UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Android Implementation of a Visualisation, Sonification and AI-Assisted Interpretation of Neonatal EEG

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Eduard Salgado Burruezo

In partial fulfillment

of the requirements for the degree in

AUDIOVISUAL SYSTEMS ENGINEERING

Advisor: Andriy Temko

Co-advisor: Climent Nadeu Camprubí

Barcelona, May 2018

Abstract

The aim of this project is the implementation of an Android App to help healthcare professionals to check newborn health status by observing neonatal EEG signals, without having extensive training in EEG interpretation.

To satisfy that aim, this project is divided in three blocks: AI-assisted neonatal EEG interpretation, EEG sonification and graphical user interface.

The AI-assisted block has the function to detect neonatal seizures using a fully-convolutional deep neural network using the offline-trained existing model.

The sonification work consisted of the adaptation of a previously developed algorithm, based on the phase vocoder, which was already implemented by another UPC student in the Android environment.

The developed application core provides both sonification and AI detection functionalities, which are integrated in a user friendly graphical user interface.

Resum

L'objectiu d'aquest projecte era la implementació d'una aplicació Android per ajudar a professionals de l'àmbit mèdic a comprovar l'estat de salut de nounats en base a l'observació de l'electroencefalograma (EEG), sense necessitat de tenir molta experiència en neonatologia.

Per tal d'acomplir aquest objectiu, el projecte s'ha dividit en tres blocs: interpretació assistida per IA, sonificació i interfície d'usuari gràfica.

El bloc d'IA s'encarrega de la detecció d'epilèpsies en nadons utilitzant una xarxa neuronal totalment convolucional implementada en Android duent a terme l'adaptació d'un model ja existent programat en Python.

El treball de sonificació de l'EEG ha consistit en l'adaptació d'un algoritme basat en Phase Vocoder realitzat per un altre estudiant de la UPC.

La finalitat de la interfície gràfica és mostrar de forma integrada la informació rebuda de la sonificació i la xarxa neuronal perquè l'usuari pugui interpretar-les amb facilitat, de manera que l'aplicació resulti útil a un gran nombre d'usuaris.

Resumen

El objetivo de este proyecto era la implementación de una aplicación Android para ayudar a profesionales del ámbito médico a comprobar el estado de salud de neonatos en base a la observación del electroencefalograma (EEG), sin necesidad de tener mucha experiencia en el campo de la neonatología.

Para cumplir dicho objetivo, el proyecto se ha dividido en tres bloques: interpretación asistida por IA, sonificación y interfaz de usuario gráfica.

El bloque de IA se encarga de la detección de epilepsias en recién nacidos utilizando una red neuronal totalmente convolucional implementada en Android llevando a cabo la adaptación de un modelo ya existente en Python.

El trabajo de sonificación del EEG ha consistido en la adaptación de un algoritmo basado en Phase Vocoder realizado por otro estudiante de la UPC.

La finalidad de la interfaz gráfica es mostrar de forma integrada la información recibida de la sonificación y la red neuronal para que el usuario pueda interpretarlas con facilidad, de forma que la aplicación resulte útil a un gran número de usuarios.

Acknowledgements

I wish to express my most sincere thanks to Dr. Andriy Temko and Dr. Emmanuel Popovici for the purpose of this project and the opportunity to work in INFANT Research Center and EmbeddedSystems@UCC, University College Cork, Ireland.

I would like to thank to Alison O'Shea, PhD candidate of the Electrical and Electronic Engineering Department in UCC, for her patience and invaluable support in the project. I would also like to thank all students I met in UCC who made the weather in Ireland sunnier than it really was.

Also thanks to Climent Nadeu for being the reason why I am in Ireland and for which I will always be grateful.

Finally, I would like to thank my close friends Ismael Akif, Daniel Moyano and Jae Hyouk Kim for all of their support during my stay in Ireland.

Revision history and approval record

Revision	Date	Purpose
0	26/4/2018	Document creation
1	12/05/2018	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Eduard Salgado Burruezo	edu.salbu@gmail.com
Andriy Temko	aatemko@gmail.com
Emanuel Popovici	e.popovici@ucc.ie
Climent Nadeu Camprubí	climent.nadeu@upc.edu

Written by:		Reviewed and approved by:	
Date	10/05/2018	Date	15/05/2018
Name	Eduard Salgado	Name	Andriy Temko
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Project overview and goals

The project is carried out at Embedded Systems Group of the Department of Electrical and Electronic Engineering and Irish Centre for Fetal and Neonatal Translational Research (INFANT), University College of Cork (UCC).

The purpose of this project is to implement the algorithm for neonatal seizure detection using fully convolutional deep neural networks on an Android platform. The algorithm is based on the study made by O'Shea et al., about detecting seizure using CNNs. This algorithm should be integrated into a frond-end app and run alongside neonatal EEG sonification system. The algorithm must run in real-time on a single channel neonatal EEG and provide an output which indicates the wellbeing of the neonatal brain. The work will be performed under the Welcome Trust funded project: Sound-based Interpretation of Neonatal EEG, led by the PI, Dr. Andriy Temko.

The project main goals are:

a. Implement a fully convolutional neural network on an Android platform and ensure its correct functionality within the algorithm developed for computer use in python language
b. Build a full-featured app called Neurobell App with all three techniques of seizure detection (Visual, CNN and Sonification).
c. Adapt the existing Android sonification algorithm for the actual project.
d. Integrate Bluetooth connectivity of Android tablet and the OpenBCI EEG acquisition board and guarantee real-time streaming of the data.
e. Ensure that the final application corresponds with the customer's requirements.

This project forms the front-end of an INFANT project called Neonatal Brain Stethoscope which is illustrated in Figure 1 and 2. The project aims at creating of an Android app called 'Neurobell App' part of a whole system containing a hand-held device which uses dry electrodes for easy-to-use acquisition of a single channel of neonatal EEG using the acquisition board (OpenBCI) as the hardware. The board sends EEG real-time via Bluetooth to an Android device. The Android device must be able to process the signal and propose three techniques for neonatal EEG interpretation – subjective visual and sonification and objective AI-assisted decision support based on CNN.

Figure 1. Proposed EEG acquisition & interpretation system

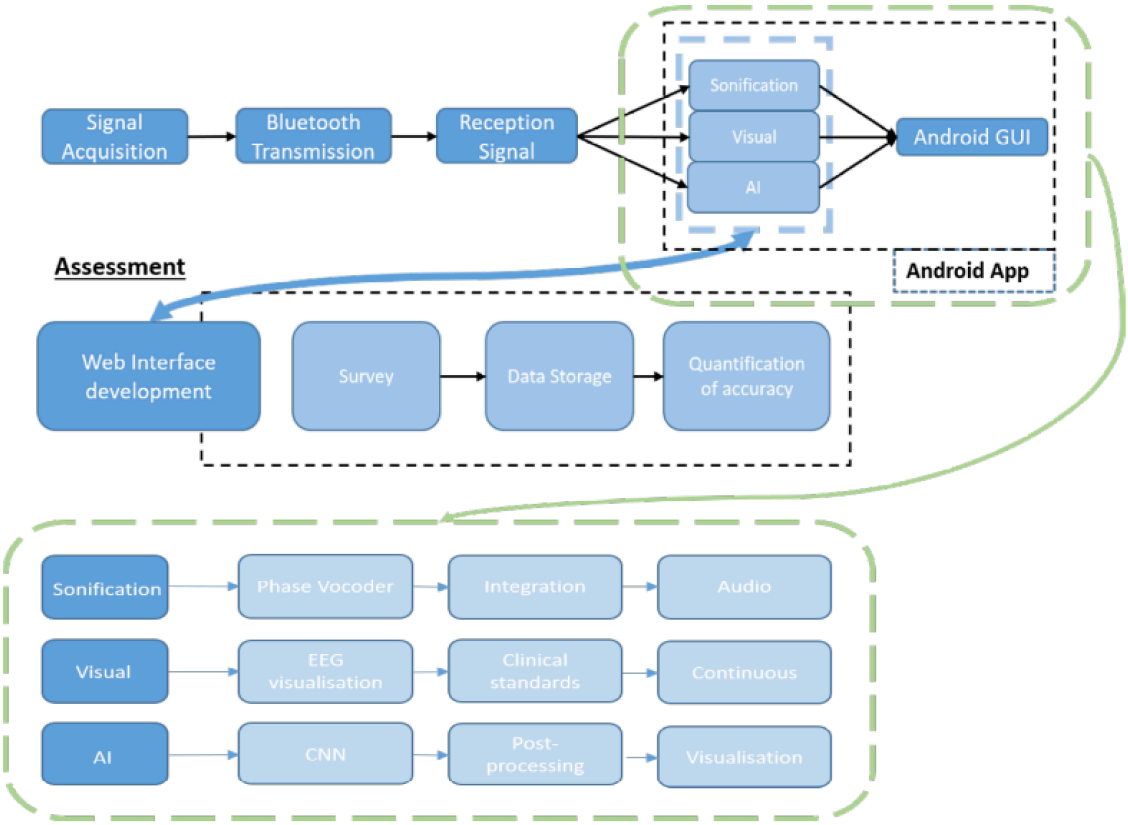

Figure 2. Project context

1.2. Project background

Electroencephalogram (EEG) is used to monitor the brain's electrical activity so that abnormalities such as seizures or fits can be detected and treated. Interpretation of neonatal EEG requires highly trained healthcare professionals and is limited to specialized units and is not available 24/7.

