HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA

CHEW YONG SHAN

SESSION 2019/2020

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

MARCH 2020

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

MARCH 2020

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: 16 March 2020

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 2. 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 the RNN model to gain a deeper insight into HAR, coupled with the improvement of its overall 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, as well as extracting deep features from the proposed model to test with different classifiers to improve HAR performance.

TABLE OF CONTENTS

DECLARATION ..... **III**

ACKNOWLEDGEMENT ..... **IV**

ABSTRACT ..... **V**

TABLE OF CONTENTS ..... **VI**

LIST OF TABLES ..... **IX**

LIST OF FIGURES ..... **X**

LIST OF ABBREVIATIONS/SYMBOLS ..... **XI**

LIST OF APPENDICES ..... **XII**

CHAPTER 1 INTRODUCTION **1**

1.1 Overview 1

1.2 Problem Statement 2

1.3 Research Objective 3

1.4 Research Scope 3

1.5 Report Organization 4

CHAPTER 2 LITERATURE REVIEW **5**

2.1 Human Activity Recognition (HAR) 5

2.1.1 Vision-based HAR 6

2.1.2 Wearable Sensor HAR 7

2.1.3 Smartphone-based HAR 8

2.2 Features used in HAR 10

2.2.1 Hand-crafted features 10

2.2.2 Learning Feature 12

2.3 Deep Learning in HAR 12

2.3.1 Convolutional Neural Network (CNN) 13

2.3.2 Recurrent Neural Network (RNN) 15

2.3.2.1 Long Short-Term Memory (LSTM) 16

2.3.2.2 Bi-directional Long Short-Term Memory (Bidir-LSTM) 17

2.3.2.3 Deep Residue Bi-directional Long Short-Term Memory 18

2.3.3 Comparison Study of Existing Deep Learning Models 18
2.4 Public Dataset Comparison (WISDM and UCI) 19
2.5 Chapter Summary 22

CHAPTER 3 THE CONCEPT AND COMPONENTS USED IN DEEP LEARNING MODEL FOR HUMAN ACTIVITY RECOGNITION 23

3.1 Convolutional Neural Network (CNN) 23
3.1.1 Architecture and Formulation of CNN 24
3.1.2 Max Pooling 24
3.1.3 Dropout layer 25
3.2 LSTM 25
3.2.1 Architecture and Formulation of LSTM 26
3.3 Bi-dir LSTM 28
3.3.1 Architecture and Formulation of Bi-dir LSTM 28
3.4 Weight decay/Regularization 29
3.5 Activation function 29
3.5.1 Rectified Linear Unit (Relu) 29
3.5.2 Sigmoid function 30
3.5.3 Tanh function 30
3.5.4 Softmax function 30
3.6 Optimizer 31
3.6.1 Stochastic Gradient Descent (SGD) 31
3.6.2 Root Mean Square Propagation (RMSProp) 31
3.6.3 Adaptive Moment Estimation (Adam) 31
3.7 Classifiers 32
3.7.1 Logistics Regression 32
3.7.2 Support Vector Machine (SVM) 32
3.7.3 Random Forest 32
3.7.4 Naïve Bayes 33
3.7.5 Multilayer Perceptron 33
3.7.6 K-Nearest Neighbours (Scaled Manhattan) 33
3.7.7 K-Nearest Neighbours (Euclidean distance) 34
3.7.8 Bayesian Network 34
3.7.9 J48 34

3.8 Chapter Summary 35

CHAPTER 4 METHOD 1: EVALUATION OF C4M4BL AND THE EFFECT OF HYPERPARAMETER CONFIGURATION 36

4.1 The Proposed Neural Network Model (C4M4BL) 36
4.2 Architecture of CNN4-MAX4-BidirLSTM model 37
4.3 Environment setup and Hardware/Software Information 39
4.4 Training and Test Sets Information 40
4.5 UCI and WISDM Datasets Experiment 41
4.5.1 UCI dataset Result and Discussion 44
4.5.2 WISDM dataset Result and Discussion 49
4.6 Effect of Hyperparameter Configuration Experiment 54
4.6.1 Experiment setup 54
4.6.2 Hyperparameter Variable Results and Discussions 55
4.6.2.1 Effect of Number of CNN kernel 55
4.6.2.2 Effect of CNN Kernel Size 57
4.6.2.3 Effect of Activation functions 58
4.6.2.4 Effect of Dropout rates 60
4.6.2.5 Effect of Epoch 61
4.6.2.6 Effect of Optimizer 63
4.6.2.7 Effect of Learning rates 64
4.7 Chapter Summary 65

CHAPTER 5 METHOD 2: C3M1BL AS DEEP FEATURE EXTRACTOR WITH WEKA CLASSIFIERS 66

5.1 Architecture of the Enhanced Proposed Model (C3M1BL) 66
5.2 C3M1BL as deep feature extractor 70
5.3 WEKA Classifiers Performance using Deep Features 72
5.4 Chapter Summary 74

CHAPTER 6 CONCLUSION 75

REFERENCE/BIBLIOGRAPHY 77

APPENDICES 84

LIST OF TABLES

Table 2.1: Comparison Study between Existing Deep Learning Models..... 19
Table 2.2: Comparison Study between UCI and WISDM dataset..... 21
Table 4.1: Hardware and Software Information 39
Table 4.2: Training and Test Sets Information..... 40
Table 4.3: UCI Hyperparameter Configuration Set..... 41
Table 4.4: WISDM Hyperparameter Configuration Set..... 42
Table 4.5: Normalized Architecture and Information of Existing Classification Methods..... 43
Table 4.6: Validation Result using UCI training dataset 45
Table 4.7: Configuration Matrix Using UCI Test datasets with c0 Configuration 47
Table 4.8: Result Comparison using UCI test dataset..... 47
Table 4.9: Validation Result using the WISDM training dataset..... 50
Table 4.10: Configuration Matrix using the WISDM test dataset 52
Table 4.11: Result Comparison using the WISDM dataset 52
Table 4.12: Default Configuration Setting..... 54
Table 4.13: Effect of number of CNN kernel..... 56
Table 4.14: Effect of CNN Kernel Size..... 58
Table 4.15: Effect of Activation Functions 59
Table 4.16: Effect of Dropout rates 60
Table 4.17: Effect of Epochs..... 62
Table 4.18: Effect of Optimizer..... 63
Table 4.19: Effect of Learning rates..... 65
Table 5.1: Comparison between C4M4BL and C3M1BL 68
Table 5.2: Confusion Matrix of UCI dataset using C3M1BL 69
Table 5.3: Confusion Matrix of WISDM dataset using C3M1BL..... 69
Table 5.4: Classifiers Result for both UCI and WISDM datasets 72
Table 5.5: Result Comparison between Different Classifiers 73

LIST OF FIGURES

Figure 2.1: Convolutional Layers and Fully Connected Layers 13
Figure 2.2: Recurrent Neural Network 15
Figure 2.3: Long Short-Term Memory 16
Figure 2.4: Bi-directional Long Short-Term Memory 17
Figure 2.5: Deep Residue Bidir-LSTM 18
Figure 4.1: CNN4-MAX4-BidirLSTM (C4M4BL) model 37
Figure 4.2: Validation Result using UCI training dataset 44
Figure 4.3: Learning curve of UCI Training data against Validation data using c0 Configuration 46
Figure 4.4: Validation Result using the WISDM training dataset 49
Figure 4.5: Learning curve of WISDM Training data against validation data using c9 Configuration 51
Figure 4.6: Effect of number of CNN kernel 56
Figure 4.7: Effect of CNN Kernel Size 57
Figure 4.8: Effect of Activation Functions 59
Figure 4.9: Effect of Dropout rates 60
Figure 4.10: Effect of Epochs 61
Figure 4.11: Effect of Optimizer 63
Figure 4.12: Effect of Learning rates 64
Figure 5.1: CNN3-MAX1-BidirLSTM (C3M1BL) model 67
Figure 5.2: Deep Feature Extraction And Transfer Learning Process 71

LIST OF ABBREVIATIONS/SYMBOLS

Adam	Adaptive Moment Estimation
API	Application Programming Interface
ARFF	Attribute-Relation File Format
Bidir-LSTM	Bidirectional Long Short-Term Memory
C3M1BL	CNN3-Max1-BidirLSTM
C4M4BL	CNN3-Max3-BidirLSTM
CNN	Convolutional Neural Network
CSV	Comma-separated Value
GRU	Gated Recurrent Unit
HAR	Human Activity Recognition
IoT	Internet of things
KNN	K-Nearest Neighbour
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
Relu	Rectified Linear Unit
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
UCI	University of California Irvine
WEKA	Waikato Environment for Knowledge Analysis
WISDM	Wireless Sensor Data Mining

σ	Weight decay coefficient
A	Activation
B	Bias
V	Input Vector
W	Weight

LIST OF APPENDICES

APPENDIX A: FYP MEETING LOGS..... 84

APPENDIX B: CHECKLIST FOR FYP FINAL REPORT SUBMISSION..... 90

APPENDIX C: CD/DVD AND ENVELOPE..... 92

CHAPTER 1

INTRODUCTION

1.1 Overview

The ever-changing technology and the introduction of the Internet of things (IoT) has accelerated the wide adoption of accelerometer data by sensor combining with cloud computing and big data to detect and determine a person's activity or commonly known as Human Action Recognition (HAR). 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, jogging, laying down, or going upstairs. 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 revolving 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).

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 explore the effect of hyperparameter configuration on the performance of the model.

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 high-end CPU that has multiple state-of-the-art GPU and incredible RAM and bandwidth accounting to more than 192GB/s. The hardware itself cost around RM20,000 and above. The hardware specifications limitation and such high numbers of epochs are impossible to run with a common processing unit.

Even if it is possible to run in a general CPU, it will take weeks, even months to get the result we intended as the processing units are not fast enough to train the model. This research has provided a cost-effective way to run the proposed model and prove that even conventional CPU can achieve high accuracy of above 90% in HAR. To validate the result, two datasets (UCI and WISDM datasets) are used.

1.3 Research Objective

The research objective is summarized as followed:

- To design an automated HAR model based on smartphone sensor data using deep learning approach and evaluate its accuracy using two public datasets
- 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 the following things:

- Study the different approaches in HAR
- Study the effect of time, frequency and deep features on accelerometer data to classify the types of human activities
- Experiment with Keras and Tensorflow to build and test the proposed model
- 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 s(UCI dataset and WISDM dataset) which are used to validate our results also explored 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 on 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. Different classifiers used in the second experiment are briefly explored 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. 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. A second 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 beginning of our third experiment whereby the proposed model in CHAPTER 4 will be enhanced to function as a deep feature extractor. The deep feature extracted from the trained model is later fed into WEKA which has different classifiers using WEKA. The results of the different classifiers are tabulated and further analysis is given. CHAPTER 6 marked the concluding remark of this research and future work is explored.

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 the WISDM dataset and UCI dataset in Section 0.

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 HAR (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 video surveillance, human and computer interaction. Poppe (2010) pointed out that the challenges faced in VHAR lies in the way people move, environment, and interpersonal differences, and the variance in each action. For instance, people can be different in the speed their walk is and how wide their arm motion is when they walk. The environment is also another major concern. The environment 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 video recording is taking place throughout the experiment. The whole processes take more time than it should be. 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 training any model and influence the prediction accuracy.

These challenges can be overcome by using some of the publicly available vision-based HAR 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 dataset. They also permit deeper insight into methods that allow researchers to be aware of the challenges that come with it.

However, it is can be argued that algorithm bias does exist in a specific dataset. This may lead to less applicable approach due to each approaches perform well in a specific dataset. 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 different parts of the users' body such as hip and ankle that supervise 20 types of activities. It was trained with 20 users using different machine learning. 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 differentiate many activities more efficiently.

In (Nishkam Ravi Preetham Mysore, Michael L. Littman, 2005), the subjects are required to wear an accelerometer near the pelvic region and performed activities such as walking, sit-ups, running, vacuuming and brushing teeth. In (Zhang et al., 2012), Gravity Estimator of Normal Everyday Activity (GENEA) is used to collect accelerometer data. They compare the prediction accuracy of GENEA worn on the wrist and waist.

Li (2017) focused on the application of wearable sensors that collect acceleration data which only requires the user to wear it on his or her wrist in HAR. Gao, Bourke, and Nelson (2014) used multiple sensors on different locations of the body in their HAR 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 lethal activities from non-lethal activities. The model created achieved fairly close accuracy due to the leverage of accelerometer, gyroscope, barometer and Bluetooth beacon for classification.

Although wearable sensor HAR has gained far more popularity than vision-based HAR, most researches fail to mention 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 specialized hardware such as gyroscope, barometer, and accelerometer are difficult to obtain. Besides, the application of the modern feature extraction algorithms and the sophisticated classifiers continue to surpassed the computational capability of the recent wearable platforms (Gao et al., 2014). Therefore, there is a need to have a more convenient way to perform HAR while not sacrificing its accuracy.

2.1.3 Smartphone-based HAR

To tackle the inconvenience problem in wearable sensor HAR and the privacy issues revolving in vision-based HAR, we can look into smartphone-based HAR. Smartphone-based HAR is 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 both the inconveniency problem of wearable sensors and the privacy issues 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; A. D. Ignatov & Strijov, 2016; Kwapisz et al., 2011; Lockhart, 2014).

