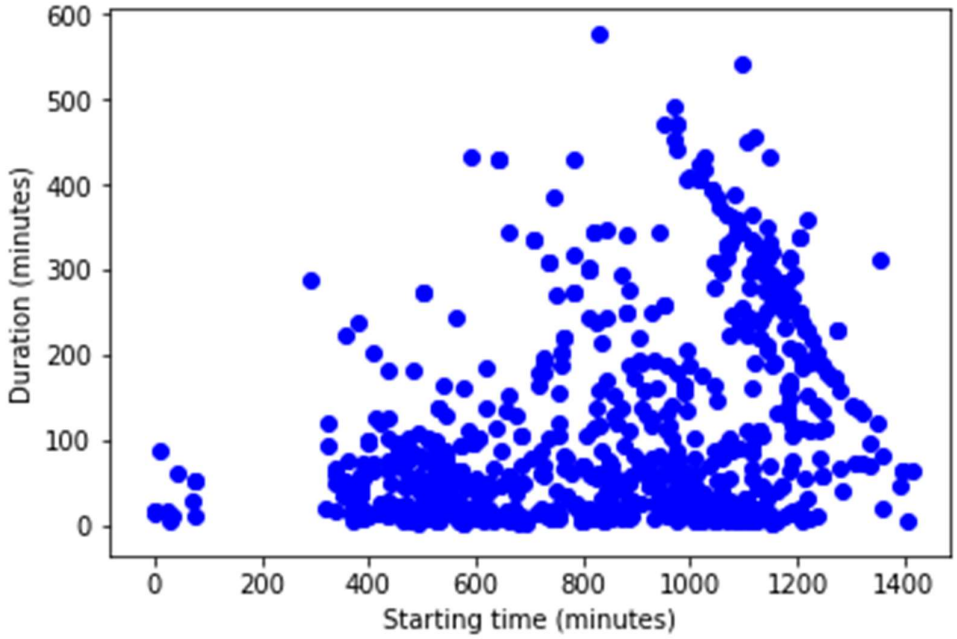

Figure 4.8: Duration vs. Starting Time in Minutes for Relax

Database method has one advantage over the combinational method and other time anomaly detection algorithms mentioned in the Chapter 2. The database method used different activities collectively in its training, therefore there is no need to set hyperparameters individually for each type of activity. On the other hand, other algorithms such as DBSCAN and combinational method requires hyperparameters to be set for each type of activity one by one and separately. For example, the number of clusters, *K* for each type activity is different and required human expertise to select the best choice. Database method does not require this because it used all activities of the training sequences all at once to build the model. This feature of database method makes it suitable for automated training and deployment. However, the database

method has a weakness due to data insufficiency. The experimental results shown that the proposed database method can perform on par with combinational method when there is sufficient data.

4.8 Summary

In this chapter, an anomaly detection algorithm using database to detect time anomalies in home activities data is introduced. The methodology of this algorithm is largely similar to the methodology described in Chapter 3 with some difference in Data Processing, Data Partitioning which involves *K* fold cross validation and the score metric.

The database method has shown excellent performance in terms of precision, recall, *F*1 score and accuracy. The main strength of the database method over other algorithms described in past works is that its model can be built using entire sequences of different home activities collectively instead of building one model for each of the activity separately. Unlike other algorithms such as *K*-means clustering which require human expertise to set hyperparameters, this algorithm uses *K* fold cross validation to select the combination of training set and validation set which maximizes the performance of the model. This makes it suitable for automated deployment.

This algorithm also shown comparable performance when compared to the combination method using Gaussian distribution and *K*-means clustering given there is sufficient data. Anomaly detection can be done for a daily sequence of home activities collectively or for a single activity data point. This makes it suitable for real time detection.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Summary and Conclusion

For this research, there are two main research objectives. The first objective is to develop accurate sequential anomaly detection algorithm to detect sequential anomalies in home activities data. The second objective is to develop time anomaly detection algorithm to detect time anomalies in home activities data. Detection algorithms for these anomalies proposed during past researches were studied. These algorithms were thoroughly analysed to find out about their strength and weakness so that better algorithms can be designed. When designing and developing new anomaly detection algorithms, their strength and weakness were taken into consideration.