The WT project aims at developing a brain stethoscope device – an alternative method to monitor neonatal brain health which can be easily applied and interpreted by clinicians without neurophysiology training. This system is based on a portable EEG acquisition device coupled with a smartphone/tablet interpretation system. In particular, the acquisition consists of a set of dry electrodes connected to an acquisition device (OpenBCI) to acquire the EEG signal. The interpretation consists of an Android-based EEG sonification combined with the machine learning based health status indicator which will provide decision support during EEG monitoring.

This work concentrated on an Android implementation of machine learning based seizure detection algorithm and creation of an interface to integrate the AI-assisted decision support with other subjective means of neonatal EEG interpretation.

The development of the app has been performed with collaboration of healthcare professionals to accommodate end-user requirements.

1.3. Project requirements and specifications

Project requirements:

- Definition of a scenario regarding when and how machine learning technologies may be used in an Android environment.
- Efficient implementation of a neonatal EEG-based seizure detection algorithm based on Convolutional Neural Networks.
- Build user-friendly interface and integrate sonification and Bluetooth blocks.

Project specifications:

- Achieve the same functionality as the CNN model which has been previously developed in Python, by testing it on a selected data stream and obtaining the same probabilistic output.
- Achieve real-time functionality through efficient implementation and multithreading.
- Design and implement optimal ways to visualise the output of the AI-block such as probabilistic output as a trace and/or meter and/or a traffic light.
- To achieve minimal power consumption to allow long monitoring functionality.
- Review mode with offline sonification and signal visualisation.
- User-interface to provide options and control over signal acquisition, recording, and create a health informatics system to manage patient metadata.

1.4. Work plan

The project has followed the work plan designed at the start of the project with some unavoidable differences between the plan and the actual development. The differences are explained in Section 1.5.

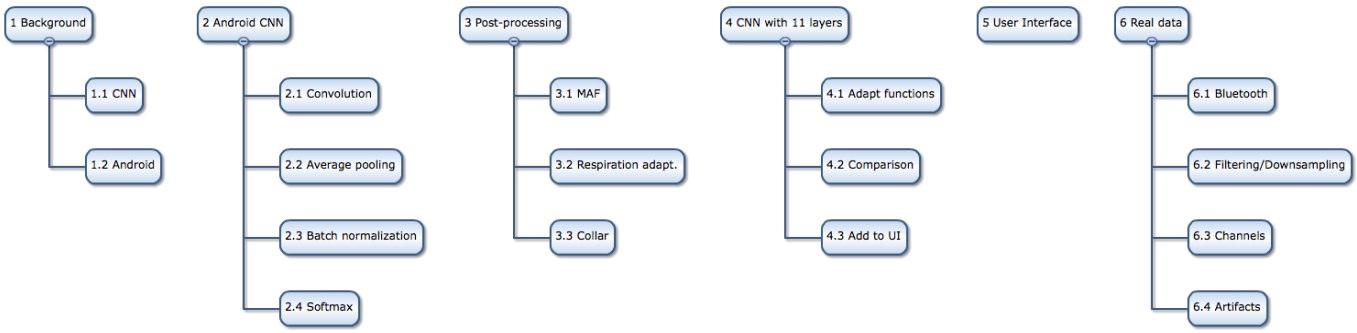

Figure 3. Work plan scheme

The Gantt diagram of the project is presented in Fig. 3.

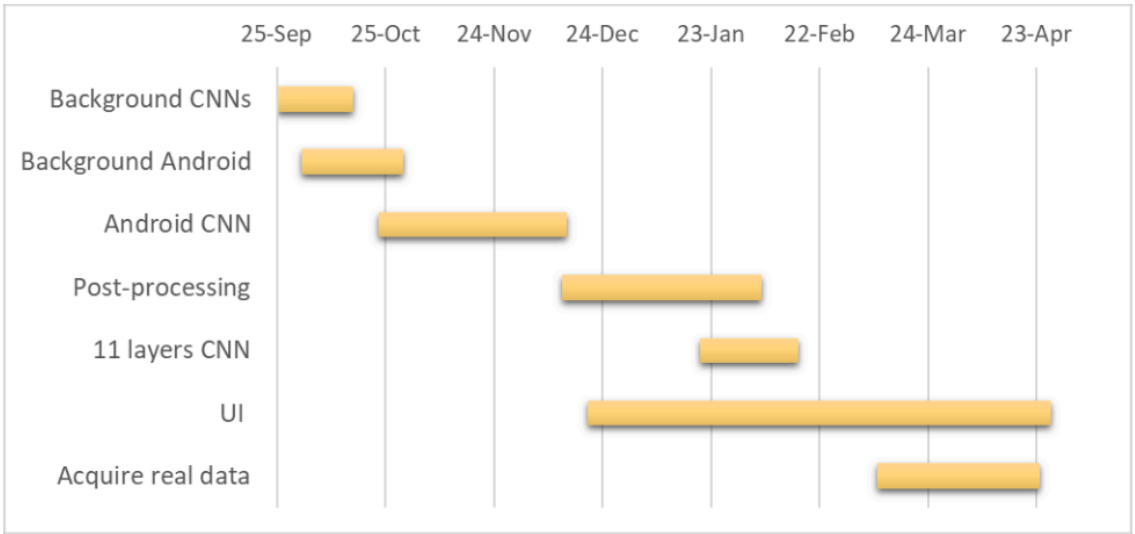

Figure 4. Gantt diagram

The description of the work packages and milestones is in the Appendix 1.

2. State of the art of the technology used or applied in this thesis:

Seizures in newborn babies are a serious concern for clinicians due to the fact that acquisition and interpretation of neonatal EEG requires deep neurophysiologist professional experience. The portable Neonatal EEG Monitoring and Sonification [1] product developed by Irish Centre for Fetal and Neonatal Translational Resarch (INFANT) with the collaboration of University College Cork., currently represents the state-of-the-art in this project. This portable stethoscope consists in a board (OpenBCI) as the EEG acquisition system using dry electrodes. Using dry electrodes over wet does not significantly degrade the EEG quality and also there is no need to use abrasive creams or conductive gels [2]. The systems aim to provide the clinician with three techniques of seizure detection: Visualisation, Sonification and decision support based on a Convolutional Neural Network. The system had an Android App with sonification based on Phase Vocoder but it lacked many features as CNN decision, Bluetooth connection, a final UI product or a database to save the acquisitions among others.

The idea of using machine learning for seizure detection is suggested in [3], where INFANT centre proved that the new technologies are more than capable to detect seizures in an EEG signal, based on different features extracted. The convolutional neural network used was already build and trained by Alison O'Shea [4] and the segmentation of raw neonatal EEG into seizure or non-seizure epochs achieved a better level of performance compared to the previously developed SVM-based system.

Experiments	SVM	6 Layer CNN	11 Layer CNN
AUC	96.59%	97.03%	97.61%
AUC90	82.87%	83.22%	86.85%
Sensitivity @ 0.25fd/h	73%	73%	81%

Table 1. SVM vs CNN

The AUC is the Area under the receiver operating curve, which is a curve that plots sensitivity against specificity. Sensitivity is the percentage of correctly detected seizure epochs. Specificity measures the percentage of epochs which were correctly labelled as being non-seizure (background). By plotting these two metrics for a range of decision threshold values an assessment of the classifier's performance can be obtained.

