HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA

CHEW YONG SHAN

SESSION 2019/2020

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

SEPTEMBER 2019

HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA

BY

CHEW YONG SHAN

SESSION 2019/2020

THE PROJECT REPORT IS PREPARED FOR

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY
MULTIMEDIA UNIVERSITY
IN PARTIAL FULFILLMENT
FOR

BACHELOR OF INFORMATION TECHNOLOGY
B.I.T. (HONS) SECURITY TECHNOLOGY

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

SEPTEMBER 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 institute of learning.

CHEW YONG SHAN

Faculty of Information Science & Technology
Multimedia University

Submission Date: 19 September 2019

Submission Time:

ACKNOWLEDGEMENT

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my utmost gratitude to my supervisor, Prof. Dr. Pang Ying Han for her invaluable advice, guidance and her enormous patience throughout this research. I would also like to express my gratitude to my co-supervisor, Dr. Ooi Shih Yin for her support.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement throughout this Final Year Project Phase I. I would also like to thank Multimedia University for providing me the required knowledge, skill and methodology approach to complete this project.

ABSTRACT

Human Activity Recognition (HAR) has been a hot topic since the emergence of the Internet of Things (IoT) and Big Data. It is widely used in health applications to track user activity and count calories burnt. There are many research papers dedicated to enhancing the application of HAR. Most applications derived from hand-crafted feature and feature extraction algorithm to determine the accuracy of HAR. The recent adoption of deep learning features has sparked a new age of race to develop the best model for HAR. However, most of the papers focus on one type of deep learning methods such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). This research is able to develop a fusion of CNN and RNN model to gain a deeper insight into HAR, coupled with the improvement of its classification accuracy.

Furthermore, the widespread use of self-collected data and multimodal sensor data has been questionable as they are not convenient to use and some results may be biased. Therefore, this research used two widely accepted public datasets which are the UCI and WISDM datasets to evaluate the proposed model. This research also has evaluated the effect of hyperparameter on the performance of the model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/SYMBOLS

Adam	Adaptive Moment Estimation
Bidir-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
HAR	Human Activity Recognition
IoT	Internet of things
LSTM	Long Short-Term Memory
ReLU	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
UCI	University of California Irvine
WISDM	Wireless Sensor Data Mining

σ	Activation function
x	Accelerometer data input vector
b	Bias
U	Input layer to hidden layer
V	Hidden layer to output layer
W	Hidden layer to hidden layer
μ	Momentum coefficient
w	Weight
λ	Weight decay

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

With the swift and violent progress of the Internet of things (IoT), big data and cloud computing, wearable device and smartphone hold the formidable power of making use of sensor data for personal health. A good example of this is that a large number of smartphone applications can conveniently count user motion steps, distance, duration, and velocity, on the basis of accelerometer data. It is well known that running or walking for a reasonable period of time is beneficial to a user's health while sitting for a long period of time in a room can harmful. Thus, with the data provided by the sensor, an application can count calories for its users to provide the best recommendation on whether to improve their exercise frequency or to provide an estimated projection of their health. Such sensor data is usually collected using an accelerometer or gyroscope built in a wearable device or smartphone.

An accelerometer is a device that tracks the triaxial acceleration of a person or object. With the recent improvement of the sensitivity of accelerometer, we can track a person's motion more accurately and determine whether he or she is sitting, standing, walking, running, lying, and so forth. There are currently a lot of approaches to accurately classify a person’s activity utilizing the accelerometer data such as a wearable-based approach, vision-based approach or smartphone-based approach (Anguita, Ghio, Oneto, Parra, and Reyes-Ortiz, 2013; Bao and Intille, 2004; Bayat, Pomplun, and Tran, 2014; Chong, 2017; Kwapisz, Weiss, and Moore, 2011; Lee and Kwan, 2018; Lockhart, 2014).

Wearable-based approach and vision-based approach are two of the most common approach in HAR. However, most people find both of these methods are inconvenient and are reluctant to them. Plus, there are also privacy issues revolve around the vision-based approach. Hence, the smartphone-based approach has gained much more popularity in recent years. Smartphone’s size is smaller and they can be conveniently put into pockets. Almost all the smartphone has a built-in accelerometer. Thus, conducting HAR using a smartphone sensor is the main focus of this research.

1.2 Problem Statement

Although there are many approaches, HAR is still relatively new if compared with other recognition techniques such as face recognition which dated back into the 80s and 90s. There is still room for improvement for HAR’s classification accuracy. As in other object and pattern recognition, deep learning analysis is extensively employed in HAR to extract the deep features of the targeted data. For example, deep learning approaches have been proposed by numerous works, such as Ronao and Cho (2016), Yu and Qin (2018), and Hernández, Suárez, Villamizar, and Altuve (2019).

From the literature review, most of the proposed architectures could perform well with access numbers of epochs, such as 5000 epochs (Ronao and Cho, 2016) or 50000 epochs (Yu and Qin, 2018). Other than the high number of epochs, advanced hardware is also needed in order to run such huge operations. One notable case is the study conducted by Ronao and Cho (2016), the hardware used composed of two Intel Xeon E5 CPUs that drive two NVIDIA Quadro K5200 GPUs. The former has six cores and twelve threads each, powerful enough to drive an NVIDIA GPU. The two NVIDIA Quadro K5200 GPUs have 8 gigabytes of RAM, 2304 CUDA cores, and a bandwidth of 192 GB/s. The hardware specifications limitation and such high numbers of epochs are impossible to run with a common processing unit.

The fusion of CNN and RNN models are also not extensively carried out, and the effect of hyperparameter on the performance is seldom highlighted in HAR. Therefore, this research seeks to provide a better HAR model in terms of classification accuracy and also studies the effect of hyperparameter.

1.3 Research Objective

The research objective is summarized as followed:

- To design an automated HAR based on smartphone sensor data
- To explore different kinds of approaches for the HAR.
- To study the time-domain features, frequency domain features, and deep features
- To explore a robust classifier for HAR.

1.4 Research Scope

In this study, the research focus is on two things:

- Study the effect of time, frequency and deep features on accelerometer data to classify the types of human activities
- Explore several classifiers to classify human activity types solely based on accelerometer data that captured using a smartphone

1.5 Report Organization

The performance of a Human Activity Recognition (HAR) model is determined by the accuracy of classifying human activities. The higher the accuracy, the better the performance is. Therefore, the existing methods of HAR approaches and different features used in HAR which include hand-crafted features and deep learning features are explored in CHAPTER 2. The two publicly available datasets which are the UCI dataset and WISDM dataset are also compared in CHAPTER 2.The performance of existing deep learning HAR models proposed by different authors is also tabulated in CHAPTER 2.

The proposed model is focused in CHAPTER 3 and each feature of the proposed model is discussed in detail which includes the architecture and formulation of CNN, LSTM, and Bidir-LSTM. The different optimizations and activations used in deep learning models are also discussed in CHAPTER 3.

The information on the computing resources of the device used and the required software to run the experiment are first summarized in CHAPTER 4. Ten different hyperparameter configurations for both UCI and WISDM datasets are tabulated and the result is exhibited and discussed in CHAPTER 4. A comparison study between the performance of the proposed model with other proposed models is also highlighted in CHAPTER 4 An additional experiment to explore the effect of hyperparameter is also conducted and its results are also summarized and discussed in CHAPTER 4. CHAPTER 5 marked the concluding discussion of the overall research and its future direction.

CHAPTER 2

LITERATURE REVIEW

In this chapter, the different approaches of HAR are discussed in Section 2.1, where wearable-based and vision-based are explored further in Section 2.1.1 and Section 2.1.2 respectively, followed by smartphone-based in Section 2.1.3. The features used in HAR is also explored in the following section (Section 2.2), where the hand-crafted features and learning features are presented in Section 2.2.1 and Section 2.2.2. Next, the topic of deep learning in HAR is opened in Section 2.3, and both CNN (Section 2.3.1) and RNN (Section 2.3.2) are discussed. Different variants of RNN such as LSTM, Bidir-LSTM, and residue LSTM are briefly remarked in Section 2.3.2.1 to Section 2.3.2.3. Next, a comparison study of the existing deep learning models is explored and tabulated in Section 2.3.3. Followed by a comparison study of WISDM dataset and UCI dataset in Section 2.4.

2.1 Human Activity Recognition (HAR)

Human Activity Recognition (HAR) is a series of accurate predictions and classification for a particular set of human activities which include walking, standing, running and climbing upstairs, etc. There are many approaches to human activity recognition (Bao & Intille, 2004; Kwapisz et al., 2011; Lee & Kwan, 2018). The deep research on HAR in recent years has driven more development of health applications such as mobile application, Fitbit and SmartWatch which can track user activity and estimating calories burned.

HAR can be categorized into vision-based HAR and wearable sensor HAR. Vision-based HAR received the least attention, however, we will move into detail about the research which utilized this approach in Section 2.1.1. Most research focuses solely on a wearable sensor such as early work by Bao and Intille (2004). Other researchers like Zhang, Rowlands, Murray, and Hurst, (2012), Gao, Bourke, & Nelson (2014), Bharti (2017) and Chong (2017) followed suit with a wearable sensor. We will be discussing this in Section 2.1.2. Next, we will discuss various research concerning the Smartphone-based HAR approach which is also a better alternative than a wearable sensor (Anguita et al., 2013; Bayat et al., 2014; Kwapisz et al., 2011; Lockhart, 2014).

2.1.1 Vision-based HAR

According to the studies carried out by Poppe (2010), Vision-based Human Action Recognition (VHAR) is the process of categorizing a sequence of image recording with action labels. VHAR has many implementations in different sectors such as visual surveillance, video retrieval, and human-computer interaction. Poppe (2010) pointed out that there is a series of difficulties in VHAR such as the variations in human motion, recording setting, and inter-personal differences. There are large differences in performance for each action. For example, people can differ in how fast they walk and how wide their stride is. The environment is also another major concern. The environment in which the action performance takes place can be too cluttered or dispersed which can affect the camera viewpoint. There is also the problem of lighting conditions which can influence how a person looks and how clearly the action was recorded.

When comes to VHAR or anything video-related, there is the privacy issue that comes along with it. In other words, the volunteers must be comfortable with being recorded. In a controlled laboratory setting, volunteers have to be explicitly reminded that there will be video recordings of them to analyze further their actions. These processes take much longer time and running classification or feature extraction on both training and testing data will be troublesome and take up more computational resources. This further causes the issue of having less data for running any model which will also influence the prediction accuracy.

These challenges can be overcome by using some of the publicly available datasets such as KTH human motion dataset (Schüldt, Laptev, & Caputo, 2004), Weizzmann human action dataset (Gorelick, Blank, Shechtman, Irani, & Basri, 2007), UCF sports action dataset (Soomro & Zamir, 2014) and INRIA XMAS multiview dataset (Weinland, Ronfard, & Boyer, 2006). These public datasets have influenced the work of VHAR by allowing for a more justifiable comparison between different strategies on ubiquitous training and test data. They also permit better insight into methods since researchers are aware of the challenges of each set.

However, algorithms may be biased to a specific dataset. This may lead to complex approaches that perform better on a specific dataset but it may be less generally applicable. The ability to generalize the prediction outcome of each model will be deteriorated as more complex approaches are introduced.

2.1.2 Wearable Sensor HAR

As pointed out in Section 2.1, wearable sensor HAR received more favors among researchers. In the early work of HAR using accelerometer data by (Bao & Intille, 2004), five biaxial accelerometers are worn on the user's right hip, dominant wrist, non-dominant upper arm, dominant ankle, and non-dominant thigh to monitor 20 types of activities, trained with 20 users using decision tables, instance-based learning, C4.5, and Naive Bayes classifiers. The results revealed that the predictability of some particular activities can be subject independent and subject-specific. Multiple accelerometers can aid in recognizing specific activities with the help of acceleration feature values that can effectively differentiate many activities. The performance dropped only slightly with just two biaxial accelerometers placed on the thigh and wrist.

In (Nishkam Ravi Preetham Mysore, Michael L. Littman, 2005), the accelerometer was worn near the pelvic region while the subject performed activities. The data generated by the accelerometer was transmitted to an HP iPAQ (carried by the subject) wirelessly over Bluetooth. The Bluetooth transmitter is wired into the accelerometer. A Bluetooth-enabled HP iPAQ running Microsoft Windows was used. The Windows Bluetooth library was used for programming Bluetooth. The authors collected data for a set of eight activities: Standing, Walking, Running, climbing upstairs, climbing downstairs, sit-ups, vacuuming and brushing teeth.

In (Zhang et al., 2012), raw acceleration data is collected from the GENEA (Gravity Estimator of Normal Everyday Activity) and the prediction accuracy of both a wrist-worn GENEA with a waist-worn GENEA were compared. 60 participants, age averaged of 46, were required to accomplish a fixed series of 10-12 semi-structured activities in the laboratory and outdoor environment. Throughout the process, three GENEA accelerometers were worn: one at the waist, one on the left wrist and one on the right wrist.

Li (2017) focused on the use of wearable sensors in human activity recognition and proposes an accelerometer-based real-time human activity recognition approach using the decision tree as the classifier. This paper aimed to create an approach that requires only one accelerometer to be worn on the user's wrist and recognizes activities in real-time based on the acceleration data. The decision tree was adopted as the classification algorithm and a classifier simplification technique and a novel decision tree storage structure were designed.

Gao, Bourke, and Nelson (2014) pointed out that there are a substantial amount of research studies exist, which focus on activity recognition using inertial sensors. A single sensor approach is adopted in many of these studies and they solely focus on proposing interesting features combined with complex classifiers to improve the overall recognition accuracy. They proposed a method to adopt multiple sensors on distributed body locations to overcome this problem. The objective of the proposed system is to achieve higher recognition accuracy with "light-weight" signal processing algorithms, which run on a distributed computing-based sensor system comprised of a computationally efficient model.

Different from multiple sensors, a multimodal concept is proposed to reduce the number of sensors needed but at the same time, allow better accuracy than multiple sensors model. Bharti (2017) created a multimodal wearable sensor that can differentiate harmful activities from non-harmful activities. The model created achieved fairly close accuracy due to the leverage of three different sensing contexts for multi-modal sensing. One is the body locomotion (accelerometer and gyroscope), ambient environment (barometer), and location context (Bluetooth beacon) for classification. It is also worth noting that a context-aware age-specific HAR model would be a better solution than all age-mixed models.

