Individual Diploma Thesis

FEATURE SELECTION AND TRAINING OF NEURAL NETWORKS TO CLASSIFY FUNCTIONAL VERSUS DYSFUNCTIONAL COPING WITH ACUTE PAIN

Panagiotis Christodoulou

UNIVERSITY OF CYPRUS

DEPARMENT OF COMPUTER SCIENCE

May 2023

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Feature Selection and Training of Neural Networks to Classify Functional versus Dysfunctional Coping with Acute Pain

Panagiotis Christodoulou

Supervisor
Chryssis Georgiou

The Individual Diploma Thesis was submitted for partial fulfilment of the requirements for the degree of Computer Science of the Department of Computer Science of the University of Cyprus

May 2023

Acknowledgements

Foremost, I would like to express my deepest gratitude to my supervisor, Dr Chryssis Georgiou, Professor at the Department of Computer Science at the University of Cyprus, for trusting me on this topic and for the guidance and support throughout the entire process of my Diploma Thesis.

Also, I am extremely grateful to the Department of Psychology of the University of Cyprus, and especially Dr Maria Karekla and Pinelopi Konstantinou for providing us with the experimental data samples that we used.

Furthermore, I would like to thank Andria Trigeorgi who provided me with important knowledge about the previous work on the topic and gave me continuous guidance and feedback.

Finally, I am deeply grateful to my friends and family for their encouragement and support throughout this journey.

Abstract

During the last years, there has been a rapid development of both wearable technologies and Artificial Intelligence, especially the part of Machine learning. A lot of different studies were conducted by the Department of Computer Science and Department of Psychology of University of Cyprus that use traditional machine learning algorithms to classify people as functional or dysfunctional coping with acute pain. These algorithms use data collected from an experiment in which the participants were asked to immerse their hands into an ice water container. The signals collected are psychophysiological signals, such as electrocardiogram (ECG), electrodermal activity (EDA), and facial electromyography (fEMG).
The main aim of this study is to substitute these traditional machine learning algorithms with Neural Networks (NN) as they have become popular the last years.

Firstly, a study and a classification of Neural Networks takes place and how they have been used in the past in related studies. Based on the study, we have identified several NNs to be used, namely MultiLayer Perceptron (MLP), Radial Basis Function (RBF), SelfOrganizing Map of Kohonen, the Long-Short Term Memory (LSTM) and a combination of the MLP, RBF and LSTM. The same feature selection is followed as in the other study because it is generic, and it seems to be working with NN as well. Different data multiplication techniques are used to multiply the dataset like the Rectangular Window Methodology and the Moving Window Methodology and different activation functions are used in the NNs. The Best NN is the **RBF** with **Tanh** activation function in the **Moving Window Methodology,** yielding similar results when compared to the ones obtained by the traditional machine learning algorithms.

Finally, another experiment is considered where people are classified again as functional or dysfunctional, but the data is collected in real time via a wearable device. The data are similar to the previous experiment and the best NN algorithm is a **MultiLayer Perceptron,** achieving very similar results to the ones obtained by the traditional machine learning algorithms.

As an overall contribution, a more generic methodology is introduced on how to “replace” the traditional Machine Learning algorithms with Neural Networks.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Motivation

The past decade there has been an increase in the use of wearable devices like smartwatches and smart bands which are able to record multiple measures, including heart and sweat gland activity. These measures are psychophysiological signals and can reflect an individual's emotional arousal. Examples of psychophysiological signals are Electrocardiogram (ECG), Electrodermal Activity (EDA), and Facial Electromyography (fEMG).

A number of previous studies [19][20][21] analyzed such data concentrating on signals recorded from stationary devices, examined and used HRV time-domain features, that come from ECG, to train their models. While other studies [22] focused on wearable devices and gave more focus in feature selection techniques to achieve better results.

All these studies used some standard *machine learning* models to do their predictions, like Random Forest [40] and AdaBoost [37] and did not consider using **Neural Networks (NN)** [1]**,** which is another popular type of machine learning technique. This study aims to investigate further this family of algorithms and how they can assist health care.

1.2 Goals of the Study

This study uses data collected from an experiment regarding pain management techniques that was conducted by the Department of Psychology of the University of Cyprus. The

ultimate goal of the present thesis is to contribute to the integration of a form of psychotherapy called *Acceptance and Commitment Therapy (ACT)* [23] in the everyday life, which encourages people to try to deal with their feelings and thoughts instead of ignoring them or blaming themselves. ACT is a vital therapy and really useful nowadays for a lot of people who struggle with OCD, anxiety, depression etc.

People are separated into the 'acceptance' or the 'avoidance' group, based on their reactions at a certain time. The 'acceptance' group, also known as the 'functional', contains people who accept their problems and try to face and overcome them. On the contrary, the 'avoidance' or 'dysfunctional' group involves people who avoid their thoughts and feelings usually by thinking themselves in another situation or by spending time on something different, like work or studying. An individual does not always fall into the same category as their classification changes based on the environment, the circumstances and how they act at that exact moment.

This thesis aims to effectively classify individuals into functional or dysfunctional regarding pain coping via Neural Networks [1]. More specifically, focus is given on using signals recorded from wearables devices to train machine learning models of the *Neural Networks* category and compare the effect they have with the previous study that used *standard* machine learning algorithms [22].

1.3 Methodology

This study is a continuation of a previous study, so it is not done by scratch. This study uses the data and the feature selection that were introduced during the previous study [21][22].

First of all, in this thesis a study about Neural Networks is conducted in the form of a survey and then the most suitable NNs are selected. Following that, two different data multiplication techniques are selected to multiply the data as the dataset is very small. Then, the different networks are trained and tested on the datasets. During the training procedure a lot of different NNs are used and a lot of variation and combinations of different NN are used. Also, a lot of different parameters are tested in the NNs such as the number of hidden neurons and different activation functions. A more detailed analysis of the methodology is explained in Chapter 4.

1.4 Document Organization

The rest of this thesis is split into eight chapters. **Table 1.1** reports the content of each chapter.

Chapter Number	Chapter Description
2	Overview of the work done before and how the feature selection and machine learning were used. Also, it analyses how the models are evaluated.
3	Overviews the various types of Neural Networks algorithms and how are used in signal related applications.
4	Explain which Neural Networks were chosen and the methodology followed in this study.
5 - 8	Provide the analysis of the experiments made on the different NNs following the methodology of Chapter 4.
9	Summarizes the work done and suggests future improvements.

Table 1.1 Document Organization

Chapter 2

Previous Work

This chapter overviews the experiment and how previous studies dealt with psychophysiological signals and which machine learning algorithms were used.

2.1 Experiment Description

The experiment from which the dataset is collected is the 'Functional Versus Dysfunctional Coping with Acute Pain' [51]. In the study 80 people took part and the aim was to compare **acceptance** and **avoidance** coping strategies in a pain-induction experiment. Participants were split into 4 groups, conditions, in random and the participants of each group were given different instructions on how to deal with pain. The conditions were: (a) Acceptance followed by avoidance; (b) Avoidance followed by acceptance; (c) No instructions given (control) followed by acceptance and (d) No instructions given (control) followed by avoidance.

The experiment was composed of three timeframes. The first one lasted 5 minutes and was used as a baseline to make sure that the participant was in a state of calm. After that, participants were instructed on how to deal with pain based on their condition as described before. The second timeframe followed, in which participants were subjected to the Cold Pressor Task, CPT. In CPT each individual is asked to immerse their hand in a contained filled with cold water for as long as they could. Then they are instructed on how to deal

with pain, in the last time frame, based on their condition. In the final timeframe, participants we subjected to a second CPT. The maximum duration of the second and third timeframe was 3 minutes.

The measures reported are behavioral, psychophysiological, and self-reported during the whole procedure. Behavioral measures are pain threshold and pain tolerance, which are the number of seconds that passed from immersion until the participant reported pain verbally and until the participant removed their hand from the container respectively. The self-reported data are the questionnaires that the participants completed which examined various aspects, including their psychological condition and their use of pain-coping strategies. The psychophysiological signals and the devices used to collect them are explained in Sections 2.3 and 2.4, respectively.

2.2 Previous Studies

A lot of studies took place that analyzed the data from psychophysiological signals [19][20][21]. These studies focused on signals recorded from stationary devices. Later on, the study in [22] focused on wearable devices and followed another *feature selection* process to train its models. So, the current study is a continuation of the last one but using *Neural Network* algorithms instead of the traditional machine learning models.

Firstly, the Department of Psychology of the University of Cyprus conducted an experiment in the lab, where psychophysiological signals were recorded from stationary and wearable devices. These signals are Electrocardiogram (ECG), Heart Rate Variability (HRV), Facial Electromyography (fEMG), and Electrodermal Activity (EDA). After the experiment, the *Rectangular Window Methodology* (RWM) [24] was used to *multiply* the amount of data as can be seen in Chapter 5.1. Thereafter the data were cleaned from noise and a feature extraction took place based on time – domain and statistical features. In order to find the best features to train the models techniques from three different *Feature Selection* methods were applied, which are further explained in Chapter 2.6.1, to find the best features. After that, five different supervised binary classification Machine Learning algorithms were examined. These are *Adaptive Boosting, Gradient Boosting Decision Tree, Bagging Decision Tree, Random Forest, and Extra Trees* which are explained in Section 2.5.

2.3 Psychophysiological Signals

In the experiment conducted by the Department of Psychology, four types of signals were collected which are explained in the current section.

2.3.1 Electrocardiogram (ECG)

Electrical signals are produced when the heart beats. These signals can be recorded non-invasively from the body surface using an *electrocardiogram ECG* [25]. Three waves, P, QRS and T [26] comprise the basic pattern of this electrical activity as **Figure 2.1** shows. The QRS is a wave complex, meaning that it consists the waves Q, R and S [26] ECG signal can extract three categories of features, *frequency-domain, spectral* and *time-domain*. The *time-domain* measures are the most *sufficient* in the topic of this thesis as a previous study explains [36]. Time-domain measures are based on Heart Rate Variability, which is known as RR intervals, is the **time elapsed** between two consecutive R peaks. The R wave consists in the QRS as mentioned before. Heart Rate Variability ,HRV, can be estimated from this signal. HRV is equal to the distance between consecutive R-peaks of the ECG signal [29].

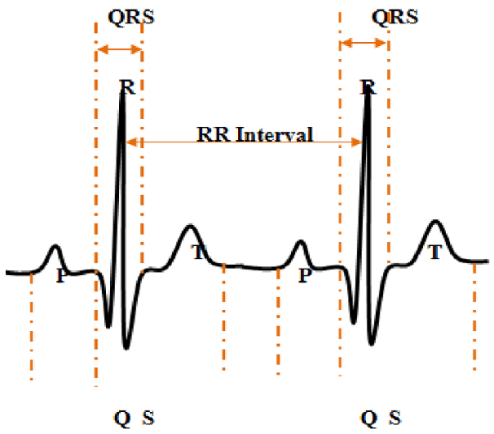

Figure 2.1 Visual Representation of ECG Signal [22]

2.3.2 Photoplethysmography (PPG)

From the beginning of a heartbeat to the beginning of the next one, cardiac cycle, the blood volume rises and falls throughout the body. The difference in blood volume can be

observed in the skin's outer layers and be measured using optical sensors [27]. Photoplethysmography (PPG) is a technology that uses a LED light source and a photodetector. The LED emits light into the microvascular bed of tissue and the photodetector, which is a light-sensitive sensor, records how much light is absorbed or reflected. The amount of light that is reflected or absorbed changes based on the blood volume [27][28].

2.3.3 Electrodermal Activity (EDA)

Electrodermal activity is the change of electrical properties of the skin follows sweating. It can reflect a person's *emotional state* or *emotional arousal* and can be measured non-invasively by applying an electrical potential between two points on the skin and measuring the current flow between them. It is linked to emotional arousal and clinical applications cover a lot of topics, including pain evaluation [30][31].

2.3.4 Facial Electromyography (fEMG)

Facial Electromyography, fEMG, is a technique that is used to detect emotional expressions by recording the movement of the muscles on the face. Each time a muscle contracts, a burst of electric activity is produced and propagated through adjacent tissue and bone, which can be recorded from neighbouring skin areas and zygomaticus major; ZYG, corrugator supercillii; COR, and orbicularis oculi; ORB, are the muscles whose activity is usually recorded [32]. ZYZG is placed on the region of cheek and is associated with *positive emotional stimuli*, while COR is linked with *negative reactions* [32][33]. The COR is found on the eyelids and its primary purpose is to close the eyelids [34]. fEMG is still able to detect muscle activation even when participants are instructed to hide their facial expressions [35].

2.4 Monitoring Devices

The devices that were used to record the variance of the physiological data of the participants are *BIOPAC MP150, Microsoft Band 2,* and *Moodmetric Smart Ring*. This

section gives a brief explanation for each one, the signals they record and the signals that are recorded in the experiment of Section 2.1.

2.4.1 BIOPAD MP15

BIOPAC MP150 [43] is a **stationary device**, installed and working in the Psychology lab. It can record a wide range of signals from 16 channels and has wide range of available sample rates, from 2 samples/hour to 200khz [44]. It was employed in the context of the research to record ECG and fEMG at sampling rate of 1000 HZ and it was used in combination with AcqKnowledge 3.9.0 [45] data acquisition software, which provides functionalities for quicker signal analysis. For the experiment analyzed in 2.1 it collected the ECG and part of the EDA signal, which is called SCL, and the fEMG with sampling frequencies of 1kHz, 250HZ and 1kHz respectively.

2.4.2 Microsoft Band 2

Microsoft Band 2 [46] is a smart band, a **wearable device**, is worn on wrist and is a hands-free gadget that can be worn in everyday life. It can record PPG in 1 Hz. Regarding the experiment it collected the PPG and EDA signals with sampling frequencies of 1Hz and 0.2 Hz respectively.

2.4.3 Moodmetric Smart Ring

Moodmetric Smart Ring [47] is a **wearable device** that can be worn on any finger of the hand and can measure EDA in sample rate of 3 Hz. It collected only the EDA signal with a sampling frequency of 3Hz for the experiment.

2.5 Machine Learning Algorithms

The machine learning algorithms that were used are *AdaBoost, Gradient Boosting Decision Tree, Bagging Decision Tree, Random Forest, and Extra Trees*.

2.5.1 AdaBoost Algorithm

AdaBoost Algorithm [37], Adaptive Boosting, uses a sequence of weak learners, which are models that perform better than random guessing. It assigns the same weight in each sample, as they are considered equally important. In the next iterations the weight of a sample increases if it was correctly classified or decreases if it was not. Thus, the algorithm focuses on data samples that are wrongly classified and this process is repeated until the desired accuracy is achieved or until a predefined number of learners are used. A Decision Tree with only one level is usually the learner that is used.

2.5.2 Gradient Boosting Decision Tree

Gradient Boosting Decision Trees (GBDT) [38] has three basic components, the weak learner, a loss function, and an additive model. In each iteration a new weak learner like a decision tree is added to the model, then the loss function is computed, which estimates how efficient the model is based on the desired and real output values of the classifier. In the next iteration, a new decision tree is added to the model to decrease the value of the loss function.

2.5.3 Bagging Decision Tree

Bagging Decision Tree is based on the *Bagging Ensemble technique* [39]. This technique combines a set of decision trees to create a result. Firstly, with a random subset of the training set, a collection of decision trees is created and when a new sample arrives, each one of them is used to classify it into a class. The class that was voted by the majority of the classifiers is the result of the algorithm.

2.5.4 Random Forest

Random Forest algorithm [40] belongs to the category of Bagging Ensemble algorithms. Each tree is created by selecting a set of attributes, without replacement, and a set of random samples, with replacement. This is repeated until a forest with a predefined

number of trees is created and the output class labels are based on the majority voting of the decision trees.

2.5.5 Extra Trees

Extra Trees algorithm [41] is similar to the Random Forest. The difference is that this algorithm chooses a random split while the Random Forest chooses the optimum split in a node, i.e., the optimum range of the node's feature value for each branch. Also, when the splits are performed for all candidate features, it chooses those with the best performance.

2.6 Model Evaluation

In the previous study [22] there was a really good feature selection process as the machine learning models achieved very good metrics. The metrics used in the previous and this study are explained in Chapter 4.

2.6.1 Feature Selection

For the feature selection in the previous study [22] three methods were used, the *Wrapper Method*, the *Embedded Method*, and the *Filter Method*. In the *wrapper method*, a *specific machine learning* algorithm is used and the whole space of possible features is searched to find the combination that yields the best performance. The *embedded method* uses classification algorithms that have built-in *feature selection functionality* [42]. The type that was used in the study utilizes inherent characteristics of decision tree algorithms, like Random Forest and classification and Regression Tree. *Filter Method* is more general and **does not relate** to any machine learning algorithm. They get the candidate features and return those that are more related to the target value.
The features that were selected are the ***mean peak amplitude*** and ***average value of heart rate*** as explained in [22]. As these features had good results in many different machine learning algorithms in the previous study and the filter method is used which is more generic and independent of the machine learning model it was decided to follow the same feature selection in the current study.

The ***mean peak amplitude*** is the average value of amplitudes in the window and is extracted from the EDA signals as [22] analyses.

The hr_***mean, average value of heart rate,*** is the average value of heart rate which is equal to the distance between consecutive R-peaks of the ECG signal as explained before.

2.6.2 Machine Learning Model Evaluation

The main metrics that were used to compare the Machine Learning models are **Accuracy**, which is the rate of the correct predictions in all the dataset, **Sensitivity** and **F1-score** which indicate how well it can predict the avoidance. These metrics are explained in **Section 4.2.** The best results are yielded with the ***Bagging Decision Tree*** with average **accuracy** of ***85%.***

Chapter 3

An Overview of Neural Networks

In this chapter we are going to make an *introduction* to *Neural Networks* and their main *categories*. They are separated into the *Basic Types* which are explained in Section 3.2 and the *Specialised Types* which are explained in Sections 3.3 to 3.6.

3.1 Neural Networks

In this thesis we are going to "replace" the *standard machine algorithms* that were used to classify *functional* versus *dysfunctional* coping with acute pain with Neural Networks which is explained in Chapter 2.

An *Artificial Neural Network* (ANN) is a biologically inspired **computational model** which is consisted of *neurons*, processing elements, and the *weights* , which are the coefficients of the edges, connections, between the neurons [1]. Neural Network modelling is like fitting a line, plane, or hyperplane, based on the number of dimensions, through a set of data. This fit defines the relationship that exists between the inputs and the outputs, or it can be used to identify a representation of the data on a smaller scale.

The *main characteristics* of NN and real neurons are learning and adaptation, generalization, massive parallelism, robustness, associative storage of information and spatiotemporal information processing [1].

The first mathematical model of a neuron was introduced by *McCulloch and Pitts* in 1943 [52]. Their model had one layer. It was consisted of the Σ unit which was used to do the

summation of the weighted inputs, it multiplied each input *x* by a weight and then if the result was greater than the threshold then the neuron was "fired", returned 1 else it returned 0 [1]. In the following years new connectionists models were introduced like Associative Memories , MultiLayer Perceptron (MLP) with the BackPropagation learning algorithm, SelfOrganizing Networks and more which will be explained later.

Neural Networks are usually defined by four *parameters* [1]:

1. *Type of neuron*, nodes, like Perceptron Pitts and McCulloch, Fuzzy neuron Yasakawa.
2. *Connectionist architecture,* which is the architecture of the NN, and it is the organization of the connections between the nodes, defining the topology of the NN like fully connected or partially connected. It can also be distinguished depending on the *number* of *input* and *output* neurons and the *layers* that are used. Regarding this there are two different networks. The *Autoassociative* where the input neurons are the output neurons, Hopfield Network and the *Heteroassociative* where they separate input and output neurons like in MLP. Also, there are two architectures based on the connections back from the output neurons to the input which are the *FeedForward* and the *FeedBackward* Networks which are explained in Section 3.2.
3. *Learning algorithm* is an algorithm, that is used to train the network[1]. There are a lot of different kind of algorithms and a lot of research is done in this field. There are three groups of learning algorithms. The first is the *Supervised learning* where the input vectors and the desired output vectors, results, are **known**. The Neural Network keeps training until it learns how to associate each input to its corresponding result, by approximating a function $y = f(x)$. It encodes the example in its *internal structure*. The second category is *Unsupervised learning* where only the input vectors are known. The NN learns by splitting and creating *clusters* among the data based on the characteristics of their input vectors. The last one is *Reinforcement learning* which consists of the idea of **reward/penalty** learning. It also has no information about the desired output vector. Based on the input vector, it calculates the corresponding output and if it is good then it gets a reward, the existing connection weights are increased, otherwise it gets a penalty where the existing connection weights are decreased .
4. *Recall algorithm* which is the learned knowledge that is extracted from the network. It is used in the test, recall phase to calculate the results for new the data in the trained network.

3.2 Basic Types of Neural Networks

The Artificial Neural Networks are classified in two main categories as shown in **Figure 3.1**. The *Basic Types of Neural Networks* which are the types that are most commonly known and the *Specialised Types* which are used to solve problems with more **specific** characteristics. The second one usually has characteristics of the first one, as all of its networks belong or are an extension of a Basic Type of Neural Networks.

