DEGREE PROJECT IN ELEKTROTEKNIK,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Development of an Intelligent Embedded Interface for Interpreting Biosignals Recorded by Novel Wearable Devices

DIEGO ARGÜELLO RON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

Development of an Intelligent Embedded Interface for Interpreting Biosignals Recorded by Novel Wearable Devices

DIEGO ARGÜELLO RON

Degree Programme in Electrical Engineering
Date: May 14, 2019
Supervisor: Pawel Herman
Examiner: Erik Fransén
KTH School of Electrical Engineering and Computer Science

Abstract

In recent years there has been a considerable development in the realm of sensing technologies, embedded systems, wireless communication technologies, nanotechnologies, and miniaturization has made it possible to create smart wearable systems that can record data from the human bodies and monitor our daily activities. Most expectations for the successful deployment of wearable devices and their tangible impact for the society is in healthcare. Nevertheless, its use has been limited by the absence of intelligent mobile device interfaces able to process, analyse and inference the recorded data, giving relevant information to the user. On the other hand, new advances in nanotechnology have allowed the creation of so-called electronic skin, which consists in thin and flexible electrodes, easy and comfortable to use. This allows building new wearable devices able to record electrical activity from the surface of the body, which has a large diagnostic and monitoring potential.

In this work, the goal is to study the feasibility of using these new sensors for continuous biopotential recording while supporting them with a mobile phone application able to receive, process and analyse the recorded biosignals in order to deliver useful feedback to the user in real-time. The wearable device known as Senso Medical Bio Pot V2 is proposed as a possible candidate to carry out electromyography (EMG), electrocardiography (ECG) and electroencephalography (EEG) recordings via skin tattoo electrodes. Moreover, an Android Application that connects to this device is created. It uses Machine Learning Algorithms embedded on it in order to classify the received signals. Finally, Long-Short Term Memory (LSTM) networks are implemented for classifying EEG and EMG signals.

Several conclusions are derived from this work. Firstly, the device Senso Medical Bio Pot V2 is not suitable for its use as a wearable device for biosignal recording. Secondly, the Application designed and simulated offline achieves good performance. As a consequence, it could be used in the future with a suitable wearable sensor and offer good potential for processing and interpreting recorded biosignals with an opportunity to provide real-time feedback to the user in Brain-Computer Interface (BCI) type of applications.

Sammanfattning

Under de senaste åren har det varit en betydande utveckling inom kännetecknande teknik, inbyggda system, trådlös kommunikationsteknik, nanoteknik och miniatyrisering gjort det möjligt att skapa smarta bärbara system som kan spela in data från människokroppen och övervaka våra dagliga aktiviteter. Bärbar teknik är viktig på grund av dess inverkan på sjukvård. Ändå har användningen begränsats av avsaknaden av intelligenta gränssnitt för mobila enheter som kan bearbeta, analysera och avleda de inspelade data, vilket ger användaren viktig information. Å andra sidan har nya framsteg inom nanotekniken gjort det möjligt att skapa så kallad elektronisk hud, som består av tunna och flexibla elektroder, lätt och bekväm att använda. Detta gör det möjligt att bygga nya bärbara enheter som kan registrera elektrisk aktivitet från kroppens yta, vilket har en stor diagnostisk och övervaknings potential.

I detta arbete är målet att studera möjligheten att använda dessa nya sensorer för kontinuerlig biopotentiell inspelning samtidigt som de stöder dem med en mobilapplikation som kan ta emot, bearbeta och analysera de inspelade biosignalerna för att ge användbar feedback till användaren i realtid. Den bärbara enheten Senso Medical Bio Pot V2 föreslås som en möjlig kandidat för att genomföra elektromyografi (EMG), elektrokardiografi (EKG) och elektroencefalografi (EEG) inspelningar via elektroniska tatueringar. Dessutom skapas en Android-applikation som ansluter till den här enheten. Den använder maskinlärningsalgoritmer inbäddade på den för att klassificera mottagna signaler. Slutligen implementeras Långt Korttidsminne Nätverk för att klassificera EEG och EMG signaler.

Flera slutsatser härrör från detta arbete. För det första är enheten Senso Medical Bio Pot V2 inte lämplig för användningen som en bärbar anordning för biosignalinspelning. För det andra åstadkommer applikationen designad och simulerad offline, god prestanda. Till följd av detta kan den användas i framtiden med en lämplig bärbar sensor och erbjuda god potential för bearbetning och tolkning av inspelade biosignaler med möjlighet att ge realtid återkoppling till användaren i hjärndatorgränssnittet typ av applikationer.

Acknowledgments

First of all, I would like to thank my supervisor at KTH, Pawel Herman, for the opportunity of carrying out this project as well for the discussions and advice throughout its development.

I would also like to thank ETSII-UPM the opportunity to come to Stockholm to course a Double Degree Program in KTH. Moreover, I would like to thank the teachers in ETSII-UPM and KTH all they have taught me during these years.

Last, but not least, I would like to thank my family for their support during my studies.

Contents

CONTENTS

Chapter 1

Introduction

During the last years a considerable development in the realm of sensing technologies, embedded systems, wireless communication technologies, nano technologies, and miniaturization, has made it possible to create smart wearable systems that can record data from the user´s bodies and monitor their daily activities [1, 2, 3, 4].

Several sectors can benefit from wearable devices technology, as it is the case for human activity recognition, sports performance, smart cities and education, among others [1, 5]. Thus, nowadays the use of wearable electronics devices is growing fast, with fitness devices as one of the most popular applications [4]. Smart watches are a good example of these devices as they can measure parameters such as the user´s heart rate, steps taken or distance travelled [6]. Lately, smart glasses have also appeared to enhance the experience of the user in the real world [7]. Finally, Electroencephalography (EEG) headsets are beginning to be used as a help in meditation or for gaming [8, 9].

Nevertheless, all these promising opportunities come with several technological challenges. Some of them are the optimization of the embedded software, wireless communication, low energy consumption, ergonomics and data storage capacity. Moreover, in order to keep the use of wearable devices growing, its price must decrease at the same time that the consumer finds them actually useful. Thus, interfaces for mobile devices, that deploy data mining and processing algorithms, must be created in order to give feedback triggered by real-time sensor data [1, 4].

Most expectations for the successful deployment of wearable devices and their tangible impact for the society is in healthcare[10]. For example, nowadays, it is already possible to monitor patients over an extended period of time, which can help clinicians to get relevant information about them and thus define strategies in rehabilitation and in treatment [10]. Nevertheless, its use is being limited by the already mentioned absence of intelligent mobile device interfaces able to process, analyse and inference the recorded data, giving relevant information to the user [4]. Moreover, until now wearable fitness and bio medical devices have only been aimed to monitor vital signs such as heart rate, respiration or skin temperature [4]. Thus, approaches as for example skin-contact electrophysiology, where electrical activity is recorded from the surface of the body, have received little attention [4, 10]. This approach can provide important information about the health condition, making use of EMG, ECG or EEG. As a consequence, it has a large diagnostic and monitoring potential. Nevertheless, the electrodes placed on the skin in order to measure electric signals have been bulky and thus difficult and uncomfortable to use. However, new advances in nanotechnology have allowed the creation of so-called electronic skin, which consists in thin and flexible electrodes, easy and comfortable to use [4, 11, 12]. This has paved the

way for new wearable devices able to measure biopotentials in a non-clinical environment, which can be used for new portable Brain-Computer Interfaces, speech recognition, recording of emotions or rehabilitation [11, 12].
Therefore, it is worth studying the possibility of using these new sensors for continuous biopotential recording, while supporting them with a mobile phone application able to receive, process and analyse the recorded biosignals, in order to deliver useful feedback to the user in real-time.

1.1 SCOPE AND OBJECTIVES

This project has three main objectives:

- Firstly, validate the feasibility of using electronic skin type sensors as wearable devices for real-time activity monitoring.
- Secondly, develop a prototype application deployable on a mobile device, in order to process and analyse the biosignals recorded by these wearable devices, so that a real-time feedback can be given to the user.
- Thirdly, test the scalability of this prototype by training the embedded machine learning algorithms used to analyse the biosignals with EEG and EMG datasets.

1.2 OUTLINE OF THE THESIS

The rest of the Thesis is organised as follows:

- In Chapter 2, the scientific and technical background is outlined. The development of wearable devices used for recording biopotentials is explained, followed by how smartphone and mobile software technology has evolved. Finally, the most relevant machine learning algorithms for the project are discussed.
- In Chapter 3, the methods used in the project are explained, focusing on implementing wireless communication and machine learning algorithms in mobile phone applications, as well as the validation of wearable sensors for real-time use.
- In Chapter 4, the results obtained in the project are presented.
- In Chapter 5 the results are discussed, pointing out their most relevant implications.
- In Chapter 6 key directions for future work are proposed.

Chapter 2

Background

New sensors that can be implanted in the body or in clothes can have a huge relevance in medicine and rehabilitation [1, 13]. Using these sensors to record data for a long period of time can yield important information for the design of clinical strategies when treating different diseases. On the other hand, this will increase the accuracy of the clinical assessment, as well as improve the situation of the patience, granting them for example more autonomy [10].

The critical steps when making this feasible are:

1. Designing of sensors that have a good recording capability and that are comfortable to wear [1, 10, 11, 13].
2. Development of a system able to receive, process and analyse the recorded data in order to deliver useful feedback to the user in real-time [1, 4].

Wearable devices have already been successfully applied in medicine. For example, they have been used to measure the movement of the muscles making use of accelerometers and EMG sensors. This has been employed to treat patients with motor disorders like Parkinson [11] and to help in the rehabilitation of people with amputated limbs [14]. Other examples of the use of wearable devices in medicine are, the control of the levels of psychological stress and help in motor restoration [15].

2.1 WEARABLE DEVICES FOR RECORDING BIOPOTENTIALS

Nowadays, a large range of wearable devices are for sale and many are oriented to provide long-term health care and well-being services. One of the main reasons is the growth of elderly population that is expected to continue during the next decades [13]. Some examples of wearable devices that can be found for sale are: smartwatches, fitness bands, smartglasses or EEG headsets [16].

In the case of a smartwatch, it allows to access applications that until now were relegated to smartphones. Some examples are email, messaging, calendar, social networking, or telephony [17].

The fitness bands are oriented to record data like: heart rate, calories burned, distance travelled, etc. The device connects to a smartphone where all this information is processed and the user receives feedback based on it [18].

On the other hand, smart glasses takes a step forward towards wearable computing. Thus, making use of virtual reality (creation of a virtual world where the users can immerse into), augmented reality (virtual content is added to the user´s real world perception) and diminished reality (objects are subtracted from reality), it tries to enhance the experience of the user in the real world, and as a consequence it can help to increase the productivity in fields like medicine, education, documentation, etc [19].

Finally, EEG devices like the Emotiv EPOC and the Neurosky MindWave, are used as a help during the meditation practice [19].

Nevertheless, among all the signals that can be measured it is worth noticing the biopotentials, especially because of their importance in the medical treatment of illnesses. These biopotentials can be recorded in the form of ECG, EEG, electrooculograms (EOG) and EMG. They provide highly useful physiological information, which can be measured in a non-invasive and inexpensive way thanks to novel dry electrodes, known as "electronic skin" [20]. In order to fabricate this new type of electrodes, the scheme presented in Figure 2.1 is followed.

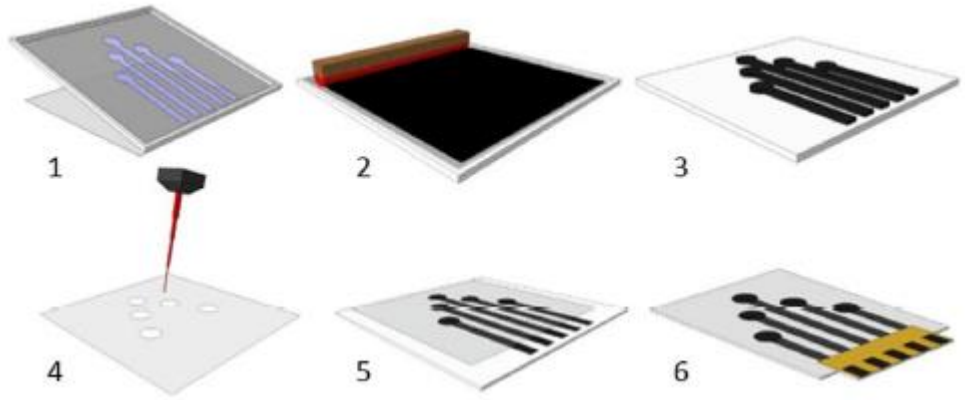

Figure 2.1: Steps followed during the fabrication process of electronic-skin type electrodes. Steps 1 and 2 correspond to screen-printing onto a temporary tattoo sheet. Step 3 displays an electrode array after printing. In step 4 a double-sided adhesive film is cut with a laser to form the passivation layer. Step 5 describes how the passivation layer and the electrode array are aligned and bonded. Finally, in 6, a polyimide film with holes is glued to the array to fit into a zero insertion force (ZIF) socket [20].

In the fabrication process the technic known as screen printing is used. It consists in printing the conducting traces using inks of Ag/Au/C. For example, Ag can be employed just to create the pattern and then the C ink is applied in order to be in contact with the skin. After this, laser cutting is used to make holes in a double-sided passivation layer, which also serves as a skin-adhesive material. The fact that the printing process is carried out on thin and soft materials, makes it possible to achieve comfortable contact and a good recording stability [20, 21].

Some of the main properties of this "electronic skin" are [21]:

- Long term use (several hours).
- Screen-printing technology is a simple way of creating patterns with different types of inks.
- High flexibility.
- Low electrode-skin impedance, resulting in a high signal to noise ratio and resolution.
- Easy to use, due to the fact that no application of a gel or any other preparation is required.

2.1.1 WIRELESS TECHNOLOGY IN WEARABLE DEVICES

Wearable devices need to be ergonomic, and have an acceptable autonomy in order to be used for an extended period of time. Advances in microelectronics have given an answer to this problem. They have made it possible to reduce the size and cost of electronic devices. Moreover, there is no longer need for cables when transmitting information between devices thanks to the apparition of cheap and small integrated radios [22, 23]. This radio technology is known as Bluetooth, and is a suitable protocol for wirelessly communication when small amounts of data are transmitted in a short range [22]. A detail description of how this technology works can be found in Appendix B.

On the other hand, a new system known as Bluetooth Low Energy (BLE), has appeared to give answer to those situations in which Bluetooth characteristics are required, but low-energy consumption is also necessary. This makes BLE ideal for wearable devices.

The amount of information transmitted is lower than in the case of the Bluetooth, but the life-time of the battery is widely increased [24].

The protocol stack of the BLE consists of a Controller, a Host and an application or non-core profiles, as it can be seen in Figure 2.3 [22, 24].

The Controller consists of a Physical Layer and a Link Layer. The Physical Layer is a System on Chip (SOC) with a Radio integrated on it. The Link Layer is the interface between the Controller and the Host. Here, all the tasks related to advertising and bidirectional communication take place. Thus, in the Link Layer it is determined if a device acts as master or as a slave [24].

The Host consists of a series of protocols. At the base, it is the Logical Link Control and Application Protocol (L2CAP), which multiplexes the signals received in the lower layers and send them to the upper one, or fragment those that comes from the upper layers and send them to the lower ones. After the L2CAP, other two protocols can be found: Security Manager Protocol (SMP) and the Attribute Protocol (ATT). The last one is in charge of defining how the communication between two devices takes place. Thus, one of the devices is set as server and the other as client. The server has a set of attributes, which are structures where data is stored. The client questions the server about its attributes in the service discovery process, and after carrying out a request, it is allowed to read and write the attributes of the server. At the same time, the server sends back updates and user data [24].

The Generic Attribute Profile (GATT) protocol operates on top of the ATT, and handles the attributes of the server. It also determines which device is the server and which one the client, independently of the master and client roles. As it has been explained above, the client accesses the attributes of the server by sending requests, which trigger response messages from the server [24]. Each attribute has a Universally Unique Identifier (UUID), which is a 128-bit (16 bytes) number that is used for identification. The attributes in a GATT server are divided in services, that have zero or more characteristics, which include zero or more descriptors (Figure 2.2) [25, 26].

In order to understand what these terms represent, a heart rate monitor that implements BLE technology can be used as example. Thus, this device could have a service called "Heart Rate Monitor", in which there are characteristics as for example "heart rate measurement". This characteristic would have a single value and zero or more descriptors, which describe the value of the characteristic. Moreover, the device could have another service such as "Manufacturer Information" with its respective characteristics and descriptors [25].

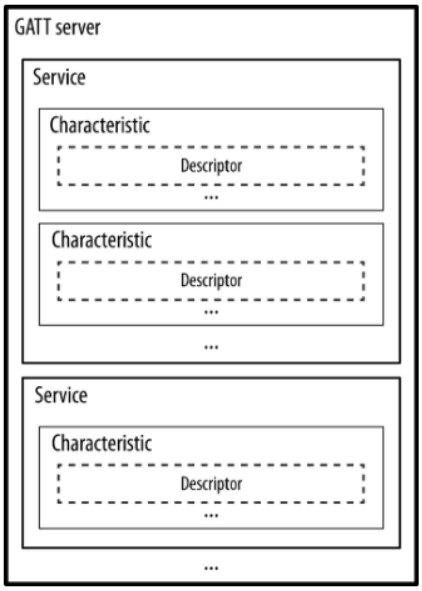

Figure 2.2: GATT hierarchy [23].

The protocol that is at the top of the BLE stack is the Generic Access Profile (GAP). It is in charge of specifying the devices roles, how the discovery of devices and services is going to be carried out, and how the connection is going to be established. The controller can act as a broadcaster, observer, peripheral or central, being each role chosen by the GAP depending on the situation. Thus, a broadcaster broadcasts data using the advertising channels but can not connect with other devices. The broadcasted data is received by an observer. A device that acts as central initiates and manages several connections. On the other hand, a peripheral has only one connection with a device in the central role. Therefore, these two roles determine the master and slave roles, respectively. A device can change its role but it can not have more than one at the same time [24, 26].

Finally, the application is the highest layer where data handling activities take place. It is materialized in a user interface. On the other hand, it also it possible that devices from different manufacturers work together [22, 24].

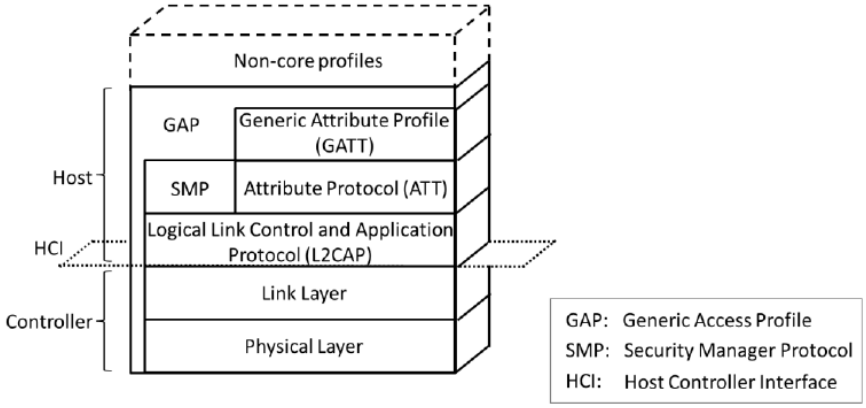

Figure 2.3: Bluetooth Low Energy Protocol Stack [24].

