GAIT TRAJECTORY PREDICTION VIA EXTREME LEARNING MACHINE

LIM HAN LEONG, MPhil.

Thesis submitted to the University of Nottingham

for the degree of Master of Philosophy

FEB 2021

ABSTRACT

In this thesis, an algorithm to estimate the gait trajectory based upon the electromyography (EMG) signal is proposed. The algorithm is developed using Extreme Learning Machine (ELM). Experiments were conducted to acquire the gait parameters from 20 healthy human subjects. EMG signals from Tibialis Anterior (TA) and Gastrocnemius Lateral (GL) muscles were obtained during the gait cycle. Three stages were completed to ensure the viability of EMG signals and ELM used to predict the gait trajectory. The first stage comprises of using EMG signals to predict temporal gait parameters and benchmarked with a basic same architecture Artificial Neural Network (ANN). The target temporal gait parameters are gait speed and stance/swing phase which were measured using an inertia sensor and camera system. The ELM algorithm was developed using a single hidden layer feedforward network architecture where the weights from the input layer to the hidden layer were randomized and not updated during the run. Results obtained from ELM were compared with an ANN model with the same architecture as the ELM algorithm. In ELM, the mean estimation errors of gait speed, stance percentage, and swing percentage were 11.86%, 7.62%, and 6.07% respectively. This was compared to the errors of 12.92%, 11.75%, and 9.56% using ANN. Besides that, ELM achieved shorter training and testing time. The robustness of the ELM algorithm demonstrated the capability of real-time computation due to superior computing performance compared to conventional ANN models. The second stage superseded the first with a more advanced variant of ELM to classify the endpoints of EMG for each gait cycle during the whole gait duration. This was done with a Weighted ELM, and for 20 participants, it was able to achieve an average testing classification rate of 99.38%. In the last part, a boosted Ensemble ELM was proposed to predict the x trajectory of the gait using features of EMG signals. The best configuration of 8 ELMs boosted is shown to produce an average RMSE of 0.1055m when compared to an un-boosted version of 0.1344m.

STATEMENT OF ORIGINALITY

I hereby declare that the submission of this thesis is the product of my work with the combined assistance from my supervisor and co-supervisors in the project's design and conception as well as the presentation of this thesis. To the best of my knowledge, it contains no materials previously published or written by another person, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, or with whom I have worked at the University of Nottingham Malaysia or elsewhere, is also explicitly acknowledged in the thesis.

Signed: 

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviations

EMG	Electromyography
ANN	Artificial Neural Network
ELM	Extreme Learning Machine
SLFN	Single-hidden layer feed-forward neural network
SCG	Scaled Conjugate Gradient algorithm
RMSE	Root Mean Squared Error
WGN	White Gaussian Noise
RNN	Recurrent Neural Network
LM	Levenberg-Marquardt algorithm
BR	Bayesian Regularization algorithm
FFNN	Feed-forward Neural Network
WPD	Wavelet Packet Decomposition
RBF-ELM	Radial Basis Function Extreme Learning Machine
AW-ELM	Adaptive Wavelet Extreme Learning Machine
QPC	Quadratic Phase Coupling
TA	Tibialis Anterior muscle
GL	Gastrocnemius Lateral muscle
CMR	Common Mode Rejection
FIR	Finite Impulse Response

Symbols

a	Neural network output
w	Input weight
b	Bias
f	Transfer function
$\{p_k\}$	Set of input vectors
H	Hessian Matrix
β_k	Hidden-output weights
L	Sigmoid activation function
Y_n	Pure lin activation function
GC_i	represents the gait cycle time for the i-th gait cycle
H_i	s the heel-strike moment (initial contact ground of heel)
T_i	is the toe-off moment (terminal of toe leaving from ground)
ST %	is the stance phase percentage
SW %	is the swing phase percentage
N	Number of elements

CHAPTER 1 - INTRODUCTION

1.1 Overview

Human gaits describe patterns in which humans walk. They are dependent on various factors such as walking speed, forces, and other personal factors (e.g. age, joint impairment, etc.). Observation and analysis of these gaits provide a vital aspect on the diagnosis of gait impairment so as to devise a proper treatment plan for the patient (e.g. gait rehabilitation after stroke). Gait profile and parameters such as gait velocity, cycle time, stride length, etc., could be acquired via camera or inertia sensor systems. Meanwhile, electromyography signals record electrical activities of the muscle as active muscles produce electrical current and this can be correlated to the level of muscle strength.

In recent years, doctors conduct objective assessment utilizing machine learning technologies and this has been applied to gait analysis [1]. The artificial neural network could distinguish normal and impaired gait due to its strong nonlinear learning ability [2]. However, one of the problems ANN faces is that it could either reach a global minimum or get stuck at a local minimum. The accuracy of the ANN classifier could also decrease through the over-trained of training samples. In addition, to apply the techniques of ANN techniques on real-time gait analysis is undesirable as it requires a huge gait database, and the process of getting a huge database would be time-consuming. As a result, the system would become more complex with a greater computational burden and longer training time.

The emergence of machine learning techniques such as extreme learning machine has become a popular area of research over the past years. It is a learning algorithm for single-hidden layer feed-forward neural networks (SLFNs) where the weights connecting inputs to hidden nodes are randomly chosen whereas the output weights of SLFNs are

determined through iterations [3]. ELM has been found to overcome some of the bottlenecks faced by the classical neural network and ever since then, it has been applied in several unique problem-solving applications areas, including engineering, biomedical, and forensic science [4-10], often with promising results.

1.2 Aim and Objectives

The overall aim of this project is to derive real-time gait trajectory using EMG signals and this report consists of the methods to achieve this and the overall result of this project. This project investigates different variants of ELM in terms of their training speed, the accuracy from the predicted output as well as the generalization on unknown data in order to make a more reliable and accurate decision for real-time prediction. Results in the first stage of the project are used to validate the viability of using ELM for prediction and it indicated that ELM could perform function approximation in a faster and more efficient way than conventional ANN. In this study, ELM yielded a lower root mean squared error (RMSE), shorter training time, and lower mean estimation error compared to conventional ANN which triggered the next stage. The second stage is to ease the process of gait segmentation using a variant of ELM known as Weighted ELM to segment each gait cycle of EMG signal for the whole gait duration. In this study, the results are shown to have a high classification rate of correct endpoint segmentation. The final stage confirms the preceding two stages through a high accuracy prediction of the x-axis trajectory with a boosted Ensemble ELM.

The current technology of measuring the gait profile relies on expensive 3D imaging methods to provide accurate gait profile measurements. Another alternative is to use inertial sensors to acquire gait trajectory through integration method. Coupled with other sensor systems such as electromyography sensor and contact transducer, the whole wearable gait sensor network could be very bulky and uncomfortable to use. Positive

impacts could be expected from the output of this research project as there is no need to use visual or inertial sensors to acquire motion data in the future.

1.3 Outline of this report

Chapter 2 is an introductory chapter that introduces general surface electromyography. It also discusses its application in gait and the major drawbacks of using EMG as pattern recognition and random noises are one of them. To reduce such noise, a few methods such as band-pass filter, band-stop filter, or the use of well electrode have been explored. Moreover, the chapter also examines a few feature extraction techniques as it is able to highlight the relevant attributes in EMG signal and the rejection of noise using filters and de-noising algorithms.

Moving on, Chapter 3 presents the basic background of Artificial Neural Network and its architecture before furthering into a variant of ANN that is of different algorithm known as ELM. The chapter starts by introducing how ANN is inspired by its biological counterpart as its features show a decent similarity with a biological neural cell, which then proceeds to explaining backpropagation as a method to train ANNs as it is a technique to minimize the error between the actual target and the predicted outputs. The ground-breaking algorithm, Extreme Learning Machine, is explored in-depth in this chapter presenting an overview of ELM on its origin, how ELM works as well as discussing other variants of ELM.

The purpose of this study is to investigate the characteristics of gait patterns and derive the correlation of EMG data (muscle activity data) with gait patterns. Chapter 4 describes the methodology to obtain, process, and train the data. It is divided into 2 types of methodology as the first type of experiment is geared towards the feasibility of the algorithm while the second and third type leans more towards real-time processing. The difference between the two methodologies is in the capturing of EMG signal and gait trajectory as well as the software used to process the data. Hence, it establishes by

showing how gait parameters and trajectories are obtained through symmetrical walking experiments while the obtaining of EMG data is exhibited in the succeeding section. Further section explains the processing needed to achieve readable signals for the ANN and ELM to train on.

Chapter 5 analyses and discusses the results obtained from the 3 types of experiment conducted. Results indicated that ELM could perform function approximation in a faster and more efficient way than conventional ANN. In this study, ELM yielded a lower root mean squared error, shorter training time and lower mean estimation error compared to conventional ANN while the second experiment proved that the ELM has the ability to classify endpoints to segment EMG signals with high accuracy. The proposed method of boosted Ensemble ELM was able to achieve a low RMSE and fast training speed for predicting x-trajectory of gait cycle.

The last chapter concludes the project with a short discussion on further improvements.

CHAPTER 2 – LITERATURE REVIEW

2.1 EMG background and related work

EMG signal is one of the electrophysiological signals, which has been thoroughly worked on and used in many clinical and engineering applications. In this study, the application of surface Electromyography signal in gait is paid attention. Other applications include the control of prostheses or other assistive devices [11-12]. Nonetheless, there is a major drawback of using EMG in pattern recognition as it could be corrupted by random noise. These noises are usually caused by artifact and interference in recorded EMG signal, which can be classified into electrode noise, electrode and cable motion artifact, and alternating current power line interference [13-14]. They can be removed by applying filtering methods such as band-pass filter, band-stop filter, or the use of well electrode and instrument [13-14] but it is difficult to remove the interferences of random noise that fall in EMG dominant frequency energy using previous procedures. Generally, the random noise in EMG signal analysis is represented by white Gaussian noise (WGN) [15-16] which leads to adaptive filter and wavelet de-noising algorithm receiving significant attention in their ability to remove WGN [17-18]. Nonetheless, it is impossible to remove WGN absolutely, and sometimes an important part of EMG signals could be removed along with noise when using an adaptive filter or wavelet de-noising algorithm. The main difficulty of removing such noise is due to the random frequency characteristic of the noise as well as the larger amplitude of noise signal when compared to the amplitude of EMG signal [19]. In order to perform EMG pattern recognition and classification, EMG signal has to be pre-processed the spectral frequency component of the signal and has its features extracted after pre-processing [11]. Furthermore, during the pre-processing of EMG signal, the procedures to reduce noise can improve some parts of the spectral component [13] while the function of feature extraction is to highlight the relevant attributes in EMG signal and rejecting the noise [15]. The pre-processing and feature extraction methods can lead to successful EMG pattern recognition when distinct features that represent raw EMG signal for classification are selected.

2.2 Feature selection

Time-domain features have been widely used in research and these features are generally based on signal amplitude. As these signal amplitudes are considered as time-varying standard deviation of the signal, the computed features would provide information about waveform energy, amplitude, and frequency. The features described below are Time Domain features:

a) Mean absolute value

This feature is an average over all the absolute values of the signal collected during the epoch.

$$y = \frac{1}{N}\sum_{i=1}^{N}|x_i| \tag{1}$$

b) Root mean square

Taken as the square root of the sum of all samples to the power of two, this feature is also a sort of average of the signal during the epoch.

$$y = \frac{1}{N}\sqrt{\sum_{i=1}^{N}x^2} \tag{2}$$

c) Slope sign change

This feature counts the number of times the sign of the slope of the signal change.

$$y = \sum_{i=1}^{N} f(x) \tag{3}$$

$$f(x) = \begin{cases} 1, IF\ (x_i < x_{i+1}\ AND\ x_i < x_{i-1})\ OR\ (x_i > x_{i+1}\ AND\ x_i > x_{i-1}) \\ 0, otherwise \end{cases}$$

d) Waveform length

The waveform length is a measure of the variation of the EMG signal.

$$y = \sum_{i=1}^{N-1} (|x_{i+1} - x_i|) \tag{4}$$

e) Zero crossings

This feature measures the number of times the signal crosses zero. To avoid interference from noise, a zero crossing is only registered if the difference between the two samples exceeds a pre-defined threshold.

$$y = \sum_{i_1}^{N-1} f(x) \tag{5}$$

$$f(x) = \begin{cases} 1, IF\ (x_i > 0\ AND\ x_{i+1} < 0)\ OR\ (x_i < 0\ AND\ x_{i+1} > 0) \\ 0, otherwise \end{cases}$$

f) Wilson Amplitude

This feature counts the number of times the signal does a jump which exceeds a threshold. The threshold was chosen manually from inspection to give a signal when a muscle contracts and none when the muscle is at rest.

$$y = \sum_{i=1}^{N-1} f(|x_i - x_{i+1}|) \tag{6}$$

$$f(x) = \begin{cases} 1, IF\ x \geq threshold \\ 0, otherwise \end{cases}$$

g) Variance

The variance of the signal gives a measure of how much the samples differ from each other.

$$y = \frac{1}{N-1} \sum_{i=1}^{N} x^2 \tag{7}$$

2.3 Non-linear analysis of EMG

2.3.1 Introduction

EMG signals are considered as time series but it is a critical issue to determine if the observed signals are purely stochastic, deterministic nonlinear, or chaotic. The intrinsic properties can usually be distinguished by looking at the nonlinear deterministic dynamics and noisy dynamics of a time series. It is known that nonlinearity in the signal is introduced by interference and muscle cross-talk [20], but methods for non-linear time series analysis have not been used widely with EMG. Among the few existing methods are phase-space reconstruction, chaotic characterization, and non-linear prediction.

Wavelet-based denoising is usually performed on the signal to minimize noise as it has been found to be effective in denoising different physiological signals [21]. It is preferred over simple frequency domain filtering due to its function to preserve signal characteristics when minimizing noise with the number of thresholding strategies available. This allows reconstruction based on selected detail coefficients [22]. One of the few most commonly used wavelets to denoise bio signals like EMG includes the orthogonal Meyer wavelet as well as the Daubechies wavelets '$db2$' and '$db6$'. Nonetheless, it is advisable to choose wavelets that have similar shapes with those of motor unit action potentials [21]. With the noise minimized, tests for non-linearity and stationarity are done for signal qualification as this step is necessary for chaotic characterization, which is later performed by the calculation of invariants such as D_2, λ_i and D_{KY}.

2.3.2 Test for non-linearity

Some well-defined null hypotheses such as those that are generated by a Gaussian linear stochastic process are specified for most non-linearity tests and statistics are computer to test against this hypothesis [23]. As the distribution of the statistic is not known, Theiler *et al.* suggested methods for estimation using surrogate data [24]. Below are examples of two methods:

(i) Phase randomized

This can be achieved by randomizing the phases of Fourier transform of the signal and then inverting it to get the time domain signal. Thus, the surrogate would have an identical mean, variance, spectrum, and autocorrelation as the original. The histogram amplitudes of the surrogate would have a different distribution when compared with the original.

(ii) Gaussian scaled conjugates

A Gaussian distributed random time series is usually shuffled with the experimental data ranked in order and the phase is randomized while the amplitudes of the original data are ranked in order and substituted. The histogram amplitudes are unchanged as it is only a shuffle of the original data.

There are a number of test statistics that can be used to discriminate statistic and time asymmetry or test for time reversibility is one of the few simple and yet powerful discriminating statistics method [22]-[24]. When the probabilistic properties of a time series are invariant with respect to time reversal, the time series can be considered to be reversible and reversibility indicates linearity. Though it is not a definite indicator for linearity but irreversibility expresses the presence of non-linearity. The statistic can be calculated for the original data set with n surrogates as follow:

$$T_{rev}(\{x_n, \tau) = \frac{< (x_n - x_{n-\tau})^3 >}{< (x_n - x_{n-\tau})^2 >^{\frac{3}{2}}} \quad (8)$$

The test was implemented by finding the values of the test statistic for the original dataset and its n generated surrogates. The mean μ_{sur} and standard deviation σ_{sur} were calculated for the distribution. The significance of the distribution is given by the difference between the TR value of the original and μ_{sur} of the distribution, in units of σ_{sur}. In order to reject the null hypothesis on a 95% confidence level, the difference should be at least

$2\sigma_{sur}$. If the distribution is non-Gaussian, at least $5\sigma_{sur}$ to $6\sigma_{sur}$ is needed to reject the hypothesis [23].

2.3.3 Tests for Stationarity

Similar to test for non-linearity, there are several methods that exist to test for stationarity with the most common tests used in practice are the runs test and the reverse arrangement test. The latter proved to be more powerful in detecting non-stationarity [25].

The runs test works by classifying data into 2 mutually exclusive categories identified by a plus (+) for values greater than the mean and a minus (-) for values less than the mean. They are then grouped into a series of runs. A run is defined as a sequence of identical observations that is followed or preceded by a different observation or none at all. The total number of runs, r, calculated is used to compute a suitable level of significance, A. If r is such that $r_{\frac{N}{2};1-\frac{\alpha}{2}} < r < r_{\frac{N}{2};\frac{\alpha}{2}}$ where N is the total number of samples, then the signal is said to be stationary [25].

The reverse arrangements test works by the following procedure: For a random variable x_i where $i = 1,2,\dots N$, count the number of times $x_i > x_j$ for $i < j$ and 0. Every such inequality is known as reverse arrangement. Let $h_{ij} = 1\ IF\ x_i > x_j$ and 0 otherwise. Then,

$$A = \sum_{i=1}^{N-1} A_i \ where\ A_i = \sum_{j=i+1}^{N} h_{ij} \tag{9}$$

At a suitable significance level, if A is such that

$$A_{N;1-\frac{\alpha}{2}} < A < A_{N;\frac{\alpha}{2}}$$

then the signal is stationary.

2.3.4 Phase Space Reconstruction

Phase Space Reconstruction is generally the first step of chaotic time series analysis and delayed coordinate embedding is used to represent a signal in the phase space domain [26]. Each observation in the signal $x(n)$ is substituted with a vector:

$$y(i) = [x(i), x(i+\tau), x(i+2\tau), \dots, x(i+(M-1)\tau)] \quad (10)$$

where *M* is the embedding dimension and τ is the delay. There are several methods that can be used to find the optimal values for the embedding dimension as well as the delay. They are false nearest neighbors (FNN) for embedding dimension and average mutual information (AMI) for delay. For a given dimension in the phase space, the FNN algorithm functions by finding the nearest neighbor of every point in that phase space and then checking to see if the points are still close neighbors in one higher dimension. If the distance between the 2 points in the higher dimension divided by the distance in the present dimension exceeds a given heuristic threshold, the point is marked as a false nearest neighbor. Once the appropriate embedding dimension has been achieved, the percentage of false neighbors should reduce to zero.

A way to choose an optimal time delay is one that, given the state of the system, gives the maximum amount of new information with each measurement $x(i+\tau)$ and the value of delay is chosen by plotting a graph of mutual information $I(\tau)$ between $x(i)$ and $x(i+\tau)$ and finding the first value of τ for which $I(\tau)$ is a minimum [27]. When the phase-space representation of the signal has been obtained, the vectors in the reconstructed space can be shown in a recurrence plot to identify if they are close and far from each other by calculating the Euclidean distance between all pairs of vectors. The recurrence plot is a color-mapped representation of a 2D array where each element (i,j) represents the Euclidean distance between the vectors $y(i)$ and $y(j)$. [28].