Kwapisz et al. (2011) popularized the adoption of smartphone-based accelerometers HAR. Their work enabled a public platform of Android-based data collection called Wireless Sensor Data Mining (WISDM) to be developed, which

contained data of six activities performed by 29 users carrying Android smartphone in their pocket. In (Anguita et al., 2013), a similar approach is also implemented whereby a group of 30 volunteers carrying the smartphone on their waist to collect triaxial acceleration and angular velocity of the six activities performed. Their work enables the creation of the public UCI HAR dataset. A comparison of both of these datasets is summarized in Table 2.2. In (Bayat et al., 2014), only four users carrying a smartphone are used to collect a variety of activities, which is similar to the previous two studies. However, all of these studies required researchers to collect data personally from a limited group of volunteers. Therefore, the collection process must follow a very strict standard, so that the reliability of the data collected are not tampered or else the data will be rendered useless.

Utilizing the WISDM dataset, Lockhart (2014) is able to analyze the learning pattern for various model types which reveal that small amounts of personal data can surpass even the best model. Similarly, with the help of the WISDM dataset, Ignatov and Strijov (2016) are able to train and evaluate the proposed algorithm.

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 0.

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 the features extracted by utilizing various algorithms with the information available in a given set of data. Hand-crafted feature relied on the specific expert domain which poses a series of challenges to researches which we will further explore it in Section 2.2.1. On the other hand, learning features are feature that permits a model to capture significant attributes that help in the HAR. They can be classified into supervised learning and unsupervised learning. The difference between supervised learning and unsupervised learning is that Supervised learning features use tag data. We will discuss more in Section 2.2.2. It is worth noting that learning features help laid the foundation work for deep learning, which is the main theme of this research. The deep learning method is further explored 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 effectiveness of the HAR system depends on the proper and accurate feature representation (Abidine, Fergani, Fergani, & Oussalah, 2018). Therefore, extracting the appropriate hand-crafted feature 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 sets a threshold to the subsequent and presiding pixels before labeling the new pixel (Asuroglu et al., 2017). Asuroglu et al. (2017) utilizes LBP to query signal and encoded the signal by the histogram of those patterns before inputting them into a k-Nearest Neighbour classifier in their HAR framework. This approach produces an accuracy of 91.37%.

On the other hand, Empirical Cumulative Distribution Function (ECDF) is computed from the empirical cumulative distribution of all axes (Hammerla et al., 2013). Hammerla et al. (2013) used ECDF to conserve important representations of the accelerometer data, such as its spatial position and general shape which also improves the performance of HAR.

The handcrafted-features were usually used in conjunction with machine learning approaches for object recognition and computer vision like Support Vector Machines (SVM) (Cortes & Vapnik, 1995b). 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 well known that quantifying the exact effectiveness of a hand-crafted feature used in different platforms and the time taken for its selection and reducing its dimension possess a series of challenges to HAR(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. With the absence of hand-crafted features, they can extract features from the data on their own. Before discussing deep learning, learning features are first explored and some examples of learning feature algorithms are mentioned in the following section.

2.2.2 Learning Feature

As mentioned previously, learning features are feature that permits a model to capture significant attributes that help in the HAR. One of the most common learning features is Principle Component Analysis (PCA) which can learn a complete set of vectors (Smith, 2002). It captures significant representations in data and reveals their similarities and differences. Sprager and Zazula (2000) utilizes PCA to minimize the dimensionality on accelerometer data for HAR, which reports an accuracy of 90.3%. Wang, Chen, Hao, Peng, and Hu (2019) build a HAR model that utilized a new supervised learning based on a modified sparse PCA model, which achieved a fairly accurate result of 86.6%.

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 makes the smartphone-based and wearable sensor HAR challenging. Current HAR relies too much on hand-crafted features that sometimes they fail to predict complex activities accurately (Friday Nweke et al., 2018).

Deep learning methods are able to remove the dependency of the time-consuming hand-crafted feature that researchers faced (Friday Nweke et al., 2018). The key component of the deep learning model is its layered architecture that enables deeper extraction of features, thus allowing the classification work much easier. Deep learning is usually employed in the field of image or video recognition such as surveillance cameras in airport or traffic. Thus, it makes sense that HAR can be employed using deep learning methods as well, given that the track record of deep learning is very promising (Ronao and Cho, 2016; Shi, Li, Zhou, and Liu, 2018; Wang et al., 2019).

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)

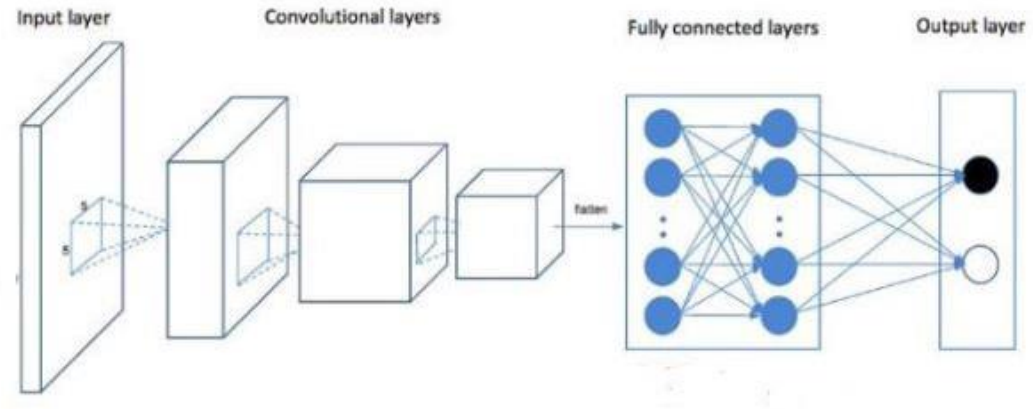

Figure 2.1: Convolutional Layers and Fully Connected Layers

Convolutional Neural Network (CNN) or commonly referred to as Covnet is proposed by a French professor named Yann LeCun (LeCun et al., 1989). A convolutional neural network consists of convolutional layers. Each layer consists of a set of neurons. It is three dimensional, consisting of height, width, and depth. Another version of the convolutional layer is called the fully connected layer, whereby each neuron receives input from the previous layer, while CNN neurons only receives a specific part of the input from the previous layer. Convolutional layers usually work with fully connected layers in image identification or, in our case, action recognition. Fully connected layers are used in this research and referred to as Dense layers. Figure 2.1 shows both the convolutional layers and fully connected layers.

CNN in HAR is already a well-researched area. For example, Zeng et al. (2014) used CNN to capture the local dependencies and spatial domain of activity signals. They utilized multichannel time series data to recognize the user's activity and hand gestures. Ronao and Cho (2016) constructed a deep CNN that perform one-dimensional (1D) convolution on the accelerometer and gyroscope triaxial sensor data. Their network achieved 94.79% accuracy on the data. Similarly, Lee, S. M., Cho, H., and Yoon (2017) also uses a 1D CNN method for HAR on the data gathered using a built-in accelerometer sensor in the smartphone. The triaxial acceleration data are converted into a vector input for the 1D CNN to learn and train. The model achieved an accuracy of 92.71%.

1D CNN is utilized for a single time series data, while two-dimensional (2D) CNN works best in multiple time series to capture local dependency and spatial domain of multi-modal data. Ha and Choi (2016) applied 2D CNN on multi-modal data and 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) utilized three-dimensional (3D) CNN model in HAR which is able to extract representation from both the spatial and the temporal dimensions. The model is able to convert multiple channels into one final representation of the feature and achieves an overall accuracy of 90.2%.

2.3.2 Recurrent Neural Network (RNN)

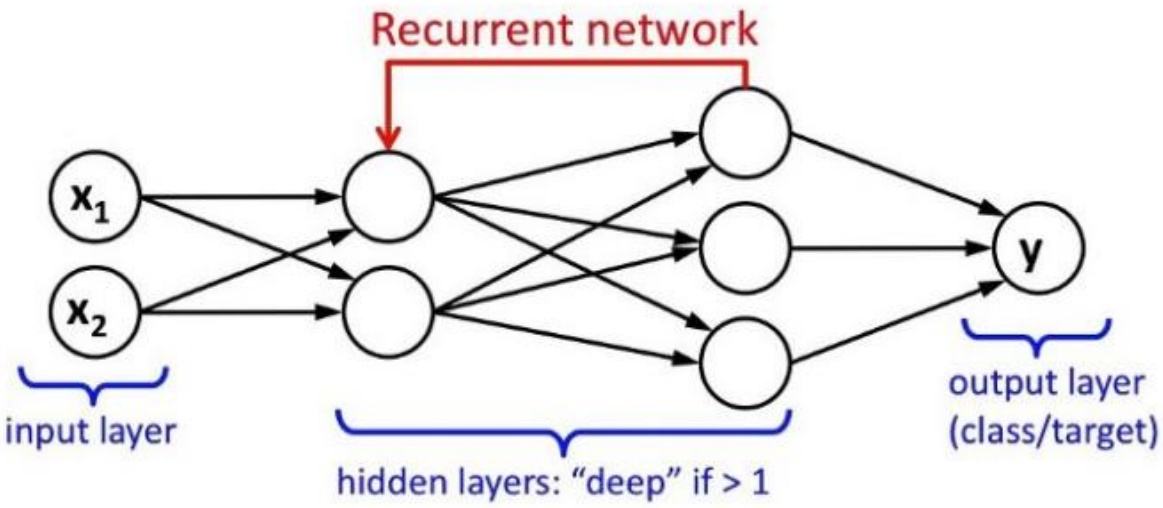

Figure 2.2: Recurrent Neural Network

Recurrent neural network (RNN) is inspired by Hopfield Net developed by John Hopfield (Hopfield, 1982). It was developed to process sequential or continuous data such as time signal and sensor data (Friday Nweke et al., 2018). Figure 2.2 shows the RNN architecture. RNN utilizes a technique call recurrent, meaning the network will consistently update each node based on the old and new information available. The state of the network will also update accordingly. In short, when the previous state is activated, the current state is predicted by estimating the next state.

However, as pointed out by Pascanu, Mikolov, and Bengio (2012), and Guan and Plötz (2017), the vanishing or exploding gradient makes training the model more challenging. Exploding gradient refers to a large error accumulated causing a significant change in the neural network while vanishing gradient refers to errors that lead gradient to be vanished. Vanishing gradient problem can attribute to RNN forgetting the first inputs due to memory cells have a short-term memory, meaning the information receives earlier may be lost and this causes the model not able to reflect its current state (Hernández et al., 2019; Yu & Qin, 2018). This limits its application for HAR, image recognition or computer vision.

2.3.2.1 Long Short-Term Memory (LSTM)

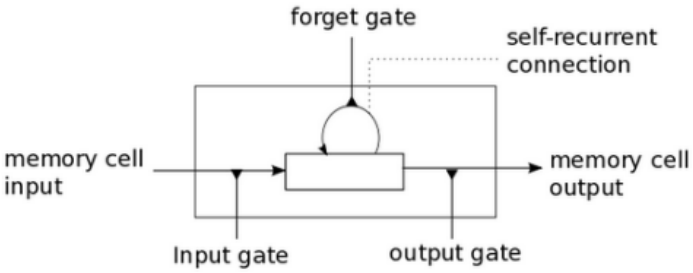

Figure 2.3: Long Short-Term Memory

Long Short-Term Memory (LSTM) is another variant of RNN. It is able to get rid of the vanishing gradient problem by implementing four gates which are discussed in detail 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. Chen, Zhong, Zhang, Sun, and Zhao (2016) utilizes the LSTM HAR model on triaxial accelerometers data from the WISDM dataset and achieves 92.1% accuracy in classification. In short, LSTM can use past information to predict the outcome of a HAR model. However, some information may not be captured as human motion is continuous (Yu & Qin, 2018).