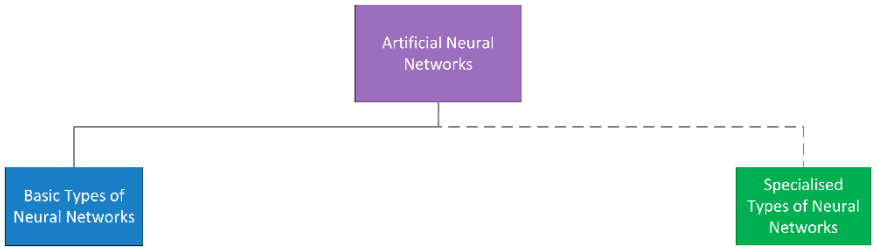

Figure 3.1 Artificial Neural Networks classification

The Basic Types of Neural Networks split into 2 main categories, the FeedForward and the FeedBackward Neural Networks as shown below in **Figure 3.2**.

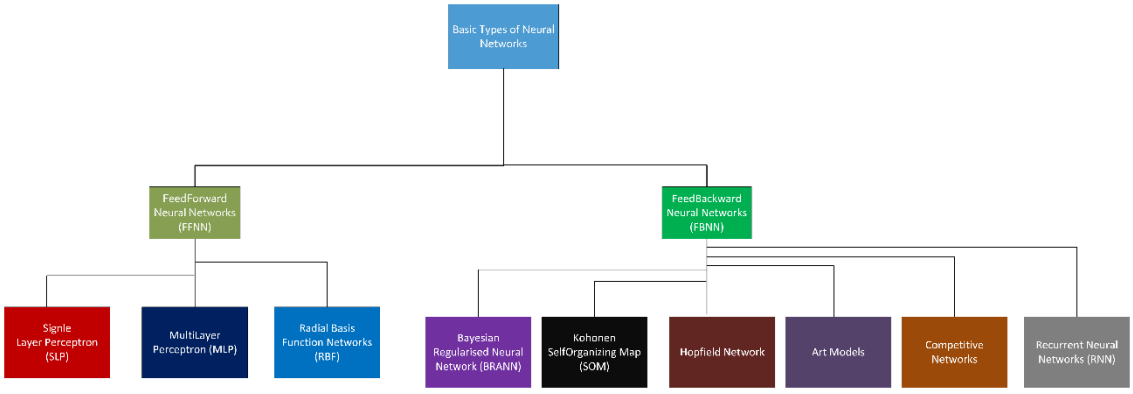

Figure 3.2 Basic Types of Neural Networks classification. It is an extension of a framework shown in [2]

3.2.1 FeedForward Neural Network (FFNN)

A *FeedForward Neural Network* (FFNN) [2] is very similar to the human neuron processing units. In FFNN all the units are related to each other in their layer, by their weight. The information processing in the network involves data entry from the input

units, passes through the network, flows from one layer to another until it gets to the output units, producing the result [2]. There is *no feedback* between the layers, so the information is transmitted only in *one direction*, from the input nodes to the hidden and then to the output. Consequently, it *does not remember* its previous output values and the activation state of its neurons. The main examples of FFNN are the *SingleLayer Perceptron*, *MultiLayer Perceptron*, and *Radial Basis Function Network*.

SingleLayer Perceptron Network (SLP)

In general, they consist of a single layer as shown in **Figure 3.3**. In the *SingleLayer Perceptron* Networks [1], there are inputs which have a *weight*. Then, the Σ is calculated which is equal to the sum of the inputs multiplied by their weights. After that, Σ is checked if it exceeds a chosen **threshold** predicting 1 or 0. SLP can solve only problems that are *linear separable*. Linear separable problems are problems which can be solved, by separating the inputs using a straight line so it is not capable to solve the problem of this study.

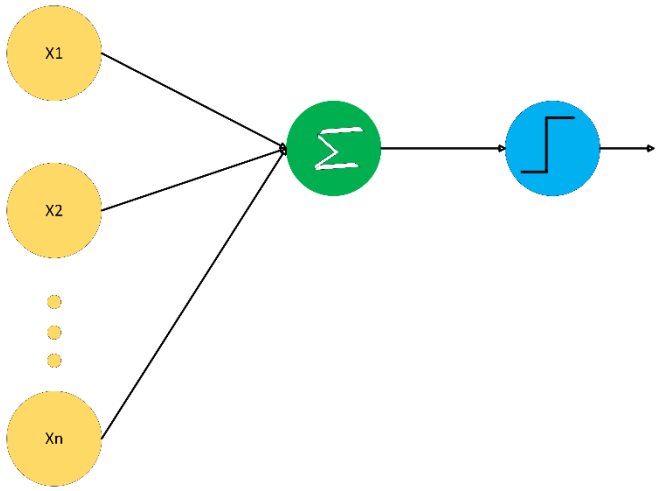

Figure 3.3 SingleLayer Perceptron Network architecture

MultiLayer Perceptron Network (MLP)

In order to *solve* linear separable problems *MultiLayer Perceptron* networks [1] were introduced. They have an *input* layer, one or two *hidden* layers and the *output* layer as shown in **Figure 3.4**. The individual neurons are fully connected or partially when some of their weights are zero. The neurons in the MLP have continuous values inputs and

outputs, summation input function and non-linear activation function [1]. The MLP with 3 active layers can *make* all the *possible classifications* of the input data, so it can *solve* all the classification problems. Also, it is usually trained with the *BackPropagation algorithm* which uses the methods of square error and *gradient descent* to change the weights of the networks until it finds a local or global minimum of the error sum of squares [16].

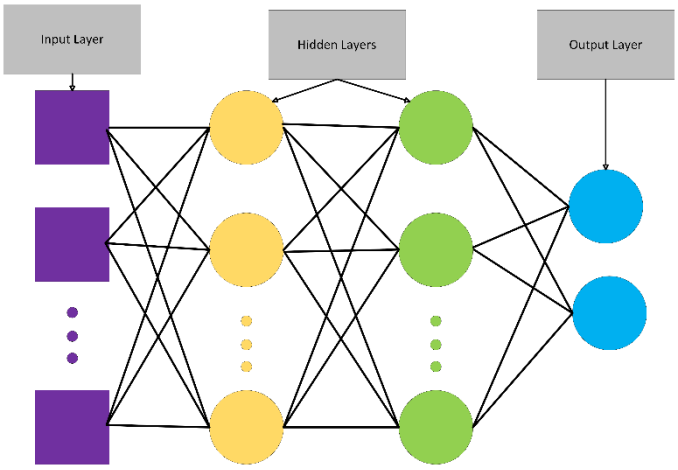

Figure 3.4 Multilayer Perceptron Network Architecture

Radial Basis Function Network (RBF)

As we can see in **Figure 3.5** *Radial Basis Function* Networks [3] consist of an input layer, a hidden layer and a Σ function which produces the result. Its main target is to create different curves for the representation of the data and try to generalize them a little too by making an approximation fit on them. The main idea is that the hidden layer consists of nodes which act like *centers*. Then, the *distances* between those centers and the *input* nodes are calculated and they are used as parameters in a *polynomial function*, like the *Gaussian function* [3]. The last part makes the non-linear separable data into *linear separable*. Finally, a Σ function is used like in the SLP as the data are now linear separable, to make the prediction. Instead of SLP, an MLP can be used to make the predictions as the problem may not become fully linear separable.

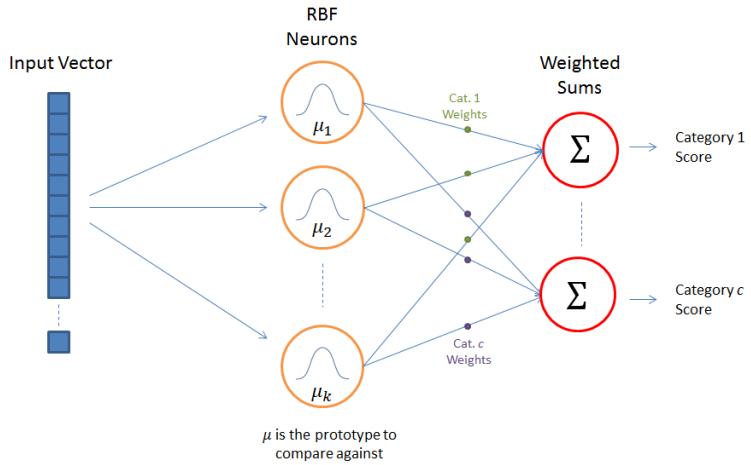

Figure 3.5 Radial Basis Function Network Architecture [50]

3.2.2 FeedBackward Neural Networks (FBNN)

In the *FeedBackward Neural Networks* [2] the information is transmitted in *all* the *directions*, so it can be also transmitted from output to input neuron and from the hidden layer to the input layer.

This kind of network can use *internal state memory* to process sequence of data inputs adding to the network the ability to *remember* previous states and the next state depends on the current input signals and the previous states of the network. The coordinated graph in sequence allows FeedBack NNs to demonstrate dynamic terrestrial behavior for a time sequence.

Some examples of these are the Bayesian Regularized Neural Networks, Kohonen's SelfOrganizing Map, Hopfield Networks, Competitive Networks, Art Models and Recurrent Neural Networks [2].

Kohonen SelfOrganizing Map (SOM):

The purpose of *Kohonen SelfOrganizing Map* [3] is to make input vectors of arbitrary dimension to discrete map comprised of neurons. The map has one or two dimensions as shown in **Figure 3.6**. Each neuron in the map, output neuron, has a vector of weights with the size of the input vector's attributes. During the training, the *location* of the neurons remains constant but the *weight vector* of each one changes. The algorithm's main idea is that it calculates the '*winning neuron'* based on the weight of each neuron and the input and then changes the weights of all the neurons in the neighborhood of it to get closer to

the winning neuron. In the end, many different *clusters* are created, labels are added to the different neurons and when a new data is used, the winning neuron is calculated and returns the label of the cluster it belongs to. This algorithm is unsupervised, so it does not use the desired output in its learning procedure.

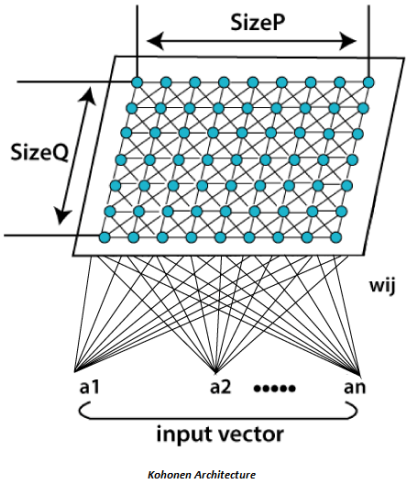

Figure 3.6 Kohonen SelfOrganizing map architecture

Recurrent Neural Networks (RNN) :

Recurrent Neural Networks [3] save the output of a layer and feed it **back** to the input, or a previous layer to help it in predicting the outcome. The first layer is like the FFNN with the sum of the weights and the features. The RNN's difference is that through the *backward edges* each neuron will *remember* some information from the *previous time-step*. So, each neuron acts like a *memory cell*. The neurons use the information they remember with the input to make better predictions.

Main examples of Recurrent Neural Networks :

- Jordan Network [4]:

  In this network ,as displayed in **Figure 3.7**, there is an extra set of neurons in the input layers called *state units*. The activation values of the output units are fed into the state units, with a fixed *weight* equal to *one*. The state units are fully connected to the hidden layer. Because learning is made only between the input to hidden layer and from the hidden layer to the output nodes, it can be trained like the *MLP Network*.

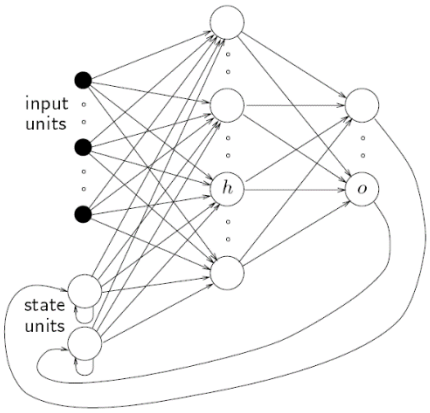

Figure 3.7 The Jordan network [4]

- Elman Network [4]:

  In this network, as displayed in **Figure 3.8**, there is an extra layer called the *context layer* which its activations values are fed back from the hidden layer. Also, there are no self-connections in it. It's like the *Jordan Network*, but the *hidden units* are fed back to the network instead of the *output units*.

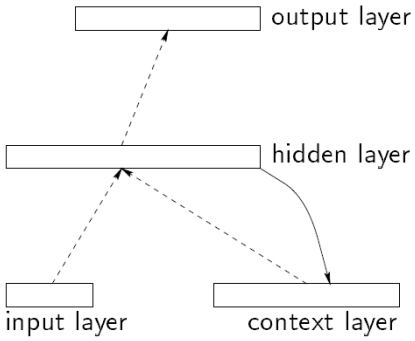

Figure 3.8 Elman Network [4]

- LSTM Network [17][18]:

  LSTM stands for *Long Short-Term Memory*; it is another type of Recurrent Neural Network and is even considered deep learning. In this network every neuron, basic unit is actually a memory block containing one or more memory cells and three adaptive, multiplicative gating units shared by all cells in the block. There are three types of gates, the input, forget and the output as can be seen in **Figure 3.9**. The input defines whether the cell is updated, the forget it resets the memory cell

to zero, to forget the information it contains, and the output is the information of the current cell state that is made visible. The gates can be trained to learn what information to store, how long to store it and when to read it out.

An advantage it has is that it makes small modifications on the data by multiplication or addition that flows through cell states to remember thing selectively, that is better than just remembering previous results as the basic RNN's are doing.

It has a lot of applications like machine translation, language modeling, handwriting recognition, speech synthesis and others. Also, it is usual to be used with other Networks such as CNN in the field of signals.

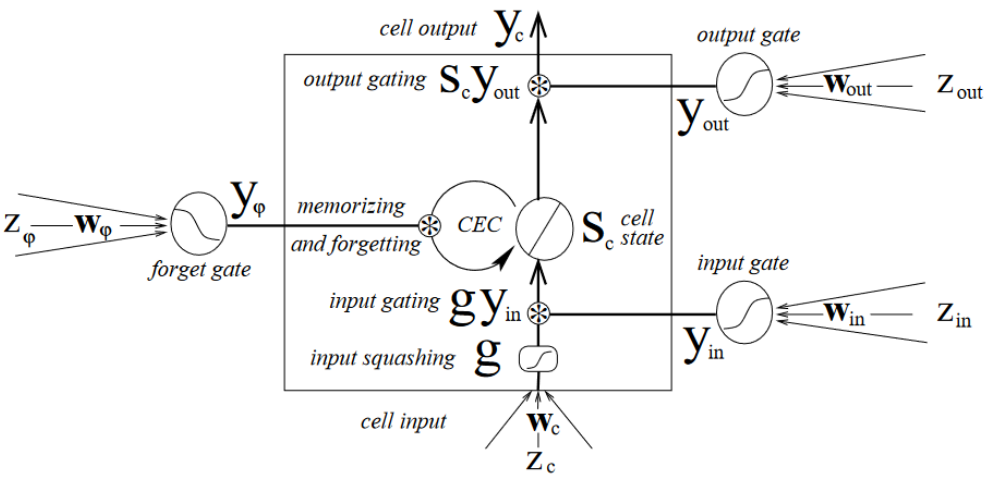

Figure 3.9 LSTM [17]

3.2.3 Comparison of Neural Networks

This section analyzes and compares the different Neural Networks. More specifically, it analyzes the advantages, disadvantages, and applications of all the networks that were mentioned above.

3.2.3.1 FeedForward Neural Network (FFNN)

In FFNN, the information is transmitted only in one direction and each layer connects only with its previous layer. Their applications are classified into two *categories*, control

of *dynamical systems*, and *spaces* where the classic machine learning techniques are applied. Also, they are used in problems where classifying the target classes are *complicated* like *speech recognition* and *computer vision*.

SingleLayer Perceptron Network (SLP)

The main disadvantage of SingleLayer Perceptron network is that it can **only** solve **linear separable** problems which is a really small subset of all the problems that exist, so it is not really used in practice.

MultiLayer Perceptron Network (MLP)

MultiLayer Perceptron Networks **overcome** the linear separability limitations of SLPs. The use of three active layers of perceptron's units can form arbitrary complex shapes that are capable of *separating any classes*, the complexity of the shapes depends on the number of nodes of the networks. Based on *Kolmogorov theorem* they don't need more than three active layers to separate the classes in every possible problem. There are many applications of MLP networks trained with the backpropagation algorithm. For example, in *finance*, *consumer products*, *quality control*, *process control* , *security*, *pattern recognition*, *speech recognition* etc.

Radial Basis Function Network (RBF)

RBF are usually used in *forecasting*. Because of their nature, they are really good at making *predictions* based on a given **time series**. They can also be used in *classification* problems, but they won't succeed as good results as the MLPs.

3.2.3.2 FeedBackward Neural Networks (FBNN)

The FeedBackward Neural Networks can use *internal memory* to store information, because there are backward connections in the network. This internal memory can be used to process sequences of data inputs. It can be applied to tasks like *un-segmentation*, and *pattern recognition*. Also, its application areas include *mathematical proofs*, *seismic data fitting*, *medicine*, *science*, *engineering*, *classification*, *function estimation*, and *time-series prediction* [2].

Kohonen SelfOranizing Map (SOM):

Kohonen SelfOranizing Map is used to recognize patterns in the data. It can be used for *medical analysis* to cluster data into different categories. For example, it was able to succeed high accuracy in classifying *patients* having *glomerular* or *tubular diseases* [3].

Recurrent Neural Networks (RNN):

In Recurrent NN each neuron remembers some information it had from its previous step. So, each neuron acts like a *memory cell* in performing computations. RNNs internal memory can be applied to problems involving *temporal context* as the current output can be calculated by using the current state, which has information about the previous output, and the input of the networks. The Jordan and Elman networks can be used to train a network on *reproducing time sequences*. Also, RNN can be found a lot of helpful for tasks that involve a sequence of inputs like *speech* and *language*, because they have high accuracy at predicting the next character in the text or the next word in a sequence.

3.3 Deep Neural Networks

As explained in Section 3.2 Neural Networks are divided into the Basic Types and the Specialised. The *Specialised Types of Neural Networks* are more *complex* and are used in more *specific problems* of classification. They are divided into the Deep Neural Networks and others as shown in **Figure 3.10**.

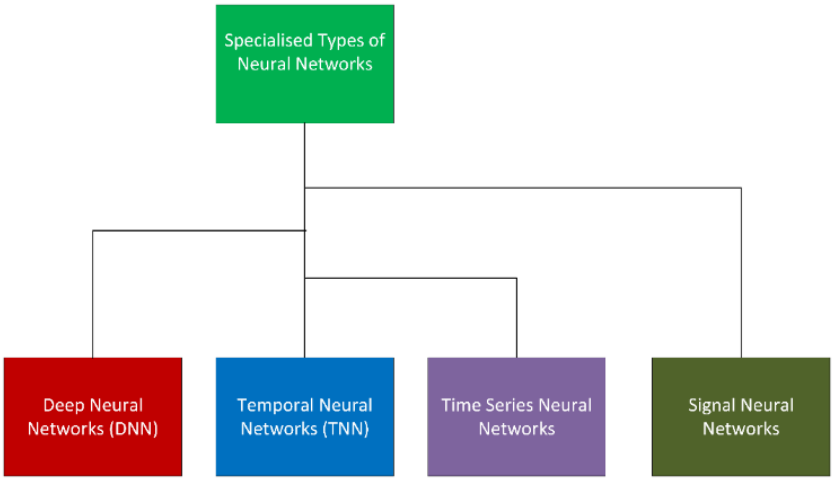

Figure 3.10 Specialised Types of Neural Network

3.3.1 What is Deep Learning

Machine Learning is a subset of AI, meanwhile *Deep Learning* in turn is a subset of machine learning as seen in **Figure 3.11**. *Deep Learning*, DL, refers to Artificial Neural Networks with *complex multilayers* [2]. A *Deep Network* is a FeedForward Neural Network with more than one hidden layer. The term is also used in a graded sense, in which the *depth* denotes the *number* of *layers*. Generally speaking, the *Deep Learning Algorithm* consists of a hierarchical architecture with many layers each of which constitutes a non-linear information processing unit [7]. An important characteristic is that they can *re-use* the *features* computed in a given hidden layer in higher hidden layers.

3.3.2 Deep learning VS Basic Neural Networks

As discussed before, Deep Learning is actually a FeedForward Neural Network. Deep Networks are more *complex* than Basic Neural Networks in a lot of ways. DL has more *complex* ways of *connecting layers*, has more neurons count to express complex models, it needs more computing power to train and has *automatic extraction* of the *features*. It can be defined as a Basic Neural Network with a *broad variables* and layers with a single basic network architecture of unsupervised pre-trained networks[2].

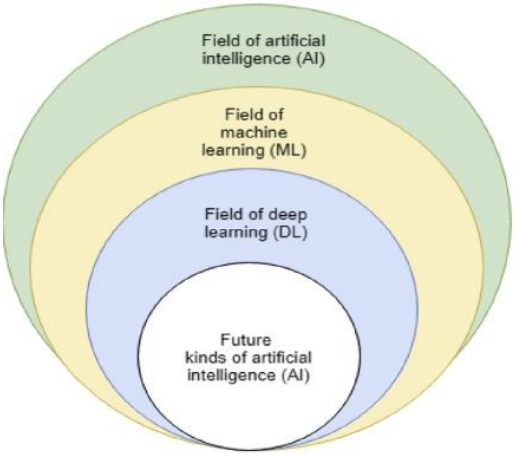

Figure 3.11 Artificial intelligence development and expansion

3.3.3 Deep Neural Networks (DNN)

As shown in **Figure 3.12** there are a lot of types and examples of DNN maintaining some common characteristics.