Although wearable sensor HAR has gained far more popularity than Vision-based HAR, most research did not point out the inconvenience of wearing the sensor. Users of younger or older age may be reluctant to wear sensors around them, and this might limit the data collection to a certain extent. Also, it is worth pointing out that, there is also a need to create such a sensor using specific hardware such as gyroscope, barometer, and accelerometer, which may be difficult to obtain. Besides, the implementation of the advanced feature extraction algorithms and the complex classifiers have surpassed the computational capability of most current wearable sensor platforms (Gao et al., 2014). Therefore, there is a need to have a more convenient way to perform HAR while not affecting the accuracy of its prediction.

2.1.3 Smartphone-based HAR

Smartphone-based HAR is also another popular alternative for collecting raw acceleration data. Most smartphones like Android and Apple has built-in gyroscope and accelerometer to measure the triaxial acceleration of our body movement. It eliminates the problem of the inconveniency of wearable sensors and the privacy issues and challenges revolving vision-based HAR. There is also wide adoption of smartphone-based HAR done by other researchers (Anguita et al., 2013; Bayat et al., 2014; Brezmes, Gorricho, & Cotrina, 2009; Ignatov & Strijov, 2016; Kwapisz et al., 2011; Lockhart, 2014)

Kwapisz et al. (2011) described and evaluated a system that uses phone-based accelerometers to perform activity recognition. Data was collected from twenty-nine users, each carrying an Android phone in their pocket as they performed six activities: Walking, Jogging, Stairs-Up, Stairs-Down, Sitting, and Standing. Then, these time series data are aggregated into examples that summarize the user activity over 10-second intervals. The resulting training data is used to develop a predictive model for activity recognition. This paper permits us to gain useful insight into the habits of millions of users passively (requiring them to carry cell phones in their pockets). This paper pointed out that there is a wide range of applications, including automatic customization of the mobile device’s behavior based upon a user’s activity and generating a daily/weekly activity profile to determine if a user (usually an obese kid) is performing a healthy amount of exercise. Their work enabled a public platform of Android-based data collection called Wireless Sensor Data Mining (WISDM) to be developed.

In (Anguita et al., 2013), a group of 30 volunteers within an age bracket of 19-48 years. Each person performed the six activities (standing, walking, laying down, sitting, walking upstairs and walking downstairs) wearing the smartphone on the waist. A Samsung Galaxy S2 smartphone has been exploited for the experiments, as it contains an accelerometer and a gyroscope for measuring 3-axial linear acceleration and angular velocity respectively at a constant rate of 50Hz. Their work enables the creation of the public UCI HAR dataset.

In (Bayat et al., 2014), four subjects, two males and two females between 29 to 33 years of age, volunteered to participate in this research study. These subjects, each carrying a cell phone, performed all the six activities: Running, slow walking, fast walking, aerobic dancing, going upstairs and going downstairs. Similar to the previous two studies. However, both of these studies required researchers to collect data personally from a limited group of volunteers. The reliability of the data collected is questionable.

Lockhart (2014) utilized the WISDM dataset (Kwapisz et al., 2011) and Human Activity Sensing Consortium (HASC) dataset from 2010 to 2011 to analyze the learning curves for various model types to shows that very small amounts of personal data dramatically outperform best-case universal models. WISDM obtained raw acceleration data from 59 users who carried an Android phone containing an accelerometer to track the triaxial acceleration while they perform six activities which include walking, jogging, sitting, standing, climbing up and downstairs, and lying down.

Ignatov and Strijov (2016) chosen an open WISDM Actitracker dataset (Kwapisz et al., 2011) for training and performance evaluation of the proposed algorithm. This dataset, similar to the WISDM dataset, contains accelerometer data from Android cell phones. However, the data was collected from thirteen different users as they performed activities such as walking, jogging, stair climbing, sitting and standing, and consists of over a million-time series points measured by an accelerometer.

Brezmes et al. (2009) presented the implementation of a real-time classification system for some basic human movements using a conventional mobile phone equipped with an accelerometer. This study aimed to evaluate the capability of the current mobile phone to execute all the necessary pattern recognition algorithms such as K-nearest neighbors to classify the corresponding human movements in real-time. The server that process data is not involved in this approach, which allows the decentralization of human monitoring and only a few additional software that remotely reports the human monitoring was required.

The Smartphone-based approach has opened up a wider entry point for more researchers to gain a deeper understanding of HAR. More insight and better focus on the classification of algorithms and better models can be built thanks to the implementation of the smartphone approach. Public datasets such as WISDM and UCI have helped researchers to focus their attention on experiments with different classification algorithms and approaches such as neural networks to accurately predict human activity. This enables a more reliable and robust HAR model which can help in domains such as healthcare, crime prevention, and better AI video surveillance. Further detail of UCI and WISDM datasets will discuss in Section 2.4.

2.2 Features used in HAR

Hand-crafted features, learning features, and deep learning are some of the features that are explored in HAR. In this study, we will only explore feature that is performed in HAR using accelerometer data. Hand-crafted features refer to properties derived using various algorithms using the information present in the data itself. Hand-crafted feature relied on the specific expert domain which poses a series of challenges to researches, we will further explore it in Section 2.2.1. Learning feature is a set of techniques that allows a model to capture significant attributes needed for feature detection or classification from raw data. It can be classified into supervised learning and unsupervised learning. Supervised learning features are features that are learned using labeled input data, while unsupervised learning features are features that are learned using unlabelled input data. We will discuss more in Section 2.2.2. It is worth noting that learning features help lay the foundation work for deep learning, which is the method that we will be focusing on in our study which we will explore deeper in Section 2.3.

2.2.1 Hand-crafted features

A significant number of hand-crafted features are manually designed or handcrafted to minimize classification error and computational complexity (Friday Nweke, Ying Wah, & Alo, 2018). The effective performance of the human activity recognition system depends on the appropriate and efficient feature representation (Abidine, Fergani, Fergani, & Oussalah, 2018). Therefore, the extraction of efficient hand-crafted feature vectors from smartphone and wearable sensor data is paramount in the HAR model to decrease computing time and predict precise recognition accuracy (Friday Nweke et al., 2018).

In this section, we will look into two hand-crafted feature algorithms performed on accelerometer data. These are the Logic Binary Pattern (LBP) (Asuroglu, Acici, Erdas, & Ogul, 2017) and Empirical Cumulative Distribution Function (ECDF) (Hammerla, Kirkham, Andras, & Ploetz, 2013).

Logic Binary Pattern (LBP) method uses a mask on image to assign a new pixel value to mask's center pixel by looking at the preceding and subsequent pixels (Asuroglu et al., 2017). The center pixel is selected as a threshold value and according to neighborhood pixel values, a binary code is produced. In producing the binary code, all neighborhood pixel values are first compared against center pixel value. The neighborhood values that are greater or equal to center value produce 1 and neighborhood values that are smaller than the threshold value produces 0. After binary code generation, it is converted to a decimal value. This new value is finally updated to the center pixel in the image. Asuroglu et al. (2017) propose a discriminative framework for activity recognition that computes the Local Binary Patterns in a query signal and encodes the signal by the histogram of those patterns to feed a k-Nearest Neighbor classifier. This approach produces an accuracy of 91.37%, significantly higher than the previous method which uses global statistics extracted from the input signal.

Empirical Cumulative Distribution Function (ECDF) is computed from the empirical cumulative distribution of all axes (Hammerla et al., 2013). A practitioner specifies the percentiles of interest (e.g., k values between 0 and 100), and these values are interpolated from the ECDF. This produces k features per axis, and excellent performance is reported. Hammerla et al. (2013) proposed using ECDF to preserve crucial information about the distribution of accelerometer data, such as its spatial position and general shape in an efficient and transparent manner. The authors demonstrated that the ECDF representation clearly outperforms other approaches to feature extraction common for the domain. This paper demonstrated ECDF is well suited for mobile and embedded applications due to its outstanding result on accelerometer data and its low computational requirements.

The handcrafted- features were commonly used with traditional machine learning approaches for object recognition and computer vision like Support Vector Machines (SVM) (Cortes & Vapnik, 1995). However, hand-crafted features required specific expert or domain knowledge. Researches have to figure out or develop the right feature algorithm that is well-suited to their specific model. It is very challenging to measure the efficient performances of hand-crafted features across different applications and also require time-consuming features selection and dimensionality reduction methods specified above to obtain acceptable results (Ronao & Cho, 2016).

Therefore, the learning feature comes into play. Deep learning such as neural networks builds multiple layers of learning nodes which contain learning feature algorithm to extract features from data. They do not need to be supplied with such hand-crafted features, as they can learn the features from the data. Before discussing deep learning, first, we need to explore what is learning features and what is some of the examples of learning feature algorithms.

2.2.2 Learning Feature

As mentioned previously, a learning feature is a set of techniques that permits a system to find a pattern in a set of data. One of the learning features is Principle Component Analysis (PCA) which can learn a complete set of vectors (Smith, 2002). It locates patterns in data, and highlight the data's similarities and differences. Since patterns in data can be hard to locate in data of high complexity, where graphical representation is not available, PCA is a powerful tool for analyzing data. PCA is an unsupervised feature learning algorithm. The other main advantage of PCA is loss of information can be minimized once the data pattern is found and data compression is done. Sprager and Zazula (2000) build feature vectors for classification using dimension reduction on calculated cumulants by principal component analysis (PCA) on accelerometer data, which reports an accuracy of 90.3%.

The extension of PCA is the Sparse PCA. Sparse PCA is a specialized statistical analysis technique that focuses on multivariate data (Smith, 2002). It reduces data dimensionality by introducing sparsity structures to the input variance. One problem with PCA is that the principal component is usually linear combinations of all input variables, Sparse PCA solved this by finding linear combinations that contain a few input variables.

Another example of a learning feature is supervised dictionary learning. Supervised dictionary learning utilizes both the structure underlying the input data and the labels for optimizing the dictionary elements (Mairal, Bach, Ponce, Sapiro, & Zisserman, 2008). A supervised dictionary learning technique performs dictionary learning on classification problems by combining the dictionary elements, weights that represent data points, and classifier parameters for better optimization based on the input data. Wang, Chen, Hao, Peng, and Hu (2019) propose a new supervised class-specific dictionary learning based on a modified sparse model for action recognition, which achieves a fairly 86.6% accuracy.

2.3 Deep Learning in HAR

Deep learning has always been a hot topic in Human Activity Recognition (Ronao and Cho, 2016; Shi, Li, Zhou, and Liu, 2018; Wang et al., 2019). The extraction of hand-crafted features as discussed in the previous section poses the most challenging part in smartphone-based and wearable sensor HAR. Current HAR relies too much on hand-crafted features that sometimes they failed to predict complex activities accurately (Friday Nweke et al., 2018). Deep learning methods have been able to remove the dependency of the time-consuming hand-crafted feature that researchers faced (Friday Nweke et al., 2018).

Deep learning consists of multiple layers of learning nodes. The deep learning approach generally involves Deep Neural Model, Convolutional Neural Network and Recurrent Neural Network. For this study, we will only focus on Convolutional Neural Network (CNN) in Section 2.3.1. Recurrent Neural Network (RNN) in Section 2.3.2, Long Short-Term Memory (LSTM) in Section 2.3.2.1, and two LSTM variants which are Bi-directional Long Short-Term Memory (Bi-dir LSTM) in Section 2.3.2.2and Deep residue Bi-dir LSTM in Section 2.3.2.3.

2.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) or commonly referred to as Covnet is proposed by a French professor named Yann LeCun. A convolutional neural network consists of an input layer, multiple hidden layers, and one output layer **Figure 2.1** shows the architecture of CNN. The hidden layers of a CNN consist of a series of convolutional layers that convolute with a multiplication product. The activation function is commonly a ReLU layer (which we will discuss in detail in Chapter 3) and is subsequently followed by additional convolutions such as pooling layers, fully connected layers, and normalization layers. They are referred to as hidden layers because their inputs and outputs are masked or hidden by the activation function and final convolution.

The final convolution often involves backpropagation in order to weight the end product more accurately. Backpropagation is proposed by LeCun et al. (1989) to learn the convolution kernel coefficients directly from images of hand-written numbers. It allows learning to be fully automatic, has better performance than handcraft coefficient design, and was suitable for many image recognition problems and image types.

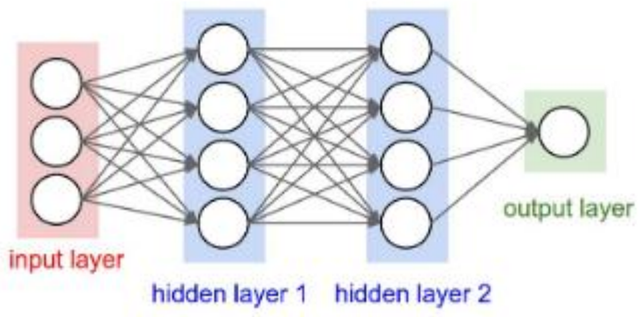

Figure 2.1: Convolutional Neural Network

CNN in HAR is already a well-researched area. For example, (Zeng et al., 2014) propose an approach based on CNN to extract human activity features while excluding any domain knowledge. The proposed approach can capture the local dependencies and scale-invariant features of activity signals. They utilize multichannel time series data to recognize the user’s activity and hand gestures.

Ronao and Cho (2016) construct a deep CNN by using the accelerometer and gyroscope triaxial sensor data to perform 6-axes one-dimensional (1D) convolution. Their network achieved 94.79% activity recognition accuracy on raw sensor data and 95.57% accuracy with additional Fast Fourier Transform (FFT) information on the sensor data. Lee, S. M., Cho, H., and Yoon (2017) also propose a one-dimensional (1D) Convolutional Neural Network (CNN)-based method for recognizing human activity with the use of user's smartphone triaxial accelerometer data. The three human activity data include walking, running, and staying still. They are gathered using a smartphone accelerometer sensor. The x, y, and z acceleration data are transformed into a vector magnitude data and used as the input for the 1D CNN to learn. The method showed an accuracy of 92.71%.

