



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Systems and Computation

Deep Learning applied to health

Master's Thesis

Master's Degree in Artificial Intelligence, Pattern Recognition and
Digital Imaging

AUTHOR: Martínez Bernia, Javier

Tutor: Gómez Adrian, Jon Ander

ACADEMIC YEAR: 2021/2022

Resum

Els avanços en el camp de la Intel·ligència Artificial ens han portat a una nova era de recerca. Els models de *Deep Learning* permeten detectar i diagnosticar malalties en pacients de manera automàtica. En aquest projecte, s'apliquen tècniques de *Deep Learning* a diversos casos d'ús relacionats amb la salut. Es fa ús de xarxes neuronals per construir models capaços d'afrontar les diverses tasques de cada cas d'ús. Les tasques presentades en aquest treball estan relacionades amb l'anàlisi d'electroencefalogrames i imatge mèdica. El projecte cau en el marc de desenvolupament de la eina EDDL, un software que permet crear i manipular els models esmentats. L'objectiu principal és fer ús de l'eina per afrontar problemes d'aprenentatge automàtic amb models de l'estat de l'art.

: Aprenentatge profund, Imatge mèdica, Xarxes neuronals, Electroencefalogrames (EEG), Reconeixement de patrons

Resumen

Los avances en el campo de la Inteligencia Artificial nos han llevado a una nueva era de investigación. Los modelos de Deep Learning permiten detectar y diagnosticar enfermedades en pacientes de manera automática. En este proyecto se aplican técnicas de Deep Learning a diversos casos de uso relacionados con la salud. Se hace uso de redes neuronales para construir modelos capaces de afrontar las distintas tareas de cada caso de uso. Las tareas presentadas en este trabajo están relacionadas con el análisis de electroencefalogramas e imagen médica. El proyecto cae en el marco del desarrollo de la herramienta EDDL, un software que permite crear y manipular los modelos mencionados. El objetivo principal es hacer uso de la herramienta para afrontar problemas de aprendizaje automático con modelos del estado del arte.

: Aprendizaje profundo, Imagen médica, Redes neuronales, Electroencefalogramas (EEG), Reconocimiento de patrones

Abstract

The progress in the field of Artificial Intelligence has led us to a new research era. Deep Learning models can automatically detect and diagnose diseases in patients. In this project, Deep Learning techniques are applied to several use cases related to health. Neural Networks are used to build models which are able to face the different tasks in each use case. The tasks presented in this work are related to the analysis of electroencephalogram signals and medical imaging. The project was carried out under the development of the EDDL toolkit, a software that allows for creating and manipulating the mentioned models. The main goal is to use this toolkit to face machine learning problems with state-of-the-art models.

: Deep Learning, Medical Imaging; Neural Networks, Electroencephalogram (EEG), Pattern Recognition

Contents

Contents	iii
List of Figures	v
List of Tables	vi

1 Introduction	1
1.1 Motivation	2
1.2 Objectives	3
1.3 Structure of the document	4
2 Related work	5
2.1 Epilepsy detection	5
2.2 Covid-19 detection from chest X-ray images	6
3 European Distributed Deep Learning Library	9
3.1 EDDL: European Distributed Deep Learning Library	9
3.2 ECVL: European Computer Vision Library	10
4 Use Case 13: Introduction and Problem Analysis	11
4.1 Problem description	11
4.1.1 Epilepsy	11
4.1.2 Patient-Specific <i>vs.</i> Patient-Independent approaches	12
4.1.3 Seizure detection <i>vs.</i> Seizure prediction task	12
4.2 Description of the data	13
4.2.1 Splits	16
4.2.2 Class imbalance	16
4.3 Solution proposal	17
4.3.1 Data preparation	17
4.3.2 Recurrent approach	17
4.3.3 Convolutional approach	19
4.3.4 Post Inference process	21
4.3.5 Evaluation Metrics	21
5 Use Case 13: Experiments and results	25
5.1 First experiment: Selection of Best Model	25
5.1.1 Recurrent Approach	26
5.1.2 Convolutional Approach	27
5.2 Second Experiment: Test Selected Model	29
5.3 Third experiment: Post Inference Process	30
5.4 Discussion	32
6 Use Case 15: Introduction and Problem Analysis	35
6.1 Problem description	35
6.1.1 Defined tasks	36
6.2 Description of the dataset	36