Restricted Boltzmann Machine (RBM):

Restricted Boltzmann Machines [7] can be interpreted as NNs with *stochastic processing* units connected *bidirectionally*. An RBM is a special type of *Markov random fields* with stochastic visible units in one layer and stochastic observable units in the other layer. There is a full connection between the visible units and the hidden ones, while no connection between units from the same layer. To train an RBM, the *Gibbs Sampler* is adopted. Starting with a random state in one layer and performing *Gibbs sampling*, we can generate data from an RBM. Once the states of the units in one layer are given, all the units in the other layers will be updated. This update process will carry on until the *equilibrium distribution is reached*.

Deep Belief Networks:

The *Deep Belief Networks* [7] are composed of multiple layers of *stochastic* and *latent variables* and can be considered as a special form of the *Bayesian Probabilistic Generative* model. They are more effective in comparison with ANNs, especially in problems with unlabeled data.

Autoencoder:

Autoencoder [7] is an *unsupervised learning algorithm* used to efficiently code the dataset to *reduce* its *dimensions*. First, the input data is converted into an *abstract representation* which is then converted back into the *original* format by the encoder function. More specifically, it is trained to encode the input into some representation so that the input can be reconstructed from that representation. It extracts useful *features continuously* and then *filters* the *useless* information.

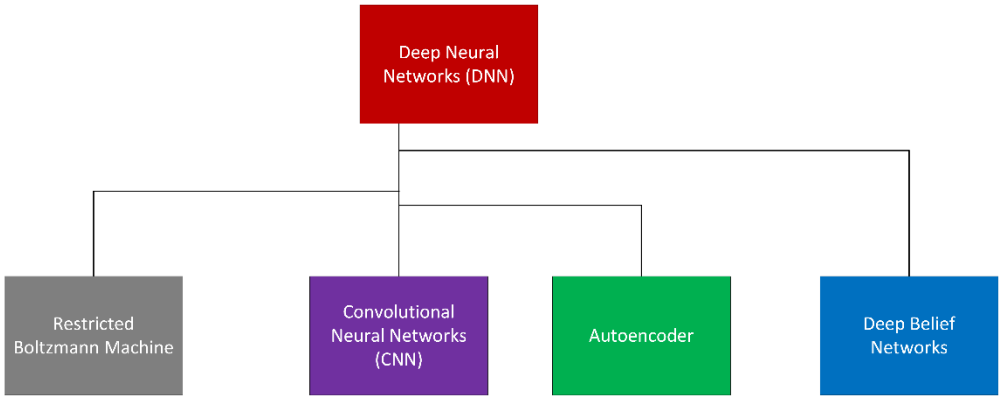

Figure 3.12 Deep Neural Networks

Convolutional Neural Network (CNN):

Convolutional Neural Networks are similar to FFNN, where the neurons have learn-able weights and biases. CNN main difference with the other FFNN is that it understands the **image shifting**, meaning that if it is given to it an image and another image shifted by a little the network will understand that it is the exact same image because it processes it in detail.

A CNN is a NN with some *convolutional layers* and some other types layers as displayed in **Figure 3.13**. A convolutional layer performs a *convolution operation* with a number of *filters*. Then, it uses the *RELU function* to pass only the significant data. After that, the *pooling layer* is used which achieves a non-linear decrease of dimensions, to avoid overfitting phenomena. Convolution, RELU and pooling repeat one after the other for many times. After the process is finished, the data are *flattened* and added to a fully connected FeedForward Network like an MLP to predict the outcome. Its applications have been in *signal*, *image*, *voice*, and *video processing*. Also, it has been used in *speech recognition* and *document reading*. Convolutional Neural Networks are important in *computer vision* too. Its main disadvantage is that it only works on *homogeneous data*, it is *sensible to noise* and there is no explanation of the results. Meaning that, CNN just produces an outcome without justifying the reasoning and the logic behind that, so it is used just as a black-box and for many applications predicting just the answer isn't enough, but we have to know the reasoning that contains. The latest, is currently being explored by a new area of AI the *Explainable AI*. Furthermore, is a common tactic to represent a problem through an image and then use CNN to extract important features.

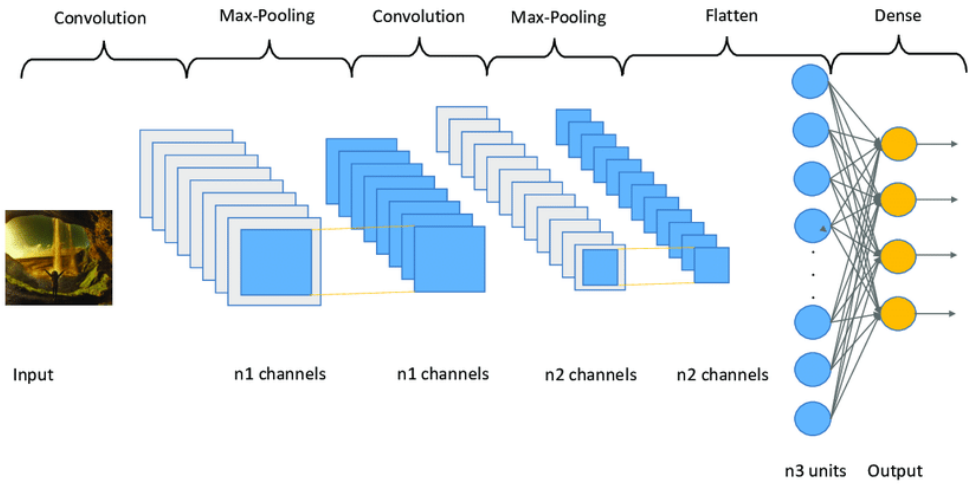

Figure 3.13 CNN Architecture

3.4 Temporal Neural Networks

In the current thesis the concept of ***time*** is really important as the data that will be used to classify people to functional and dysfunctional are *time series* of many different *signals*. There are a lot of different Neural Networks which use the time as a mechanism as displayed in **Figure 3.14**.

Firstly, time can be an internal or external mechanism. *External Mechanism* is when the data are *preprocessed*, like in the rectangular window, and then a standard NN is used, e.g., MLP with BackPropagation.

The time as an *Internal Mechanism* is divided into Time Represented in the Architecture and to Time being Implicitly used. *Implicit Time* means that in the calculation of the result takes into consideration some of the previous results produced by the network. This is mainly about Recurrent Networks like Jordan and Elman that were explained in Section 3.2.2.

Finally, when time is *Explicitly Represented*, in the architecture can be either at the *Network Level*, like in the connections, edges, or at the *Neuron Level*. Some examples of NN in which time is represented at the Neuron Level are Algebraic, Bias(t) and Leaky Integrator Type Models.

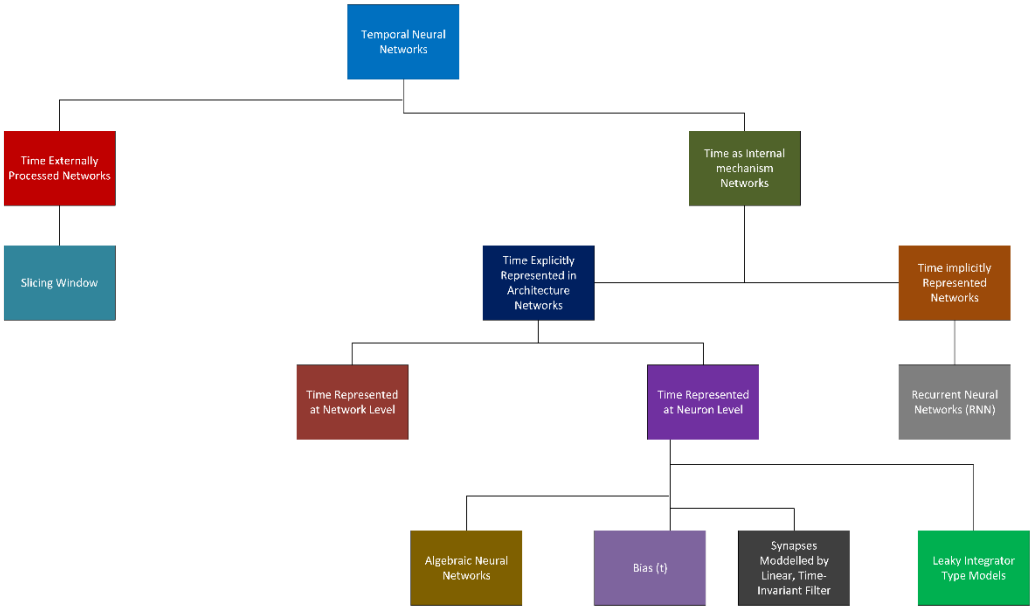

Figure 3.14 Temporal Neural Networks

3.5 Time Series Neural Networks

A *time series* is a series of data through time. An observed time series is usually decomposed into four parts. First, it is the mean value of the data, the long-term trend which can be linear, exponential, logarithmic, etc. Also, another part is its cyclical change, seasonality, and the noise [8]. The most common NN used for time series is the MLP with BackPropagation learning. Its major problem is that it is global approximators, assuming that one relationship fits for all locations in an area.

Nowadays, there a lot of new and more specific kinds of Time Series Neural Networks as shown in **Figure 3.15**.

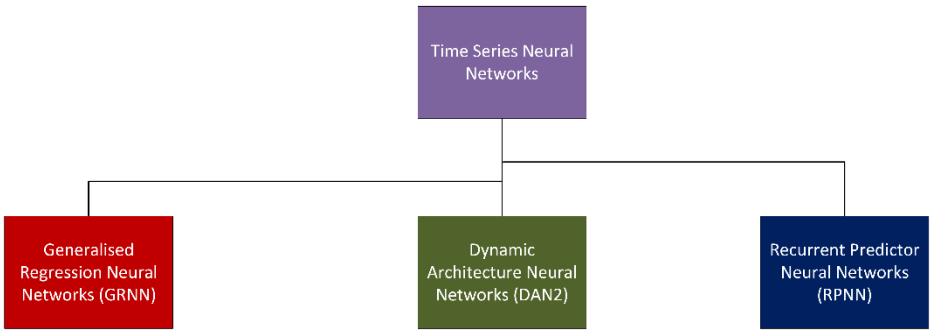

Figure 3.15 Time Series Neural Networks

3.5.1 Generalized Regression Neural Networks (GRNN)

Generalized Regression Neural Networks' [8] major advantage is that they are a *local approximator*. In GRNN each observation in the training set forms its *own cluster*. When a new input pattern is presented to the GRNN for the prediction, each training pattern assigns a *membership* value to it based on the *Euclidean distance*. It forecasts the expected future value *x* and the expected future volatility *y* of the time series. The trained GRNN ensemble first one *x* and the trained GRNN ensemble the second one *y*. They are used to make successive one step ahead forecasts by *rolling* the sample forward one time step, using the forecast as an input, and making another one-step-ahead forecast and so on. Multiple predictors perform significantly worse compared to other algorithms. Also, lagged input variables are highly correlated, so they make each other *redundant*.

3.5.2 Dynamic Architecture for ANN (DAN2)

DAN2 [9] is a type of FeedForward Neural Network as shown in **Figure 3.16**. The major logic behind this network is the *repetition* of **learning** and **accumulating** knowledge at each layer, *propagating* and *adjusting* the knowledge forward to the next layer until the desired performance is met. Each hidden layer has 4 nodes. The first node is the *input*. The second node is a *function* that encapsulates the current accumulated knowledge element during the previous training step, CAKE. The other two nodes, *Curnole nodes*, represent the current remaining nonlinear component of the process via a transfer function of a weighted and normalized *sum* of the input variables. The final cake node is the output or the dependent variable.

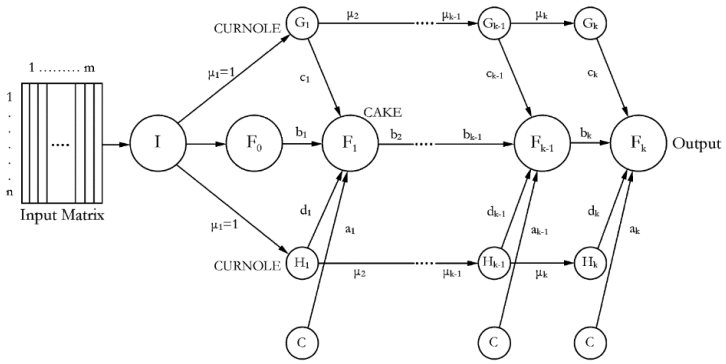

Figure 3.16 The DAN2 Network Architecture [9]

DAN2 vs FFBP

The main differences between *DAN2* and *FFBP* models is that in FFBP the model systemically *updates* the parameters in *every iteration* until it is fully trained. On the other hand, DAN2 uses the *entire data* set in every iteration to estimate the weights and the network's five parameters. Also, DAN2 uses the trigonometric *cosine* function instead of the *sigmoid* to catch the nonlinearity of the process. Furthermore, the number of hidden nodes is fixed at four, while in FFBP they decide the number of nodes through experimentation. Their final difference is that the architecture of DAN2 is less complex as it allows many to one relationship only, while the FFBP models uses many to many relationships.

3.5.3 Recurrent Predictor Neural Networks (RPNN)

RPNN [10] is used in the *long-term prediction* of chaotic time series. It consists of nonlinearly operated nodes whose outputs are only connected with the inputs of themselves, and the latter nodes as shown in **Figure 3.17**. Before each prediction, a teaching signal, a set of the latest signal records, is presented to the network and uses a gradient-based learning algorithm to train the network.

RPNN is a special type of RNN. It consists of two types of elements, *nodes*, and *branches*. Its target is to provide a direct link between the activity of a neuron and output of the network at a later moment. The nodes correspond to instant points of variables, the output node is fed back into the inputs itself, and those nodes represent the instant points at latter moments. Also, RPNN has many branches with time delays between corresponding nodes, that represent the relations between those nodes. The node activations functions, internal states, are fed back into themselves and their latter nodes at every time step to provide additional input. This provides network *dynamic characteristics* and *internal memory* which is important for time series. The RPNN algorithm tries to adjust the weights of each branch between nodes to minimize a criterion function of the network.

To sum up, RPNN structure is designed according to *reconstructing phase space*, and a *gradient-based, self-adaptive* algorithm is used. Prediction is made by examining trajectories on the reconstructed phase space [10].

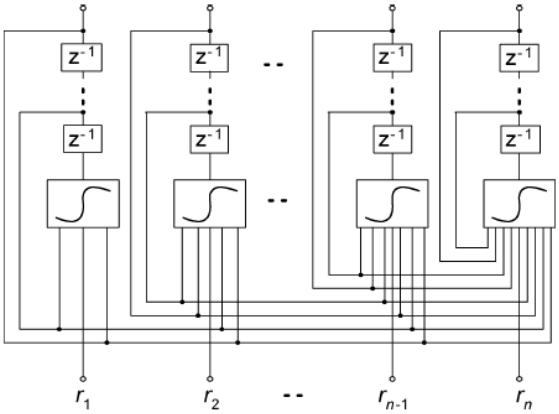

Figure 3.17 Architecture of the RPNN [10]

3.6 Signals Neural Networks

In the last years it has become more and more common to use NN to make predictions based on **signals**. It is very popular to use trained *Concurrent Neural Networks* with signals, like EEG, for emotion recognition [11][12][13]. It is a usual tactic to make some *preprocessing* in the signal to recreate, represent it as a picture which then, can be given as an input in a CNN and use other classification algorithms to make the predictions. This section describes some different approaches for this subject.

3.6.1 "Deep fusion of multi-channel neurophysiological signal for emotion recognition and monitoring" [13]

This solution is interesting because it takes into consideration the *concept of time*, which is important as the data is a time series. It explores the *emotion recognition* based on neurophysiological signals like *EEG*, *fMRI*. The architecture of the network is displayed in **Figure 3.18**. The preprocessing encapsulates the multi-channel signals into *grid-like frame* cubes within a specific time window. Then, CNN is used to mine inter-channel and inter-frequency correlation and *extract features* from the frame cubes. After that, Long Short-Term Memory, *LSTM*, that is a refined RNN structure, makes the predictions. This models the contextual information for sequences that have arbitrary length. CNN provides the process of data with 2- or 3-dimensional *structure* and the RNN resolves the delayed effect though accumulating the weak signals characteristics in each time step, its

good at sequential modelling. A specific structure of LSTM is used which has 3 kinds of gates the *forget*, *input* and *output* gate as shown in Figure **3.19**.

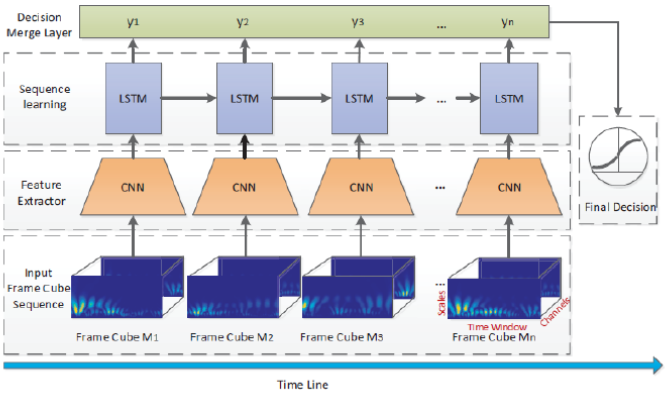

Figure 3.18 Architecture of Network [13]

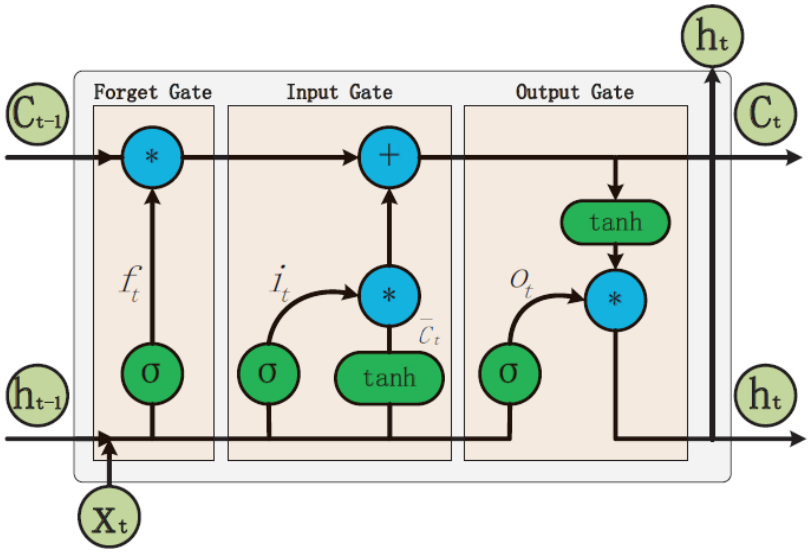

Figure 3.19 LSTM Unit **[13]**

3.6.2 "Emotion recognition from EEG-based relative power spectral topography using convolutional neural network" [11]

Firstly, *preprocessing* takes place using the *Fourier Transformation* on the signal and *RPSD*. The multichannel emotional *EEG* signals are mapped into two-dimensional tomographic images using *RPSD*. These images combined frequency and spatial domain information of the EEG signals. The exact preprocessing process is shown in **Figure 3.21**.

Then, uses CNN with 2 layers, result the use of the convolutional layer, batch normalization layer, *RELU* layer and *pooling* layer twice as displayed in **Figure 3.20**.

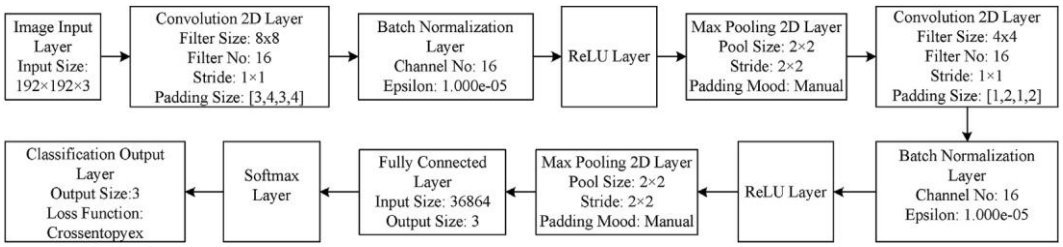

Figure 3.20 The values of the different parameters of the layers in the proposed CNN along with the clarification of the regarding layers [11]

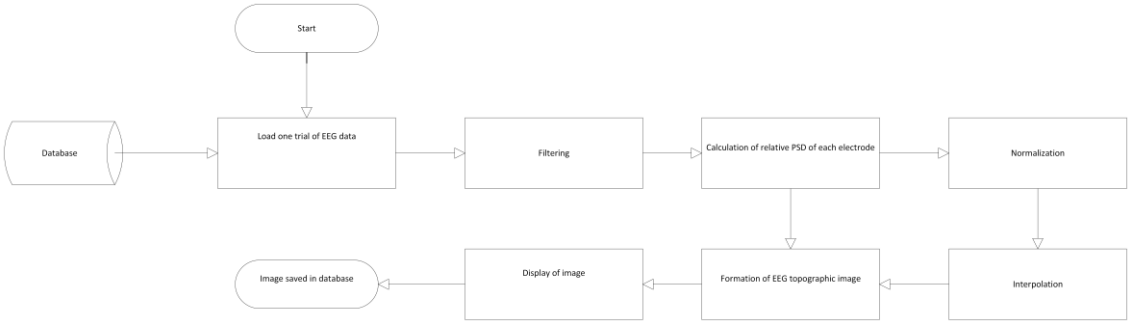

Figure 3.21 Flow diagram of the procedure to construct the topographic image from multichannel EEG [11]

3.6.3 "Emotion recognition with convolutional neural network and EEG-based EFDMs" [12]

This system, network is used to solve the same problem as Section 3.6.2 and its solution is very similar to that. *Short-time Fourier Transform*, *STFT*, is used for *preprocessing* because it is mainly used to analyze the frequency features of time series. It provides the frequency information *averaged* over the entire signal time interval and does not know the time when each frequency component appears. Also, STFT is ideal for *non-stationary* signals like *EEG* that is used in the proposed paper. Then, it uses a novel concept of *EFDMs* based on multiple channel EEG signals. These can be treated as grayscale images to apply 2-dimensional convolution operations.