One of the problems faced when developing the detection algorithms is the relatively small training set size of this task. Usually, machine learning algorithms require several thousand data points per class label to deliver satisfactory results. Preliminary experiment with HMM demonstrated that the test accuracies are around 50% on the test set derived from Aruba dataset. In chapter 3, two new sequential anomaly detection algorithms to combat the weaknesses are presented. The first method is LSTM method. To model sequence, the state-of-the-art algorithm is LSTM. A LSTM model architecture used for language model was discovered to have properties suitable for sequential anomaly detection on home activities data. However, there is limitation to the performance imposed by the relatively small training set size which is in the range of hundred days. Preliminary experiment revealed that many-to-many LSTM architecture and no segmentation during data processing results in average accuracy of 59.87%. Design considerations were to increase the training set size by segmentation and using a many-to-one LSTM architecture instead of many-to-many LSTM architecture which was used by language model. In addition, addition of <EOS> was found to raise the ability of LSTM method to better distinguish scores of normal data points from scores of anomalous data points. Besides, a new score metric was devised for this modified LSTM model architecture. Overall, the results

demonstrated excellent test accuracy of 85.53%. A sequence consists of many activities, the ability to identify the activities which are causing the anomaly is important. The experimental result demonstrated an accuracy of 89.47%.

The second method designed for sequential anomaly detection method is the database method. This method is much simpler when compared to LSTM method because the database just saves all the segmented sequences. Similar to LSTM method, segmentation is used during data processing to break down long daily sequence into smaller segments with window size of 3. The experiment demonstrated an excellent test accuracy of 84.21%. In this experiment, both the overall best LSTM model and model for database method have similar performance in terms of precision, recall, $F1$ score and accuracy on the test set.

From the experiment using 4-fold cross validation, the average test accuracy database method is higher than LSTM method by 7.56%. Together, they performed better than HMM method and LSTM language model on the test set obtained from Aruba dataset. In addition, both LSTM method and database method can be trained with relatively small training set when compared to the dataset size of other machine learning problems.

In Chapter 4, a time anomaly detection algorithm using database which saves each activity in the training set into respective entry of the database based on when it happened. K fold cross validation is used to divide the dataset into different combinations of training set and validation set. This method can be used for anomaly detection when given data point is a daily sequence of activity or just a single activity with its associated starting time and duration. On daily sequence consisting different activities, the experimental result shown that it has an excellent performance on test set in terms of precision, recall, $F1$ score and accuracy which 93.57%. To compare the performance of database method with the combination method using K-means clustering and Gaussian distribution, experiment was performed on each activity type separately. This is because the combination method trains its model separately for each type of activity and the trained model can only detect the anomalies for a specific

activity type it was trained for. Both database method and combinational method using *K*-means clustering and Gaussian distribution achieved average accuracy above 84%. Besides that, the proposed database method is more straightforward during training. All the methods described in the literature require user to train at least one model per activity. For database method, there is no such problem. User can train this algorithm in one go without having to separate out each type of activity and still have high accuracy during deployment. However, database method has a weakness of data insufficiency. This is evidenced by significantly worse accuracy when using database method for Housekeeping and Dish Washing when compared to accuracy when using combinational method. In other words, database method may require more data to achieve accuracy on par with combinational method.

In a nutshell, anomaly detection algorithms presented in Chapter 3 and Chapter 4 fulfilled objectives of this research to develop sequential and time anomaly detection algorithms to detect anomalies within home activities dataset.

5.2 Recommendation for Future Work

In this research, detection algorithms to detect sequential anomalies and time anomalies in home activities were proposed and evaluated using Aruba dataset. The result demonstrated excellent and promising performance. The recommendation for future works includes:

1. To develop a complete anomaly detection software and hardware framework that can actively detect anomalies. For software, it would include development of embedded device which can detect elderly home activities. In addition, it would also include exploring big data software platform such Hortonworks to cater to multiple elderly people.

2. To develop a practical use case for these proposed algorithms. For example, these algorithms could be used to detect symptoms of chronic diseases which manifest as change in behavior of elderly home activities.

REFERENCES

[1] Abujar, S., Mohammad Masum, A., Hoque Chowdhury, S., Hasan, M., & Hossain, S. (2019). Bengali Text generation Using Bi-directional RNN. *2019 10th International Conference on Computing, Communication and Networking Technologies* (pp. 1-5). Kanpur, India: IEEE.

[2] Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. *20th International Conference on Very Large Data Bases*, (pp. 487-499).