2.1.2 TECHNOLOGY FOR WEARABLE DEVICES WITH FOCUS ON SENSO MEDICAL BIO POT V2

In this Project, the platform Senso Medical Bio Pot V2 is used. This device is aimed to measure EEG, EMG and ECG biopotentials and has a total of 8 channels. It uses BLE 4.2 and 5.0 protocols to stablish a connection with the host. Thus, the device can take measurements in a continuous and prolonged way. On the other hand the number of measurements varies from one operating system to other, being of 500 samples per second in Android and up to 2000 samples per second in Windows. Moreover, the device has an on board memory buffer that avoids to lose data as a consequence of RF blind spots [27].

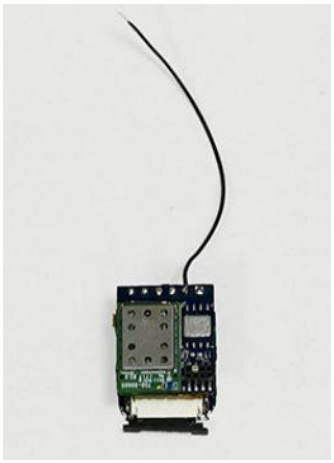

Figure 2.4: Senso Medical Bio Pot V2 [24].

The device has five characteristics [28]:

- Characteristic 1: Read/Write to INTAN register.
- Characteristic 2: Start/Stop acquisition.
- Characteristic 3: Enable/Disable impedance.
- Characteristic 4: Data Notifications.
- Characteristic 5: Configure parameters.

A more detailed description of these characteristics can be found in Appendix C.

Finally, the electrodes used by this device are the known as “electronic skin”, which have already been described in this Chapter. An example of this type of electrodes can be seen in Figure 2.5. Thus, just by removing the protecting film, the electrode can be attached to the skin of the user, and connecting its terminals with the recording device the biopotential signal can be measured.

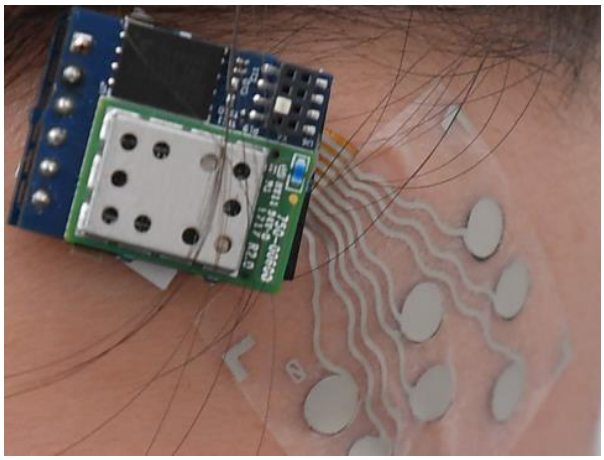

Figure 2.5: Senso Medical Bio Pot V2 connected to an electrode [27].

2.2 INTRODUCTION TO SMARTPHONE TECHNOLOGY

The first concept of mobile phone dates back to the 1970s, with the first commercial model being released in 1979. The first models were bulky and slow. Nevertheless, they opened the door to integrated devices [3, 29]

In 2011, it was calculated that around the 80% of the world population had a mobile phone, which makes an idea of the impact that this kind of devices has had in our society [3]. Nowadays, mobile phones have become accessible for everyone and a new era has begun with the apparition of the smartphones, which in addition to combining telephony and computing, have sensors integrated and a much larger computation power. It is worth noticing that this has been achieved reducing the size of the devices, although it is gradually becoming a challenge [3, 29].

Thanks to smartphones, Internet applications can be used by anyone whenever and wherever they want. On the other hand, now it is possible to have high speed data networks as for example 4G [29], being expected 5G to arrive by 2020 [30]. Moreover, the interfaces are

easy to use. The user is able to navigate through it thanks to capacitive multitouch screens, carrying out complex tasks that some time ago where only done in a computer [29].

There are six dimensions in which the smartphones are already evolving and will continue evolving in the future, changing the paradigm of integrated devices: Personal Computing, Internet of Things, Multimedia Delivery, Context Awareness and Low Power Consumption [29]. For this project, all of these fields are important because they will influence in making the exploitation of the data generated by wearable devices a reality.

The last trend when increasing the computation power of the new smartphones is the use of multicore processors (e.g. quad-core, octa-core). The development of CMOS transistors, system-on-chip designs, and memories with higher capacities, will allow new smartphones to carry out costly computational tasks, such as personal data mining [29].

On the other hand, the Internet of Things (IoT) is becoming a reality. It is essential for wearable devices to use the capabilities of smartphones and vice versa. Thus, new technologies like the Bluetooth Low Energy are being developed in order to allow these devices to connect with each other so that the user can benefit from all the data generated by the sensors around [29].

One of the most critical aspects in the performance of a smartphone is the duration of its battery. In order to make it possible for smartphones to meet future expectations, new technologies must be developed in order to have batteries with higher capacities. For example, CMOS semiconductor technology can increase the duration of the battery by decreasing the leakage current of the components. Finally, efficient software and algorithms are essential to save energy by making use of less components as well as turning off those that are not being used [29].

One day, flexible electronics will make it possible to have smartphones that are wearable. For example, the smartphones would pass from being a pocket device to part of our clothes, making use of our own movements to collect the energy necessary to work. They could even be bio-electronic devices and become part of our bodies in order to enhance our own physiological and mental capabilities [29, 31].

In Figure 2.6 it can be seen that some years ago most of the devices used Android as operating system. iOS was the second most used one [29]. Now, the situation is even more polarized as the operating systems Windows Phone and Blackberry OS have disappeared. As a consequence, Android and iOS dominate the global market in the sector.

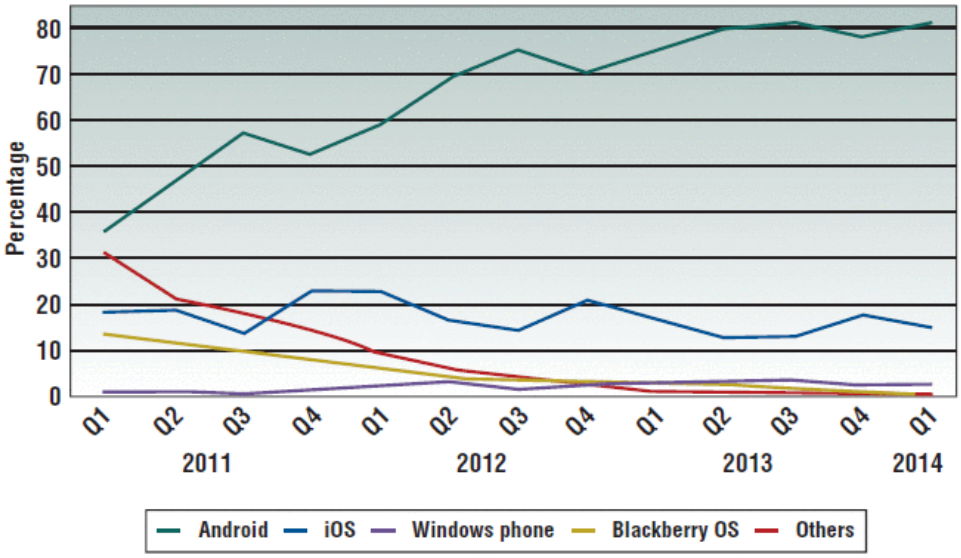

Figure 2.6: Software distribution in the smartphone market [29].

It is worth noticing Android, because of its interest in joining Artificial Intelligence and mobile applications. Thus, Android 8.1 and 9.0 has a new Android Neural Networks API, which makes it possible to carry out costly machine learning operations [32].

2.3 MOBILE SOFTWARE DEVELOPMENT FOR WEARABLE DEVICES

2.3.1 ANDROID SOFTWARE

When building an embedded interface for data mining, there are several mobile operating systems that can be used. As it has been pointed out above, Android is the most popular one nowadays.

Android is a software for smartphones and other devices created by Google. It includes an operating system, middleware and key applications [29, 33].

Its popularity can be explained thanks to its two main advantages: being open source and having Linux kernel-based architecture model. Thus, its characteristics are constantly being improved thanks to developers around the World. Moreover, as Google came to an agreement with several phone services and mobile accessories providers, Android gives the consumer choices that for example iOS does not. Most of the prestigious Smartphones manufacturers, as for example Samsung and LG, have had Android phone offering since it was launched, and now models with Android as operating system have been generalized. On the other hand, manufacturers of mobile accessories as for example headsets can use Android for software development. Finally, having a Linux kernel-based architecture model makes it possible to use the characteristics offered by Linux [33, 34, 35].

A detail explanation of the Android software architecture can be found in Appendix D.

2.3.2 DEVELOPMENT OF AN ANDROID APPLICATION

The basic tool when building an Android application is the Android SDK, which compiles the code, data and resource files, putting the result in an Android package and .apk file. This file will be the one that is installed on the device [35].

When creating an Android application there are several Integrated Development Environments (IDE). Nevertheless, the one created by Google, Android Studio, is the most used nowadays.

The basic structure of an Android application can be seen in Figure 2.7. There are two main sections: the app folder and the Gradle Scripts. The first one contains: the manifest folder, with the file AndroidManifest.xml, in which components and permissions required by the applications are declared [35]; the java folder with all the java code corresponding to the activities and other components of the application; the asset folder that contains the extra data as for example the Machine Learning files with the algorithms used by the app (compiled in the .apk file); the res folder with the activities layouts (.xml files), strings for the User Interface (UI) and bitmaps images.

Finally, the Gradle Scripts contain the files resulted from the compilation.

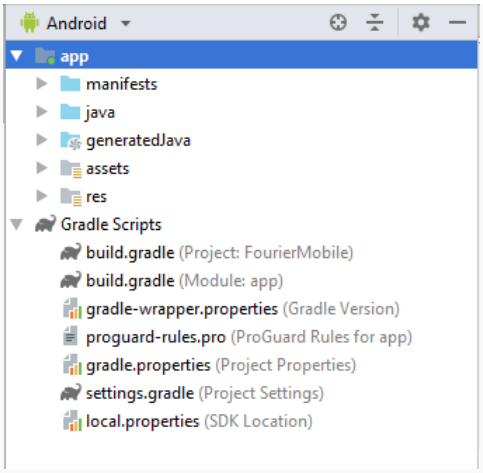

Figure 2.7: General Structure in an Android Application.

Another important step when building an Android application is choosing the level of the API (framework version). This version determines its functionalities as well as the number of devices that are able to use it.

Since the release of the Android operating system on 5th November 2007 [33], there have been a total of 28 APIs. The last API level corresponds to Android 9.0. In this Project, Android 4.3 (API 18) and Android 8.1 (API 27) are the most relevant ones. Android 4.3 is the first version of the software that introduced a module in order to connect the smartphone with BLE devices [25]. After this, all the Android versions have had a module to implement BLE connections, which has suffered constants upgrades. Finally, with the release of Android 8.1 a huge step towards the implementation of Artificial intelligence in smartphones has been taken. Thus, this API contains a Neural Networks API that allows

accelerated computation and inference for on-device machine learning frameworks like TensorFlow Lite and Caffe2 among others [32, 36]. Its architecture can be seen in Figure 2.8.

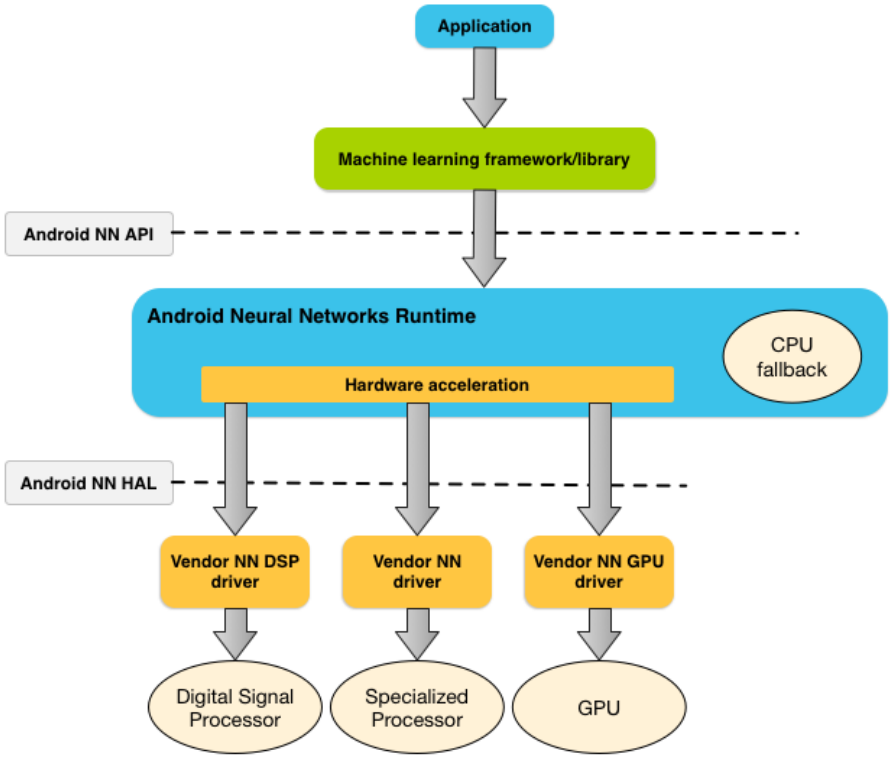

Figure 2.8: Architecture of the Neural Network API [36].

On the other hand, an Android Application always has some basic elements [35]:

- View.
- Viewgroup.
- Layout.
- Activity.

When an Application grows in complexity more elements appear. It is worth noticing [35]:

- Service.
- Intent.
- Broadcast receiver.
- Content provider.

A layout determines how the views are distributed in an activity inside an application. There are four types of layout: LinearLayout, CoordinatorLayout, RelativeLayout, ConstraintLayout, FrameLayout. The layout of an application is defined making use of an .xml file [35].

A view is a component of the User Interface (UI) of an activity. It is an element of the class View. A hierarchy of this class can be seen in Figure 2.9. The views are defined in the .xml file of the layout. A view has different attributes: position, id, style, etc [35].

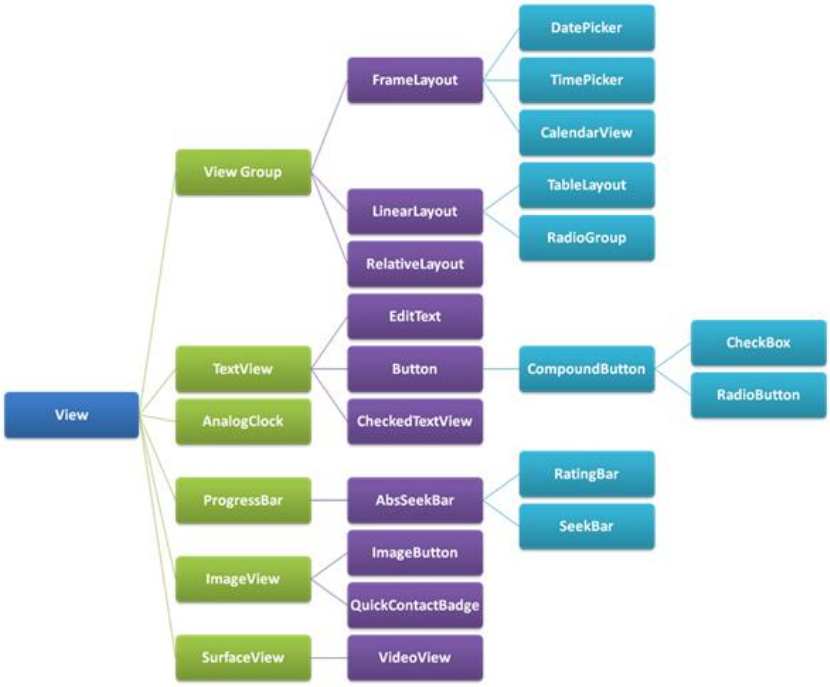

Figure 2.9: Class View Structure [37].

Finally, the Activity is the element with which users interact when using an application. An Application can have one or more activities which are in different states. The Activity class has a series of callbacks that allow the activity to know that its state has changed [35, 38].

Thus in this callback methods it is defined the behaviour of the activity so that the proper actions are taken at the proper time. On the other hand, it is necessary to handle transitions between states properly in order to achieve a robust performance [38]. A detailed explanation of how the life-cycle of an Android activity works can be found in Appendix E.

2.4 BASIC INTRODUCTION TO MACHINE LEARNING ALGORITHMS RELEVANT IN DATA MINING

As it has been pointed out before, algorithms are the key when trying to extract the patterns hidden in the information recorded by wearable devices, making them really useful for the user [1].

The recorded data can have different formats. Independently of this, the data can not be used directly but what is known as a feature vector must be built in order to train the machine learning algorithms. This feature vector must be a compact and relevant representation of the data [39].

The two main patterns recognition tasks that can be solved making use of machine learning algorithms are: classification and regression problems. In the first case, the input data is

associated with a class or category. As a consequence, the output is a discrete variable. Nevertheless, if the output variable is continuous, the problem to solve is a regression one. Basically in this case it is necessary to fit a mathematical function [40, 41]. In the case of personal data mining, both regression and classification algorithms can be used. Nevertheless the second option is the most popular approach [1].

Unfortunately, the data recorded is far from perfect. The algorithms have to deal with heterogeneity and redundancy problems. On the other hand, there are other issues that must be taken into account, as for example: finding an equilibrium between bias and variance, the well known as curse of dimensionality, low signal to noise ratio, the amount of training and testing data needed for the initial calibration of the algorithm, as well as linearity and non-linearity of the feature vector space [1, 39, 41].

When choosing an algorithm, the main parameter used to evaluate its performance is the accuracy. In order to measure it, the Classification Rate (CR) is calculated [42]. By naming the correctly classified data as CC and the incorrectly classified one as IC, the CR can be defined as [42]:

$$CR = \frac{CC}{CC + IC} \quad (2.1)$$

On the other hand, it is possible to distinguish between five types of learning approaches: Supervised Learning, Unsupervised Learning, Reinforcement Learning [1, 41], Semi-Supervised Learning [43], and Evolutionary Learning [41].

In the first approach, the set of data that is going to be used in the learning process is labelled, which means that has a target value associated that gives information about the class this data belongs to [1, 41].

In the second approach, the data sets are not labelled and as a consequence, the algorithm has to find similarities between the data in order to classify it [41].

In the case of Reinforcement Learning, the algorithm receives feedback. It is told if the answer it has provided is correct or not, but it is never told which is the correct one [41].

On the other hand, Semi-Supervised Learning tries to deal with the problem of needing a large amount of labelled data [39]. Thus, this approach tries to improve the results obtained with labelled data thanks to the information obtain with the unlabelled one [43]

Finally in the case of Evolutionary Learning, the idea of Biological Evolution is used. Thus, different solutions are graded, being the one with the highest grade or fitness, the correct one [41].

From the previous learning methods described, the most common one is Supervised Learning [39, 41].

2.4.1 ARTIFICIAL NEURAL NETWORKS

When mining personal data using smartphones and wearable devices, the algorithm known as Artificial Neural Network (ANN) has become highly relevant. An ANN consists of a series of artificial neurons, distributed in layers, which can be used to approximate any non-linear decision boundary. The most common type of ANN used for mining personal data is the Multi-Layer Perceptron (MLP), which has been used specially in activity recognition [1, 39].

2.4.1.1 Perceptron

The basic idea behind ANNs is simulating how the human brain works and specially how it learns. The learning process is the change in the strength of the connections between the neurons, which are known as synapses. Thus, the Hebb´s rule stablish that when two neurons fire frequently and at the same time, then the synapses between them are strengthened and vice versa. [41]

In the model designed by McCulloch and Pitts a mathematical analogy is stablished between a brain neuron and an artificial one. As it can be seen in Figure 2.10, this artificial neuron is composed by a set of weights (synapses), an adder that sums all the input signals that have been previously multiplied by the weights, and an activation function that determines if the neuron fires or not for a specific set of inputs. Initially, the activation function was a threshold function, and as a consequence the neuron fired if the value of $h = \sum_{i=1}^{m} w_i x_i$ was bigger than a threshold value θ [41].