After that, CNN is used as shown in **Figure 3.22**, with one convolutional layer, four *residual blocks*, four *max pooling* layers, two fully connected layers, *SoftMax* layers.

Furthermore, it has 5 batch *normalization* and four *dropout layers* for over-fitting consideration. After the CNN, two fully connected layers are used for emotion *classification*. *Gradient-weight* class activation mappings, *Grad-CAM*, are used to make CNN based models more transparent by producing visual explanations. It is important to add that this solution achieves positive emotion recognition with high accuracy, while negative was confused with the neutral emotion.

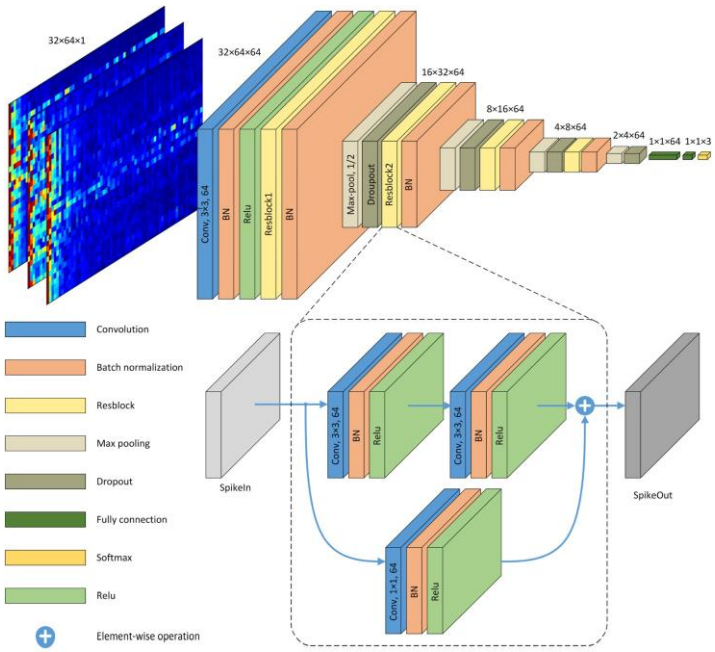

Figure 3.22 The proposed residual block-based CNN for EEG emotion recognition [12]

3.6.4 Specific Signals: ECG, HeartRate Variability (HRV) and Electrodermal Activity (EDA)

ECG, HRV and EDA signals are usually used to predict *Stress-Related Mental Disorders*. It is *common* to use neural networks to predict pain based on some of these features. CNN, SVM, multi-task-learning NN, MLP, a *combination* of CNN with LSTM like it is analyzed in Section 3.6.1 and a lot of other NN, can be used to predict pain based on these signals.

For example, one way to predict pain is proposed by *"Using Deep Convolutional Neural Network for Emotion Detection on a Physiological Signals Dataset (AMIGOS)"* [14].

Firstly, the data is preprocessed and DCNN (Deep CNN) are used, the preprocessing was different, simpler than in Section 3.6.1. Then, the result is sent to the input neurons of the three FCN, to perform the training and testing process of the model.

Another paper that discusses a lot of different models to predict pain intensity (0-4) based on those signals is *"Exploration of physiological sensors, features, and machine learning models for pain intensity estimation"* [15].

This paper uses standard *machine learning*, without Neural Networks but it is presented in this section because of its *methodology*. This study uses different techniques to develop the models and based on this study an equivalent technique is used. The technique that is used in this study is the *subject-depended as explained below*.

To get the data, they used a dataset that was collected through an experiment. The experiment includes 87 healthy people, was made in a lab and is about physical pain. They considered the baseline temperature of 32˚C and they conducted heat pain in the subjects' right hand, and they found the temperature in which they started feeling pain until the temperature at which they could not tolerate it anymore. They also added two intermediate temperatures such that the four temperatures are consecutively separated by an equal distance. So, for each subject they had four different temperatures which were correlated with four pain levels, 1 - 4. In the actual experiment a subject was experiencing a mix of these four temperatures in random order, for 25 minutes. The subject experienced each temperature level for 5.5 seconds followed by a no-pain recovery phase at the baseline temperature for eight to twelve seconds. Through this they collected the EDA, ECG and EMG signals.

The Machine learning algorithms they use are *Support Vector Regression* (SVR) with RBF kernel function, *Extreme Gradient Boosting* R*egression, Random Forest, K-Nearest Neighbor* (KNN) and *Linear Regression*. They use 15% of the samples outside the train and test sets for *hyperparameter tuning* and they extract *features* from the three signals *ECG, EDA, EMG* and train the models with them and with a subset of them separately. They develop models based on the *subject-independent*, the *subject-dependent* and a *hybrid* model pain estimation scenario. The *subject-independent* model is a *generic* model for all the subjects, that does not take into consideration the exact measures for each subject. They build it from subjects in the training set and tested it with the testing set. They use leave-one-person-out-cross-validation, and they estimate the pain intensity of a new subject based on the *patterns discovered* on the *labeled* data samples of other

subjects. The best *EDA features* for this method are the time interval between successive extreme events above the mean, time interval between successive extreme events below the mean and the exponential fit to successive distances in 2-dimensional embedding space. The *subject-depended* model is dedicated to each *person individually*, is trained and tested on different data samples of the same person and they use 10-fold stratified cross-validation for each person. Finally, they use a *hybrid model* combining subject-dependent and subject-independent approaches to benefit from the advantages of both approaches. They use *KNN* to categorize the population data and then they use the *cluster-specific model* to assess the pain intensity of the patient. It is less challenging than subject-dependent model as they *don't need* to do a *signal recording* for a long time, and they don't need to *build* a new model for *each patient*.

Overall, their results indicate that for both scenarios, subject-independent and subject-dependent, **EDA** signal was the best to identify the pain level and the **SVR** algorithm gave the best error. Also, The ECG and EMG signals give better results for the **subject-dependent** model than the subject-independent. Furthermore, from the hybrid model they find similar results as the other 2, but it also helps the ECG, EMG signals to predict better results.

Finally, they *reference other studies* that use these kinds of signals to make similar predictions and compare them with their work. Some of these studies use Neural Networks to predict pain like *LSTM RNN*, *RNN-ANN*, and *CNN-LSTM*.

Chapter 4

Neural Networks and Methodology

In this chapter the different NNs that were used are analyzed and the methodology that was followed in the current study is presented.

4.1 Chosen Neural Networks

From the research that was conducted in Chapter 3 five Neural Networks models were chosen to Classify Functional versus Dysfunctional Coping with Acute Pain. NN's from different types to check which one had the best results. The NN that were chosen are **MultiLayer Perceptron (MLP)**, **Radial Basis Function (RBF)**, **Kohonen SelfOrganizing Map (SOM)**, **LSTM,** and a hybrid model which consists of a combination of **RBF, MLP** and **LSTM**. Most of the networks belong to the *Basic Types of NN* as we presented in Chapter 3. No network was selected from *the Specialised Types of Neural Networks* because those networks need a lot of data to be trained and the dataset in this study is very small. An exception is the **LSTM** which belongs in both *FeedBackward Neural Network* and in the *Temporal Neural Network* that is a type of *Specialised Neural Network*.

4.1.1 MultiLayer Perceptron

MultiLayer Perceptron is a FeedForward Network and with the use of 3 active layers can form arbitrary complex shapes that are capable of separating any classes. Also, it is the

most common NN and usually produces good results with a small amount of data. It can probably learn the data well as the number of their features is low.

4.1.2 Radial Basis Function

Radial Basis Function is also an FFNN and uses the MLP. We chose this as with the use of the extra RBF layer, it can normalize the input data as it can create different **curves** for the representation of the data and **generalize** them by making an approximation fit on them. In this way, the normalized data can be **more easily separable** by the MLP that follows. Also, it has the same advantages as MLP like the need of a small amount of data to train.

4.1.3 Kohonen SelfOrganizing Map

Kohonen SelfOrganizing Map was selected as an **unsupervised** NN, so it does not use the desired output to train, and it creates cluster of the data which can help in this study, as there are two clusters, teams, avoidance, and acceptance. Furthermore, it belongs to the FeedBackward NN which is different from the previous NN selected and is really simple so it may not need so much data like other NN clustering algorithms.

4.1.4 LSTM

LSTM was chosen because is a Recurrent Neural Network which belongs to the FBNN and the RNN's can be helpful for this study as they **remember** previous outputs. This network was chosen because it is more complex and has more logic on what data to remember and forget and it is also used in other studies that work with signals in combination with other networks as analyzed in Chapter 3.

4.1.5 RBF – MLP – LSTM

The final NN that was selected is a hybrid model consist of an RBF layer to **normalize** the data as explained in Section 4.1.3, the MLP which has the ability to **separate** any classes and the LSTM which can provide a feature to the network to **remember** previous

outcomes. Each network is a layer of neurons followed by another network. So, there is an RBF layer followed by one or two layers of MLP and followed by a layer of LSTM as shown in **Figure 4.1**.

This network may combine the advantages of each different NN to make better predictions and succeed better generalization.

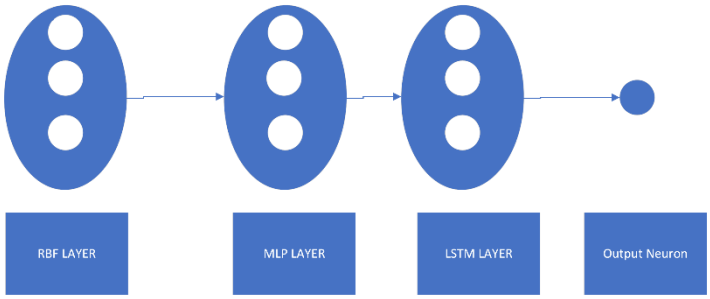

Figure 4.1 RBF – MLP -LSTM model

4.2 Model Evaluation

To determine the most-efficient Neural Network in the topic of this thesis, it is important to select the most suitable evaluation methodology and performance metrics to compare the algorithms. Also, it is important to follow the same methodology as previous studies in which we compare our results with, i.e., [22].

4.2.1 Evaluation Methodology

The evaluation methodology that was used is *Stratified k-fold cross-validation* [48] which was used in all related works in the past [19][20][21][22]. The data are split into *k* different groups and in each of the *k* iterations one group is used as the test set and the others *k-1* as the training set. The model after every iteration is discarded and at the end the average evaluation score is returned. With this method, the result is more generic and trusted as it does not depend on the randomness of which data belongs to the test and training set.

4.2.2 Performance Metrics

Four important measures are needed to compute the performance metrics, the *True Positives (TP)*, *False Positives (FP)*, *True Negatives (TN)*, and *False Negatives (FN)*. The *True Positives* are the number of samples **correctly** classified as **positive** and the *False Positives* are the number of samples **incorrectly** classified as **positive**. The *True Negatives* are the number of samples **correctly** classified as **negatives** and the *False Negatives* are the number of samples **incorrectly** classified as **negative**.
In this study, by **positive** it is meant that the participant is classified as **dysfunctional** while **negative** as **functional**. Also, it is more important to find the dysfunctional than the functional people as they need medical help. For this reason, *Sensitivity*, *Recall*, and *F1-score* are selected as the most important metrics. Also, *Accuracy* is selected to check the **overall** results.

Sensitivity (Recall) is the True Positives in total positives in the data ratio.

$$\textbf{Sensitivity} = \textbf{(TP) / (TP + FN)}$$

Precision is the True Positives to total predicted positives ratio.

$$\textbf{Precision} = \textbf{(TP) / (TP + FP)}$$

F1-score is the harmonic mean of precision and recall.

$$\textbf{F1-score= 2 * ((Recall * Precision) / (Recall + Precision))}$$

Accuracy is the correct predictions to total predictions ratio.

$$\textbf{Accuracy} = \textbf{((TP + TN) / (TP + FP + TN + FN))}$$

4.2.3 Subject Dependent Methodology

There are a lot of ways to use the machine learning models as another study suggests [15]. There is the **subject-independent** , the **subject-dependent** and even a **hybrid** model of these two as explained in Section 3.6.4. For this study, the ***subject-dependent*** method is chosen which uses the signals metrics from the subjects to train the data and handles each subject as a lot of different rows of data as his signals is collected throughout time. Then,

a data multiplication methodology is applied as discussed in the section below. This method is used as it is the method of the previous study [22] which will be **compared** with. Also, it is the most suitable as the number of subjects is low, so it would be very difficult to train Neural Networks, which in general need more data than the standard machine learning models, with metrices like average and standard deviation of all the subjects throughout time. Furthermore, signals like ECG and EMG gave better results than subject-independent method in a related study [15].

4.3 Methodology

The methodology that was followed in this study is split into **five** phases, parts as explained below.

4.3.1 Phase 1

The first phase is shown in **Figure 4.2**. In the first phase we conducted a study about Neural Networks and the most suitable NN were selected. Then, the feature selection is followed which is the same as in [22] to have a more **precise comparison**. This phase is already analyzed in Chapters 3 and 4.

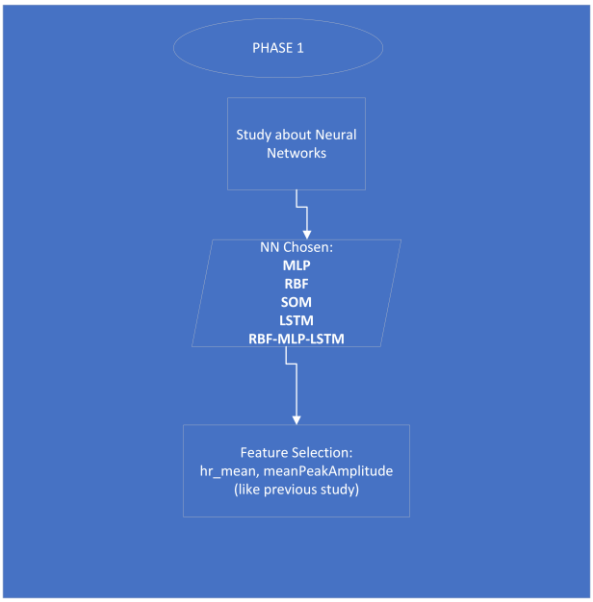

Figure 4.2 Phase 1

4.3.2 Phase 2

In the second phase the **Rectangular Window Methodology (RWM)** (see Chapter 5.1), is used to multiply the data just like in [22]. The five Neural Networks are used and in the LSTM model some more variation models are used to find the most suitable Recurrent Network for this study as shown in **Figure 4.3**. This phase is analyzed in Chapter 5.

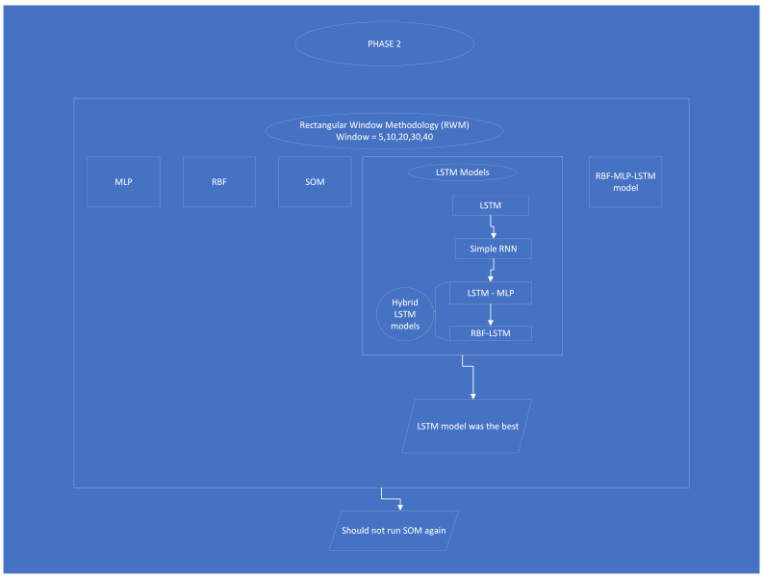

Figure 4.3 Phase 2

4.3.3 Phase 3

In the third phase another method is used to multiply the data. This is the **Moving Window Methodology (MWM)** (see Chapter 6.1) and then the different NNs are executed. As shown in **Figure 4.4**, two versions of MWM are used, one with big and one with small slides. This phase is presented in Chapter 6.

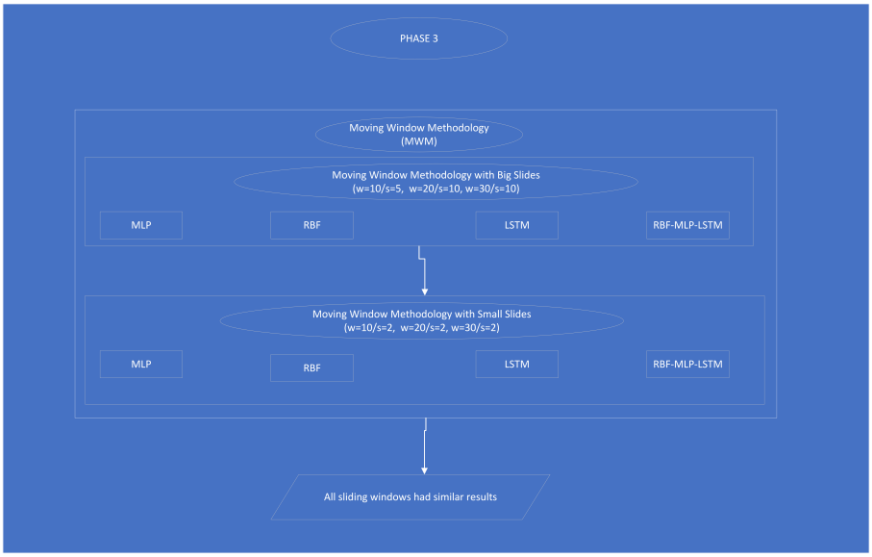

Figure 4.4 Phase 3

4.3.4 Phase 4

In Phase 4 there is a change in the *activation function* and both Moving Window Methodology and Rectangular Window Methodology are used as shown in **Figure 4.5**. This is further explained in Chapter 7.

Figure 4.5 Phase 4

4.3.5 Phase 5

Phase 5 is the final phase in which an expansion takes place to a dataset created by **another** experiment with real time wearable data as shown in **Figure 4.6** and it is discussed later on in Chapter 8.

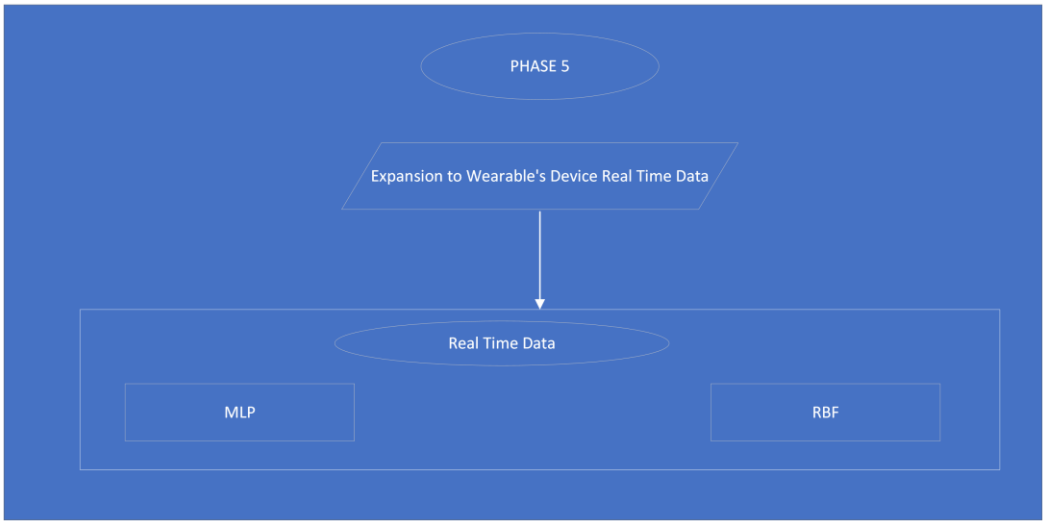

Figure 4.6 Phase 5

Chapter 5

Analysis Using the Rectangular Window Methodology

In this chapter *Phase 2* is analyzed as displayed in **Figure 4.3**. The Rectangular Window is used to multiply the data like the previous study [22]. A lot of different NNs are used like MLP, RBF, SOM, RBF-MLP-LSTM. Also, a lot of different variations of LSTM and Recurrent networks are used to find the best for the current problem.

5.1 Rectangular Window Methodology

The Rectangular Window Methodology (RWM) [24] creates multiple artificial training samples by taking advantage of the nature of the raw data provided by the experiment explained in Chapter 2. The raw signals, ECG, fEMG, EDA, that were recorded for each person during the experiment are in a time series form. RWM goes through each individual person's monitoring data and attempts to multiply them as seen in **Figure 5.1**. More precisely, it splits the time series of the data into **non overlapping windows** with the **same** size. In this way, many different samples are created where each sample starts where the previous one ended, succeeding a multiplication of the data without any overlapping which produced overfitting phenomena as discussed in [21]. During each window a lot of statistical features are extracted like the selected features of Section 2.6.1 and those explained in previous studies [21][22].