[3] Ali , T., Asghar, S., & Sajid, N. (2010). Critical Analysis of DBSCAN Variations. *2010 International Conference on Information and Emerging Technologies* (pp. 1-6). Karachi: IEEE.

[4] Anthony, M., & Bartlett, P. (2009). *Neural Network Learning: Theoretical Foundations.* Cambridge University Press.

[5] Asplund, R. (2002). Nocturia in relation to sleep, somatic diseases and medical treatment in the elderly. *BJU International*.

[6] Babiker, M., Khalifa, O., Htike, K., Hassan, A., & Zaharadeen, M. (2018). Automated daily human activity recognition for video surveillance using neural network. *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)* (pp. 1-5). Putrajaya, Malaysia: IEEE.

[7] Bao, L., & Intille, S. (2004). Activity Recognition from User-Annotated Acceleration Data. *International Conference on Pervasive Computing*, (pp. 1-17).

[8] Barshan, B., & Yurtman, A. (2020). Classifying Daily and Sports Activities Invariantly to the Positioning of Wearable Motion Sensor Units. *IEEE Internet of Things Journal*, 1.

[9] Bengio, Y. (1999). Markovian Models for Sequential Data. *Neural Computing Surveys*, 129-162.

[10] Bux , A., Angelov, P., & Habib, Z. (2016). Vision Based Human Activity Recognition: A Review. In *Advances in Computational Intelligence Systems Volume 513* (pp. 341-371). Springer, Cham.

[11] Chen , Y., & Xue, Y. (2015). A Deep Learning Approach to Human Activity Recognition Based on Single Accelerometer. *2015 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1488-1492). Kowloon, China: IEEE.

[12] Cho, K., Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio , Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Doha, Qatar: Association for Computational Linguistics.

[13] Chollet, F. (2017). *Deep Learning with Python.* Manning Publications Company.

[14] Cook, D. (2010). Learning Setting-Generalized Activity Models for Smart Spaces. *IEEE Intelligence Systems*.

[15] De Maesschalck, R., Jounan-Rimbaud, D., & Massart, D. (2000). The Mahalanobis Distance. *Chemometrics and Intelligent Laboratory Systems, Volume 50, Issue 1*, 1-18.

[16] Dekking, F., Kraaikamp, Lopuhaa, & Meester, L. (2006). *A Modern Introduction to Probability and Statistics: Understanding Why and How.* Springer Science & Business Media.

[17] Dong, Y., Wen, R., Li, Z., Zhang, K., & Zhang , L. (2019). A New RNN Based Approach to Diabetic Blood Glucose Prediction. *2019 IEEE 7th International Conference on Bioinformatics and Computational Biology* (pp. 50-55). Hangzhou, China: IEEE.

[18] Ester , M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 226–231). AAAI Press.

[19] Forkan, A., Khalil, I., Tari, Z., Foufou, S., & Bouras, A. (2015). A context-aware approach for long-term behavioural change detection and abnormality prediction in ambient assisted living. *Pattern Recognition* , 628-641.

[20] Franz, A., & Brants, T. (2006, August 3). *All Our N-gram are Belong to You*. Retrieved from Google AI blog: https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

[21] Freedman, D. (2012). *Markov Chains.* Springer Science & Business Media.

[22] Friday, N., Al-garadi, M., Mujtaba, G., Alo, U., & Waqas, A. (2018). Deep learning fusion conceptual frameworks for complex human activity recognition using mobile and wearable sensors. *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)* (pp. 1-7). Sukkur, Pakistan: IEEE.

[23] Gagniuc, P. (2017). *Markov Chains: From Theory to Implementation and Experimentation.* Wiley.

[24] Galea, A., & Capelo, L. (2018). *Applied Deep Learning with Python: Use scikit-learn, Tensorflow, and Keras to create intelligent systems and machine learning solutions.* Packt Publishing Ltd.

[25] Georgii, H.-O. (2008). *Stochastics: Introduction to Probability and Statistics.*

[26] Giri. (1993). *Introduction to Probability and Statistics.* CRC Press.

[27] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

[28] Greff, K., Srivastava, R., Koutnik, J., Steunebrink, B., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 2222 - 2232.

[29] Gurney, K. (1997). *An Introduction to Neural Network.* CRC Press.