2.3.5 Chaotic Characterization

When a system shows sensitive dependence on initial conditions, which manifests an unpredictable long-term behavior in a deterministic dynamical system, the system can

be said to be chaotic. This behavior is usually characterized by the exponential divergence of nearby trajectories in the phase space and such chaotic behavior can be found in biological signals such as ECG and EMG [29]. There are ways to classify if the certain system is chaotic and that is through a calculation of a number of characteristic invariants such as correlation dimension (D_2), Lyapunov exponents (λ_i) and Kaplan-Yorke dimension (D_{KY}). Chaotic attractors often have a non-simple, highly fractured structure, resulting in non-integer dimensions in the phase-space called fractal dimensions. These dimensions can be estimated using [30]:

$$D_q = \lim_{r \to 0} \frac{\log[C(q,r)]}{(q-1)\log[r]} \tag{11}$$

where $C(q,r)$ is the correlation function of the attractor, i.e. a measure of the probability that two points on the attractor are separated by a distance of r. When q is 2, the resulting dimension is the correlation dimension, D_2. For small r, the function $log[C(q,r)]$ is approximately linear in $log[r]$ and thus, D_2 is simply given by the slope of the log-log curve. Multiple methods exist to estimate the correlation dimension [31]. When the correlation dimension, D_2, is calculated to be fractal and converges with increasing embedding dimension, the system can be said to be chaotic. Lyapunov exponents (λ_i) are a measure of the sensitivity to initial conditions, which causes nearby trajectories to converge or diverge exponentially with respect to time. The positive exponents show the divergence while the negative exponents show the convergence of the trajectories. The phase-space evolution forms an ellipsoid shape for short time and length scales, will have some of its directions stretched and others contracted. For an ellipsoid with an initial fiducial volume of $r(0)$ and where the length of the i th principal axis at time t is given by $li(t)$, the i th Lyapunov exponent is defined as [16]:

$$\lambda_{\mathrm{i}} = \lim_{t \to \infty} \frac{1}{t} \log\left(\frac{l_i(t)}{r(0)}\right) \tag{12}$$

The set of all the exponents is called the Lyapunov spectrum and a chaotic system has at least one positive Lyapunov exponent while the sum of all exponents should be negative.

The KY-dimension can be derived from the λ_i values as follows. If there are n Lyapunov exponents, then:

$$D_{KY} = K + \frac{\sum_{a=1}^{K} \lambda_a}{|\lambda_{K+1}|} \tag{13}$$

where the λ_i (up to λ_K) are positive, arranged in descending order of magnitude, their sum is greater than 0 and

$$\sum_{a=1}^{K+1} \lambda_a < 0 \tag{14}$$

D_{KY} is generally close to D_2 [30].

2.4 Artificial Neural Network

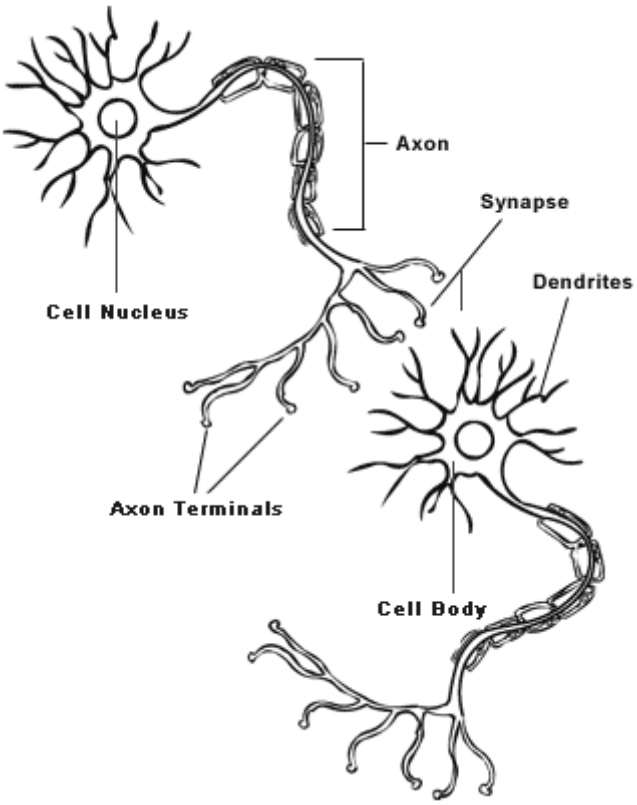

Figure 1: Biological neural cell [32]

Artificial neural networks are inspired by their biological counterparts. It can be seen from Figure 1 that a neuron has three main components: the axon, the cell body, and the dendrites. The dendrites, that resemble tree-like connections, carry electrical signals into the cell body, where the signals are processed, which later then sent out to other neurons via the axon. The synapse is the point of contact between the axon of a cell and the dendrite of another cell. Though it can be said that biological neural networks are much slower than electrical circuits, they are in fact able to perform much faster in many tasks because of the massive parallel structure of the biological neural networks, at which all the neurons fire at the same time [32].

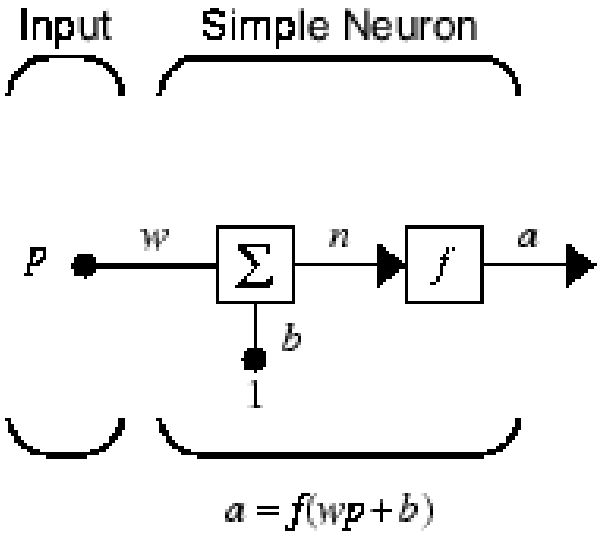

Figure 2: Single-Input neuron model [32]. The output of this model is $a = f(wp + b)$. Where p is the input, w is the weight, b corresponds to the bias, and, is the transfer function by which output of the neuron is derived.

Figure 2 shows the simplest structure of an artificial neuron, the single-input neuron model. A scalar input is multiplied by its weight, w, and summed with a bias, b. The resultant is then fed to a transfer function, f, to produce the output. Examples of transfer functions are sigmoid, hard limit, and linear. Relating this structure to the biological neuron discussed earlier, the input corresponds to the dendrites, the weight to the strength of the synapse, the summation and the transfer function to the cell body, and the output, a, corresponds to the axon of the cell.

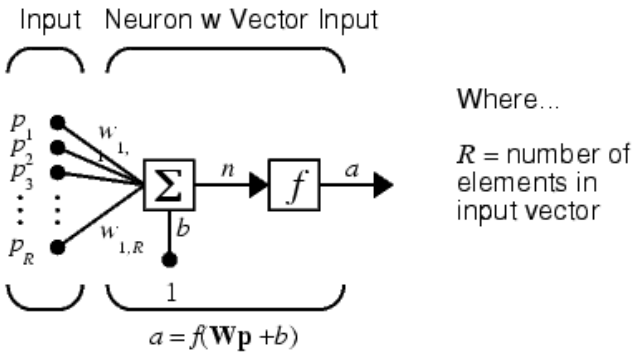

Figure 3: Multiple-input neuron where the input is p, and the weights, w and the output of this model is $a = f(\sum w_R p_R + b)$ [32].

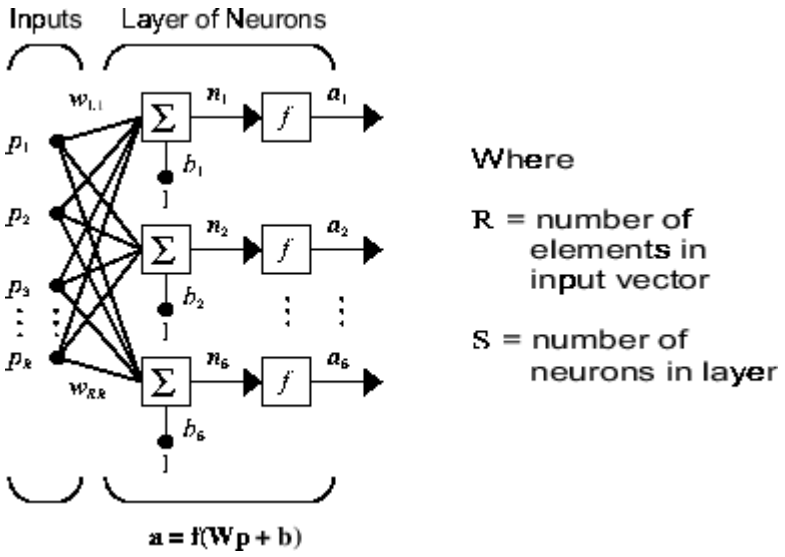

Figure 4: A layer of neurons each with multiple inputs and one output [32]. The output of this model, $a = f(\sum w_R p_R + b_s)$

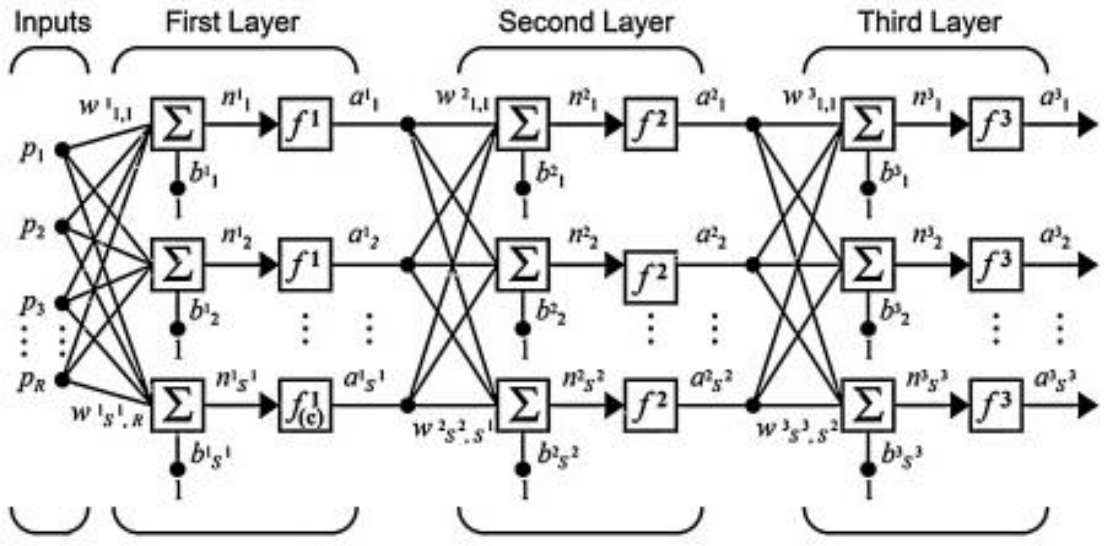

Figure 5: Multiple-layers-of-neurons model. Here only three layers are shown. The number, however, can be extended to whatever number of layers is desired [32].

Generally speaking, more complex neural structures are more capable. A single input neuron can be extended to a multiple-input neuron which can be seen in Figure 3 where the input and the weights become vectors instead of scalars. As shown in Figure 4, a multiple-input neuron also can be extended to a single layer of neurons—with multiple inputs. Moreover, a single layer can be extended to multiple layers as observed in Figure 5. The complexity of the artificial neural network can be built up as desired to achieve

more complicated tasks. It is, however, worth mentioning that too complex structures can be troublesome as larger ANNs usually lead to learning of irrelevant statistical fluctuations in the training data which causes poor generalization on test data [33].

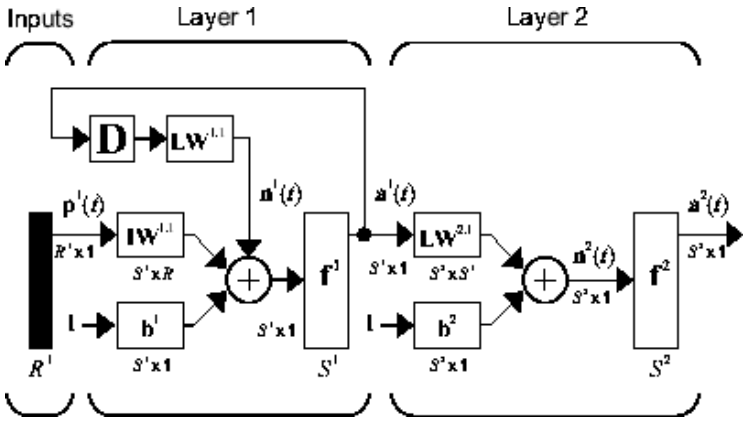

Figure 6: An example of Recurrent ANN model [32]. The delay block takes inputs as initial conditions and shift its input by one which determines the output.

There are two major types of neural networks: feedforward and recurrent. All the examples observed here are feedforward, where the input is piped all the way till the end, i.e. it does not get reused at any stage of the process. Recurrent neural network (RNN), by contrast, are ones with feedback, where some of the outputs get connected to the input throughout the process. An example of an RNN is shown in Figure 6. The output of the delay block is the input signal delayed by one-time step. This specific arrangement of a recurrent neuron work such that the future outputs are computed from past values with inputs being the initial condition. RNNs are more powerful than feedforward NNs as they possess the ability to exhibit temporal behavior. In another word, feedforward NN can only implement static input-output mapping, while RNN's can deal with dynamical systems, where time is involved.

Artificial Neural Networks (ANNs) is one very important topic, to which analysis of the biological human control system has led. One can think of ANNs as small artificial brains built to learn and accomplish desired tasks.

2.5 Backpropagation

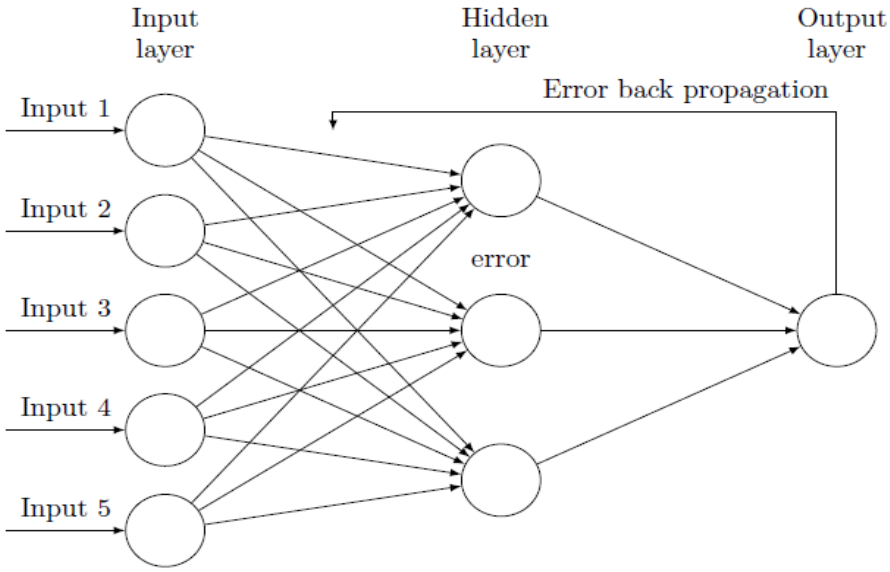

Figure 7: A structure of Backpropagation on ANN

Backpropagation is an abbreviation of "Backward Propagation of Error" [34]. It is a method used to train ANNs. Because the error (difference between predicted and target outputs) is eventually minimized, the algorithm is usually conjoined with an optimization technique. Backpropagation training involves two phases: the feedforward and the feedback propagation (See Figure 7) [35]. During the first phase, the teacher-input (a vector in most cases) is fed to a network built with random weights. The output is computed using the values produced by the hidden layer, while the weights of each neuron remain unchanged. In the second phase, the error is computed by subtracting the network predicted output from the teacher-input (target training signal). The error (collected in a vector) is then back-propagated through the hidden layer(s) and minimized using techniques that result in updating the weights of the network. This procedure is repeated until a terminating condition is met (i.e. satisfactory result is achieved). An example of a termination condition would be achieving a value of a mean square error that is smaller than a preset threshold. Figure 7 shows the structure of the backpropagation training method. As explained in the Introduction, this study uses the backpropagation algorithm to train the ANN. The difference between the training approaches is the optimization

technique. Levenberg-Mardquardt (LM), Bayesian Regularization (BR), and Scaled Conjugate Gradient (SCG) are nothing but different methods to minimize the error vector(s). Next, the algorithm of SCG is explained as it is used in the training of this study's neural network.

2.6 Scaled Conjugate Gradient

The Scaled Conjugate Gradient (SCG) algorithm is developed with the claim that large-scale problems are best handled by a group of optimization techniques called the "conjugate gradient methods" [36]-[38]. The scaled conjugate gradient algorithm is similar to the LM method in the sense that it avoids the calculation of the Hessian matrix (second-order derivative) [32] as the calculation of the Hessian matrix can be very complicated and may abuse the computation power. If the gradient (first derivative) has n elements, the Hessian (second derivative) has n^2 elements [32]. The conjugate gradient method provides quadratic termination without computing the Hessian. The conjugate direction is one way of guaranteeing quadratic termination. It can be shown that if a sequence of exact linear searches is made along with any set of conjugate direction, the minimum of any quadratic function with n parameters can be found in a maximum of n searches [39], [40]. The solution, therefore, is to find a way to force the search in the conjugate direction(s). A set of vectors $\{p_k\}$ is mutually conjugate with respect to a positive definite Hessian Matrix, H, if and only if:

$$p_k^T H p_j = 0 \qquad k \neq j \tag{15}$$

Similar to orthogonal vectors, there is an infinite number of conjugate vectors spanning an n-dimensional space. This, however, includes the hessian matrix. Therefore, the conjugacy condition in (1) must be stated without H. Recall that for quadratic functions:

$$\nabla f(x) = Hx + d \tag{16}$$

$$\nabla^2 f(x) = H \tag{17}$$

The combination of these formulas presents the change in gradient at iteration $k + 1$:

$$\Delta g_k = g_{k+1} - g_k = (Hx_{k+1} + d) - (Hx_k + d) = H\Delta xk \tag{18}$$

$$x_k = (x_{k+1} - x_k) = \alpha_k p_k \tag{19}$$

where α_k is chosen to minimize $f(x)$ in the direction p_k. (5) Now the conjugacy condition can be restated as:

$$\alpha_k p_k^T H p_j = \Delta x_k^T H p_j = \Delta g_k^T p_j = 0 \qquad k \neq j \tag{20}$$

(6) restates the conjugacy condition in terms of change in the gradient at successive iterations. In words, the search directions will be conjugate if they are orthogonal to the change in gradient. It is worth to mention that p_0 is arbitrary and p_1 can be any vector that is orthogonal to Δg_0. Therefore, there is an infinite number of conjugate vectors. It is common to start in the steepest descent direction:

$$p_0 = -g_0 \tag{21}$$

Next, at each iteration, a vector p_k, that is orthogonal to $\{\Delta g_0, \Delta g_1, \dots \dots, \Delta g_{k-1}\}$. This procedure can be simplified to iterations of the form [40]:

$$p_k = -g_k + \beta_k p_{k-1} \tag{22}$$

β_k is a scaler that can be chosen in different methods. All of which produce equivalent results when it comes to quadratic functions [40]. The most common methods are [40]:

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{\Delta g_{k-1}^T p_{k-1}} \tag{23}$$

$$\beta_k = \frac{(\Delta g_k^T g_k)}{\Delta g_{k-1}^T g_{k-1}} \tag{24}$$

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{\Delta g_{k-1}^T g_{k-1}} \tag{25}$$

2.7 Extreme Learning Machine

The Extreme Learning Machine has been proposed as a learning algorithm for single-hidden layer feed-forward neural networks. In ELM, the hidden nodes are randomly initiated and never updated. The only parameters to be updated are the weights between the hidden layer and the output layer. Compare to the conventional feed-forward neural network (FFNN), ELMs are known to shorten the learning time, better generalization performance, and ease of implementation. In general, the learning rate of FFNN is relatively longer and this has become the bottleneck in their applications. According to [3], the reasons for FFNN having longer learning rate are firstly because it uses slow gradient-based learning algorithms to train the network. Secondly, the parameters must be iteratively tuned. The recently proposed ELM for SLFN by [3, 41-42] was meant to overcome these problems as it is a simple tuning-free algorithm to achieve fast learning speed and provide the best generalization performance [42]. Therefore, ELM has great potential for developing an efficient and accurate model as a real-time predictor for applications that require better generalization ability.