This methodology is built on the assumption that a person's emotional state fluctuates through time. Therefore, a window of psychophysiological signals can be treated as another individual person and record in the machine learning model's training process. Because of the lack of labeling throughout the experiment, each individual sample was given the original label of the person being recorded[21].

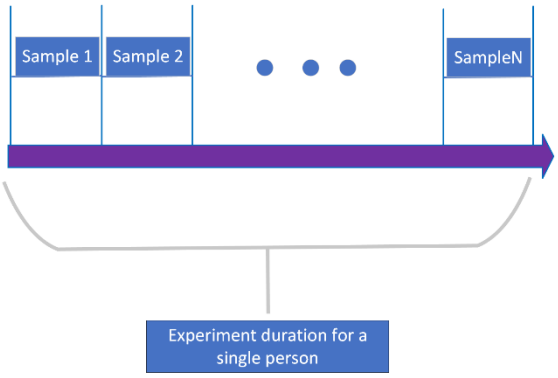

Figure 5.1 Rectangular Window Methodology

Five windows sizes are used for the comparison of the NN. The window sizes are 10,20,30,40 like in study [22] and 5 seconds. The RWM is applied only on Wearable Devices as it produced better results in the previous study [22] and it is closer to the real time data of chapter 8. This is a great way to multiply the data collected by the experiment explained in Chapter 2 and increase the number of rows as it can be seen below in **Table 5.1**.

Total Number of Samples	
Window Size (sec)	Wearable Devices
5	2976
10	1210
20	590
30	386
40	257

Table 5.1 Total Number of Samples in RWM

5.2 MLP

The MLP is run for all the window sizes, 5,10,20,30,40 with the same feature selection as in [22] and then it is run with all the features. As it can be seen in **Figure 5.2** the number of neurons in the input layer is fixed at 2 neurons for the experiments of Section 5.2.1 and at 33 for the experiments of Chapter 5.2.2. The number of neurons in the hidden layers one and two is adjustable and a lot of different experiments are made in the current chapter for them. Finally, the output layer consists only one neuron which predicts if someone is functional or dysfunctional with the sigmoid function.

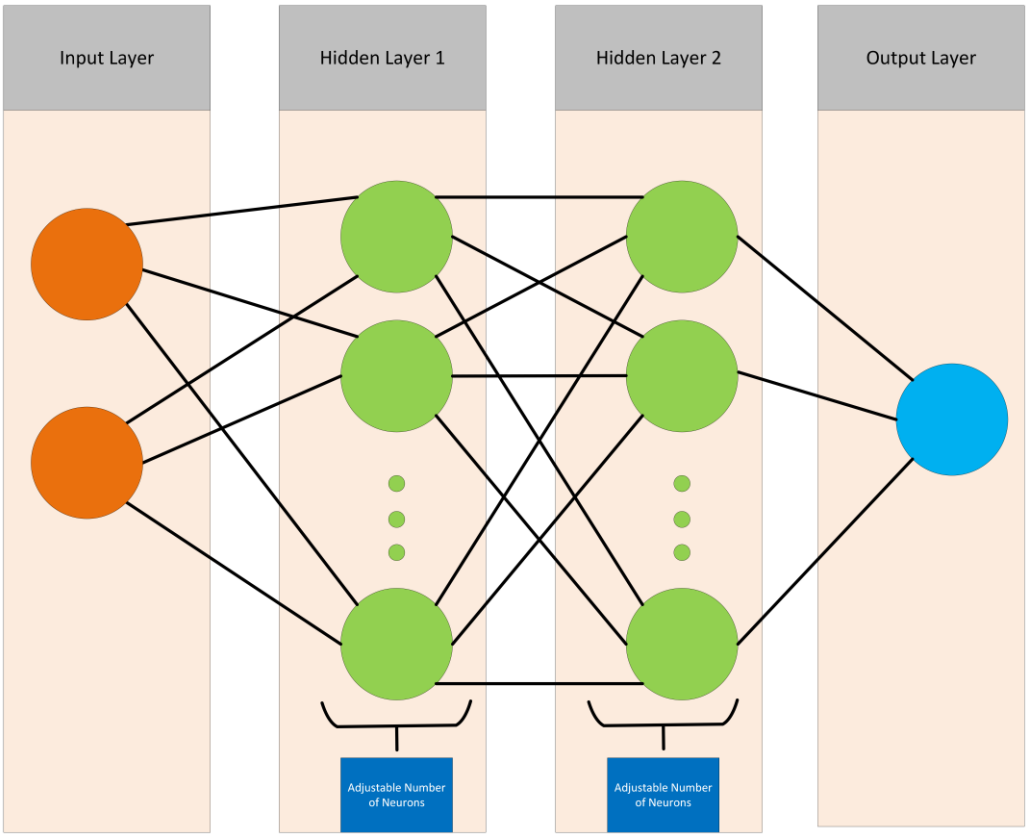

Figure 5.2 MLP Experiments Network

5.2.1 MLP with hr_mean and meanPeakAmplitude

The explanation of the two features, hr_mean and meanPeakAmplitude, is provided in Section 2.6.1. The analysis of the MLP network with window size of 10 is described in **Table 5.2**. The comments' column **compares** the results.

The **best** result from this window is when the epochs is 500, the hidden layer 1 and 2 neurons are 100 as shown in **Figure 5.3** and when the neurons are 400 and 100 in hidden layers two and three respectively as seen in **Figure 5.4**.

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10	500	20	10	The training accuracy is around 65%.
2		500	50	45	Similar to 1.
3		500	100	100	Accuracy around 70%.
4		500	200	200	Little better than 3.
5		700	100	100	Same as 3.
6		500	400	400	Similar to 4, so maybe cannot succeed better results with more neurons.
7		3000	50	50	A little worse than 6.
8		500	800	800	No difference than 6 and does not need 500 epochs to converge.
9		500	400	100	Runs with different number of neurons in the 2 layers. It learns faster, and it is similar to the best third run.
10		500	600	100	Same result as 9.
11		500	400	20	Similar to 9,10

Table 5.2 MLP Executions of window 10 with 2 features

Figure 5.3 MLP with Window of 10 and 100,100 Neurons

Figure 5.4 MLP with Window of 10 and 400,100 Neurons

The analysis of the MLP network with the rest window sizes, 20,30,40 and 5, is described in **Table 5.2**. The color green indicates the best results in the whole table and the color blue the best within that window size.

The window size of **20**, produces the best results but the **difference** with the other windows is **very little**. The results of this window are displayed in **Figure 5.5**.

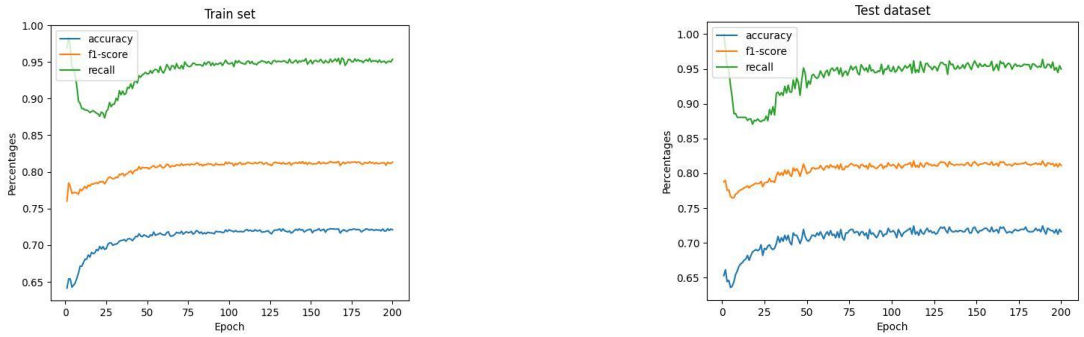

Figure 5.5 MLP with Window of 20

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	20	500	10	10	Bad results.
2		500	50	50	Better results than 1.
3		500	100	100	Better results but not converged.
4		1000	100	100	Better results and it is converged.
5		500	200	200	Better than 3, similar to 4.
6		1000	200	200	Better than 5.
7		1000	400	400	Similar to 6.
8		600	800	800	Similar to 7.
9	30	500	100	100	Moderate results, it needs more neurons and epochs.
10		500	200	200	Better than 9.
11		1000	200	200	Better than previous within this window.
12		800	400	400	Similar to 11.
13		800	800	800	Similar to 11.
14	40	800	100	100	Moderate results.
15		800	200	200	Better than 14.
16		1000	200	200	Better than previous runs.
17		1200	200	200	The best results
18	5	500	100	100	Moderate results.
19		300	400	200	Same as 18.
20		300	600	600	Better accuracy than the previous
21		300	400	300	The learning rate changed from 0.01 to 0.3 and the results were a lot worse than before.

Table 5.2 MLP Executions of Windows 20,30,40,5

Overall**,** in every window size, the MLP can achieve **similar metrics**, accuracy, recall, F1-score, and if a network has less neurons, then it needs more epochs to converge and learn. Also, with a **bigger** window size, the dataset is **smaller**, so **more neurons and epochs** are needed for the network to converge. All the windows sizes produce **similar** results but with **bigger** window sizes the results are **little worse** and there is **higher standard deviation** between them. The best metrics were produced with window size of **10** as shown in **Figure 5.3**.

5.2.2 MLP with all Features

In this section the MLP model is trained with **all the features** instead of only two like in the previous section to check if the feature selection that was made in [22] stands, works for NNs too. Also, it is made only for window size of 10 seconds, as it produced the best results in the previous section. The results are displayed in **Table 5.3**. As shown in **Figure 5.6** the MLP network with all the features is being overfitted as the training metrics are going to 100% and the testing data's metrics are getting worse over time. In conclusion, it seems that the features selection made in [22] is **appropriate** for Neural Networks too, as it can produce good results and it **stops** the **overfitting**. It is appropriate for NNs as it takes into consideration three different methods of which one is independent of the machine learning model that is used as discussed in Section 2.4.1.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	500	100	100	It is not converged.
2	1000	100	100	Overfitting phenomenon.
3	1400	800	800	Similar to 2.
4	800	800	300	Similar to the rest.
5	1000	30	20	Overfitting happens even with less network complexity.
6	800	30	0	Really bad results.
7	800	400	0	The network is still being overfitted.

Table 5.3 MLP with All the Features

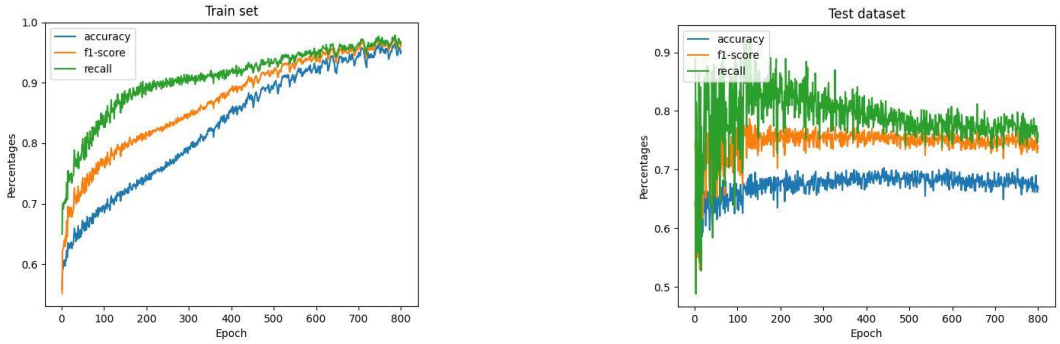

Figure 5.6 MLP with All the Features

5.3 RBF

The RBF network is run for all the window sizes, 5,10,20,30,40 with the same feature selection as in [22] and then it is executed with all the features. The RBF network for the experiments of this study is displayed in **Figure 5.7**. It is just like the MLP network which is explained in Section 5.2 but between the input and the first hidden layer there is the RBF layer. The RBF layer has the same number of neurons as the input layer so in this study it has two neurons for the initial experiments and 33 for the later. The RBF neurons are trainable centers which are used to normalize the input data.

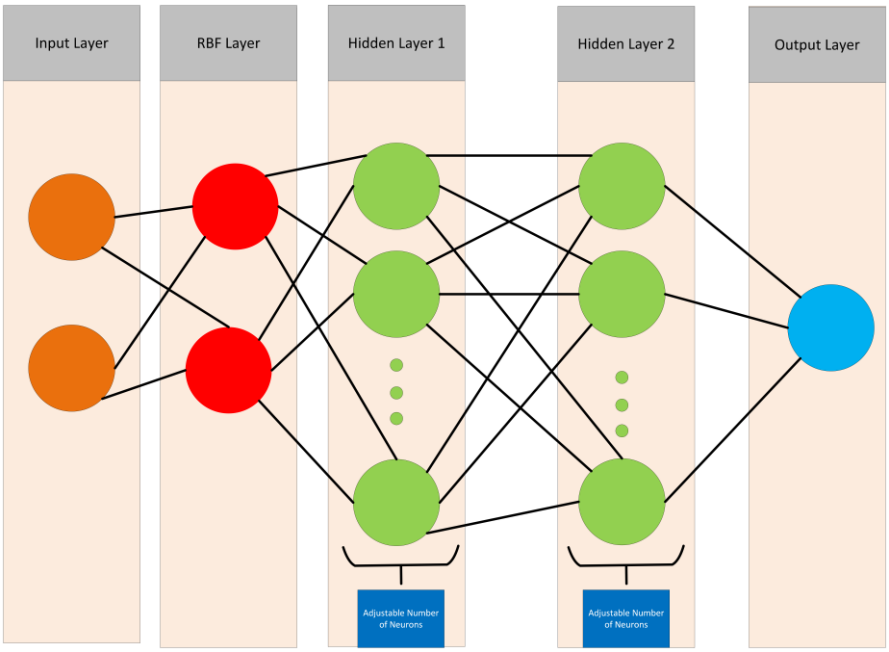

Figure 5.7 RBF Experiments Network

5.3.1 RBF with hr_mean, meanPeakAmplitude

The analysis of the RBF with window size 10 is analyzed in **Table 5.4**. Overall, the window size of **10** has **better** results than the other window sizes but the **difference** is **not so intense** except with the window size of **5**. Also, in the window size of 5, the learning rate was changed as it performed so poorly, but it did not succeed any difference. So, in RBF the default learning rate, 0.001, is also the best, like in MLP.

As shown in **Figure 5.8** the RBF network is learning better than the MLP. It achieves 95% of recall, which is very important for this study, it needs less time to converge and there is a low standard deviation as the metrices are not changing a lot because of the normalization that takes place in the first layer.

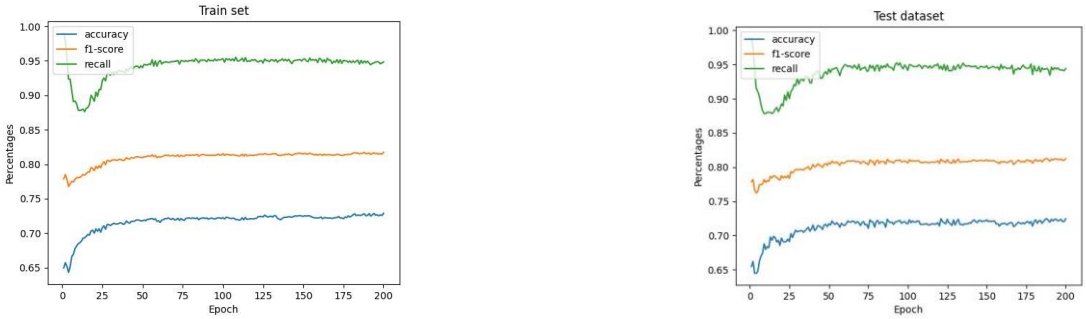

Figure 5.8 RBF with Window Size of 10

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10	100	50	50	Around 65% accuracy, so moderate results.
2		100	200	50	Better than 1.
3		200	300	200	Same as 2 but converges faster.
4		200	500	400	A little better than 3.
5		200	100	100	Same as previous runs.
6		200	600	200	Same as 5.
7		200	300	0	Worse accuracy.
8		200	700	0	Same as 7.
9		200	700	700	Similar to 4.
10	20	200	200	200	Moderate results.
11		200	600	600	Better than 10.
12		200	1000	800	Better than before.
13		200	700	0	Worse results.
14	30	200	300	300	Moderate results.
15		200	500	500	Better results.
16		200	1000	900	Better results than 15.
17		200	700	0	Worse results like in previous windows.
18		200	2000	1800	Same results as 16.
19	40	200	300	300	Moderate results.
20		200	600	500	Better results.
21		200	1200	1100	Better than 20.
22		200	2000	2000	Same as 21.
23		200	700	0	Worse than before.
24	5	300	500	400	Overall, it is not good.
25		300	700	200	Similar to 24.
26		300	600	0	Worse than 24.
27		300	600	600	Similar to 25.
28		300	600	500	The learning rate is 0.5 and the results are still as bad as before.
29		300	600	500	The learning rate is 0.3 but the results similar to 28.
30		300	600	500	The learning rate is 0.1 and is the same with 29.

Table 5.4 RBF Executions window: 10,20,30,40,5

5.3.2 RBF with all the features

In this section the RBF is trained with **all the features** of the dataset for window size of 10 like in Section 5.2.2. The results are displayed in **Table 5.5**.
The conclusions are the exact **same** as in the MLP's section. **Overfitting** phenomenon is happening as there is a lot of correlation among the input data so the network overfits the training data and fails to make good predictions in the testing set.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	200	600	600	Overfitting occurs.
2	200	200	100	Better results, but there is still overfitting.
3	200	30	10	It is not converged.
4	600	30	10	Overfitting like in 1.
5	600	100	0	Same as 4.

Table 5.5 RBF with All Features

5.4 SOM

The Self-Organizing Map of Kohonen is run with window the sizes of 10,20,30 and 40 and overall, the results are **not good** as shown in **Table 5.6**. This network must have a fixed number of neurons as shown in the documentation https://libraries.io/pypi/sklearn-som.
The SOM architecture is displayed in **Figure 3.6**. The number of neurons and the size of the grid must be the same as the number of features. So, as the features are two the grid should be 2x1, SizeP is 2 and sizeY as shown in **Figure 3.6**. The only parameters that are tested and changed are the number of epochs and the learning rate which is defined as how much the network adapts to the new data. With high learning rate the network learns faster which means that the weights of the edges change in a greater way.

To sum up, this network **cannot** produce good results as the best accuracy it has got is around 55%. Also, it is run with all the features and consequently with more neurons, but the results remain the same. It is logical that its results are worse than the MLP and the RBF in the previous sections as it is a type of **unsupervised** machine learning, which **does not use** the **real outcome** to train and in general the clustering algorithms need a lot more data to be trained well. As shown in **Figure 4.3** it was decided that there is **no point** to **run** this NN in the rest of the study as the metrics were so low.

Run #	Window	Epochs	Learning Rate	Comments
1	10	1000	0.5	Really bad results, around 55% accuracy.
2		1000	0.01	Worse than 1.
3		1000	1	Better than 2, but worse than 1.
4	20	1000	0.5	Worse than the previous window.
5		300	0.7	Same as 4.
6	30	500	0.5	Worse than window of size 10.
7		500	0.8	A little better than 6.
8		1000	0.8	Same as 7.
9	40	500	0.5	Same as the previous window.
10		1000	0.7	Worse than 9.

Table 5.6 SOM with all the Window Sizes

5.5 LSTM

In this section the LSTM models are run in all the different window sizes to find the most appropriate for this study. The main LSTM model is displayed in **Figure 5.9** which is just

like the MLP network but in the hidden layers the neurons are more complex just like they are explained in **Figure 3.9**.

In the LSTM models we used there is an additional computation before the input layer of the network. The data that is given in every neuron during the training is not a scalar value but a vector which consists of the value and the next values defined by the value of the variable *lookback*. The lookback variable can be defined as the number of previous inputs or like in this thesis the next inputs that are fed into the network in every iteration.

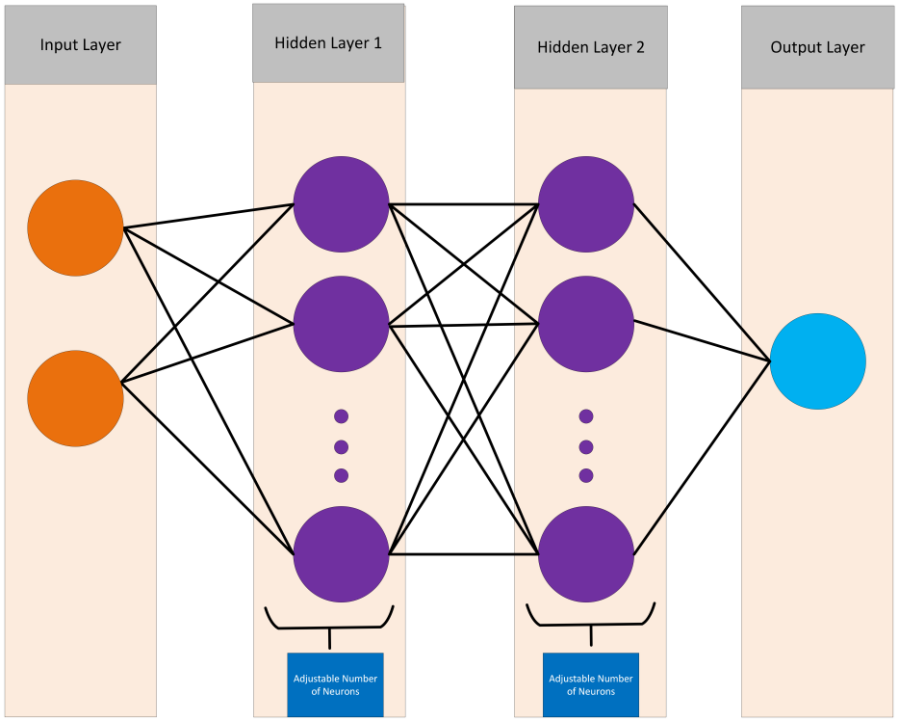

Figure 5.9 LSTM Experiments Network

5.5.1 LSTM and its Variations

In the current section, a lot of different LSTM models' variations are tested for window size of 10. The networks are LSTM, LSTM-MLP which is an **LSTM** network **followed** by an **MLP** network and RBF-LSTM where is an **RBF** network **followed** by an **LSTM** network. A

Also, another **simpler** recurrent network is used which does not have the complexity of LSTM.