[30] Gurnon, E. (2017). *The Staggering Prices Of Long-Term Care 2017.* Forbes.

[31] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 1735-1780.

[32] Holicky, M. (2013). *Introduction to Probability and Statistics for Engineers.* Springer Science & Business Media.

[33] Hoque, E., Dickerson, R., Preum, S., Hanson , M., Barth, A., & Stankovic, J. (2015). Holmes: A Comprehensive Anomaly Detection System for Daily In-home Activities. *2015 International Conference on Distributed Computing in Sensor Systems*, (pp. 40-51).

[34] Huang, H., Li , X., & Sun, Y. (2016). A triboelectric motion sensor in wearable body sensor network for human activity recognition. *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, (pp. 4889-4892). Orlando.

[35] James , G., Witten, D., Hastie , T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R.* Springer.

[36] Karpathy, A. (2015, May 21). *The Unreasonable Effectiveness of Recurrent Neural Networks.* Retrieved from Andrej Karpathy blog: https://karpathy.github.io/2015/05/21/rnn-effectiveness/

[37] Kaytaran, T., & Bayindir, L. (2018). Activity Recognition with Wrist Found in Photon Development Board and Accelerometer. 2018 26th Signal Processing

[38] Ke, S.-R., Thuc, H., Lee, Y.-J., Hwang, J.-N., Yoo, J.-H., & Choi, K.-H. (2013). A Review on Video-Based Human Activity Recognition. *Computers*, 88-131.

[39] Kriegel, H.-P., Kroger , P., Sander, J., & Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery Volume 1, Issue 3*, 231-240.

[40] Madhavan, S. (2015). *Mastering Python for Data Science.* Packt Publishing Ltd.

[41] Mendenhall, W., Beaver, R., & Beaver, B. (2012). *Introduction to Probability and Statistics.* Cengage Learning.

[42] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, September 7). *Efficient Estimation of Word Representations in Vector Space.* Retrieved from arXiv: https://arxiv.org/abs/1301.3781

[43] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. *11th Annual Conference of the International Speech Communication Association.* Makuhari, Chiba, Japan: ISCA Archive.

[44] Morris, B. (2004). *Molecular Biology of the Neuron.* OUP Oxford.

[45] Murphy, K. (2013). *Machine Learning: a Probabilistic Perspective.* MIT Press.

[46] Natingga, D. (2017). *Data Science Algorithms in a Week.* Packt Publishing Ltd.

[47] Ng , A. (2017). Language Model and Sequence Generation.

[48] Ng , A., & Katanforoosh, K. (2018, October 29). *Deep Learning.* Retrieved from CS229: Machine Learning: http://cs229.stanford.edu/notes/cs229-notes-deep_learning.pdf

[49] Ng, T., Jin, A., Feng, L., Ma, S., Khuan, Y., Feng, L., & Ngan, P. (2015). Mortality of older persons living alone: Singapore Longitudinal Ageing Studies. *BMC Geriatrics*, 216.

[50] Norris, J. (1998). *Markov Chains.* Cambridge University Press.

[51] Norvig, P., & Russell, S. (2009). *Artificial Intelligence: A Modern Approach.* Prentice Hall.

[52] Nweke, H., Teh, Y., Al-garadi, M., & Alo, U. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications Volume 105*, 233-261.

[53] Olah, C. (2015, August 27). *Understanding LSTM Networks.* Retrieved from colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[54] Ordonez, F., de Toledo, P., & Sanchis, A. (2013). Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors. *Sensors*, 5460-5477.

[55] Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.-C. (2001). PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. *17th International Conference on Data Engineering* (pp. 215-224). Heidelberg, Germany: IEEE.

[56] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Visual Computing*, 976-990.

[57] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep Contextualized Word Representations. 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 2227-2237). New Orleans.

[58] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 257-286.

[59] Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine* , 4-16.

[60] Ramanathan, M., Yau, W.-Y., & Teoh, E. (2014). Human Action Recognition With Video Data: Research and Evaluation Challenges. *IEEE Transactions on Human-Machine Systems*, 650 - 663.

[61] Revuz, D. (2008). *Markov Chains.* Elsevier.

[62] Rohatgi, V., & Ehsanes Saleh, A. (2011). *An Introduction to Probability and Statistics.* John Wiley & Sons.

[63] Rojas, R. (2013). *Neural Networks: A Systematic Introduction.* Springer Science & Business Media.