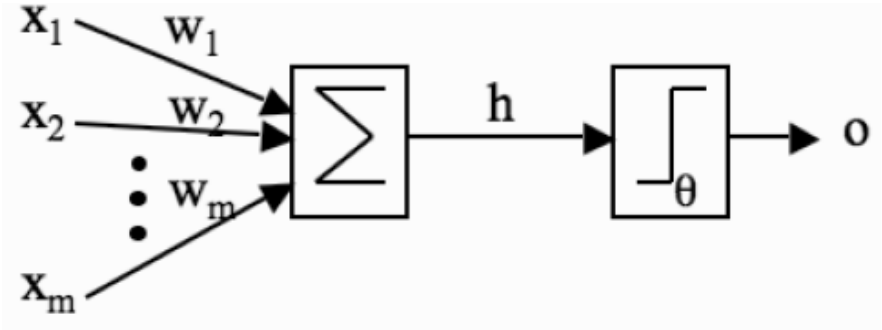

Figure 2.10: Mathematical model of a neuron proposed by McCulloch and Pitts [41].

Thus, the output value will be [41]:

$$o = g(h) = \begin{cases} 1 \; if \; h > \theta \\ 0 \; if \; h \leq \theta \end{cases} \quad (2.2)$$

The learning process takes place by changing the weights if the neuron has not fired when it should. This tries to imitate how the synapses are strengthened or not in a normal neuron. Other parameter that can be changed is the value of the threshold, which will be discussed later when the model of the Perceptron is explained [41].

There is no need to say that this model is very limited, as it is unable to catch the asynchronous nature of the synapses strengthening. Moreover, the value of the weights can change from positive to negative. The synapses in the brain are excitatory or inhibitory but can not change from one type to the other [41].

The concept of the Perceptron arises from the fact that one neuron alone can not do much by itself, so the logical thing to do is gathering some neurons together. Thus, there will be a set of neurons put in a layer and a set of weights w_{ij} that connects them to every input, as it can be seen in Figure 2.11. On the other hand, a threshold activation function will be necessary [41, 44].

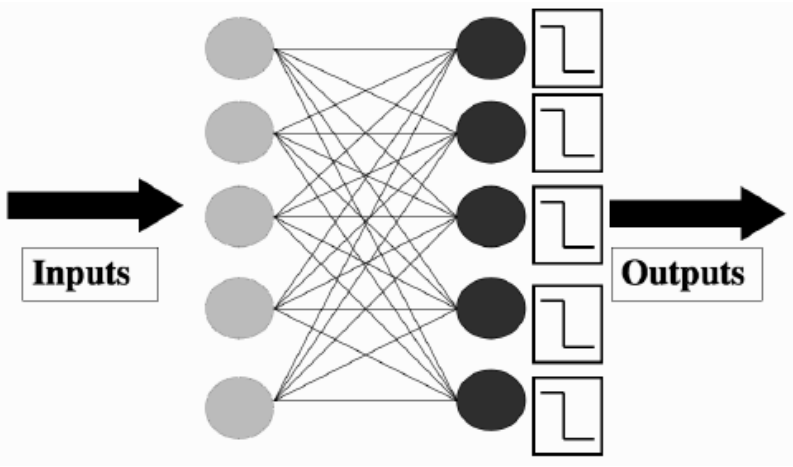

Figure 2.11: Architecture of a Perceptron [41].

As it has been suggested before, in order to make the perceptron learn, changing the value of the threshold or of the weights is necessary. In order to change the weights, the value produced by a neuron given an input, is compared with the expected result or target. Naming w_{ij} the weight that connects the input i with the neuron j, y_j the value produced by the neuron, and $t_j$the target, the learning rule can be written as [41]:

$$\Delta w_{ij} = -(y_j - t_j)x_i \quad (2.3)$$

$$w_{ij} \leftarrow w_{ij} + \eta \cdot \Delta w_{ij} \quad (2.4)$$

It is worth noticing that if the value produced by the neuron is bigger than the target, then $(y_j - t_j)$ will be positive and this means that the respective weight should be decreased. Nevertheless, due to the fact that the input may be negative, it is necessary to take into account its value, as it can be seen in (2.3) [41].

The parameter η is known as learning rate and determines how fast the neural network learns. Its value is taken between 0 and 0.4 in order not to make the network unstable [41].

As it has been pointed out above, the other parameter that can be changed is the threshold of the activation function. In order to do this, an extra weight is added and connects to a new input known as bias, which value is going to be a number different to 0, in general 1 or -1 [41, 44]. Thus, it can be controlled if a neuron is going to fire or not [41].

In order to understand all what have been exposed in the previous paragraph, it is worth taking a look at Figure 2.12. There, it can be seen that the weight vector is perpendicular to the decision boundary and that the bias vector determines its position. That the weight vector is perpendicular to the decision boundary is easy to demonstrate by using two points x_A and x_B that are on it. Taking into account that $y(x_A) = 0 = y(x_B)$, then [41, 44]:

$$y(x_A) - y(x_B) = 0 = w(x_A - x_B) \quad (2.5)$$

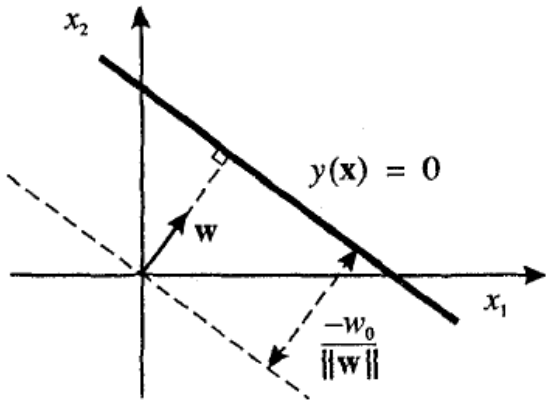

Figure 2.12: A linear decision boundary where the weight vector defines its orientation and the bias w_0 its position [44].

The main problem of the Perceptron is that it is only able to solve problems where classes are linearly separable. Nevertheless, for any data set that is linearly separable it will find a solution in a finite number of steps [44].

2.4.1.2 Multi-Layer Perceptron

After the Perceptron, a new type of neural network was designed in order to solve problems that are not linearly separable. This was the MLP.

As it can be seen in Figure 2.13, the structure of a MLP consists in an input and output layer, like in the case of the Perceptron, plus a certain number of hidden layers which are formed by a certain number of neurons [40].

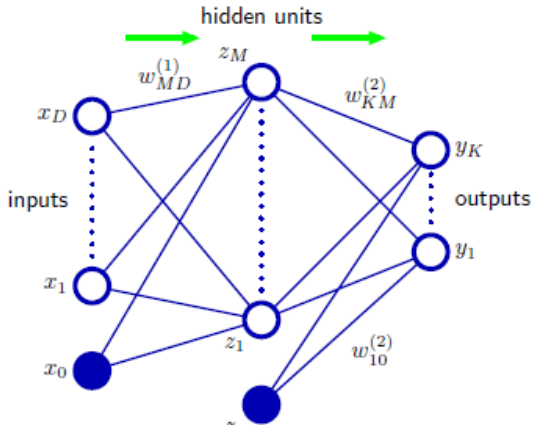

Figure 2.13: General structure of a MLP [40].

A MLP is not a group of Perceptrons, but a group of logistic regression functions with non-linarites [40].

In a MLP there are two general steps in the learning process. First, a forward step in which the inputs go into the network and the outputs are computed. Secondly, the backward step, in which according to an error function the weights are updated. This last one is the most difficult part, because the error has to be taken backwards across the network. In order to do this, the Gradient Descent method is used [41].

After calculating the error, the gradients are computed in order to update the weights. Nevertheless, it is worth noticing some issues, as for example that the inputs are unknown for the output neurons. Moreover, the targets are unknown for the hidden neurons, and for the extra hidden layers neither the inputs nor the targets are known. In order to deal with this problem the chain rule is used [41].

In order to visualize how backpropagation works, the path $K - M - D$ in Figure 2.13 is used. During the following explanation, the error will be defined as [40, 41]:

$$E(\boldsymbol{t}, \boldsymbol{y}) = \frac{1}{2}\sum_{k=1}^{N}(y_k - t_k)^2 \quad (2.6)$$

Firstly, it is necessary to derivate the error ε with respect to the weights of the second layer:

$$\frac{\partial \varepsilon}{\partial w^{(2)}_{KM}} = \frac{\partial \varepsilon}{\partial y_k} \cdot \frac{\partial y_k}{\partial w^{(2)}_{KM}} = -(t_k - y_k) \cdot \frac{\partial \varphi(y^{in}_k)}{\partial w^{(2)}_{KM}} = -(t_k - y_k) \cdot \varphi'(y_k) \cdot \frac{\partial \varphi(y_k)}{\partial w^{(2)}_{KM}}$$
$$= -(t_k - y_k) \cdot \varphi'(y_k) \cdot z_M \quad (2.7)$$

Secondly, the derivate the error ε with respect to the weights of the first layer is calculated:

$$\frac{\partial \varepsilon}{\partial w_{MD}^{(1)}} = \sum_k \frac{\partial \varepsilon}{\partial y_k} \cdot \frac{\partial y_k}{\partial w_{MD}^{(1)}} = -\sum_k (t_k - y_k) \cdot \frac{\partial y_k}{\partial w_{MD}^{(1)}} = = -\sum_k (t_k - y_k) \cdot \varphi'\left(y_k^{in}\right) \cdot \frac{\partial y_k^{in}}{\partial w_{MD}^{(1)}}$$
$$= = -\sum_k \delta_k \cdot w_{KM}^{(2)} \cdot \frac{\partial z_M}{\partial w_{MD}^{(1)}} = -\sum_k \delta_k \cdot w_{KM}^{(2)} \cdot \varphi'\left(z_M^{in}\right) \cdot \frac{\partial z_M^{in}}{\partial w_{MD}^{(1)}} =$$
$$= -\sum_k \delta_k \cdot w_{KM}^{(2)} \cdot \varphi'\left(z_M^{in}\right) \cdot x_D \quad (2.8)$$

The terms y_k^{in} and z_M^{in} are respectively the input of the neurons M and K.

Other important point is that in the MLP the activation function has to be differentiable and as a consequence the threshold function can not be used. Some examples of activation functions used are [41, 45]:

- Linear: $y_k = g(h_k) = h_k$
- Sigmoid: $y_k = g(h_k) = 1/(1 + \exp(-\beta h_k))$
- Soft-max: $y_k = g(h_k) = \exp(h_k)/\sum_{k=1}^{N} \exp(h_k)$
- ReLu : $y_k = \max(0, h_k)$
- Leaky ReLu: $y_k = \begin{cases} h_k \; if \; h_k > 0 \\ k \cdot h_k \text{ otherwise} \end{cases}$

One of the problems related to ANNs is that the function that has to be optimized is not convex, and as a consequence convergence may take place in a local minima, instead of in the global one [40, 41]. On the other hand, an ANN requires more data for its training and validation than other algorithms as for example Support Vector Machine (SVM) [39].
Finally, in order to improve the performance of an ANN, some good practices must be taken into account: learning rate reduction and weight decay with the training time, adding momentum when updating the weights in order to avoid local minima, and proper choosing of weight initialization ($-1/\sqrt{n} < w < 1/\sqrt{n}$, where n is the number of datasets) [41].

2.4.2 OTHER ALGORITHMS

Apart from ANN, the SVM is another widely used algorithm when mining personal data. The SVM was designed to solve the optimal separation problem. In this case, the members of each class are separated by the biggest possible margin, which is defined by the group of members that are more likely to be misclassified [1, 41]. These members are known as support vectors. SVMs have been successfully used to analyse data recorded by wearable devices, in areas like injury rehabilitation, real-time activity recognition, or prediction of the usage of mobile applications, among others [1].

In addition to SVMs, other statistical classifiers have been used together with wearable devices. For example, the K-Nearest Neighbour has been applied in medicine in order to help with rehabilitation tasks, or in other areas like energy efficiency, differentiation between stress and cognitive load, real-time activity recognition, etc. [1].

Other examples of widely used algorithms in data mining are the tree-based models, like the Random Forest, which have been successfully used in tasks like activity recognition or privacy management [1].

Also, it is worth noticing probabilistic models such as Naïve Bayes, which have been used in order to carry out activity recognition and physiological data analysis [1].

Finally, Deep Learning algorithms can also be used. They are models in which a series of feature extractors and nonlinearities are distributed in different levels. The complexity of the features learned increases with the depth of the model. Some examples of these models are: Convolutional Neural Networks, Restricted Boltzman Machines or Long-Short Term Memory (LSTM) Networks. They have been used in many types of EEG-based BCI systems [39]. A detailed explanation of how LSTM Networks work can be found in Appendix A.

Chapter 3

Method

In this Chapter, the steps followed to build an Android device based interface for interpreting the biosignals recorded by wearable devices are explained. This Application is able to connect to the wearable device known as Senso Medical Bio Pot V2, and receive the biosignals recorded by it, making use of BLE protocols. Finally, pretrained TensorFlow and Keras models are deployed on the application in order to classify these biosignals.

Both, the models used in this project. and how Senso Medical Bio Pot V2 has been validated, are explained in this chapter.

3.1 VALIDATION OF SENSO MEDICAL BIO POT V2

As it has been mentioned before, Senso Medical Bio Pot V2 is considered in this project as a candidate wearable device suitable for collecting EEG, EMG and ECG recordings via skin tattoo electrodes.

EMG recordings have been carried out using this device. Three types of movements were recorded: hand opened, hand closed, and thumb and index finger pinched. A 24 years old intact limbed male volunteer executed all the movements for the different recordings. The electrode was placed around the circumference of the upper part of the forearm at the volunteer's dominant side, as it can be seen in Figure 3.1. The records have a duration of 60s. During this time, only one movement was executed. Thanks to these recordings, it is possible to calculate the sampling frequency of the device, using the recording time and the number of samples. Moreover, it is necessary to analyse if there are information losses consequence of RF blind spots. This is done using the time stamp of the samples. Also, other EEG and ECG recordings of arbitrary length have been carried out. These signals were filtered using Matlab. Nevertheless, the filtering depends on the nature of the signal. In the case of ECG, the main components of the signal have a frequency from 1 to 100 Hz. On the other hand, in addition to the typical sources of noise, like the power line interference, motion artefacts, and the noise generated by the electronic device itself, there are other sources that must be taken into account, as for example muscle contraction or breathing sound [46]. Thus, in order to improve the quality of the signal, it should be enough to use a band pass filter with cut-off frequencies of 1Hz and 100Hz and a notch filter at 50Hz for the power line noise.

CHAPTER 3: METHOD

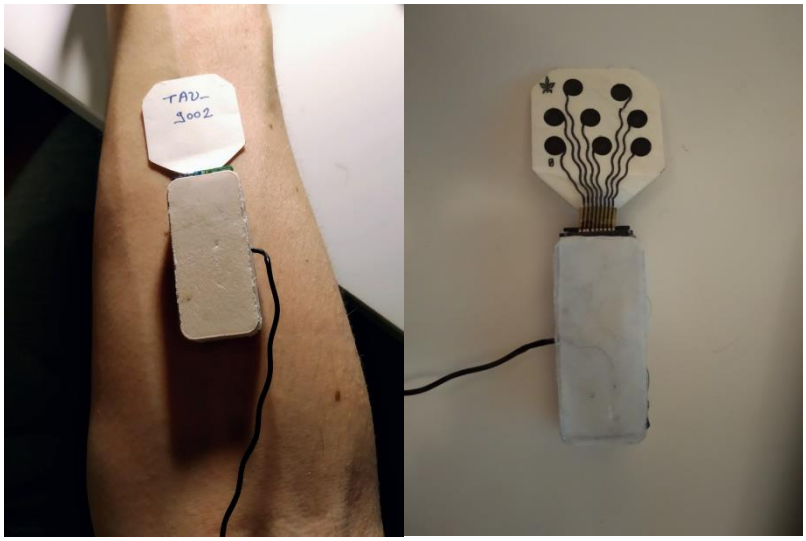

Figure 3.1: At the left, forearm placement of Senso Medical Bio Pot V2 and the tattoo electrode for signal recording. At the right, Senso Medical Bio Pot V2 connected to the tattoo electrode.

In the case of EEG signals, the same reasoning is followed. Nevertheless, it is worth noticing that the frequencies of interest are distributed in bands: δ (0-4 Hz), θ (4-8 Hz), α (8-12 Hz), β (12-30 Hz), γ (0-4 Hz), or μ (30-70 Hz) [47]. Thus, the signal is filtered in these bands separately, using a bandpass filter with cut-off frequencies coincident with the limits of a specific band, and a notch filter at 50Hz.

In the case of an EMG signal the frequencies of the components of interest are between 20 and 450 Hz, and like in other cases there is a noise component at 50Hz, consequence of the power line [17]. Therefore, an EMG signal should be filtered using a bandpass filter with cut-off frequencies of 20 and 450 Hz, and a notch filter at 50 Hz.

It is also interesting to study if the signal recorded has a high enough quality to keep the most relevant patterns that allows it to be classified. Thus, the Matlab Toolbox "Neural Net Pattern Recognition" has been used in this Project to build a MLP in order to classify the EMG data described above. The Fast Fourier Transform (FFT) and Principal Component Analysis (PCA) algorithms are employed for feature extraction. This architecture has been selected as it has been used successfully for the same task in other works [14]. The data was filtered using a bandpass filter with cut-off frequencies of 20 and 243 Hz, and a notch filter at 50 Hz. The number of units in the hidden layer of the network has been determined manually, starting with 5 neurons and increasing its number in 5 units every time. Thus, 30 units have been used as it allowed achieving the highest classification accuracy in the test dataset. Only one hidden layer can be used in the Toolbox. Finally, 70 % of the data is used for training, 15 % for validation and the other 15 % for testing. It is of the utmost importance to understand that for the current Project the MLP architectures are used just as a baseline for a neural network approach to classification.

In Table 3.1, relevant information about the EMG data used to train the MLP can be found.

Class	Initial size	Data size after applying PCA
Hand opened	51x200x8	51x64
Hand closed	63x200x8	63x64
Thumb and index finger pinched	66x200x8	66x64

Table 3.1: In this table, the size of the EMG data recorded using Senso Medical Bio Pot V2 is displayed. Initially, the data is represented as a tridimensional tensor, in which the first dimension is the number of elements of each class, the second dimension is the number of sample points, and the third dimension is the number of channels of the recording device. After applying the PCA algorithm for dimensionality reduction, a bidimensional tensor is obtained.

3.2 DEVELOPMENT OF AN ANDROID DEVICE BASED INTERFACE

The application has two classes that extend the interface AppCompatActivity and as a consequence act as activities. These are MainActivity.java and Result.java. They count with their respective layouts, which can be seen in Figure 3.2.

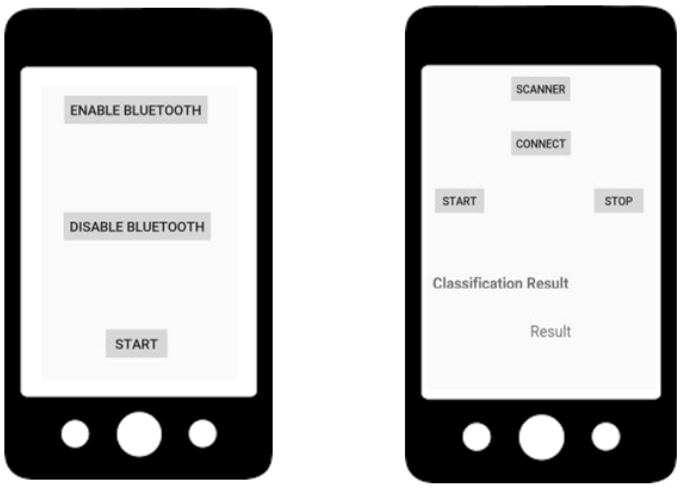

Figure 3.2: Layouts of MainActivity.java (left) and Result.java (right).