5.5.1.1 LSTM model

The LSTM executions are displayed in **Table 5.7**. The best results are with **lookback 2** as shown in **Figure 5.10** and with lookback 5 the difference is very **small**. Overall, **smaller lookback** is **better** as it produces better results in all the metrices for this window size.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	60	32	0	5	Accuracy under 65%.
2	60	100	0		Same as 1.
3	100	100	0		Same as before.
4	600	100	0		Better results.
5	400	300	0		Similar to 4.
6	600	70	32		Better results than with only one hidden layer.
7	400	100	30		A little worse than 6.
8	400	150	100		Worse than 6.
9	400	40	30		It had learning rate of 0.1 and the results were worse than before.
10	600	80	40		It had one more hidden layer of 40 neurons and the results are similar to 6.
11	450	70	32	2	Similar to 6.
12	600	150	32		Similar to 11.
13	400	500	400		Worse than 12.
14	400	30	10		Similar to 11.
15	400	5	2		Worse than 14.
16	600	100	0	10	Worse than previous lookback.
17	400	70	32		The testing is similar to 16.
18	400	200	150		Worse than before.
19	400	70	0	20	Worse than previous lookback.
20	600	70	32		A little better than before.
21	400	200	110		Worse than in 20.
22	400	32	20		Similar to before.

Table 5.7 LSTM of Window 10

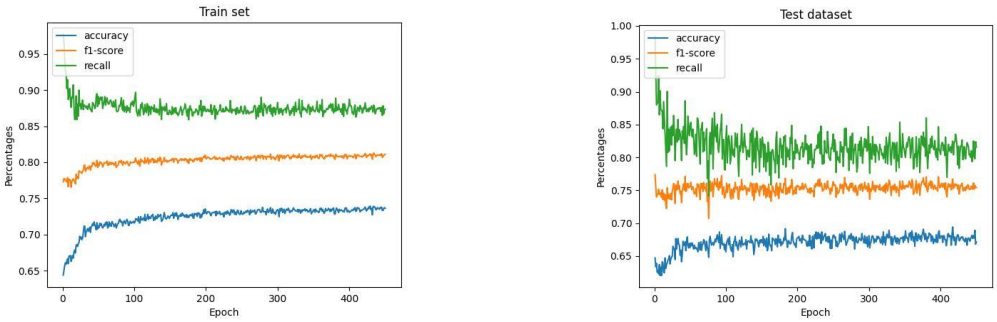

Figure 5.10 LSTM with Window Size of 10

5.5.1.2 LSTM – MLP model

The **LSTM-MLP** network has one or two LSTM layers followed by one or two MLP layers. The results of the network are **worse** than the LSTM model of the previous section and they are displayed in **Table 5.8**. The testing **accuracy** is **under 65%,** the **F1-score** is around **75%** and the **recall** around **85%.**

Run #	Epochs	Hidden layer 1	Hidden Layer 2	MLP Layer 1	MLP Layer 2	Lookback	Comments
1	300	82	40	100	0	5	Worse than the LSTM model.
2	300	72	32	400	0		Similar to 1.
3	300	72	32	200	100		Worse than before.

Table 5.8 LSTM-MLP

5.5.1.3 RBF - LSTM model

In this section an RBF layer of centers is added before the LSTM network to try to normalize the data before they get into the LSTM model. The model is **not good** as it is being **overfitted** and the testing accuracy and other metrics are getting worse over time. The different executions are shown in **Table 5.9.**

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	300	72	32	5	Testing results are low.
2	450	100	50		Same as 1.

Table 5.9 RBF – LSTM

5.5.1.4 Simple Recurrent Neural Network

In the current section a Simple Recurrent Network is used which **does not have** the **complexity** and the **logic** that LSTM has on which data to remember and forgets so it just keeps the last *k* outputs, where *k* is the lookback variable. Its overall results were really bad, **worse** than the LSTM model of the Section 5.5.1.1 for all the different combinations, displayed in **Table 5.10**.

The network does not learn at all, and all the metrics are kept static in both training and testing set. While the F1-score was kept at 99% the accuracy is at 65% which means that it predicts many people as avoidance which they are not and therefore has many False Positives.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	400	70	32	2	Training accuracy at 65%.
2	400	250	150		Worse than 1.
3	400	200	0		Worse than 1.
4	400	70	32	10	Same as before.
5	400	150	0	20	Similar to before.
6	400	150	0		Similar to 5.

Table 5.10 Simple RNN

5.5.1.5 LSTM Variations Conclusion

Overall, among the different variations of LSTM model and the SimpleRNN the **LSTM model** yields the **best** results as shown in **Figure 5.10** and for lookback better results are produced as mentioned before. With that said, it was decided to continue with only the **LSTM** model in the rest of the study as it has the best results, as shown in **Figure 4.3**.

5.5.2 LSTM for Window Sizes of 20,30,40,5 Seconds

In this section, the different experiments for the LSTM model's other window sizes are described. The conclusion from this is that the **window sizes** and the **lookback** are **propositional**. When the window size is small then the LSTM learns better when the lookback is small too and when it is big then the lookback has to be big too. This network yields similar results for all the window sizes but when the window size is 5 and lookback is 2 the best network appears as shown in **Figure 5.11**. The different experiments are shown in **Table 5.11**.

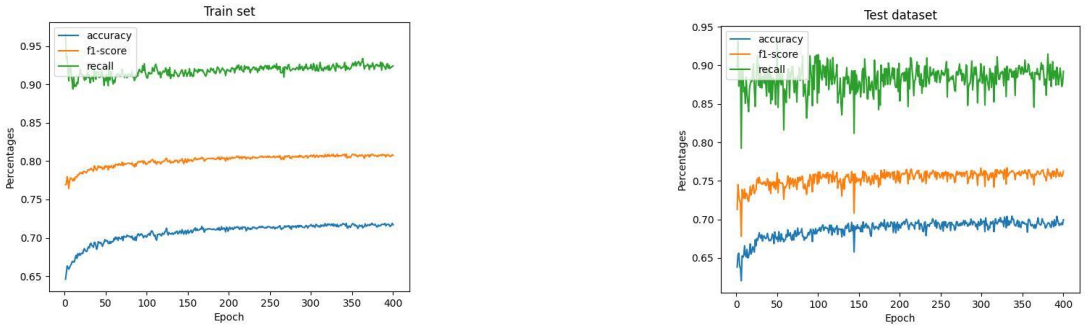

Figure 5.11 LSTM Model with Window Size of 5

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	20	400	70	32	2	Testing is a little worse than previous window.
2		400	150	100	2	Worse than 1.
3		400	150	120	10	Worse than 1.
4		400	70	32	10	Testing worse than 3.
5	30	400	70	30	2	Worse than before.
6		400	150	100	2	Similar to 5.
7		400	150	200	10	Better results than 6.
8		400	70	30	10	Similar to 7 and the training are a little better than the testing results.
9		400	40	0	10	Worse than 8.
10	40	400	70	30	2	Worse than 7, 60% accuracy.
11		400	200	150	2	Similar to 10.
12		400	200	150	10	Better than 11.
13		400	200	150	10	Better than 12.
14		400	200	150	20	Better results and more stable.
15	5	400	70	30	10	Similar to window 10 best results.
16		400	70	30	2	Better than 15 especially in the testing set.
17		400	200	150	2	Worse than 16.
18		400	70	0	2	Worse than before.

Table 5.11 LSTM with Window Sizes of 20,30,40,5

5.6 RBF – MLP – LSTM Model

The RBF – MLP – LSTM model is a combination of the previous models, and it is shown in **Figure 5.13**. It consists of an RBF, two MLP and two LSTM layers which are defined as hidden as shown in **Figure 5.12**. All the different layers are working in the exact same way as was explained before in this chapter. The experiments are displayed in **Table 5.12** and the result of the best network of this model can be seen in **Figure 5.13**.

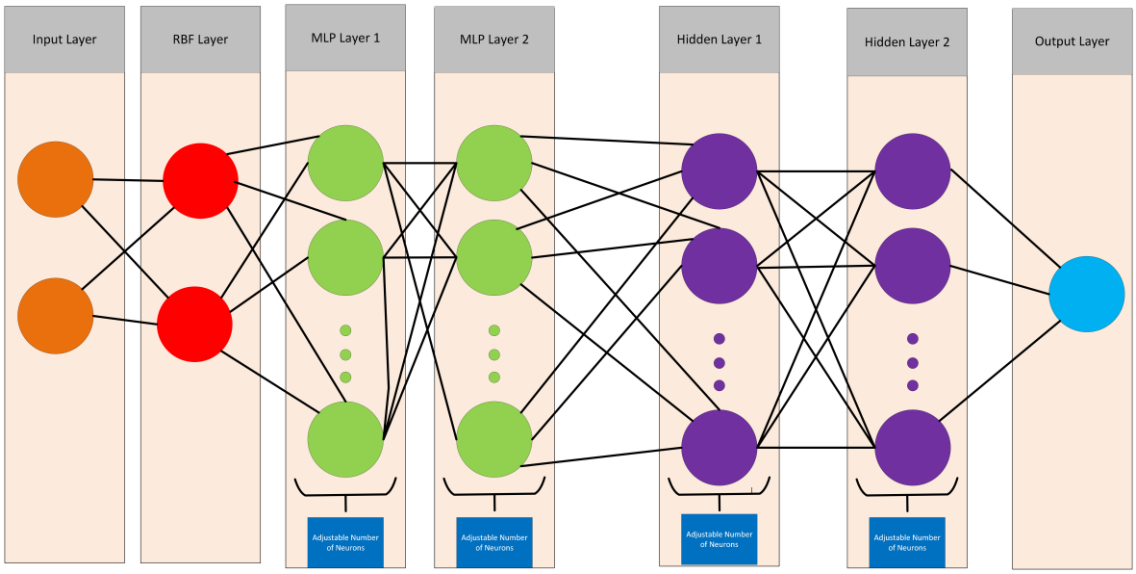

Figure 5.12 RBF-MLP-LSTM Experiments Network

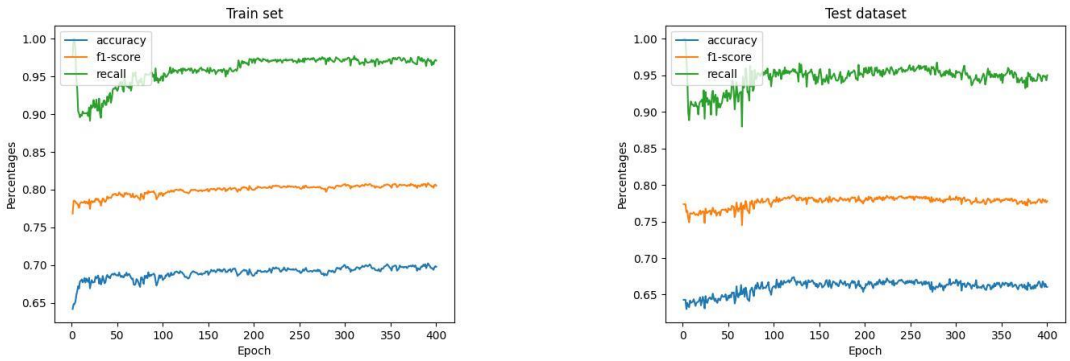

Figure 5.13 RBF-MLP-LSTM model with window size of 10

Run #	Window	Epochs	MLP Layer 1	MLP Layer 2	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	10	400	150	100	70	30	2	Moderate results.
2		200	500	400	70	30	2	Similar to 1.
3		200	500	400	70	30	10	Overfitting phenomenon.
4		200	150	400	70	30	10	Similar to 3.
5		200	200	150	70	30	10	Similar to 4.
6		200	50	100	70	30	2	Similar to 1.
7		200	50	30	70	30	2	Better than 1.
8		200	150	100	10	5	2	Worse than 7.
9		200	150	100	150	90	2	Worse than 1.
10	20	200	150	100	70	30	2	Moderate results.
11		200	150	100	70	30	10	Worse than 10.
12		200	100	100	70	30	2	Similar to 10.
13	30	200	100	100	30	30	2	Worse than previous windows.
14		200	150	100	70	30	2	A little better than 13.
15		200	150	100	70	30	10	Better than 14 .
16		200	150	100	70	30	20	Unstable results.
17		200	400	250	70	30	20	Worse than before.
18	40	200	150	100	70	30	2	Moderate results
19		200	150	100	70	30	10	Better than 18.
20		200	150	100	70	30	20	Better than 19.
21		600	150	100	70	30	20	Similar to 20.
22		400	150	100	200	150	10	More unstable network.
23	5	200	150	100	70	30	2	Bad results
24		200	150	100	70	30	10	Similar to 23.

Table 5.12 RBF – MLP – LSTM with All the Window Sizes

5.7 Comparison of All Networks in RWM

The best networks are the **MLP** and the **RBF** with **window size** of **10**. But overall, the **RBF** is **better** as it achieves better **recall**, and the other metrics are around the **same**. As shown in **Figure 5.8**, the training and testing **accuracy** is around **73%**, the **F1-score** around **80%** and the **recall 95%** which is very important for the goal of the thesis as mentioned in a previous chapter.

Chapter 6

Analysis Using the Moving Window Methodology

In this chapter, **Phase 3** is presented as shown in **Figure 4.4**. The Moving Window Methodology is used to multiply the data instead of the Rectangular Window Methodology and MLP, RBF, LSTM and RBF-MLP – LSTM models are used with big and smaller slides to make the predictions.

6.1 Moving Window Methodology

The Moving Window Methodology (MWM) [21] is similar to Rectangular Window Methodology as it also multiplies the data by creating artificial samples using windows. The difference is that instead of each sample starting right after another ends, it **starts** after a **specific time unit** t **after** the **previous** one as shown in **Figure 6.1**, so there is some overlap between the windows.

The MWM is based on the same assumption as the RWM, that a person's emotional state constantly fluctuates throughout time. A previous study [21] explains that a hidden danger of this technique is the overfitting phenomenon because each sample has a difference of only t from the previous one, so they are almost identical. To resolve this problem in every NN different windows-slides were tested, where window is the size of the whole window of each new record that is created, and slide is the variable t of the **Figure 6.1**, where is the time unit that each window starts after the previous one. Also, we certainly believe that the Neural Networks will not have the problem of overfitting as by their

nature need a lot more data that the standard machine learning algorithms that were used in [21].

The Moving Window Methodology produces **more rows** than the RWM as shown in the **Table 6.1**.

Total Number of Samples		
Window Size (sec)	Slide	Wearable Devices
10	5	2882
20	10	1089
30	10	996
10	2	5716
20	2	5251
30	2	4887

Table 6.1 Total Number of Samples in MWM

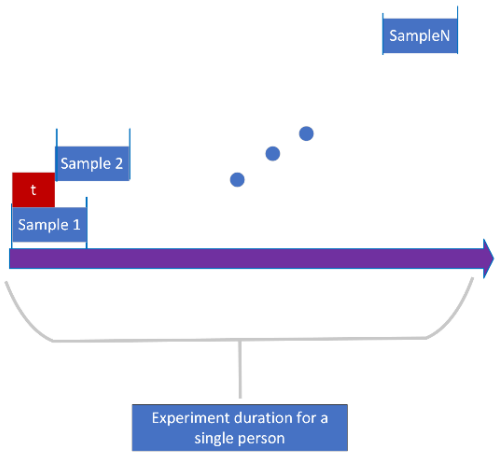

Figure 6.1 Moving Window Methodology

6.2 Big Slides

In this section the different NN are used for bigger slides such as 5 and 10. More precisely the combinations used are window 10 and slide 5, window 20 and slide 10 and window 30 and slide 10.

6.2.1 MLP

The MLP results are very similar to the previous methodology as shown in **Figure 6.1**. The best combination is the seventh as shown in **Table 6.2** with the window size of 20 and slide 10.

Figure 6.2 MLP

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 5	200	150	100	Worse results than in RWM.
2		400	600	400	Similar to 1.
3		400	800	600	Similar to 1.
4		200	50	20	Worse than before.
5		600	50	20	Similar to 4.
6	20 – 10	200	150	100	Worse than 1.
7		200	500	400	Better than before, similar to the best of RWM.
8		200	1000	800	Similar to 7.
9		400	150	100	Similar to 7.
10		600	150	100	Similar to 9.
11	30 - 10	200	500	400	It is not converged.
12		400	500	400	Better results but the results are worse than 7.

Table 6.2 MLP

6.2.2 RBF

The best combination of window slide is the **same** as in Chapter 6.2.1 and the results are similar to the RBF of the RWM shown in **Figure 5.8**. The different executions can be seen in **Table 6.3**.

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 5	200	100	50	Moderate results.
2		200	150	200	Similar to 1.
3		400	400	200	Better than 1.
4		200	500	400	Similar to 3.
5		200	1000	800	Similar to 4.
6	20 – 10	200	800	700	Better than before.
7		200	500	400	Similar to 6.
8		200	150	100	Worse than 6.
9	30 - 10	200	500	400	Worse than 7 un F1-score.
10		200	100	80	Worse than 9
11		200	1000	800	Similar to 9.

Table 6.3 RBF of MWM with Big Slides

6.2.3 LSTM

In LSTM the best window-slide is **10 - 5** instead of 20 – 10 like before and the lookback variable of value **two** is the best in all of them. The best result is **slightly better** than in the RWM as the testing accuracy increases from 67 to 70 %, as shown in **Figure 6.3** and the different executions are displayed in **Table 6.4**.

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	10 - 5	200	100	80	10	Worse than previous Methodology.
2		200	100	80	5	Better than 1.
3		200	70	30	2	Slightly better than 2.
4	20 – 10	200	70	30	10	Worse than 3.
5		200	70	30	5	Better than 4.
6		200	70	30	2	Better than 5, but worse than 3.
7		200	400	300	2	Worse accuracy.
8		400	70	30	2	Similar to 6.
9		200	20	15	2	Similar to 8.
10	30 – 10	200	70	30	2	Similar to 6.
11		200	70	30	10	Worse than 10.
12		200	10	8	2	Worse than 11.

Table 6.4 LSTM of MWM with Big Slides

Figure 6.3 LSTM

6.2.4 RBF – MLP - LSTM

The results in all the MWM are **worse** than the RWM as the accuracy remains steady at 65% and recall at 99%, that means that the network just predicts **everybody as avoidance** without actually learning and generalizing. The different experiments are shown in **Table 6.5**.

Run #	Window - Slide	Epochs	MLP Layer 1	MLP Layer 2	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	10 - 5	300	150	100	70	30	2	Bad results, the network does not learn at all.
2		200	100	80	50	30	2	Worse than 1.
3		200	500	400	70	30	2	Similar to 2.
4		200	500	400	70	30	10	Similar to 3.
5	20 - 10	200	500	400	70	30	2	Similar to 1.
6		200	500	400	70	30	10	Worse than 5.
7		200	50	20	30	30	2	Better than 5.
8		200	10	13	10	5	2	Similar to 7.
9	30 - 10	200	500	400	70	30	2	Worse than before.
10		200	50	20	30	30	2	Similar to 9.

Table 6.5 RBF – MLP – LSTM of MWM with Big Slides

6.3 Small Slides

In this section the different NN are used for **smaller slides,** like two. The windows used are 10, 20 and 30.

6.3.1 MLP

The best results of the MLP are very similar in the different window – slides but the window size of **20** and slide of two gives **slightly** the best result of them as displayed in. As can be seen in **Figure 6.4**, the results are slightly better than in the RWM as the accuracy goes up to 75%. The **Table 6.6** shows in detail the experiments made for MLP.

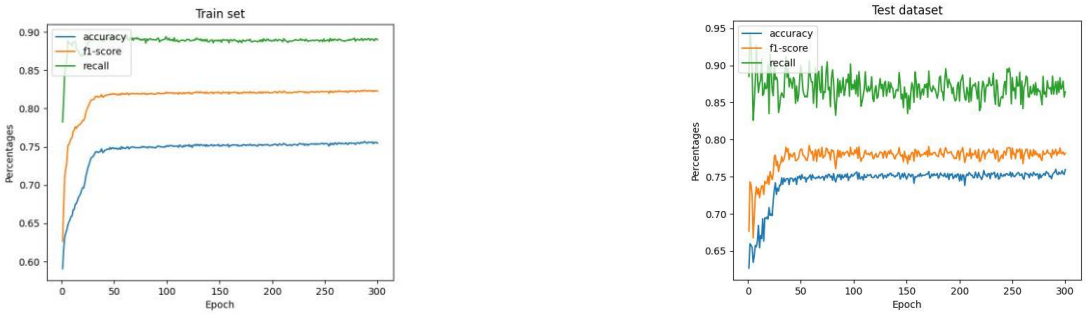

Figure 6.4 MLP of MWM with Small Slides

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 2	150	520	400	Moderate results.
2		150	800	700	Better than 1.
3		150	1500	1300	Similar to 2
4	20 - 2	300	500	400	Better than 2.
5		150	150	100	Worse than 4.
6		200	800	700	Worse than 4.
7	30 - 2	150	500	400	A little worse than 4.
8		200	900	800	Similar to 7.
9		200	80	60	Worse than 8.