AUC90 corresponds to the area of the AUC curve where specificity is greater than 90%. This higher specificity score will correspond to a lower number of false seizure alarms per hour of EEG being recorded i.e. a reduction in the number of False Detections per Hour.

FD/h is the rate of false seizure detections per hour of EEG being recorded.

2.1. Convolutional Neural Networks

Convolutional neural networks are based on neurons with weights which can learn. Each neuron receives some entries, computes a scalar product and then an activation function, working similarly to a real brain neuron.

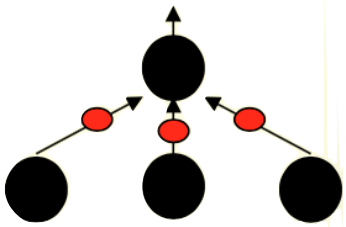

Figure 5

Each neuron receives inputs from other neurons and the weights can be positive or negative (Figure 5).

$$y = b + \sum x\,w \tag{1}$$

Considering equation (1) the output of each neuron is a bias *b* plus the sum of each input *x* by its weight Figure 6.

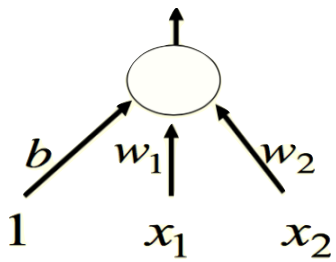

Figure 6

To learn biases using the same rule that is used for learning weights, bias is taken as a weight on an extra input line that always has an activity of 1.

The network used is fully convolutional, composed of convolutional layers without any fully connected layers. So, even the decision-making layers at the end of the network are filters. It has 4 different types of layers:

1. Convolutional layer
2. Pooling layer
3. Batch normalization layer
4. Activation layer

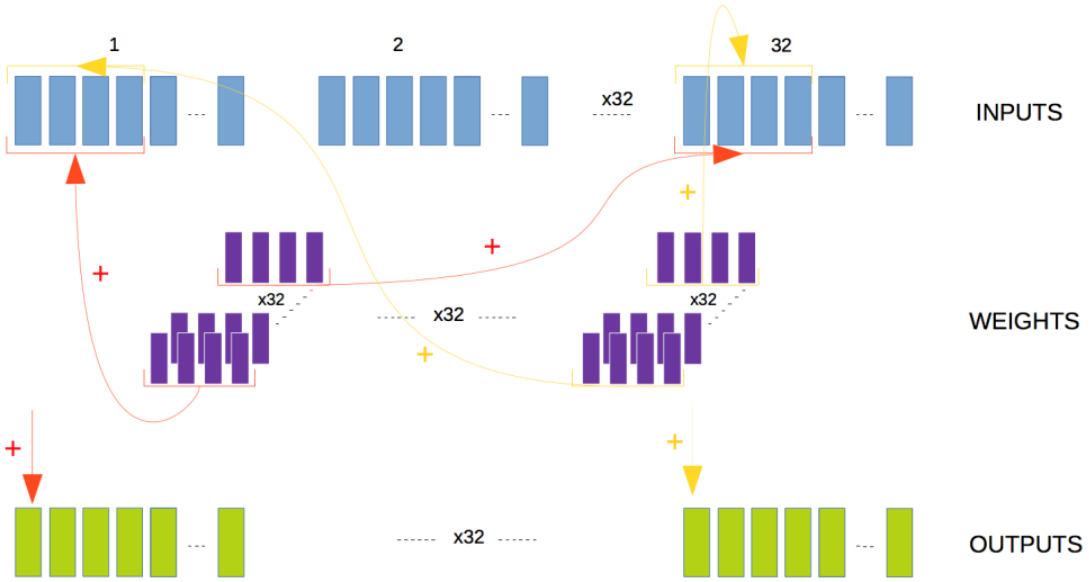

Figure 7 Convolutional layer

2.1.1. Convolutional layer

This is the layer that makes the convolutional network different than other neural networks. Instead of using multiplication of matrix, this layer performs a convolution between the input and the filter or kernel resulting in a feature map of the original input reducing the number of parameters.

In Figure 7, the core of the layer is shown. Each weight or feature-map has as many dimensions as inputs and the number of weights matches the number of outputs. In the network used, each filter has 32 dimensions and kernel 4 with stride 1. It means that dimension one of first filter, is multiplied by the firsts 4 samples of input one and then the filter is moved only one sample forward. The results of each dimension of the same filter are summed up in the same output sample. The activation function used for convolutional layer is ReLu function., more commonly used in CNN's.

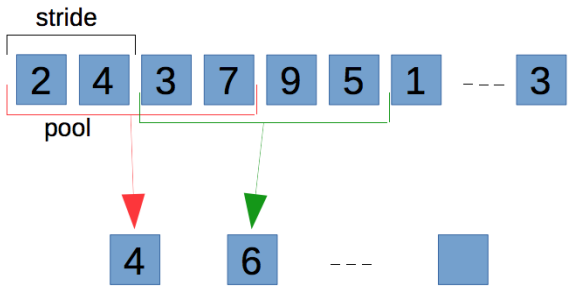

Figure 8. Pooling layer

2.1.2. Average pooling layer

This layer (Figure 8) is usually placed after the convolutional layer. Its main utility lies in the reduction of the spatial dimensions of the input volume for the following layer. With this layer the network loses information, but this lost can be positive for the layer, since it leads to less calculation overload for the next layers.

The layer takes the number of samples that correspond to the pool, it sums up the values and then makes an average of them.

2.1.3. Batch Normalization layer

In the training stage, Batch Normalization was used to achieve high accuracy using less training steps. This is possible because the normalization have been part of the model architecture performing normalization for each training mini-batch. Batch normalization allows to use much higher learning rates and be less careful about initialization. To build a BN layer in the testing stage, we need to consider the next parameters provided by the network model:

$$y(k) = \gamma(k)x(k) + \beta(k) \qquad (2)$$

In (2) *gamma* and *beta* are learned along with the original model parameters, which scale and shift the normalized value.

Then the layer needs the mini-batch mean (3) and the mini-batch variance (4) to normalize each input.

$$\mu_B = \frac{1}{m}\sum_{i=0}^{m} x_i \qquad (3)$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=0}^{m} (xi - \mu_B)^2 \qquad (4)$$

The normalized values correspond to:

$$x_i = \frac{xi - \mu_B}{\sqrt{\sigma_B^2 + \in}} \qquad (5)$$

And their linear transformations:

$$y_i = \gamma \hat{x_i} + \beta \qquad (6)$$

2.1.4. Activation layer

Throughout the network we used two different activations, relu in each Conv Layer and softmax at the end of the network.

The last layer used in the network is considered as a neuron separated from the rest. This final classification layer will have as many neurons as the number of classes to predict. In our network there is only one neuron and the function used is the *Softmax* function.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=0}^{K} \text{ezk}} \quad (7)$$

Softmax function squashes the outputs of each unit to be between 0 and 1 and divides each output such that the total sum of the outputs is equal to 1. Mathematically, Z is a vector of the inputs to the output layer (7).

Regarding the **rectified linear unit** (ReLu), when the input is smaller than zero, the function will output zero. Else, the function will mimic the identity function. It's very fast to compute the ReLu function.

Architecture structure of 6 layers CNN build:

Layer Type	Shape	Output Shape	Parameters
Input	256	256x1	0
1D Convolution	32 filters 4x1 kernel Stride 1	32x253x1	160
1DConvolution	32 filters 4x1 kernel Stride 1	32x250x1	4128
1DConvolution	32 filters 4x1 kernel Stride 1	32x247x1	4128
Batch Norm.		32x247x1	64
Average Pooling	Pool 8 Stride 2	32x120x1	0
1DConvolution	32 filters 4x1 kernel Stride 1	32x117x1	4128
1DConvolution	32 filters 4x1 kernel Stride 1	32x114x1	4128
Average Pooling	Pool 4 Stride 2	32x56x1	0
1DConvolution	2 filters 4x1 kernel Stride 1	2x53x1	258
GAP **Softmax**		2x1 2	0 0

Table 2: CNN Architecture

2.1.5 CNN post-processing

In order to improve the seizure probability curve at the output of the network, different techniques have been used.

Moving Average Filter

This filter consists in a simple average with the last 60s of EEG. This way we get a smoother output avoiding false alarms Figure 9.