6.3	Solution proposal	37
6.3.1	Preprocess	37
6.3.2	Neural Network Models	38
6.3.3	Data augmentation	40
7	Use Case 15: Experiments and results	41
7.1	Lung alignment	41
7.2	Detecting Pulmonary conditions	42
7.2.1	Covid-19 vs Healthy results	43
7.2.2	Multi-label task results	43
7.3	Discussion	44
8	Conclusions and future works	47
8.1	Conclusions	47
8.2	Future Work	48
	Bibliography	51

Appendix

A	Beyond Use Case 13	55
A.1	Autoencoders	55
A.2	Feature Extraction	55
A.2.1	Feature extraction process	56
A.3	Review on the prediction task	58

List of Figures

4.1	Epilepsy brain stages.	12
4.2	Seizure detection <i>vs</i> seizure prediction task. The seizure prediction aims to differentiate between interictal and preictal stages. The detection aims to distinguish interictal and ictal stages. Image extracted from [16].	13
4.3	The 10-20 International system of EEG electrode placement. Image extracted from [29].	15
4.4	This is a 10 seconds extract of an EEG recording from file 3 of patient chb01. On the left side we can observe the labels for the recorded channels. Snapshot taken from the LightWave visualizer of Physionet [18].	15
4.6	Dataset splits.	16
4.5	Extract of a summary file of patient chb01.	16
4.7	Recurrent model architecture. The recurrent layer type could be either LSTM or GRU. The input sequence has 19 timesteps, and the data dimension is 256 because we are using one-second-long periods. Each of the 23 channels that compose the signal is processed independently.	18
4.8	Training vs Inference process in Use Case 13. In the training phase, the 23 channels that compose the signal are passed independently and in the inference phase the results are combined.	19
4.9	Convolutional approach model architecture.	20
5.1	Experimentation with different window lengths in the post-inference function.	32
5.2	Experimentation with configurations A, B, C, and D for the hyperparameters α_{pos} and α_{neg} in the post-inference function.	32
6.1	BIMCV-COVID19+: Summary of data. Image extracted from the dataset publication [14].	36
6.2	Image preprocessing: All images were padded, resized (lanczos) and normalized using the Contrast Limited Adaptive Histogram Equalization (CLAHE).	38
6.3	Residual connection. The weight layer is the convolutional block composed of a convolutional layer and a Batch Normalization layer in our case. Image extracted from [19]	39
7.1	Lungs alignment: To align the lung images (cropping and centering), we first trained a U-net model to segment the lung regions. Then, we combined the bounding boxes of the segmented regions to extract the region of interest.	42

A.1	Hamming window shape.	56
A.2	Processed signal. The top figure shows the frequency of the original signal and the filtered signal. The figure in the middle shows the energy that falls on different filter banks. The bottom figure shows the time-domain statistics of the signal.	59

List of Tables

4.1	Summary of the available recordings.	14
5.1	Results in the test set with the GRU architecture, SGD optimizer and initial learning rate of $1e-5$	26
5.2	Results in the test set with the LSTM architecture, Adam optimizer and initial learning rate of $1e-4$	26
5.3	Results in the test set with the best GRU model in terms of balanced accuracy. This model was trained with SGD as optimizer and an initial learning rate of $1e-5$	27
5.4	Results in the test set with the GRU model trained with Adam as optimizer and an initial learning rate of $1e-5$	28
5.5	Results in the test set by the convolutional architecture. The optimizer used was Adam with an initial learning rate of $1e-6$	28
5.6	Results in the test set with the convolutional approach selected model.	29
5.7	Results of the neural network classifier for all of the patients in the dataset.	30
5.8	Results after the post-inference process with the selected model for all the subjects in the dataset.	31
6.1	Dataset partitions: Classes are not mutually exclusive.	37
7.1	Best results on the binary classification task.	43
7.2	Average results of five experiments obtained with the ResNet-18 model using different input sizes. Mean and standard deviation are shown.	43
7.3	Results with ResNet-101 on the multi-label task. The input image size used was 512×512	44

CHAPTER 1

Introduction

Artificial Intelligence has made significant steps in latter years. That is because of the field of Deep Learning, which is nowadays in the spotlight for many researchers worldwide. It is more relevant than ever because we can design models that learn by themselves from large amounts of data. Deep Learning is part of the Machine Learning techniques used to make computers learn from data. The results that are being reported in real-world tasks are making this field very important at this time. These techniques are being applied in many industries, thanks to the large amounts of data that can be collected nowadays.