Table 6.6 MLP of MWM with Small Slides

6.3.2 RBF

The RBF **produced similar** results to the Rectangular Window Methodology, and the experiments are displayed in **Table 6.7**. The best network is the second with 150 epochs, 500 and 400 in the first and second hidden layer respectively in the window size of 10 with slide 2 seconds.

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 2	150	500	400	Moderate results.
2		150	100	80	Similar to 1 with better recall.
3		150	20	15	Worse than 2.
4	20 - 2	150	500	400	Similar to 2 but recall is a little lower.
5		150	100	80	Similar to 4.
6		30	100	20	Worse than 5.
7	30 - 2	200	500	400	Similar to 4.
8		200	100	80	Similar to 7.
9		200	1000	800	Similar to 7.

Table 6.7 RBF of MWM with Small Slides

6.3.3 LSTM

The LSTM produced **slightly better** results in the **MWM** than in the RWM as accuracy increased to 75% in the window size of 30. The different experiments are displayed in **Table 6.8**.

Run #	Window - Slide	Epochs	Hidden Layer 1	Hidden Layer 2	Lookback	Comments
1	10 - 2	150	70	30	2	Slightly better than in RWM.
2		150	70	30	5	Worse than 1.
3		150	200	180	2	Worse than 1.
4	20 - 2	150	70	30	2	Similar to 1.
5		150	70	30	10	A lot worse than 4.
6		150	200	180	2	Worse than 4 in training set.
7	30 - 2	200	500	400	2	Worse than 1.
8		200	500	400	10	Similar to 7.
9		200	50	40	2	Better than all previous runs.
10		200	10	6	2	Similar to 9, with better recall.

Table 6.8 LSTM of MWM with Small Slides

6.3.4 RBF – MLP - LSTM

Overall, this NN's results are a lot worse than in the RWM as the accuracy converges at 65% and recall to 100%, so it predicts everybody as avoidance which means that it cannot actually succeed generalization and it fails to differentiate the two classes. The experiments are shown in **Table 6.9**.

Run #	Window - Slide	Epochs	MLP Layer 1	MLP Layer 2	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	30 - 2	1000	500	400	70	50	2	Similar results with the RELU activation function.
2		200	50	40	50	40	2	Similar to 1.
3		200	500	400	50	40	2	Similar to 1.

Table 6.9 RBF – MLP – LSTM of MWM with Small Slides

6.4 Comparison of MWM

The MWM with different slides, big and small, gave some interesting results in the current study. Initially there was the idea that with the small slide of 2, the NN networks **would overfit** and would **not** be able to **generalize** as [21] analyses for the standard machine learning algorithms. But actually, it produces the **same** or **even better results** in some networks.

Both the **RBF** and the **RBF – MLP -LSTM** networks produce the same metrics with different sizes of slides. But the **MLP** and especially the **LSTM** produces **better results**, as the LSTM's accuracy increases to 73% as can be seen in **Figure 6.5**.

The **MLP** is still the **best model** in the **Moving Window Methodology** as shown in **Figure 6.4**.

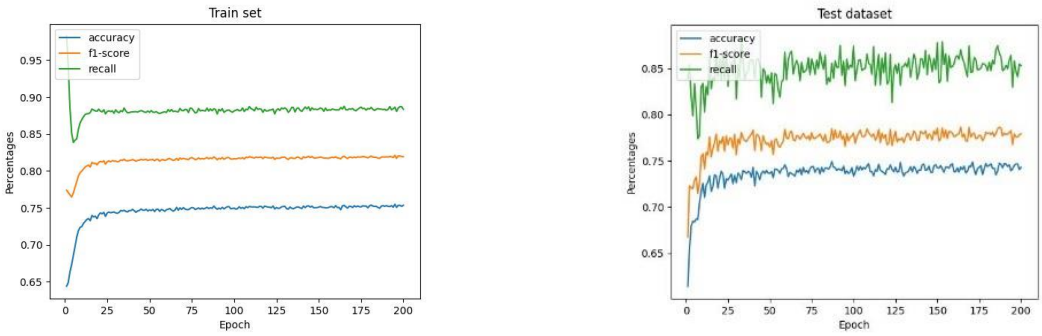

Figure 6.5 LSTM of MWM

6.5 Comparison of MWM and RWM

The comparison of the two multiplication methodologies **cannot** give a clear winner as it depends on the network in which methodology can be learned better.

Firstly, the **RBF – MLP – LSTM** model is **better** in the **RWM**, and its best combination is shown in **Figure 5.13**. The **LSTM** and the **MLP** succeed **better generalization** in the **MWM** as shown in **Figure 6.3** and **Figure 6.2** respectively.

Finally, the **RBF** can produce the exact **same** results in **both** methodologies as displayed in **Figure 5.8**. The **RBF** results can be considered the **best** so far as it achieves the best metrics in total. More precisely, while its **accuracy** is **slightly lower** than in the MLP its recall is around **10% higher**, which is really important for this kind of study as explained in previous chapters.

Chapter 7

Analysis Using a Different Activation Function

In this chapter, **Phase 4** is explained as shown in **Figure 4.5**. The **activation function** is changed to **tanh** and different NN like MLP, RBF and the RBF-MLP-LSTM, are used on the datasets produced by the two multiplication techniques, RWM and MWM.

7.1 Tanh Function and Hypothesis

First of all, as explained in Chapter 2, the neurons of the NNs compute the **sum** of their **in going edges**, which are first **multiplied** by the **weights** of each edge. Then, this result is used as an **input, *X*,** in an **activation function** and the value of the outgoing edge is created. So far, the **RELU** activation function has been used which is defined as **max {0,*X*}**, where *X* is its input, which means that every **negative** input value becomes **zero** and the **positive** input values do **not change**.

The activation function that is used in this section is **Tanh** which is equal to $((e^{x})-(e^{-x}))/((e^{x})+(e^{-x}))$. The main purpose of an activation function is to represent the data, given the input values x to produce the result value y and to be as close it can be to the real value. The RELU function probably deletes useful data as it makes all the negative results to 0. On the other hand, the Tanh function does not delete any information, its output is from -1 to 1 and the larger the input is, the closer to 1 its output is. Also, its Zero centered; the output can be mapped as negative, neutral, and positive [53].

The main **hypothesis** is that the networks **cannot** predict the **acceptance subjects** , the 0's, easily so maybe **information is missing** after **converting** all the **negatives** values to **zero**. Also, the **Tanh** is a function that can be **derived** which is very important in NN such as **MLP**, **RBF** as they are trained with the **backpropagation** method that uses the derivative.

7.2 Analysis Using the Moving Window Methodology

In this section, the RBF – MLP – LSTM, MLP and RBF NN with the tanh activation function are executed on datasets created by the MWM.

7.2.1 RBF – MLP – LSTM

The RBF – MLP – LSTM network's different experiments are displayed in **Table 7.1**. The results are the **same** as with the RELU activation function. So, there is **no point** in continuing running this particular NN as it **cannot** succeed in any of this study's goals.

Run #	Window - Slide	Epochs	MLP Layer 1	MLP Layer 2	Hidden layer 1	Hidden Layer 2	Lookback	Comments
1	30 - 2	1000	500	400	70	50	2	Similar results with the RELU activation function.
2		200	50	40	50	40	2	Similar to 1.
3		200	500	400	50	40	2	Similar to 1.

Table 7.1 RBF – MLP – LSTM

7.2.2 MLP

In the MLP network, the results are very **similar** to when Relu is used as shown in **Figure 6.2** as the accuracy is around 75%, F1-score at 80% and recall at 87% as displayed in **Figure 7.1**. The different experiments are shown in **Table 7.2**.

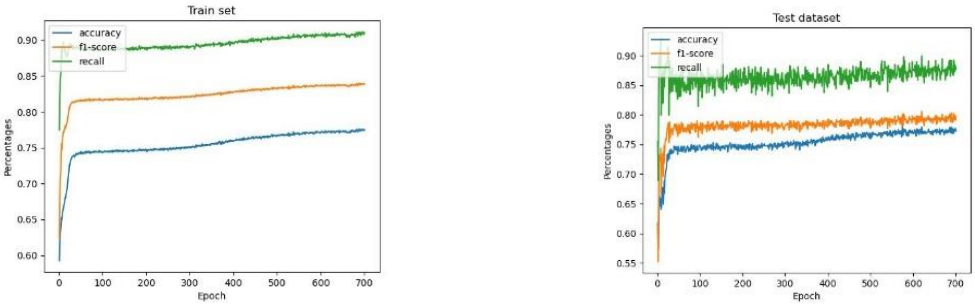

Figure 7.1 MLP of MWM with Tanh

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 2	700	500	400	As good as with RELU activation function.
2		800	1000	800	Similar to 1.
3		800	100	80	Worse than 1.
4	30 - 2	300	100	80	A little worse than 1.
5		300	200	100	Similar to 4.
6		200	500	400	Similar to 4.
7		300	1000	800	Worse than before.
8	30 - 10	500	500	400	Similar to 5.
9		500	1000	800	Similar to 8.

Table 7.2 MLP of MWM with Tanh

7.2.3 RBF

The RBF with the Tanh activation function in both hidden layers produce a **lot better** results than in the previous activation function as the **accuracy** increases to **80%** from 73%, the F**1-score** to **83%** from 80% and the **recall** decreases to **87%** from 95% as shown in **Figure 7.2**. While the recall has fallen, the fact that both accuracy and F1-score increase indicate that this network is **better** and achieves **better generalization**. The different experiments are shown in **Table 7.3**.

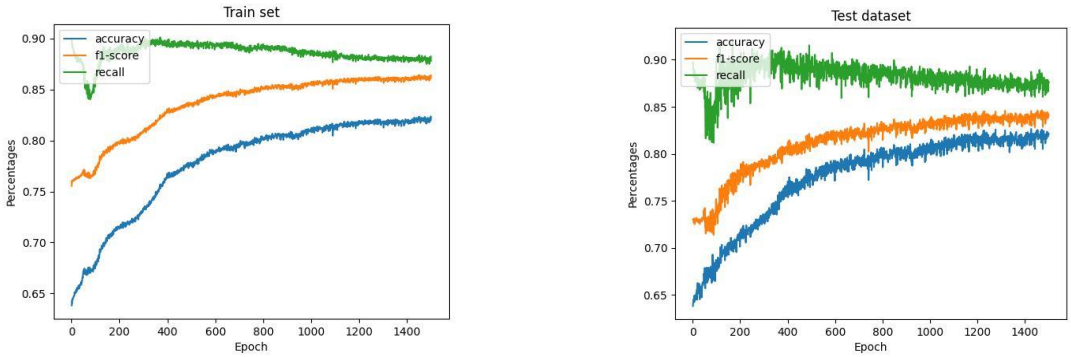

Figure 7.2 RBF of MWM with Tanh

Run #	Window - Slide	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10 - 2	1000	800	700	It is not converged.
2		1500	500	400	Better results than the previous activation function.
3		1800	500	400	Better results than 2.
4	30 - 2	300	500	400	It is not converged.
5		1000	500	400	Better than 4.
6		800	800	700	Worse than 5.
7		1000	800	700	Better than 5 and 3.
8		1500	800	700	Similar to 7 but converged
9		3000	800	700	Same as 8, so it does not need so many epochs.
10		800	50	40	Worse than before.
11		1000	1500	1300	It is not converged.
12		1400	1500	1300	Similar to 7.
113		800	500	400	In the second hidden layer the RELU function was used but the results were worse than before.
14	30 – 10	1800	500	400	Good results but it is not converged.
15		4000	500	400	Good results but worse than 7.

Table 7.3 RBF of MWM with Tanh

7.3 Analysis Using the Rectangular Window Methodology

In this section the MLP and RBF are used with the **Tanh** activation Function as they have the best results so far, with dataset created by RWM. Also, the window size is **10** seconds as it has slightly better results than the other windows sizes.

7.3.1 MLP

The MLP in the RWM with the Tanh activation function is **slightly worse** than with the RELU function as the **accuracy** and the F**1-score drop** a little. The experiments can be seen in **Table 7.4**.

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10	1000	800	700	Worse results than with RELU activation function.
2		1000	400	300	Similar to 1.

Table 7.4 MLP of RWM with Tanh

7.3.2 RBF

The RBF with the Tanh activation in the RWM is a **lot worse** than with the RELU function as the accuracy **drops** to 70%, the F1-score to 79% and the recall drop to 87%. The experiments are displayed in **Table 7.5**.

Run #	Window	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	10	500	500	400	Worse than the RELU activation function.
2		500	1000	800	Similar to 1.

Table 7.5 RBF of RWM with Tanh

7.4 Comparison of Results

It seems that the combination of the models, **RBF – MLP – LSTM**, **cannot** produce good results with this activation function either.
Also, both **MLP** and **RBF** results in **RWM** are **similar** and **worse** than with the **RELU** function. That is probably because there are **not so many data to train** on.
But, in the **Moving Window Methodology**, the **MLP** results are **similar** and **slightly better** with the **Tanh** function.
Finally, the **RBF** with the **Tanh** activation function, in the **MWM**, is a **lot better** as it achieves **better accuracy** and **F1-score** metrics, and the **recall drops** a little as displayed in **Figure 7.2**.
So, in conclusion, the activation function of **Tanh** seems to be **working better** with **bigger datasets**, that are created using the **MWM**, and can **generalize** more **precisely** as it predicts the **acceptance** with more **accuracy** and the **recall**, of the avoidance people, does not drop by a lot. Also, with the **MWM** the **RBF** works **better** as **normalization** takes place in the first layer which probably stops any overfitting phenomena that could occur.

7.5 Comparison with Previous Study

As described in Chapter 2, the **best** Machine Learning algorithm in the **previous study** [22] is the **Bagging Decision Tree** in the **window** size of **10** in the **Rectangular Window Methodology** (RWM). Its **accuracy** is **85%,** the **F1-score** is **83%** and **recall 89%**. The results of the NN's are **worse** in the **RWM** than the **standard** machine learning algorithms as explained in Chapter 5. The best Neural Network in the Rectangular Window Methodology of window size of 10 is the **RBF** network with **similar F1-score** of 80%, **worse accuracy** of 73% and **better recall** of 95%. As can be seen, the results are a little worse in the NN's, as it **cannot** predict the **acceptance** as good as the **Bagging Decision Tree**. But it can predict the **avoidance** class a **little better**, by 5%. As it was displayed in the previous study [22] all the different traditional machine learning models have similar results but in the territory of the NN some networks like SOM and LSTM had a lot worse performance than others, like MLP and RBF.

The fact that the NNs performed worse in the RWM was the reason to try new multiplication techniques such as the **Moving Window Methodology** (MWM) to create **bigger** datasets, as it is commonly known that Neural Networks **need more data** than the traditional machine learning models.

While the standard machine learning algorithms were overfitted when the MWM was used [21] this does not happen in the Neural Networks. Using the MWM better results are achieved in some NN's as explained in Chapter 7. The best NN as shown in **Figure 7.2** is the **RBF**, with the **tanh** activation function, in the **Moving Window Methodology** with window size of 30 and slide two. Its **accuracy** is **82%,** the **F1-score** is **84%** and the **recall** is **87%.**

So, the **F1-score** of the **RBF** is a little **better** than the best traditional machine learning model of the RWM mentioned before, the **recall** is slightly **lower**, and the **accuracy** is around 3% **lower**.

The results of the best traditional machine learning algorithm of RWM and the best NN of MWM are **very similar,** and their differences are in the bound of the standard deviation. So, it seems that the NNs can be **useful** in this study too as they can produce **similar results** to the **traditional** Machine Learning algorithms with the help of other data multiplication techniques that produce bigger datasets than the RWM.

Chapter 8

Extension to Real Time Data

In the current chapter **Phase 5** as shown in **Figure 4.6** is analyzed. A new experiment is described which measures people's psychophysiological signals in real time via a wearable device. The best NN are run for this experiment such as **RBF** and **MLP**. Also, **comparisons** take place with a related study [49] that uses the traditional machine learning algorithms to make classifications for this experiment. No method of data multiplication is used.

8.1 Description of Experiment

The experiment was conducted by the Psychology department of UCY. The people that took part in this experiment were provided with **smartphones** and a **wearable psychophysiological monitor** that they wore for three days. This device is the **Empatica E4 wristband** [54], which is designed to collect and analyze real time physiological data for a lot of days continuously [55]. It has a PPG sensor, it measures blood volume pulse to derive heart rate, heart rate variability and it has an EDA sensor [56] [57], so it measures similar signals with the experiment that was analyzed so far.

The participants responded to questions on the smartphones at fixed times throughout the three days. Then, using the answers they are classified as **avoidance** or **acceptance** or ***mindfulness*** during that time. The avoidance and acceptance are the two categories that were explained in previous chapters. The **mindfulness** is a new category in which someone is classified when he is neither an acceptance nor an avoidance. Informally speaking, it is an intermediate state of the acceptance and avoidance state.

In the experiments the number of avoidance samples is 124, the acceptance 79 and the mindfulness 316.

8.2 Optimized Dataset with Mindfulness Treated as Avoidance

The dataset is trimmed and preprocessed as explained in [49]. This dataset is created after removing invalid samples. The samples tha are removed are those with a mean temperature under 28°C as 28 degrees Celsius is approximately room temperature and this indicates that the patient **was not wearing** the device when the signals were recorded. Also, records of people who did not use the techniques that were taught in the lab were removed.

Records from all the three categories were removed so the three categories still have a good representation in the dataset. In the current section the **mindfulness** people are treated as **avoidance** and the dataset consists of **220 records**. In this dataset all the 39 statistical features are used to classify the people as avoidance or acceptance.

In this section both activation functions are used, the Tanh and the Relu, for the MLP and RBF networks.

8.2.1 MLP

In the current section the different MLP experiments are displayed. Two different activation functions are used for the MLP network, the Relu and the Tanh.

8.2.1.1 MLP with Relu Activation Function

The overall results for this network are **good** as around **90% accuracy** is achieved and **95%** of **recall** and **F1-score**. The different experiments are displayed in **Table 8.1**.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	**700**	**1000**	**900**	**Good results, 90% of accuracy.**
2	700	1500	1000	Worse than 1.
3	700	500	400	Similar to 1.
4	1000	600	600	Worse than before.
5	1000	100	80	Worse results, especially in the training set.

Table 8.1 MLP of Real Time Experiment Optimized with Relu

8.2.1.2 MLP with Tanh Activation Function

The MLP with the Tanh activation function is **even better**, as it achieves **94%** of **accuracy**, **97%** of **F1-score** and **99%** of **recall** as shown in **Figure 8.1**. The different experiments are shown in **Table 8.2**.

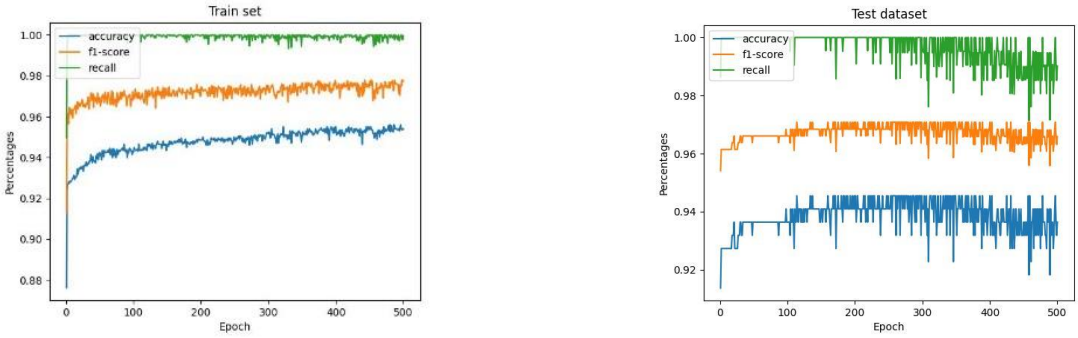

Figure 8.1 MLP of Real Time Experiment Optimized with Tanh

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	500	600	300	Better results than with the Relu function.
2	500	600	600	Worse testing accuracy than 1.
3	500	600	10	Worse than before.
4	500	1000	900	Worse than before.
5	700	500	400	Similar to 1.
6	1000	100	80	Testing accuracy is worse than 5.
7	700	200	100	Similar to 6.
7	500	600	300	Similar to 1.
8	4000	600	300	Same as 6.

Table 8.2 MLP of Real Time Experiment Optimized with Tanh

8.2.2 RBF

Two different RBF networks were executed, one with **Relu** and one with the **Tanh activation function** as happened previously with the MLP. As the dataset has **39 features**, **39 centers** are used as the number of features has to be the same as the number of centers as explained before.

8.2.2.1 RBF with Relu

The NN performed **poorly** as the **accuracy** was **under 30%** and the experiments can be seen in **Table 8.3**.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	500	500	400	Accuracy under 30%.
2	500	30	40	Similar to 1.

Table 8.3 RBF of Real Time Experiment Optimized with Relu

8.2.2.2 RBF with Tanh Activation Function

The RBF is also used with Tanh activation function instead of Relu and the results are similar as in Section 8.2.2.1. The experiments are shown in **Table 8.4**.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	500	500	400	Similar with the relu activation function.
2	500	30	40	Similar to 1.
3	500	1500	1100	Similar to 1.

Table 8.4 RBF of Real Time Experiment Optimized with Tanh

8.2.3 Comparison of MLP and RBF

The **RBF** performed a lot **worse** as the network could not learn at all. This is probably because of the fact that **39 centers** are being used. Because of the small size of the dataset the centers could not be trained well, and the normalization process was not done correctly.