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

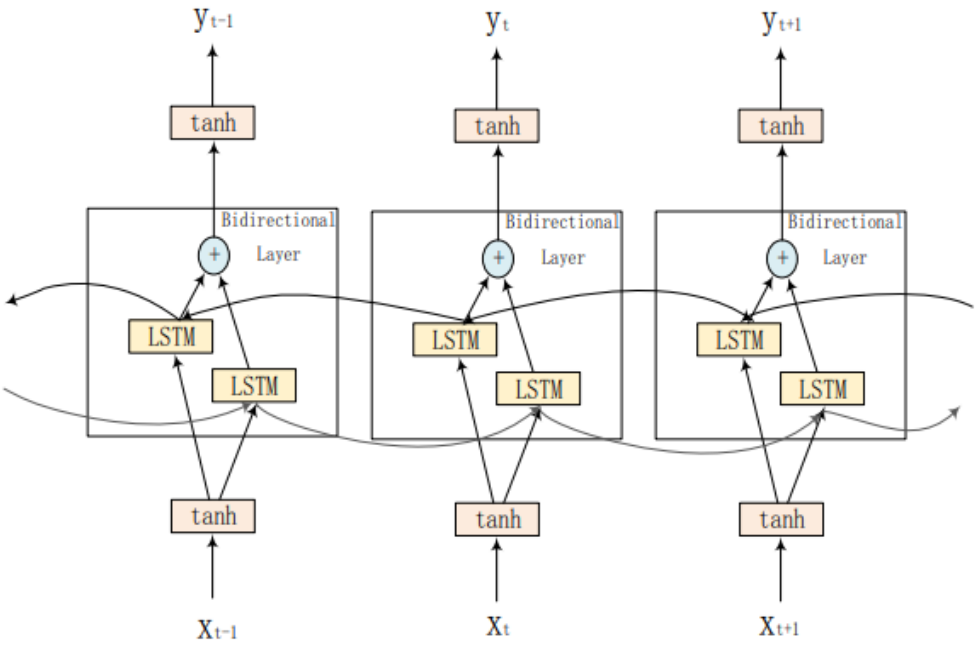

Figure 2.4: Bi-directional Long Short-Term Memory

Bi-directional Long Short-Term Memory (Bidir-LSTM) is a variant of LSTM. However, different from LSTM who only takes in past information, Bidir-LSTM uses both past information and future information, as shown in Figure 2.4. In other words, Bidir-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) utilizes Bidir-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 Bidir- 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

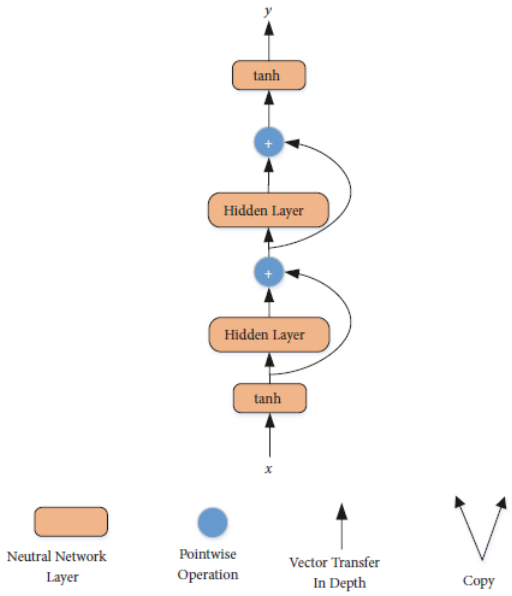

Figure 2.5: Deep Residue Bidir-LSTM

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 uses residual connections, rather than a gate. This residual connection ensures that the Bidir-LSTM gate is always open and allow information to be channeled to the additional residual functions to be learned. The information below can be relayed to the higher layer directly through a shortcut connection, bypassing some restrictive layer. They achieve 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

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 (Danesi, 1995). Hardware specifications used and better tuning of the model's hyperparameter

configuration also have to be taken into account. The techniques involved in creating these models are sophisticated and beyond the discussion of this research.

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 • 2 Intel Xeon E5 CPUs • 2 NVIDIA Quadro K5200 GPUs • 8 GB of RAM, 2304 CUDA cores, and a bandwidth of 192 GB/s
	(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 • Intel(R) Core(TM) i7-6700HQ 2.60GHz (8 CPUs) • NVIDIA GeForce GTX 950M GPU with 8GB of RAM
Deep Residue Bidir-LSTM	(Zhao et al., 2018)	UCI	93.60	• Utilizes deep residue and bidirectional information • i7 CPU • NVIDIA GTX 960m GPU with 8 GB RAM

2.4 Public Dataset Comparison (WISDM and UCI)

The two public datasets WISDM and UCI are used in this research to validate the result of the proposed model. The reasons for using two of these datasets can be summarized as follow:

- The data collected using the accelerometer in the smartphone
- Safe time in data collection
- Well-documented
- Huge amount of data

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.

In the UCI dataset, 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 UCI and WISDM dataset

Feature	UCI dataset	WISDM dataset
Activities	Walking, Walking upstairs, Walking downstairs, Sitting, Standing and Laying down	Walking, Jogging, Stairs-Up, Stairs-Down, Sitting, and Standing
Number of volunteers	30	29
Location of smartphone	waist	Front leg pocket
Collect	Triaxial gravity acceleration from the accelerometer and triaxial body acceleration and triaxial angular velocity from the gyroscope	Triaxial gravity acceleration from accelerometers
Frequency of data collection	50Hz (50 samples per second)	20Hz (20 samples per second)
Time steps	2.56 second (128)	10 second (200)
Overlap window, moving window	50%, 128 samples	Defined by user, usually 50% (Lee & Kwan, 2018), 200 sample
Number of samples	10,299	27,452
Number of attributes/features	561	6 (raw data file) 46 (transformed data file)
Training-testing split	70-30	Defined by user, usually 70-30
References	Anguita et al. (2013).	Kwapisz et al. (2011)

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, it can be concluded 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. Theoretically, it can be argued that utilizing the fusion of these two neural networks will enable the model to extract a deeper and more insightful features and thus better classification of the human activities. To prove this hypothesis, a fusion of CNN and RNN (Bidir-LSTM) model for better human action recognition is proposed.

In the next chapter, the individual architecture and formula of CNN, LSTM and Bi-dir LSTM used are discussed as well as the proposed model.

CHAPTER 3

THE CONCEPT AND COMPONENTS USED IN DEEP LEARNING MODEL FOR HUMAN ACTIVITY RECOGNITION

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. Classifiers used in the third experiment will also be briefly 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 a French professor named Yann LeCun. A convolutional neural network consists of convolutional layers. Each layer consists of a set of neurons. It is three dimensional, consisting of height, width, and depth. Another version of the convolutional layer is called the fully connected layer, whereby each neuron receives input from the previous layer, while CNN neurons only receives a specific part of the input from the previous layer. Convolutional layers usually work with fully connected layers in image identification or, in our case, action recognition. Fully connected layers are used in this research and referred to as Dense layers.

3.1.1 Architecture and Formulation of CNN

CNN conducts convolution operations on the time-series signal. In our model, we based on Yu and Qin's (2018) equations. Below is the simplified calculation for the output of the CNN:

$$C_N = A(B + W_{N-1}V) \tag{3.1}$$

Where C denotes CNN output at N layer, A is the activation function such as Relu, B refers to the bias term in CNN, W is the weight from the previous layer in CNN, V is the input vector of our accelerometer signal in our datasets.

3.1.2 Max Pooling

Max Pooling conduction pooling operation, usually on CNN output. The pooling operation used in this paper is also based on Yu and Qin (2018)'s equation: The pooling operation converts the input sets to the highest. The equation is as simple as:

$$P = MAX(C_N) \tag{3.2}$$

Where P is the output of the pooling operation, and C_N is the CNN output at the N th layer. CNN and Maxpooling often place subsequent to each other in deep learning models to learn representations of a set of input. The more the number of CNN-Maxpooling layers the deeper the learning is. Distinctive and significant features can be extracted utilizing this method.

3.1.3 Dropout layer

Dropout arbitrary select and remove a region of nodes in the neural network model for a period of time. This process enables the neurons to be independent and allow a more robust learning. The neurons will not be dependent on specific neurons. The dropout layer helps in solving the overfitting problem whereby the model performance varies on training and test datasets.

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 chooses which information the model need to "forget" or remove. The input gate is a simple activation function usually a sigmoid or tanh function (refer Section 3.5) which output the value in the range of 0 to 1. The input gate outputs the value to the state gate. The state gate is the main driver of the nodes' state which will be passed to the next LSTM node. The state is represented by a combination of input vectors from both the input gates and forget gates. The output gate outputs the combined input vectors to the next LSTM node, whereby the process repeats itself. The forget gate enables the model to decide whether to take in past information and to subsequently use this information to update the node, which gives the LSTM the ability to overcome the short-term memory problem faced in RNN (Yu & Qin, 2018).

3.2.1 Architecture and Formulation of LSTM

An LSTM node consists of four gates, which are the forget gate, input gate, state gate, and output gate. LSTM has two outputs which are coming from the output gate and node state. The following equation derived from Yu and Qin (2018). First, we look into the forget gate:

$$F_n = A\left(W_f[L_{n-1,}V_n] + B_f\right) \tag{3.3}$$

Where F_n is the forget gate output at n layer and A is the activation function, W_f denotes the weight of the connection at forget gate, L is the LSTM output from the previous layer, V_n denotes the input vector of the dataset, and B_f denotes the bias term for the forget gate.

The next gate is the input gate which is as follows:

$$I_n = A(W_i[L_{n-1}, V_n] + B_i) \tag{3.4}$$

Where I_n is the input gate output at n layer and A is the activation function, W_i denotes the weight of the connection at the input gate, L is the LSTM output from the previous layer, V_n denotes the input vector of the dataset, and B_i denotes the bias term for the input gate.

Similar to the input gate and forget gate, the output gate exhibits similar equations:

$$O_n = A(W_o[L_{n-1}, V_n] + B_o) \tag{3.5}$$

Where O_n denotes the output gate output at n layer and A is the activation function, $W_o$denotes the weight of the connection at the output gate, L is the LSTM output from the previous layer, V_n denotes the input vector of the dataset, and B_o denotes the bias term for the output gate.

The state gate does state update by combining with the forget gate to give the following equation:

$$C_n = F_n C_{n-1} + A(W_c[L_{n-1}, V_n] + B_c) \tag{3.6}$$

Where C_n denotes the state gate output at n layer, F_n refers to the forget gate output at n layer, and A is the activation function, W_c denotes the weight of the connection at the state gate, L is the LSTM output from the previous layer, V_n denotes the input vector of the dataset, and B_c denotes the bias term for the state gate.

And the LSTM outputs the combination of the output gate and state gate, with the following equation:

$$L_n = O_n * \mathrm{A}(C_n) \tag{3.7}$$

Where L_n denotes the LSTM output at n layer, O_n refers to the output gate's output at n layer, and A is the activation function, C_n denotes the state gate's output at n layer as well.

The LSTM node output two values to the next LSTM nodes. The subsequent LSTM node will use these two information accordingly to update their state and eventually the update status of the whole network. This gives the LSTM network takes the ability to take into past information. In the next section, the Bidir-LSTM is able to combine both past information and future information to update the network.

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 following Bidir-LSTM is derived from Yu and Qin's (2018) equations. Bidir-LSTM has both forward sequences $\vec{L}$ and backward sequences $\overleftarrow{L}$ in the hidden layer. At time t, the hidden layer and the input layer can be defined as follows:

$$\vec{L} = A(W_{\vec{L}}\,\vec{L}_{n-1} + B_{\vec{L}}) \tag{3.8}$$

$$\overleftarrow{L} = A(W_{\overleftarrow{L}}\,\overleftarrow{L}_{n-1} + B_{\vec{L}}) \tag{3.9}$$

$$\overleftrightarrow{L} = \mathrm{A}(\vec{L} + \overleftarrow{L} + B_{\overleftrightarrow{L}}) \tag{3.10}$$

Where $\vec{L}$ denotes THE forward sequence, $\overleftarrow{L}$ denotes the backward sequence of LSTM operation, $\overleftrightarrow{L}$ refers to the Bidir-LSTM output, A refers to the activation function used in the network, W refers to the weight of the connection and B refers to the bias term.

These forward and backward sequences allow the LSTM nodes to takes in previous and subsequent information to update its state, this will subsequently update the state of the whole network. This gives the Bidir-LSTM's properties to use past and future information to effectively output a deeper representation of a set of data inputs.

3.4 Weight decay/Regularization

A model will reach a bottleneck where the weight updates are prohibited beyond a certain value. This is because gradient descent, in nature, only allow small changes. This makes exploring the weight space to tune the model to avoid overfitting very challenging. Ergo, Weight decay is used to penalize weight size to the loss function. It is a form of regularization. The following equation is derived from Ronao and Cho's (2016) equation:

$$R = R_0 + \sigma \sum_{W} W^2 \tag{3.13}$$

Where R_0 is the initial cost function, and σ is the weight decay coefficient, W denotes the weight.

3.5 Activation function

The activation function is used to determine the output of neural networks 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. However, the issue is that all the negative values become zero immediately which leads to underfitting. Relu activation function converts any negative number into 0, which leads to no learning and cause all the neurons are not functioning. The advantage of using Relu as it has a simpler mathematical operation.

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, this allows 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 allows the drawing of the sigmoid slope at any two points.

3.5.3 Tanh function

Tanh, on the other hand, has a range from -1 to 1. Tanh allows negative inputs to be mapped negative and 0 to be mapped 0 in the graph. The function is differentiable and repetitive while its derivative is not repetitive. The tanh function is mainly used in classification between two classes.

3.5.4 Softmax function

Softmax function is a function that uses a set of the input vector and performs normalization so that it turns into a set of probability distribution. Applying softmax will output the number between 0 to 1 and can be interpreted as a probability. Bigger inputs usually associates with larger probabilities. Softmax is popular in deep learning especially neural networks such as CNN and LSTM. It is able to output a set of probability distribution for a given output class. It is considered a more generalized logistic activation function that is often implemented in multiclass classification

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 optimizer that provide differentiable property to a given function which allows convergence to be faster. It is stochastic in nature because random samples are chosen to evaluate the gradient and updates are performed more frequently. SGD is essentially a stochastic approximation of gradient descent. (Herbert Robbins and Sutton Monro, 1951)

3.6.2 Root Mean Square Propagation (RMSProp)

Root Mean Square Propagation (RMSProp) is an optimization method that discovers a single global learning rate by adapting the step size which defines the 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 extension of the SGD optimizer which uses two Adaptive Gradient Algorithm (AdaGrad) and RMSProp. In this optimization algorithm, it used running averages of the first and second gradients to adapt the learning rate. (Kingma and Ba, 2015)

3.7 Classifiers

Classifiers or classification algorithms are generally used to classify images such as dogs and cats. It is usually employed in the image recognition field. However, it is also widely adopted in HAR. In this research, nine machine learning classifiers are used to classify the different activities based on the temporal data obtained through accelerometers.