In MainActivity.java it is checked if the smartphone has a Bluetooth connection and it is possible to activate or deactivate it. From this activity it is possible to access to Result.java.

In MainActivity.java the following permissions are requested:

- Access to the location of the device.
- Access to the Bluetooth connection of the device.

On the other hand, in the activity Result.java it is possible to search for BLE Devices, and connect with Senso Medical Bio Pot V2 once it is found. Moreover, there are two buttons to start and stop the recording process. On the other hand, the result of the classification of the signal is showed in a text at the bottom of the screen, below the title Classification Result.

In addition to these two activities the application makes use of another important class named Sensor.java, in which the BLE connection and communication are handled using a background thread.

It is worth noticing that when an activity starts and a new Linux process is created, an execution thread is created with it. This thread is the User-Interface (UI) thread and is in charge of everything that happens on screen [48]. Nevertheless, thanks to the Java Virtual Machine it is possible to have multiple threads of execution running at the same time [49]. Nevertheless, every thread has a priority, and thus in the case of having a problem of lack of resources those with a higher priority are executed before [48].

3.2.1 IMPLEMENTATION OF BLUETOOTH LOW ENERGY IN ANDROID

One of the key points in this project is having a robust implementation of a BLE connection between the smartphone and the wearable device. In order to achieve this objective, additional threads to carry out work in the background have been created. The main reason for this is that if the execution time of the operations carried out in the UI thread exceeds 16 ms, “lagging problems” may occur, and if they take more than 5s, on screen appears an Application Not Responding (ANR) dialog, allowing the user to close the app directly. Moreover, blocking binder threads, which are used by an application to communicate with other applications or with the system, can cause several problems in the smartphone [48, 49]. All this justify the use of a HandlerThread to create a background thread where all the asynchronous processes, which characterize the communication between devices that use BLE technology, can take place. A class named Sensor.java that implements Handler.Callback is created. This way a Handler, MessageQueue, and a Looper are created for the background thread [50, 51]. How these elements interact with each other can be seen in Figure 3.3.

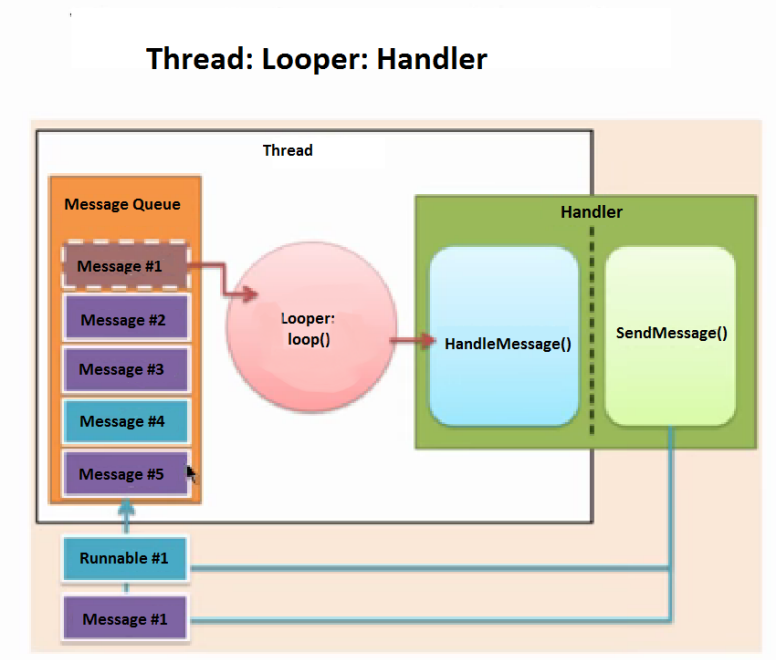

Figure 3.3: Scheme of how a Thread, a Handler and a Looper work [52].

In order to create a new thread with a Handler and a Looper, the next steps are followed. First, a HandlerThread is built using a String, which is its name, in this case "BleThread". Secondly, the thread is started and finally a Handler itself is created. Thus, everything sent to this Handler is processed in this thread. It is worth noticing that when a Handler is created, it is necessary to use as parameters a Looper and a callback interface in which to handle messages. The callback interface is the class Sensor itself, and the Looper the one corresponding to the background thread already created.

The next critical step is implementing the HandleMessage() method for the Handler. In this method a switch statement is defined. In it, the received message has its "what" field evaluated, and then, depending on its value, a method is executed in the background thread. The specific implementation of this method as well as the values of the "what" fields are linked to in Appendix F.1.

Nevertheless, not all the BLE activities are executed out of the UI thread. The discovery process of the BLE device is implemented in an API different to the one in which the communication is carried out. The discovery process does not have the thread safety problems described above. Thus, it is done in the activity Result.java, and all the operations are executed in the UI thread. The discovery process has a first scanning operation which due to performance issues can not go on forever, and therefore it is limited to 10s. Its result is received in the form of a callback. Thus, once Senso Medical Bio Pot V2 is found, its address is stored using BluetoothDevice.getAddress(), and the device itself is represented by an object of the class Bluetooth Device, named in this case mDevice.

Once the scanning process is finished, the device of interest has been found and the Bluetooth Device object created, the connection process must be carried out. In order to do this, the method connectGatt() is called. In this method, three parameters are used: the context of the Application, a

boolean which value is false (in order not to have to wait for the device to be visible for the BLE framework, as it is already stored in mDevice), and a callback that is known as BluetoothGattCallback. The methods of this callback take place in a binder thread by defect, which has been avoided by making them take place in a background thread.

The BluetoothGattCallback has different methods. For example, onConnectionStateChanged() is executed when the connection state has suffered any changes, and determines the new statues of the connection. It is worth noticing that if the connection has been successful the message MSG_CONNECTED is sent to the Handler in order to carry out the discovery of the services. On the other hand, onServiceDiscovered() is called when the services have been successfully discovered, and onCharacteristicChanged when the value of one of the characteristic of the device with which the connection is being stablished changes. In the case of this project there are 5 characteristic, but it is interesting just the change of the characteristic number 4, as it stores the values of the measured signal. Thus, when a new value is received the message MSG_DATA is sent to the handler so that the new value is processed and stored. The whole structure of the BluetoothGattCallBack can be seen in Appendix F.1.

As it has already been mentioned above, the connection and communication processes are not thread-safe and as a consequence a background thread is created for it. All this is implemented in the class Sensor.java. The connection process starts by clicking the button Connect in Result.java. When this is done, if the device has been found, an object of the class Sensor is created, passing as parameters to the constructor: mDevice, the Application Context and a Handler for Result.java. On the other hand, in the constructor of Sensor.class, the connection itself takes place by using the method connectGatt(). This can be seen with more detail in Appendix F.1.

Once the connection takes place and the services have been discovered, it is possible to start the measurement process. In order to do this, the value of the Characteristic 2 must be changed, as it is explained in Appendix C, by writing 0x1 on it to start the acquisition, or 0x0 to finish it. This has been implemented in the method acquisition, which receives the boolean startAcq as parameter. If startAcq is “true” the acquisition begins, and if not, it stops.

It is worth noticing that if the acquisition is going to start, the notifications must be enabled. This makes it possible to start reading and writing data from the remote device.

The last point it is necessary to discuss is the disconnection process. It is carried out calling the method BluetoothGatt.disconnect(), which triggers the call to the method onConnectionStateChange() in BluetoothGattCallback. The call to the method BluetoothGatt.disconnect() takes place when the Handler receives the message MSG_DISCONNECT. Moreover, once onConnectionStateChange() is called, the Handler receives the message MSG_DISCONNECTED and the method BluetoothGatt.close() is used. Thus, the system stops triggering the methods of BluetoothGattCallback.

3.2.2 DEPLOYMENT OF MACHINE LEARNING MODELS ON AN ANDROID APPLICATION

As it has been pointed out before, the implementation of machine learning models in Android has been done making use of the framework TensorFlow Mobile.

The first step in order to use TensorFlow Mobile in Android, is implementing the appropriate dependency in the gradle field that every Android application has. In this case, the dependency used is: "org.tensorflow:tensorflow-android:1.13.0-rc0".

In Chapter 2, the Android SDK has already been mentioned, as it contains all the API libraries and developer tools that are going to be used when building, testing, and debuging an Android application [53]. Nevertheless, this is not enough if a framework like TensorFlow Mobile is going to be implemented in an application. It is also necessary to make use of the Android Native Development Kit (NDK), which contains a series of tools that enables the developer to use C/C++ code in Android, control native activities, and access components of physical devices, as for example sensors [54]. In this Project, the Android NDK is basically used to run C++ code in the application as the core of TensorFlow is written in this language.

Once the dependency of TensorFlow Mobile has been implemented, it is time to build a TensorFlow or Keras model, which details are explained later in this chapter, and convert it to a protobuf (.pb) file. This file contains the parameters learned by the machine learning algorithm which has been trained outside the smartphone. In order to get this file, it is necessary to save the TensorFlow model in a checkpoint file (.ckpt) when the training process is finished. After this, the graph defined in our model is passed to a .pbtxt file. Once this is done, the graph parameters are turned into constants. In this last step the .pb file is generated and stored in the Asset folder inside the Application. All these steps are done making use of specific TensorFlow functions: "tf.train.Saver()", "tf.train.write_graph" and "freeze_graph.freeze_graph()". How all the parameters used in these functions are defined can be seen with more detail in Appendix F.3.

The next step is implementing and defining the specific java objects in order to carry out the inference of the data received by the application using the generated .pb file. Thus, when the activity Result.java is created, a TensorFlowInferenceInterface object is created with it. This is done in an additional thread, as this process requires too much time to be done in the UI thread. The object name is "tfHelper", and in order to create it an AssetManager and the path to the .pb file is given as parameters. The AssetManager enables the access to the .pb file (see Appendix F.2).

Once the TensorFlowInferenceInterface object is created, the inference process can be carried out. Thus, the activity Result.java receives a CopyOnWriteArrayList object that contains a certain number of recordings. In this Project the idea is classifying windows of 200ms, which should be enough to extract the important information from EMG and EEG signals. Thus, the size of the data received in Result.java is determined by the windows size and the sampling frequency of the sensor. Once this data is received it is converted to a float vector and sent to the tfHelper.feed() method. This method receives as additional parameters the size of the input, which is a long type array that allows defining tensors as the input array is one-dimensional, and the input names which coincides with the names of the nodes that have been defined in the TensorFlow graph. After this method is called, it is the turn for the methods run() and fetch(). The first one runs the inference, and the

second one allows accessing the result of the inference, that in this case is the probability of the biosignal being of one type or another. All this code can be seen with more detail in Appendix F.2.

Finally, it is necessary to design a multithreading strategy that enables receiving and classifying signals, without having problems in the UI thread. In order to do this, a ThreadPool is used. The main reason behind this is its suitability when large numbers of asynchronous tasks have to be processed, and when a real-time performance needs to be achieved. Thus, a ThreadPoolExecuter is created and named mExecuter. This object uses as parameters the work queue where the task are going to be processed, the number of cores that can be used, and the time a thread can wait for receiving new tasks when there are more threads than cores. As in the previous cases, a more detailed implementation can be seen in Appendix F.2.

3.3 DESIGN OF A LSTM NETWORK FOR BIOSIGNALS CLASSIFICATION

3.3.1 DESIGN OF A LSTM NETWORK FOR EMG SIGNALS CLASSIFICATION

In this work a LSTM Network was designed and implemented thanks to TensorFlow (see Appendix F.3) so that it could be deployed in an Android Application using the framework TensorFlow Mobile. The main reason of choosing this algorithm to analyse EMG signals, is its suitability to work with sequential data (see Appendix A).

Moreover, in order to study the scalability of the application prototype, the LSTM Network has been trained using an EMG dataset that has been taken from the web [55]. The structure of the network can be seen in Figure 3.4.

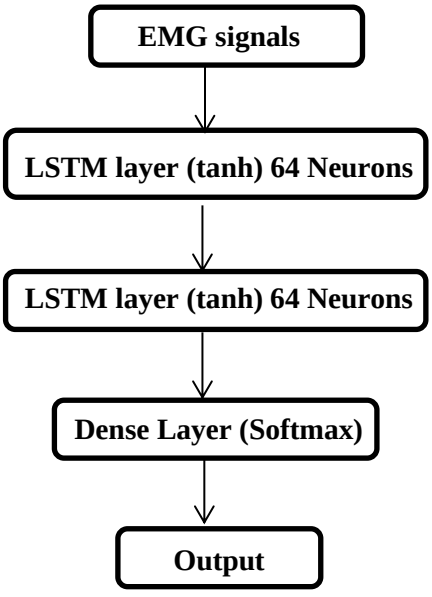

Figure 3.4: LSTM Network Architecture for EMG data classification.

It is formed by two LSTM layers followed by a fully connected layer. The layers have a number of 64 neurons. The last fully connected layer uses softmax as activation function, and outputs the

probability of the input being a member of one of the classes of movements that are going to be classified.

The dataset used consists in a series of recordings from 20 intact limbed subjects. These correspond to 4 channel myoelectric signals, recorded with a sampling frequency of 2 kHz with a 14-bits resolution. The electrodes were equally spaced around the forearm proximal third. Most of the subjects were untrained (80%) and only one subject had the electrodes placed in the dominant side. The average age was 30.1 (±11.1) years; 177 (±8) cm height; 68.8 (±11.0) kg weight; and 10 were females (50%). None of the subjects had history of neuromuscular disorders. The experiment was carried out with the consentient of all the participants who agreed with the publication of their recording session [56]. Finally, every movement was executed in 3s contraction followed by 3s relaxation, all this is done a total of three times per record (total record time of 18 s) [55].

A total of 10 movements were registered. But only three have been used in this Project. Moreover, only data from two trained subjects has been employed. The movements chosen were: hand closed, hand opened and supination.

Taking into account that the window used is a 100 ms one and that the sampling frequency is 2 kHz, the length of the input data is 200. The fact that the recording device uses 4 channels results in an input size like the one in Table 3.2.

Class	Size
Hand opened	90x200x4
Hand closed	90x200x4
Supination	90x200x4

Table 3.2: In this table the data size of EMG recordings is displayed. There are three dimensions. The first dimension represents the number of elements of each class, the second dimension is the number of sample points, and the third dimension the number of channels of the recording device.

The loss function used to train the network is the Cross Entropy Loss with Softmax. The optimizer employed to minimize it, was the Adam algorithm. Other relevant parameters used to train the network can be found in Table 3.3.

Parameters	Tested Values
Epoch	100
Mini batch size	20
Learning Rate (η)	0.0025
L2 Regularization term (λ)	0.0015

Table 3.3: Parameters used when training the LSTM Network in order to classify EMG data.

In previous studies good results have been obtained when classifying EMG signals using MLP [14]. Therefore, a MLP has been built and trained with the same dataset used with the LSTM Network. This is done using the Matlab Toolbox "Neural Net Pattern Recognition". As it has been already mentioned in this Chapter, the MLP is used just as a baseline for a neural network approach to classification. Its main purpose is helping to evaluate the performance achieved by a LSTM Network which architecture is optimised to be deployed in Android. The MLP used has one hidden layer with 35 units, which are determined manually, starting with 5 neurons and increasing or decreasing its number in 5 units depending on the performance. The data is pre-processed using the FFT and PCA for feature extraction. The size reduction of the input data can be appreciated in Table 3.4.

Class	Data size after applying PCA
Hand opened	90x16
Hand closed	90x16
Supination	90x16

Table 3.4: In this table the data size of EMG recordings after applying PCA for dimensionality reduction is displayed.

3.3.2 DESIGN OF LSTM NETWORKS FOR EEG SIGNALS CLASSIFICATION

In addition to the previous LSTM Network used to classify EMG signals, another one has been designed to classify EEG data. Like in the previous case, this one has also been converted to .pb format in order to be deployed in an Android Application using the framework TensorFlow Mobile. As it can be seen in Appendix F.4 this model has not been built using Tensorflow but Keras. This makes no difference as a Keras model can be converted to .pb format like a TensorFlow one.

In this case the dataset used for training the algorithm can also be found in the web [57]. It consists in EEG signals corresponding to answers of the brain to visual and audio stimuli. The data was recorded by a 32-channel cap and preprocessed using the Independent Component Analysis (ICA) algorithm. This dataset has already been used in another work to train a LSTM Network to differentiate between audio and visual stimuli. The accuracy achieved in this case was 85% and the

structure of the network can be seen in Figure 3.5 [57]. More information about the data can be found in Table 3.5.

Class	Size
Visual stimuli	1528x1300x32
Audio stimuli	1528x1300x32

Table 3.5: In this table the data size of EEG recordings is displayed. There are three dimensions. The first dimension represents the number of elements of each class, the second dimension is the number of sample points, and the third dimension is the number of channels.

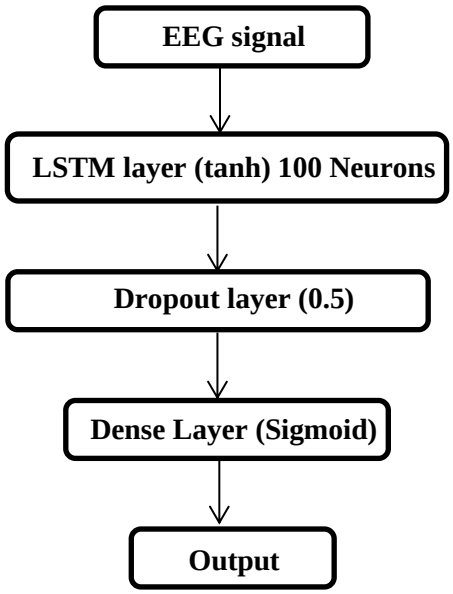

Figure 3.5: LSTM Network Architecture used in [54] for EEG data classification.

In this project the previous algorithm was changed in order to make it deployable on an Android Application. The main change introduced is the use of L2 regularization instead of Dropout layers. The structure of the network can be seen in Figure 3.6.

The loss function used to train the network is the Binary Cross Entropy Loss. The optimizer employed to minimize it, was the RMSProp algorithm. Other relevant parameters used to train the network can be found in Table 3.6.

Parameters	Tested Values
Epoch	50
Mini batch size	16
Learning Rate (η)	0.001
Momentum (ρ)	0.9
Epsilon(ϵ)	$1 \cdot 10^{-7}$
L2 Regularization term (λ)	0.00001

Table 3.6: Parameters used when training the LSTM Network in order to classify EEG data. The parameter Epsilon is used for numerical stability.

Although LSTM Networks have already shown its potential when classifying EEG data, it is worth comparing its performance with the one of algorithms used previously in this task, like the MLP [39]. Nevertheless, this is just done to stablish a baseline for a neural network approach to classification, as the current work focuses on those architectures that can be deployed on Android. Thus, a MLP is built and trained with the same dataset used with the LSTM Network. The Matlab Toolbox "Neural Net Pattern Recognition" is employed. Moreover, the MLP used has one hidden layer with 30 units, which are determined manually, starting with 5 neurons and increasing or decreasing its number in 5 units depending on the performance. Finally, the data is pre-processed using the FFT and PCA algorithms. The dimension of the data after being reduced can be seen in Table 3.7.

Class	Size after applying PCA
Visual stimuli	1528x1024
Audio stimuli	1528x1024

Table 3.7: In this table the data size after applying PCA for dimensionality reduction is displayed.