Nonetheless, ELM learning algorithm is different from the classical gradient-based learning algorithms as it tends to reach the minimum training error while considering the magnitude of weights [5] and that it can be used to train SLFN with non-differentiable activation functions [41]. ELM has been widely exploited in EMG pattern recognition due to its fast learning speed and better generalization, which are summarised in Table 1. [42] has investigated the prediction of handgrip force based on the RMS feature of EMG signals using ELM and has stated that ELM possesses a fast learning speed and good generalization performance. Moreover, from the results obtained in [42], ELM is shown to produce relatively good accuracy of mean RMSE 1.165 ± 0.475 while producing an appropriate correlation coefficient within a short duration. While not many have used features from frequency domain, [43] uses the energies of bases at the fourth level of wavelet packet decomposition (WPD) of EMG as a feature to train the ELM classifier. The work [43] explores the diagnosis of obstructive sleep syndrome and the periodic limb movement

syndrome and in the end, Necmettin managed to achieve a classification accuracy of 96.85%. The advancement of ELM has led to many ELM variants being established. [44] evaluated the performance of radial basis function ELM (RBF-ELM) and basic sigmoid ELM on the classification of individual and combined finger movements using EMG. Furthermore, [44] also evaluated the various feature combinations as well as investigated feature projections to improve the class separability of the features. The features used are summarised in Table 1 and it was found from the paper that basic sigmoid ELM takes a lesser training time when compared to RBF-ELM but it does not fare as well in the accuracy section when compare with RBF-ELM. Sigmoid could only identify 92.99% of the finger movements correctly whereas RBF-ELM could identify 99.5% correctly [44]. [45] presented a paper on the comparison of the classification of sick and healthy muscles on the legs using ELM and Support Vector Machine and has resulted in ELM achieving higher accuracy than SVM. Their study was achieved with only using two muscles namely the Tibialis Anterior and Gastrocnemius muscle and multiple time-domain features from the EMG signals obtained [45]. Moreover [46] also explores the use of adaptive wavelet ELM (AW-ELM) to classify different gait conditions for hemiplegia and healthy subjects. Seven sets of time-domain features and one set of frequency-domain features were used for dimensionality reduction which leads to AW-ELM achieving a maximum testing accuracy of 91.15% [46]. In another analysis and classification study of EMG, on aggressive and normal activities, [47] uses bi-spectrum to analyze the EMG signals which reveal phase information known as quadratic phase coupling (QPC). In terms of accuracy, the ELM has reached the best performance of 99.75% accuracy by using QPC as a feature [47].

Table 1: Summary of the types of ELM used, their results along with the features extracted.

Author	Types of EMG	Features extracted	Types of ELM	Testing Results
Hongxin *et al.* (2015)	Forearm	RMS	Basic ELM	Mean RMSE of 1.165 ± 0.475
Sezgin, N (2013)	chin and left/right legs	Wavelet Packet Transform	Basic ELM	96.85%
Khairul *et al.* (2017)	fingers	SSC, ZC, WL, HTD, SKW, AR5, MAV	RBF-ELM	99.50%
Sahar *et al.* (2016)	legs	MAV, RMS, WL, ZC, VAR	Basic ELM	96.80%
Sahar *et al.* (2016)	legs	MAV, RMS, WL, ZC, SSC, AR	AW-ELM	91.15%
Sezgin, N (2012)	left/right biceps, left/right triceps, left/right thigh, left/right hamstring	QPC	Basic ELM	99.75%

There is another variant of ELM which is known as Weighted Extreme Learning Machine where an $N\,x\,N$ diagonal matrix W defined is associated with every training sample x_i given a set of training data $[x_1 \dots x_i, t_1 \dots . . t_i]\; for\; i = 1, \dots . , N$ where t belonging to two classes, where t_i is either -1 or +1 to indicate the negative class or the positive class. Assuming x_i comes from a minority class (indicated as +1 for t), the associated weight W_{ii} is relatively larger than the others. We have an optimization problem mathematically written as in order to minimize the weighted cumulative error and also to maximize the marginal distance with respect to each sample:

Minimize:

$$\|H\beta - T\|^2 \; and \; \|\beta\| \tag{26}$$

The optimization problem is mathematically written as

Minimize:

$$L_{P_{ELM}} = \frac{1}{2}\,\|\beta\|^2 + CW\,\frac{1}{2}\sum_{i=1}^{N}\|\epsilon\|^2 \tag{27}$$

Subject to:

$$h(x_i)\beta = t_i^T - \epsilon_i^T, \quad i = 1, \ldots, N \tag{28}$$

Where $h(x_i)$ is the feature mapping vector in the hidden layer, and β represents the output weight vector. As it is a regressor, there is only one node in the output layer. The difference between the desired output t_i and the actual output $h(x_i)\beta$ produces a training error, x_i. According to KKT theorem, the equivalent dual optimization problem with respect to (2) is

$$L_{D_{ELM}} = \frac{1}{2} \|\beta\|^2 + CW\frac{1}{2}\sum_{i=1}^{N} \epsilon^2 - \sum_{i=1}^{N} \alpha_i(h(x_i)\beta - t_i + \epsilon_i) \tag{29}$$

where the Lagrange multiplier α_i is the constant factor of sample x_i.

Furthermore, the KKT optimality conditions can be obtained by equating the partial derivatives with respect to variables $(\beta, \epsilon_i, \alpha_i)$ to zero.

$$\frac{\partial L_{D_{ELM}}}{\partial \beta} = 0 \rightarrow \beta = \sum_{i=1}^{N} \alpha_i h(x_i)^T = H^T\alpha \tag{30}$$

$$\frac{\partial L_{D_{ELM}}}{\partial \epsilon_i} = 0 \rightarrow \alpha_i = CW\epsilon_i \qquad i = 1, \ldots, N \tag{31}$$

$$\frac{\partial L_{D_{ELM}}}{\partial \alpha_i} = 0 \rightarrow h(x_i)\beta - t_i + \epsilon_i = 0, \qquad i = 1, \ldots, N \tag{32}$$

Two versions of solutions of β can be derived regarding left pseudo-inverse or right pseudo-inverse.

when

$$N\ is\ small : \beta = H^T\left(\frac{1}{C} + WHH^T\right)^{-1} WT \tag{33}$$

when

$$N\ is\ large : \beta = \left(\frac{1}{C} + H^TWH\right)^{-1} H^TWT \tag{34}$$

Given a new sample x, the output function of the ELM classifier is obtained from $f(x) = sign\ h(x)\beta$:

$$f(x)_{NxN} = signh(x)H^T\left(\frac{1}{C} + WHH^T\right)^{-1} WT \tag{35}$$

$$f(x)_{LxL} = signh(x)\left(\frac{1}{C} + H^T WH\right)^{-1} H^T WT \tag{36}$$

The output function in terms of kernel is naturally derived from the $N\ x\ N$ version:

$$f(x)_{kernel} = signh(x)H^T\left(\frac{1}{C} + WHH^T\right)^{-1} WT \tag{37}$$

$$= sign\begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T \left(\frac{1}{C} + W\boldsymbol{\Omega}_{\mathbf{ELM}}\right)^{-1} WT \tag{38}$$

The hidden layer output in ELM, also called feature mapping, maps the data with dimensionality L from the original data space to the hidden layer space, and as the main feature of ELM, the input weights and bias are randomly generated according to any continuous probability distribution.

ELM has a universal approximation capability and if the theorems are satisfied, then theoretically, the hidden layer node function can include almost every nonlinear piecewise continuous function [2,6]. The many feature mapping functions can be considered which include but not limited to

Sigmoid function

$$G(a, b, x) = \frac{1}{1+\exp\left(-(a.x+b)\right)} \tag{39}$$

Gaussian function

$$G(a, b, x) = \exp(-b\|x - a\|^2) \tag{40}$$

Hardlimit function

$$G(a, b, x) = \begin{cases} 1, & if\ a.x - b \geq 0 \\ 0, & otherwise \end{cases} \tag{41}$$

Multiquadrics function

$$G(a,b,x) = (\|x-a\|^2 + b^2)^{\frac{1}{2}} \tag{42}$$

If the feature mapping function $h(x)$ is not known, a kernel matrix is defined [6] according to Mercer's conditions is shown in (12) and the most popular kernel being used is the Gaussian kernel $K(u,v) = \mathrm{e}^{(-\gamma\|u-v\|^2)}$, where γ is the kernel parameter.

$$\Omega_{ELM} = HH^T : \Omega_{ELMi,j} = h(x_i).h(x_j) = K(x_i.x_j) \tag{43}$$

Hence, the kernelization of output function of an SLFN is as

$$f(x) = h(x)H^T\left(\frac{1}{C}HH^T\right)^{-1} = \begin{bmatrix} K(x,x_1) \\ \vdots \\ K(x,x_N) \end{bmatrix}^T \left(\frac{1}{C} + W\boldsymbol{\Omega}_{\mathbf{ELM}}\right)^{-1} T \tag{44}$$

Hence, the weighted ELM is able to provide a unified solution for the generalized Single Layer Feedforward Networks through different feature mappings or kernels, and by adding a weight matrix in ELM to affect the impact of minority and majority class, the weighted ELM also provided a solution for the learning of imbalanced data sets. From the perspective of optimization, the traditional ELM learns by finding a mathematical function that approximates the underlying relationship between the input and its target while maximizing the marginal distance and minimizing the error. In traditional ELM, the training error is proportional to the Lagrange multiplier $\alpha_i = C\epsilon_i, \forall_i \, of \, x_i$, which indicates the weight of fraction in decision function. This is so that inputs with more information, usually with larger training error, are prioritized in the training model.

However, as a good function approximator, it must also give priority to inputs that can characterize the data distribution and not just inputs with larger errors. Hence, the weighted ELM provides additional emphasis to the inputs which characterizes the imbalanced class distribution. As seen from the Lagrange multiplier $\alpha_i = CW\epsilon_i, \forall_i \, of \, x_i$, samples from the minority class is assigned with larger weight W_{ii} such that the information of imbalanced class distribution is well perceived.

The weight matrix $W = diag\{W_{ii}\},\ i = 1, \ldots, N$ also plays a vital part as it determines how much further the boundary is pushed towards the majority class and the degree of re-balance between the minority and majority class. The weight matrix W_{ii} for every sample x_i can be defined so that the proposed algorithm actually belongs to the family of cost-sensitive learning. Thus, a weighting scheme can be automatically generated from the class information:

$$Weighting\ Scheme\ W1:\ W_{ii} = \frac{1}{\#(t_i)} \tag{45}$$

where $\#(t_i)$ is the number of samples belonging to class $t_i, i = 1, \ldots, m$.

With the application of weighting scheme $W1$, the quantity of minority class and majority class is rebalanced into the ratio of $1{:}\,1{:}\cdots{:}\,1$. Golden ratio, 0.618:1, is also another weighting scheme that can be considered to diminish the balancing step between the minority and the majority classes. In binary classification, the minority class is assumed to be the +1 and the majority class is assumed to be -1. As for multiclass classification, classes whose number is below the average samples per class are referred to as the minority classes while the majority classes are those with sample numbers above the average. Therefore, the boundary is pushed slightly back to the minority class if compared to the weighting scheme $W1$. This is to alleviate the misclassification cases in compromise of the majority class.

$$Weighting\ Scheme\ W2 = \begin{cases} W_{ii} = \dfrac{0.618}{\#(t_i)}, & if\ t_i > AVG(t_i) \\ W_{ii} = \dfrac{1}{\#(t_i)}, & if\ t_i \leq AVG(t_i) \end{cases} \tag{46}$$

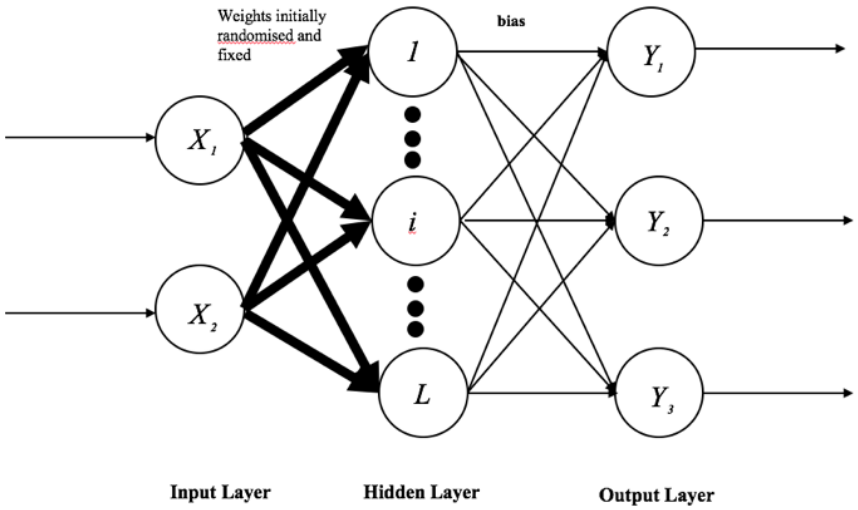

Figure 8: ELM architecture ($X_1 and\ X_2$ are the inputs, 1…i…L are the activation functions, Y_1, Y_2 and Y_3 are the output neurons)

Figure 8 shows the ELM architecture used in this study. The input layer consists of two neurons which are EMG signals from two muscles, i.e. Tibialis Anterior (TA) muscle and Gastrocnemius Lateral (GL) muscle. As for the hidden layer, the number of hidden neurons is set to 250, 300, 350, and 400 as this is the parameter to be tuned for performance comparison. A sigmoid activation function, L, is used to compute the inputs and bias. The output layer consists of 3 output neurons since we have three parameters (gait speed, %stance, and %swing) to output as targets. The layers are all interconnected with the previous and next layer of that specific layer. The connections (weights) from the input layer to the hidden layer are initiated, randomized, and never updated during the iterative tuning; it is only the weights from the hidden to the output layers that are iteratively tuned.

In this study, the ANN architecture has the same input, the same amount of hidden neurons configuration, and the same target as the ELM model. The difference in ANN is that $tan - lin\ (L)$ and $pure - lin\ (Y_n)$ are used as transfer function for hidden and output layer respectively. A three-layered network can accurately classify any non-linear function [49].

2.8 Conclusion

In this chapter, EMG has been shown to be a widely used parameter for feature selection and many features have been presented, both statistical and non-linear analysis of EMG. It is in many applications which include prosthesis control or other assistive devices [11-12]. EMG as a feature is not new to ANN but to ELM and its application, it is still at the infantry stage. Table 1 shows studies done by [42-47] where EMG from different parts of the body and ELM variants were used. Thus, using the leg muscles' EMG is common but different features were used in this study to approximate the gait trajectory using the Ensemble variant of ELM. The result is shown to have fast ELM model deployment and good prediction accuracy. This is as a result of the advantages ELM can bring.

CHAPTER 3 – EXPERIMENT

3.1 Type 1

Experiments were conducted to acquire gait profiles from 18 healthy human subjects. Ethical approval was obtained from the institution's Research Ethics Committee. The inclusive criteria of the subjects are (i) no other known neurological disorder disease, (ii) understand and follow basic instruction. Participants were provided with written consent for this research. The data acquisition and preprocessing are divided into two parts: i) Gait parameters and ii) EMG signals. The data would then feed into the ANN/ELM model for training and testing.

3.1.1 Gait Parameters

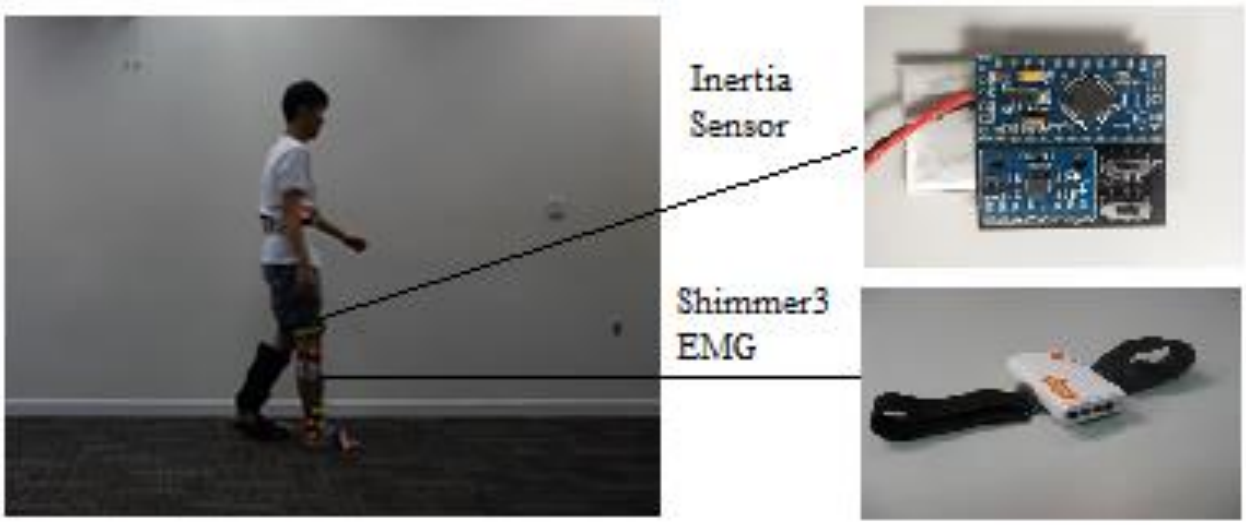

Figure 9: A subject completing the gait cycle.

All participants (mean ± std, Age: 23 ± 1.6 years old, Height: 173.3 ± 6 cm, Weight: 63.7 ± 10.3 kg) were asked to wear the inertia sensor system (Figure 9) along with EMG sensor device (Shimmer Sensing – Shimmer3 EMG) and perform a series of

walking tasks, namely slow speed walking, normal speed walking and fast speed walking - for 2 minutes respectively on a 5-meter walkway. The inertia sensor system consists of a programmed Arduino Mini Pro and an MPU-6050 module (InvenSense – MPU6050) (Figure 10). Before starting the experiment, a measuring tape was used on the wall and on the floor to measure the 5 meter distance. On the wall, 2 visible colored tapes were stuck on both ends of the measuring tape while on the floor, the visible colored tape was stuck on the floor along the 5 meter distance. This was to ensure that the participant was able to walk in a straight line with the tape on the floor as a guide. Once done, a phone camera operating at 60 frames per second was set up to capture walking activities concurrently. Red LED lights were placed on the areas of interest which include knee joints and heel and toe part of the outsole as shown in Figure 9. The videos and data recorded were then transferred to the computer where MATLAB was used to analyze them.

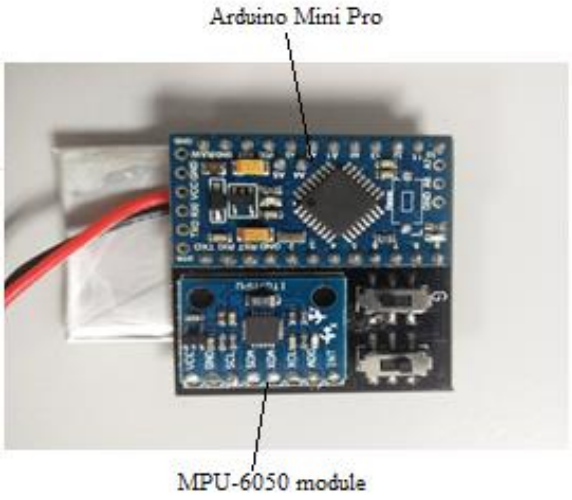

Figure 10: Inertia sensor used for the experiment

Figure 11: Shimmer3 EMG unit (Shimmer Sensing)

To acquire EMG signals, 2 Shimmer3 EMG (Shimmer Sensing – Shimmer3 EMG, Figure 11) units were used. The units were configured to have a sampling rate of 512 Hz to ensure high quality reproducibility of the actual summation of the muscle's activity. Each EMG unit was connected to five electrodes, namely, a positive and a negative electrode for each for two channels and a neutral reference electrode. The Shimmer3 EMG units were placed on the shins, whereas the reference electrodes were placed on the knee and the other electrodes were placed on two muscles, namely Tibialis Anterior (TA) muscle and Gastrocnemius Lateral (GL) muscle. The reason for using three electrodes in this way was that the EMG signals, typically, exhibit a low signal-to-noise ratio. Noise interferences are usually from power lines and nearby electrical sources. The signal recorded up by each individual electrode consists of noise from the environment along with the local electrical signal from the muscles at the position of skin contact. The noise from the environment is common to all electrodes, whilst the local electrical signal depends on the electrode's position. Thus, if one signal is subtracted from another, the common component (the undesired noise) will be canceled by the subtraction, whilst the local signals (the desired EMG component) will remain after subtraction and can be amplified to make it easier to process. This process is called Common Mode Rejection (CMR) and is used in the Shimmer3 EMG unit.