3.7.1 Logistics Regression

Logistic regression is named for the function used which is the logistic function (Scott, Hosmer, & Lemeshow, 1991). The logistic function is actually the sigmoid function discussed in Section 3.5.2 which takes in a real number and map it in the range of 0 and 1.

3.7.2 Support Vector Machine (SVM)

The SVM algorithm is implemented by transforming the problem using some linear algebra that enables a deeper insight using the inner product of any two given observations, rather than a single observation (Cortes & Vapnik, 1995). The inner product refers to the total of each input pairs' multiplication.

3.7.3 Random Forest

Random Forest is a type of classifier commonly referred to as Bootstrap Aggregation which is an extension of bagged decision trees (Breiman, 2001). It combines the predictions from various models in a group. It is efficient if there is no or weak correlation in the prediction of the sub-trees. It manipulates the algorithm in a way that allows the sub-trees to learned and reduce the correlation between the predictions generated from each sub-tree.

3.7.4 Naïve Bayes

Naive Bayes is a classifier for binary and multiclass classification such as HAR and image recognition. It is naïve because calculation can be traced as the probabilities of each class is simplified (Lewis, 1998). Each class is independent in terms of their condition for a given value. This eliminates the need to calculate every single value's probability, assuming that there is no interaction between the attributes.

3.7.5 Multilayer Perceptron

As discussed earlier in CHAPTER 2 about deep learning which involves neural networks. The term neural networks and multilayer perceptron are interchangeable. A perceptron is a single neuron model that lays the foundation for neural networks. It is commonly used to tackle difficult tasks such as HAR and image prediction in machine learning. The ability of multilayer perceptron to be trained, mapping (approximates) the output variables through their network hierarchical structure, and deriving representation from low-level features to high resolutions features is the main driving force to our research.

3.7.6 K-Nearest Neighbours (Scaled Manhattan)

K-Nearest Neighbours (KNN) is a classifier that uses cases with the highest frequency from the k-nearest instance to calculate the output (Cunningham & Delany, 2007). Every attributes need to vote a case. The prediction is generated when the case has the highest number of votes. Cases can be distance function, size and so forth.

A distance measure is a feature that decides which cases have the closest resemblance to new information. Manhattan distance uses the total absolute difference to compute the real vector distance. It works best with non-similar type data like colour, occupation, and sex.

3.7.7 K-Nearest Neighbours (Euclidean distance)

Another popular distance measure is the Euclidean distance. Euclidean works best with similar type data like heights measurement. Euclidean distance computes the square root of the total square differences between two points

3.7.8 Bayesian Network

Bayesian Network (Friedman, Geiger, & Goldszmidt, 1997) provides a basic solution of applying Bayes Theorem to sophisticated issues. Albeit both the nodes' probability distribution and edges are indicated abstractly, the systems are not Bayesian per se. The extraction of the model on the probabilities of the outcome is the main theme of Bayesian Network, different from the conventional way of relying on past events.

3.7.9 J48

J48 is a classifier that applies the C4.8 algorithm using Java programming language. C4.8 is an improvement of the C4.5 algorithm (Salzberg, 1994). C4.5 is an algorithm that applies a decision tree that is used in classification, commonly known as a statistical classifier.

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 different machine learning classifiers are discussed. In the next chapter, the first proposed model of this experiment is exhibited, followed by the experiment set up and the result evaluation.

CHAPTER 4

METHOD 1: EVALUATION OF C4M4BL AND THE EFFECT OF HYPERPARAMETER CONFIGURATION

In this chapter, the proposed model of this research are discussed in Section 4.1 and its architecture information in Section 4.2, followed by the environment set up in Section 4.3. Next, the training and testing set information are explored in Section 4.4. The main experiment of ten different configuration settings for both datasets to determine the best model is explored and compared with other classification methods using UCI and WISDM datasets in Section 4.5. Additional experiment to see the effect of hyperparameters on the performance model is also explored and evaluated in Section 4.6.

4.1 The Proposed Neural Network Model (C4M4BL)

In this study, we proposed to set up a neural network model integrating CNN and Bidirectional-LSTM. It consists of four layers of CNN, four layers of the Maxpooling layer, and one layer of Bidir-LSTM. henceforth will be known as CNN4-MAX4-BidirLSTM (C4M4BL) model in this paper. One CNN layer will be followed by one Maxpooling layer, creating a CNN-Maxpooling layer. The idea is that CNN could properly seize neighborhood dependency and keep scale invariance which means it is well suited for coming across relationships in spatial dimensions. However, CNN cannot extract temporal dependency in time-series data.

RNN as memory units are designed to work with time-series data, and it is suitable for capturing relationships in the temporal dimension. RNN takes a series of input and disregards the local dependencies from the original sensory data that form a feature vector (Lv, Xu, and Chen, 2019). 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 the WISDM dataset. The differences of both datasets are tabulated

in Section 0. 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.

4.2 Architecture of CNN4-MAX4-BidirLSTM model

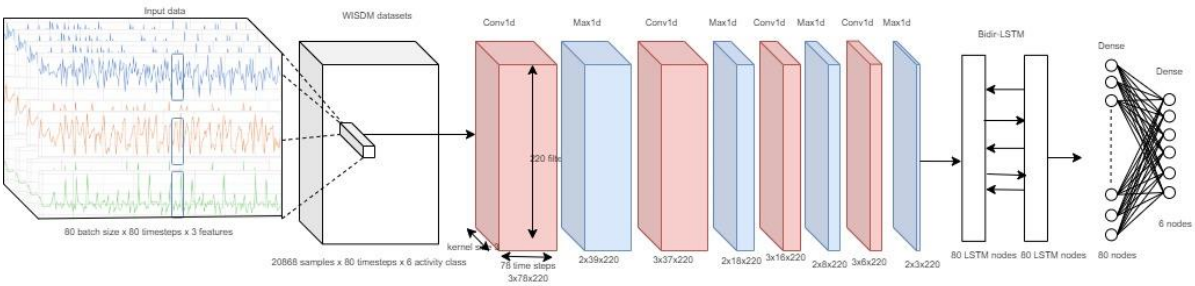

Figure 4.1: CNN4-MAX4-BidirLSTM (C4M4BL) model

Figure 4.1 shows the proposed model of CNN-Bidir-LSTM (C4M4BL) 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 the 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 is implemented to reduce overfitting, by removing or turning off on a selective region of nodes in a neural network. It makes training easier and neurons will not depend on any specific neurons. This allows the learning feature to be more reliable. 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 the default activation function for CNN layers and the first Dense layer, while Softmax will be implemented at the final activation function of the last Dense layer. The learning rate is 0.001 by default. The weight decay is also 0 by default. The proposed model is later used with ten different hyperparameter configuration settings for two datasets and the best model is chosen 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.

4.3 Environment setup and Hardware/Software Information

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 user-friendly, highly advanced Application Programming Interface (API) that provides a framework for creating and training a neural network model. Jupyter notebook is an open-source web application that allows users to interact with documents that provide visualizations of interactive codes, texts, and equations. The Jupyter notebook is used to visualize a variety of configuration results for 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.

4.4 Training and Test Sets Information

The data in UCI and WISDM is separated into a 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
Timestep	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.

4.5 UCI and WISDM Datasets 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 while configuration for the WISDM dataset 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

To make a justifiable comparison with the proposed model, other architecture models such as those used by Ronao and Cho 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. Deep residue Bidir-LSTM is also ignored as it is beyond the scope of this research.

With the limited information given, all of the stated architectures are built based on the information available, information that is not given is set to a default value, for example, the learning rate, weight decay, etc. Different models are trained using a high number of epochs (5000++) and processed on high-end workstations. The limitation in the tools available and time for this research can be overcome by having a default epoch. 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 weight decay are both set to 0 and 0.5 respectively. The batch size for UCI is set to 128 and WISDM is set to 80, equivalent to their respective timestep.

The models created are built using Keras Tensorflow and Jupyter Notebook is used to graphically display the process. All of the models are trained and test using two public datasets. 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 accuracy of each model proposed by other authors is set as a benchmark for later evaluation with the proposed model. Each Maxpooling layer has a default kernel size of 2 for both UCI and WISDM datasets. The default configuration is bolded in the table. The normalized models are set as the baseline of the performance of this experiment, whereby the proposed model must exceed.

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

The confusion matrix is used in this study to analyze the performance of our model when running these two datasets. We will look into Accuracy, Recall, and Precision which are the most common performance indexes. In this study, True class means the model predicts a class correctly. Accuracy is quite straightforward, which asks how often the model's predictions are true. Recall often asks, if the actual label is correct how often the model's predictions are true, it usually associates with the model sensitivity; whereas Precision often asks if the predicted label is correct, how often the

model's predictions are true. In this research, we will focus on both accuracy and recall to evaluate a model. The formula of each index can be summarized as follow:

$$Accuracy = \frac{True\ Class}{Total\ number\ of\ samples} \tag{4.1}$$

$$Recall = \frac{True\ Class}{Actual\ Class} \tag{4.2}$$

$$Precision = \frac{True\ Class}{Predicted\ Class} \tag{4.2}$$

4.5.1 UCI dataset Result and Discussion

The proposed model of CNN4-MAX4-BidirLSTM (C4M4BL) is first configured using the configuration setting in Table 4.3 for the UCI dataset. Each of the configuration has a different effect on the accuracy of the test datasets. Figure 4.2 and Table 4.6 show the validation result for each configuration. In this case, the c0 configuration achieved the highest validation accuracy on the training dataset which is 91.84%.

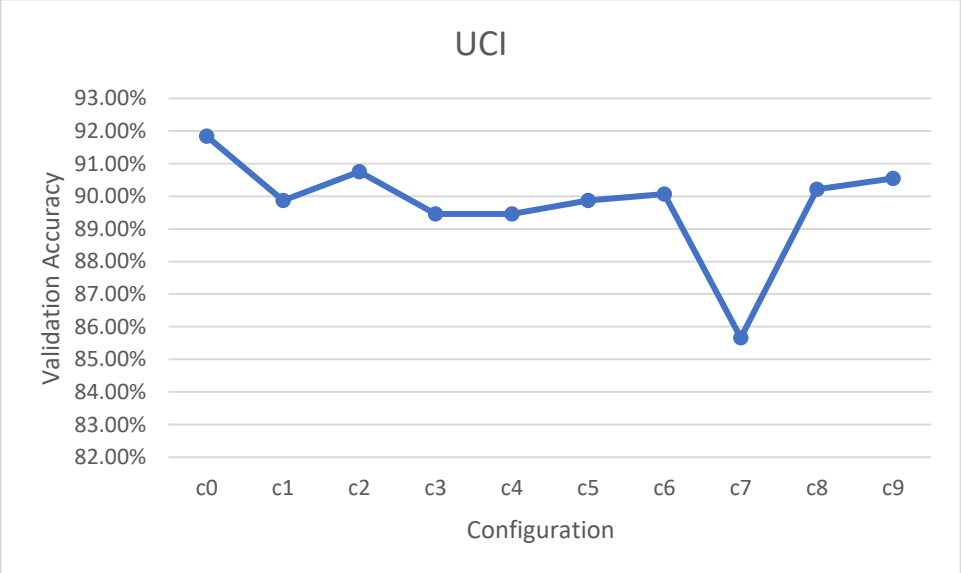

Figure 4.2: Validation Result using UCI training dataset

Table 4.6: Validation Result using UCI training dataset

	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
Validation Accuracy	**91.84 %**	89.87 %	90.75 %	89.46 %	89.46 %	89.87 %	90.07 %	85.66 %	90.21 %	90.55 %

Figure 4.3 shows the learning curve of the model using UCI training data against validation data using c0 configuration. The accuracy of both training data and validation data rose gradually and hover above 90%, while the loss of the training data gradually dropped below 20%. The loss of validation data, however, fluctuated in the range of 20% and 40%, which is relatively reasonable. Since the median filtered is used during the preprocessing phase of the model as mentioned in Section 0, we can rule out the effect of noise. This can only mean either the number of training/validation data is not sufficient or the hyperparameter configuration is not properly tuned. The effect of the hyperparameter configuration is further explored in the additional experiment (Section 4.6).