[64] Rosenkrantz, W. (2008). *Introduction to Probability and Statistics for Science, Engineering, and Finance.* CRC Press.

[65] Ross, S. (2009). *Introduction to Probability and Statistics for Engineers and Scientists.* Academic Press.

[66] Sasaki, Y. (2007, October 26). The truth of the F-measure. Retrieved from Old Dominion University: https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

[67] Schubert, E., Sander, J., Ester, M., Kriegel, H., & Xu, X. (2015). DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems (TODS)* , 1-21.

[68] Shi, X., Li, Y., Zhou , F., & Liu , L. (2018). Human Activity Recognition Based on Deep Learning Method. *2018 International Conference on Radar (RADAR)* (pp. 1-5). IEEE.

[69] Summerfield, M. (2008). *Programming in Python 3: A Complete Introduction to the Python Language.* Pearson Education.

[70] Symth, P., Heckerman, D., & Jordan, M. (1997). Probabilistic independence networks for hidden Markov models. *Neural Computation*, 227-269.

[71] Tian, D., Xu, X., Tao, Y., & Wang, X. (2017). An Improved Activity Recognition Method Based on Smart Watch Data. *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, (pp. 756-759). Guangzhou.

[72] United Nations. (2017). *World Population Ageing.*

[73] van Kasteren, T., Englebienne, G., & Krose, B. (2010). Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software. In *Activity Recognition in Pervasive Intelligent Environments of the Atlantis Ambient and Pervasive Intelligence series* (pp. 165-186). Asterdam: Atlantis Press.

[74] VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data.* O'Reilly Media.

[75] Weiss, G., Timko, J., Gallagher, C., Yoneda, K., & Schreiber, A. (2019). Smartwatch-based Activity Recognition: A Machine Learning Approach. *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, (pp. 426-429). Las Vegas.

[76] Weiss, J., & Blaivas, J. (2000). Nocturia. *The Journal of Urology Volume 1*, 5-12.

[77] Wu, J. (2012). *Advances in K-means Clustering: A Data Mining Thinking.* Springer Science & Business Media.

[78] Yuen, M. (2017, July 9). *Growing demand for retirement villages.* The Star. Retrieved from The Star.

[79] Zhang, S., Wei, Z., Nie, J., Huang, L., Wang, S., & Li , Z. (2017). A Review on Human Activity Recognition Using Vision-Based Method. *Journal of Healthcare Engineering*, 31.

[80] Zhao, T., Ni, H., Zhou, X., Qiang, L., Zhang, D., & Yu, Z. (2014). Detecting Abnormal Patterns of Daily Activites for the Elderly Living Alone. *International Conference on Health Information Science.* Shenzen, China: Springer.

[81] Zimek, A., & Schubert, E. (2017). Outlier Detection. In *Encyclopedia of Database Systems* (pp. 1-5). New York: Springer.

PUBLICATION LIST

Conference Paper Presentation:

[1] Poh, S. C., Tan, Y.F., Cheong, S. N., Ooi, C. P., & Tan, W. H. (2018, November). Anomaly detection for elderly in-home activity monitoring. In *The 3rd International Conference on Electrical, Electronic, Communication and Control Engineering.* Johor Bahru, Malaysia.

[2] Poh, S. C., Tan, Y.F., Guo, X., Cheong, S. N., Ooi, C. P., & Tan, W. H. (2019, March). LSTM and HMM comparison for home activity anomaly detection. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control*. Chengdu, China.

[3] Poh, S. C., Tan, Y.F., Cheong, S. N., Ooi, C. P., & Tan, W. H. (2019, March). Anomaly detection for home activity based on sequence pattern. In *2019 International Conference on Advanced Science, Engineering and Technology*. Selangor, Malaysia.

Journal:

[1] Poh, S. C., Tan, Y.F., Cheong, S. N., Ooi, C. P., & Tan, W. H. (2019, August). Anomaly detection on in-home activities data based on time interval. *Indonesian Journal of Electrical Engineering and Computer Science vol 15, No 2*, (pp. 778-785).

[2] Poh, S. C., Tan, Y. F., Cheong, S. N., Ooi, C. P., & Tan, W. H. (2020, February). Anomaly detection for home activity based on sequence pattern. *International Journal of Technology vol 10, No 7,* (pp. 1276-1285)