In most of the existing work, the 1D convolution operation is applied to yield single-variable (univariate) time series, while two-dimensional (2D) convolution and pooling operations are applied to multi-sensors or multi-modality to yield multiple-variable (multivariate) time series to capture local dependency along with both temporal and spatial domains for unimodal data. However, the existing CNN models with 2D operation handles different modalities for multimodal data in the same way as to how they handle. unimodal data. Therefore, there is a need to handle multimodal data in a more efficient way.

Ha and Choi (2016) propose two new variants of CNNs (CNNpf and CNN-pff), especially CNN-pff, for multi-modal data. Both partial weight sharing and full weight sharing are applied for CNN models in such a way that allows modality-specific characteristics and common characteristics across modalities to be learned from multi-modal (or multi-sensor) data. The outputs are eventually aggregated in the upper layers. They achieved a classification accuracy of 99.66% based on their model. It achieves significantly high performance with less number of parameters when compared to 1D operation.

Ji, Xu, Yang, and Yu (2013) develop a novel three-dimensional (3D) CNN model for action recognition which can extract features from both the spatial and the temporal dimensions. Their model is able to capture the motion information encoded in multiple adjacent frames by applying 3D convolution. The developed model generates multiple channels of information from the input frames, and combine the information from all channels to form the final feature representation. The model achieves an overall accuracy of 90.2%.

2.3.2 Recurrent Neural Network (RNN)

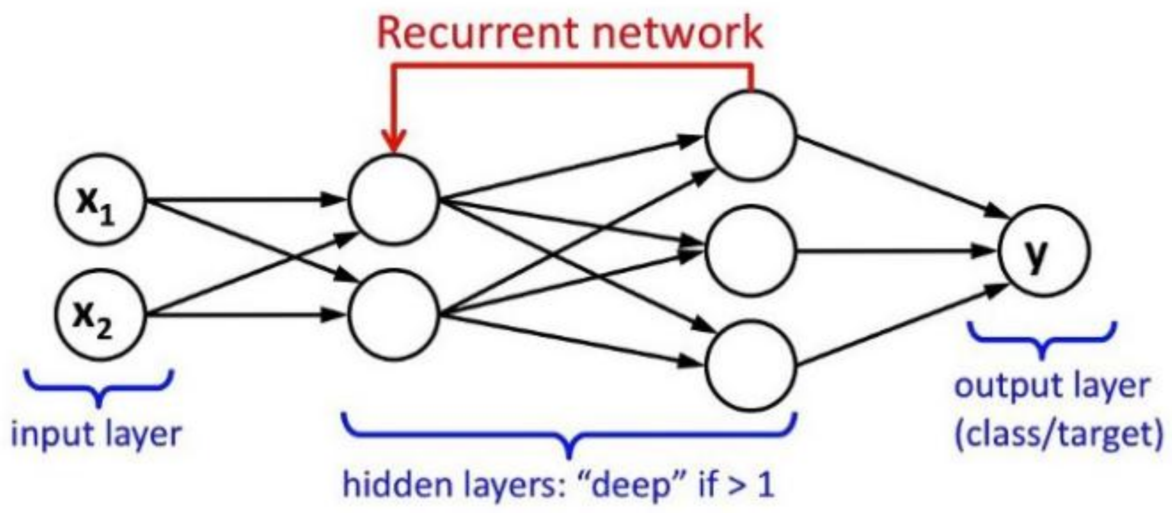

Figure 2.2: Recurrent Neural Network

Recurrent neural network (RNN) is inspired by Hopfield Net developed by John Hopfield (Hopfield, 1982). It was developed to model sequential data such as time series or raw sensor data (Friday Nweke et al., 2018). **Figure 2.2** shows the RNN architecture. RNN consists of a temporal layer that captures sequential information and learns complex changes using the hidden nodes of the recurrent cell. The hidden nodes can change based on the information available to the network, and this information is constantly updated to reflect the current status of the network. RNN predicts the current hidden state by estimating the next hidden state when the previous hidden state is activated. However, as pointed out by Pascanu, Mikolov, and Bengio (2012), and Guan and Plötz (2017), the model is difficult to train and suffer from vanishing or exploding gradients limiting its application for modeling long-time activity sequence and temporal dependencies in sensor data.

2.3.2.1 Long Short-Term Memory (LSTM)

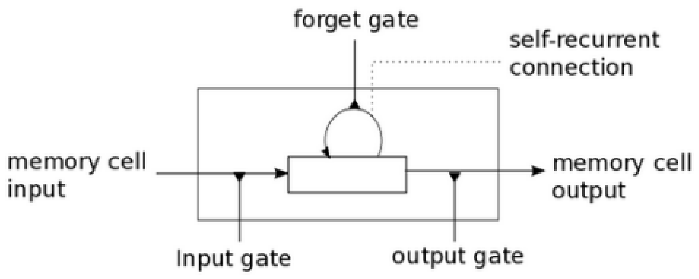

Figure 2.3: Long Short-Term Memory

RNN has the problem of forgetting the first inputs due to memory cells have a short-term memory, this is what is commonly addressed as vanishing gradient problem (Hernández et al., 2019; Yu & Qin, 2018). LSTM solves this problem by implementing four gates which we will discuss in detail about each gate in Chapter 3. The four gates are the forget gate, input gate, state gate (or a neuron with a self-recurrent connection) and output gate in each node, as can be seen in **Figure 2.3**. In the LSTM node, the forget gate decides whether or not to retain information when new information arrived from the previous node. The input gate consists of sigmoid function and its output is channeled to the state gate. The state gate creates a vector with different values, these values are combined with the output of the input gate to update the cell state. The output gate represents the information from a cell state that combined with past information. The update status consists of the forget gate output and the output of both input gate combined with the state gate. The output gate's information combined with the update status and is passed to the next LSTM node.

Chen, Zhong, Zhang, Sun, and Zhao (2016) proposed an LSTM-based feature extraction approach to recognize human activities using tri-axial accelerometers data. The experimental results on the WISDM public datasets indicate that their LSTM-based approach is practical and achieves 92.1% accuracy as compared to other models. In short, LSTM takes in past information to predict the outcome of a HAR model. However, significant information may not be captured as human motion is continuous (Yu & Qin, 2018).

2.3.2.2 Bi-directional Long Short-Term Memory (Bi-dir LSTM)

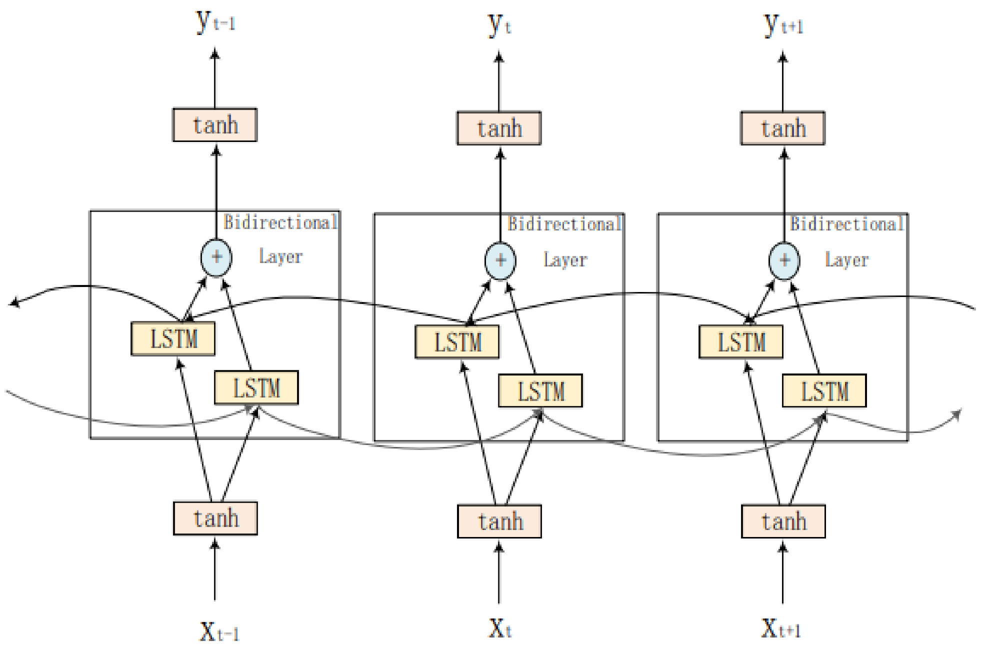

Figure 2.4: Bi-directional Long Short-Term Memory

Bi-directional Long Short-Term Memory (Bi-dir LSTM) as shown in **Figure 2.4** is a variant of LSTM. However, different from LSTM who only takes in past information, Bi-dir LSTM takes into account past information and future information, as shown in Figure. In other words, Bi-dir LSTM is stacked into layers both horizontally and vertically. A single LSTM node can take in information from the horizontal layer for both past and future information and also from a vertical layer which is the lower hidden layer.

Yu and Qin (2018) proposed bidirectional LSTM on HAR using the inertial sensor in the smartphone, achieving an accuracy of 93.79%. The result is comparable to the study conducted by Ronao and Cho (2016) who used a deep learning convolutional neural network (CNN) on raw data and achieve an accuracy of 94.79%. Similar work has also been done by Hernández et al. (2019) who proposed a bi-dir LSTM network that takes in accelerometer and gyroscope data from a smartphone. They achieve and accuracy of 92.67%.

2.3.2.3 Deep Residue Bi-directional Long Short-Term Memory

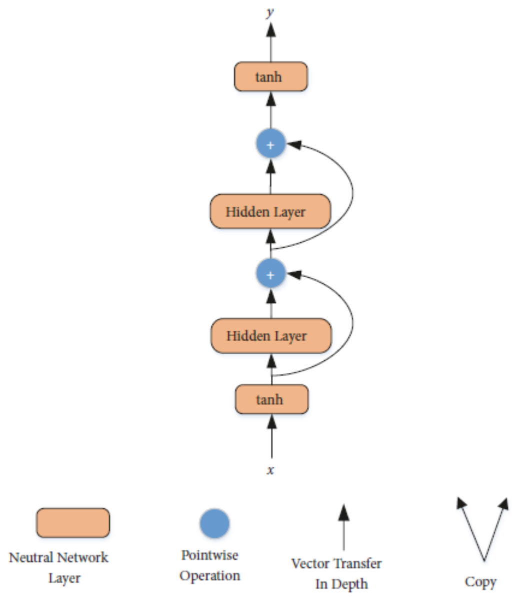

Figure 2.5: Deep Residue Bidir-LSTM

Zhao, Yang, Chevalier, Xu, and Zhang (2018) proposed a new concept of Bidir-LSTM which is called the deep residue Bidir-LSTM as shown in **Figure 2.5**. The residual network has no gates but residual connections. This residual connection ensures that the gate is never closed and the network is free of the parameter. All information is always channeled to the additional residual functions to be learned.

An important advantage of residual networks, as pointed out by Zhao et al. (2018) is that they are much easier to train because the gradients can be passed through the layers more directly with the addition operator that enables them to bypass some layers that would have otherwise been restrictive. It enables both better training and a deeper network because residual connections do not impede gradients and still contribute to refining the output of several stacked layers composed of such residual connections.

On top of a collection of residual connections is a bottleneck where the next layers stop being residual and where batch normalization is generally applied to normalize and restrict the usage of the feature space represented by the layer (Pascanu et al., 2012). The lower information can transmit to the upper layer directly through a shortcut connection. The paper reported an accuracy of 93.6% due to the residue and bidirectional direction (Zhao et al., 2018).

2.3.3 Comparison Study of Existing Deep Learning Models

For simplicity, a table to summarize the discussion on CNN and the three variants of RNN is constructed, as shown in **Table 2.1**. It is worth noting that although it seems each new model introduced is significantly better than the previous model, all the models are inherently stochastic in nature, meaning it can have a random probability distribution or pattern that can be analyzed statistically but may not be predicted precisely. Hardware specifications used also have to be taken into account.

Table 2.1: Comparison Study between Existing Deep Learning Models

Deep Learning Method	**Authors (Year)**	**Dataset**	**Accuracy (%)**	**Remark**
CNN	(Ronao & Cho, 2016)	UCI	94.79	• Use 1D convolution
	(Lee, S. M., Cho, H., & Yoon, 2017)	Self-collected	92.71	• Use 1D convolution
	(Ha & Choi, 2016)		99.66	• Use 2D convolution • Multi-sensor
	(Ji et al., 2013)	KTH	90.20	• Use 3D convolution • Capture motion in multiple adjacent frames
LSTM	(Chen et al., 2016)	WISDM	92.10	• Better than RNN • Takes in past information to predict the outcome

Bidir-LSTM	(Yu & Qin, 2018)	UCI	93.79	• Takes in past and future information (bidirectional) to predict the outcome
	(Hernández et al., 2019)	UCI	92.67	• Takes in past and future information (bidirectional) to predict the outcome
Deep Residue Bidir-LSTM	(Zhao et al., 2018)	UCI	93.60	• Utilizes deep residue and bidirectional information

2.4 Public Dataset Comparison (WISDM and UCI)

The UCI dataset comprises six activities: Walking, Walking upstairs, Walking downstairs, Sitting, Standing and Laying down, while the WISDM dataset comprises Walking, Jogging, Sitting, Standing, Walking upstairs, and Walking downstairs. In the UCI case, 30 volunteers aged 19-48 are asked to perform these activities smartphones on their waist. In the WISDM case, 29 volunteers are asked to perform the respective activities using a smartphone in their leg pocket. In the UCI case, data was recorded at a frequency of 30Hz and was trimmed into windows of 128 time steps (2.5 seconds). In the WISDM case, data was recorded at a frequency of 50Hz and was trimmed into windows of 200 time steps (10 seconds). Both datasets have their data split into 70% for training and 30% for testing.