Deep Learning systems are based on the use of neural networks, which have existed since the sixties. The improvement that made this field grow up a lot in recent years has been the computational power acquired by GPUs. Thanks to this, Deep Learning systems are being used for many real-world problems and different industries.

In the field of healthcare, data scientists are applying Deep Learning to many use cases, and some of the results that are being reported are better than human experts. One of the most popular applications is Medical Imaging, where different kinds of images are used to visualize the human body to diagnose, monitor or treat medical conditions. Some types of images that are being used are CT scans, MRI scans, X-rays and different images generated by medical devices such as microscopes. In the Medical Imaging field, researchers are approaching tasks such as lung analysis from X-rays, brain segmentation for multiple sclerosis detection, skin lesion analysis to diagnose skin cancer, and many more tasks. Apart from Medical Imaging, there are a lot of Deep Learning applications in healthcare, such as the analysis of Electronic Health Records, drug discovery, genomics analysis, and more. Deep learning experiments are cheap and can perform well on many tasks, sometimes better than human experts. This has raised the interest of many researchers, as they could develop systems that could play an essential role in medical diagnoses.

In this work, we are presenting two applications of Deep Learning on problems related to healthcare. The first one will approach the automatic detection of epileptic seizures through the analysis of electroencephalogram recordings. The second application will be a Medical Imaging task, where the aim will be to analyse lung X-ray images in order to detect covid-19 and other pathologies.

Both applications are use cases of DeepHealth [1], a project funded by the European Commission that aims to offer a unified framework with High-Performance Computing infrastructures combined with Deep Learning and Artificial Intelligence techniques to support biomedical applications. This final work has been carried out under the development of the EDDL library, a part of the DeepHealth project. The applications that we are presenting in this work are two use cases that were set to test the library. The work done in both use cases resulted in two papers: "*Automatic Detection of Epileptic Seizures with Recurrent and Convolutional Neural Networks*" [10] and "*Detection of Pulmonary Conditions Using the DeepHealth Framework*" [11]. These papers were presented at the DeepHealth Workshop at the International Conference on Image Analysis and Processing (ICIAP2021), celebrated in May 2022 in Lecce, Italy. The presented work has been carried out by a team of the Pattern Recognition and Human Language Technology (PRHLT) research centre [5], one of the teams that are working on the DeepHealth project.

1.1 Motivation

The progress in technology and medicine has been improving the quality of life of humanity over history. The 21st century is currently living a technological revolution, where Artificial Intelligence seems to be one of the most critical technologies for the future. This discipline is developing in giant steps, and the possibility of using Artificial Intelligence techniques to help individuals in healthcare has motivated many researchers to develop systems that could improve the quality of life of many people.

Epilepsy is a chronic non-communicable disease of the brain that affects around 50 million people worldwide. According to the World Health Organization, this disease is characterised by recurrent seizures, which are episodes of involuntary movement that involve parts of the body or even the entire body. These seizure episodes are unpredictable events; therefore, they affect the daily lives of the people who suffer them. Epilepsy patients are prone to suffer unexpected accidents and have increased stress due to the life conditions of suffering this disease. Early detection and immediate warning of these seizures could significantly improve the quality of life of these people. Additionally, detecting these seizures with high reliability could help to better understand this disease as well as open the door to improve existing treatments or computer-aided diagnosis to help neurologists.

The most common way of detecting seizures is through the analysis of the scalp Electroencephalogram (EEG), which is a non-invasive recording of the electrical activity of the brain. Many studies have demonstrated that the automatic detection and the automatic prediction of seizures through EEG signals is possible [17, 35, 25, 27]. The automatic detection of seizure events would significantly contribute to improving the quality of life of epileptic patients in many aspects. First, it will provide advantages to the patients such as avoidance of injuries, increasing the feeling of security, the possibility to drive a car, and reduction of anxiety [32]. Additionally, if a system was able to raise alarms before the onset of a seizure and provide enough time for intervention, it could open the door for

new treatment methods and strategies.