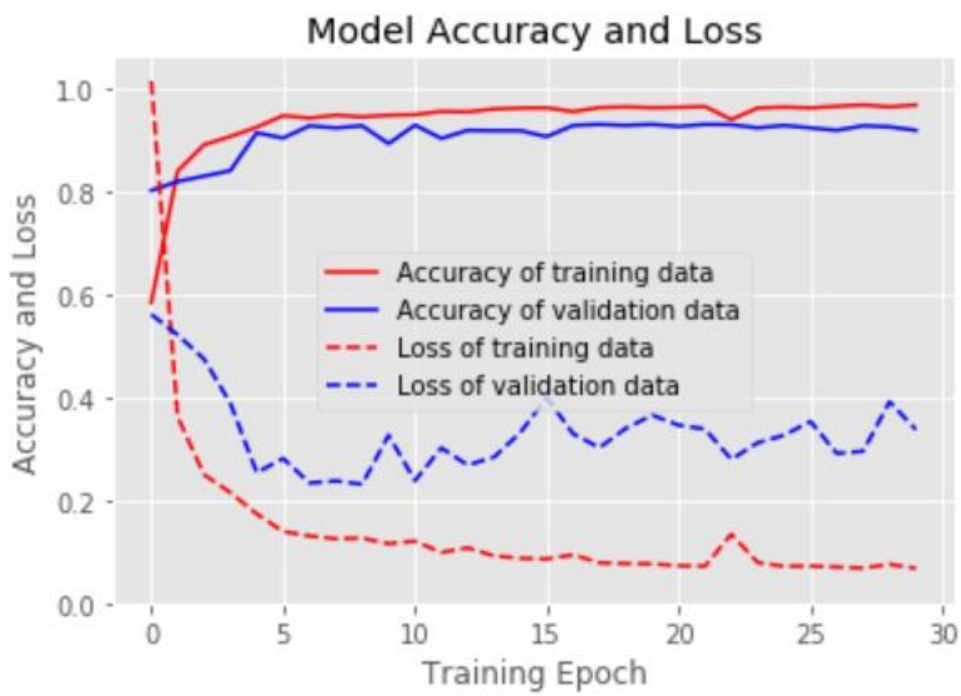

Figure 4.3: Learning curve of UCI Training data against Validation data using c0 Configuration

Table 4.7 shows the configuration matrix using the UCI test dataset with the c0 configuration. It achieves a very high accuracy of 90.57% in predicting the true class of the activity. Most of the motion activities such as walking (99.19%), walking upstairs (88.32%) and walking downstairs (99.05%) achieve high accuracy in prediction. With the exception of laying which achieves a high prediction of 94.97%, sitting and standing achieves the lowest prediction, accounting 84.11% and 79.32% respectively. This can attribute to both of their motion or triaxial accelerations are similar or does not fluctuate much, which lead the model to confuse the two activities.

Table 4.7: Configuration Matrix Using UCI Test datasets with c0 Configuration

		Predicted Class						
		W	**WU**	**WD**	**Si**	**St**	**L**	**Recall**
Actual Class	**Walking**	492	2	2	0	0	0	99.19%
	W. Upstair	37	416	18	0	0	0	88.32%
	W. Downstair	1	3	416	0	0	0	99.05%
	Sitting	0	25	0	413	53	0	84.11%
	Standing	0	1	0	109	422	0	79.32%
	Laying	0	27	0	0	0	510	94.97%
	Precision	92.83%	87.76%	95.41 %	79.12%	88.84 %	100 %	**90.57 %**

Since the configuration c0 achieves the highest validation accuracy, it 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 with the UCI test dataset. The result is compared with other deep learning models running the same dataset and shown in Table 4.8. 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.

Table 4.8: Result Comparison using UCI test 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	(Hernández, Suárez, Villamizar, & Altuve, 2019)	B(L(175))-B(L(175)-B(L(175))-DL(128)	87.41%
CNN4-MAX4-BidirLSTM	-	C(32)-MAX-C(32)-MAX-C(32)-MAX-C(32)-MAX-B(L(128))-DL(128)-DL(6)	**90.57%**

This can be attributed to the combination of CNN and Bidir-LSTM, which use four layers of CNN with 32 feature maps, one 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 means no penalty for additional weights. Large weight inhibits the weight vector from passing the local minimum, which in turn affects the performance of the model (Ronao and Cho, 2016).

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 layers 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 takes lesser time to obtain the 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.

4.5.2 WISDM dataset Result and Discussion

The CNN4-MAX4-BidirLSTM is configured using the configuration setting in Table 4.4 for the WISDM dataset. Figure 4.4 and Table 4.9 show the validation result of each configuration. In this case, c9 achieved the highest validation accuracy which is 84.04%.

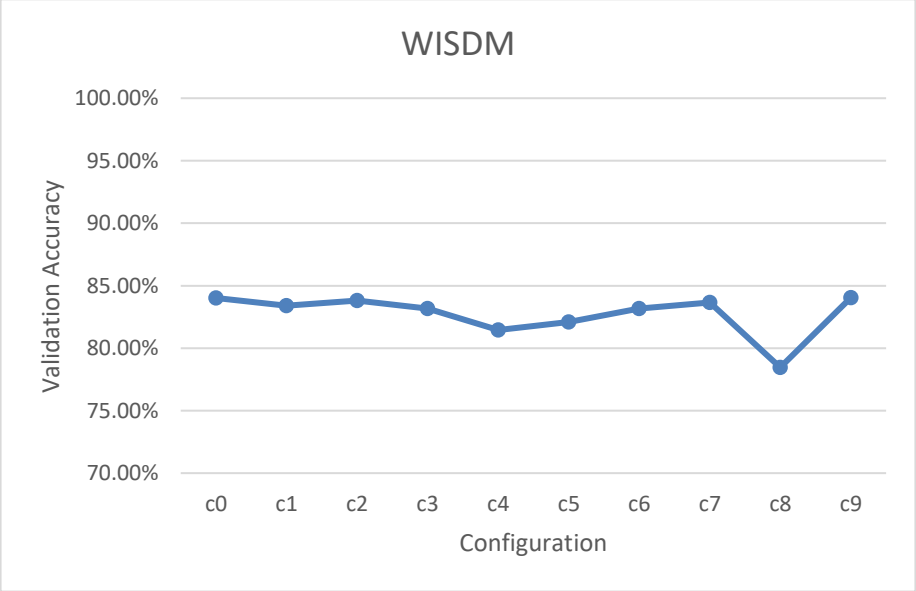

Figure 4.4: Validation Result using the WISDM training dataset

Table 4.9: Validation Result using the WISDM training dataset

	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
Validation Accuracy	84.02 %	83.40 %	83.80 %	83.16 %	81.46 %	82.10 %	83.18 %	83.66 %	78.46 %	**84.04 %**

Figure 4.5 shows the learning curve of WISDM training data against validation data using c9 configuration. The scenario is different than that of the UCI datasets. The accuracy of the training data did gradually rose to above 99%, very close to achieving 100%, but the validation accuracy did not follow the same pattern, it fluctuated in the range above 80%. The training loss followed the conventional pattern, as it decreased close to 0%, but the validation loss rose and fluctuated way above 50%. This can be attributed to three factors: lack of noise reduction implementation, number of training/validation data not enough and the hyperparameter configuration is not properly tuned.

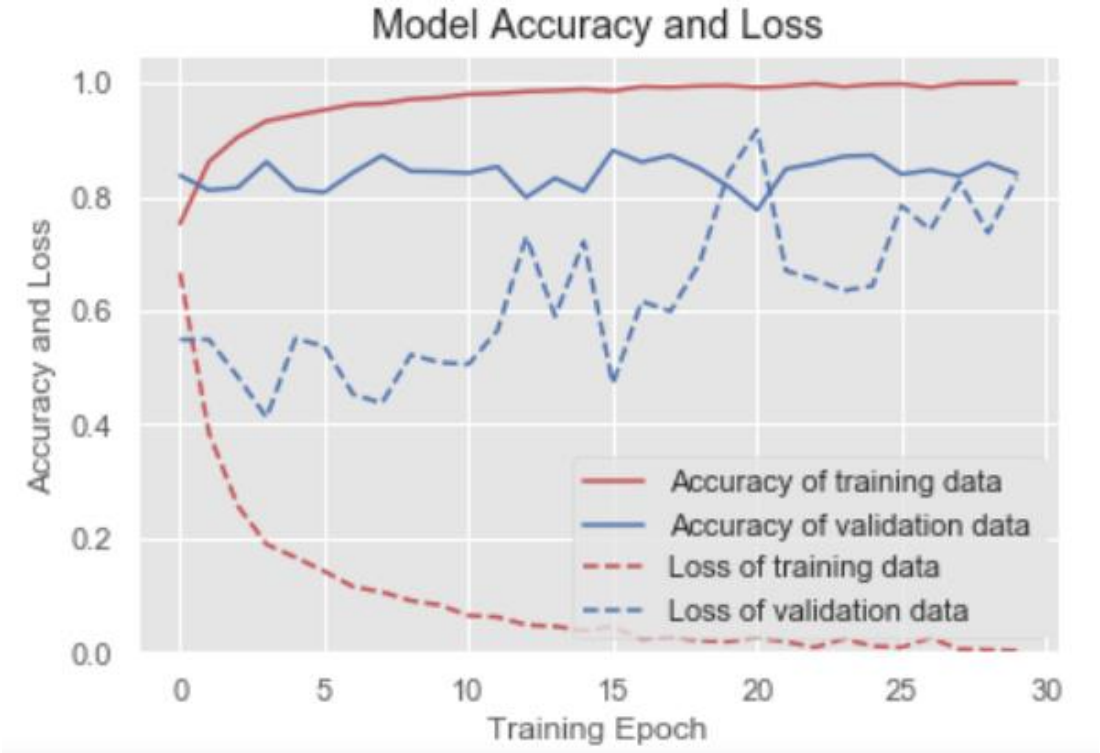

Figure 4.5: Learning curve of WISDM Training data against validation data using c9 Configuration

Table 4.10 shows the configuration matrix using the WISDM test dataset with the c9 configuration. It achieves a relatively high accuracy of 87.62%. The jogging class achieves the best classification in prediction among the six activities, achieving 95.58%, followed by sitting (91.15%). The lowest classification prediction accuracy is going downstairs, which achieve around 70%. The second-lowest is going upstairs which achieves 81.66%. The model usually confuses both going upstairs and going downstairs, which leads to more classification errors between the two activities.

Table 4.10: Configuration Matrix using the WISDM test dataset

		Predicted label						
		Down	**J**	**Si**	**St**	**Up**	**W**	**Recall**
Actual Label	**Downstair**	456	9	0	0	136	49	70.15%
	Jogging	18	1902	0	0	22	48	95.58%
	Sitting	0	0	412	0	1	39	91.15%
	Standing	0	0	41	328	1	0	88.89%
	Upstairs	73	20	2	0	592	38	81.66%
	Walking	38	231	1	0	48	2079	86.73%
	Precision	77.95 %	87.97 %	90.35 %	100 %	74.00 %	92.28 %	**87.62 %**

Since c9 achieved the highest validation accuracy of 84.04%, it is chosen to run the WISDM test dataset to obtain the best result of the model. The model achieves an 87.62% accuracy with the WISDM test dataset. The result is compared with other deep learning models running the same dataset and shown in Table 4.11. 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.

Table 4.11: Result Comparison using the 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)	**87.62**%

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 means there is no penalty for additional weight incurred. Large weight can inhibit the weight vector from going beyond the local minimum, which in turn, affects the performance of the model (Ronao and Cho, 2016).

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., and Yoon, 2017) and (Ronao and 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 takes lesser time to obtain results, 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.

4.6 Effect of Hyperparameter Configuration Experiment

The objective of the experiment is to examine the effect of hyperparameter on the performance of the CNN4-MAX4-BidirLSTM. Hyperparameter configuration ensures that the model is tuned to the best performance. The best performance of a model depends on the classification accuracy of the activities. WISDM dataset is selected for this experiment. Hyperparameters are parameters neural networks that are unable to learn themselves via gradient descent or some other variant. These include learning rate, number of layers, number of neurons in a given layer. Tuning the hyperparameters refers to the process of choosing the best values of the hyperparameters.

4.6.1 Experiment setup

The experiment is set up using the same environment as in Section 4.3. The default configuration setting is set up as shown in Table 4.12.

Table 4.12: 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

4.6.2 Hyperparameter Variable Results and Discussions

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. The other configuration will follow the default configuration

4.6.2.1 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 4.12, 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.