We selected 6 features from the total 561 linear hand-made preprocessed features from the UCI dataset: tri-axial gravity acceleration and tri-axial body acceleration from the accelerometer. For WISDM datasets, 3 features comprising of tri-axial gravity acceleration is selected. These raw signals with a time component fall in the time domain instead of the frequency domain. Denoising median filters were used to pre-process the sensor data, clipping the approximately 20 Hz mark. Additionally, those sensor data were sampled with sliding windows of 2.56 seconds. An overlap of 50% of those windows was provided to ease training. A comparison study of the public dataset WISDM and UCI is summarized in **Table 2.2**.

Table 2.2: Comparison Study between WISDM and UCI dataset

Feature	WISDM dataset	UCI dataset
Activities	Walking, Jogging, Stairs-Up, Stairs-Down, Sitting, and Standing	Walking, Walking upstairs, Walking downstairs, Sitting, Standing and Laying down
Number of volunteers	29	30
Location of smartphone	Front leg pocket	waist
Collect	Triaxial gravity acceleration from accelerometer	Triaxial gravity acceleration from the accelerometer and triaxial body acceleration and triaxial angular velocity from the gyroscope
Frequency of data collection	20Hz (20 samples per second)	50Hz (50 samples per second)
Time steps	10 second (200)	2.56 second (128)
Overlap window, moving window	Defined by user, usually 50% (Lee & Kwan, 2018), 200 sample	50%, 128 samples
Number of instances	1,098,207	10,299
Number of attributes/features	6 (raw data file) 46 (transformed data file)	561
Training-testing split	Defined by user, usually 70-30	70-30
References	Kwapisz et al. (2011)	Anguita et al. (2013).

2.5 Chapter Summary

Most of the recently published papers focus on one aspect of neural networks such as RNN for one experiment and CNN for another. However, we can safely conclude that the neural network is better at extracting more detailed features or commonly referred to as deep features. These deep features are more closely representative of their respective activities to help the model to accurately classify each action than handcrafted features.

The two types of neural networks- CNN and RNN have their advantages and disadvantages. For example, RNN is suitable for extracting time-domain features, while CNN is suitable for image processing. If we manage to get the best of both worlds, it can be argued that utilizing both types of neural networks will be better in extracting more detailed features and thus better classification of the human activities. To prove our hypothesis, we propose to build a fusion of CNN and RNN model for better human action recognition.

In the next chapter, we will focus more on the individual architecture and formula of CNN, LSTM and Bi-dir LSTM and discuss our proposed model in detail.

CHAPTER 3

THE PROPOSED HUMAN ACTION RECOGNITION MODEL USING FUSION OF CNN AND BIDIRECTIONAL LSTM

In this chapter, the different components of the proposed model are first discussed. An overview of CNN is discussed in Section 3.1, followed by its architecture and formulation in Section 3.1.1. The Max Pooling layer and Dropout layer are also discussed in Section 3.1.2 and Section 3.1.3 respectively. Next, an overview of LSTM and its architecture is discussed in Section 3.2 and Section 3.2.1, followed by a discussion of Bidir-LSTM and its formulation in Section 3.3 and Section 3.3.1 respectively. Besides that, the regularization or weight decay is explored in Section 3.4. Different activation functions in the deep learning model are discussed in Section 3.5, followed by different optimization in Section 3.6. Lastly, the proposed model is discussed in Section 3.7.

3.1 Convolutional Neural Network (CNN)

As discussed earlier, Convolutional Neural Network (CNN) or commonly referred to as Covnet is proposed by LeCun et al. (1989). A convolutional neural network consists of an input layer, multiple hidden layers, and one output layer. The hidden layers of a CNN consist of a series of convolutional layers that convolute with a multiplication product. The activation function is commonly a Relu layer and is subsequently followed by additional convolutions such as pooling layers, fully connected layers, and normalization layers. They are referred to as hidden layers because their inputs and outputs are masked or hidden by the activation function and final convolution.

The final convolution often involves backpropagation in order to weight the end product more accurately. Backpropagation is also proposed by LeCun et al.

(1989) to learn the convolution kernel coefficients directly from images of hand-written numbers. It allows learning to be fully automatic, has better performance than handcrafted-feature extraction algorithm, and was suitable for many image recognition problems and image types.

3.1.1 Architecture and Formulation of CNN

Convolutional neural networks perform convolution operations instead of matrix manipulation. In our model, if $x_i^0 = [x_1, \dots, x_N]$ is the accelerometer data input vector and N is the number of values per window, the output of the first convolutional layer is:

$$c_i^{1,j} = \sigma\left(b_j^1 + \sum_{m=1}^{M} w_m^{1,j} x_{i+m-1}^{0,j}\right) \tag{3.1}$$

Where l is the layer index, σ is the activation function, b_j is the bias term for the jth feature map, M is the kernel/filter size, and w_m^j is the weight for the feature map j and filter index m. Similarly, the output of the lth convolutional layer can be calculated as follows:

$$c_i^{l,j} = \sigma\left(b_j^l + \sum_{m=1}^{M} w_m^{l,j} x_{i+m-1}^{l-1,j}\right) \tag{3.2}$$

3.1.2 Max Pooling

A summary statistic of nearby outputs is derived from $c_i^{l,j}$by the pooling layer. The pooling operation used in this paper, max-pooling, is characterized by outputting the maximum value among a set of nearby inputs, given by

$$p_i^{l,j} = \max_{r \in R}(c_{i \times T + r}^{l,j}) \tag{3.3}$$

Where R is the pooling size, and T is the pooling stride. Several convolutional and pooling layers can be stacked on top of one another to form a deep neural network architecture. These layers act as a hierarchical feature extractor; they extract discriminative and informative representations with respect to the data, with basic to more complex features manifesting from bottom to top.

3.1.3 Dropout layer

Dropout modifies the network itself to avoid overfitting instead of modifying the cost function. It works by randomly and temporarily deleting a node in the network while leaving input and output neurons intact, which makes it equivalent to training a lot of different neural networks. It also forces neurons not to rely on the presence of other particular neurons, enabling the learning of more robust features. Dropout is accompanied by an include probability and is done independently for each training example. The proposed CNN-BidirLSTM architecture, dropout is applied before the fully-connected layer (details in Section 3.7).

3.2 LSTM

As mentioned earlier, RNN has the problem of forgetting the first inputs due to memory cells have a short-term memory, this is what is commonly addressed as vanishing gradient problem (Hernández et al., 2019). LSTM solves this problem by implementing four gates. The four gates are the forget gate, input gate, state gate, and output gate in each node.

In the LSTM node, the forget gate decides whether or not to retain information when new information arrived from the previous node. The input gate consists of sigmoid function and its output is channeled to the state gate. The state gate creates a vector with different values, these values are combined with the output of the input gate to update the cell state. The output gate represents the information from a cell state that combined with past information. The update status consists of the forget gate output and the output of both input gate combined with the state gate. The output gate's information combined with the update status and is passed to the next LSTM node. In short, LSTM takes in past information. However, significant information may not be captured as human motion is continuous (Yu & Qin, 2018).

3.2.1 Architecture and Formulation of LSTM

An LSTM cell is made up of four gates (forget, input, state, and output) and one cell state that provides additional interactions. In the following equations by Yu and Qin (2018), W_q represents the weights of the input and h_q represents the recurrent connection, q can be the input gate (*i*), output gate (*o*), the forget gate (*f*) or the memory cell (*c*). X represents the input vector to the LSTM unit. Our architecture of LSTM is based on the architecture proposed by Hernández et al. (2019). The forget gate is responsible for taking information from the past that is not important and should be forgotten, the recurrent input h_{t-1} and current input x_t multiplied by their weights are the input of a sigmoid (σ) function, the (σ) output (h_{f_t}) is a number between 0 and 1 that is multiplied with cell state (c_{t-1}), if (h_{f_t} = 1) the LSTM keep this new information, otherwise, if (h_{f_t} = 0) the LSTM forget completely this new information:

$$h_{f_t} = \sigma\left(w_f h_{t-1} + W_f x_t\right) \tag{3.4}$$

The input gate consists of a sigmoid function and its output h_{i_t} represents the value that is going to update the LSTM cell:

$$h_{i_t} = \sigma(W_i h_{t-1} + W_i x_t) \tag{3.5}$$

The state gate creates a vector with different values, these values are combined with h_{i_t} and do a status update:

$$h_{c_t} = \tanh(h_l h_{t-1} + W_c X_t) \tag{3.6}$$

The output gate represents the information from the cell state that should come out immediately combined with past information:

$$h_{o_t} = \ \sigma(W_o h_{t-1} + W_o X_t) \tag{3.7}$$

The update states consist in forget what is to be forgotten and add what is to be added:

$$C_t = h_{f_t} * C_{t-1} + h_{i_t} * h_{c_t} \tag{3.8}$$

And the LSTM output combines short and long terms together with

$$h_t = h_{o_t} * \tanh(C_t) \tag{3.9}$$

An LSTM network takes into account past information, however, if present and future information are available (offline classifier), it would only take advantage of past information; future information would not be used.

3.3 Bi-dir LSTM

Bi-directional Long Short-Term Memory (Bidir-LSTM) is a variant of LSTM. The improvement of the Bidir-LSTM is that the current output is related to the previous information and subsequent information. The Bidir-LSTM structure is made up of some of Bidir-LSTM cells. For a Bidir-LSTM layer, it gets information from the horizontal direction (both the past information and future information) and the vertical layer (from the lower layer).

3.3.1 Architecture and Formulation of Bi-dir LSTM

The architecture followed LSTM architecture (Yu and Qin, 2018). The only difference is there are forward sequences $\overrightarrow{h}$ and backward sequences $\overleftarrow{h}$ in the hidden layer. At time t, the hidden layer and the input layer can be defined as follows:

$$\overrightarrow{h} = \sigma(U_{\overrightarrow{h}}x_t + W_{\overrightarrow{h}}\,\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{3.10}$$

$$\overleftarrow{h} = \sigma(U_{\overleftarrow{h}}x_t + W_{\overleftarrow{h}}\,\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \tag{3.11}$$

$$y_t = \sigma(V_{\overrightarrow{h}}\overrightarrow{h}_t + V_{\overleftarrow{h}}\,\overleftarrow{h}_t + b_y) \tag{3.12}$$

Where *U*, *W*, *V* denote the weight metrics from the input layer to the hidden layer, from the hidden layer to subsequent hidden layer and from the hidden layer to the output layer, σ denotes the activation function and *b* denotes the bias term.

3.4 Weight decay/Regularization

Weight vector can get stuck in a local minimum by large weights easily since gradient descent only permits small changes to the direction of optimization. This will eventually make it hard to explore the weight space. Weight decay or L2 regularization is a regularization method that adds a penalty for weight size to the loss function. For each set of weights, the penalizing term $\lambda \sum_{w} w^2$ is added to the cost function:

$$E = E_0 + \lambda \sum_{W} w^2 \tag{3.13}$$

Where E_0 is the unregularized cost function, and λ is the weight decay coefficient. With this new cost function, the learning rule becomes:

$$w_i = (1 - \eta\, \lambda) w_i - \eta \frac{\partial E_0}{\partial w_i} \tag{3.14}$$

Where $1 - \eta\, \lambda$ is the weight decay factor.

Momentum-based gradient descent introduces the notion of velocity for the parameters being optimized, in such a way that the gradient changes the velocity rather than the position in weight space directly. Let $v = [v_i, \dots, v_k]$ as velocity variables, one for each weight variable. The gradient descent update rule becomes:

$$v \rightarrow v' = \mu v - n \nabla E \tag{3.15}$$

$$w \rightarrow w' = w + v' \tag{3.16}$$

Where μ is the momentum coefficient

3.5 Activation function

The activation function is used to determine the output of neural network like 1 or 0. It maps the resulting values in between 0 to 1 or -1 to 1, based on the function. It allows generalization or adaptation of a variety of data. It also allows differentiation of output. The most common activation employed in CNN is the Rectified Linear Unit (ReLU) function.

3.5.1 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) has a range of 0 to infinity. The function and its derivative are both repetitive. However, the issue is that all the negative values become zero immediately which leads to underfitting. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which means no learning can be performed, this will cause the dead ReLU problem where there is a lot of neuron is not functioning at all in the whole process.

3.5.2 Sigmoid function

Sigmoid function is a non-linear activation function that looks like an S curve. The main reason why it is used in LSTM is that it has a range from 0 to 1. In LSTM, the forget gate has to either set 0 to forget the new information or set to 1 to retain the new information. In other words, it is useful for models to predict the probability of an output. The same can be said to its input gate and output gate, which can output either to let no information flow (0) or let information flow (1) to update the LSTM node. The function is differentiable. That means, the slope of the sigmoid curve can be drawn at any two points.

3.5.3 Tanh function

Tanh, on the other hand, has a range from -1 to 1. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable. The function is repetitive while its derivative is not repetitive. The tanh function is mainly used in classification between two classes. Both tanh and logistic sigmoid activation functions are used in feed-forward nets.

3.5.4 Softmax function

Softmax function is a function that takes in an input of vector of a set of real numbers and normalizes it into a probability distribution consisting of a set of probabilities proportional to the exponentials of the input numbers. That is, before softmax is applied, some vector components could be negative, or greater than one; and might not sum to 1. After applying softmax, each component will be in the interval (0,1), and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities. Softmax is often used in neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes. The Softmax function is a more generalized logistic activation function that is used for multiclass classification while sigmoid is useful for binary classification. Softmax is thus used in the proposed model to classify the six activities

3.6 Optimizer

Optimizers are used in neural networks to optimize the model, either in evaluating the gradient to prevent vanishing or exploding gradient problem or optimize learning rate so that the model abstract representation can be learned more quickly. The most famous optimizer is the Adaptive Moment Estimation (Adam). Different optimization functions are discussed in the following section.

3.6.1 Stochastic Gradient Descent (SGD)

Stochastic gradient descent (SGD) is an iterative method for optimizing an objective function with suitable smoothness properties such as differentiable or subdifferentiable properties. It is stochastic because the method uses randomly selected (or shuffled) samples to evaluate the gradients, hence SGD can be regarded as a stochastic approximation of gradient descent optimization. (Herbert Robbins and Sutton Monro, 1951)

3.6.2 Root Mean Square Propagation (RMSProp)

Root Mean Square Propagation (RMSProp) is also a method in which it divides the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. Thus, the learning rate is adapted for each of the parameters.