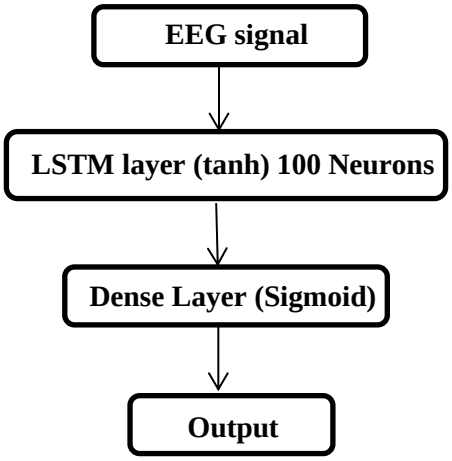

Figure 3.6: LSTM Network Architecture used for EEG classification.

.

Chapter 4

Results

4.1 PERFORMANCE OF SENSO MEDICAL BIO POT V2

One of the main purposes of this work has been to validate the use of Senso Medical Bio Pot V2 as a wearable device for continuous biosignal recording. The hypothesis was that SensoMedical BioPot V2 would be suitable for this task. However, the tests described in Chapter 3 have not provided convincing evidence in favour of the hypothesis.

The first problem found when using this sensor was the data losses, up to 50% of a total recording time of 60s. This can be a consequence of the RF blind spots even though the device is supposed to have a board memory buffer that should prevent it.

On the other hand, there have been other issues related to the sample frequency. As it has already been pointed out, the device should have a f_s = 500 Hz if it is used in an Android interface and a f_s = 2000 Hz if it is used in a Windows one. Nevertheless, the test carried out in Windows gave a f_s below 500Hz. More importantly, it was not constant. In some cases the value of f_s was around 487 Hz and as low as 321 Hz in others. This and the data losses are obstacles almost impossible to overcome if this device is supposed to be used as part of a decision support system.

Nevertheless, some test were carried out in order to see if applying the proper preprocessing a good EMG, EEG and ECG signals could be obtained.

As it has been mentioned in Chapter 3, a band pass filter with cut-off frequencies of 1Hz and 100Hz and a notch filter at 50Hz were applied to the ECG recordings. Nevertheless, there were no visible improvement for frequencies between 1Hz and 100Hz. Due to the fact that the dominant frequencies in ECG are in the range of 4-12 Hz [58], if a bandpass filter with cut off frequencies equal to 1 Hz and 10 Hz is used, it is possible to differentiate the heart bits, as it can be seen in Figure 4.1. In the power spectrum, it can also be noticed that the signal is too noisy to offer meaningful content in higher frequencies.

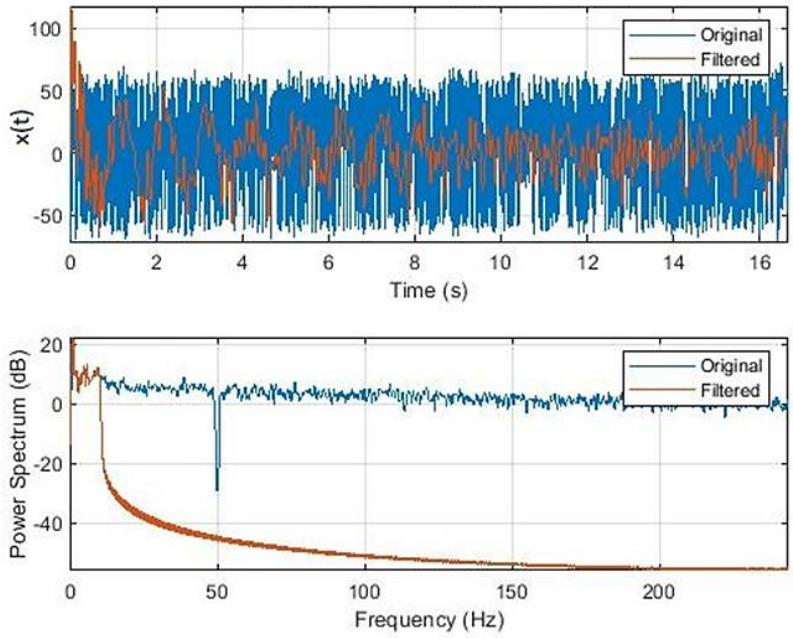

Figure 4.1: ECG signal recorded by Senso Medical Bio Pot V2. Its representation and the one of its Power Spectrum with respect to the time, at the top and at the bottom respectively. In both cases the original signal appears in blue and the filtered one in orange. The signal has been filtered using a bandpass filter with cut off frequencies equal to 1 and 10 Hz, and a notch filter at 50 Hz.

In the case of the EEG signals, the cut off frequencies in the bandpass filters were the ones corresponding to the bands: δ (0-4 Hz), θ (4-8 Hz), α (8-12 Hz), β (12-30 Hz), γ (0-4 Hz), or μ (30-70 Hz) [47]. On the other hand, there is a noise component of 50 Hz consequence of the power line. The results of filtering can be seen in Figure 4.2, and if the reader takes a look at the Power Spectrum, it is obvious that the signal is too corrupted by noise.

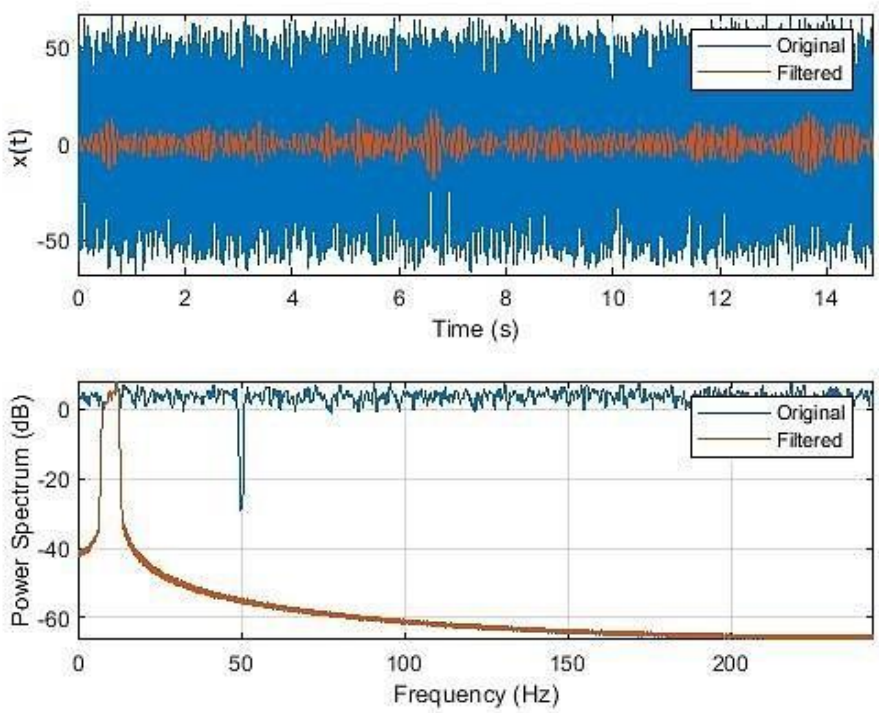

Figure 4.2: EEG signal recorded by Senso Medical Bio Pot V2. Its representation and the one of its Power Spectrum with respect to the time, at the top and at the bottom respectively. In both cases the original signal appears in blue and the filtered one in orange. The signal has been filtered using a bandpass filter with cut off frequencies equal to 4 and 8 Hz and a notch filter at 50 Hz.

Finally, in the case of the EMG records, there is also a noise component of 50Hz consequence of the power line. An EMG signal should be filtered using a bandpass filter with cut-off frequencies of 20 and 450 Hz, but the sampling frequency of the sensor is 487 Hz. The Nyquist–Shannon sampling theorem does not allow to filter in this interval of frequencies, as it is necessary to sample at least in a frequency that doubles the value of the maximum frequency of the signal that is going to be reconstructed. Therefore, it is only possible to use a bandpass filter with cut-off frequencies of 20 and 243 Hz. The results of filtering can be seen in Figure 4.3.

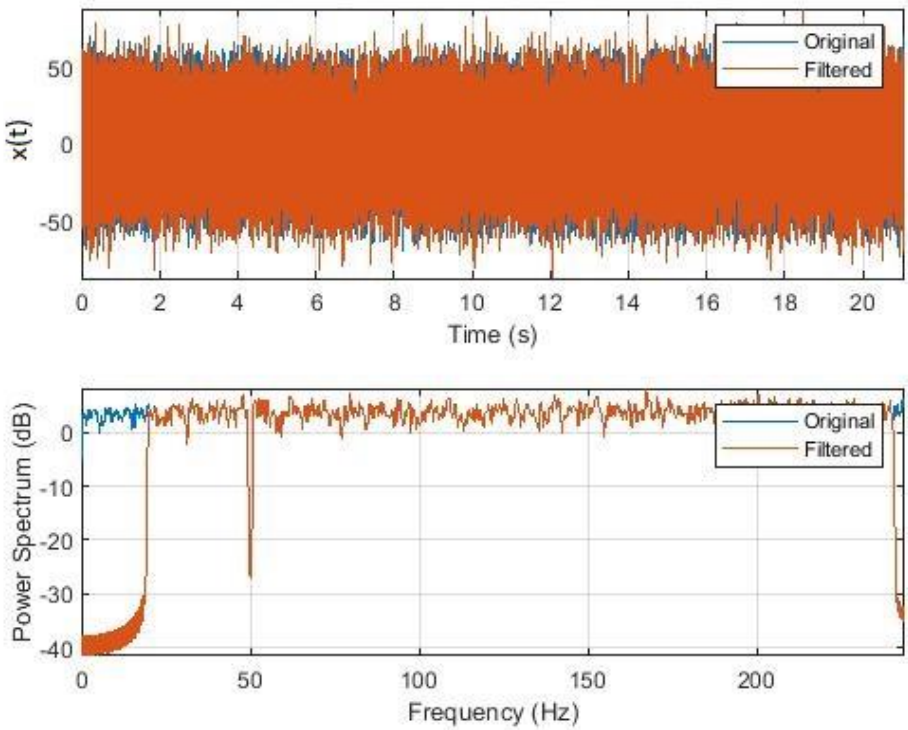

Figure 4.3: EMG signal recorded by Senso Medical Bio Pot V2. Its representation and the one of its Power Spectrum with respect to the time, at the top and at the bottom respectively. In both cases the original signal appears in blue and the filtered one in orange. The signal has been filtered using a bandpass filter with cut off frequencies equal to 20 and 243 Hz and a notch filter at 50 Hz.

In this case the signal is also significantly corrupted by noise.

Finally, it is relevant to examine if the MLP architecture described in Chapter 3, after some preprocessing, is able to classify the EMG data recorded using Senso Medical Bio Pot V2. This data corresponds to three hands movements: hand closed, hand open and thumb and index fingers pinched.

The highest accuracy in the validation dataset is reached at the 10^{th} epoch. At this point, the accuracy achieved in the training dataset is equal to 48.1 %. Both the confusion matrixes and loss evolution can be seen in Figure 4.4 and 4.5.

Figure 4.4: Confusion Matrixes which result from the classification of EMG data making use of a MLP. Three classes are classified: Hand Open, Hand Closed, and Thumb and Index Fingers Pinched. There is a Confusion Matrix per dataset (Training, Validation and Testing) and another one with the three types of datasets combined. In the diagonal of the matrix the percentage of the values correctly classified are displayed. Out of the diagonal the percentage of misclassified values is shown. Those squares that are not red or green show the overall accuracies.

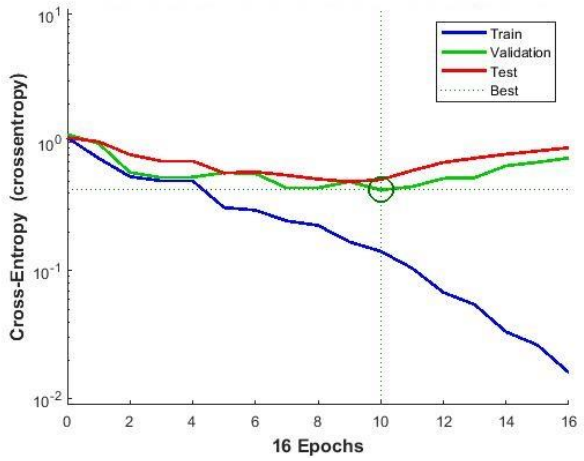

Figure 4.5: Loss evolution for the Training, Validation and Test datasets during the training of a MLP with EMG signals. Three classes are classified: Hand Open, Hand Closed, and Thumb and Index Finger Pinched. The features of the data have been extracted using FFT and PCA. The 10th epoch appears marked as it is the point in which a higher accuracy in the validation dataset is achieved.

4.2 ANDROID PROTOTYPE APPLICATION PERFORMANCE

After validating the use of Senso Medical Bio Pot V2 as a wearable device for recording EEG, ECG and EMG signals, it was concluded that the results were not favourable. Thus,

the real-time feasibility of the Android Application was not tested using this device. Instead, an extra thread has been added to have a signal that the Application can classify. In this thread, signals of different types are generated periodically and sent to the activity Result.java to be classified. The signals generated are:

- $x_1(t) = \cos(2 \cdot \pi \cdot t) + \cos(10 \cdot \pi \cdot t)$
- $x_2(t) = \sin(3 \cdot \pi \cdot t)$
- $x_3(t) = \cos(10 \cdot \pi \cdot t) + \cos(30 \cdot \pi \cdot t)$
- $x_4(t) = random.normal(0,1,size(t))$

The first three signals are a combination of cosines and sines with different amplitudes. The fourth signal is generated drawing random samples from a normal (Gaussian) distribution with values between 0 and 1. The value of the signal is not relevant as they are only used to test the real-time performance of the application.

This way the thread simulates the behaviour of the wearable device connected to the application.

The device used to test the application is a Samsung Galaxy Tab A SM-P580 which main characteristics are:

- RAM Memory: 3 GB.
- Processor: Exynos 7870 Octa-Core, 1.6 GHz.
- Inner Storage: 16 GB.
- Operating System: Android 8.1.

The performance of the application has been measured making use of a tool in Android Studio known as Android Profiler. The following parameters have been analysed:

- Maximum CPU usage (%).
- Average Memory usage (MB).
- Energy consumption (cualitative): light, medium or high.

Moreover, these parameters have been measured in the most important stages of the application: Start of the activity MainActivity.java, Bluetooth Enabling/Disabling, Start of the activity Result.java, Scanning Process, Connection Process, and Classification Process.

It is worth mentioning that Android Profiler also measures the Network activity. Nevertheless, it is not taken into account in this project as it is not a characteristic used in the application.

4.2.1 PERFORMANCE IN MAINACTIVITY.JAVA

When MainActivity.java starts, it can be seen in Figure 4.6 that there is some activity before it is made visible, which is a consequence of the activity being launched and the OnCreate() method being called. In this process, the values of the parameters measured are:

- Maximum CPU usage: 25 %.
- Average Memory usage: 280 MB.
- Energy consumption: Medium.

When the OnStart() method is called and the activity is made visible, the values of the parameters analysed are:

- Maximum CPU usage: 32 %.
- Average Memory usage: 207 MB.
- Energy consumption: Medium.

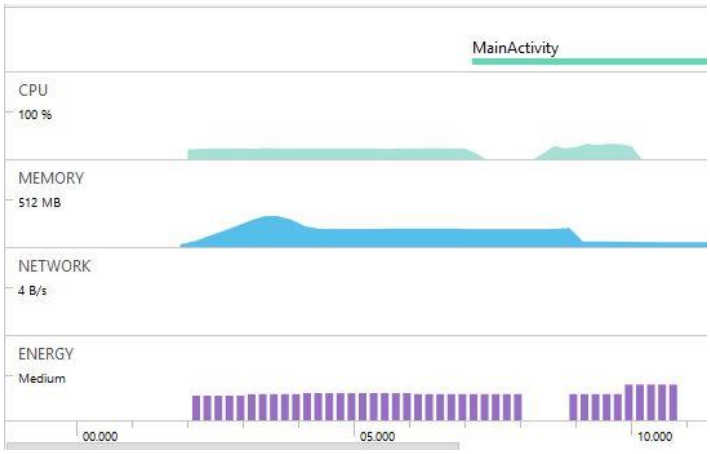

Figure 4.6: Resources consumed during the initialization process of the application and MainActivity.java.

Finally enabling and disabling the Bluetooth connection does not consume too many resources, as it can be seen in Figure 4.7.

When the Bluetooth connection is enabled, the resources consumed are:

- Maximum CPU usage: 17 %.
- Average Memory usage: 47 MB.
- Energy consumption: Light.

On the other hand, when the Bluetooth is disabled the values taken by the studied parameters are:

- Maximum CPU usage: 6 %.
- Average Memory usage: 53 MB.
- Energy consumption: Light.

Figure 4.7: Resources consumed during the enabling and disabling of the Bluetooth connection.

It should be pointed out that the memory used is divided in [59]:

- Java: memory used by objects created using Java or Kotlin code.
- Native: memory used by objects created using C or C++ code.
- Graphics: memory used in threads of graphs, aimed to show the pixels in the screen.
- Stack: Memory used in native and Java piles. Correlated with the number of subprocesses in execution.
- Code: Memory used for the code and resources like, byte code, libraries, etc.
- Other: Memory used and that the system is unable to categorize.
- Allocated: Amount of objects in Java and Kotlin assigned to the app without taking into account the C or C++ ones.

In this activity, when no other process is being done, the memory consumption is equal to 47.3 MB. It is distributed as follows:

- Java: 3.9 MB.
- Native: 9.8 MB.
- Graphics: 17.6 MB.
- Stack: 0.3 MB.
- Code: 14.1 MB.
- Others: 1.5 MB.

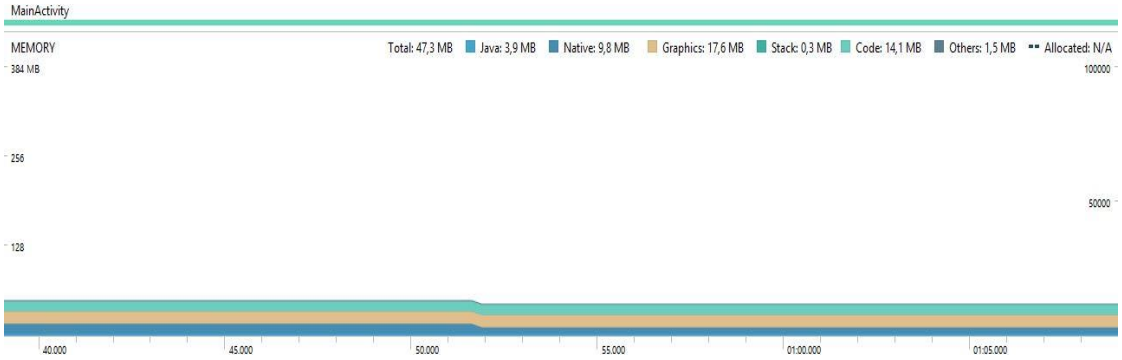

Figure 4.8: Memory consumption in MainActivity.java.

Therefore the graphics are the most memory consuming element. Nevertheless, in overall the activity is not heavy or difficult to process by the device.

4.2.2 PERFORMANCE IN RESULT.JAVA

This activity is the one in which more processes take place. On the other hand, in this activity multithreading is used in order to achieve a good real-time performance.

The most important processes that take place are: Start of the activity Result.java, Scanning Process, Connection Process and Classification Process.

When Result.java starts, the value of the parameters measured are:

- Maximum CPU usage: 30 %.
- Average Memory usage: 104 MB.
- Energy consumption: Medium.