On the other hand, humanity is currently living through a pandemic caused by Covid-19. Covid-19 is a contagious disease caused by the SARS-CoV-2 virus. It was first detected in Wuhan, China, in December 2019 after diagnosing a group of people with pneumonia of unknown cause. The World Health Organization recognised this disease as a global pandemic on March 11, 2020 [6]. The number of cases kept growing and the virus spread out, reaching a total of 385 million cases in January 2022 and a total of 5.7 million deaths worldwide. This virus has changed our way of living in the last two years. From the beginning of the pandemic, the scientific community has made great efforts to help fighting against the problems arising from the pandemic, such as finding a vaccine or developing medicines to help patients with severe symptoms. In the field of machine learning, researchers have been developing systems to detect Covid-19 from X-ray images using the potential of Convolutional Neural Networks (CNNs). It might seem that there would be no reason to perform an X-ray scan on a potential Covid-19 patient with the success of PCR tests. However, it is common to perform chest X-ray scans on patients presenting respiratory problems in medical practice, so having an automated system for Covid-19 detection could save healthcare systems a lot of time and money.

Additionally to the consequences on the health of the people suffering from Epilepsy or Covid-19, another reason for accomplishing this project is to test the EDDL library to demonstrate that it is suitable to solve real-world problems.

1.2 Objectives

The main objective of this work is to develop solutions for two applications of Deep Learning to real-world problems in relation to healthcare: Epilepsy detection and Covid-19 and other pulmonary conditions detection.

On the one hand, in the Epilepsy detection task, the objective is to provide patient-specific classifiers capable of distinguishing between a normal state and a seizure state within an electrical signal from the brain. This task will involve a data processing part, model designing, and experimental part. In order to verify the performance of the models, the solutions will be compared with state-of-the-art results.

On the other hand, for the Covid-19 detection task, the aim is to build a system based on convolutional neural networks able to detect the lesions caused by Covid-19 in chest X-ray images.

To sum up, the goals of this project can be listed as follows:

- Use case 13: Epilepsy detection
 1. Implement a pipeline with models based on recurrent architectures for detecting seizure events in epilepsy subjects.
 2. Implement a pipeline with models based on convolutional architectures for detecting seizure events in epilepsy subjects.

3. Test the best models on different subjects and analyse the results.
- Use case 15: Covid-19 detection
 1. Apply preprocessing techniques to the images.
 2. Build classifiers based on state-of-the-art image classification architectures.
 3. Test the models and compare the results with state-of-the-art studies.

1.3 Structure of the document

This document is structured in eight chapters. In Chapter 2, we will give an overview of the recent works and studies on similar tasks as the use cases. In Chapter 3, we will introduce and describe the software library in which this project will be carried out. After that, there are two chapters for each use case. In Chapter 4, Use Case 13 is introduced, and the proposed solution is described. In Chapter 5, we describe the experiments carried out on the Use Case 13, as well as the results and a discussion. After that, in Chapter 6, we describe the Use Case 15 task and the proposed solution. In Chapter 7, we report the experimental results of the Use Case 15, and we give a discussion of the reported results. Finally, in Chapter 8 we conclude this report of the work, summarising the most important aspects of each part, as well as mentioning future works to explore.

CHAPTER 2

Related work

In this chapter, we present recent works that have been developed to solve tasks similar to the ones of the use cases of this work and the difficulties that have been found. First, we explain some of the current approaches used to solve the epileptic seizure detection task. After that, we will review the solutions that have been presented in recent years for the detection of Covid-19 and other pathologies from X-ray scans.

2.1 Epilepsy detection

Epilepsy detection using EEG signals has been approached with many different solutions, which generally rely on a first feature extraction and selection process and a machine learning classifier afterwards. Most approaches achieve high scores in detecting seizures in terms of sensitivity and specificity. One of the first solutions that were successful was a self-organized map (SOM) neural network to detect seizures in 24 long-term EEG recordings. It was presented in [17] in 1996 by Gabor et al., and they demonstrated the possibility of automatic seizure detection. In [31], discrete wavelet transform (DWT) was used in order to extract features from the EEG signals, and the method achieved a 76% of sensitivity in detecting seizures.

In [37], a novel algorithm for feature extraction called MinMaxHist is presented to describe the waveform characteristics of the spikes and sharp waves of the EEG signals in order to classify the events by using these kinds of features. The solution achieved 86.27% of accuracy in a patient-independent classifier. In [35], it was designed a feature vector with spectral features using filter banks to measure the energy falling within the passband of each filter. They classified the feature vectors into ictal or interictal using an SVM and generated non-linear boundaries with an RBF kernel. Overall, 96% of the 173 test seizures were detected with an average detection latency of 4.6 seconds. Finally, in [9] they trained a CNN with raw signals in order to get the probability of each time window being ictal. Afterwards, they proposed an onset-offset detector for determining the seizure onsets and offsets. The model correctly detected 90% of the seizures in the CHB-MIT Database. In [7], a deep convolutional autoencoder with a Bi-LSTM classifier achieved a 98.86% of accuracy in the CHB-MIT dataset, trained with

raw signal. Additionally, an undersampling technique was applied to the Interictal class samples in order to balance the data before feeding the network

Regarding the preprocessing techniques used in similar tasks, the signal is processed in most cases before using it in the classifiers. The typical way is to extract spatial and spectral features in order to create a feature vector to be classified by a machine learning based system. This work aims to show different perspectives of approaching the task with raw signal, without applying any preprocessing to the data.