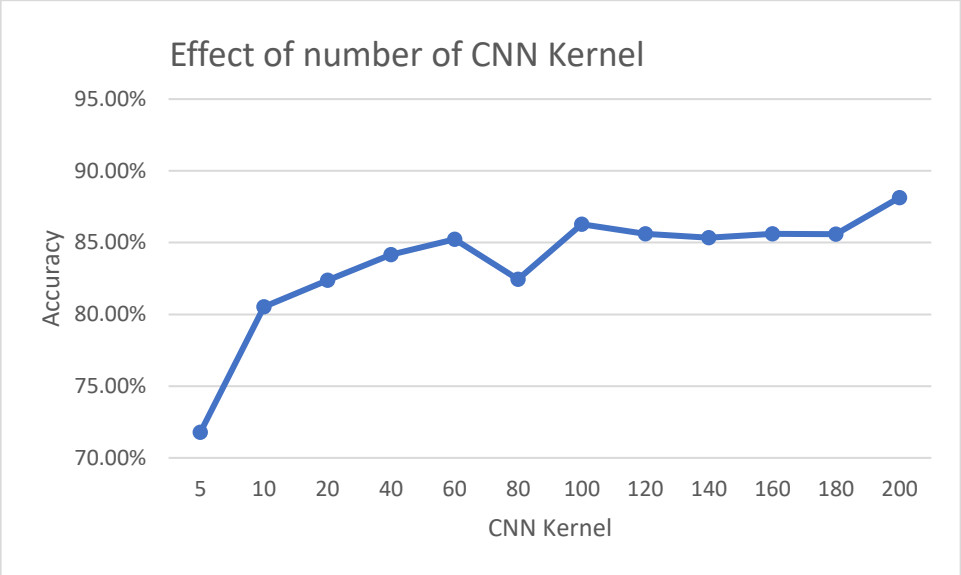

Figure 4.6: Effect of number of CNN kernel

Table 4.13: Effect of number of CNN kernel

Number of CNN kernel	Accuracy
5	71.78%
10	80.52%
20	82.37%
40	84.16%
60	85.22%
80	82.44%
100	86.27%
120	85.60%
140	85.34%
160	85.60%
180	85.58%
200	**88.14%**

Figure 4.6 and Table 4.13 shows that the number of feature maps is directly proportional to the validation accuracy of the model for WISDM models. The best performance is the number of CNN kernels = 200 (88.14%). Theoretically, the more the feature maps or CNN kernels, the more abstract representation can be produced.

It improves an overall 0.52% from the result in evaluation experiment (87.62%), while having the model configuration of 200-200-200-200-100-100-6, where the first four values denotes the number of kernel in the CNN/Maxpooling layer, and the fifth values denotes the number of Bidir-LSTM nodes and the last two value indicates the number of the dense nodes. It can be seen the accuracy rose significantly twice, once is about 8.74% from 5 to 10 feature maps, and about 2.56% from 180 to 200 feature maps. From 10 to 180 feature maps, it fluctuated around 1% to 2%.

4.6.2.2 Effect of CNN Kernel Size

The experiment is set up using CNN kernel size with the range from 1 to 5. Kernel size or sometimes calls "filter size" is the dimensions of the feature maps or filters on CNN. The model is set up using the configuration setting shown in Table 4.12, 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.

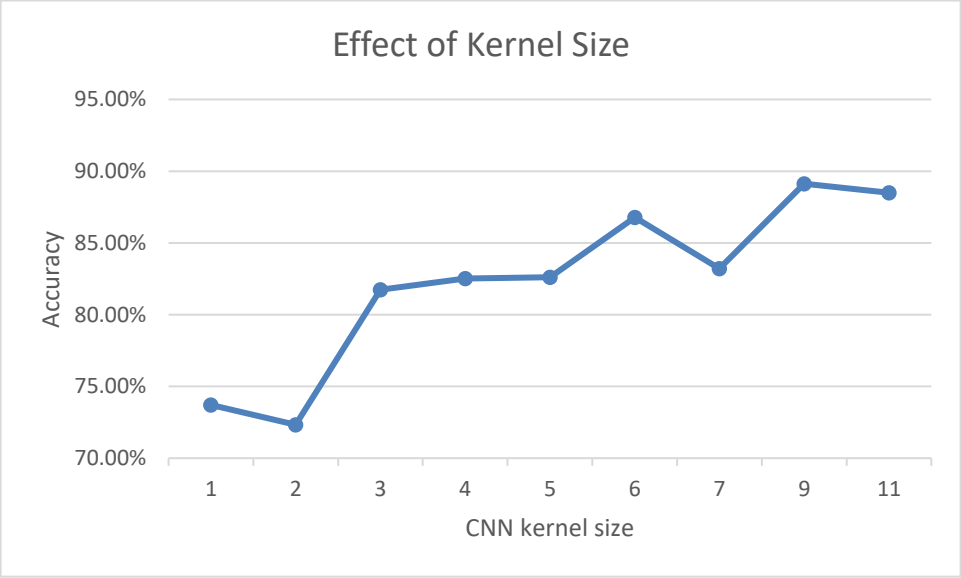

Figure 4.7: Effect of CNN Kernel Size

Table 4.14: Effect of CNN Kernel Size

Kernel Size	Accuracy
1	73.72%
2	72.33%
3	81.74%
4	82.53%
5	82.61%
6	86.77%
7	83.20%
9	**89.13%**
11	88.50%

Figure 4.7 and Table 4.14 show that the best performance for the WISDM dataset is the kernel size 9 which achieves an accuracy of 89.13%. The accuracy rose steadily from 72.33% to 86.77%, which accounts for a whopping 14.44% increase, then it dropped slightly to 83.20% before heading to the highest peak of 89.13%, about 6% increase. This experiment implied we can exploit a higher range of temporal local dependency (kernel size 1 to 9) to explore its effect on performance accuracy, as opposed to only using kernel size of 3 which is the usual configuration. A larger size kernel can overlook the features and could skip the essential details in the images whereas a smaller size kernel could provide more information leading to more confusion. Thus there is a need to determine the most suitable size of the kernel/filter.

4.6.2.3 Effect of Activation functions

As discussed in Section 3.5, 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.

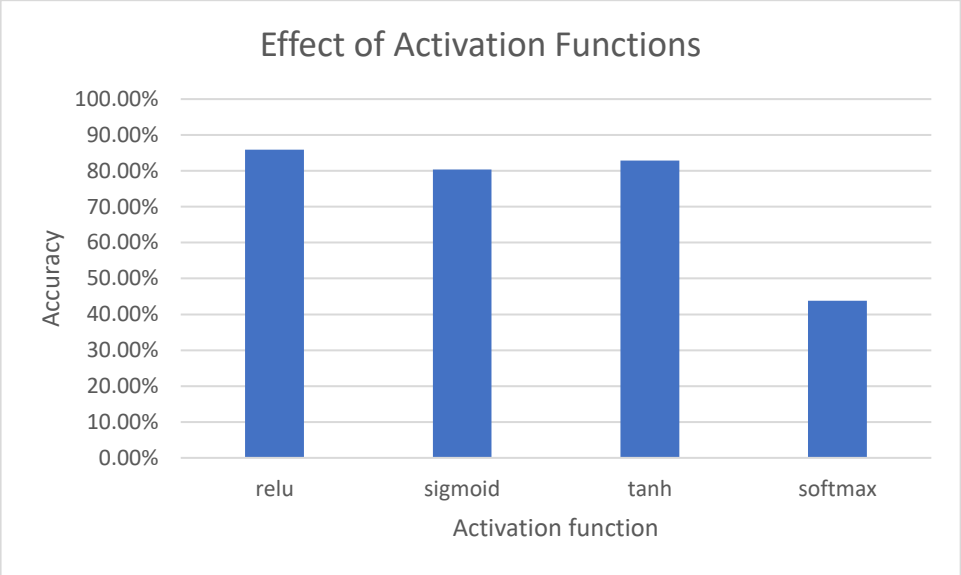

Figure 4.8: Effect of Activation Functions

Table 4.15: Effect of Activation Functions

Activation	Accuracy
Relu	**85.91%**
Sigmoid	80.35%
Tanh	82.89%
Softmax	43.82%

Figure 4.8 and Table 4.15 show 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 cheaper than Tanh and Sigmoid in terms of computational processing because it involves easier mathematical operations as discussed in Section 3.5. Softmax has the least accuracy of less than 50% as it is usually used at the final layer, not suitable in between the layers. Tanh and Relu has roughly the same performance, but it seems Relu is better at generalizing data.

4.6.2.4 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 4.12, 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.

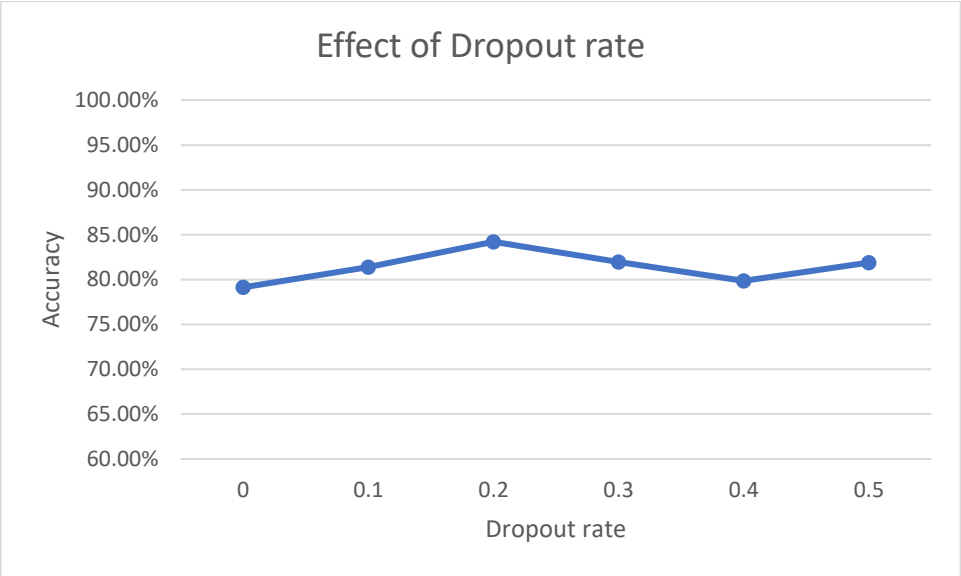

Figure 4.9: Effect of Dropout rates

Table 4.16: Effect of Dropout rates

Dropout	WISDM
0	79.13%
0.1	81.38%
0.2	**84.21%**
0.3	81.96%
0.4	79.85%
0.5	81.89%

Figure 4.9 and Table 4.16 show that the best dropout rate is 0.2 which achieves an accuracy of 84.21% during validation**.** Dropout arbitrary chooses neurons to be turned off during training. They are "dropped-out" randomly meaning that weight updates will not be applied in backward pass and the connections of the neurons are removed out of the equation in the forward pass (Srivastava, Hinton, Krizhevsky,

Sutskever, & Salakhutdinov, 2014). The performance improved by about 5.08% from dropout rate of 0 to 0.2. It declined to 79.85% from 84.21% after 0.2. The changes fluctuated in the range of 79% to 84%. Therefore, there is a lack of solid justification for the effect of dropout rate on the performance. Anything beyond dropout rate of 0.5 makes more than 50% of neuron inactive which means rendering the model inefficient.

4.6.2.5 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 4.12, only the epoch is changed for each experiment. This is to see the effect of epoch have on the overall performance of the model.

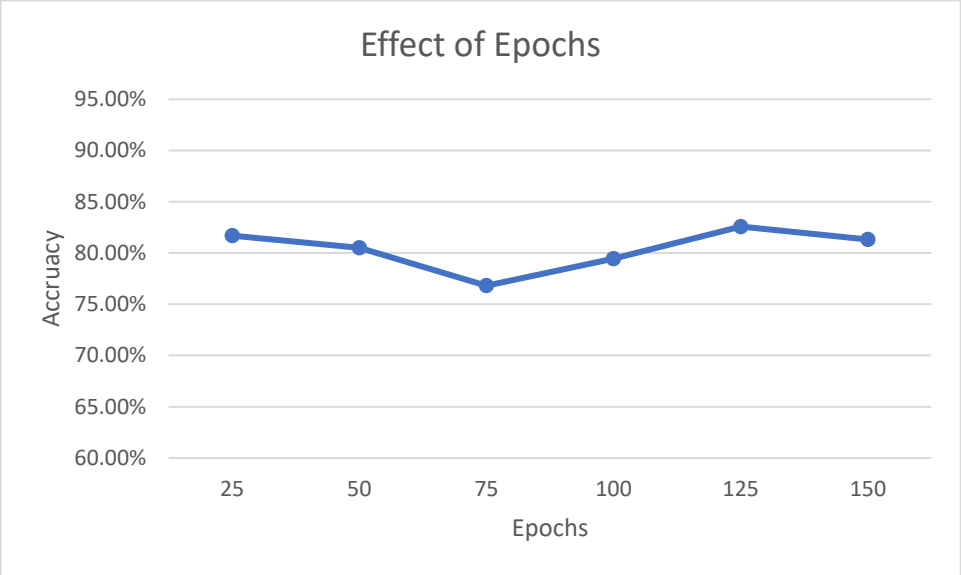

Figure 4.10: Effect of Epochs

Table 4.17: Effect of Epochs

Epochs	Accuracy
25	81.70%
50	80.50%
75	76.81%
100	79.44%
125	**82.58%**
150	81.31%

Figure 4.10 and Table 4.17 show that the best epoch is 125, which achieves an accuracy of 82.58% during validation. The accuracy of the model dropped from 81.70% to 76.81% initially, and rise to a peak of 82.58%. That is a drop of about 4.89% and a rise of about 6%. There might be a rise if we run high number epoch like 5000 or 10000 epochs to effectively train the model, but due to limitation of the processing power and time, we can only see a certain range of epoch (less than 200) which only summarized the effect of epoch. However, it is worth noting that more epochs may lead to overfitting. So better computational power is required to observe the effect of epochs on the performance of the model in future work.

4.6.2.6 Effect of Optimizer

The WISDM dataset is tested using both Adam and Stochastic Gradient Descent (SGD) optimizer. As discussed in Section 3.6, 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. In this experiment, the model is set up using the configuration setting shown in Table 4.12, 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.