Understanding of anatomy of each muscle is required for proper EMG electrode placement. Hence, TA and GL muscles are selected as TA contributes to dorsiflexion and inversion of the foot while GL is responsible for plantar flexion of foot at ankle joint and the flexing of leg at knee joint. Another reason why GL was chosen was because GL was primarily involved in fast movements of the leg which provides a contrast to slow, normal and fast speed. Quality of EMG signal depends on muscle's shape, fiber directionality, motor points, tendon positions, and insertion points. In this study, the placement of EMG electrodes follows the recommendation from SENIAM [50].

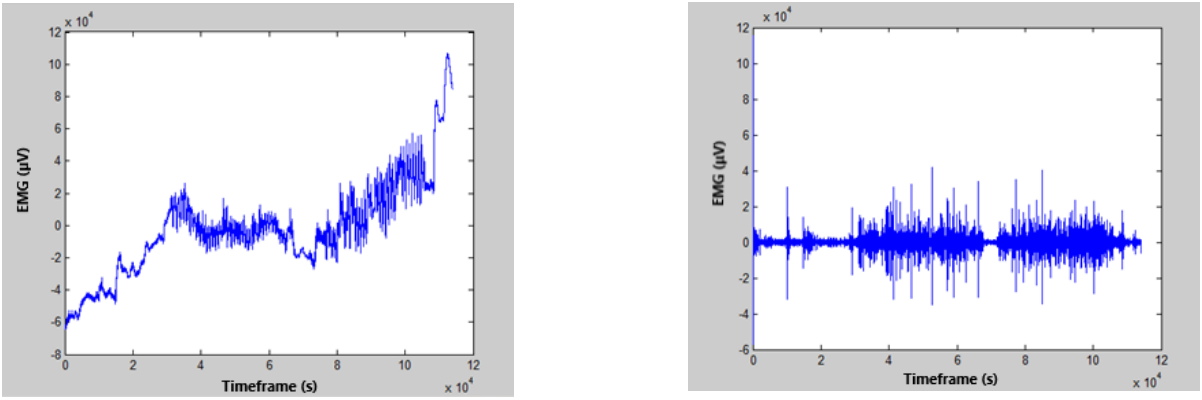

Figure 12: Raw EMG data (left). Processed EMG data (right).

The raw EMG signals as seen in Figure 12 were processed in MATLAB. To remove signal interference from mains electricity, notch filtering was used where a band-stop filter was adjusted to remove the local mains frequency of 50 Hz and noises at 100 Hz and 200 Hz (noisy peaks were observed at these frequencies in the Fast Fourier Transform frequency spectrum). Next, to observe the overall level of activity in a particular muscle, the linear envelope of the EMG signals was extracted by first applying full wave rectification to the signal and subsequently pass it through a low pass filter of 6 Hz.

Normalization was used to eliminate variability across subjects, electrode placement, and day-to-day differences in measures of the same muscle site and this is done prior to the start of the experiment. In this study, the EMG signals were normalized against the maximum voluntary contraction of every subject.

3.1.2 Processing data with camera system

An algorithm was coded to track the red LED lights in the video which outputs the coordinates of the LED lights against time. The algorithm detects the red components from the grayscale of each frame and outputs their coordinates. The outputs were later used to obtain the heel-strike and toe-off time. The purpose of the camera system was mainly used to validate the measurement from the inertia sensor system as well as to obtain gait speed. while gait speed was determined manually using a video tracking process acquired from a camera system where the x-coordinates between toe-off points were converted into meters and divided by its gait cycle time.

3.1.3 Processing inertial and EMG data

After transferring the data to the computer using the Shimmer Dock, the data is saved as an excel file and is passed through to MATLAB to be processed. The data is processed using an algorithm which will be explained in brief next.

Algorithm

1. Once passed through, the program should look like the figure below (Figure 13):

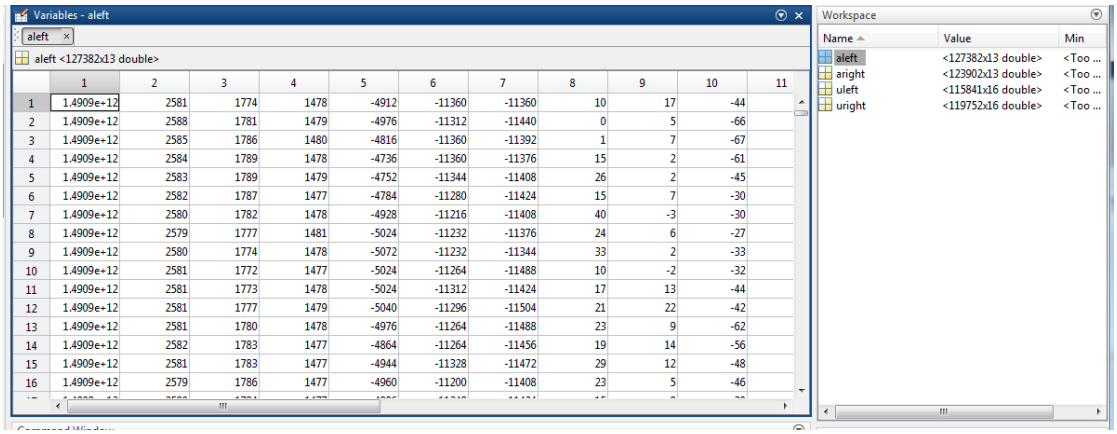

Figure 13: Left ankle inertial data with Unix time shown.

2. The Unix time (column 1) for all sets were converted to Unix time(s) and replaced with the newly converted time (Figure 14).

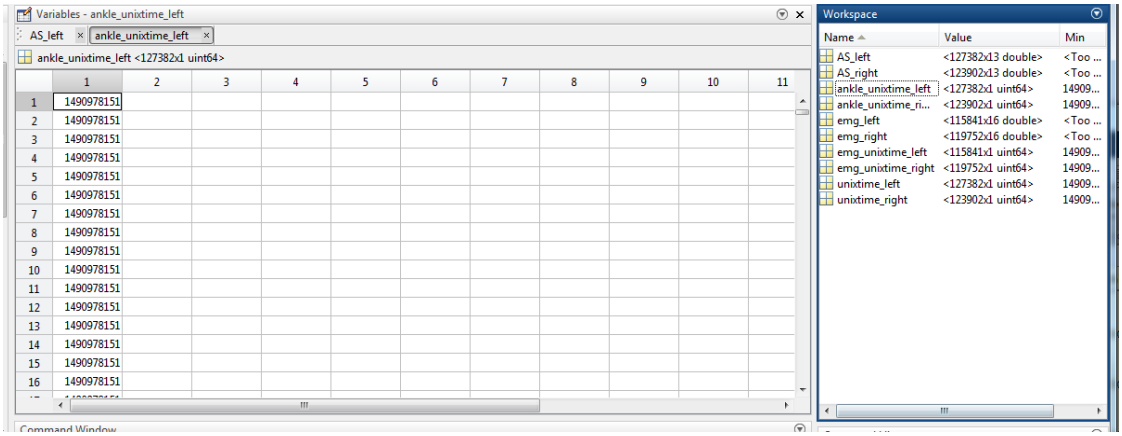

Figure 14: Left ankle converted Unix time in seconds.

3. The data was synchronized by overlapping the Unix time from different devices and only the data that is in the overlapped Unix time was used. Before synchronizing both inertial and EMG data, it was less confusing to synchronize the left and right

side of inertial data first (should have the same dimension), then the left and right side of EMG data (should have the same dimension), and lastly, both EMG and inertial data synchronized together (should have the same dimension) as shown in Figure 15. The starting number and the ending number for Column 1 should be the same for both inertial and EMG data.

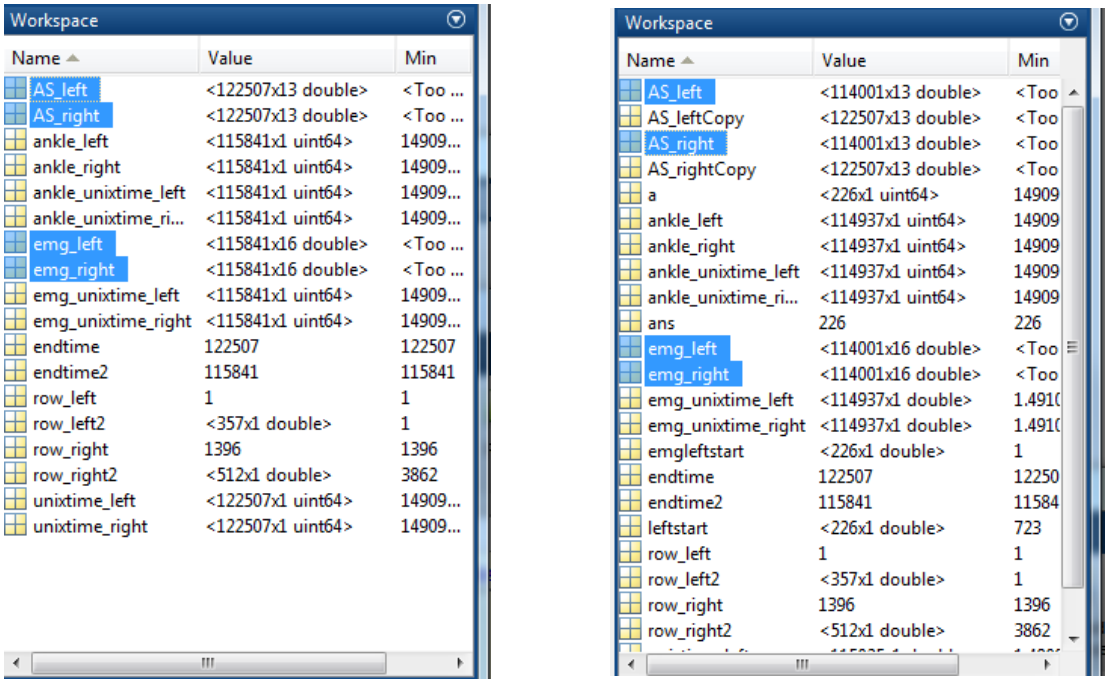

Figure 15: Left and Right inertial data showing the same dimension. Left and Right EMG showing the same dimension (left). Both Inertial and EMG data showing the same dimension (right).

4. Inertial data were calibrated based on the settings used in the device. The acceleration data was multiplied by $9.81 m/s^2$ and divided by the sensitivity value from MPU which was 16384 while the gyroscope data was converted into degree/seconds by dividing its value with the sensitivity value from MPU which was 131. Once done, it was then passed through a 5^{th} order low pass filter.
5. Check the gyroscope z-axis data to see if it has been inversed. An inverse data would look like figure 16 below (Right Gyro Data). If it is so, then multiply the gyroscope z-axis by -1 and it will look similar to the Left Gyro Data.

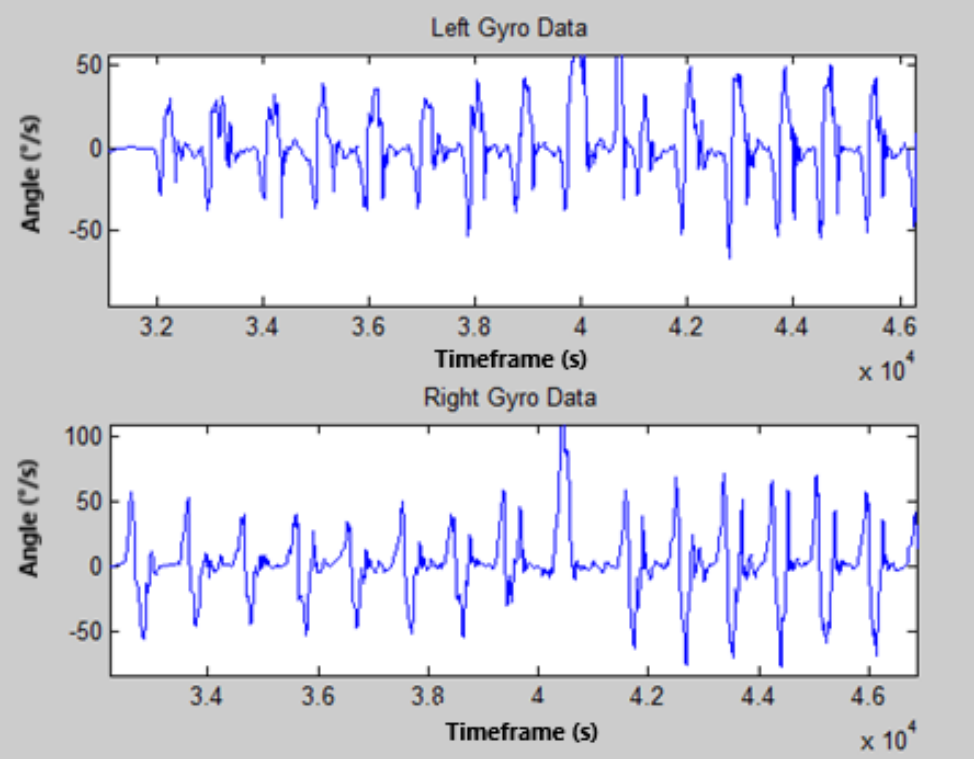

Figure 16: Normal Left Gyroscope Data (top). Inversed Right Gyroscope Data (bottom).

6. EMG data were processed by first having Fast Fourier Transformed to find the noise frequency and removed using 3 band-stop filters at 50Hz, 100Hz and 200Hz, and a band-pass filter with a cut off frequency at 20Hz and 250Hz. After that, it was then rectified and filtered again using a 4th order low-pass Butterworth filter and it should look something similar to Figure 17 below.

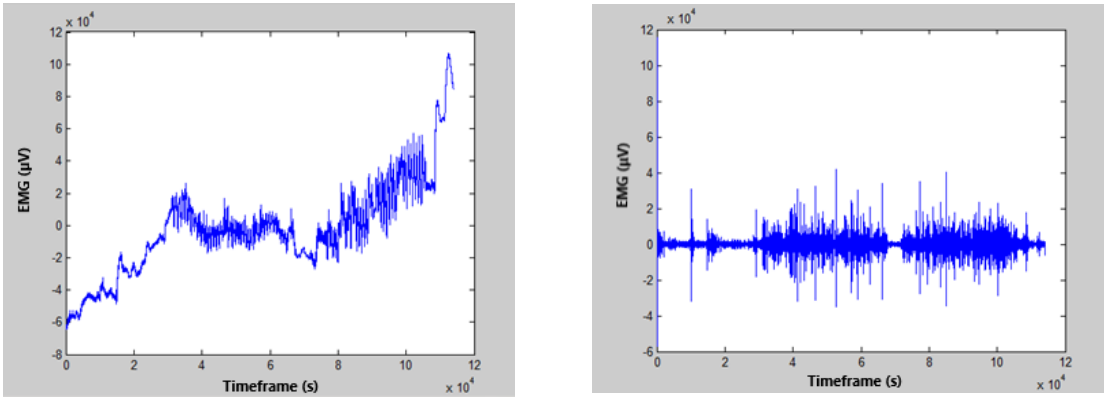

Figure 17: Raw EMG data (left). Processed EMG data (right).

7. A graph of the z-axis gyroscope data was plotted with its peak labeled using 'findpeaks' function in MATLAB. Unwanted peaks were removed manually using the brush to get the desired heel-strike and toe-off points as observed in Figure 18.

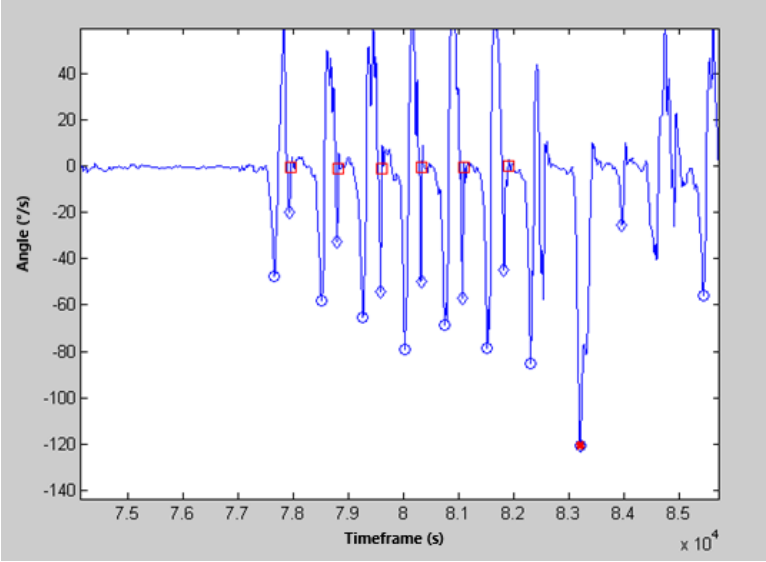

Figure 18: Z-axis gyroscope data with its peaks labeled.

8. With the obtained heel-strike and toe-off points, EMG can be segmented using the toe-off points to obtain EMG (right) for one gait cycle(left) as displayed in Figure 19.

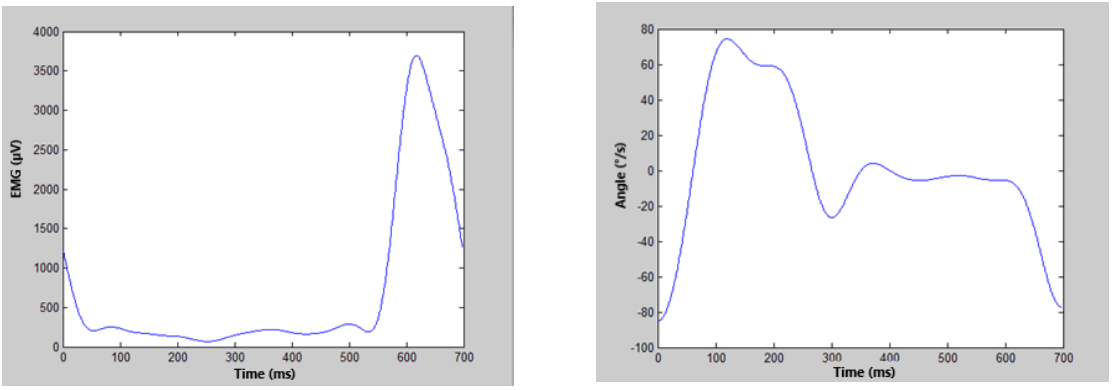

Figure 19: One gait cycle (left). EMG corresponding to the gait cycle (right)

9. With the obtained heel-strike and toe-off points, the temporal gait parameters were derived from the inertia sensor system using the method described in [51]. The gait cycle time, percentage of swing phase, and stance phase were calculated based on the heel-strike and toe-off time obtained as follows:

$$GC_i = T_{i+1} - T_i \tag{47}$$

$$ST_i \ \% = \frac{T_{i+1} - H_i}{GC_i} \ x\ 100 \tag{48}$$

$$SW_i\ \% = \frac{H_i - T_i}{GC_i}\ x\,100 \tag{49}$$

where GC_i represents the gait cycle time for the i-th gait cycle, H is the heel-strike moment (initial contact ground of heel), T is the toe-off moment (terminal of toe leaving from the ground), ST% is the stance phase percentage, and SW% is the swing phase percentage.

A total of 1471 gait cycle EMG signal was extracted from 18 subjects and this data was then fed into the ANN and ELM networks.

3.1.4 ANN Setup

The ANN SCG model has been developed using built-in MATLAB functions while the ELM model was based upon the open-source algorithm from [52]. All data were divided into 75% training, 10% testing, and 15% validation. For effective comparisons, the training and testing accuracy was calculated using root mean square error. The mean estimation error (in percentage), which were the measure of differences between estimated gait parameters and the actual measurement.