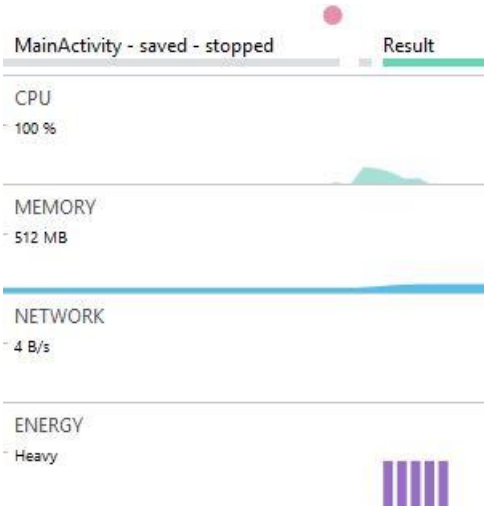

Figure 4.9: Resources consumed during the starting process of Result.java.

In the case of the scanning process, the values of the parameters measured are:

- Maximum CPU usage: 35 %.
- Average Memory usage: 104 MB.
- Energy consumption: Light.

In Figure 4.10 two picks can be distinguished and they correspond to the scanning start and stop process.

Figure 4.10: Resources consumed during the Scanning process.

When the connection with Senso Medical Bio Pot V2 device is stablished, the resources consumed are:

- CPU usage: 15 %.
- Memory usage: 104 MB.
- Energy consumption: Light.

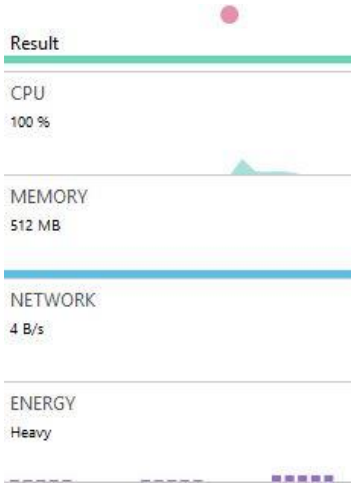

Figure 4.11: Resources consumed during the connection process.

Finally, it is the turn of the classification process, which is the most important process for the application. As it has been mentioned before, the classification is being done using the extra thread that substitutes Senso Medical Bio Pot V2 for the signal generation.

The values measured are:

- Maximum CPU usage: 30 %.
- Average Memory usage: 104 MB.
- Energy consumption: Light.

As it can be seen in Figure 4.12 there is a greater CPU consumption when the classification is about to start. Moreover, the starting is delayed a bit, causing some lag. After this, with every classification process the CPU consumption is distributed in picks. In these cases the CPU consumption is low, around 9 %. Moreover, it can be seen that the inference process takes around 195 ms, which makes the application adequate to be used for real-time EMG and EEG signal inference. The reason of this is that these signals can be analysed in windows of 200 ms or less, being the sampling frequency required in these cases below 1000 Hz [56, 60].

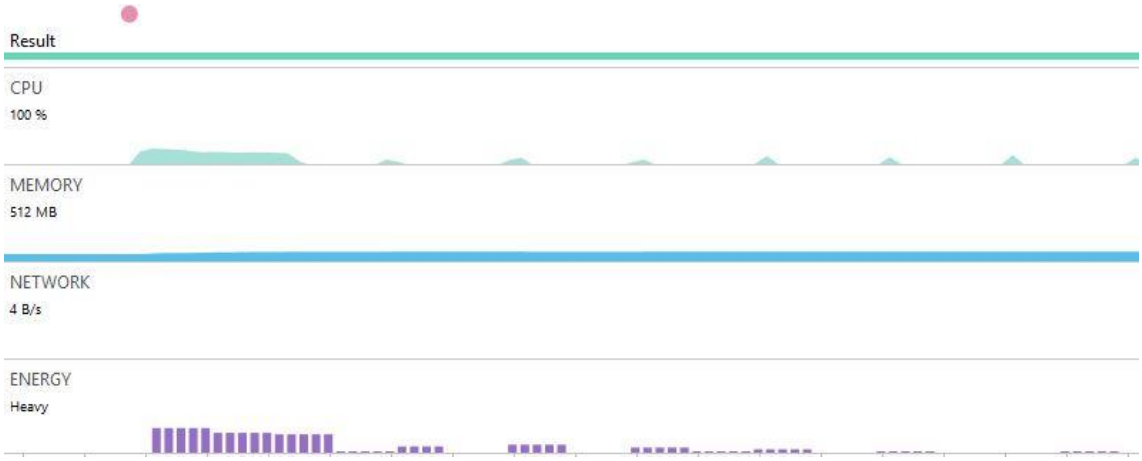

Figure 4.12: Resources consumed during the classification process.

Finally, in this activity when no other process is being done, the memory consumption is equal to 47.3 MB. It is distributed as follows:

- Java: 5.9 MB.
- Native: 51.1 MB.
- Graphics: 26.4 MB.
- Stack: 0.5 MB.
- Code: 21.2 MB.
- Others: 1.8 MB.

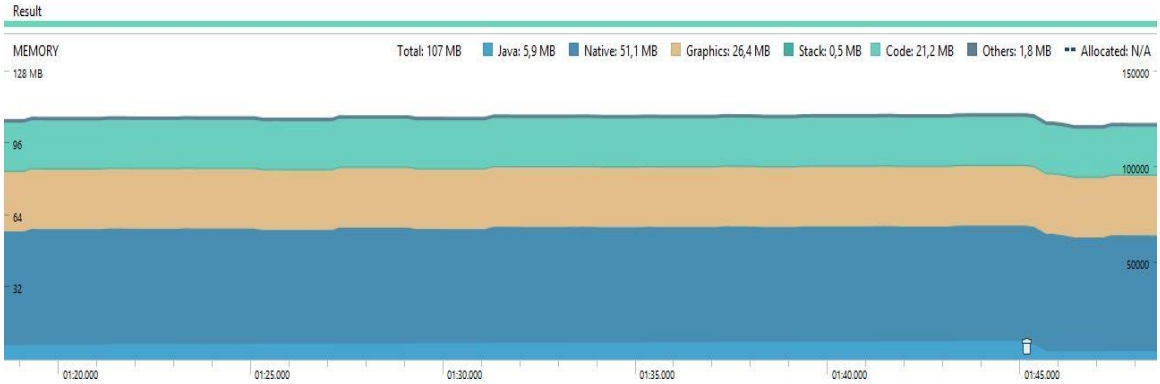

Figure 4.13: Memory consumption in Result.java.

In this case the native code is the one that consumes the most memory. The reason for this is that the code is used in the classification process. Another observation is that the graphics consume more memory than in the previous activity, as the interface is a bit more complex. Nevertheless, in overall this activity is not heavy or difficult to process by the device.

4.3 PERFORMANCE OF LSTM NETWORKS WHEN CLASSIFYING BIOSIGNALS

4.3.1 PERFORMANCE OF A LSTM NETWORK WHEN CLASSIFYING EMG SIGNALS

In order to study the scalability of the application prototype, the LSTM Network deployed on it has been trained using an EMG dataset. First of all, the data has been filtered using a bandpass filter with cut-off frequencies of 20 and 450 Hz, and a notch filter at 50 Hz. In Figure 4.14, it can be seen a comparison between the original signal and the filtered one.

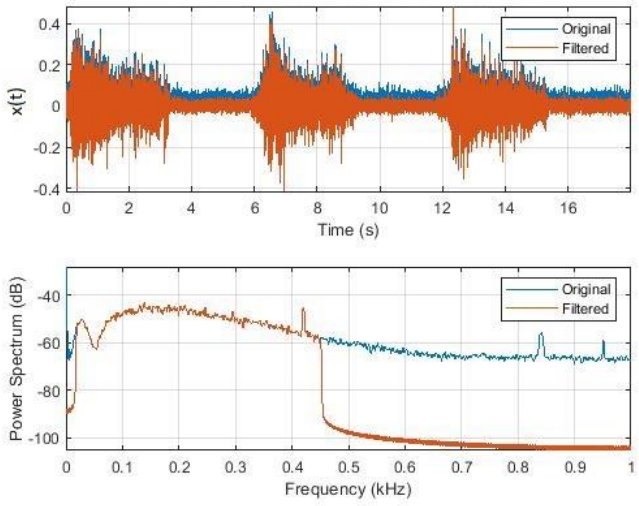

Figure 4.14: EMG data filtered using a bandpass and notch filters. The bandpass filter has cut off frequencies equal to 20 and 450 Hz. The notch filter is applied at 50 Hz.

After this, the Network is trained, obtaining a classification accuracy in the training dataset equal to 64%, and a loss equal to 1.05. Moreover, the loss evolution can be seen in Figure 4.15.

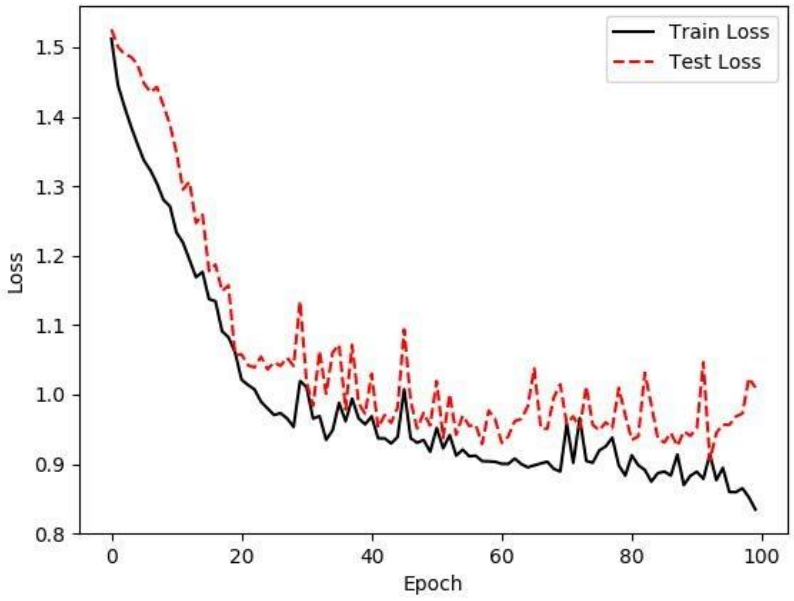

Figure 4.15: Loss evolution per epoch during the training process of the LSTM Network used to classify EMG data. Three classes are classified: Hand Open, Hand Closed and Supination. The loss function is the known as Cross Entropy Loss with Softmax. The graph includes the evolution of the loss of the training and test datasets.

Finally, the results obtained after training a MLP with the same dataset, and using FFT and PCA for feature extraction can be seen in Figures 4.16 and 4.17. Although the network was trained for 14 epochs, the results are those obtained at the 8th epoch. As this is the point in which a higher accuracy in the validation dataset is obtained. For this case the classification accuracy achieved in the training dataset is 82.9 %.

Figure 4.16: Confusion Matrixes which result from the classification of EMG data using a MLP. FFT and PCA are used for feature extraction. Three classes are classified: Hand Open, Hand Closed and Supination. There is a Confusion Matrix per dataset (Training, Validation and Testing) and another one with the three types of datasets combined. In the diagonal of the matrix the percentage of the values correctly classified are displayed and out of the diagonal the percentage of misclassified values. The squares that are not red or green show overall accuracies.

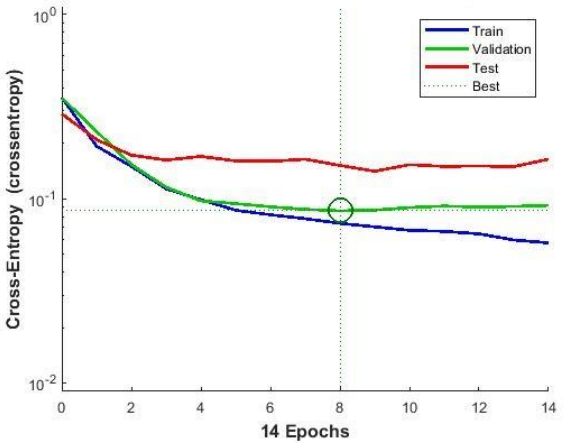

Figure 4.17: Loss evolution for the Training, Validation and Test datasets during the training of a MLP in order to classify EMG data corresponding to three hand movements: Hand Opened, Hand Closed and Supination. FFT and PCA are used for feature extraction. The 8th epoch appears marked as it is the point in which a higher accuracy in the validation dataset is achieved.

4.3.2 PERFORMANCE OF A LSTM NETWORK WHEN CLASSIFYING EEG SIGNALS

The best accuracy achieved by the LSTM Network when classifying the EEG test dataset was 92 %. In Figures 4.18 and 4.19 the evolution of the loss function and accuracy during the training process is displayed.

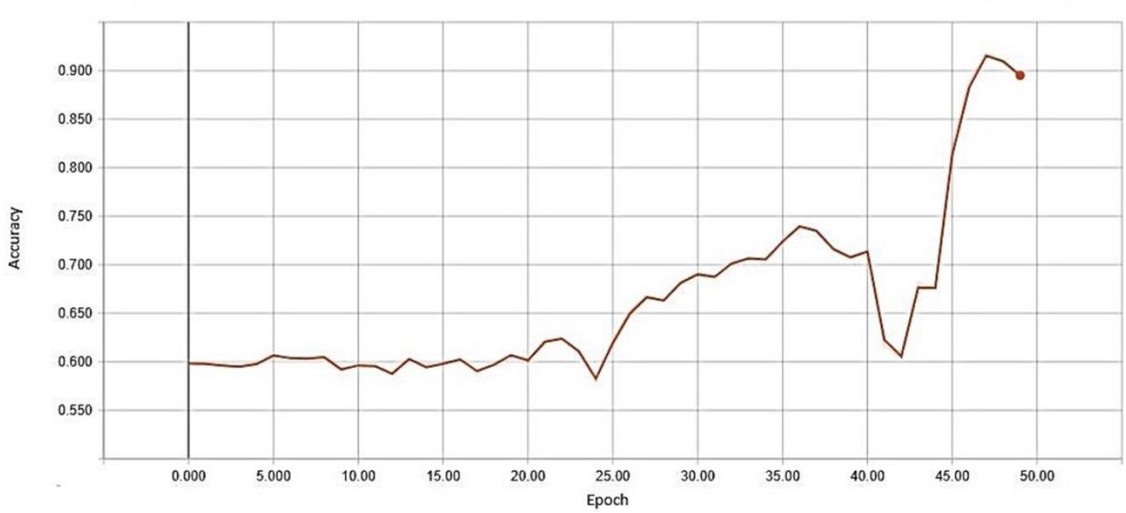

Figure 4.18: Evolution of the accuracy of the test dataset during the training process of the LSTM Network used to classify EEG data into visual or audio stimuli.

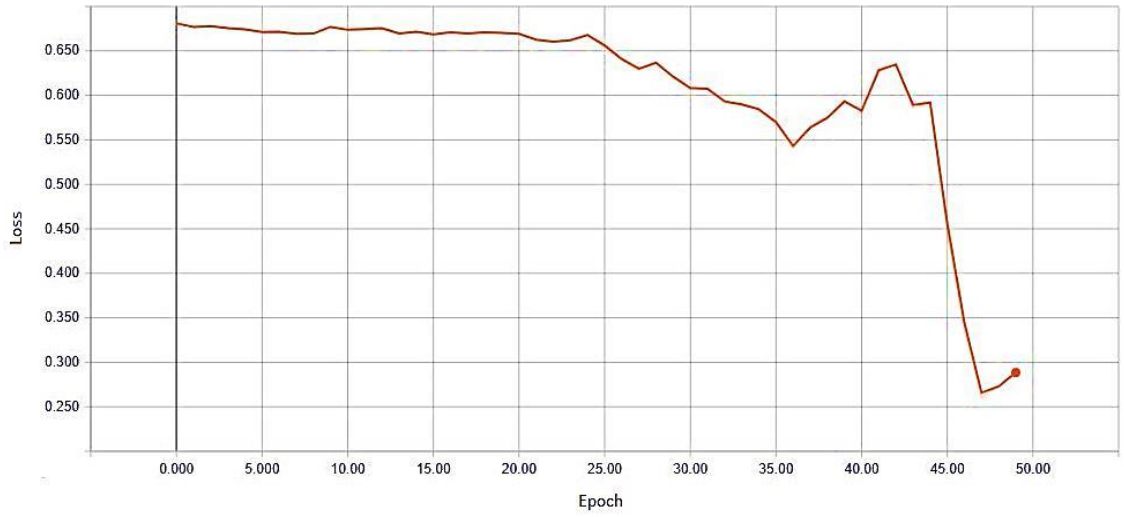

Figure 4.19: Evolution of the loss of the test dataset during the training process of the LSTM Network used to classify EEG data into visual or audio stimuli. The loss function is the known as Binary Cross Entropy Loss.

Finally, the results obtained after training a MLP with the same dataset, and using FFT and PCA for feature extraction can be seen in Figures 4.20 and 4.21. Although the network was trained for 18 epochs, the results are those obtained at the 12^{th} epoch. As this is the point in which a higher accuracy in the validation dataset is obtained. For this case the classification accuracy achieved in the training dataset is 82.1 %.

Figure 4.20: Confusion Matrixes which result from the classification of EEG data using a MLP. FFT and PCA are used for feature extraction. The data corresponds to signals generated in the brain when a person is submitted to visual or audio stimuli. There is a Confusion Matrix per dataset (Training, Validation and Testing) and another one with the three types of datasets combined. In the diagonal of the matrix the percentage of the values correctly classified are displayed. Out of the diagonal the percentage of misclassified values is shown. The squares that are not red or green show overall accuracies.

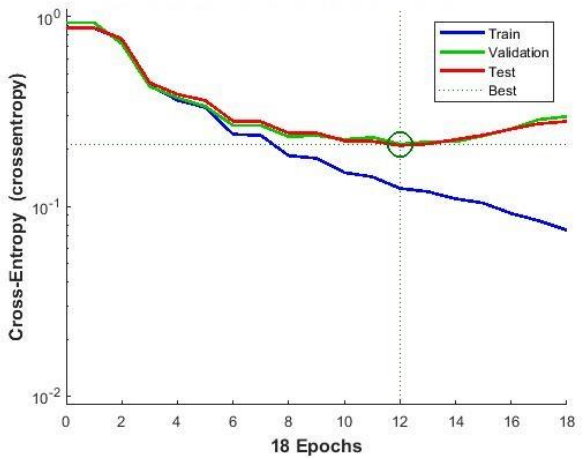

Figure 4.21: Loss evolution for the Training, Validation and Test datasets during the training of a MLP in order to classify EEG data into visual or audio stimuli. FFT and PCA are used for feature extraction. The 12th epoch appears marked as it is the point in which a higher accuracy in the validation dataset is achieved.

Chapter 5

Discussion

5.1 COMMENTS ON THE APPLICATION PERFORMANCE AND TOOLS USED

5.1.1 BLUETOOTH LOW ENERGY DATA TRANSMISON MANAGEMENT

The main conclusion in this section is that using a HandlerThread in order to manage the data transmission between Senso Medical Bio Pot V2 and the mobile device is appropriate. This approach consumes almost no resources allowing a fluid transmission of information. Another approach could have been the use of Services. A Service is a common solution in Android when lasting operations must be carried out without blocking the UI Thread [61]. Nevertheless, this method was discarded because of the fact that starting and stopping the Service as well as sharing information between the Service and the activity Result.java supposed an extra effort in comparison to using directly a HandlerThread.

5.1.2 DEPLOYMENT OF MACHINE LEARNING ALGORITHMS IN AN ANDROID APPLICATION

As it has been pointed out before, the results of the application prototype are quite satisfactory. The latencies during the classification process are low enough to work when analysing biosignals using windows of 200ms. This permits a sampling frequency of 1 kHz, which is definitely sufficient in these cases. Moreover, it is worth noticing that TensorFlow Mobile has been used instead of TensorFlow Lite, which is now TensorFlow´s official framework for deploying machine learning models on mobile and IoT devices. The substitution of TensorFlow Mobile by TensorFlow Lite has officially begun in January 2019. Thus, for this work the old version has still been used.