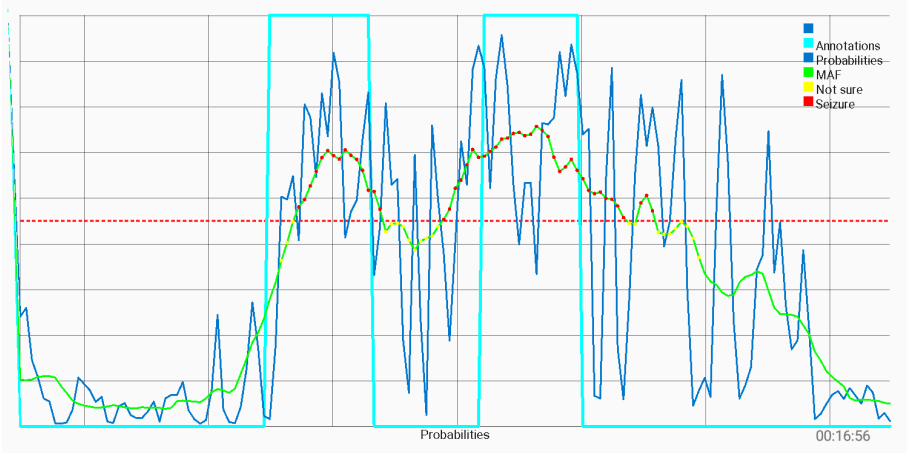

Figure 9. Green line corresponds to the output after the MAF

Respiration Adaptation

A seizure is usually detected when an EEG epoch has a constant period. The respiration of the patient may interfere with this period, thus causing a false convulsion to be detected. To avoid this phenomenon, an adaptation algorithm is applied (Figure 10).

This adaptation only applies to those epochs detected as seizure. The probability of the epoch is subtracted by the average of the 3 min background before that detection and hence the epoch will be labelled as non-seizure if the probability after the adaptation does not go beyond the decision threshold.

Collar

Due to the different post-processing techniques, the duration of the seizure could be shortened. In terms of the patient brain health status the duration of the seizure is essential, so the collar aims to restore this duration. The constant value that has been chosen is half the MAF (30s) at the beginning and the end of the seizure. Such modification is not applied in the real-time values but in the later seizure list history since it is computed offline [3].

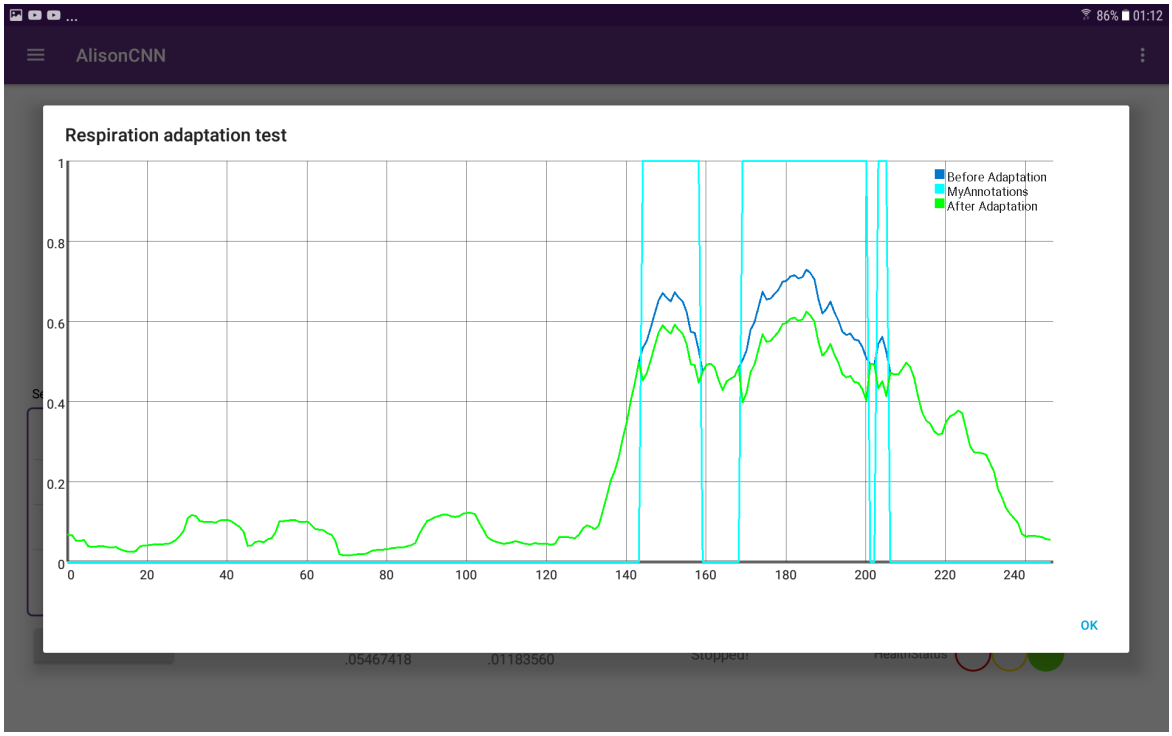

Figure 10. Comparison between before and after resp. adapt.

2.2. Neonatal EEG sonification

The Phase Vocoder is an algorithm for timescale modification of audio. This whole block has been implemented by another student and it was working on real time in the previous app, so only the main guidelines of the block will be shown. This block is in charge of the sonification technique. Basically it picks up the EEG signal and takes it into the audible frequency band to ~~the~~ human beings. After a long study made by Dr. Andriy Temko, it has been successfully concluded that there is an audible pattern when the patient has a seizure.

The scheme is as follows in Figure 11:

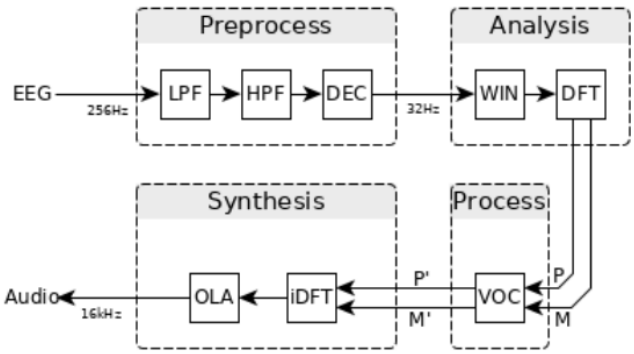

Figure 11. Neonatal EEG Sonification using Phase Vocoder

2.3 Android OS

Android is the environment picked for the front-end of the project for several reasons. The existent app and therefore the sonification algorithm were implemented in Android in such a way that continue the project in the same language made the integration easier.

Android is the most widely used system today allowing greater accessibility to the app. Also, the Android market is the most accessible marketplace for developers to publish an app.

A new official language call Kotlin [7] developed by JetBrains is now recognised by Google (owner of Android). This language is expressive, concise, powerful and interoperable with the existing Android languages and runtime. This language provides safer coding avoiding numerous Java exceptions and it's readable and concise. In my opinion, the app would be more valuable if it's rewritten in Kotlin in a near future, making the app attractive in the medical environment for the novel techniques in seizure detection and for the promising language used in its coding.

Finally, there are thousands of libraries developed in Java that can be easily imported in Android opening up a world of possibilities when it comes to programming graphical interfaces or developing complex algorithms.

2.3.1 SDK

The project has been developed using Android Studio 3.0.1 powered by JetBrains. The compile version SDK 26 has been used and the min SDK version target is 15.

2.3.2 Communication

The Android app has to be connected with the acquisition board (OpenBCI) via Bluetooth. This block is part of the final year project done by Kevin Huillca. The integration of this block is considered for the close future and as soon as it's implemented the project will be in an almost finished stage.

3. Methodology / project development:

3.1. Software

The project uses the existent Phase Vocoder algorithm implemented by Jonatan Poveda in the previous app and the model of CNN built by Alison O'Shea in Python.

Sonification was implemented to work in real-time but in the actual app the sonification is used in the review mode (offline).

The CNN model build in KERAS (Python based), has no interoperability with Java so an own library has been developed for the project.

All the Android SDK libraries are free to use in the app although these do not provide convolution of matrix among others.

Furthermore, for all charts the MP AndroidChart library by PhillJay [8] has been used. This library provides many features that the last library used in the previous app, GraphView, did not.

In order to fix memory leaks in Android, Memory Analyzer Tool (MAT) had been used.

Regarding the battery consumption the software picked is Battery Historian using the platform Docker.

To manage the project a private Github repository had been created. This platform allows to make changes and add features without overwriting the existing code using different branches. Also makes the software versioning much easier assigning unique version names or unique version numbers to unique states of the software being developed.