The conventional ANN SCG model consists of a single hidden layer and it is a feed-forward neural network. As EMG signals were segmented to each gait cycle, each set of EMG signal would have a different data length (due to the differences in gait cycle time). Each set of EMG signals was then resized to 2000 elements and they are trained based on scaled conjugate gradient algorithm with sigmoid as the hidden layer transfer function and pure-lin as the output layer transfer function. Parameters such as learning epoch, goal error, learning rate, and performance goal were set to 1000, 0.001, 0.01 and $1x10^{-6}$ respectively. In the case of ELM, sigmoid activation function was chosen. The hidden neurons for both models were set to 250, 300, 350, and 400. To strengthen the network and reduce over-fitting, a 10-fold cross-validation was performed 30 times for each set of hidden neurons and the average results were used.

3.1.5 Defining parameters for SCG

In the construction of neural network, it is necessary to set the parameters – eg. Learning epoch, goal error, learning rate, activation function, training algorithm, and the number of hidden neurons. As the perfect amount of hidden neurons is unknown, the amount is set to 250, 300, 350, and 400 to investigate where the recommended amount falls. Here, the learning rate is set to 0.01, and the maximum number of learning epochs is set to 1000 while the goal error is set to 0.001. This is so that the learning process can be efficient if the performance goal has not been met. In another case, out performance goal was set to a small value of $1x10^{-6}$ to ensure that the errors were minimized from the neural network. There were 3 conditions to which the training of neural network terminates. Firstly, it is when the goal error reaches 0.001 and secondly, it is when the performance goals have been met. The last case is when the epoch of 1000 has been reached as early stopping can avoid overfitting.

The type of training function used for this neural network was the scaled conjugate gradient backpropagation with sigmoid and pure-lin as the hidden layer and output layer activation function respectively. For the activation function, after some initial experimentation, it was found that sigmoid performs better than hyperbolic tangent. SCG was chosen as it performs well particularly for networks with a large number of weights and the performance does not degrade as quickly when the error is reduced. The most important feature is that it has relatively modest memory requirements.

3.1.6 Defining parameters for ELM

The ELM algorithm has already been coded and is downloaded from [16]. Hence, it is only recommended to configure the parameters – number of hidden neurons, types of ELM (classification or regression), and the activation function. The comparisons between the SCG and ELM models were required to have the same parameters. Hence, the number

of hidden neurons was set to 250, 300, 350, and 400 while the sigmoid activation function was used. In our case, it is a regression type of ELM.

3.1.7 Normalization of data

Data were normalized to facilitate training. This is because when the net input is greater than a certain value, the sigmoid transfer function used in the hidden layer will become saturated and if this happens during the start of the training process, the gradients of error will be very small. In other words, when looking at the error surface, it will look flat and it will be hard to find the minimum point which leads to a slow training process. Hence normalizing the data within the sigmoid transfer function limit will give the error surface a more spherical shape. This means that it would not take much effort in finding the minimum which leads to easier learning.

Normalization was applied to both the inputs and the outputs and reversed transformed back into the original scale after the output is predicted. The Max-Min normalization method was used to scale the matrix where:

$$normalised\ matrix = \frac{matrix - \min(matrix)}{\max(matrix) - \min(matrix)} (\max range - \min range) + \min range \qquad (50)$$

Max range refers to the maximum value of range while Min range refers to the minimum of range where the maximum of 0.8 and minimum of 0.2 was used.

3.1.8 10-fold cross-validation

To perform 10-fold cross-validation, a randomized label(number) for our input data had to be initialized. The randomized label will have the same matrix dimension as our input matrix except that it contains randomized order of numbers from 1 to N (N being the number of elements in the input matrix). Next, we sort the input matrix based on the randomized label as this will give us cross-validation randomness. Before we try to accomplish 10-fold cross-validation, it is necessary to take out 15% of the data as a validation set while using the remaining to testing and training. Having 10-folds in this

study, it is important for this step to be completed 10 times. For example, a (1,:) testing set and (9,:) training sets, 1st fold would be (1,:) for testing, and (9,:) for training. 2nd fold would be (2,:) for testing and (1,:) and (3:9,:) for training and so on until it completes it 10 times.

3.2 Type 2

3.2.1 Objective

Experiments were conducted to predict the y-trajectory of gait cycles from 20 healthy human subjects. Ethical approval was obtained from the institution's Research Ethics Committee. The inclusive criteria of the subjects are (i) no other known neurological disorder disease, (ii) understand and follow basic instruction. Participants were provided with written consent for this research. The data acquisition and preprocessing are divided into two parts: i) Gait y-trajectory and ii) EMG signals. The data would then feed into the ELM model for training and after, testing in real-time.

3.2.2 *Experimental Setup and Procedure*

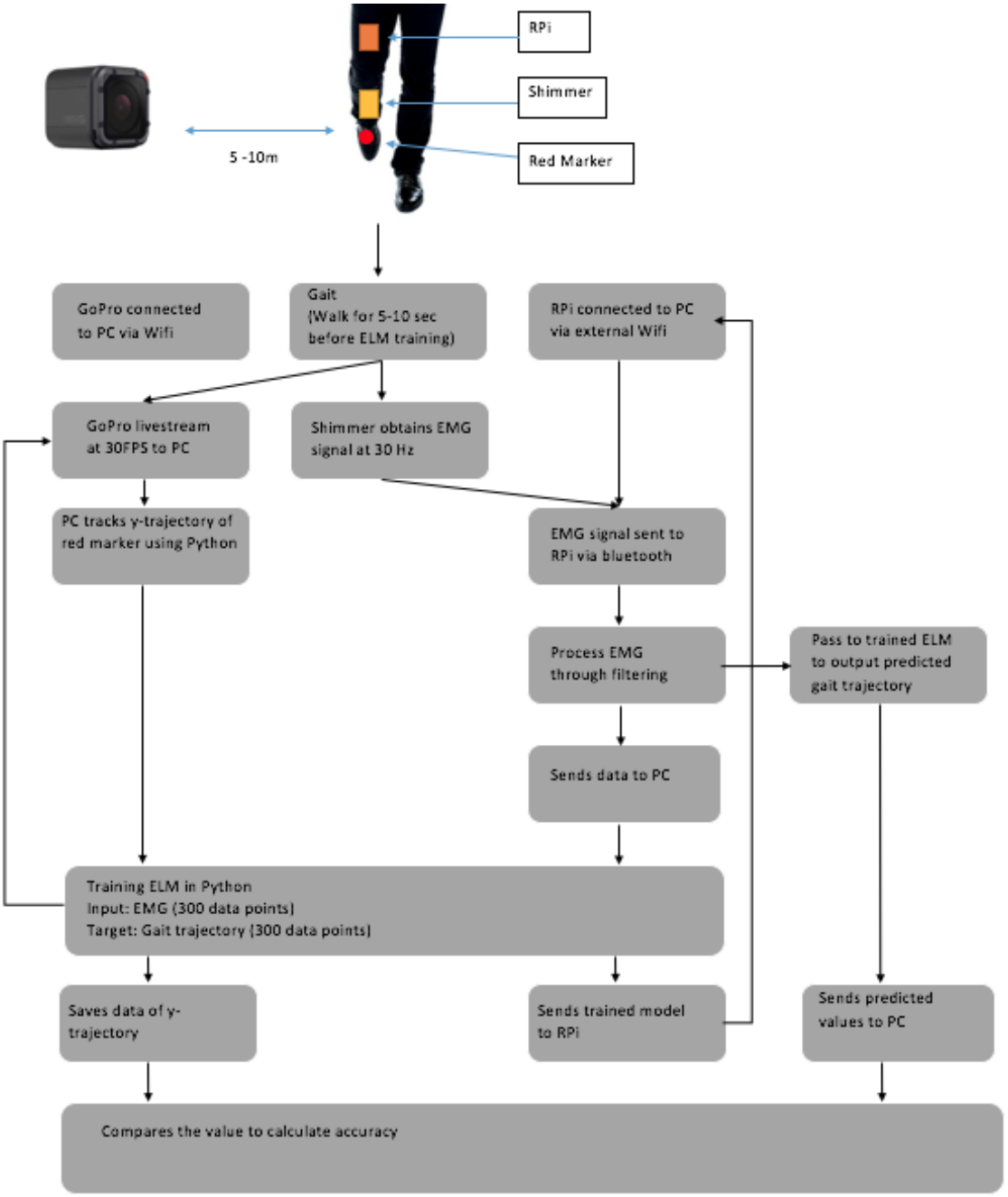

Figure 20: Type 2 Experiment Flow Chart

As the project progresses to real-time implementation, a Raspberry Pi was introduced for data transfer via Wi-Fi. The experiment starts off by having the participant wear a Shimmer EMG, a Raspberry Pi, and a red marker. There was a GoPro or a Camera live streaming the participant when he or she was walking and a laptop that controlled the data transfer among the devices. While the experiment proceeded, the laptop streamed

two types of data: EMG signals and gait trajectories. EMG signals were obtained from Shimmer EMG which was connected to the Raspberry Pi using Bluetooth and the Raspberry Pi was connected to the PC using an external Wi-Fi. It is true that the Shimmer could be connected to the PC using Bluetooth as well, but the connectivity range of Bluetooth eliminates the potential of a smooth data transfer when the participant walks out of range. The GoPro or a camera was connected to the same Wi-Fi as the laptop as it live-streamed the frames into the laptop where the laptop then extracted the coordinates of the red color marker using Python. Both the EMG signal and the gait trajectory were filtered to eliminate noises with the EMG signals having to go through more steps, like rectifying and lowpass, so as to have more readable data.

After walking for a few steps and pausing the participant, the gait trajectory was segmented and then used as the target for the training set while the segmented EMG signal was used as the input for the training set. The trained model was then sent back to the Raspberry Pi and the experiment was repeated again but this time, the Raspberry Pi will be predicting the gait trajectory based on the EMG signal. The GoPro or camera still live-streamed the trajectory data back to the laptop so that the data could be used to calculate the testing accuracy after the experiment.

3.3 Type 3

3.3.1 Objective

Experiments were conducted to obtain the 3 constants from a sigmoid function to predict the gait cycles from 20 healthy human subjects. Ethical approval was obtained from the institution's Research Ethics Committee. The inclusive criteria of the subjects are (i) no other known neurological disorder disease, (ii) understand and follow basic instruction. Participants were provided with written consent for this research. The data acquisition and preprocessing are divided into three parts: i) Gait x-trajectory and ii) EMG signals iii) Segmentation of data. The preprocessed and segmented data of x-trajectory were then used to extract the 3 constants of a sigmoid function while the EMG signals

were data engineered to extract 5 distinct features. The data would then feed into a boosted and non-boosted ensemble ELM model for training and testing.

3.3.2 Experimental Setup and Procedure

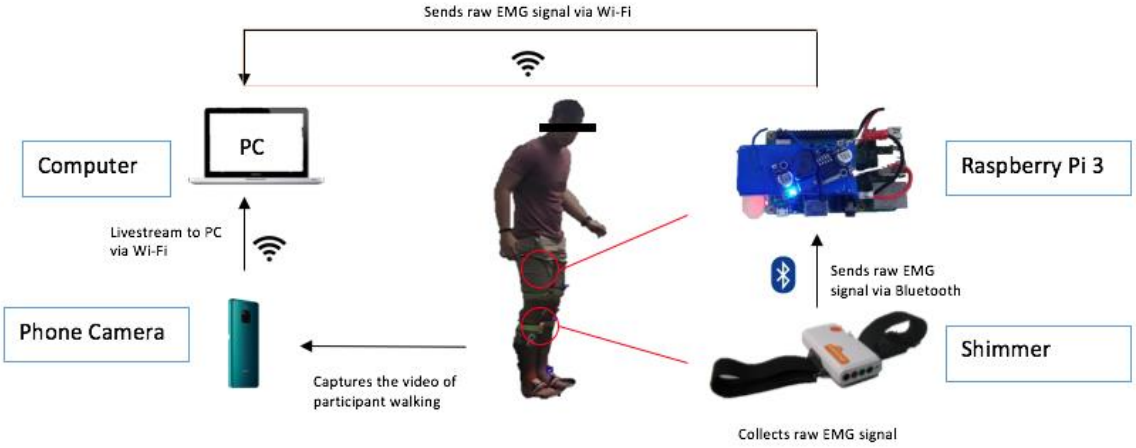

Figure 21: Type 3 Experiment Setup

All participants (mean ± std, Age: 25 ± 1.8 years old, Height: 168.5 ± 6.3 cm, Weight: 59.7 ± 11.4 kg) were asked to wear the inertia sensor system (Figure 10) along with EMG sensor device (Shimmer Sensing – Shimmer3 EMG) and performed a series of 20 cycles walking of gait length 1m on a 5-meter walkway. A phone camera operating at 30 frames per second was setup to capture walking activities concurrently. Red LED lights were placed on the ankle, as shown in Figure 21, to track the trajectory using an algorithm coded in Python which outputs the coordinates of the red LED lights against time. The video was streamed to the laptop via hotspot and processed on the laptop. The x and y coordinates were later used to segment each gait cycle and the segmented x co-ordinates, which was the x-gait trajectory, were further segmented again into n parts. The n parts were inputted into a sigmoid function where the n parts were iteratively approximated into the 3 constants of a sigmoid function which is $a, b \; and \; c \; \text{from} \frac{a}{e^{-b}+c}$.

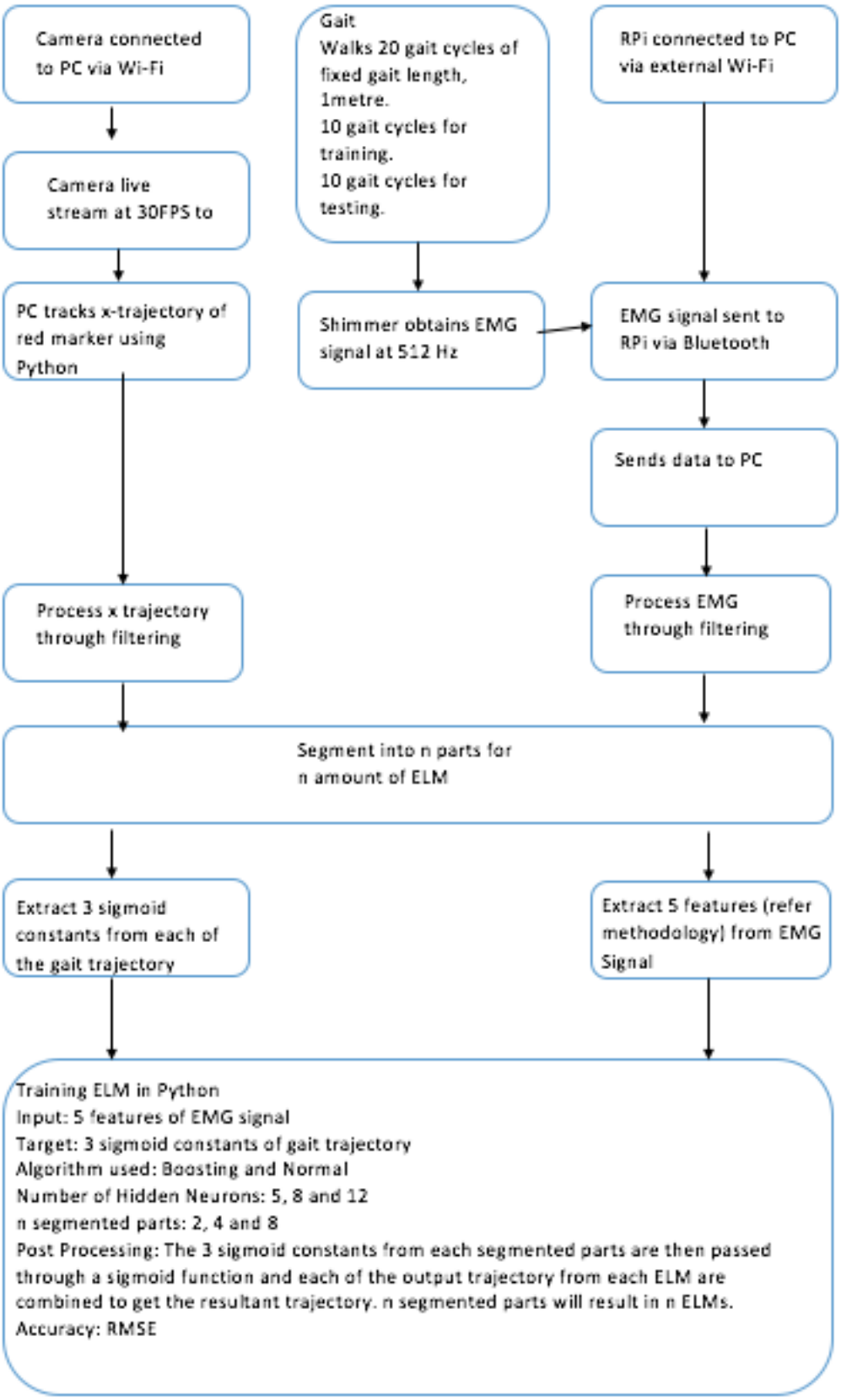

Figure 22: Type 3 Experiment Flow Chart

3.3.3 Normal Ensemble

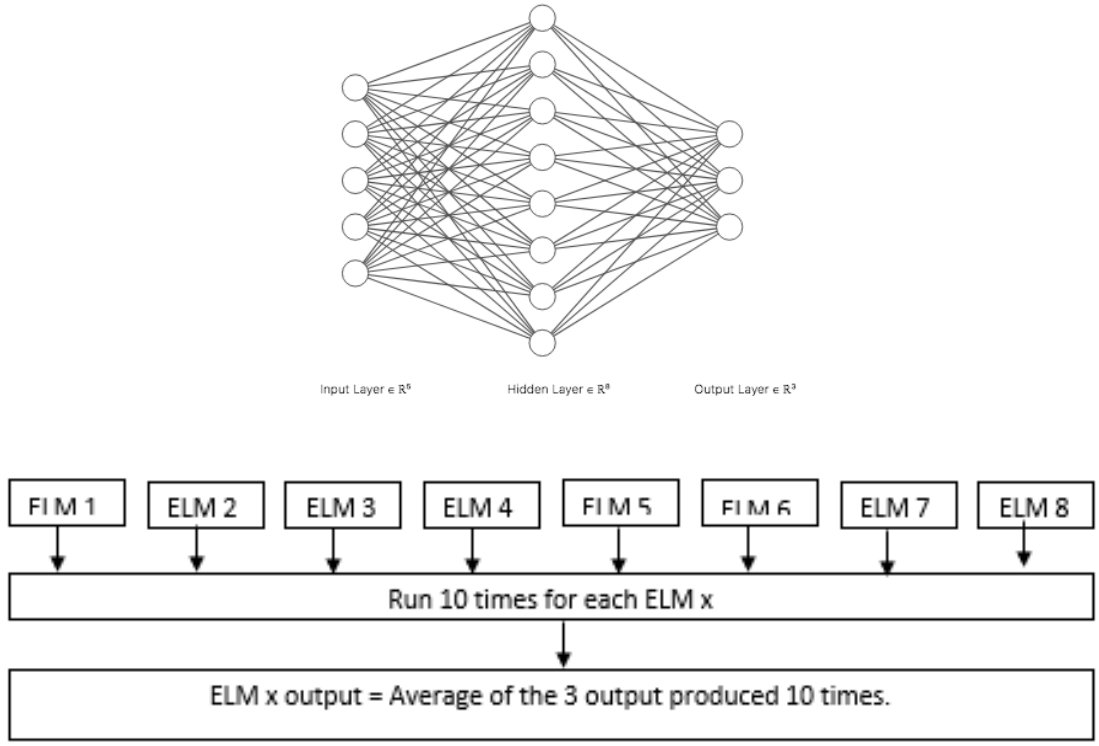

Figure 23: ELM x Architecture

Algorithm Training of Ensemble ELM x

Input: Features X_n, Sigmoid function parameters target T , number of individual models M, and number of hidden neurons L. $n = number\ of\ segmented\ EMG\ parts$

Output: Sigmoid function parameters.

for m = 1 to M (M=10) do

Initialize random input weights and biases W^m and b^m.