Nevertheless, TensorFlow Lite has several advantages over TensorFlow Mobile that could improve the application and solve some of its negative points. For example, it was pointed out that it took some time for the application to start the classification process. This is solved in TensorFlow Lite thanks to its new mobile-optimized interpreter. It makes use of static graph ordering and a custom (less-dynamic) memory allocator, reducing the load, initialization, and execution latency. The fact that the execution latency is also lower in TensorFlow Lite could help to ensure that no lag problems take place during the classification process. Moreover, TensorFlow Lite allows using hardware acceleration via the Android Neural Networks API already mentioned and available on Android 8.1 (API level 27) and higher [62].

On the other hand, TensorFlow Lite uses .tflite files for the model instead of .pb. This means that for the same model, the application will occupy less storage in the device.

Moreover, in the application the model used to classify the signals have been a LSTM Network. This has been done because this type of Neural Network has already been used successfully when classifying the biosignals that were interesting for this project: EEG and EMG [63, 64]. Nevertheless it does not mean that other algorithms could not be used. The problem here is consequence of the limited number of operations found when using TensorFlow Mobile and Lite as all the operations that can be found in TensorFlow are not available yet. The other option would have been implementing the model directly in Java, inside the application classes. Nevertheless, the latency in this case would have made the application useless for a real-time use. Thus TensorFlow's framework was the only option left. It is worth noticing that in this work it was tried to use TensorFlow Lite with custom operations, but this option has not been supported until March 2019, being the documentation until now scarce. For example, at the beginning of this project the first idea was using some feature extraction algorithm that generated the input for a classifier like a Support Vector Machine or an Artificial Neural Network. Nevertheless the feature extraction algorithm that was going to be used was the Fast Fourier Transform which is not supported in both TensorFlow Mobile and Lite. For the time being this is still not supported in TensorFlow Lite but thanks to the possibility of implementing custom operations it will be possible to use it in the future. Another way of solving this problem is using TensorFlow Lite builtin operator, which consists in building a TensorFlow Lite model that has been converted and that included TensorFlow operations that are not supported. This is done using bazel in order to create a new Android AAR target for the application. This method is also easier than the previous one, but it is still in an experimental phase, and as a consequence TensorFlow team discourages people from using it. In the close future it could suppose a huge step forward when deploying machine learning models on mobile devices.

5.2 COMMENTS ON THE USE OF SENSO MEDICAL BIO POT V2 AS A WEARABLE DEVICE FOR BIOPOTENTIALS RECORDING

Electronic skin based devices are a promising new technology that has already showed its potential for recording EMG, EEG and ECG signals.

In some studies they have already been used to classify facial expressions [21]. In this case, the Signal to Noise Ratio values achieved by the printed electrodes were similar to the ones of gelled electrodes. Moreover, an eight electrode integrated array achieved results similar to the ones obtained with a five individually wired gelled electrodes [20, 21]. Unfortunately, this has not been achieved with Senso Medical Bio Pot V2.

Accuracies above 90 % have been achieved when employing MLP to classify EMG signals [14]. Using similar methods it has not been possible to obtain accuracies above 50% when classifying the EMG signals recorded by Bio Pot Sensor Medical V2. The reason for this is that the information loss problems that the device suffers make it impossible to reconstruct the signal properly. Therefore, SensoMedical BioPot V2 does not fulfil the necessary requirements to be used as a wearable device.

5.3 COMMENTS ON THE USE OF LSTM NETWORKS FOR CLASSIFYING EMG SIGNALS

In this project, the accuracy achieved when classifying EMG signals using the LSTM Network designed has been below 70 %. It must be taken into account that in this case the architecture was optimized in order to be deployed on a mobile device making use of TensorFlow Mobile, as this was the main objective of the project. Achieving a high accuracy is overall important but was not considered a priority.

When classifying EMG signals, accuracies up to 97% have been achieved [14]. In this case, the approach was using some feature extraction method in addition to a classifier like MLP, SVM or Linear Discriminator Analysis (LDA). Moreover, in these cases it was seen that the feature extraction method used had more influence than the classifier itself. For example, it was more interesting to apply the Autoregressive Coefficients method than FFT [14]. This is in accordance with the results obtained in other projects, in which the FFT and PCA algorithms were used to extract the main features of the EMG data in order to train a MLP. In this case accuracies around 84 % were reported [64]. These results are similar to the ones obtained in the current project when training a MLP with the same dataset employed to train the LSTM Network. Here, using the FFT and PCA algorithms for feature extraction, a classification accuracy equal to 82.9 % is achieved. As it is higher than the one obtained by the LSTM Network, it would be relevant for the future work to implement some feature extraction algorithm using TensorFlow Lite builtin or custom operators.

Furthermore, LSTM Networks and other Deep Learning methods have also been used in a successful way when trying to classify EMG signals [63]. In this case the hyperparameters have been optimized and other types of filtering have been used. Thus, in order to improve the results obtained with the current architecture, better hyperparameters should be found.

5.4 COMMENTS ON THE USE OF LSTM NETWORKS FOR CLASSIFYING EEG SIGNALS

The accuracy achieved when classifying EEG signals related to visual and audio stimuli using the LSTM Network designed in this work has been equal to 92%. It must be taken into account that the architecture was optimized in order to be deployed on a mobile device making use of TensorFlow Mobile, as this was the main objective of the project.

In Chapter 3, it was mentioned that in a different work the same dataset was used when training a LSTM Network that used Dropout layers for regularization. In this case the accuracy was a bit lower: 85%. Thus, it seems interesting to employ L2 regularization instead of Dropout layers. Nevertheless, the dropout value was 0.5, which is quiet high, at the same time that a low λ value is used in this project. On the other hand, training the model using Dropout layers is less computationally expensive than using L2 regularization.

The use of LSTM Networks when classifying EEG data is a relatively new approach. Nevertheless, it has been used in other problems apart from classifying Audio/Visual stimulus. Thus, accuracies of 86-89 % have been achieved when differentiating between 4 types of emotions [65]. In the case of movement classification, accuracies up to 90% have

been registered [66]. In all these cases the results are better than those obtained when using more traditional classifiers. This has been demonstrated in this project by training a MLP with the same dataset that was used to train the LSTM Network. In the first case, the classification accuracy achieved was 82.1 %, lower than the 92 % achieved in the second one.

Nevertheless, in all these works the headset used for recording the signal have several channels (32 or 64), which makes them difficult to wear. Thus, it is necessary to try to replicate the same results using wearable devices with fewer channels as it is the case of Bio Pot Senso Medical V2. Only this way portable BCI will begin to be used in daily activities.

5.5 CRITICAL EVALUATION AND LIMITATIONS OF THE STUDY

Wearable technology is becoming more present in our society nowadays. Nevertheless, in order to spread its use, it is required that interfaces for mobile devices that deploy data mining and processing algorithms are created with real-time capabilities in mind. New wearable devices to record biopotentials are being designed, and are specially relevant to healthcare due to their large diagnostic and monitoring potential. Thus, this project is intended to answer the question about the feasibility of devising a mobile interface able to provide biofeedback based on the biopotentials recorded by wearable devices.

In order to answer the question formulated above, the methods developed in this project are validated with reliability of the feedback as one of the key performance indicators. In the case of performance of the application developed in the project, a specific tool is used for this purpose: Android Profiler. It allows to measure variables as for example: latencies, use of memory or energy consumption. The assumptions on the potential of the application for receiving, processing and interpreting recorded biosignals, while providing real-time feedback to the user in brain-computer interface type of applications, are based on the good values of the measured variables.

The validity of using Senso Medical Bio Pot V2 for EMG, EEG, and ECG recording, has been tested using metrics as for example, a qualitative signal-to-noise ratio of the recorded signal, and the presence of information losses during the recording process. In the project, as the device does not record high quality signals and losses information during the recording process, its use as a wearable device for long-term biosignals monitoring has been discarded.

The project has some limitations as a consequence of using cutting-edge technology that is in the first stages of its development process. For example, the fact that Senso Medical Bio Pot V2 has not been as reliable as expected when recording biopotentials, has limited the possibility of carrying out a more complete real-time testing performance of the application designed. Moreover, the framework used to deploy machine learning algorithms on Android is TensorFlow Mobile, which has already been substituted by a new and more promising one known as TensorFlow Lite. Nevertheless, due to the fact that TensorFlow Lite is still not fully functional, it has not been possible to use it in this project. Therefore, in the future all the implementation carried out in TensorFlow Mobile will have to be migrated to TensorFlow Lite in order to avoid obsolescence problems. In addition to this, other

limitations found when carrying out this project have been related to constraints in the available computer power as well as lack of hardware alternatives to Senso Medical Bio Pot V2.

5.6 ETHICAL, SUSTAINABLE, AND SOCIAL ASPECTS

What makes Wearable Technology so promising is its capacity to improve the user´s quality of life. People can benefit from the application of wearable technology in areas as diverse as sport performance, smart cities or education. Nevertheless, it is in healthcare where this technology is expected to have a bigger impact. For example, Wearable Technology can impulse people to have a more active and healthy lifestyles. On the other hand, the doctors can use wearable devices to monitor the patients in real-time during their daily life. Thus, the symptoms of patients can be determined more accurately, making it possible to prescribe more precise and personalized treatments. Finally, people will not have to go to the hospital anymore for rehabilitation treatments. Thanks to the use of wearable devices it is possible to carry out the rehabilitation from home, saving this way money and time.

Nevertheless, all these positive societal impacts come with some ethical challenges. As it has been mentioned in Chapter 1, the use of wearable devices will only be spread if interfaces for mobile devices that deploy data mining and processing algorithms are created in order to give real-time feedback to the user. Nevertheless, these systems must guarantee the preservation of the user´s privacy, at the same time that a personalized service is provided. This is an essential aspect due to the fact that in many cases the data treated can be very personal, as for example emotions, banking information, genomic data sequences, etc.

On the other hand, these data mining algorithms must be trained. Thus, similar problems to the ones mentioned above appear, as data must be specially recorded for this. Therefore, the privacy of the volunteers that allow their data to be used for training these algorithms must be ensured, at the same time that their safety is guaranteed during the recording process.

Finally, due to the fact that natural resources are not limitless, sustainability is a critical point when developing a new technology. In the case of wearable devices like the Senso Medical Bio Pot V2 that uses a component that needs to be changed frequently, in this case the electrodes, using components that are biodegradable or recyclable is of the utmost importance. Nevertheless, it should not be a problem in the future as new electronics fabrication technologies have already opened the door to biodegradable electronic skin type electrodes.

Chapter 6

Conclusions and Future Work

This project has allowed to study the feasibility of using wearable devices based on electronic skin for continuous biopotential recording, while supporting them with a mobile phone application able to receive, process and analyse the recorded biosignals in order to deliver useful feedback to the user in real-time.

Several conclusions are derived from this work. Firstly, the device Senso Medical Bio Pot V2 is not suitable for its use as a wearable device for biosignal recording. Secondly, the application designed and simulated offline achieves good performance. As a consequence, it could be used in the future with a suitable wearable sensor to process and interpret recorded biosignals, at the same time that real-time feedback is provided to the user. Nevertheless, this will only be possible if the next tasks that are carried out in the future:

- Use a wearable device based on electronic skin type electrodes able to record a signal with a higher quality, and test the application prototype real-time performance with the signals recorded by this device.
- Deploy new machine learning algorithms making use of TensorFlow Lite custom operations or builtin operations. Use both a feature extraction algorithm and a machine learning traditional classifier.
- Deploy other Deep Learning algorithms like Convolutional Neural Networks or Bidirectional LSTM Network to classify biosignals.
- Optimize the already used LSTM architectures calculating better hyperparameters.

Bibliography

[1] Liew, C. S., Wah, T. Y., Shuja, J., & Daghighi, B. (2015). Mining personal data using smartphones and wearable devices: A survey. *Sensors*, *15*(2), 4430-4469.

[2] Billinghurst, M., & Starner, T. (1999). Wearable devices: new ways to manage information. *Computer*, *32*(1), 57-64.

[3] Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012, December). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living* (pp. 216-223). Springer, Berlin, Heidelberg.

[4] Mukhopadhyay, S. C. (2015). Wearable sensors for human activity monitoring: A review. *IEEE sensors journal*, *15*(3), 1321-1330.

[5] Li, R. T., Kling, S. R., Salata, M. J., Cupp, S. A., Sheehan, J., & Voos, J. E. (2016). Wearable performance devices in sports medicine. *Sports health*, *8*(1), 74-78.

[6] Guo, F., Li, Y., Kankanhalli, M. S., & Brown, M. S. (2013, October). An evaluation of wearable activity monitoring devices. In *Proceedings of the 1st ACM international workshop on Personal data meets distributed multimedia* (pp. 31-34). ACM.

[7] Schweizer, H. (2014). Smart glasses: technology and applications. *Student report*.

[8] Maskeliunas, R., Damasevicius, R., Martisius, I., & Vasiljevas, M. (2016). Consumer-grade EEG devices: are they usable for control tasks?. *PeerJ*, *4*, e1746.]

[9] Liao, L. D., Chen, C. Y., Wang, I. J., Chen, S. F., Li, S. Y., Chen, B. W., & Lin, C. T. (2012). Gaming control using a wearable and wireless EEG-based brain-computer interface device with novel dry foam-based sensors. *Journal of neuroengineering and rehabilitation*, *9*(1), 5.

[10] Bonato, P. (2005). Advances in wearable technology and applications in physical medicine and rehabilitation.

[11] Bareket, L., Inzelberg, L., Rand, D., David-Pur, M., Rabinovich, D., Brandes, B., & Hanein, Y. (2016). Temporary-tattoo for long-term high fidelity biopotential recordings. *Scientific reports*, *6*, 25727.

[12] Inzelberg, L., Pur, M. D., Schlisske, S., Rödlmeier, T., Granoviter, O., Rand, D., & Hanein, Y. (2018). Printed facial skin electrodes as sensors of emotional affect. *Flexible and Printed Electronics*, *3*(4), 045001.

[13] Soh, P. J., Vandenbosch, G. A., Mercuri, M., & Schreurs, D. M. P. (2015). Wearable wireless health monitoring: Current developments, challenges, and future trends. *IEEE Microwave Magazine*, *16*(4), 55-70.

[14] Al-Timemy, A. H., Bugmann, G., Escudero, J., & Outram, N. (2013). Classification of finger movements for the dexterous hand prosthesis control with surface electromyography.*IEEE Journal of Biomedical and Health Informatics*,*17* (3), 608-618.

[15] Nicolas-Alonso, L. F., & Gomez-Gil, J. (2012). Brain computer interfaces, a review. *Sensors*, *12*(2), 1211-1279.

[16] Maskeliunas, R., Damasevicius, R., Martisius, I., & Vasiljevas, M. (2016). Consumer-grade EEG devices: are they usable for control tasks?. *PeerJ*, *4*, e1746.

[17] Komninos, A., & Dunlop, M. (2014). Text input on a smart watch. *IEEE Pervasive Computing*, *13*(4), 50-58.

[18] Guo, F., Li, Y., Kankanhalli, M. S., & Brown, M. S. (2013, October). An evaluation of wearable activity monitoring devices. In *Proceedings of the 1st ACM international workshop on Personal data meets distributed multimedia* (pp. 31-34). ACM.

[19] Schweizer, H. (2014). Smart glasses: technology and applications. *Student report*.

[20] Bareket, L., Inzelberg, L., Rand, D., David-Pur, M., Rabinovich, D., Brandes, B., & Hanein, Y. (2016). Temporary-tattoo for long-term high fidelity biopotential recordings. *Scientific reports*, *6*, 25727.

[21] Inzelberg, L., Pur, M. D., Schlisske, S., Rödlmeier, T., Granoviter, O., Rand, D., & Hanein, Y. (2018). Printed facial skin electrodes as sensors of emotional affect. *Flexible and Printed Electronics*, *3*(4), 045001.

[22] Haartsen, J. C. (2000). The Bluetooth radio system. *IEEE personal communications*, *7*(1), 28-36.

[23] Haartsen, J. C., & Mattisson, S. (2000). Bluetooth-a new low-power radio interface providing short-range connectivity. *Proceedings of the IEEE*, *88*(10), 1651-1661.

[24] Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, *12*(9), 11734-11753.

[25] ANDROID DEVELOPERS: https://developer.android.com/guide/topics/connectivity/bluetooth-le (Online February 2019).

[26] Townsend, K., Cufí, C., & Davidson, R. (2014). *Getting started with Bluetooth low energy: tools and techniques for low-power networking.* " O'Reilly Media, Inc.".

[27] SENSO MEDICAL: http://www.sensomedical.com/assets/biopot2brochure.pdf (Online February 2019).

BIBLIOGRAPHY

[28] Senso Medical. (2018). *Senso Medical Bio Pot V2: Bluetooth Characteristics WEEG*. Nazareth.

[29] Islam, N., & Want, R. (2014). Smartphones: Past, present, and future. *IEEE Pervasive Computing*, (4), 89-92.

[30] Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C., & Zhang, J. C. (2014). What will 5G be?. *IEEE Journal on selected areas in communications*, *32*(6), 1065-1082.

[31] Kaku, M. (2012). *Physics of the future: How science will shape human destiny and our daily lives by the year 2100*. Anchor.

[32] ANDROID DEVELOPERS: https://developer.android.com/ndk/guides/neuralnetworks/?hl=es-419 (Online January 2019)

[33] Gandhewar, N., & Sheikh, R. (2010). Google Android: An emerging software platform for mobile devices. *International Journal on Computer Science and Engineering*, *1*(1), 12-17.

[34] Butler, M. (2011). Android: Changing the mobile landscape. *IEEE Pervasive Computing*, *10*(1), 4-7.

[35] MIT: https://stuff.mit.edu/afs/sipb/project/android/docs/guide/components/fundamentals.html (Online February 2019).

[36] ANDROID DEVELOPERS: https://developer.android.com/about/versions/oreo/android-8.1?hl=en (Online February 2019).

[37] CONDENTRICK: https://codentrick.com/android-view-lifecycle/ (Online February 2019).

[38] ANDROID DEVELOPERS: https://developer.android.com/guide/components/activities/activity-lifecycle (Online February 2019).

[39] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., & Yger, F. (2018). A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, *15*(3).

[40] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* . Springer.

[41] Marsland, S. (2011). *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.[42] García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, *13*(10), 959.

[43] Zhu, X. (2006). Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, *2*(3), 4.

[44] Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

[45] Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.

[46] Nayak, S., Soni, M. K., & Bansal, D. (2012). Filtering techniques for ECG signal processing. *International Journal of Research in Engineering & Applied Sciences*, *2*(2), 671-679.

[47] Van Drongelen, W. (2018). *Signal processing for neuroscientists*. Academic press.

[48] ANDROID DEVELOPERS: https://developer.android.com/topic/performance/threads (Online February 2019).

[49] Fürst, J., Chen, K., Kim, H. S., & Bonnet, P. (2018, April). Evaluating Bluetooth low energy for IoT. In *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)* (pp. 1-6). IEEE.

[50] ANDROID DEVELOPERS: https://developer.android.com/reference/android/os/Handler (Online February 2019).

[51] ANDROID DEVELOPERS: https://developer.android.com/reference/android/os/MessageQueue.html (Online February 2019).

[52] HACKING FOREVER: http://ashu-hackingforever.blogspot.com/2015/04/sending-message-from-one-worker-thread.html (Online February 2019).

[53] MIT: https://stuff.mit.edu/afs/sipb/project/android/docs/sdk/index.html (Online March 2019).

[54] ANDROID DEVELOPERS: https://developer.android.com/ndk/guides (Online March 2019).