All the platforms and libraries used are free.

3.2. Hardware

For debugging and testing purposes, a Samsung A7 10 tablet provided by INFANT is used. Currently, the project is only working on that device, it has to be considered in the future to adapt the interface for all devices.

3.3. Project development

As discussed in other sections, the main block of this project is the implementation in Android of a deep learning algorithm based on convolutional neural networks. Both the implementation of the algorithm and the interface have been done in parallel, this is an attempt to explain the project development including screenshots of each stage.

The best strategy to start building the network was trying to build it step by step, strengthening each sub-block before looking at the whole network.

The first layer build was the convolutional layer. The main obsession was getting strong accuracy with many decimal places due to the importance of accuracy in a medical environment, so the library Big Decimals was used. This library allows to work with long big decimal places. Nevertheless, this library has its own elementary functions, and these added big delays to the processing time (about 50s for each 8s epoch). Small processing time was required for the real-time acquisition so after many tests we concluded we could work with float format for the weights vectors without loss of accuracy and achieving each forward pass through the network to be processed in under 1 second.

Once the convolutional layer was developed and tested, Average pooling, Batch Normalization and Softmax layers were developed following the mathematical expressions explained in previous sections.

All the weights of the network were imported as a txt file in the app. These filters were vectors of great length thereby producing memory leaks in the Android memory heap. This means that the Random-access memory (RAM) of the device was constrained. This issue was fixed proactively reducing memory usage. The weights were memory allocated in an asset directory and realising any reference object at the appropriate time.

The first network implemented had 6 layers and the UI build for the test is shown in Figure 12.

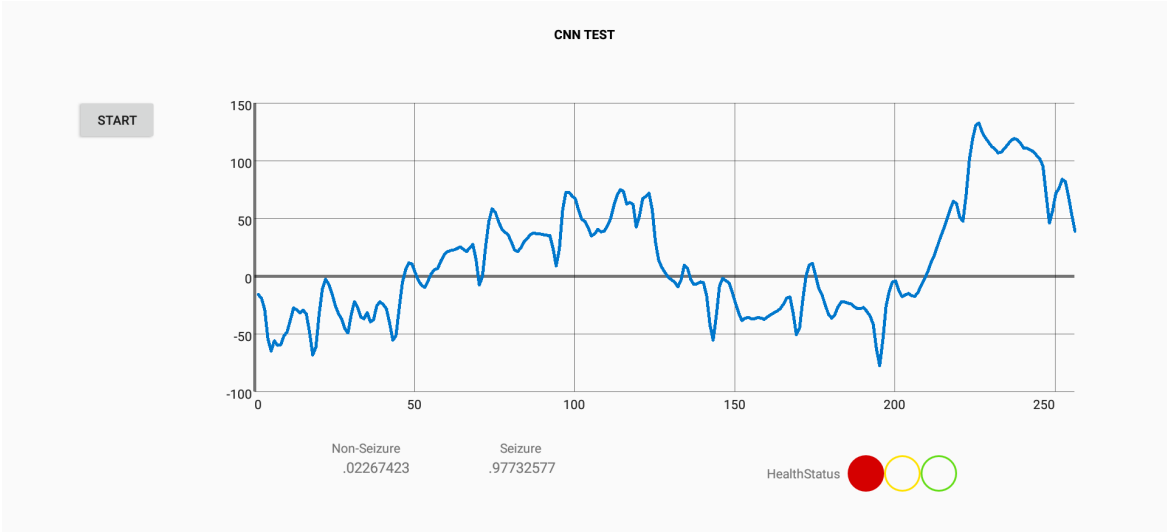

Figure 12. First UI test

The interface had a chart to visualise the EEG epoch and the probability of seizure. Also it has a traffic light system to show the health status.

Then the 3 post-processing techniques came into the picture. The integration of the three of them was made in parallel trying not to make them influence each other ensuring its correct functioning, shown in Figure 13 and Figure 14. A buffer of 60 seconds was needed to satisfy the MAF integration. This interface was working in real-time simulated with an EEG signal loaded in a .txt file. The chart showed only the probabilistic output of the CNN so, all the seizures detected were saved in a list of seizure with the option to visualise the EEG signal of the corresponding seizure.

In that stage an 11 layer Network was introduced adapting the existing own library to custom layers. The purpose was to have an alternative with a higher power consumption but more accuracy.

Moreover, until now, the decision threshold of seizure was fixed at 0.5 (with the option to be changed by hardcoding it). A new seek bar feature was added to give the user the option to change the threshold value in real-time.

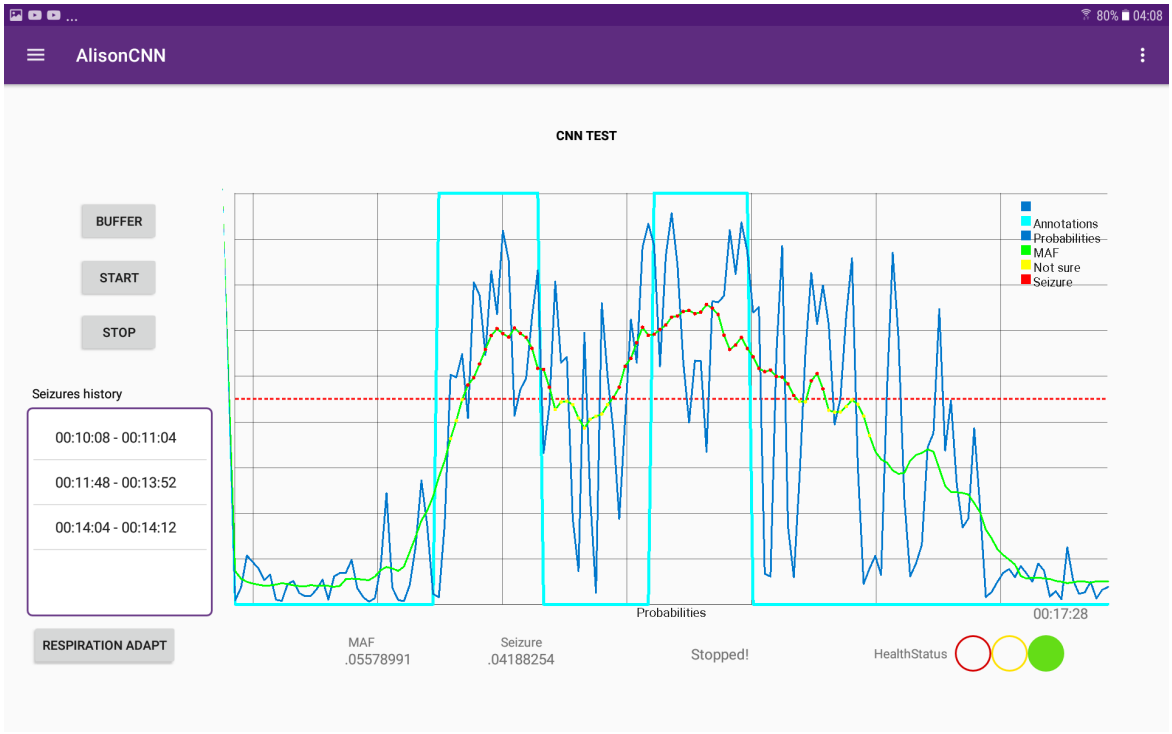

Figure 13. Main layout

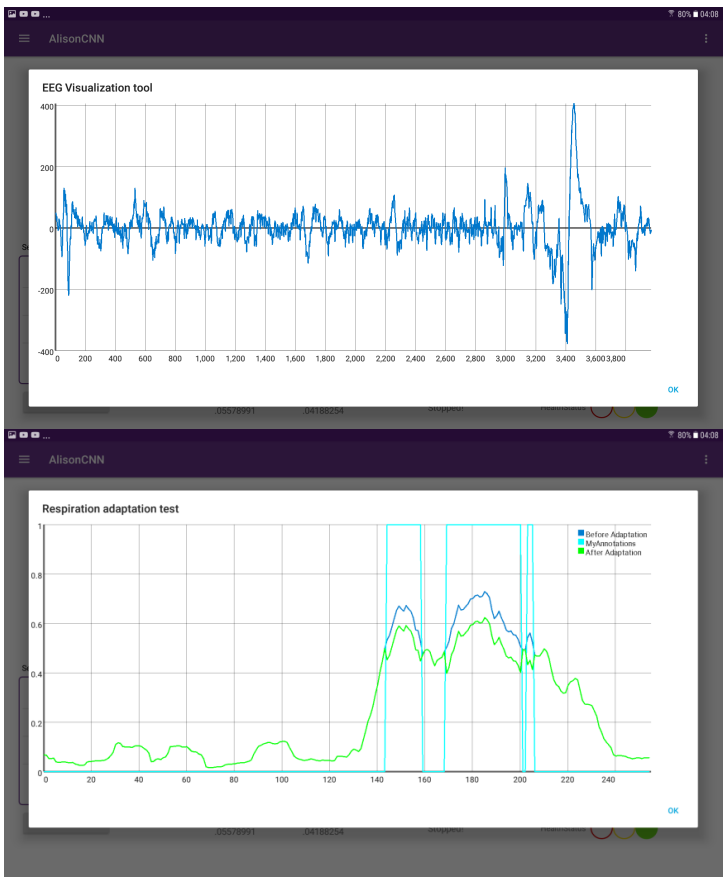

Figure 14. Seizure EEG signal and Respiration Adaptation

Another feature to make the app battery friendly was the shift selector. The CNN needs 256 samples of EEG (8 seconds) to make a decision. The shift values correspond to the number of seconds the CNN has to wait to process another epoch. Options:

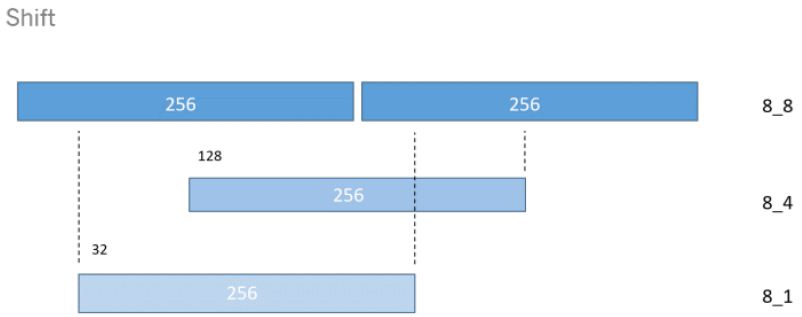

Figure 15. Shift

This means if we have shift 8_8 selected, the CNN will make a decision every 256 samples (8 seconds) received without overlapping. If the 8_4 option is selected every 128 samples the CNN will make a decision considering the lasts 128 samples (128 new samples received plus the 128 previous samples, which equals 256 samples required for the CNN pass). The same thing happens when selecting shift 8_1 but with 32 samples (1 second of EEG).

A meter level was proposed to replace the traffic light system.

All this new features were adapted for both layers and post-processing functions resulting in an app with both EEG signal and probabilistic output in real-time. With a review mode to check the seizure detected and possibility to sonify from the chosen period time with a value of a second.

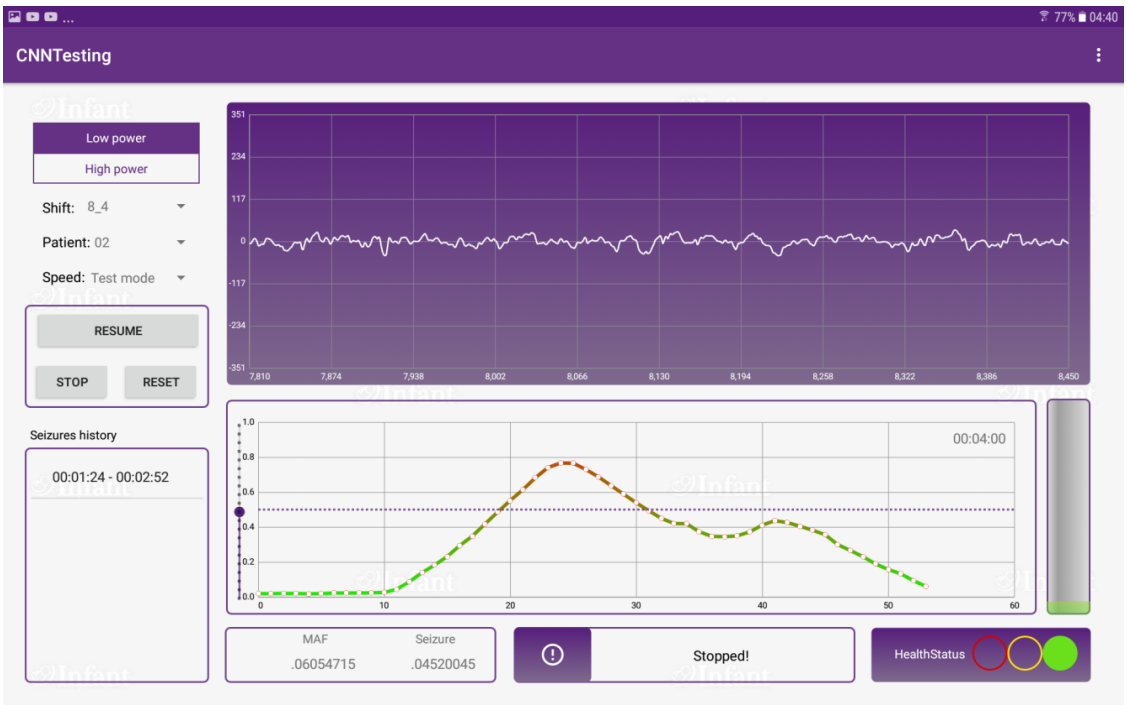

Figure 16. Overview of the main layout of the project

Figure 17. Review mode

3.4. Multithreading on Android

To achieve real-time acquisition and processing multithread processing must be used. By default, in Android every application code runs in the main thread. When all the actions are finished a call to the UI runnable is made to update changes.

This means, that without multithreading, our app should wait every 1, 4 or 8 seconds (depending on the shift selected) plus the times it takes the processor to update the UI. When such events occur, the interface freezes and the user could think that the app has crashed because it is not responding until the action finishes and the UI is updated.

Having said that in our case it is mandatory to implement multi-threading. Several options have been studied.

Android supports the usage of Thread class to perform asynchronous processing. Also Android offers the *java.util.concurrent* package to perform heavy operations in the background using ThreadPools and Executor classes.

To update the UI from a Thread, synchronization with the main thread is needed and because this restriction, it is better to use Android specific code constructs such as *android.os.Handler* class or AsyncTasks classes.

If posting data multiple times to the main thread is needed, for instance, plotting the values acquired via Bluetooth each second, Handler is particular useful.

In the app, handlers are reused to avoid object creation using existing *handlers* object of the activity. Basically, with the *post()* method, it is possible to post objects in the view.

The AsyncTask in our case, is perfect to do heavy short operations (as CNN or sonification) in the background and to synchronize again with the main thread. A very interesting feature of AsyncTask is to report progress of the running task while the heavy operations are being performed.

The AsyncTask class is executed via *execute()* method. The heavy calculations are done with the *doInBackground()* method and the output is passed to *onPostExecute()* method. To update the UI before or during the background operations *onPreExecute()* and *onProgressUpdate* can be used.

3.5 Bluetooth and Phase Vocoder integration

The Bluetooth block is part of another student's work. This block is not yet finished but its integration is already considered in my project with a thread and its respective buffer. Once the block is integrated, we will start working on the filtering and downsampling functions of the EEG signal to work with it in the Android device.

Regarding the PhaseVocoder and therefore the sonification, functions coded by Jonatan Poveda in the previous app where extracted and implemented in the new Android project. The project had to be adapted to C++ debugging due to some of the functions where build in C++. Currently the system is sonifying every 128 x 4 samples. This value can be easily changed.

3.6 Menu and patients' management

To satisfy costumer's needs, a patient list with the EEG acquired had to be implemented (Figure 18). Currently is under development but this are the main features that this option has:

- List of the patients.
- Search motor to filter the patients by name.
- Patient card with relevant information and a pie chart with the seizure history.

This layout is not finished yet and does not work in parallel with the acquisition fragment. So if you are acquiring signal but you change to patient list tab, the acquisition would be stopped.