Calculate the hidden matrix $H^m = G(XW^m + b^m)$.
Calculate the output weights $\beta^m = H^{m+}T$.

Calculate the outputs $Output^m = H^m\beta^m$.

end

Calculate the average output $FOutput_n = \frac{\sum_{m=0}^{m} Output^m}{\mathrm{m}}$.
Repeat for n EMG parts.

3.3.4 *Ensemble with boosting*

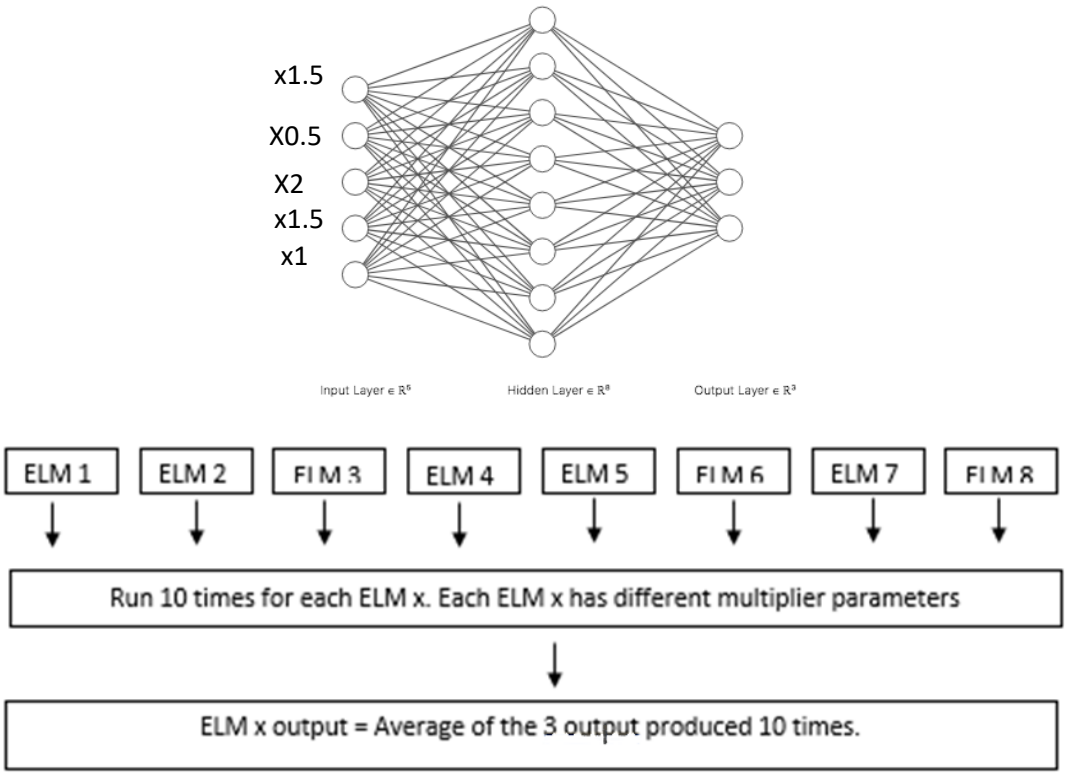

Figure 24: ELM x with multiplier Architecture

Algorithm Training of Ensemble ELM x with Boosting

Input: Features X_n, Sigmoid function parameters target T, number of individual models M, and number of hidden neurons L. $n = number\ of\ segmented\ EMG\ parts$

Output: Sigmoid function parameters.

for m = 1 to M (M=10) do
Features X_n are multiplied with different multipliers. (Large multiplier shows great importance and vice versa)

Initialize random input weights and biases W^m and b^m.

Calculate the hidden matrix $H^m = G(XW^m + b^m)$.
Calculate the output weights $\beta^m = H^{m+}T$.

Calculate the outputs $Output^m = H^m\beta^m$.

end

Calculate the average output $FOutput_n = \frac{\sum_{m=0}^{m} Output^m}{\mathrm{m}}$.
Repeat for n EMG parts.

The ELM model has been developed based upon the open-source algorithm from [16]. The ELM consists of a single hidden layer and it is a feed-forward neural network All data were divided into 50% training, 45% testing, and 5% validation. In our case, we have n =2, 4, and 8. So 8 parts will have 10 ELMs each with a total of 80 ELMs. Each training and testing will have 5, 8, and 12 hidden neurons and a sigmoid activation function set. The hidden weights and biases were randomly initialized for the 80 ELMs while the output from the ensemble of the 10 ELMs of each part will be averaged and later passes through a sigmoid function to get the resultant output. The resultant 8 resultant outputs were combined and compared with a filtered trajectory (to reduce noise) from the phone camera. For effective comparisons, the training and testing accuracy was calculated using root mean square error. Having an ensemble of ELMs can strengthen the network and reduce over-fitting.

CHAPTER 4 – RESULTS & DISCUSSIONS

4.1 Experiment Type 1

Experimental results are presented from Table 2 to Table 5 and they represent the performance and duration for both ANN and ELM models. Both models output 3 parameters, namely Gait Speed, % Stance Phase, and % Swing Phase. From Table 2 to 5, it can be observed that, overall, ELM performed better than ANN for all the three parameters; not just in terms of lesser error, but also shorter training duration.

Table 2: Experimental results of Basic ANN and ELM for the parameter – Gait Speed (m/s).

Gait Speed (m/s)					
ANN (SCG)					
Hidden Neurons	Testing RMSE (m/s)	Testing Time(s)	Training RMSE (m/s)	Training Time(s)	Mean Test Estimation Error (%)
250	0.1419	0.0354	0.0166	15.7352	12.9271
300	0.1650	0.0511	0.0173	20.7394	13.5948
350	0.1543	0.0580	0.0192	20.3116	13.2240
400	0.1722	0.0614	0.0217	22.4238	14.0108
ELM					
Hidden Neurons	Testing RMSE (m/s)	Testing Time(s)	Training RMSE (m/s)	Training Time(s)	Mean Test Estimation Error (%)
250	0.1236	0.0170	0.1020	0.1434	11.8646
300	0.1357	0.0152	0.1003	0.2091	12.6031
350	0.1428	0.0114	0.0987	0.2552	12.6902
400	0.1483	0.0213	0.0935	0.2813	12.9716

Table 3: Experimental results of Basic ANN and ELM for the parameter – % Stance Phase.

% Stance Phase					
ANN (SCG)					
Hidden Neurons	Testing RMSE (%)	Testing Time(s)	Training RMSE (%)	Training Time(s)	Mean Test Estimation Error (%)
250	5.1087	0.0220	1.4372	15.4622	11.7519
300	6.1807	0.0411	1.6775	17.1960	13.1505
350	7.2955	0.0489	1.8534	20.2427	15.1321
400	7.9291	0.0538	1.8587	22.0761	16.4648
ELM					
Hidden Neurons	Testing RMSE (%)	Testing Time(s)	Training RMSE (%)	Training Time(s)	Mean Test Estimation Error (%)
250	3.2709	0.0149	3.0043	0.0420	7.6201
300	3.2412	0.0155	3.1441	0.0989	7.3545
350	3.3217	0.0157	2.5674	0.0953	8.0131
400	3.4432	0.0163	2.4719	0.1028	8.5426

Table 4: Experimental results of Basic ANN and ELM for the parameter – % Swing Phase.

% Swing Phase					
ANN (SCG)					
Hidden Neurons	Testing RMSE (%)	Testing Time(s)	Training RMSE (%)	Training Time(s)	Mean Test Estimation Error (%)
250	5.0545	0.06972	0.92185	19.26792	9.55680
300	6.16469	0.06217	1.27358	22.92138	12.20432
350	5.81581	0.06334	1.48631	23.21076	11.86815
400	7.39040	0.07205	1.62517	28.17323	13.40730
ELM					
Hidden Neurons	Testing RMSE (%)	Testing Time(s)	Training RMSE (%)	Training Time(s)	Mean Test Estimation Error (%)
250	3.4462	0.02170	2.01872	0.24800	6.06871
300	3.6835	0.02274	1.97239	0.23296	6.51340
350	3.8749	0.01876	1.68943	0.25170	6.73352
400	3.9461	0.02164	1.37365	0.21653	6.91475

Table 5: Validation results for Gait Speed, Stance % and Swing %

	Validation RMSE for 250 hidden neurons		
	Gait Speed(m/s)	% Stance	% Swing
ANN (SCG)	0.0925	12.5184	9.2920
ELM	0.2939	1.6134	1.7952

Another important trend shown by ELM is that as the amount of hidden neurons increases, the training RMSE decreases while the testing RMSE increases. This trend could not be observed from the ANN which makes ELM a more predictable and stable model. It also can be seen that though, for ELM, the training RMSE is higher than that of the ANN for gait speed, but generally, ELM yielded better testing results. For 250 hidden neurons, ELM yielded 11.86%, 7.62%, and 6.07% mean test estimation error for gait speed, % Stance, and % Swing respectively. These were compared to 12.92%, 11.75%, and 9.56% generated by ANN. Hence, it can be deduced that ELM is less prone to over-fitting and can generalize function better than the ANN. The reason is that when the training RMSE is small, the function produced is almost exactly dedicated to the training set. So, if unseen data is being tested on the model, it will most likely not show appropriate results if the data is not similar to the data trained. This can be seen in Table 4 where the neural network is being validated with unknown data (data from 3 out of 18 subjects were used in validation without training). The results further strengthen the view of ELM being a better generalizer with the validation RMSE of % Stance and % Swing being closer to the testing RMSE. The ELM performed relatively poorer for gait speed recognition during validation. Furthermore, it is more difficult to do a parameter search for ANN as the training duration is much longer than ELM. In addition to that, there are more parameters to be tuned (e.g. learning rate) for the ANN which can either be an advantage or a disadvantage. The advantage is that the function fits properly with the dataset while the disadvantage is that complexity brings inefficiency which leads back to a longer training duration and a lower performance on unseen data. Though, ANN triumphs in having a better training

performance but in the end, the testing performance is a major consideration as better testing performance will depict a more robust model.

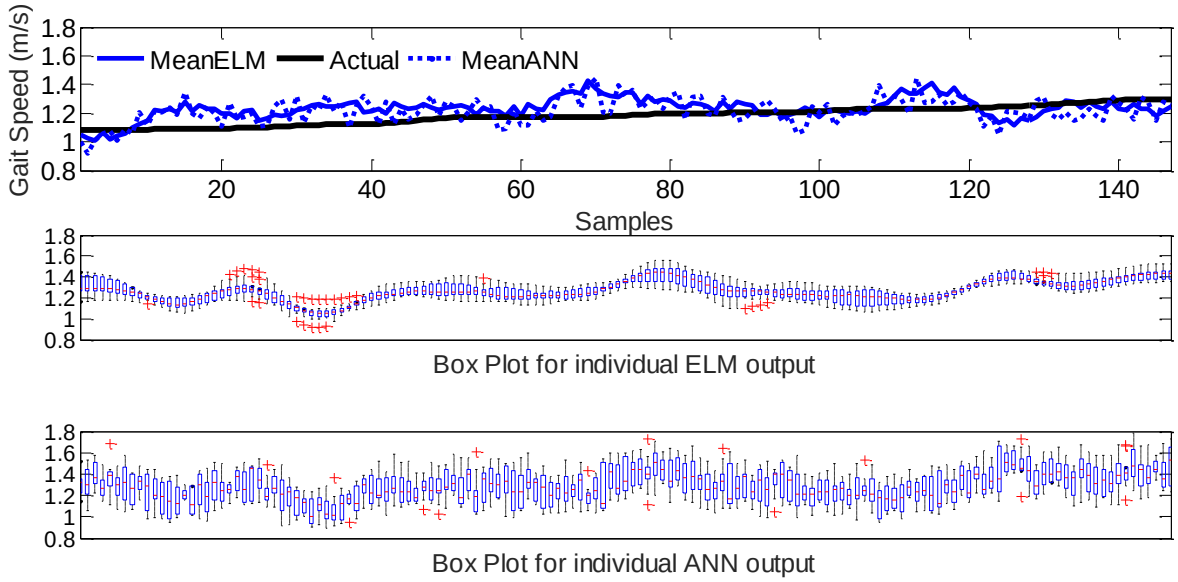

Figure 25: Comparison between the actual target, and ELM and ANN max, min predicted values for 250 hidden neurons (gait speed). Box plots for individual ELM and ANN output are shown in the bottom two plots.

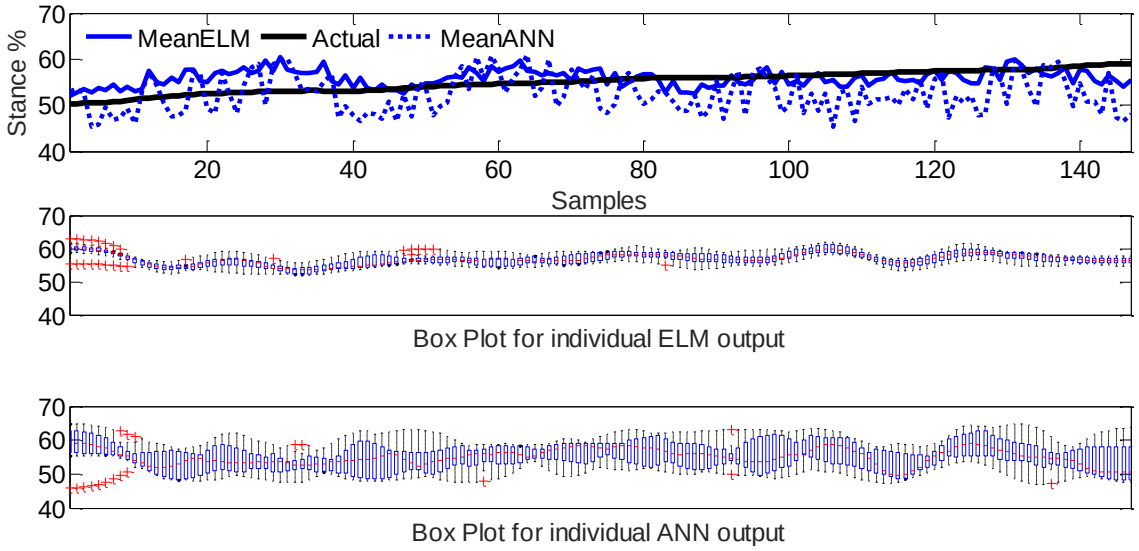

Figure 26: Comparison between the actual target, and ELM and ANN max, min predicted values for 250 hidden neurons (%Stance). Box plots for individual ELM and ANN output are shown in the bottom two plots.

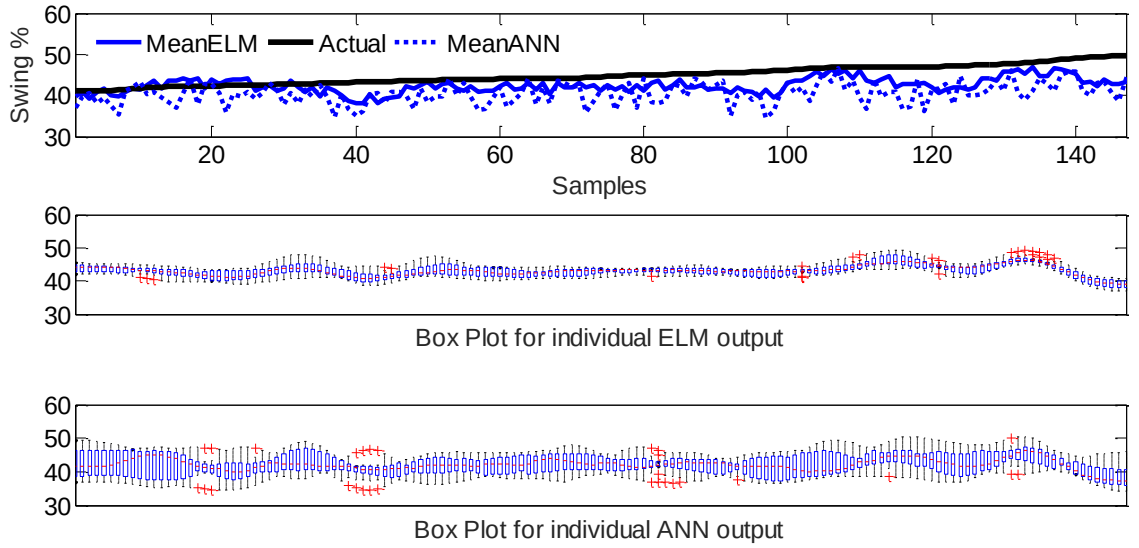

Figure 27: Comparison between the actual target, and ELM and ANN max, min predicted values for 250 hidden neurons (%Swing). Box plots for individual ELM and ANN output are shown in the bottom two plots.

Figure 25 to 27 shows the comparison between the actual target and the maximum, minimum predicted values by ELM, and ANN for 250 hidden neurons. Box plots for individual ELM and ANN output (total of 10 outputs for each sample/target data) are also included in Figure 25 to Figure 27. The red marker represents the outlier of the 10 data and the blue color box represents the data in the interquartile range (between the first and third quartile). The reason for choosing 250 hidden neurons is because it shows the lowest RMSE or mean estimation error when compared with other amounts of neurons used. Results showed that the ELM outputs have less variation in value (i.e. smaller interquartile range in the box plot) whereas the variation range is large for ANN outputs. ELM mean estimated gait parameters were also closer to the actual target.

Overall, the estimation errors of both ELM and ANN models are still large. This might be due to using EMG signal as the only input data. Further improvement could be made by extracting useful features from EMG signals as other inputs to ELM and ANN models.

4.2 *Experiment Type 2*

Using the data acquired from the previous experiments, with the processed data, the EMG signals and the angular velocity values from the gyroscope were synchronized so that both signals can have the same starting point. From the angular velocity, negative peaks as shown in Figure 28 as these peaks are the heel strike and toe-off location according to [11]. The index of the locations was stored in an array and the heel strike and toe-off index on the EMG signal were boosted by a certain value or amplifier. In the simulation, the amplifier used was 2000times. A window was used to arrange them for a usable input for the ELM. The window size was tuned to improve the accuracy of the detection.

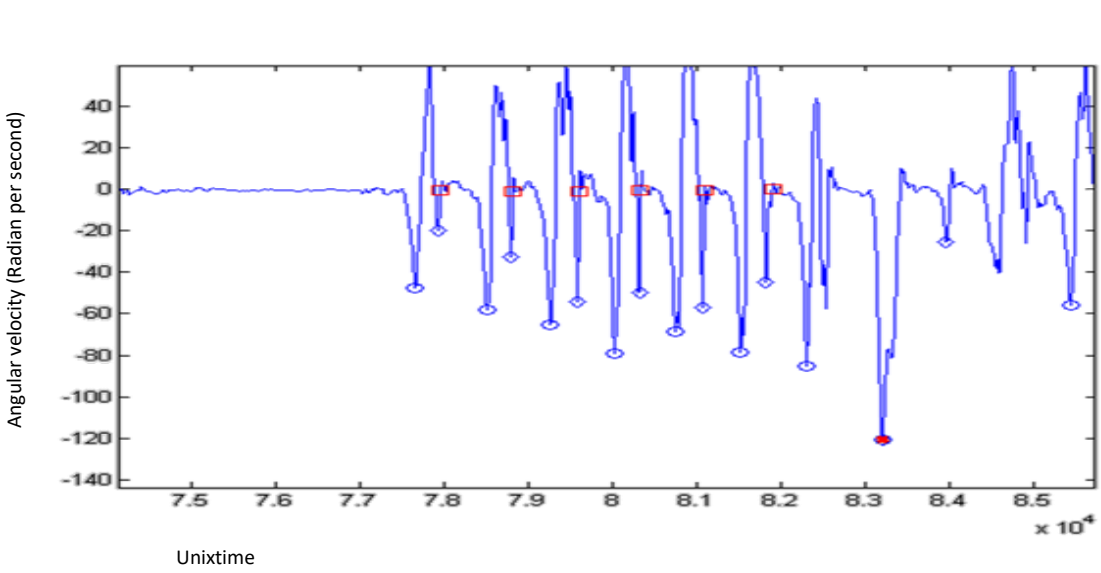

Figure 28: Angular velocity from gyroscope

Hence to improve on this, the training and testing set converted into a time series prediction.