[55] GITHUB: https://github.com/biopatrec/biopatrec/tree/Data_Repository/10mov4chForearmUntargetedd (Online March 2019).

[56] Ortiz-Catalan, M., Brånemark, R., & Håkansson, B. (2013). BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms. *Source code for biology and medicine*, *8*(1), 11.

[57] GITHUB : https://github.com/Cerebro409/EEG-Classification-Using-Recurrent-Neural-Network (Online March 2019).

[58] Ng, J., & Goldberger, J. J. (2007). Understanding and interpreting dominant frequency analysis of AF electrograms. *Journal of cardiovascular electrophysiology*, *18*(6), 680-685.

BIBLIOGRAPHY

[59] ANDROID DEVELOPERS: https://developer.android.com/studio/profile/memory-profiler.html (April 2019).

[60] Herman, P., Prasad, G., McGinnity, T. M., & Coyle, D. (2008). Comparative analysis of spectral approaches to feature extraction for EEG-based motor imagery classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *16*(4), 317-326.

[61] ANDROID DEVELOPERS: https://developer.android.com/reference/android/app/Service (Online March 2019).

[62] TENSORFLOW: https://www.tensorflow.org/lite/guide (Online April 2019).

[63] Laezza, R. (2018) *Deep Neural Networks for Myoelectric Pattern Recognition. An Implementation for Multifunctional Control.* Chalmers University of Technology, Göteborg, Sweden.

[64] Güler, N. F., & Koçer, S. (2005). Classification of EMG signals using PCA and FFT. *Journal of Medical Systems*, *29*(3), 241-250.

[65] Alhagry, S., Fahmy, A. A., & El-Khoribi, R. A. (2017). Emotion recognition based on EEG using LSTM recurrent neural network. *Emotion*, *8*(10), 355-358.

[66] Chen, W., Wang, S., Zhang, X., Yao, L., Yue, L., Qian, B., & Li, X. (2018, May). EEG-based motion intention recognition via multi-task RNNs. In *Proceedings of the 2018 SIAM International Conference on Data Mining* (pp. 279-287). Society for Industrial and Applied Mathematics.

[67] Woodings, R. W., Joos, D. D., Clifton, T., & Knutson, C. D. (2002, March). Rapid heterogeneous ad hoc connection establishment: accelerating Bluetooth inquiry using IrDA.

[68] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

[69] BLUETOOTH: https://www.bluetooth.com/specifications/profiles-overview (Online February 2019).

[70] INTANTECH: http://intantech.com/files/Intan_RHD2000_series_datasheet.pdf (Online February 2019).

Appendix A

Long Short-Term Memory Neural Network

A Recurrent Neural Network (RNN) is a Machine Learning algorithm specially aimed to work with sequential data. It is able to learn from inputs that are sequences and has showed its value in applications like: image captioning, language translation, and handwriting recognition. Nevertheless, a RNN has problems when learning long-term dependencies. In order to overcome this problem, a different architecture of RNN known as Long Short-Term Memory (LSTM) Network, has been designed. Its structure resembles a chain, in which the repeated module is a memory cell like the one that can be seen in Figure A.1. Every input cell will receive as input: the input vector at the current time step and the hidden vector at the previous time step [70].

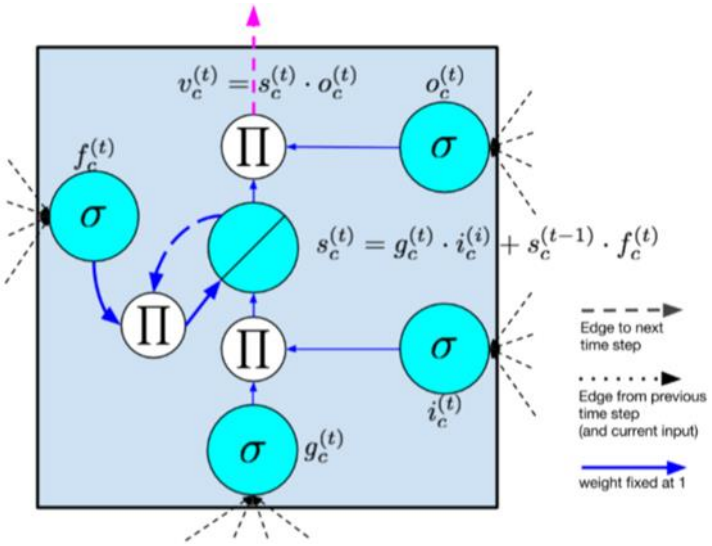

Figure A.1: Architecture of a LSTM memory cell [70].

A LSTM Network cell has the next key elements [70]:

- Input node: g_c at Figure A.1. It applies a sigmoid (old approach) or a tanh (modern approach) activation function to the concatenation of the input vector at the current time step and the hidden vector at the previous time step.

- Input gate: A gate in a LSTM Network is a sigmoid layer used to regulate the transfer of information through a node. If the value of the gate is zero, when it multiplies the node, the flow from it is stopped. In the case of the input gate, the input will be a concatenation of the input vector at the current time step and the hidden vector at the previous time step. The value of the input gate multiplies the value of the input node.

- Internal state: A node that is used to remember the long short term dependencies.

APPENDIX A

- Forget gate: It determines which information is going to be removed from the internal state and which is going to be kept.
- Output gate: The value produced by a memory cell is the result of multiplying the internal state s_c by the output gate o_c.

Appendix B

Bluetooth Technology

Bluetooth is a radio technology suitable for wirelessly communication when small amounts of data are transmitted in a short range [22].

The Bluetooth networks are known as piconets and are composed by a master and a certain number of slaves (maximum 7). The master will receive and send information to any of the slaves, meanwhile the slaves can only send and receive information from one master. [22, 24].

On the other hand, Bluetooth technology makes use of a public radio spectrum that can be accessed in any part of the world. This radio spectrum is known as the Industrial, Scientific, Medical (ISM)) band. Its frequency corresponds to 2400-2483.5 MHz. On the other hand, there are 40 Radio Frequencies or channels defined in this band. The channels can be advertising channels or data channels. The first ones (3 channels) are used for device discovery, connection establishment, and broadcast transmission. The second ones (37 channels) are used for the communication between the connected devices [22]. Finally, the Bluetooth radio uses a method known as adaptive frequency hopping, in order to hop between channels and select one during a time interval in which a connection takes place. It is used in order to solve interference and wireless propagation issue [22, 24].

The information in a Bluetooth system is divided in packets and only a single packet is sent at a time [22, 24].

The Bluetooth connection process consists in a series of steps: Firstly, the master will try to discover the other devices around. The addresses of the different devices are made known in this process. It is worth pointing out that a Bluetooth device wakes up periodically in order to listen if other devices want to connect. After this, the connection between two devices is stablished and the information transmission process begins. This connection process is much shorter than the discovering step [22, 67]. The data transmission process can be carried out actively, or in low-power modes. In these cases, the slaves are in sleep mode but wake up periodically in order to listen for possible packets sent by the master [24]

A Bluetooth device can have different profiles, depending on the activity it is used for. Some examples are: Basic Printing Profile (BPP), Device ID Profile (DI), Advanced Audio Distribution Profile (A2DP) or Headset Profile (HSP) among others [69].

The Bluetooth profile will determine some elements of the protocol stack which can be seen in Figure B.1. This protocol stack is what defines all the Bluetooth functionalities. Some protocols like the LMP or the L2CAP are always present, and others vary depending on

the use of the device. On the other hand, the Bluetooth stack can be divided in an Application, a Host and a Controller. The Controller consists in the RF and Baseband blocks and the Host is all those blocks between the Controller blocks and the Application ones. [22, 24].

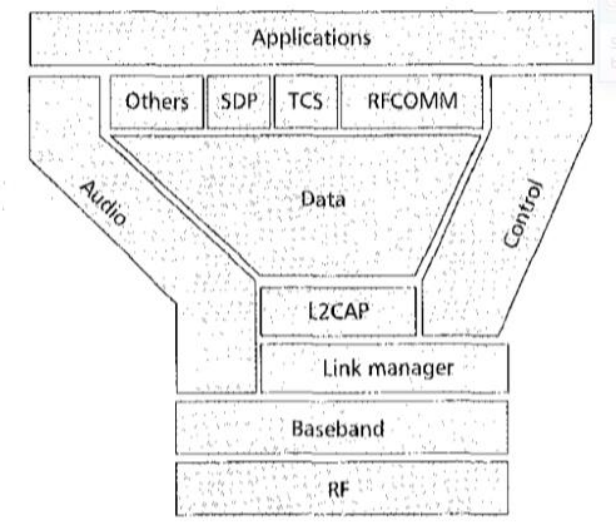

Figure B.1: Bluetooth Protocol Stack [22].

Appendix C

Technical Specifications of Senso Medical Bio Pot V2

As it has been introduced in Chapter 2, Senso Medical Bio Pot V2 has 5 Characteristics.

Characteristic 1 has 4 Bytes, which represent the number of the INTAN register to read and write. Byte 0 and 1 contain the number of the register where the reading process is going to be carried out, and the value that has been read, respectively. On the other hand Byte 2 and 3 contain the number of the register where the writing process is going to be done, and the written value, respectively [28].

Characteristic 2 is used to start or stop the recording of the signal. It has one Byte, being the first bit set to 0x1 in order to begin the recording and to 0x0 to stop it.

On the other hand, the device has a specific circuitry to generate a current that directed to the electrode can be used to test its impedance. Thus, if this is going to be done, the bit of the Zcheck DAC power register must be set to 1 by changing the value of Characteristic 3. If not, it will be set to 0 [28, 70]. It is worth noticing that the impedance scale and frequency are set making use of Characteristic 5. The value of the impedance will be [28, 70]:

$$i = 2 \cdot \pi \cdot C_s \cdot 0.6125$$

The value of Characteristic 4 is the most interesting one because of the fact that its value corresponds to the measurement of the device. Thus, when the acquisition is started the value of Characteristic 4 changes and becomes the value of the measured signal. This value has the first 4 bytes assigned as time stamp. The rest of the bytes are the recording in the respective channels, beginning with the channel 0 and finishing with the channel 7. Each channel has two bytes assigned, represented in hexadecimal (signed) [28].

Finally, Characteristic 5 is used to set the parameters for the measurement process. Writing in Byte 2 and 3, the preamplifiers can be set on or off. Writing in Byte 5 the value 0, 1, 2, 3, 4, 5 or 6, a low-pass filter with f_c equal to 100Hz, 150 Hz, 250Hz, 300Hz, 500Hz or 750Hz is stablished. Similarly a high-pass filter with f_c equal to 0.1 Hz, 0.25 Hz, 0.3 Hz, 0.5 Hz, 0.75 Hz, 1 Hz, 1.5 Hz, 2 Hz or 2.5 Hz can be used by writing the values 0, 1, 2, 3, 4, 5, 6, 7 or 8 in Byte 5. The sampling rate per Channel in Hz can be specified in Byte 6 and 7. Finally, the impedance frequency and the impedance scale are set using Byte 8 and 9 [28].

Appendix D

Android Software Architecture

As it can be seen in Figure D.1, the Android software stack consists in a variety of layers. Android applications are written in Java, Kotlin, C, C++, and XML and run on a virtual machine (VM) environment. The VM used in Android is known as Dalvik [33, 34].

Android Architecture consists in the following layers: Applications, Application framework, Libraries, Android runtime and Linux kernel [34]. At the top, it is the Applications layer, which contains several basic applications as for example: email, SMS program, calendar, maps, browser, contacts, etc. It is worth noticing that multiple applications can be run at the same time, making it possible hearing music meanwhile reading online news.

The Application layer is followed by the Application Framework. This layer is in charge of creating a standard structure for the application depending on the operating system used. Moreover, it can implement functions making use of those already existing applications. After the Application Framework, the Libraries layer can be found. It has two components, being all of them written in C/C++. The first libraries are used thanks to a Java interface. Some examples are the SQL database or the web browser engine WebKit. The second component is the Android Runtime which has a series of basic libraries that contain many of the functionalities that can be found in the core libraries of the Java programming language. Android Runtime also contains the Dalvik Virtual Machine [33]. In general, every application in Android has its own process as an instance of this VM. This means that every application has its own Linux user ID, which is only known by the system, although it can be arranged so that in very particular cases two applications share an ID and process in the VM [33, 35]. The system sets permissions to its components so that only specific IDs can access them. The permissions are approved by the user when the application is installed. Thus, each application is able to access only the necessary components to do its job. As a consequence, a very safe environment is created [35], a clear advantage over other softwares as for example iOS [33, 34]. In the particular case of two applications sharing the same ID, each application can access the files of each other [35]. The use of Dalvik VM is justified because it compiles portable Java-style byte code files after turning them into .dex format. This result in using minimal memory and as a consequence a much better performance [33].

Finally, at the bottom it is the Linux Kernel. It is used by Android for security, memory management, process management, network stack, and driver model. Moreover this layer is used as an abstraction layer between the hardware and the rest of the software stack [33].

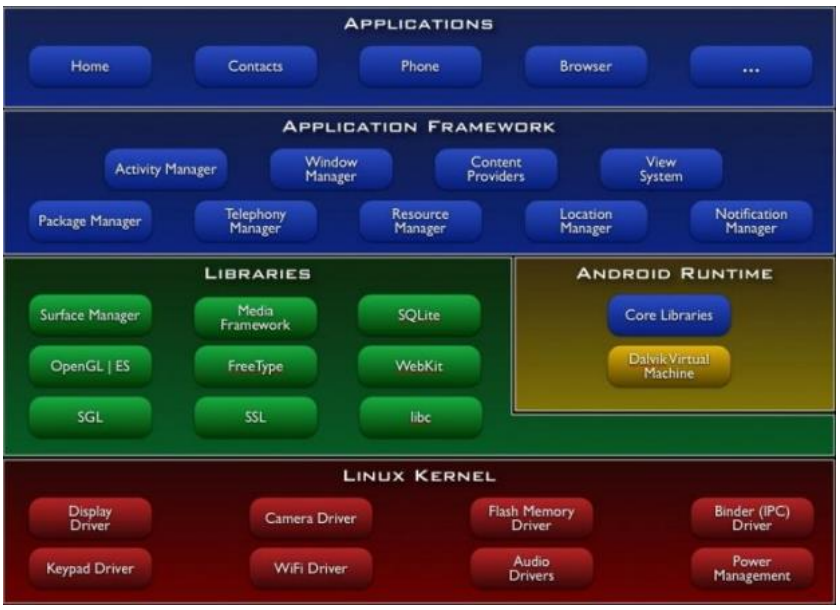

Figure D.1: Android Architecture [34].

Appendix E

Android Activity Life-Cycle

As it has been pointed out in Chapter 2, the Activity class has a series of callbacks that allow an activity to know that its state has changed. It is crucial to implement the right functionalities in them so that the application fulfils the user´s expectations [38].

In Figure E.1, the life-cycle of an application can be seen. Once the Activity is being launched the OnCreate() method is called. Here, initializations that should happen only once for the entire life of the activity are carried out, as for example the creation of the UI. After this, the method onStart() is called, being its main function to make the activity visible to the user and as a consequence the app initializes the code that maintains the UI [38].

Once the onStart() method is done, the activity goes into the Resumed state. After this, the activity goes to the foreground and the onResume() method is called. At this moment, the user interacts with the app until the moment in which the user leaves the activity and the onPause() method is called. An activity can go back to the Resumed state from the Paused state by calling again the method onResume(). Therefore, there is an important relation between the code that is implemented in these two methods. It will be necessary to initialize in the onResume() method those components that were erased in onPause(). Moreover, the onPause() method stops those operations that are not needed in the Paused state and prepare them to be started again. It also stores important data [38].

Once the activity can not be seen by the user, it means that the Stopped state has been reached, and the system invokes the onStop() method. This is used if a new Activity is in the If an activity is not destroyed after reaching the Stopped state it will go back to the Started screen now or when the activity is going to finish. It is worth noticing that if there are memory problems this method may not be called and the activity will be destroyed directly. state by calling the method onRestart() [38].

Finally, the onDestroy() method is called before the activity is destroyed. Like in the case of onPause() it may not be called if the activity is destroyed directly [38].

An activity is going to be destroyed if it is finished or there is a configuration change like the device rotation that leads to having a change in the way the content is displayed [38].

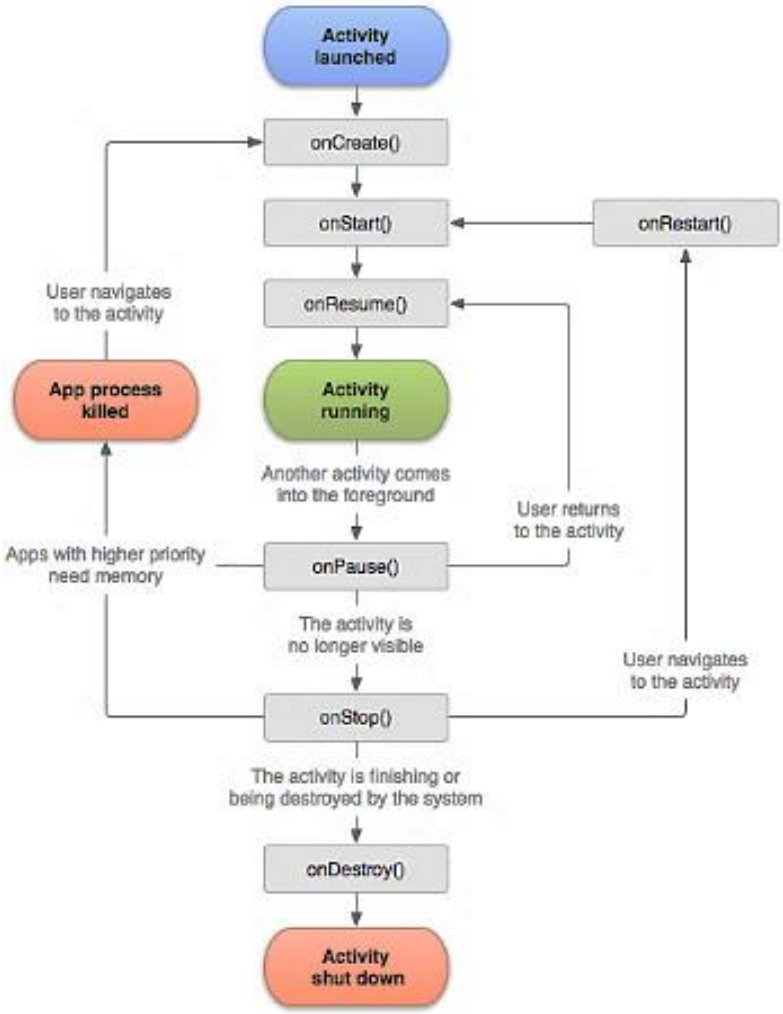

Figure E.1: Life-cycle of an Android activity [38].

Appendix F

Implementation

F.1: Deployment of Bluetooth Low Energy on Android.

The code corresponding to this Appendix can be found in:
https://github.com/Dieguli/Appendix_E_Masters_Thesis/tree/master/E1

F.2: Use of TensorFlow Mobile Framework in Android.

The code corresponding to this Appendix can be found in:
https://github.com/Dieguli/Appendix_E_Masters_Thesis/tree/master/E2

F.3: Implementation of a LSTM Network in TensorFlow for EMG Signals Classification and Deployment on an Android Application.

The code corresponding to this Appendix can be found in:
https://github.com/Dieguli/Appendix_E_Masters_Thesis/tree/master/E3

F.4: Implementation of a LSTM Network in Keras for EEG Signals Classification and Deployment on an Android Application.

The code corresponding to this Appendix can be found in:
https://github.com/Dieguli/Appendix_E_Masters_Thesis/tree/master/E4