The **MLP** performed **really good**, especially with the **Tanh** function as shown in **Figure 8.1** as the accuracy is at 94%, F1-score at 97% and recall at 99%.

But there is a problem in all these metrices. The number of avoidance and acceptance in this dataset is **not similar** so the **F1-score** does not give the full picture of the network. The network cannot predict the **acceptance** as the *specificity*, which is the **corresponding** metric of the **recall** for the **acceptance** people, is **around 50%.** This is logical though as only a very little number of acceptance people exist in the dataset. For these reasons, the original dataset is used to make the predictions, as displayed in the next section, in hope that the specificity metric can increase too.

8.2.4 Comparison with Other Study

In the current section a comparison takes place with another study [49] that uses these data to make classifications with the standard machine learning algorithms. The results

are very close as in the other study the **accuracy** remains the **same** at 96%, and the **F1-score** is a little **worse** at 75% instead of 97% that is in the current study. Also, the same problem appears, as the **specificity** value is only at 52%.

8.3 Original Dataset

In the current section there is not a preprocessing of the data, so the outliers of the previous section are **not removed**. Also, the people defined as **mindfulness** are **removed** from this dataset, so the number of records is 203. In this dataset the number of acceptance and avoidance is **closer** to each other, as the number of avoidance is 79 and acceptance is 124, so there is more data to succeed the **generalization**.

Initially the network is trained with all the features and then the feature selection made in [49] is followed. The feature selection process of [49] is similar to the feature selection of the previous study [22] that is explained in Chapter 2, so it is generic and useful for Neural Networks too. The features selected are hrv_rmssd, hrv_sdsd, hrv_sdnn, hrv_lf, hrv_hf, temp_mean, bpm, ibi.

These features are extracted from PPG signal and are based on Heart Rate Variability, known as RR intervals, as explained in Chapter 2. **RR intervals** are also referred as NN intervals and are the **time elapsed** between **two consecutive** normal R peaks which are the R peaks that do not include artifacts.

Hrv_rmssd is the root mean square of successive RR interval differences, hrv_sdsd is the standard deviation of successive differences between consecutive RR intervals and hrv_sdnn is the standard deviation of NN intervals. Hrv_lf and hrv_hf are the power in the low and high frequency band respectively. Temp_mean is the mean value of a person's temperature, bpm is the beats per minute, which indicates the average number of heart beats per minute, and IBI is the Inter-Beat Interval, which is the average of RR intervals, the interval of consecutive R waves in milliseconds.

Only the MLP network with the Tanh activation function is used in this dataset as it provided the best results in the preprocessed dataset of Section 8.2.

8.3.1 MLP Trained with All Features in Original Dataset

In this section the MLP with all the features is executed and the results of the original dataset are moderate. Despite the fact that the dataset is small, the **recall** is **around 90%,** F**1-score** is at **75%, accuracy** at **65%** and **specificity** at **40%** as shown in **Figure 8.2**. The experiments are displayed in **Table 8.5**.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	600	100	80	Moderate results.
2	600	300	200	Similar to 1.
3	600	600	300	Better than the previous.
4	600	1000	300	Worse than before.

Table 8.5 MLP of Real Time Experiment Original with All Features

Figure 8.2 MLP of Real Time Experiment Original with all Features

8.3.2 MLP with Feature Selection in Original Dataset

The MLP is trained on the original dataset with the **same features** as in the other study [49] and the executions can be seen in **Table 8.6**. The features are hrv_rmssd, hrv_sdsd, hrv_sdnn, hrv_lf, hrv_hf, temp_mean, bpm, ibi which are explained in the beginning Section 8.3.

The results are **good**, the **accuracy** is around **60%,** the F**1-score** is around **70%,** the **recall** is at **75%** and the **specificity** at **85%**, as shown in **Figure 8.3**.

Run #	Epochs	Hidden layer 1	Hidden Layer 2	Comments
1	600	700	500	Moderate results with a lot of standard deviation.
2	600	300	200	A little better than 1.
3	600	200	100	Similar to 2.
4	600	150	100	The best in all metrices.
5	600	80	60	Overfitting phenomenon as the testing results get worse throughout the epochs.

Table 8.6 MLP of Real Time Experiment Original with Same Feature Selection as [49]

Figure 8.3 MLP of Real Time Experiment Original Dataset with Same Feature Selection as [49]

8.3.3 Comparison of Feature Selection in Original Dataset

Both features' selections produce **different results**. While the feature selection of Section 8.3.1 has **better** results in **accuracy**, **F1-score** and **recall,** the feature selection of Section 8.3.2 is preferred as the **specificity increases** from **40%** to **85%**. This indicates that when **all** the **features** are used the network **cannot succeed generalization** in **both** classes and it **fails** to learn the **acceptance class**. When only the eight features analyzed before are used, the **recall** drops by 15% but the **F1-score** and the **accuracy** drop only by 5% which is not important as the **specificity increases** by 45% so the overall generalization of both classes is better.

8.3.4 Comparison with Other Study in Original Dataset

In this section, comparison takes place with another study [49] that uses traditional machine learning algorithms to make the classifications in the original dataset as described in Section 8.3.

The results are good with the standard machine learning algorithms too. The **accuracy** is a little under **70%,** the **F1-score** is at **67%,** the **recall** at **81%** and the **specificity** at **52%.** When Neural Networks are used the **F1-score** remains the **same** and **recall drops** by 5% as they are **70%** and **75%** respectively. The **accuracy** is around **60%** but the **specificity increases** to **85% from 52%.** So, a clear winner **cannot be determined** as there are some metrices where **traditional** machine learning achieves **higher scores** and others where **Neural Networks** achieve **higher scores**.

It seems that both types of algorithms can solve the same problem with similar effectiveness, but with different approaches as the algorithms learn, generalize the data differently. But it can be shown from the results that the Neural Networks can achieve better generalization in both classes, the avoidance, and the acceptance. As the specificity is around 30% higher this means that the traditional machine learning algorithms cannot predict acceptance people so easily while the NN can predict both classes with similar accuracy.

Chapter 9

Discussion

9.1 Summary

This thesis is the continuation of a series of experiments and analyses of emotional coping using psychophysiological features. The dataset is collected from an experiment that is conducted by the Department of Psychology of the University of Cyprus. Then, the dataset is multiplied with the Rectangular Window Methodology (RWM) and traditional machine learning algorithms, like random forest and decision tree, are used to classify people to avoidance and acceptance

The main objective of the current thesis is to substitute these algorithms with Neural Networks and to investigate if this kind of algorithms can solve these kinds of problems in a similar or with even better effectiveness.

Overall, the NNs that are used in this thesis are simple. The NNs are **MLP**, **RBF**, **SOM**, **LSTM** and some of its variations and a hybrid model , the **RBF-MLP-LSTM**.
The **different LSTM** models that are used are the LSTM, the **LSTM-MLP**, **RBF-LSTM** and a **simple RNN** but the **LSTM** produces the **best results** as its complexity differentiates it from the others and produces a lot better results.

The multiplication methodologies used on the dataset are the Rectangular Window Methodology (RWM), like in previous studies, and the Moving Window Methodology (MWM). In the MWM big and smaller slides are used. Also, in all the NN except the

LSTM, the tanh activation function is used instead of the relu producing greater results in the MWM.

The **best network** is the **RBF** in the dataset produced by the **MWM** with the **tanh activation function**. It seems that there may be a justification behind this as the **MWM** produces a **larger dataset**, that as known, is needed in Neural Networks. Then, the RBF is used to **filter**, normalize the data and then it is trained with the backpropagation method. After this, a comparison takes place with the previous study that uses standard machine learning, and it is noted that the **results** are **very similar**.

Finally, an **extension** of this study is made as these NNs are tested on datasets that originated from a real time experiment. There are some NNs that can achieve high results in all the metrices and achieve great generalization. Also, the NN's achieve similar or even slightly better results than the standard machine learning algorithms.

9.2 Replacement of Standard Machine Learning Algorithms

The main conclusion of this study is that there are some ways to replace the standard machine learning algorithms used with **Neural Networks**. The method introduced in this study is displayed in **Figure 9.1**.

First of all, another multiplication method must be used that can produce **larger** datasets such as the **Moving Window Methodology**. This leads to using more complex **activation functions** in the dataset that do not delete information and are more appropriate to the learning process of the Neural Networks, that is, they can be derived which is important for the learning of NN like MLP and RBF. This activation function is Tanh but there are probably more complex activation functions that can be used by this logic.

Then, to overcome any overfitting phenomena a data filtering method that normalizes the data can be used. This is actually used in a **lot of studies**. Most studies suggest that the *Fourier Transformation*, or *Short Time Fourier Transformation* can be used. But in this study a **data driven normalization**, that uses the dataset is used using the **RBF layer** to normalize the data. Finally, the Neural Network can be used to make classifications.

Overall, there are many different and complex NNs to be used such as CNN, but it seems that for simple binary classification and small datasets, the MultiLayer Perceptron can be extremely useful and efficient.

Another reason MLP is useful in this study is that its logic is similar to the *Gradient Boosting Decision Tree* which yielded great results in the previous study. Both algorithms use a loss function and try to minimize it in their own unique way. The MLP tries to find the best weights in the NN to minimize the loss and the Gradient Boosting Decision Tree adds weak learners in every iteration to reduce and minimize the loss.

Consequently, the minimization of the loss function seems to be a great way to make good classifications. So, maybe in the future more Neural Networks can be used that target in minimizing their loss function.

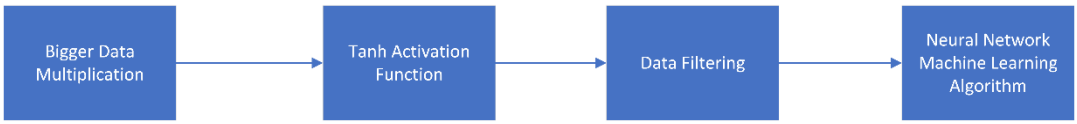

Figure 9.1 Replace of Traditional Machine Learning with Neural Networks Methodology

9.3 Future

Overall, the Neural Networks yielded **very good** results in both studies, conducted by the Department of Psychology in University of Cyprus, similar to the standard machine learning algorithms' results.

This is an indicator that even the basic, less complex NNs can **perform great** in classification problem with datasets from **Psychophysiological measures.**

Also, the *replacement methodology* that is described **Figure 9.1** can probably be used in a lot of different datasets, **problems** to **replace** the **traditional** machine learning algorithms.

Furthermore, in this dataset if more complex methods and Neural Networks such as Fourier Transformation and CNN respectively can be used, perhaps better results can be obtained.

So, the continuation of this thesis can be a study where the steps of **Figure 9.1** are followed but in a more *complex way*. For example, **other data multiplication methodologies** can be used and other filtering techniques such as Fourier Transformation, or other layers of neurons instead of the RBF, and **more complex** Neural Networks to make the predictions that probably still use the **minimization of the loss function logic**

as it produced the **best results** in this study. Also, **other activation functions** can be used that still can be **derived** but they may be more complex and represent the data in an even better way.

Finally, Neural Networks are often referred as *online algorithms,* which means that they can be deployed in the real world and while they are used to make classification, they collect data, and they keep on learning and becoming even better over time. In this way, NNs can be combined with the final purpose of all these studies and project to have a device which is going to be used at all time by people, and when they use the **avoidance** coping mechanism **some advice** will pop up to help them **deal** with their **stress** and **feelings**. With the use of Neural Networks these algorithms will keep on training with new, unseen data and becoming even better with time.

References

[1] Artificial Neural Network Modelling: An Introduction by Subana Shanmuganathan, 2016 Available : https://link.springer.com/chapter/10.1007/978-3-319-28495-8_1

[2] State-of-the-art in artificial neural network applications: A survey by Oludare Isaac Abiodun , Aman Jantan , Abiodun Esther Omolara , Kemi Victoria Dada , Nachaat AbdElatif Mohamed, Humaira Arshad, 2018
Available : https://www.sciencedirect.com/science/article/pii/S2405844018332067

[3] An Overview of Neural Network by Mohaiminul Islam, Guorong Chen, Shangzhu Jin, 2019

[4] An introduction to Neural Networks by Ben Krose, Patrick van der Smagt
Available:
http://14.99.188.242:8080/jspui/bitstream/123456789/1991/1/An%20Introduction%20to%20Neural%20Networks.pdf

[5] Deep learning by Yann LeCun, Yoshua Bengio & Geoffrey Hinton, 2015
Available : https://www.nature.com/articles/nature14539

[6] Neural network models and deep learning by Nikolaus Kriegeskorte and Tal Golan, 2019
Available : https://www.sciencedirect.com/science/article/pii/S0960982219302040

[7] A survey of deep neural network architectures and their applications by Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, Fuad E. Alsaadi, 2017
Available : https://www.sciencedirect.com/science/article/pii/S0925231216315533

[8] A Neural Network Approach to Time Series Forecasting, 2009
Available : https://www.researchgate.net/profile/Leslie-Smith-23/publication/44260191_A_Neural_Network_Approach_to_Time_Series_Forecasting/links/0fcfd508ffc793c329000000/A-Neural-Network-Approach-to-Time-Series-Forecasting.pdf

[9] A dynamic artificial neural network model for forecasting time series events by M. Ghiassi, H. Saidane, D. K. Zimbra, 2005
Available : https://www.sciencedirect.com/science/article/pii/S0169207004001116

[10] Prediction of Chaotic Time Series Based on the Recurrent Predictor Neural Network by Min Han, Member, IEEE, Jianhui Xi, Shiguo Xu, and Fu-Liang Yin, 2004
Available : https://ieeexplore.ieee.org/abstract/document/1356236

[11] Emotion recognition from EEG-based relative power spectral topography using convolutional neural network by Md. Asadur Rahman, Anika Anjum, Md. Mahmudul Haque Milu, Farzana Khanam, Mohammad Shorif Uddin, Md. Nurunnabi Mollah, 2021
Available : https://www.sciencedirect.com/science/article/pii/S2590005621000205

[12] Emotion recognition with convolutional neural network and EEG-based EFDMs by Fei Wang, Shichao Wu, Weiwei Zhang, Zongfeng Xu, Yahui Zhang, Chengdong Wu,Sonya Coleman, 2020
Available : https://www.sciencedirect.com/science/article/pii/S0028393220301780

[13] Deep fusion of multi-channel neurophysiological signal for emotion recognition and monitoring by Li, Xiang; Song, Dawei; Zhang, Peng; Hou, Yuexian and Hu, Bin, 2017
Available : http://oro.open.ac.uk/52877/

[14] Using Deep Convolutional Neural Network for Emotion Detection on a Physiological Signals Dataset (AMIGOS) by LUZ SANTAMARIA-GRANADOS, MARIO MUNOZ ORGANERO , (Member, IEEE), GUSTAVO RAMIREZ-GONZÁLEZ , ENAS ABDULHAY, N. ARUNKUMAR, 2018
Available : https://ieeexplore.ieee.org/abstract/document/8543567

[15] Exploration of physiological sensors, features, and machine learning models for pain
intensity estimation by Fatemeh Pouromran, Srinivasan Radhakrishnan, Sagar Kamarthi, 2021
Available : https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0254108

[16] Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement by Jing Li, Ji-hang Cheng, Jing-yuan Shi, and Fei Huang
Available : https://link.springer.com/chapter/10.1007/978-3-642-30223-7_87

[17] Learning Precise Timing with LSTM Recurrent Networks
Available : https://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf

[18] What is LSTM? Introduction to Long Short Term Memory
Available : https://intellipaat.com/blog/what-is-lstm/?US

[19] C. Galazis, "Non-Intrusive Physiological Wearable Devices for Identifying Individual Difference Parameters Using Supervised Classification Learning Algorithms," Computer Science Department, Univeristy of Cyprus, Nicosia, Cyprus, 2017.

[20] A. Trigiorgi, "Μελέτη Μεθόδων Μηχανικής και Βαθιάς Μάθησης και Εφαρμογή σε Ψυχομετρικά Δεδομένα," Diploma Project, Department of Computer Science, University of Cyprus, 2016.

[21] G. Demosthenous, "Machine Learning Approach to Predict Emotional Coping Using Psychophysiological Signals," MSc thesis, Department of Computer Science, University of Cyprus, Nicosia, 2019.

[22] E. Georgiou , " Feature Selection and Training of Machine Learning Algorithms to Classify Functional Versus Dysfuncional Coping with Acute Pain ," Diploma Project, Department of Computer Science, University of Cyprus, 2022.

[23] C. E. Ackerman, "How Does Acceptance And Commitment Therapy (ACT) Work?," 29 March 2022. [Online]. Available: https://positivepsychology.com/act-acceptance-and-commitment-therapy/. [Accessed 23 May 2022].

[24] D. C. Swanson, Signal Processing for Intelligent Sensor Systems with MATLAB, Boca Raton: CRC Press, 2011

[25] Y. Sattar and L. Chhabra, in Electrocardiogram, StatPearls Publishing, 2021.

[26] E. Ashley and J. Niebauer, in Cardiology Explained, London, Remedica, 2004.

[27] B. Farnsworth, "What Is ECG and How Does It Work?," August 3 2021. Available: https://imotions.com/blog/what-is-ecg/

[28] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur and H. Nazeran, "A review on wearable photoplethysmography sensors and their potential future applications in health care," Int J Biosens Bioelectron, vol. 4, no. 4, pp. 195-202, 2018.

[29] A. Alqaraawi, A. Alwosheel and A. Alasaad, "Heart rate variability estimation in photoplethysmography signals using Bayesian learning approach," Healthc Technol Lett., vol. 3, no. 2, pp. 136-142, 2016.

[30] M. Benedek and C. Kaernbach, "A continuous measure of phasic electrodermal activity," Journal of Neuroscience Methods, vol. 190, no. 1, pp. 80-91, 2010.

[31] "Galvanic Skin Response (GSR): The Complete Pocket Guide," iMotions, 2020.

[32] J. Wilson, "What Is Facial EMG and How Does It Work?," iMotions, 2018.

[33] T. Marur, Y. Tuna and S. Demirci, "Facial anatomy," Clinics in Dermatology, vol. 32, no. 1, pp. 14-23, 2014.

[34] J. Tong, M. J. Lopez and B. C. Patel, "Anatomy, Head and Neck, Eye Orbicularis Oculi Muscle," StatPearls Publishing, Treasure Island (FL), 2021.

[35] J. T. Cacioppo, R. E. Petty, M. E. Losch and H. S. Kim, "Electromyographic Activity Over Facial Muscle Regions Can Differentiate the Valence and Intensity of Affective Reactions," Journal of Personality and Social Psychology, vol. 50, no. 2, pp. 260-268, 1986.

[36] Ά. Τριγιώργη, «Εξόρυξη Γνώσης από Ψυχοφυσιολογικά Δεδομένα και Συγκριτική Αξιολόγηση Αλγορίθμων Μηχανικής Μάθησης,» 2018.

[37] sunil, "Quick Introduction to Boosting Algorithms in Machine Learning," 9 November 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/. [Accessed 16 April 2022].

[38] V. Kurama, "Gradient Boosting In Classification: Not a Black Box Anymore!," 2020.
Available: https://blog.paperspace.com/gradient-boosting-for-classification/

[39] J. Brownlee, "A Gentle Introduction to Ensemble Learning Algorithms," 19 April 2021.
Available: https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

[40] S. E. R, "Understanding Random Forest," 17 June 2021
Available:https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[41] P. Aznar, "What is the difference between Extra Trees and Random Forest?," 17 June 2020.
Available:https://quantdare.com/what-is-the-difference-between-extra-trees-and-random-forest/#:~:text=Random%20Forest%20chooses%20the%20optimum

[42] H. Liu, M. Zhou and Q. Liu, "An Embedded Feature Selection Method for," *IEEE/CAA JOURNAL OF AUTOMATICA SINICA*, vol. 6, no. 3, pp. 703-715, 2019.

[43] Available: https://www.biopac.com/wp-content/uploads/MP150-Systems.pdf

[44] "Hardware Specifications BIOPAC MP150,"
Available: https://imotions.com/hardware/biopac-mp150/

[45] "ACQKNOWLEDGE 3.9 SOFTWARE GUIDE,"
Available: https://www.biopac.com/manual/acqknowledge-3-9-software-guide/

[46] D. S. McConnell, N. S. Chudy and J. Smither, "The Performance of Microsoft Band 2: A Validity and Reliability Study," in Human Factors and Applied Psychology Student Conference, Daytona, 2016.

[47] Available: https://moodmetric.com/. [Accessed 22 May 2022].

[48] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," 23 May 2018. Available: https://machinelearningmastery.com/k-fold-cross-validation/

[49] S. Zenios "Analysis of real-time E4 psychophysiological data for machine-learning-based classification and prediction ",s Diploma Project, Department of Computer Science, University of Cyprus, 2023.

[50] Radial Basis Function Network (RBFN) Tutorial
Available : https://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/

[51] P. Konstantinou, A. Trigeorgi, C. Georgiou, A. T. Gloster, G. Panayiotou and M. Karekla, "Functional versus dysfunctional coping with acute pain: An experimental comparison of acceptance vs. avoidance coping".

[52] W. Pitts, W.S. McCulloch, How we know univesrsals, The perception of auditory and visualforms. Bull. Math. Biophys. 9(3), 127–147 (1947)

[53] Activation Functions in Neural Networks
Available : https://www.v7labs.com/blog/neural-networks-activation-functions

[54] E4 wristband, the legacy industry-standard for real-world, and academic research.

[55] Empatica. Empatica e4 specifications. [Online]. Accessed in 2022.

[56] E4 wristband.

[57] E4 data.