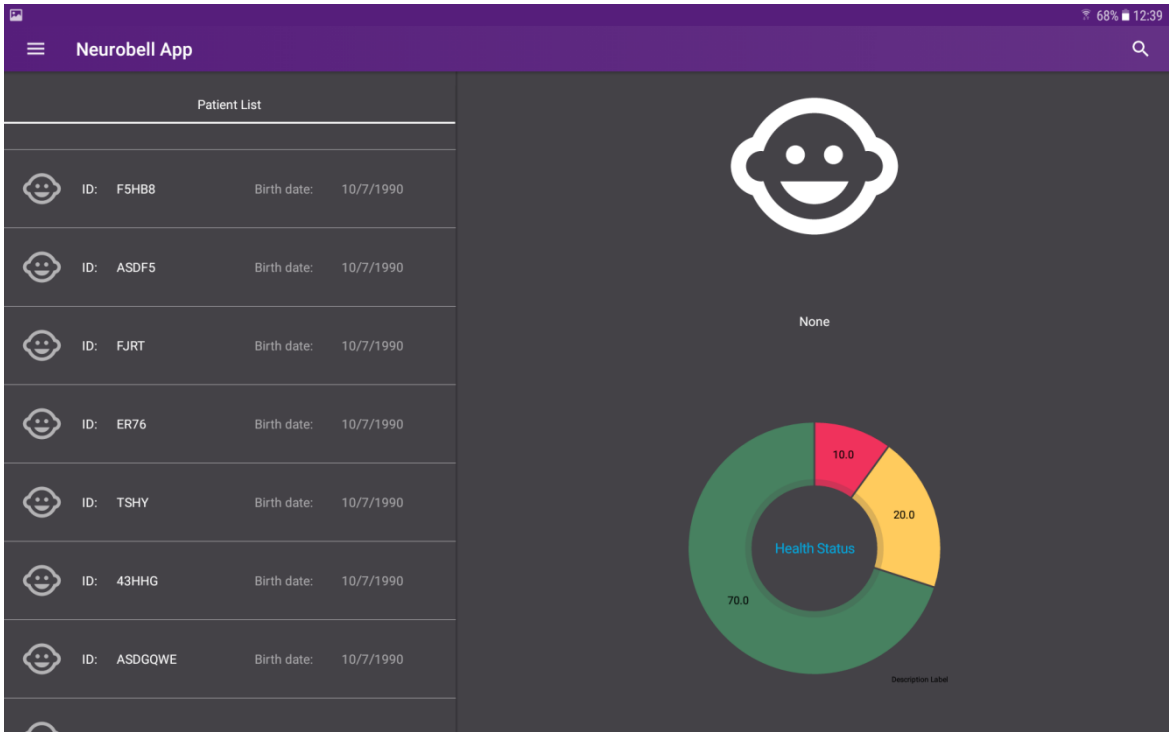

Figure 18. Patient list layout

4. Results

At the end of the project, the Convolutional Neural Network is working as expected. The first network probabilities are compared with the Python environment outputs (Table 1).

Classification	**Python**	**Android**
seizure	0.02267421	0.02267423
non-seizure	0.97732580	0.97732577

Table 3 First full network test with one epoch of 8s

After that, we concluded that the network was successfully build and the probabilities to had a different output was depreciable.

Computational expenses are shown in table 2. Nevertheless, this values are isolating each block from the main thread, so the overall power consumption is not the sum up of all of them. Acquisition and Sonify were computed by other students.

	CPU(%)	**RAM(MB)**	**Power (mAh)**
Acquisition	10.72±1.16	25.34±0.38	19.13
Plotting	-	43.64±0.52	-
CNN 6	21.29±2.8	23.33±1.18	10.22
CNN 11	25.63±1.01	25.96±0.72	26.11
Sonify	43±15	46.00±14	10.1

Table 4. Computational expenses results

The difference in number of parameters located on memory and accuracy of the 2 different networks (6 & 11) is shown in Table 3.

	Number of parameters	**Sensitivity at 0.5 fd/h**	**Sensitivity at 1 fd/h**	**Processing time (ms)**
CNN 6-layer	17,058	79%	85%	115.58±3.23
CNN 11-layer	28,642	83%	89%	283,083±3.28

Table 5. 6 vs 11 layers comparison.

The processing time achieved by both networks is really short, allowing the network to work in real time without skipping any sample. The worst-case scenario is when the shift 8_1 is selected, but with a medium prestation's device, the system will operate without delays.

In Figure 19 and 20 the final product application is shown. It integrates both CNN and sonify algorithms. Provides a seizure list of seizures that had been detected permitting a full tracking of all the convulsions detected by the CNN.

Figure 19. Main Stethoscope layout

Some patient's EEG signals are loaded as test signal since Bluetooth is not integrated yet, allowing to visualise how the network works with different signals.

Both networks can be selected using the buttons 'Low power' or 'High power' corresponding to 6 and 11 layers respectively.

While the system is still acquiring, there is an option to visualise the seizures already detected and verify with the 3 techniques (Visualisation, Sonification and CNN decision) if the segment labelled as seizure is really a seizure or not.

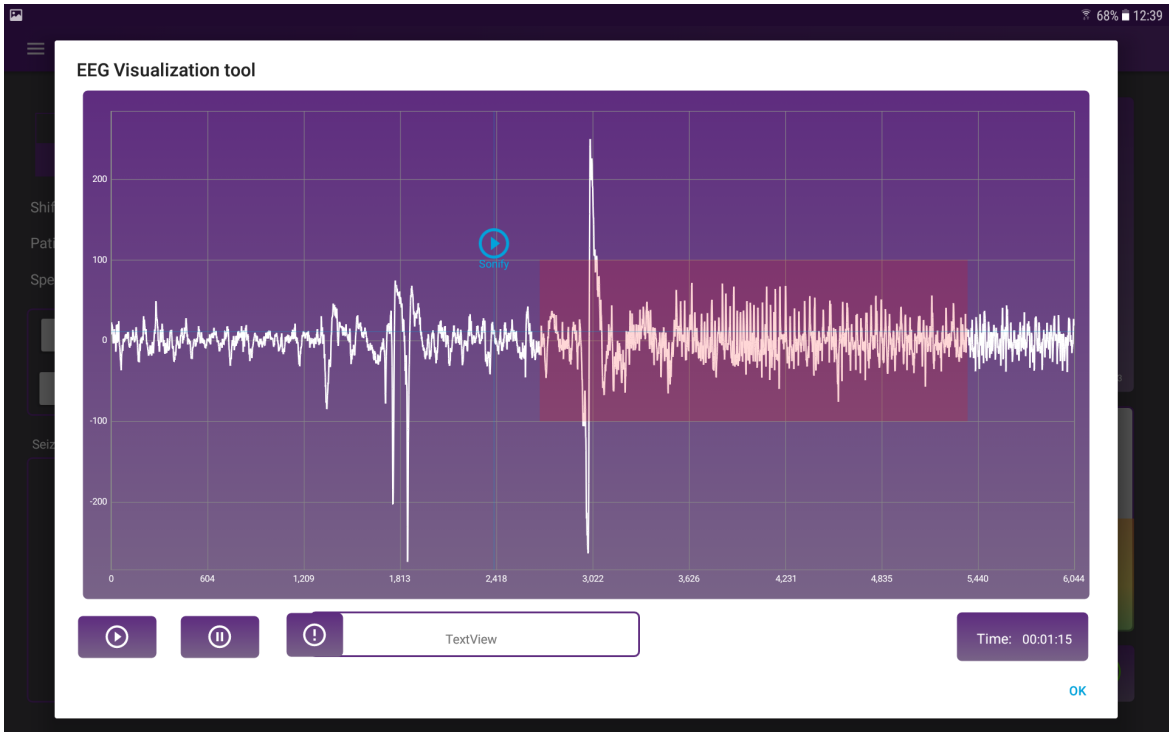

Figure 20. Fast check mode while the system is still acquiring.

Figure 21. Menu bar to navigate through the different fragments

5. Budget

The total budget of this project is:

	Cost (€)
Software	0
Instruments	199
Personnel	8000
Total	**8199**

Table 6. Total costs.

The cost of the project is divided in three blocks, software (all free of charge), instruments and personnel:

Instruments

The only device used was the tablet Samsung A6 10 provided by infant.

Instrument	Units	Days used	Cost (€)
Samsung Tablet A7 10.1	1	212	199

Table 7. Instrument costs.

Personnel

Name	Rank	€/month	Total months	Cost (€)
Edu Salgado	Junior Engineer	1000	8	8000

Table 8. Personnel costs

6. Conclusions and future development:

The main goals of this project have been satisfied giving to the Neurobell App the capability to use CNN as a seizure technique detection in Android; therefore, the first project proposal has been fulfilled and even extended. Nevertheless, after the lifetime of the project was prolonged, new features were added and a new app was designed. The application is still far from the final product. Areas for improvement have been identified and are now under implementation, specifically an offline review mode with all the features, a better integration of the sonification to ensure a great user experience, and a more suitable way to visualise the probabilistic output.