3.6.3 Adaptive Moment Estimation (Adam)

Adaptive Moment Estimation (Adam) is an extended version of the RMSProp optimizer. In this optimization algorithm, running averages of both the gradients and the subsequent gradients are used. (Kingma & Ba, 2015)

3.7 Proposed Neural Network Model

In our study, we proposed to set up a neural network model integrating CNN and Bidirectional-LSTM with time-domain features. CNN could well capture local dependency and maintain scale invariance, so it is suitable for discovering relationships in spatial dimensions. However, CNN lacks the capability to capture temporal dependency in time-series sensory data.

RNN (wither LSTM or gated recurrent unit-GRU) as memory units are designed to model time series data, and it is suitable for discovering relationships in the temporal dimension. However, RNN takes a sequence of feature vectors as input, ignoring the local dependencies from the original sensory data that form a feature vector. By fusing the two deep learning methods, we may have a more significant result. We used two public datasets which are the UCI HAR dataset and WISDM dataset. The differences of both datasets are tabulated in Section 2.4. In this study, the research only focuses on the accelerometer data, instead of the gyroscope data. The model is modified to only retrieve the accelerometer data as input and ignore the gyroscope data. WISDM and UCI datasets will focus on the raw triaxial acceleration data.

3.7.1 Architecture of CNN-Bidirectional LSTM model

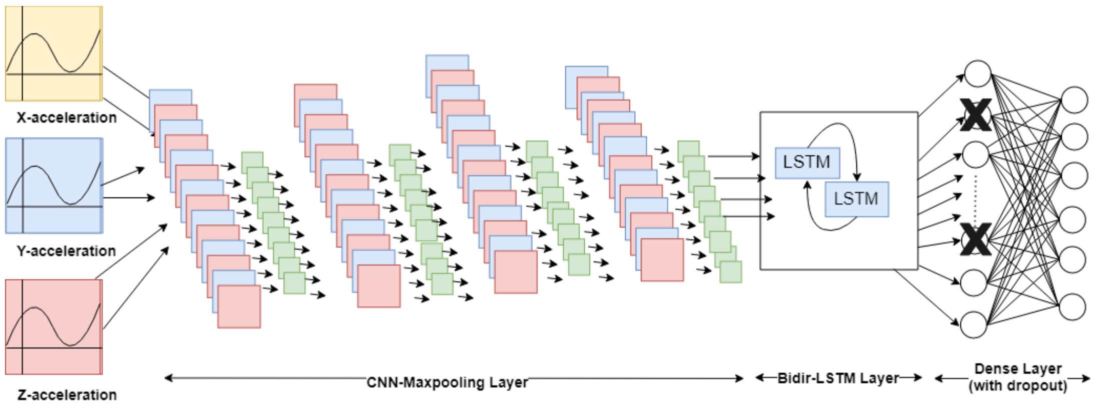

Figure 3.1: CNN-Bidir-LSTM model

Figure 3.1 shows the proposed model of CNN-Bidir-LSTM and the architecture can be summarized as followed:

$$C(m,k,v) - M(m) - C(m,k,v) - M(m) - C(m,k,v)$$
$$-M(m) - C(m,k,v) - M(m) - B(L(m)) - Drop(m) - D(m,v)$$
$$- D(m,v)$$

Where $C(m)$ denotes a convolutional operation with m feature maps/kernel, k kernel size and v activation function, $M(m)$ denotes a max-pooling operation with pool size set to m, $B(L(m))$ denotes bidirectional LSTM operation with m units, $Drop(m)$ denotes the dropout layer with m dropout rates, $D(m,v)$ denotes dense or fully connected operation with m units and v activation function.

The model is built using four layers of CNN-Maxpool layers followed by a Bidir-LSTM layer, Dropout layer, and two Dense layers. The first four layers are built to extract deeper features of the accelerometer raw data before outputting it to the Bidir-LSTM layers. The four CNN-Maxpool layers are proven to be able to extract more meaningful data and learn more effectively from the dataset.

Due to the structure of the Bidir-LSTM layer, it will utilize the information from the horizontal direction (both the past information and future information) and vertical layer (from the lower layer). This will prevent the vanishing gradient problem and also allow the output to not only related to the previous information as a baseline, but also to the subsequent information. Bidir-LSTM is selected as it has proven it worked best in time domain data (Yu and Qin, 2018). Both UCI and WISDM’S raw signals with a time component fall in the time domain instead of the frequency domain.

Next, the dropout layer modifies the network itself to avoid overfitting. It works by randomly and temporarily deleting a node in the network while leaving input and output neurons intact, which makes it equivalent to training a lot of different neural networks. It also forces neurons not to rely on the presence of other particular neurons, enabling the learning of more robust features.

Lastly, the two Dense layers which represent the feature vector of input ensure the output of the classification of the six activities of each dataset.

By default, ReLU is used as the activation function for CNN layers and the first Dense layer, while Softmax will be used as the final activation function of the last Dense layer. The learning rate is fixed as 0.001, and the weight decay set to 0. The proposed model is later used with ten different hyperparameter configuration settings for two datasets and the best model is selected based on the highest validation accuracy on the training set. The result of the proposed model is determined by the final evaluation of the model using the test set. The best model for both datasets is compared with other proposed classification method.

3.8 Chapter Summary

In this chapter, the architecture and the formulation of CNN, LSTM, and Bidir-LSTM are discussed. LSTM addressed the vanishing gradient problem by utilizing the previous information with the combination of forget gate, input gate, state gate, and output gate. Bidir-LSTM is capable of utilizing previous and future information to extract time-series features. Weight decay or regularization which penalizes large weights used in neural networks is also adequately defined. Different activation functions such as tanh and sigmoid, are also discussed, followed by a brief review of various optimizers. Finally, the proposed model is exhibited and discussed. In the next chapter, the experiment is set up and the result is exhibited and discussed.

CHAPTER 4

EXPERIMENT

In this chapter, the experiment setup is first discussed (Section 4.1), followed by the training and testing set information (Section 4.2). The main experiment of ten different configuration settings for both datasets to determine the best model is explored and compared with other classification methods in Section 4.3. Additional experiment to see the effect of hyperparameters on the performance model is also explored and evaluated in Section 4.4.

4.1 Experiment Setup

The information regarding the software and hardware used in this experiment are tabulated in **Table 4.1**. All the software or program used can be downloaded online for free as they are all open-source projects. The laptop used has a fairly good processor and a fairly good Random Access Memory (RAM). Apart from that, it is worth noting that a typical laptop can be used to run the proposed model as long it has a minimum 8GB requirement and contain an i5 Intel processor.

Table 4.1: Hardware and Software Information

Software	**Hardware**
Jupyter notebook	Lenovo Yoga 530
Anaconda Python	Intel Core i7-8550u processor, 1.85GHz
Keras, Numpy, Scipy, Seaborn and Pandas library	16GB RAM
	64-bit Microsoft Operating System, x86-based processor
	500GB SSD

Anandonca python is a popular distribution that simplifies package management and deployment. Keras is a high-level API to build and train deep learning model. Jupyter notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. The Jupyter notebook is used to visualize various configuration results of this study while running Keras as the backend. Both of them are accessed using Ananconda prompt which simplified the importing of various Python libraries such as Numpy, Scipy, Seaborn, and Pandas

4.2 Training and Test Sets Information

The data in UCI and WISDM is split into 70% training set and 30% test set. The training set is split into 80% for training and 20% for validation. Both training and test sets are summarized in **Table 4.2**.

Table 4.2: Training and Test Sets Information

	UCI dataset	**WISDM dataset**
Training	7352 (Train on 5881 samples, validate on 1471 samples)	20868 (Train on 16694 samples, and validate on 4174 samples)
Testing	2947	6584
Time step	128	80
Number of features	6 (xyz body acceleration and xyz gravity acceleration)	3 (xyz gravity acceleration)

UCI has 7352 training data and 2947 test data. The model will train on 5881 samples and validate on 1471 samples. The time step is fixed at 128. There are six features from 561 features in UCI which are the triaxial body acceleration and the triaxial gravity acceleration. WISDM, on the other hand, has 20868 training data and 6584 data. The model will train on 16694 samples and validate on 4174 samples. The time step is set to 80. There are only three features extracted which are the triaxial gravity acceleration. Both datasets' features are raw signal data which work best with the deep learning feature of the proposed model.

4.3 Hyperparameter Configuration Experiment

Ten different hyperparameter configurations are set up in order to choose the best model to represent this research for both public datasets. The configuration experiment for the UCI dataset is set up as shown in **Table 4.3**. For the WISDM dataset, the configuration is set up as shown in **Table 4.4**. The model is trained on both datasets using the hyperparameter configuration setting below to determine the best validation accuracy.

Table 4.3: UCI Hyperparameter Configuration Set

	c0	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**
Feature map	32	64	64	88	96	128	128	164	196	230
Kernel size	3	2	3	3	2	5	3	3	3	2
Bidirlstm	128	128	64	128	186	64	128	78	64	88
Dropout	0.1	0.4	0.5	0.4	0.5	0.3	0.5	0.7	0.2	0.5
1st Dense	128	128	128	128	128	128	128	128	128	128
2nd Dense	6	6	6	6	6	6	6	6	6	6
Epochs	30	25	50	50	25	40	40	30	30	30
Learning rate	0.001	0.005	0.005	0.003	0.001	0.001	0.001	0.002	0.004	0.001
Decay	0.0001	0.001	0.0001	0	0	0	0	0.00005	0.001	0.001
Batch size	128	64	128	64	32	32	64	64	64	32

Table 4.4: WISDM Hyperparameter Configuration Set

	c0	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**	**c9**
Feature map	20	40	50	60	80	80	100	140	200	220
Kernel Size	3	2	5	2	2	2	2	3	3	3
Bidirlstm	50	100	100	100	80	160	100	200	100	80
Dropout	0.2	0.5	0.3	0.4	0.5	0.5	0.3	0.5	0.7	0.1
1st Dense	80	80	80	80	80	80	80	80	80	80
2nd Dense	6	6	6	6	6	6	6	6	6	6
Epochs	30	25	40	25	30	30	50	30	30	30
Learning rate	0.004	0.001	0.001	0.005	0.004	0.001	0.003	0.001	0.001	0.001
Decay	0.001	0	0	0.001	0.0001	0.001	0.00005	0	0	0.0001
Batch size	40	200	50	50	80	100	200	80	50	100

In order to make a justifiable comparison with the proposed model, other state-of-the-art classification methods in Chapter 2 are first normalized and run in the same environment as the proposed model to fit the experiments. It is worth noting that only the smartphone-based HAR and deep learning models are selected. The normalized architectures and information of other classification methods are tabulated in Table 4.5. Based on **Table 4.5**, *C(n)* denotes the Convolutional layer with n feature maps, *MAX* denotes the Maxpooling layer, DL(n) denotes Fully Connected Dense layer with n nodes, and *L(n)* denotes the LSTM layer with n nodes, and *B(L(n))* denotes the bidirectional LSTM layer with n nodes.

The default epoch is set to 30. The learning rate is followed as mentioned in the papers, if none is provided, a default learning rate of 0.001 is given. Similarly, the default dropout rate and default weigh decay are both set to 0 and 0.5 respectively. The accuracy of each model proposed by other authors is set as a benchmark and later evaluated. The default configuration is bolded in the table. The batch size for UCI is set to 128 and WISDM is set to 80, equivalent to their respective timestep.

Table 4.5 Normalized Architecture and Information of Existing Classification Methods

Model	**Authors**	**Architecture**	**Learning rate**	**Drop out rate**	**Weight decay**	**Epochs**
CNN	(Ronao & Cho, 2016)	C(96)-MAX-C(96)-MAX-C(96)-MAX-DL(1000)	0.01	0.8	0.00005	30
CNN	(Lee, S. M., Cho, H., & Yoon, 2017)	C(128)-MAX-C(128)-MAX-DL(384)	**0.001**	0.5	0	30
LSTM	(Chen, Zhong, Zhang, Sun, & Zhao, 2016)	L(100)-L(100)-DL(128)	**0.001**	**0.5**	**0**	30
Bidir-LSTM	(Yu & Qin, 2018)	B(L(28))-B(L(28)-B(L(28))-DL(128)	**0.001**	**0.5**	**0**	30
Bidir-LSTM	(Hernández, Suárez, Villamizar, & Altuve, 2019)	B(L(175))-B(L(175)-B(L(175))-DL(128)	0.001	**0.5**	**0**	30

4.3.1 UCI dataset Result and Discussion

The proposed model of CNN-Bidir-LSTM is run using the configuration setting in **Table 4.3** for the UCI dataset.