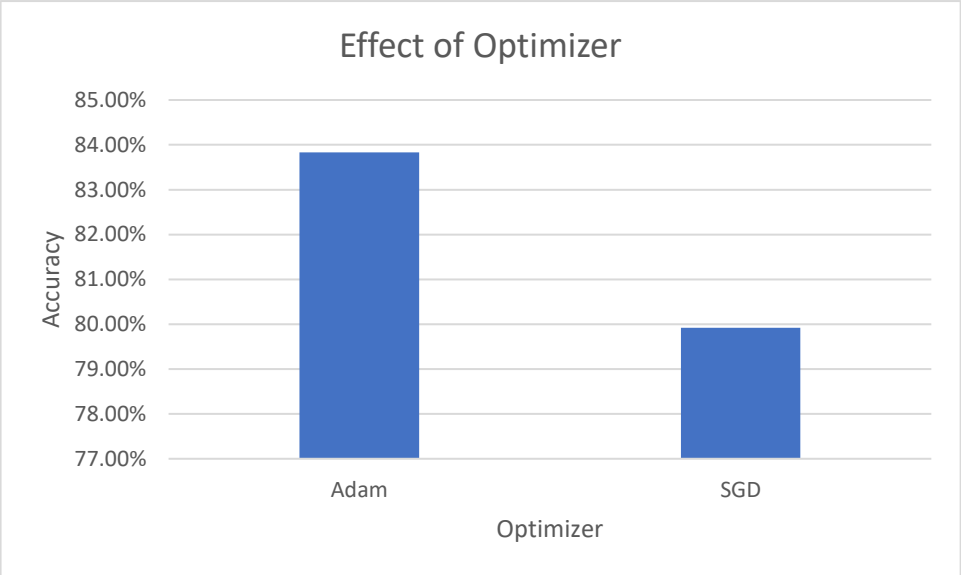

Figure 4.11: Effect of Optimizer

Table 4.18: Effect of Optimizer

Optimizer	Accuracy
Adam	**83.83%**
SGD	79.92%

Figure 4.11 and Table 4.18 show that Adam has the highest performance of 83.83% when compared to that of SGD (79.92%). That is a difference of about 4% in terms of the accuracy of these two optimizers. This can be attributed to SGD maintains a global learning rate for all weight updates. This means that the learning rate remains the same during training. Adam combining the advantages of two other extensions of SGD which are the AdaGrad and RMSProp as discussed in Section 3.6.3. It also applies the average of the first and second gradients moments (Kingma & Ba, 2015).

4.6.2.7 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 4.12, 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.

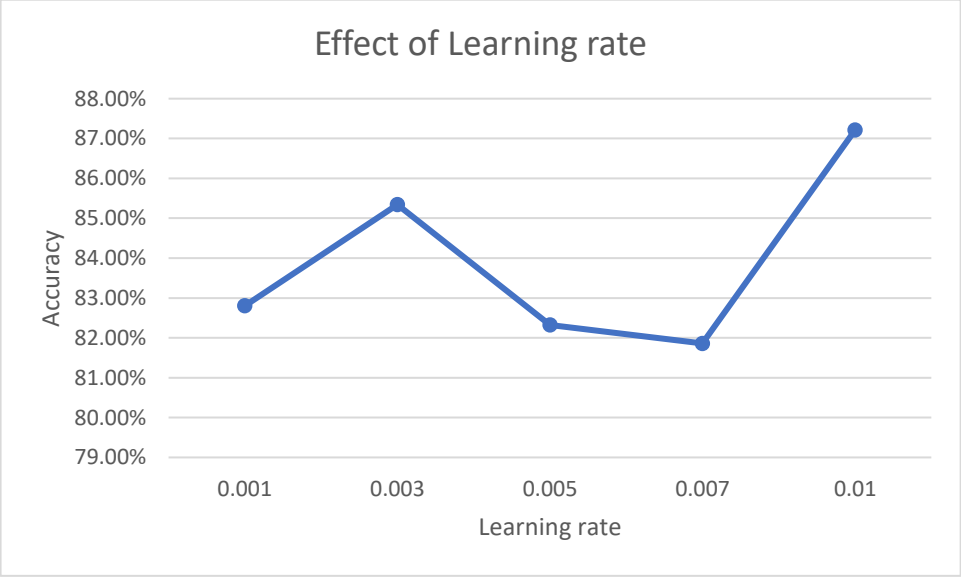

Figure 4.12: Effect of Learning rates

Table 4.19: Effect of Learning rates

Learning rate	Accuracy
0.001	82.80%
0.003	85.34%
0.005	82.32%
0.007	81.86%
0.01	**87.21%**

Figure 4.12 and Table 4.19 show that the best learning rate is 0.01 which achieves an accuracy of 87.21% during validation. It can be seen there was a gradual rise from a learning rate of 0.001 to 0.003, which registered a performance of 82.80% and 85.34% respectively. Then, the accuracy dropped to 81.86% at a learning rate of 0.007. The accuracy rose to its peak of 87.21%, which accounted for about 5% rise at a learning rate of 0.01. The learning rate is a hyperparameter that affects the adjustment of the weights of a network corresponding to the loss of gradient. It controls the convergence speed of the model to a local minimum (best accuracy). Lower learning rate means the speed of the model learning a set of data is slower which leads to a drop in performance as the time of convergence increases. However, the higher learning rate will cause the model to miss out on a lot of information, therefore, an optimum learning rate must be set to ensure the best performance of the model.

4.7 Chapter Summary

It is discovered in this second experiment 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. The effect of epochs can further be explored in future work using a high-end processing unit.

CHAPTER 5

METHOD 2: C3M1BL AS DEEP FEATURE EXTRACTOR WITH WEKA CLASSIFIERS

As we have seen the result of the proposed model (C4M4BL) and the effect of the hyperparameter configuration, a different methodology is discovered that has better accuracy in terms of activity classification. It involves turning the proposed model (Section 5.1) into a deep feature extractor so that the deep feature extracted can be used with different machine learning classifiers (Section 5.3). A revision of the proposed model must be done before the experiment can proceed.

5.1 Architecture of the Enhanced Proposed Model (C3M1BL)

An enhanced model of C4M4BL proposed in CHAPTER 4 is carried out, as the results are yet to reach a satisfactory level. The previous model consists of:

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

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.

Further analysis led us to conclude that although more convolution layers theoretically lead to high performances, it is not the case in our experiments, as the occurrence of overfitting in the network also increases. Furthermore, too many max-pooling has lead to fewer trainable weight to be passed to the subsequent layer. Thus, an enhanced model called CNN3-MAX1-BidirLSTM (C3M1BL) model is proposed as follow:

$$C(m,k,v) - C(m,k,v) - C(m,k,v)$$

$$-M(m) - B(L(m)) - D(m,v) - D(m,v)$$

In Figure 5.1, notice that in the new proposed model, three max-pooling layers and one convolutional layer have been removed. The new model is running using both UCI and WISDM datasets.

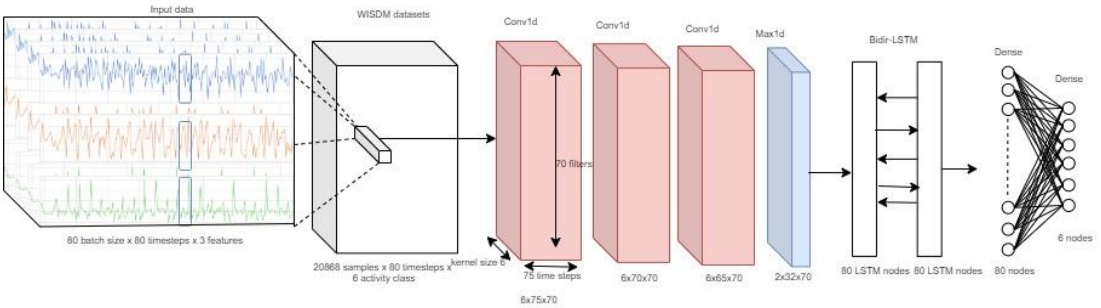

Figure 5.1: CNN3-MAX1-BidirLSTM (C3M1BL) model

The hyperparameter configuration of the proposed model has also been fine-tuned to reduce overfitting and improve utilization of the processing unit. The result is tabulated in Table 5.1. It can be seen that C3M1BL significantly achieves higher accuracy than the previous model of C4M4BL. For UCI datasets, the accuracy improves 0.68%, while the accuracy of WISDM improves 0.31%. The max-pooling layer is usually used to summarize the obtained representation from the convolutional layer by aggregating a group of encodings into one representation (A. Ignatov, 2018). However, the more summarization has led to less representation for the subsequent layer (Bidirectional LSTM) to compute, which leads to a loss of information (Murray & Perronnin, 2014).

Table 5.1: Comparison between C4M4BL and C3M1BL

	C4M4BL		C3M1BL	
Dataset	**UCI**	**WISDM**	**UCI**	**WISDM**
System Architecture	C(32,3)-MAX-C(32,3)-MAX-C(32.3)-MAX-C(32,3)-MAX-B(L(128))-DL(128)-DL(6)	C(220,3)-MAX-C(220,3)-MAX-C(220,3)-MAX-C(220,3)-MAX-B(L(80))-DL(80)-DL(6)	C(80,5)-C(80,5)-C(80,5)-MAX-B(L(50))-DL(40)-DL(6)	C(70,6)-C(70,6)-C(70,6)-MAX-B(L(20))-DL(40)-DL(6)
Accuracy	90.57%	87.62%	91.25%	87.93%

We will also look into the confusion matrix of the model. Table 5.2 shows the confusion matrix of the UCI dataset using C3M1BL. Comparing the previous configuration matrix of UCI dataset using C4M4BL in Table 4.8, there is a significant improvement in Walking Upstairs which achieves 94.69% as compared to 88.32% in C4M4BL. Standing also improves from 79.32% to 81.77%, with the C3M1BL. While Walking and other activities drop slightly. This can be improved with more epochs (training) but nevertheless, the overall accuracy has improved by 0.68% (91.25%) and thus we can use this model to proceed with our next objectives.

Table 5.2: Confusion Matrix of UCI dataset using C3M1BL

		Predicted Class						
		W	**WU**	**WD**	**Si**	**St**	**L**	**Recall**
Actual Class	**Walking**	486	8	2	0	0	0	97.98%
	W. Upstair	1	446	24	0	0	0	94.69%
	W. Downstair	3	2	415	0	0	0	98.81%
	Sitting	15	2	0	412	62	0	83.91%
	Standing	2	2	0	93	435	0	81.77%
	Laying	0	39	3	0	0	495	94.97%
	Precision	95.86 %	89.38 %	93.47 %	81.58 %	87.53 %	100.00 %	**91.25 %**

Table 5.3 shows the confusion matrix of the WISDM dataset using C3M1BL. We can see an overall improvement of both stationary and motion activity such as sitting which improves 6.86% from 91.15% to 98.01%, walking which improves 6.26% from 86.73% to 92.99%. There is a significant drop in the performance for both downstair and going upstairs, which we might attribute to the reason regarding the model confuse these two activities easily which we discussed earlier. However, it is not the case in this model, walking is also part of the reason, as they share the same motion acceleration pattern. Nevertheless, the overall performance of the model improves.

Table 5.3: Confusion Matrix of WISDM dataset using C3M1BL

		Predicted label						
		Down	**J**	**Si**	**St**	**Up**	**W**	**Recall**
Actual Label	**Downstair**	427	0	0	0	92	131	65.69%
	Jogging	37	1863	0	0	45	45	93.62%
	Sitting	1	0	443	4	4	0	98.01%
	Standing	2	0	42	324	2	0	87.57%

	Upstairs	77	0	0	2	503	143	69.38%
	Walking	154	0	0	0	14	2229	92.99%
	Precision	61.17%	100.00%	91.34%	98%	76.21%	87.48%	**87.93%**

5.2 C3M1BL as deep feature extractor

Throughout this research so far, we have only used Softmax as the classifier to classify both datasets' different activities. Softmax is arbitrary a simple classifier calculating probabilities of a set of data and cannot be used solely to justify the performance of C3M1BL. Therefore, we must consider different classifiers which may improve the classifier

To further explore different classifiers to improve our model performance, we remodeled our architecture to function as a deep feature extractor. This process is commonly known as transfer learning. Transfer learning typically utilizes features extracted from the trained model. There are two ways to do transfer learning.

- Feature Extraction from pre-trained model and then training a classifier on top of it.
- Fine-tuning the pre-trained model keeping learned weights as initial parameters.

In our case, we use the former. In this experiment, we extracted two sets of deep features of the pre-trained model, which is the Bidir-LSTM and Dense layer outputs using Keras for both UCI and WISDM dataset. The deep features extracted are converted to Comma-separated Value (CSV) files. The deep features are then further normalized by scaling the input vectors into unit norms. Although we already use Adam optimizer for normalization during training to improve optimization (Salimans & Kingma, 2016), a further normalization of the deep features can benefit the

performance of the subsequent machine learning algorithms. At the end of each unit norm, the true label of each sample is appended.

For the platform running different classifiers, we have chosen Waikato Environment for Knowledge Analysis (WEKA) as it has a large database of classification algorithms such as Logistic Regression, Naive Bayes, Decision Tree, k-Nearest Neighbors and Support Vector Machines. Moreover, WEKA is open source and free to download online. WEKA prefers to load data in Attribute-Relation File Format (ARFF) which is an extension of the CSV file format where a header that provides information about the metadata is used. Thus, we have converted our CSV files to the ARFF format. The process is summarized in Figure 5.2.