I will continue working in the research centre at least until July, so that the app will have a continuity and a final prototype could be seen finalised by the end of July.

I am considering to work on the acquisition in a deeper layer of the Android environment to give the user the flexibility to navigate through the app while the signal is still being acquired.

Currently another novel technique of sonification is studied and under development and it could be implemented in the app to see which one performs better in Android.

The acquisition board has the option to acquire signals from various channels. This has to be contemplated in future versions when the Bluetooth is fully integrated.

Another feature that must be developed is the management of the database for the patient metadata. In my opinion the best way is to use the Json format for each set of attributes.

I think that the migration of the app to Kotlin will give it an extra value to squeeze into the market and be one of the first apps in the medical environment fully developed in Kotlin.

The results of the CNN inte stated in this project are reflected in the abstract “Neonatal Brain Stethoscope: Design and validation of a low-cost EEG acquisition & interpretation system prototype”, which will be presented in The 7th Congress of the European Academy of Paediatric Societies (EPAS 2018), and also in the conference paper “Neonatal EEG interpretation and decision support framework for mobile platforms”, which will be presented in the 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE EMBC 2018), both already accepted.

Bibliography:

[1] Jonatan Poveda, Mark O'Sullivan, Emanuel Popovici and Andriy Temko, 'Portable Neonal EEG Monitoring and Sonification on an Android Device'.

[2] Mark O'Sullivan et al., 'Comparison of Electrode Technologies for Dry and Portable EEG Acquisition'.

[3] A. Temko et al., 'EEG-based neonatal seizure detection with Support Vector Machines', Clin. Neurophysiol., vol. 122, no. 3, pp. 464-473, Mar. 2011.

[4] Alison O'Shea et al., 'Neonatal Seizure Detection using Convolutional Neural Networks', IEEE Mach. Learn., Sept. 2018.

[5] Sergey Ioffe and Christian Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Interval Covariate Shift.

Appendices 1: Work packages

Project: Android Deep Learning Algorithm	WP ref: 1
Major constituent: Background/Introduction to CNN and Android	Start event: 25/09/2017 End event: 30/10/2017
Short description: Studying the state-of-art of Convolutional Neural Networks, their main functionality blocks and constituents. Studying software development for Android.	
Internal task T1: Introduction to Machine Learning and Neural Networks. **Internal task T2:** Simple first implementation of Neural Networks. **Internal task T3:** A closer look at Convolutional Neural Network. **Internal task T4:** Understand the CNNs developed in Keras. **Internal task T5:** Android basics.	Finished

Project: Android Deep Learning Algorithm	WP ref: 2
Major constituent: Android CNN	Start event:30/10/2017 End event: 10/12/2017
Short description: Implement a Convolutional Neural Network in Android environment doing an interface to test the results. The implementation is divided in layers.	
Internal task T1: Implement Network with 1 layer, 2 channels in and 2 channels out. **Internal task T2:** Network with n inputs and n outputs **Internal task T3:** Network with n layers and with an average pooling layer. **Internal task T4:** Build a Batch Normalization layer. **Internal task T5:** Build the full architecture with Softmax activation. **Internal task T6:** Optimization of the code.	Finished

Project: Android Deep Learning Algorithm	WP ref: 3
Major constituent: Post-processing	Start event: 20/12/2017 End event: 06/02/2018
Short description: Different techniques to get better accuracy in terms of seizure detection.	
Internal task T1: Moving Average Filter **Internal task T2:** Respiration Adaptation **Internal task T3:** Collar **Internal task T4:** Integration in the network	Finished

Project: Android Deep Learning Algorithm	WP ref: 4
Major constituent: 11 layers Neural Network	Start event: 20/01/2018 End event: 15/02/2018
Short description: Build a 11 layers Neural Network and compare its performance with the 6-layer one	
Internal task T1: Adapt existing library of CNN functions to variable layers **Internal task T2:** Create de model of the new network **Internal task T3:** Test the network with different inputs previously labelled **Internal task T4:** Add the post-processing to the new network. **Internal task T5:** Add network to the UI with the option to change network between the 11 and the 6 layer already implemented.	Finished

Project: Android Deep Learning Algorithm	WP ref: 5
Major constituent: GUI	Start event: 20/12/2017 End event: 27/04/2018
Short description: Develop a UI of the client	
Internal task T1: Add CNN in a traffic light system **Internal task T2:** Possibility to change between Networks **Internal task T3:** Real time processing **Internal task T4:** Review mode with the 3 techniques of detection. **Internal task T5:** Plot EEG in time **Internal task T6:** Change shift in real-time **Internal task T7:** Integrate Sonification algorithms **Internal task T8:** Save the signal acquired (e.g. in a server) **Internal task T9:** Add options for metadata info management **Internal task T10:** Client feedback	Finished

Project: Android Deep Learning Algorithm	WP ref: 6
Major constituent: Acquire real data	Start event: 10/03/2018 End event: 24/04/2018
Short description: Complete the whole system connecting with the portable battery operated EEG acquisition board.	
Internal task T1: Integrate Bluetooth connection with main app **Internal task T2:** Filtering and downsampling **Internal task T3:** Different channels selector **Internal task T4:** “Is it EEG?” artifact detector	Pending on external project

Milestones

WP#	Short title	Milestone / deliverable	Date (week)
WP1	Background	Android and Deep learning	30/10/2017
WP2	First CNN	6 layers CNN build and tested	10/12/2017
WP3	Post processing	MAF, Resp. adapt. and collar	06/02/2018
WP4	Second CNN	11 layers CNN build and tested	15/02/2018
WP5	First UI	User Interface with both networks working and different types of visualization	21/02/2018
WP5	EEG time	Add EEG visualization in real time (time domain)	9/03/2018
WP5	Review mode	Interface where user can play with the 3 different techniques of detection	16/03/2018
WP5	Sonification	Add sonification method in the main algorithm	23/03/2018
WP5	Save	Option for saving the signal acquired	30/03/2018
WP5	Menu	Different options to manage baby info	30/03/2018
WP5	Feedback	1st round of changes in UI considering costumers recommendations	6/04/2018
WP6	Bluetooth	Adding Bluetooth connection to the system	Pending on external project
WP6	Filtering/Downsampling	Prepare the signal for CNN pass	Pending on external project
WP6	Channel selector	Be able to choose electrode	Pending on external project
-	Final App	Final product of the project	27/04/2018
-	Feedback/Redesign	Second round of changes in UI considering costumers recommendations	04/05/2018
-	Final Report	Program documentation done	11/05/2018

Appendices 2: Incidences and Deviations

- Optimisation of the code has been performed to achieve real-time processing with 1s shift of the processing window. This includes implementation of reusable memory, better syntax for loops, smart memory allocation for the network model and multithreading among others.
- Important issue with Big Decimal library. At the very beginning, strong accuracy with many decimal places was required. Only Big Decimal library allows to work with long decimal places in Java. Nevertheless, this library has its own elementary functions, and these added big delays to the processing time (about 50s for each 8s epoch). Short processing time is required for the real-time acquisition so after many tests we concluded we could work with float format for the weight vectors without loss of accuracy and achieving a processing time of under 1 second for each forward pass through the network.
- The first network that was built had 6 layers. A new more accurate network with 11 layers was then implemented and the computational cost was compared and contrasted with the gain in performance.
- There was a long delay with respect to what was planned for building the 11-layer network due to the model creation with the new weights. The own software library made with all the layer functions (Convolution, Batch Normalization, Activation, ...) was not adapted for different number of layers. Currently, layers can be easily added or removed to implement new networks.
- Big issue with sonification. The whole algorithm was implemented in another project including a front-end app by another student during a previous term. This made the algorithm difficult to read and understand in terms of coding.
- WP4 (Bluetooth Connection) was moved to another student (Kevin Huillca Aparicio) due to lack of time. Accordingly, WP5 has been delayed until the real signal from the board is finally acquired.
- The UI is not adapted for all devices, currently it is working only on Samsung A7 10, tablet provided by INFANT.

Glossary

EEG: Electroencephalogram

CNN: Convolutional Neural Network.

SVM: Support Vector Machine

AUC: Area under the receiver operating curve.

RELU: Rectified liner unit.

FD: False detections

SDK: Software Development Kit

WT: Well Trust Seed Award