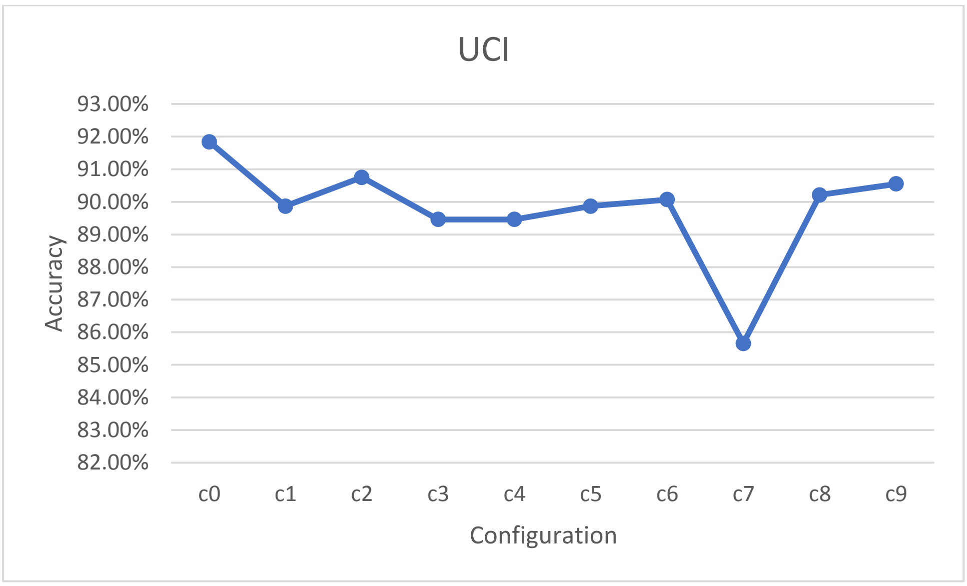

Figure 4.1: Configuration Result using UCI dataset

Table 4.6: Result Comparison using UCI dataset

Model	**Authors**	**Architecture**	**Accuracy**
CNN	(Ronao & Cho, 2016)	C(96)-MAX-C(96)-MAX-C(96)-MAX-DL(1000)-DL(6)	89.41%
CNN	(Lee, S. M., Cho, H., & Yoon, 2017)	C(128)-MAX-C(128)-MAX-DL(384)-DL(6)	89.24%
LSTM	(Chen, Zhong, Zhang, Sun, & Zhao, 2016)	L(100)-L(100)-DL(128)-DL(6)	89.79%
Bidir-LSTM	(Yu & Qin, 2018)	B(L(28))-B(L(28)-B(L(28))-DL(128)-DL(6)	89.07%
Bidir-LSTM	(Hern ández, Su árez, Villamizar, & Altuve, 2019)	B(L(175))-B(L(175)-B(L(175))-DL(128)	87.41%
CNN-Bidir-LSTM*	-	C(32)-MAX-C(32)-MAX-C(32)-MAX-C(32)-MAX-B(L(128))-DL(128)-DL(6)	**90.57%**

Figure 4.1 shows that the configuration c0 achieved the highest validation accuracy which is 91.84%. Therefore, the configuration c0 is chosen to run the UCI's test dataset to obtain the best result of the model. The model achieves a 90.57% accuracy using configuration c0. The result is compared with other deep learning models running the same dataset and shown in **Table 4.6**. It can be seen that the proposed model outperformed other classification using the same environment with the UCI dataset, around 1% higher than the second-highest model.

This can be attributed to the combination of CNN and Bidir-LSTM, which use four layers of CNN with 32 feature maps, Bidir-LSTM layer with 128 hidden nodes and two Dense layers. The dropout rate is set to 0.1, meaning only 10% of the nodes are closed during training. Most proposed models have either a default dropout rate of 50% or 80%. Although dropout seeks to prevent overfitting, too much dropout seems to lead to underfitting (low variance but high bias) in this study. It also uses a 0.001 learning rate which allows a relatively fast for the model to converge to local minima or the classification accuracy.

It also has a 0.0001 weight decay or L2 regularization which regularizes the model in each epoch. Most of the proposed models do not set weight decay, which does not add an extra term into the cost function that penalizes large weights. Large weight can cause the weight vector to get stuck in a local minimum easily since gradient descent only makes small changes to the direction of optimization.

It can be seen more Bidir-LSTM layers do not necessarily translate to better performance in UCI datasets. For example. both models proposed by (Yu and Qin, 2018) and (Hernández et al., 2019) have three layers of Bidir-LSTM, which theoretically should translate to better performance. However, this is not the case in this study.

More CNN layers lead to a better result as compare to less CNN layers, which can be seen in (Lee, S. M., Cho, H., & Yoon, 2017) and (Ronao & Cho, 2016) case. Although they have more feature maps per layer, they did not have more than three CNN layer like our proposed model. Deeper abstraction and representation of data can be obtained with a deeper layer.

Overall, the proposed model is significantly better in UCI dataset as it has the optimum setting in hyperparameter. It uses less computing resources and take lesser time to obtain result, around 10 to 15 minutes. This may not be the case with the model proposed by (Hernández et al., 2019) which took more than four hours to obtain the final result.

4.3.2 WISDM dataset Result and Discussion

The proposed model of CNN-Bidir-LSTM is run using the configuration setting in **Table 4.4** for the WISDM dataset.

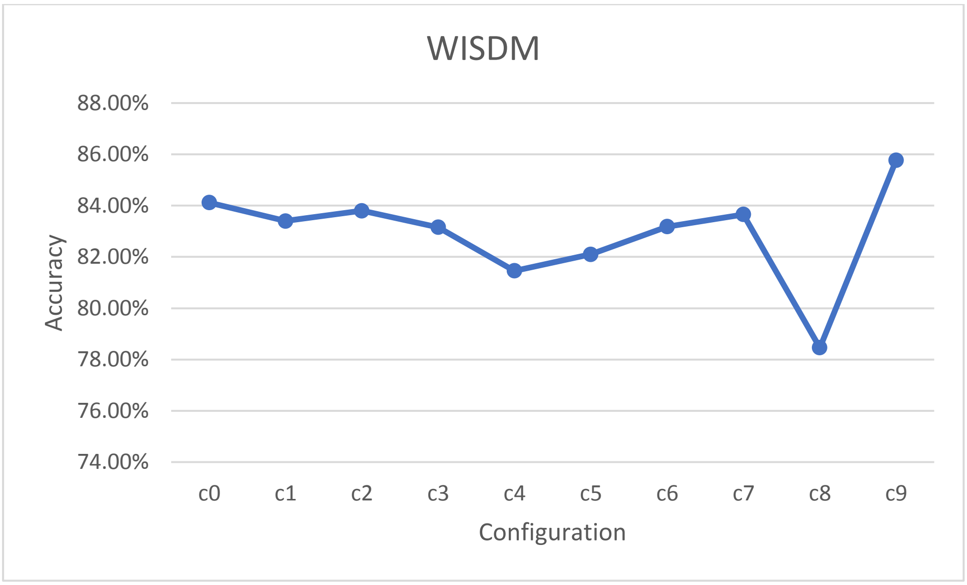

Figure 4.2: Configuration Result using WISDM dataset

Table 4.7: Result Comparison using WISDM dataset

Model	**Authors**	**Architecture**	**Accuracy**
CNN	(Ronao & Cho, 2016)	C(96)-MAX-C(96)-MAX-C(96)-MAX-DL(1000)-DL(6)	74.79%
CNN	(Lee, S. M., Cho, H., & Yoon, 2017)	C(128)-MAX-C(128)-MAX-DL(384)-DL(6)	85.98%
LSTM	(Chen, Zhong, Zhang, Sun, & Zhao, 2016)	L(100)-L(100)-DL(80)-DL(6)	79.62%
Bidir-LSTM	(Yu & Qin, 2018)	B(L(28))-B(L(28)-B(L(28))-DL(80)-DL(6)	72.98%
Bidir-LSTM	(Hernández, Suárez, Villamizar, & Altuve, 2019)	B(L(175))-B(L(175)-B(L(175))-DL(80)-DL(6)	82.09%
CNN-Bidir-LSTM*	-	C(220)-MAX-C(220)-MAX-C(220)-MAX-C(220)-MAX-B(L(80))-DL(80)-DL(6)	**87.06%**

Figure 4.2 shows that the configuration c9 achieved the highest validation accuracy which is 85.77%. Therefore, the configuration c9 is chosen to run the WISDM test dataset to obtain the best result of the model. The model achieves an 87.06% accuracy using configuration c9. The result is compared with other deep learning models running the same dataset and shown in **Table 4.7**. It can be seen that the proposed model outperformed other classification using the same environment with the WISDM dataset, around 1% higher than the second-highest model, similar to the UCI case.

This can also be attributed to the combination of CNN and Bidir-LSTM, which use four layers of CNN with 220 feature maps, Bidir-LSTM layer with 80 hidden nodes and two Dense layers. The dropout rate is set to 0.1, meaning only 10% of the nodes are closed during training. Most proposed models have either a default dropout rate of 50% or 80%. Although dropout seeks to prevent overfitting, too much dropout seems to lead to underfitting (low variance but high bias) in this study. It also uses a 0.001 learning rate which allows a relatively fast for the model to converge to local minima or the classification accuracy.

It also has a 0.0001 weight decay or L2 regularization which regularizes the model in each epoch. Most of the proposed models do not set weight decay, which does not add an extra term into the cost function that penalizes large weights. Large weight can cause the weight vector to get stuck in a local minimum easily since gradient descent only makes small changes to the direction of optimization.

Similar to the UCI case, it can be seen more Bidir-LSTM layers do not necessarily translate to better performance in WISDM datasets. For example. both models proposed by (Yu and Qin, 2018) and (Hernández et al., 2019) have three layers of Bidir-LSTM, which theoretically should translate to better performance. However, this is not the case in this study.

More CNN layers do lead to a better result as compare to less CNN layers, which can be seen in (Lee, S. M., Cho, H., & Yoon, 2017) and (Ronao & Cho, 2016) case. Different from the UCI case, our proposed model has more feature maps per layer than the two models. However, the significant factor relies on the four CNN layers which lead to deeper abstraction of data.

Overall, the proposed model is significantly better in WISDM dataset as it has the optimum setting in hyperparameter. It uses less computing resources and take lesser time to obtain result, around 15 to 20 minutes. This may not be the case with the model proposed by (Hernández et al., 2019) which took more than four hours to obtain the final result, similar to the UCI case.

4.4 Additional experiment (WISDM)

The purpose of the experiment is to examine the effect of hyperparameter on the performance of the model. WISDM dataset is selected for this experiment. The default configuration setting is set up as shown in **Table 4.8**.

Table 4.8: Default Configuration Setting

Hyperparameter	**WISDM**
CNN kernel	20
CNN kernel size	3
CNN activation	relu
Maxpool size	2
Bidirlstm nodes	100
Dropout rate	0.5
1st Dense nodes	100
1st Dense activation	relu
2nd Dense nodes	6
2nd Dense activation	softmax
Epochs	50
Optimizer	Adam
Learning rate	0.001
Decay	0

The effect of different hyperparameters such as the number of CNN kernels, CNN kernel size, activation functions, dropout rates, learning rates, epochs, and optimizers is examined and evaluated in the following section. For each experiment, only the particular hyperparameter that is evaluated is changed.

4.4.1 Effect of Number of CNN kernel

The experiment is set up using a range of 5 to 120 number of kernels. The model is set up using the configuration setting shown in **Table 4.8**, only the number of CNN kernel is changed. This is to see the effect of the number of kernels have on the overall performance of the model.

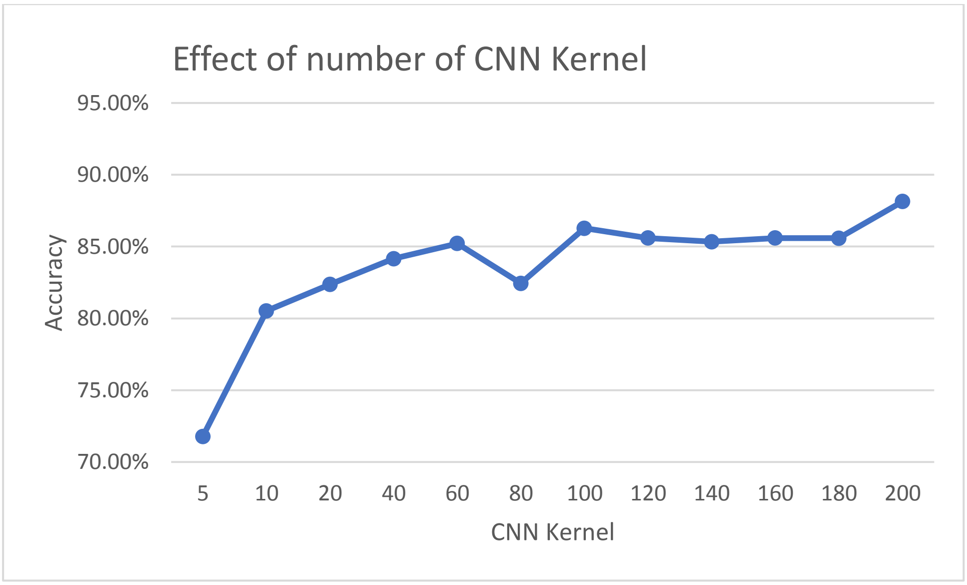

Figure 4.3: Effect of number of CNN kernel

Figure 4.3 shows that increasing the number of feature maps increases the validation accuracy of the model for WISTM models. The best performance is the number of CNN kernels = 200 (88.14%). The performance might increase when the number of feature maps is greater than 200. The more the feature maps or CNN kernels, the more abstract representation can be produced.

4.4.2 Effect of CNN Kernel Size

The experiment is set up using CNN kernel size with the range from 1 to 5. The model is set up using the configuration setting shown in **Table 4.8**, only the CNN kernel size is changed for each experiment. This is to see the effect of CNN kernel size have on the overall performance of the model.

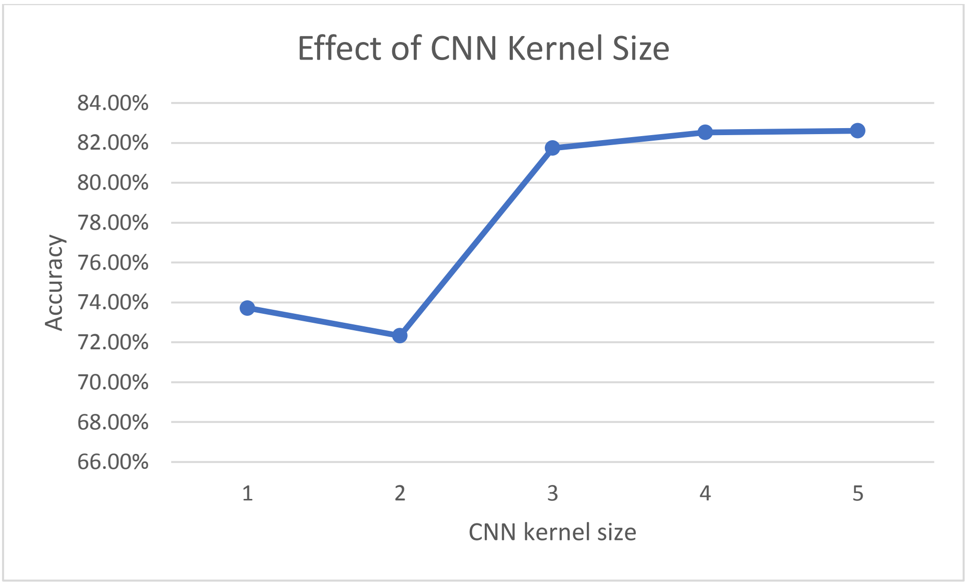

Figure 4.4: Effect of CNN Kernel Size

Figure 4.4 shows that the best performance for the WISDM dataset is the kernel size 5 which achieves an accuracy of 82.61%. The performance might increase when the kernel size is greater than 5. This experiment implied we can exploit a certain range of temporal local dependency is needed to achieve a better result.