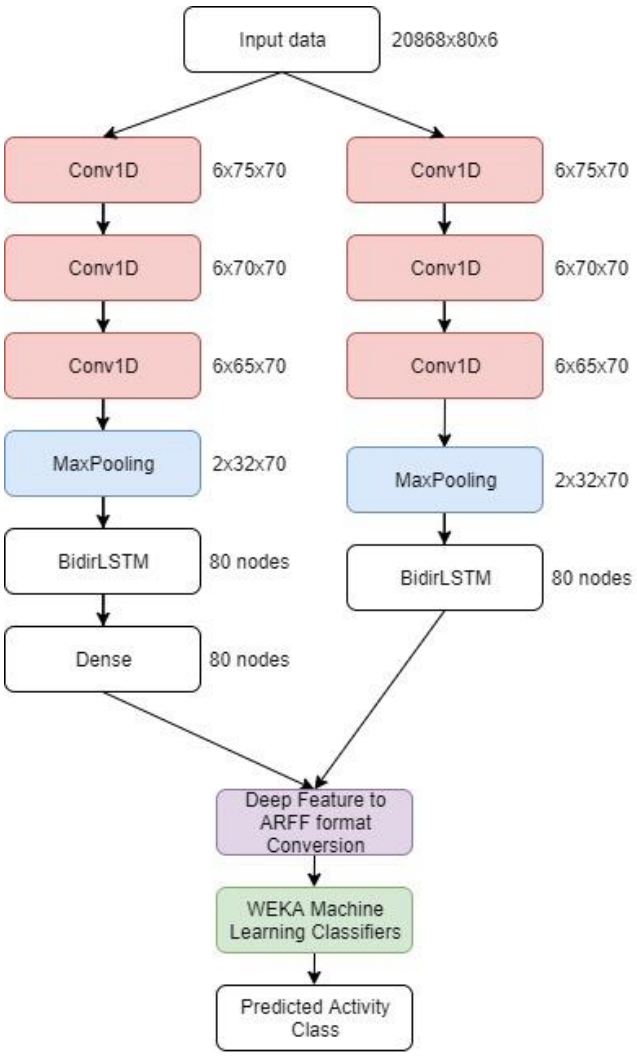

Figure 5.2: Deep Feature Extraction And Transfer Learning Process

5.3 WEKA Classifiers Performance using Deep Features

Next, a total of ten machine learning classifiers were chosen to evaluate which one can improve the performance of C3M1BL. WEKA allows us to specify our training datasets and test datasets. Thus, we first supplied the training dataset to WEKA and subsequently provides the test dataset.

The result is tabulated in Table 5.4. Overall, we can conclude that the result is better than using Softmax classifier in C3M1BL. For instance, in UCI datasets, J48 achieved the highest accuracy of 91.41% as compared to other classifiers using Dense outputs, while Bidir-LSTM outputs perform better in Multilayer Perceptron (MLP) achieving 91.86% accuracy. In WISDM datasets, Logistic Regression gained the best performance (89.40%) using Dense outputs, while Random Forest achieved 90.22% when using Bidir-LSTM outputs.

Table 5.4: Classifiers Result for both UCI and WISDM datasets

	UCI		**WISDM**	
Classifier	**Dense Output**	**Bidir-Lstm Output**	**Dense Output**	**Bidir-Lstm Output**
Logistic Regression	91.25%	90.30%	**89.40%**	90.19%
SVM	90.97%	91.25%	89.28%	89.90%
Random Forest	90.67%	90.53%	89.85%	**90.22%**
Naive Bayes	89.75%	91.31%	86.95%	85.21%
Multilayer Perceptron	91.01%	**91.86%**	88.84%	87.91%
kNN (Ibk) - distance Scaled Manhattan	89.04%	88.60%	87.73%	86.01%

kNN (Ibk) - distance Euclidean	88.94%	88.53%	87.24%	86.44%
Bayesian Network	91.01%	90.70%	88.02%	85.48%
J48	**91.41%**	88.70%	87.55%	85.97%

The results vary with different data supplied, thus each machine learning classifiers perform at their best, provided that the correct data is supplied. The best classifier (in bold) for each output of each dataset is selected and compared with the Softmax classifier which is used in the previous experiment. The result is tabulated in Table 5.5.

Table 5.5: Result Comparison between Different Classifiers

Activity	**UCI**			**Activity**	**WISDM**		
	Softmax	**J48**	**MLP**		**Softmax**	**Logistic Regression**	**Random Forest**
Walking	97.98%	97.78%	98.99%	**Downstair**	65.69%	65.54%	69.85%
W. Upstair	94.69%	92.14%	94.27%	**Jogging**	93.62%	95.28%	95.93%
W. Downstair	98.81%	98.81%	98.81%	**Sitting**	98.01%	85.18%	86.50%
Sitting	83.91%	81.26%	84.73%	**Standing**	87.57%	84.32%	87.03%
Standing	81.77%	84.59%	81.02%	**Upstairs**	69.38%	74.62%	75.03%
Laying	94.97%	94.97%	94.97%	**Walking**	92.99%	97.04%	96.79%
Precision	**91.25%**	**91.41%**	**91.86%**	**Precision**	**87.93%**	**89.40%**	**90.22%**

Table 5.5 shows the result comparison between different classifiers for each activity. For UCI datasets, Softmax performs better in predicting walking upstairs (94.69%) than J48 (92.14%) and MLP (94.27%), but J48 performs better in predicting standing (84.59%). MLP predicts better in walking (98.99%) and sitting (84.73%). Walking downstairs (98.81%) and laying (94.97%) have the same result for three classifiers.

For WISDM datasets, Softmax performs better in predicting stationary motion such as sitting (98.01%) and standing (87.57%), while Logistic Regression performs better in predicting walking (97.04%) only. Random Forest scores better in predicting motion activities such as going downstairs (69.85%), jogging (95.93%) and going upstairs (75.03%) than the other two classifiers. Overall, predicting using Bidir-LSTM outputs is better than using Dense outputs in both datasets.

5.4 Chapter Summary

In this chapter, we have enhanced our previous model (C4M4BL) to C3M1BL, and the overall result improves for both UCI and WISDM datasets, we also have a look into the confusion matrix and find that the overall the model did extremely well in predicting some of the motion activities, as well as stationary activities. Then, we also use the new model to extract two outputs from the Dense and Bidir-LSTM layers which we referred to as deep features. The extracted deep features are then supplied to WEKA to perform classification using nine different classifiers. J48 and MLP performs better using Dense and Bidir-LSTM output respectively in UCI datasets, while Logistics Regression and Random Forest perform better using Dense and Bidir-LSTM outputs respectively in WISDM dataset.

CHAPTER 6

CONCLUSION

In this paper, a summary of the modern approaches toward HAR 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.

The problem of handcrafted features that can be easily solved using the learning feature is also the main focus of this paper, especially using a deep learning feature model. Therefore, this research proposed an HAR model 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. The model is named C4M4BL. A comparison of this C4M4BL with other state-of-the-art classification methods shows that this network outperformed other previous results, achieving an accuracy of 90.57% with UCI dataset and an accuracy of 87.62% with WISDM dataset using minimum configuration of 30 epochs. An additional study is also conducted to see the effect of the different hyperparameters configuration on the performance of the model.

Next, we also remodeled C4M4BL to C3M1BL by removing some of the max-pooling layers and fine-tune it, which results in a better performance than C4M4BL. C3M1BL accounted for 91.25% and 87.93% using UCI and WISDM dataset respectively. Then, we use the model to function as a deep feature extractor and outputs two types of deep features from the Dense and Bidir-LSTM layer respectively. Both the deep features are then fed into WEKA which performs different classification using nine classifier algorithms. Overall, Bidir-LSTM layer outputs perform better than Dense layer outputs for both datasets. C3M1BL work best with MLP and Random Forest which accounted for 91.86% and 90.22% accuracy using UCI and WISDM dataset respectively.

In retrospect, for each experiment, the performance improves for both datasets. We also achieves all of the research objectives in this study including studying different time and frequency domain feature, designing an automated HAR utilizing the combination of CNN and Bidir-LSTM which proves that it is much better in extracting deeper representation than other proposed method using only a low epoch (30), we also have studied the effect of hyperparameter configuration on performance and create a deep feature extractor from the proposed model, and lastly study the different classifiers using deep feature extracted.

Future work will include experimenting with different methods of classifiers, finetuning the model to better optimize it for better results, and use a better workstation that is able to train the model using higher epochs (5000 epochs) in a shorter time span. Further study using a larger dataset should also be conducted.

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

Breiman, L. (2001). Random forests. *Machine Learning*. https://doi.org/10.1023/A:1010933404324

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. (1995a). Support-vector networks. *Machine Learning*. https://doi.org/10.1007/bf00994018

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

Cunningham, P., & Delany, S. J. (2007). K -Nearest Neighbour Classifiers. *Multiple Classifier Systems*. https://doi.org/10.1016/S0031-3203(00)00099-6

Danesi, M. (1995). Learning and teaching languages: the role of "conceptual fluency." *International Journal of Applied Linguistics*. https://doi.org/10.1111/j.1473-4192.1995.tb00069.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

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning*. https://doi.org/10.1002/9780470400531.eorms0099

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. (2018). Real-time human activity recognition from accelerometer data

using Convolutional Neural Networks. *Applied Soft Computing Journal*, *62*, 915–922. https://doi.org/10.1016/j.asoc.2017.09.027

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

Lewis, D. D. (1998). Naive(Bayes)at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science (Including Subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/bfb0026666

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

Lv, M., Xu, W., & Chen, T. (2019). A hybrid deep convolutional and recurrent neural network for complex activity recognition using multimodal sensors. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2019.06.051

Murray, N., & Perronnin, F. (2014). Generalized max pooling. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2473–2480. https://doi.org/10.1109/CVPR.2014.317

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

Salimans, T., & Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*.

Salzberg, S. L. (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning*. https://doi.org/10.1007/bf00993309

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

Scott, A. J., Hosmer, D. W., & Lemeshow, S. (1991). Applied Logistic Regression. *Biometrics*. https://doi.org/10.2307/2532419

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

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

MEETING DATE: 28/11/2019	**MEETING NO.: 1**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - discussed on the enhancement of Method 1 - planned on next phase FYP2
2. WORK TO BE DONE
3. PROBLEMS ENCOUNTERED
4. COMMENTS

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

NOTES:
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)

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 4/12/2019	**MEETING NO.: 2**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - Extract deep features in different layers of the model - Normalize deep features into unit vector and label each row with their respective activity - Study the input into Weka to test with different classifier
2. WORK TO BE DONE
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: 25/12/2019	**MEETING NO.: 3**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - remove additional CNN and Max pool layers - Extract deep features from dense layer and Bidir-LSTM layer and convert into csv file - Run Weka using csv file and test it with 10 different classifiers
2. WORK TO BE DONE
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)

Faculty of Information Science and Technology (FIST)

Final Year Project Meeting Log

MEETING DATE: 30/12/2019	**MEETING NO.: 4**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - convert deep feature into csv file and feed into Weka - run ten different machine learning classifiers
2. WORK TO BE DONE
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: 2/1/2020	**MEETING NO.: 5**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - Correct error in the previous experiment - Begin documentation of report
2. WORK TO BE DONE
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: 12/1/2020	**MEETING NO.: 6**
PROJECT ID: T681546	
PROJECT TITLE: HUMAN ACTIVITY RECOGNITION USING ACCELEROMETER DATA	
SESSION: Trimester 2 2019/2020	**SUPERVISOR: PANG YING HAN**
STUDENT ID & Name: 1161101857 CHEW YONG SHAN	**CO- SUPERVISOR: OOI SHI YIN**

All to be filled in by student

1. WORK DONE [Please write the details of the work done after the last meeting.] - Reporting and rerun experiment - Tabulate results from the WEKA classifiers
2. WORK TO BE DONE
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)

Appendix B: Checklist for FYP Final Report Submission

Faculty of Information Science and Technology (FIST)

Checklist for Final 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 Final Report		
2. Title Page of the Final Report		
3 Copyright page of I Final Report		
4. Declaration Page of Final 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: THE CONCEPT AND COMPONENTS USED IN DEEP LEARNING MODEL FOR HUMAN ACTIVITY RECOGNITION		
15. Chapter 4: METHOD 1: EVALUATION OF C4M4BL AND THE EFFECT OF HYPERPARAMETER CONFIGURATION		
16. Chapter 5: METHOD 2: C3M1BL AS DEEP FEATURE EXTRACTOR WITH WEKA CLASSIFIERS		
17. Chapter 6: CONCLUSION		
18. References – APA style		
19. Appendices		
20. CD/ DVD and envelope as shown in Appendix K		
21. 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. Spine Format		
6. Comb Bind (For evaluation)		
7. Permanent Bind (After Approval)		
6. Colour of the Front Cover		
7. Number of words > 10000 (Main content only)		

Checked by

_____

Student's Signature & Date

Appendix C: CD/DVD and Envelope