Input signal=

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ \vdots & \vdots & \vdots & \vdots \\ n & n+1 & n+2 & n+3 \end{bmatrix} \quad (50)$$

target=

$$\begin{bmatrix} 5 \\ 6 \\ 7 \\ \vdots \\ n+4 \end{bmatrix} \tag{51}$$

For the training set, instead of labeling the endpoints as 1, it is labeled as a very high value e.g. 1000.

new input signal=

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1000 \\ 3 & 4 & 5 & 6 \\ \vdots & \vdots & \vdots & \vdots \\ n & n+1 & n+2 & n+3 \end{bmatrix} \tag{52}$$

new target=

$$\begin{bmatrix} 1000 \\ 6 \\ 7 \\ \vdots \\ n+4 \end{bmatrix} \tag{53}$$

Points 5, 12, and 15 are still the endpoints here.

Below is a picture of before and after: (1) original signal and (2) labeled signal

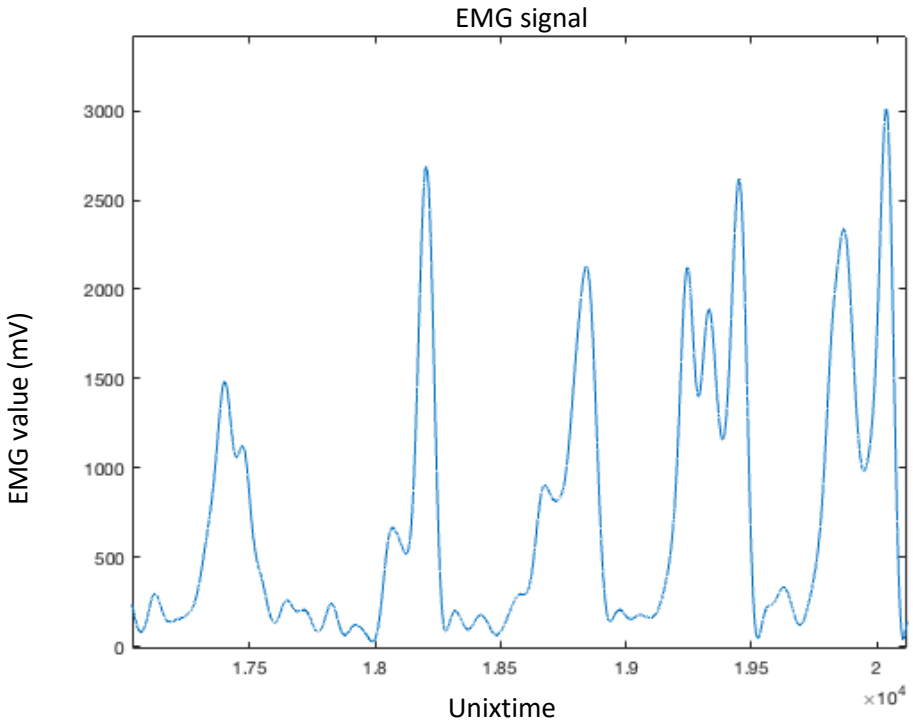

Figure 29: Original EMG signal

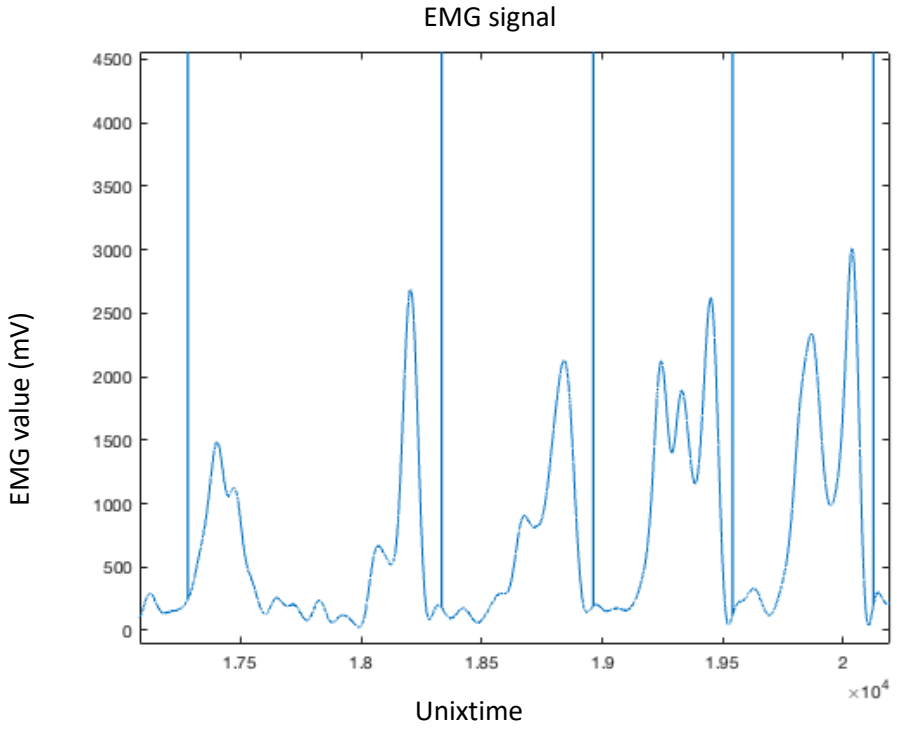

Figure 30: Labelled EMG signal

For the testing set, the testing signal, in this case, would be the original signal but in time-series form meaning that the endpoint locations are not replaced by high values. So, if this is computed in the Weighted ELM to be trained and then tested, the output for testing would look like this:

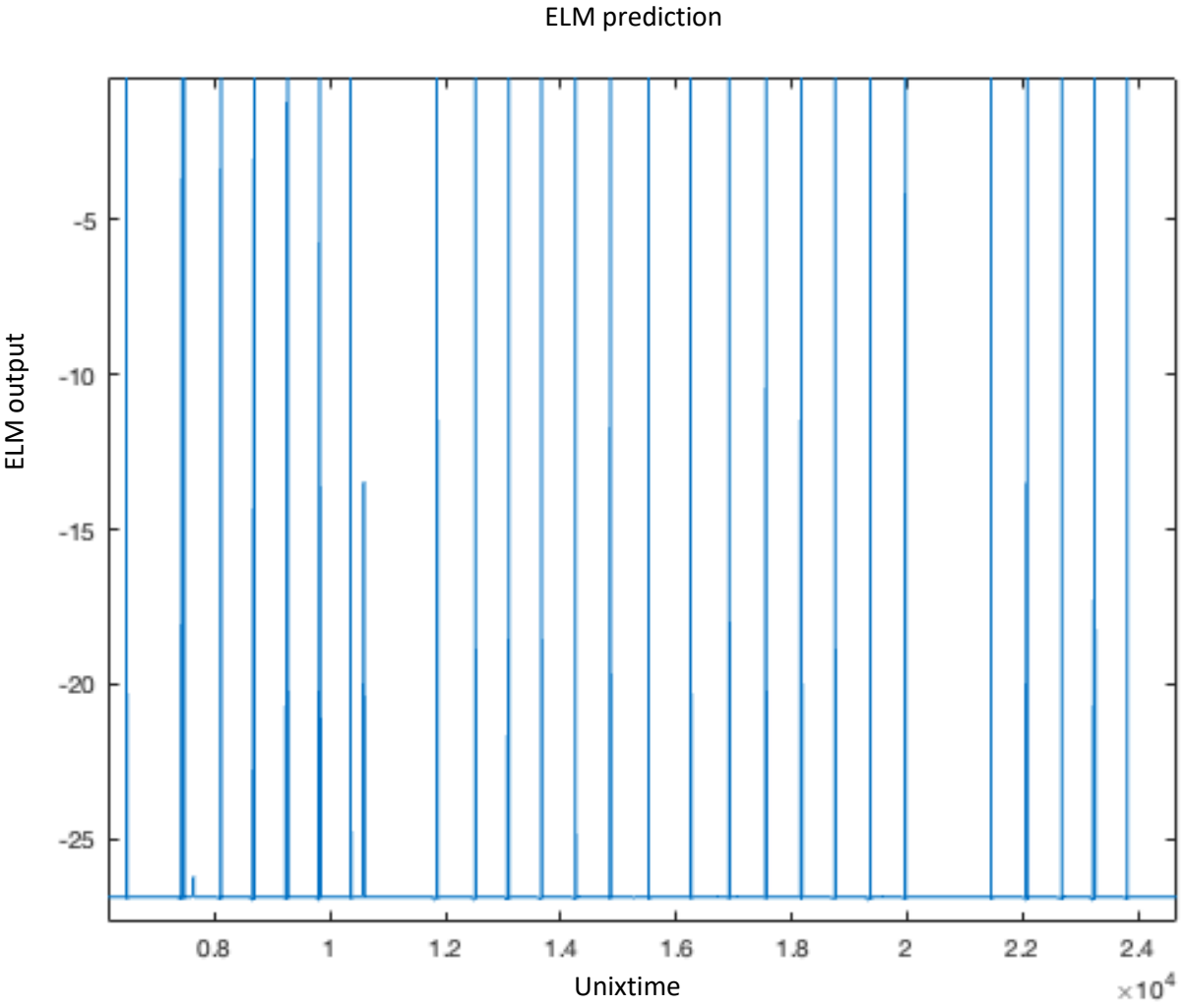

Figure 31: Output of ELM

Here, a threshold is set where if there is a peak above -10, then the x value of that peak will be the location of the endpoints. By normal convention, the supposedly testing output should look similar to the EMG pattern but in this study, only the endpoints are needed, so the pattern outputted in between the endpoints can be ignored.

To compare the testing output and the testing target, below is a figure of the testing target:

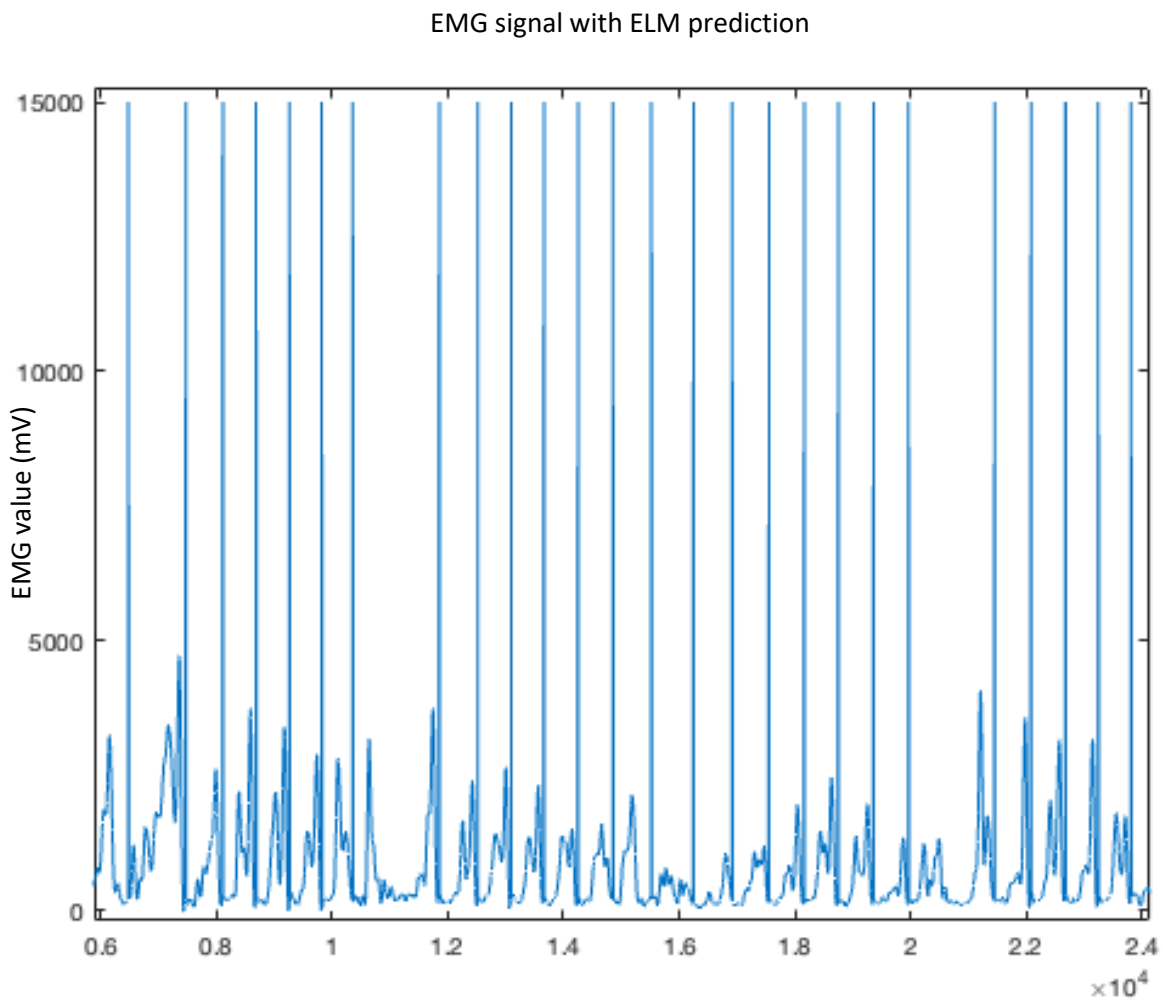

Figure 32: Output of testing target

The lines are where the endpoints are.

For hidden neurons, tests from 10, 50, 100...500 amount were done. As for the regularisation coefficient, it was tested with values ranging from $2^{-15}, 2^{-10} \ldots . 2^{15}$. And for the activation function, the ELM was tested with sigmoid, sine, and hard limit. In the end, it is concluded that 10 hidden neurons, 2^0 regularisation coefficient and sigmoid appeared to be the most viable parameters. The reason as to why 10 hidden neurons was chosen was because the lesser the amount of hidden neurons used, the simpler the model was and the faster the processor could compute. Below (Table 6) are the results for 21 subjects:

Table 6: Classification results

Subject	Training Time (s)	Testing Time(s)	Classification Rate (%)
1	8.5623	0.08	100
2	6.9715	0.08	100
3	6.4172	0.1	100
4	6.7793	0.09	100
5	7.9708	0.08	100
6	8.0556	0.08	100
7	9.185	0.48	98.3607
8	9.4231	0.39	100
9	6.3418	0.11	100
10	5.9835	0.1	98.7013
11	7.447	0.09	100
12	8.1492	0.07	100
13	7.1918	0.06	97.1831
14	7.107	0.1	92.6471
15	7.5556	0.08	100
16	7.265	0.06	100
17	6.9111	0.08	100
18	6.9908	0.06	100
19	7.7648	0.1	100
20	7.565	0.06	100
21	6.7434	0.08	100
Average	7.4467	0.1157	99.3758

If it is tested with 1 second of the signal, then the testing time would take 0.1 seconds. The above was tested with 100 seconds of the signal. Compared to the previous method that took more than 5 seconds, this would be considered an improvement in terms

of accuracy and testing time. The drawback of this method is that values of the peak of the testing output were not known as it differed depending on the value of input neurons when it was first initialized as the values were randomized. Hence, it is recommended to normalize from finding the max value within the first three peaks.

Another barrier is the absence of supporting evidence for the high value used on the EMG signal. Hence, further improvements to research on ways of standard statistics or algorithms to map or transform the value are needed.

4.3 Experiment Type 3

Table 7: Results for subject 1

Subject1			
Gait	RMSE Training (m)	RMSE Testing (m)	Training time(s)
1	0.1123	0.0472	
2	0.0665	0.0395	
3	0.0757	0.0975	
4	0.0620	0.0494	
5	0.0370	0.0503	0.32
6	0.0400	0.2400	
7	0.0878	0.1126	
8	0.1312	0.0972	
9	0.0485	0.0843	
Average RMSE	0.0734	0.0909	

Table 8: Results for different amount of hidden neurons and segmented parts

L hidden neurons		5			8			12	
Average of	Training RMSE (m)	Testing RMSE (m)	Training speed (s)	Training RMSE (m)	Testing RMSE (m)	Training speed (s)	Training RMSE (m)	Testing RMSE (m)	Training speed (s)
n parts									
2	0.0956	0.1578	0.28	0.1164	0.1496	0.28	0.1342	0.1769	0.28
4	0.0937	0.1305	0.28	0.088	0.1285	0.29	0.1297	0.1338	0.30
8	0.0734	0.0909	0.32	0.0837	0.1293	0.33	0.0921	0.1074	0.33

Table 9: Results for ELM without boosting

Subject	Average RMSE Training (m)	Average RMSE Testing (m)	Average Step Length (m)	Average Training time(s)
1	0.1375	0.1276	0.937	0.25
2	0.1350	0.1417	0.992	0.23
3	0.1300	0.1246	1.080	0.26
4	0.1397	0.1252	1.080	0.25
5	0.1610	0.1640	0.941	0.27
6	0.1370	0.1428	1.053	0.26
7	0.1061	0.1262	1.097	0.30
8	0.1239	0.1376	0.980	0.27
9	0.1247	0.1207	0.997	0.29
10	0.1313	0.1037	1.080	0.32
11	0.1400	0.1627	1.063	0.25
12	0.1366	0.1295	1.080	0.23
13	0.1307	0.1582	1.011	0.24
14	0.0983	0.1251	1.034	0.28
15	0.1338	0.1553	1.016	0.23
16	0.1202	0.1764	0.997	0.26
17	0.1275	0.1205	1.027	0.24
18	0.1202	0.0924	0.976	0.30
19	0.1040	0.1261	1.021	0.23
20	0.1133	0.1287	1.107	0.27
Average over 20 subjects	0.1275	0.1344	1.028	0.26

Table 10: Results for ELM with boosting

Subject	Average RMSE Training (m)	Average RMSE Testing (m)	Average Step Length (m)	Average Training time(s)
1	0.0734	0.0909	0.937	0.32
2	0.1421	0.1592	0.992	0.32
3	0.1285	0.1219	1.080	0.25
4	0.0807	0.1070	1.080	0.32
5	0.1344	0.1364	0.941	0.29
6	0.0831	0.0958	1.053	0.31
7	0.0747	0.0802	1.097	0.27
8	0.0823	0.0857	0.980	0.29
9	0.0722	0.1028	0.997	0.30
10	0.1213	0.1010	1.080	0.43
11	0.1172	0.0837	1.063	0.28
12	0.0930	0.0919	1.080	0.31
13	0.0840	0.1041	1.011	0.38
14	0.0975	0.1196	1.034	0.27
15	0.0746	0.0877	1.016	0.35
16	0.1323	0.1104	0.997	0.30
17	0.0831	0.0795	1.027	0.28
18	0.1228	0.1432	0.976	0.41
19	0.0737	0.1134	1.021	0.29
20	0.0927	0.0953	1.107	0.37
Average over 20 subjects	0.0982	0.1055	1.028	0.32

Table 11: Summary of Results for ELM with boosting

Average number of parts	Average RMSE Training (m) of 20 subjects	Average RMSE Testing (m) of 20 subjects	Average Training time (s)
5 hidden neurons			
2	0.1282	0.1150	0.20
4	0.1257	0.1402	0.24
8	0.1084	0.1198	0.24
8 hidden neurons			
2	0.1091	0.1986	0.21
4	0.1086	0.1128	0.30
8	0.0982	0.1055	0.32
12 hidden neurons			
2	0.1183	0.1142	0.31
4	0.1011	0.1268	0.29
8	0.1069	0.1322	0.34