4.4.3 Effect of Activation functions

There are four activations to be tested in the WISDM dataset. They are relu, sigmoid, tanh, and softmax which can be imported from Keras Tensorflow library. It is noted that the last dense layer will remain to have the softmax classifier while the other remaining activation will be altered. This is to see the effect of activation functions have on the overall performance of the model.

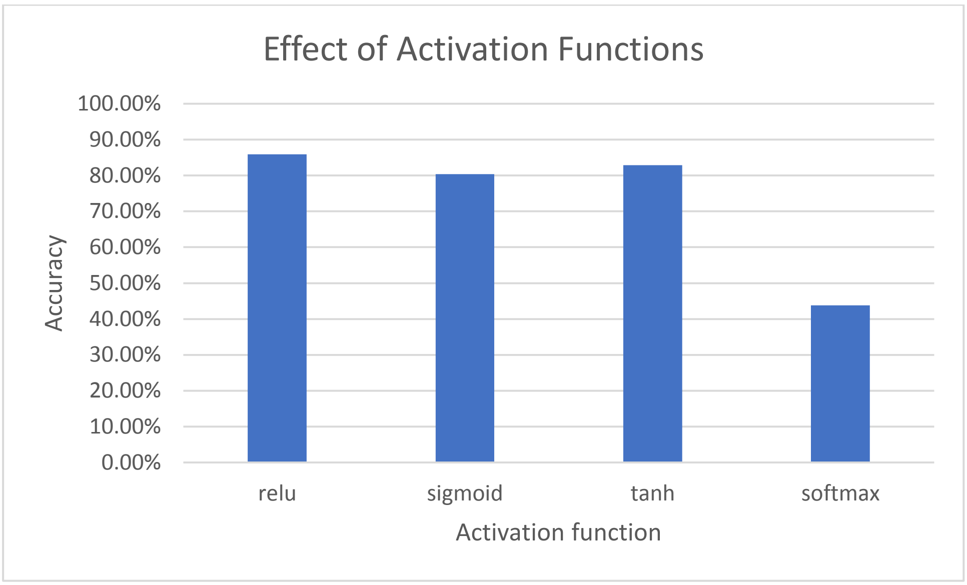

Figure 4.5: Effect of Activation Functions

Figure 4.5 shows that the best performance is relu activation for the WISDM dataset which achieves a performance of 85.91%. The second-best performance is tanh, followed by sigmoid and lastly, softmax. ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.

4.4.4 Effect of Dropout rates

The experiment is set up using a range of 0 to 0.5 dropout rates. The model is set up using the configuration setting shown in **Table 4.8**, only the dropout rate is changed for each experiment. This is to see the effect of dropout rates have on the overall performance of the model.

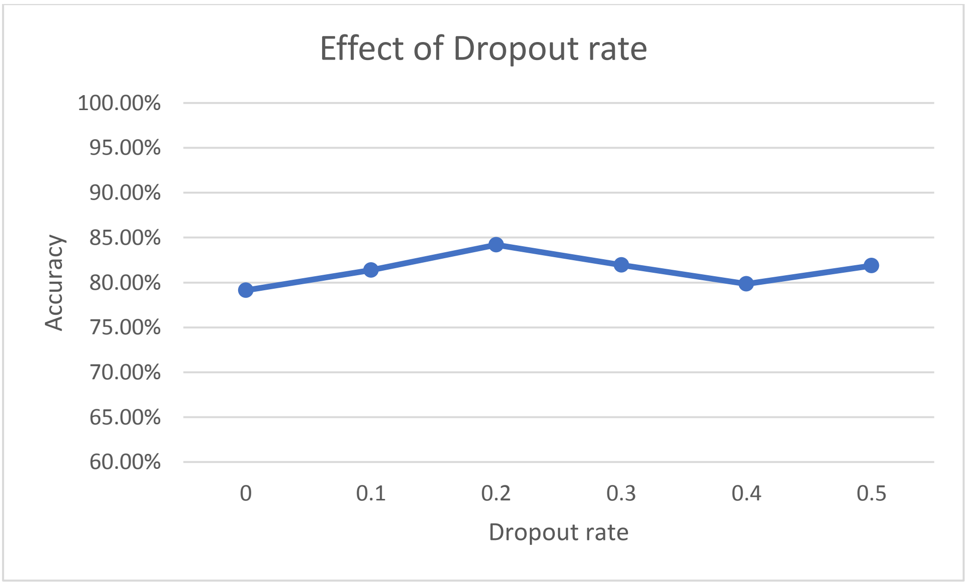

Figure 4.6: Effect of Dropout rates

Figure 4.6 shows that the best dropout rate is 0.2 which achieves an accuracy of 84.21% during validation**.** Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). However the changes are less than 5%, therefore the dropout does not have much influence on the model performance.

4.4.5 Effect of Epoch

The experiment is set up using a range of 25 to 150 epochs. The model is set up using the configuration setting shown in **Table 4.8**, only the epoch is changed for each experiment. This is to see the effect of epoch have on the overall performance of the model.

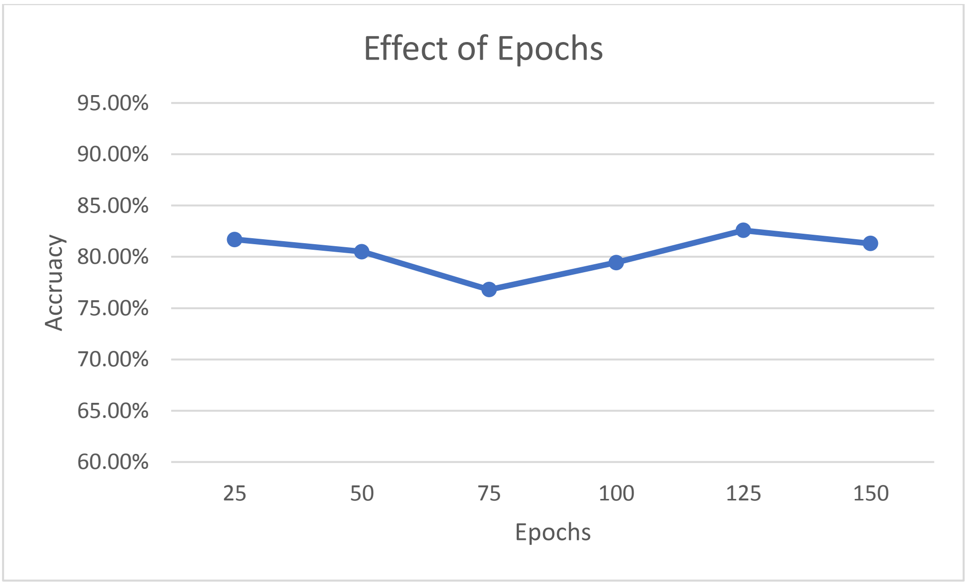

Figure 4.7: Effect of Epochs

Figure 4.7 shows that the best epoch is 125 which achieves an accuracy of 82.58% during validation. The performance does not increase when the epoch is greater than 125. More epochs may lead to overfitting. Thus, the changes in performance are less than 5%.

4.4.6 Effect of Optimizer

The WISDM dataset is tested using both Adam and Stochastic Gradient Descent (SGD) optimizer. The model is set up using the configuration setting shown in **Table 4.8**, only the optimizer for the model is changed for each experiment. This is to see the effect of different optimizers has on the overall performance.

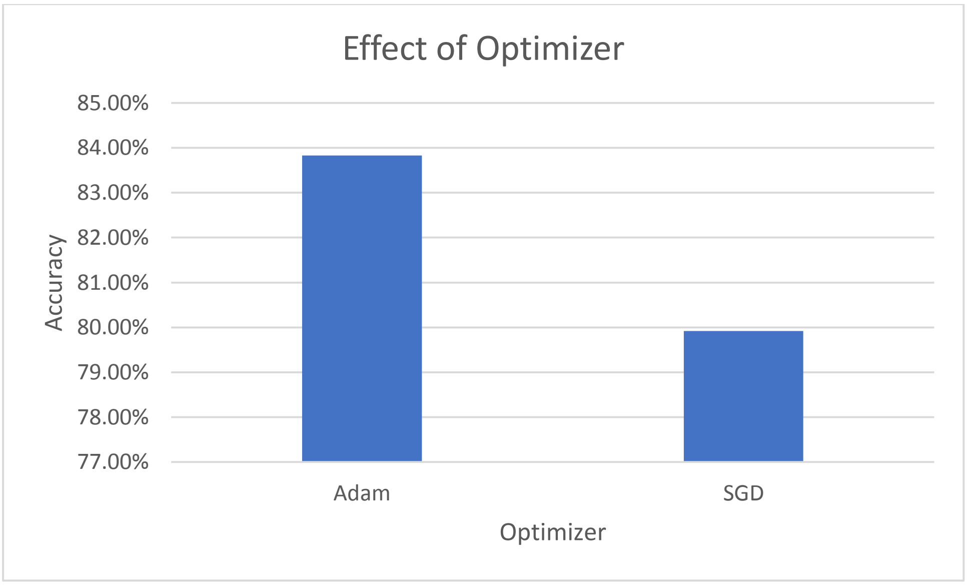

Figure 4.8: Effect of Optimizer

Figure 4.8 shows that Adam has the highest performance. This can be attributed to the Stochastic gradient descent (SGD) maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. Adam combining the advantages of two other extensions of SGD which are the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). It also makes use of the average of the second moments of the gradients which is the uncentered variance. (Kingma & Ba, 2015)

4.4.7 Effect of Learning rates

The experiment is set up using a range of 0 to 0.001 learning rates. The model is set up using the configuration setting shown in **Table 4.8**, only the learning rate is changed for each experiment. This is to see the effect of learning rates have on the overall performance of the model.

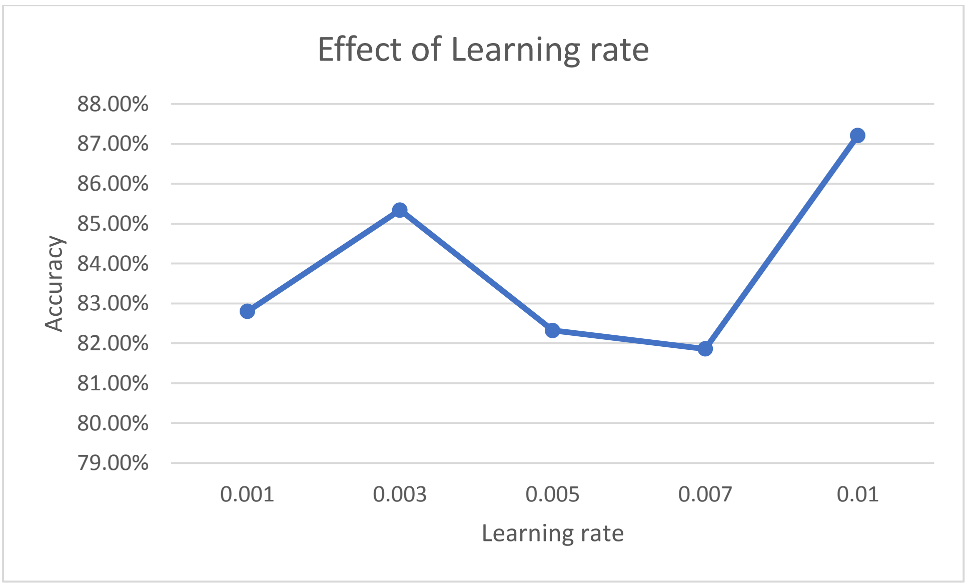

Figure 4.9: Effect of Learning rates

Figure 4.9 shows that the best learning rate is 0.01 which achieves an accuracy of 87.21% during validation. The learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect to the loss gradient. It affects how quickly our model can converge to local minima (aka arrive at the best accuracy). Lower learning rate means the speed of getting the result is much faster than a high learning rate, which will cause the model to miss out on some essential data.

4.5 Chapter Summary

In the first experiment (Section 4.3), it is found that the proposed model, performed better than the existing classification model using two different public datasets. A combination of CNN and RNN has given a slight edge in classifying the six human activities, as it maximizes the temporal analysis of RNN and the deep-feature extraction of CNN. However, it is worth noting that different architectures have different advantages over one another. It is also discovered in Section 4.4 that there is indeed a significant influence of hyperparameter settings on the model performance. The most notable case is the number of CNN kernels, where the increase in CNN feature improves the model performance significantly. Other notable cases are CNN kernel sizes, learning rate, activation function, and optimizers. On the other hand, epochs have some influence on the performance. However, due to the hardware limitation, no significant result is recorded.

CHAPTER 5

CONCLUSION

In this paper, an overview of different approaches toward human activity recognition is provided, such as wearable-based methods, vision-based methods, and smartphone-based methods. However, due to the inconveniency of wearable-based and vision-based methods, the smartphone-based method became the main theme of this study. This research proposed a method for human activity recognition using a fusion of convolutional neural networks and bidirectional LSTM architecture to the time series data collected from two public datasets. It worked directly on the raw sensor data with minimal pre-processing. A comparison of this proposed model with other state-of-the-art classification methods shows that this network outperformed other previous results, achieving an accuracy of 90.75% with UCI dataset and an accuracy of 87.06% with WISDM dataset. An additional study is also conducted to see the effect of the hyperparameter on the performance of the model. In future work, this study will focus on various time and frequency features as well as various classifiers.

REFERENCE/BIBLIOGRAPHY

Abidine, B. M., Fergani, L., Fergani, B., & Oussalah, M. (2018). The joint use of sequence features combination and modified weighted SVM for improving daily activity recognition. *Pattern Analysis and Applications*. https://doi.org/10.1007/s10044-016-0570-y

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. *ESANN 2013 Proceedings, 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Asuroglu, T., Acici, K., Erdas, C. B., & Ogul, H. (2017). Texture of Activities: Exploiting Local Binary Patterns for Accelerometer Data Analysis. *Proceedings - 12th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2016*, (Figure 1), 135–138. https://doi.org/10.1109/SITIS.2016.29

Bao, L., & Intille, S. S. (2004). *Activity Recognition from User-Annotated Acceleration Data*. https://doi.org/10.1007/978-3-540-24646-6_1

Bayat, A., Pomplun, M., & Tran, D. A. (2014). A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, *34*, 450–457. https://doi.org/10.1016/j.procs.2014.07.009

Bharti, P. (2017). Context-based Human Activity Recognition Using Multimodal Wearable Sensors. *Graduate Theses and Dissertations*, (November). Retrieved from https://scholarcommons.usf.edu/etd/7000%0Ahttps://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=8197&context=etd

Brezmes, T., Gorricho, J. L., & Cotrina, J. (2009). Activity recognition from accelerometer data on a mobile phone. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes* *in Bioinformatics)*, *5518 LNCS*(PART 2), 796–799. https://doi.org/10.1007/978-3-642-02481-8_120

Chen, Y., Zhong, K., Zhang, J., Sun, Q., & Zhao, X. (2016). *LSTM Networks for Mobile Human Activity Recognition*. (Icaita), 50–53. https://doi.org/10.2991/icaita-16.2016.13

Chong, L. (2017). *WEARABLE COMPUTING : ACCELEROMETER-BASED HUMAN ACTIVITY CLASSIFICATION USING DECISION TREE by Chong Li A thesis submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Computer Science Approved : Xiaojun Qi , Ph . D . K.*

Cortes, C., & Vapnik, V. (1995). Supprot-Vector Networks. *Machine Learning*, *297*(20), 273–297. https://doi.org/10.1111/j.1747-0285.2009.00840.x

Friday Nweke, H., Ying Wah, T., & Alo, U. (2018). *Deep Learning Algorithms for Human Activity Recognition using Mobile and Wearable Sensor Networks: State of the Art and Research Challenges Mobile Cloud Computing View project Novel Deep Learning Architecture for Physical Activities assessment, mental Res*. *105*, 233–261. https://doi.org/10.1016/j.eswa.2018.03.056

Gao, L., Bourke, A. K., & Nelson, J. (2014). Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical Engineering and Physics*, *36*(6), 779–785. https://doi.org/10.1016/j.medengphy.2014.02.012

Gorelick, L., Blank, M., Shechtman, E., Irani, M., & Basri, R. (2007). Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2007.70711

Guan, Y., & Plötz, T. (2017). Ensembles of Deep LSTM Learners for Activity Recognition using Wearables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. https://doi.org/10.1145/3090076

Ha, S., & Choi, S. (2016). Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. *Proceedings of the International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN.2016.7727224

Hammerla, N. Y., Kirkham, R., Andras, P., & Ploetz, T. (2013). *On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution*. 65. https://doi.org/10.1145/2493988.2494353

Herbert Robbins and Sutton Monro. (1951). A Stochastic Approximation Method on JSTOR. *The Annals of Mathematical Statistics*.

Hernández, F., Suárez, L. F., Villamizar, J., & Altuve, M. (2019). Human Activity Recognition on Smartphones Using a Bidirectional LSTM Network. *2019 22nd Symposium on Image, Signal Processing and Artificial Vision, STSIVA 2019 - Conference Proceedings*. https://doi.org/10.1109/STSIVA.2019.8730249

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities (associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices). *Biophysics*.

Ignatov, A. D., & Strijov, V. V. (2016). Human activity recognition using quasiperiodic time series collected from a single tri-axial accelerometer. *Multimedia Tools and Applications*, *75*(12), 7257–7270. https://doi.org/10.1007/s11042-015-2643-0

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D Convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 221–231. https://doi.org/10.1109/TPAMI.2012.59

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic gradient descent. *ICLR: International Conference on Learning Representations*.

Kwapisz, J. R., Weiss, G. M., & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*. https://doi.org/10.1145/1964897.1964918

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*. https://doi.org/10.1162/neco.1989.1.4.541

Lee, S. M., Cho, H., & Yoon, S. M. (2017). Human Activity Recognition From Accelerometer Data Using Convolutional Neural Network. *IEEE International Conference on Big Data and Smart Computing (BigComp).*, *62*, 131–134. https://doi.org/10.1016/j.asoc.2017.09.027

Lee, K., & Kwan, M. P. (2018). Physical activity classification in free-living conditions using smartphone accelerometer data and exploration of predicted results. *Computers, Environment and Urban Systems*, *67*(September 2017), 124–131. https://doi.org/10.1016/j.compenvurbsys.2017.09.012

Li, C. (2017). *DigitalCommons@USU Wearable Computing: Accelerometer-Based Human Activity Classification Using Decision Tree*. Retrieved from https://digitalcommons.usu.edu/etd/5799

Lockhart, J. W. (2014). *The Benefits of Personalized Data Mining Approaches to Human Activity Recognition with Smartphone Sensor Data*. 46. Retrieved from http://search.proquest.com/docview/1627154494

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2008). *Supervised Dictionary Learning*. Retrieved from http://arxiv.org/abs/0809.3083

Nishkam Ravi Preetham Mysore, Michael L. Littman, N. D. (2005). Activity recognition from accelerometer data. *The Seventeenth Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence (IAAI-05)*. https://doi.org/10.1007/978-3-642-02481-8_120

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). Understanding the exploding gradient problem. *Proceedings of The 30th International Conference on*

Machine Learning. https://doi.org/10.1109/72.279181

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, *28*(6), 976–990. https://doi.org/10.1016/j.imavis.2009.11.014

Ronao, C. A., & Cho, S. B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, *59*, 235–244. https://doi.org/10.1016/j.eswa.2016.04.032

Schüldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: A local SVM approach. *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2004.1334462

Shi, X., Li, Y., Zhou, F., & Liu, L. (2018). Human Activity Recognition Based on Deep Learning Method. *2018 International Conference on Radar, RADAR 2018*. https://doi.org/10.1109/RADAR.2018.8557335

Smith, L. (2002). A tutorial on Principal Components Analysis. *Communications in Statistics - Theory and Methods*, *17*(9), 3157–3175. https://doi.org/10.1080/03610928808829796

Soomro, K., & Zamir, A. R. (2014). Action recognition in realistic sports videos. *Advances in Computer Vision and Pattern Recognition*. https://doi.org/10.1007/978-3-319-09396-3_9

Sprager, S., & Zazula, D. (2000). Gait Identification Using Cumulants of Accelerometer Data. *Sensors, Signals, Visualization, Imaging and Materials*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Wang, J., Chen, Y., Hao, S., Peng, X., & Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*.

https://doi.org/10.1016/j.patrec.2018.02.010

Weinland, D., Ronfard, R., & Boyer, E. (2006). Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*. https://doi.org/10.1016/j.cviu.2006.07.013

Yu, S., & Qin, L. (2018). Human activity recognition with smartphone inertial sensors using bidir-LSTM networks. *Proceedings - 2018 3rd International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2018*, 219–224. https://doi.org/10.1109/ICMCCE.2018.00052

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., & Zhang, J. (2014). *Convolutional Neural Networks for human activity recognition using mobile sensors Article*. 381–388. https://doi.org/10.4108/icst.mobicase.2014.257786

Zhang, S., Rowlands, A. V., Murray, P., & Hurst, T. L. (2012). Physical activity classification using the GENEA wrist-worn accelerometer. *Medicine and Science in Sports and Exercise*, *44*(4), 742–748. https://doi.org/10.1249/MSS.0b013e31823bf95c

Zhao, Y., Yang, R., Chevalier, G., Xu, X., & Zhang, Z. (2018). Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors. *Mathematical Problems in Engineering*, *2018*. https://doi.org/10.1155/2018/7316954

APPENDICES

Appendix A: FYP Meeting Logs

MULTIMEDIA UNIVERSITY ®

Faculty of Information Science and Technology (FIST)
Final Year Project Meeting Log

MEETING DATE: 12/7/2019	**MEETING NO.: 1**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] Update on progress, presented Recurrent Neural Network (RNN)
2. WORK TO BE DONE Read up more about LSTM, install Jupyter Notebook, Keras and Tensor Flow, explore GitHub repository which run WISDM activity recognition using RNN
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

<u>**NOTES**</u>:

1. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
2. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
3. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
4. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 16/7/2019	**MEETING NO.: 2**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] Read up more about LSTM, install Jupyter Notebook, Keras and Tensor Flow, explore GitHub repository which run WISDM activity recognition using RNN
2. WORK TO BE DONE - Understand the code and rerun three GitHub projects, understand how they input data into RNN, what is the classifier used, - Explore the possibility of Fusion between CNN and RNN to create a model for a better prediction accuracy
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp

Co-Supervisor's Signature & Stamp (if any)

Student's Signature

NOTES:

5. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
6. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
7. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
8. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

Faculty of Information Science and Technology (FIST)
Final Year Project Meeting Log

MEETING DATE: 18/7/2019	**MEETING NO.: 3**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] Rerun three GitHub projects, explore the possibility of Fusion between CNN and RNN to create a model for a better prediction accuracy,
2. WORK TO BE DONE Prepare Chapter 2 Literature Review and Chapter 3
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp

Co-Supervisor's Signature & Stamp (if any)

Student's Signature

NOTES:

9. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
10. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
11. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
12. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 26/7/2019	**MEETING NO.: 4**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] Chapter 2 Literature Review and Chapter 3
2. WORK TO BE DONE Discuss and analyze Github code input method and LSTM model
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp

Co-Supervisor's Signature & Stamp (if any)

Student's Signature

NOTES:

13. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
14. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
15. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
16. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 31/7/2019	**MEETING NO.: 5**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - Supervisor reviewed Chapter 2 Literature Review done, made recommendation for improvement - Tested three sample model which is RNN, CNN+LSTM and convLSTM (which LSTM is part of CNN), save finding.
2. WORK TO BE DONE - Complete Chapter 3 with more detail information about LSTM and Bidirectional LSTM - Look into setting own model using a combination of max-pooling, covnet and bidirectional LSTM
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp

Co-Supervisor's Signature & Stamp (if any)

Student's Signature

NOTES:

17. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
18. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
19. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
20. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)

**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 7/8/2019	**MEETING NO.: 6**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] -Finish testing WISDM dataset and UCI dataset using LSTM, CNN-LSTM, CovLSTM, and Bidirectional LSTM
2. WORK TO BE DONE - Tabulate result of different methods - Create two code for testing WISDM and UCI dataset using proposed model, which is fusion between CNN and Bidirectional LSTM
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp

Co-Supervisor's Signature & Stamp (if any)

Student's Signature

NOTES:

21. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
22. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
23. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
24. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)

**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

MULTIMEDIA UNIVERSITY

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 26/8/2019	**MEETING NO.: 7**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.]

-Finish tabulating WISDM and UCI dataset comparison, finish tabulating the different hyperparameter configuration of different models proposed by other researchers (CNN, LSTM, BidirLSTM and Deep Residue LSTM)

-Set up proposed model CNN-Bidirlstm for both WISDM and UCI

2. WORK TO BE DONE

- Run tests with different architecture of the proposed model
- Run tests with different parameter configuration and record result

3. PROBLEMS ENCOUNTERED

4. COMMENTS

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

25. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.

26. For FYP Phase 1, total six log sheets are to be submitted (every other week*).

27. For FYP Phase 2, total six log sheets are to be submitted (every other week**).

28. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)

**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 12/9/2019	**MEETING NO.: 8**
PROJECT ID: T681546	
PROJECT TITLE : HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION : Trimester 1 2019/2020	**SUPERVISOR : PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR : OOI SHIH YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - Run tests with different architecture of the proposed model - Run tests with different parameter configuration and record result
2. WORK TO BE DONE Complete Chapter 4 and Chapter 5, edit and format report
3. PROBLEMS ENCOUNTERED
4. COMMENTS

Supervisor's Signature & Stamp	Co-Supervisor's Signature & Stamp (if any)	Student's Signature

NOTES:

29. Items 1 – 3 are to be completed by the students before coming for the meeting. Item 4 is to be completed by the supervisor.
30. For FYP Phase 1, total six log sheets are to be submitted (every other week*).
31. For FYP Phase 2, total six log sheets are to be submitted (every other week**).
32. Log sheets are compulsory assessment criteria for FYP. Student who fails to meet the requirements of log sheets will not be allowed to submit FYP report.

*: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the first trimester (week 11: report submission, weeks 13 & 14: presentation)
**: week 1, 3, 5, 7, 9, 11 or 2, 4, 6, 8, 10 of the second trimester (week 11: report submission, weeks 13 & 14: presentation)

Appendix B: Checklist for FYP Interim Submission

MULTIMEDIA UNIVERSITY

Faculty of Information Science and Technology (FIST)

Checklist for Interim Report Submission

(To be filled in by Student)

STUDENT'S DETAILS

Project Code	**FIST**
Name	
ID No	
Title of Thesis	
Supervisor Name	

REPORT ARRANGEMENT	√	**Comments** **(if any differences)**
1. Cover of The Interim Report		
2. Title Page of the Interim Report		
3 Copyright page of I Interim Report		
4. Declaration Page of Interim report		
5. Acknowledgement		
6. Table of Contents		
7. Abstract		
8. List of Tables		
9. List of Figures		
10. List of Symbols		
11. List of Appendices		
12. Chapter 1: Introduction – objectives, scope		
13. Chapter 2: Literature Review		
14. Chapter 3: Title:		
15. Chapter 4: Title:		
16. Chapter 5: Title:		
17. References – APA style		
18. Appendices		
19. CD/ DVD and envelope as shown in Appendix K		
20. Attachment : FYP Meeting Logs (all) 1 set		

FORMAT OF REPORT	√	**Comments**
1. Page Numbering		
2. Font and Type Face		
3. Font Cover		
4. Tables and Figures		
5. Comb Bind		
6. Colour of the Front Cover		
7. Number of words > 5000 (Main content only)		

Checked by

____

Student's Signature & Date