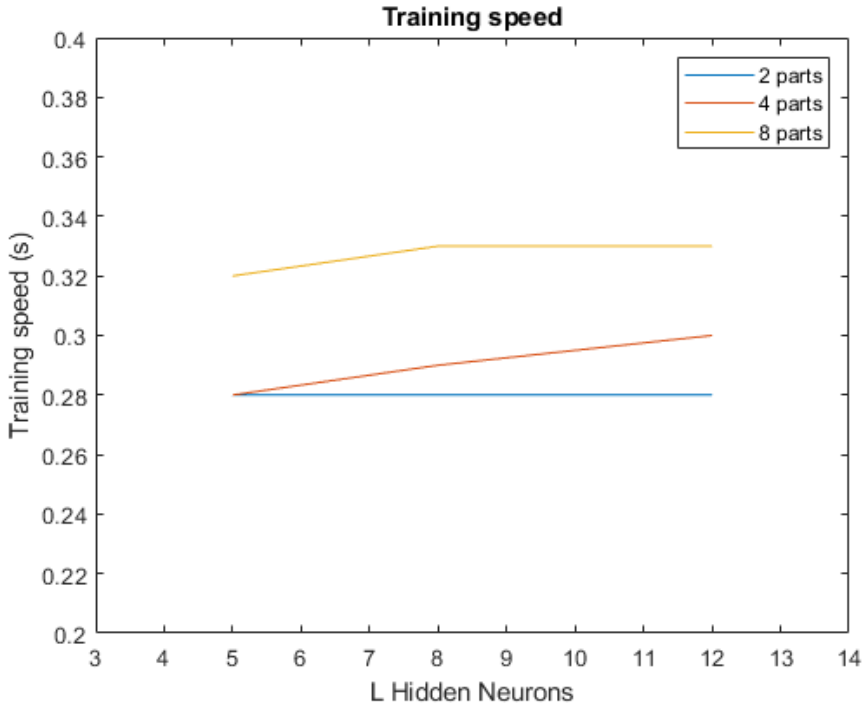

Figure 33: Training speed for L Hidden Neurons

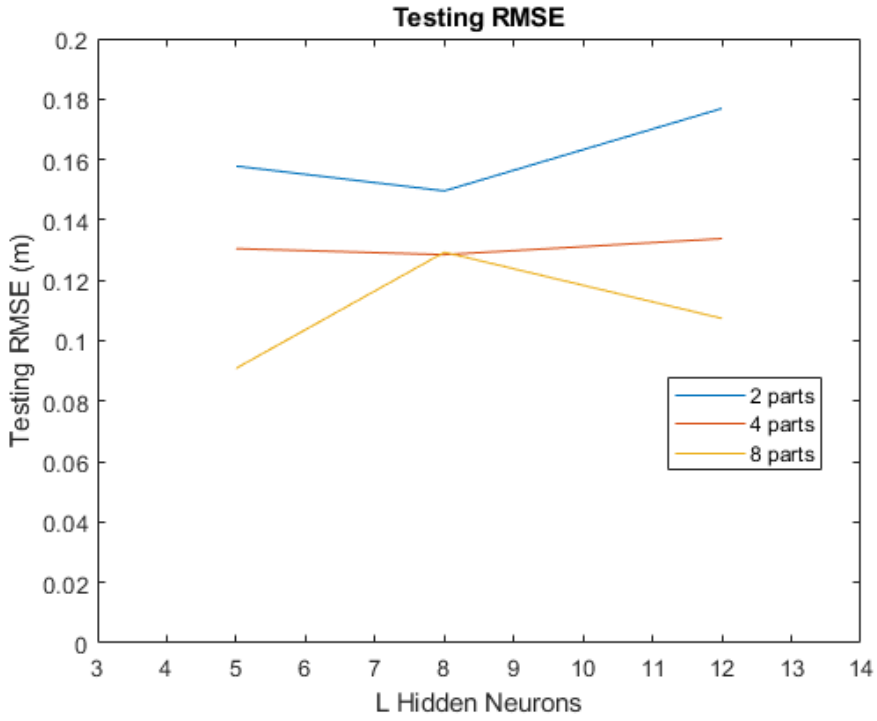

Figure 34: Testing RMSE for L Hidden Neurons

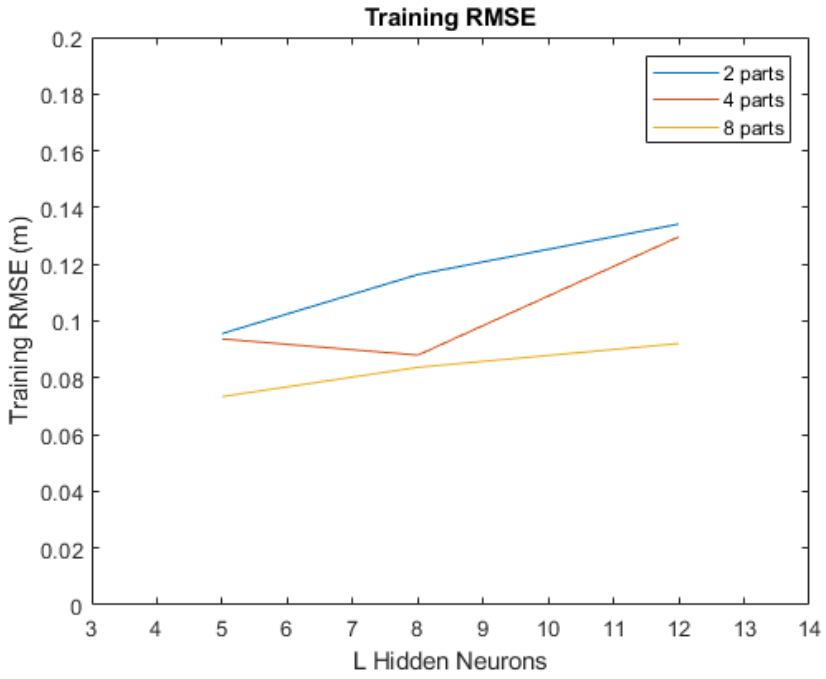

Figure 35: Training RMSE for L Hidden Neurons

Table 7 shows the results extracted from subject 1. There are no patterns observable in both training RMSE and testing RMSE but when compared to other models trained with different amount of hidden neurons and segmented parts, as shown in Table 8, the training and testing RMSE decreases with increasing segmented parts. It is the same trend when the number of hidden neurons are fixed and the number of segmented parts increases but when the number segmented parts are fixed, with increasing number of hidden neurons, training RMSE increases with the number of hidden neurons but no observable trend in the case of testing RMSE. According to the results presented in Table 11 and Figure 33, 34, and 35, the proposed ensemble strategy can be considered as a fast and accurate model for predicting x-trajectory. The fast training can be attributed by its base learners which are ELM models as the ELM algorithm itself is not required to tune the input weights and the output weights can be computed analytically which leads to a fast training phase. Nonetheless, the accuracy of our method can be interpreted with two factors. According to Bartlett's theory, the output weight β^m is one of the solutions that can minimize the weights norm and the output error, and hence, leading to better generalization performance. Furthermore, [16] also proved that an ensemble always performs better than a single model. The ensemble only requires two hyperparameters:

the number of hidden neurons L and the number of parts n. Table 9 and Table 10 have proven that with a small training set of 10 gait cycles, the ensemble of ELMs is still able to maintain a decent RMSE while still having fast training speed and this makes our approach very suitable for real-time training and testing which is planned for future work. To understand the behavior of the ensemble ELMs in term of training RMSE, testing RMSE, and training speed, Figure 33, 34, and 35 show with respect to the number of L hidden neurons and the number of n parts. Figure 33 shows that training speed increases with the number of parts as well as the number of hidden neurons and this can be explained with the increase in the calculation when the number of parts and hidden neurons increases. Besides that, based on Figure 35, there is a visible increasing training RMSE when the number of hidden neurons increases while the RMSE improves with increasing parts. The effectiveness of a model depends on its testing results and Figure 34 presented an increase in accuracy with increasing parts while no trend can be observed with an increasing amount of hidden neurons. When the number of hidden neurons is closer to the number of features, the training and testing RMSE also decreases and it indirectly leads to an increase in accuracy.

CHAPTER 5 - CONCLUSION

In this study, ANN and ELM models are compared when trained and tested on a gait dataset where the EMG signal served as the input while the parameters – gait speed, % Stance, and % Swing phase – serves as the target. Overall, ELM performs better in terms of predicting the values and it takes a shorter duration to train than the ANN. ELM also shows a more balanced value approximation as there is less difference between the training RMSE and the testing RMSE which gives rise to a robust model when compared with ANN. As for the segmentation classification ability of the ELM, it was able to achieve an average accuracy of 99.38%. There is also a difference in accuracy and training speed based on the amount of segmented parts as well as the number of hidden neurons where the training speed increases with the number of segmented parts and the number of hidden neurons. Training and testing RMSE become lowered with increasing segmented parts. From the observation of the results, the boosted Ensemble ELM is able to predict the x-trajectory within close accuracy. However, the ensembled ELM model faced certain deficiencies during the study. One is that as the number of segmentation parts increase, the individual ELMs are more prone to over-fitting due to the decreasing amount of data available for it to train on. That could be improved through higher resolution or sampling rate of data. Another drawback is that a good predictive model requires an amount of trial and error due to the random nature of hidden node initialization. Future work should focus on the further development of ELM model to achieve a more accurate estimation of gait trajectory.

REFERENCE

[1] J. Anderson, "Increasing the Acceptance of Clinical Information Systems Ⅱ", *MD Computing*, pp. 62-65, 1999.

[2] P. Beynon-Davies, M. Lloyd-Williams, "When health information systems fail", *Topics Health Inform Manage*, 20(1), 66-79, 1999.

[3] G. B. Huang, Q. Y. Zhu and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks", *Proc. IEEE IJCNN*, Vol. 2, pp. 985 – 990, 2004.

[4] H. Liu, J. Li and L. Wong, "Use of Extreme Patient Samples for Outcome Prediction from Gene Expression Data", *Bioinformatics*, Vol. 21, No. 16, pp. 3377-3384, 2005.

[5] S. A. Mahmoud and S. O. Olatunji, "Automatic Recognition of Offline Handwritten Arabic (Indian) Numerals Using Support Vector and Extreme Learning Machines", *Int. J. Imaging and Robotics*, Vol. 2, No. A09, pp. 34 – 53, 2009.

[6] S. O. Olatunji, A. Selamat and A. A. A. Raheem, "Modelling Permeability Prediction Using Extreme Learning Machines", *Proc. 4th Asia Int. Conf. Mathematical/Analytical Modelling and Computer Simulation (AMS '10)*, pp. 29-33, 2010.

[7] S. O. Olatunji, "Comparison of Extreme Learning Machines and Support Vector Machines on Premium and Regular Gasoline Classification for Arson and Oil Spill Investigation", *Asian J. Eng., Sci. & Technol.*, Vol. 1, No. 1, pp. 1-7, 2011.

[8] S. O. Olatunji, I. A. Adeleke and A. Akingbesote, "Data Mining Based on Extreme Learning Machines for the Classification of Premium and Regular Gasoline in Arson and Fuel Spill Investigation", *Journal of Computing*, Vol. 3, No. 3, pp. 130-136, 2011.

[9] S. O. Olatunji, Z. Rasheed, K. A. Sattar, A. M. Al-Mana, M. Alshayeb and E. A. El-Sebakhy, "Extreme Learning Machine as Maintainability Prediction model for Object-Oriented Software Systems", *Journal of Computing*, Vol. 2, No. 8, pp. 49 – 56, 2010.

[10] M. A. Oskoei and H. Hu, "Myoelectric Control Systems-A Survey," Biomedical Signal Processing and Control, vol. 2, no. 4, pp. 275-294, Oct 2007, Available:10.1016/j.bspc.2007.07.009.

[11] P. Parker, K. Englehart, and B. Hudgins, "Myoelectric Signal Processing for Control of Powered Limb Prostheses," Journal of Electromyography and Kinesiology, vol. 16, no. 6, pp. 541-548, Dec 2006, Available: 10.1016/j.jelekin.2006.08.006.

[12] E. A. Clancy, E. L. Morin, and R. Merletti, "Sampling, Noise reduction and Amplitude Estimation Issues in Surface Electromyography," Journal of Electromyography and Kinesiology, vol. 12, no. 1, pp. 1-16, Feb 2002, Available:10.1016/S1050-6411(01)00033-5.

[13] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG Signal Analysis: Detection, Processing, Classification and Applications," Biological Procedures Online, vol. 8, no. 1, pp. 11- 35, Mar 2006, Available:10.1251/bpo124.

[14] M. Zardoshti-Kermani, B. C. Wheeler, K. Badie, and R. M. Hashemi, "EMG Feature Evaluation for Movement Control of Upper Extremity Prostheses," IEEE Trans. Rehabilitation Engineering, vol. 3, no. 4, pp. 324-333, Dec 1995, Available:10.1109/86.481972.

[15] A. Phinyomark, C. Limsakul, and P. Phukpattaranont, "A Comparative Study of Wavelet Denoising for Multifunction Myoelectric Control," Proc. Int. Conf. Computer and Automation Engineering (ICCAE '09), pp. 21-25, Mar. 2009, Available:10.1109/IC CAE.2009.57.

[16] A. Phinyomark, C. Limsakul, and P. Phukpattaranont, "EMG Denoising Estimation Based on Adaptive Wavelet Thresholding for Multifunction Myoelectric Control," Proc. Conf. Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA '09), pp. 171-176, July 2009, Available:10.1109/CITISIA.2009. 5224220.

[17] R. Ortolan, R. Mori, R. Pereira, C. Cabral, J. Pereira and A. Cliquet, "Evaluation of adaptive/nonadaptive filtering and wavelet transform techniques for noise reduction in EMG mobile acquisition equipment", IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 11, no. 1, pp. 60-69, 2003. Available: 10.1109/tnsre.2003.810432.

[18] R. Merletti and P. Parker, "Electromyography Physiology, Engineering, and Noninvasive Applications", John Wiley Sons, Inc., New York, 2004.

[19] Y. Zhou, R. Chellappa, G. Bekey, "Estimation of intramuscular EMG signals from surface EMG signal analysis", IEEE International Conference on Acoustics, Speech and Signal Processing, v11, pp.1805-1808, 1986.

[20] P. Carre, H. Leman, C. Fernandez, C. Marque, "Denoising of the uterine EHG by an undecimated wavelet transform," IEEE Trans. on Biomedical Engineering, vol. 45(9), pp. 1104-1113, 1998.

[21] S. Mallat, "A wavelet tour of signal processing", 2nd edition, Academic Press, 1998.

[22] A. Galka, "Topics in nonlinear time series analysis – with implications for EEG analysis", World Scientific Publishing, 2000.

[23] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, J.D. Farmer, "Testing for nonlinearity in time series – the method of surrogate data", Physica D, vol 58, pp. 77-94, 1992.

[24] J.S. Bendat, A.G. Piersol, "Random data – Analysis and Measurements", Wiley, 2nd edition, 1991.

[25] F. Takens, "Detecting strange attractors in turbulence," Lecture notes in Math., vol. 898, 1981.

[26] A.M Fraser., H.L. Swinney, "Independent coordinates for strange attractors from mutual information", Phys. Rev. A, vol33, pp.1134- 1140, 1986.

[27] J.P. Eckman, S.O. Kamphorst, D. Ruelle, "Recurrence plots of dynamical systems", Europhys. Lett., vol4, no. 9, pp.973-977, 1987.

[28] G. J. Small, N. B. Jones, J. C. Fothergill, A. P. Mocroft, "Chaos as a possible model of electromyographic activity," IEE Intl. Conf. On Simulation, pp. 27-34, 1998.

[29] S. Haykin, S. Puthusserypady, "Chaotic dynamics of sea clutter", Wiley Interscience, 1999.

[30] R. Hegger, H. Kantz, and T. Schreiber, "Practical implementation of nonlinear time series methods: The TISEAN package," Chaos, vol. 9, pp. 413, 1999.

[31] M. T. Hagan, H. B. Demuth and M. H. Be, "Neural Network Design", Boston: PWS pub, 1996.

[32] H. Jaeger, "A tutorial on training recurrent neural networks, covering BPPT, RTRL,

EKF and the "echo state network" approach," Fraunhofer Institute for Autonomous Intelligent Systems, Bermen, 2002.

[33] D. A. Winter, "Biomechanics and Motor Control of Human Movement", New Jersey: Wiely & Sons Inc., 2009.

[34] E.R. David, E.H. Geoffrey, J.W. Ronald. "Learning representations by back-propagating errors", Nature International Weekly Journal of Science, 1986.

[35] Lin Wang and T. Buchanan, "Prediction of joint moments using a neural network model of muscle activations from EMG signals," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 10, no. 1, pp. 30-37, 2002.

[36] I. Batzianoulis, S. El-Khoury, S. Micera and A. Billard, "EMG-Based Analysis of the Upper Limb Motion," in Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts, New York, 2015.

[37] J.-S. R. Jang, C.-T. Sun and E. Mizutani, "Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence", Upper Saddle River: Prentice-Hall. Inc., 1997.

[38] F. Chan, Yong-Sheng Yang, F. Lam, Yuan-Ting Zhang and P. Parker, "Fuzzy EMG classification for prosthesis control", IEEE Transactions on Rehabilitation Engineering, vol. 8, no. 3, pp. 305-311, 2000. Available: 10.1109/86.867872.

[39] S. Micera, A. Sabatini, P. Dario and B. Rossi, "A hybrid approach to EMG pattern analysis for classification of arm movements using statistical and fuzzy techniques", Medical Engineering & Physics, vol. 21, no. 5, pp. 303-311, 1999. Available: 10.1016/s1350-4533(99)00055-7.

[40] R. O. Duda, P. E. Hart and D. G. Stock, "Pattern Classification", 2nd Edition, Wiley-Interscience, 2001.

[41] G. B. Huang, Q. Y. Zhu, K. Z. Mao, C. K. Siew, P. Saratchandran and N. Sundararajan, "Can threshold networks be trained directly?", *IEEE Trans. on Circuits and Syst. II*, Vol. 53, No. 3, pp. 187-191, 2006.

[42] H. Cao, S. Sun and K. Zhang, "Modified EMG-based handgrip force prediction using extreme learning machine", Soft Computing, 21(2), pp.491-500, 2015.

[43] N. Sezgin, "EMG classification in obstructive sleep apnea syndrome and periodic limb movement syndrome patients by using wavelet packet transform and extreme learning machine", Turkish Journal of Electrical Engineering & Computer Sciences, 23, pp.873-884, 2015.

[44] K. Anam and A. Al-Jumaily, "Evaluation of extreme learning machine for classification of individual and combined finger movements using electromyography on amputees and non-amputees", Neural Networks, 85, pp.51-68, 2017.

[45] S. Adil, T. Anwar, Adel and A. Jumaily, "Extreme learning machine-based EMG for drop-foot after stroke detection", 2016 Sixth International Conference on Information Science and Technology (ICIST), 2016.

[46] S. Adil, A. Jumaily and K. Anam, "AW-ELM-based Crouch Gait recognition after ischemic stroke", 2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), 2016.

[47] N. Sezgin, "Nonlinear Analysis of Electrocardiography Signals for Atrial Fibrillation", The Scientific World Journal, 2013, pp.1-4, 2013.

[48] T Mantoro, A. K. Olowolayemo, S. O. Olatunji, M. A. Ayu and A. O. Md. Tap, "Extreme learning machine for user location prediction in mobile environment", *Int. J. Pervasive Computing and Commun.*, Vol. 7, No. 2, pp. 162 – 180, 2011.

[49] Seniam.org. Recommendations for sensor locations in lower leg or foot muscles. [online] Available at: http://seniam.org/lowerleg_location.htm [Accessed 11 Aug. 2017].

[50] P. Lopez-Meyer, G. Fulk and E. Sazonov, "Automatic Detection of Temporal Gait Parameters in Poststroke Individuals", *IEEE T. Info Technology. B.,* 15(4), pp.594-601, 2011

[51] Basic ELM Algorithms, Extreme Learning Machine, http://www.ntu.edu.sg/home/egbhuang/elm_codes.html. 2017 (accessed 24.05.17)

[52] K. Fukunaga, "Introduction to statistical pattern recognition". Academic press, 2013.