2.2 Covid-19 detection from chest X-ray images

Due to the immense impact of Covid-19 and the necessity of quick and reliable screening methods, many studies have arisen to use X-ray images to provide a fast way to detect infected patients. Many deep learning approaches have been developed thanks to the power of convolutional neural networks to recognise patterns from the images and use such patterns to make predictions. CNNs are the dominant approach for almost all recognition and detection tasks [24].

Many works can be found in the literature achieving high accuracy in classifying chest X-ray images, usually in two (healthy vs. Covid-19) or three (healthy vs. pneumonia vs. Covid-19) class classification tasks. In Khan et al. [22], they used a novel architecture called STM-RENet combined with Channel Boosting, using two pre-trained models, to perform the two-class classification task achieving an accuracy of 96.53%. In de Moura et al. [15], some architectures well-known in Computer Vision and pre-trained with Imagenet (DenseNet-121 and DenseNet161; ResNet-18 and ResNet-34; VGG-16 and VGG-19) were used to build classification models to analyse different classification scenarios: Healthy vs Pneumonia, Healthy vs Pneumonia/Covid-19, Healthy/Pneumonia vs Covid-19, and Healthy vs Pneumonia vs Covid-19. They analysed the grade of separability of Covid-19 from Pneumonia and Healthy X-ray images, with satisfactory results for all the experiments and accuracy of 97.44% for the three-class classification task. In Kumar et al. [23], they propose a model called SARS-Net, a combination of a CNN with inception blocks and a graph convolutional network. It was used to solve the three-class classification task achieving an accuracy of 97.60%. The model predictions were analysed using GRAD-CAM (Selvaraju et al. [34]) to see heat maps on the input images with the most suitable pixels for the model in order to make the prediction. In Bhattacharyya et al. [8], they used first a Conditional GAN to segment the lungs; with the extracted lungs, they applied two feature extractors, a Pre-trained VGG19 and the feature detector algorithm BRISK. The extracted features were combined and fed to a final Random Forest classifier that makes the final prediction, achieving an accuracy of 96.6% in the three-class classification task. In Nayak et al. [26], they tested different popular CNN architectures (AlexNet, VGG-16, GoogleNet, MobileNet-V2, SqueezeNet, ResNet-34, ResNet-50, and Inception-V) pre-trained with the ImageNet dataset, and concluded that the ResNet-34 was the one achieving the best results with an accuracy of 98.33% in the two-class classification task. In Qi et al. [28], they

followed a semi-supervised approach with a Teacher-Student architecture to perform the three-class classification task achieving an accuracy of 93%. The model used had two inputs, where one of them was the original CXR image, and the other was a three-channel image where each channel was a filtered version of the original image.

One of the problems of working with medical data is the lack of images. The available datasets are usually small, and it is challenging to train robust models with a good generalisation capacity. A crucial technique to deal with the lack of data is applying Data Augmentation during the training phase. In the works analysed here, the common practice is to apply affine transformations like Rotation, Flip, Scaling, and Shear. Furthermore, in some cases, other transformations are also applied: Gaussian Noise in Nayak et al. [26]; Elastic Transform, CutBlur, MotionBlur, Intensity Shift, and CutNoise in Kumar et al. [23].

From the analysed works, it can be seen that researchers usually take more than one dataset to create a more extensive dataset for training; for example, in Nayak et al. [26], they picked the COVID-19 images from one dataset and the healthy samples from another one. This kind of combination can introduce a bias that the model can use just for classifying the samples depending on specific characteristics of each dataset and not original medical-related patterns of Covid-19. In [12], authors analyse various Covid-19 datasets and discuss the kinds of biases that can be present when using these datasets. This practice could be hazardous on these use cases and even more when the samples of each class come from different datasets.

CHAPTER 3

European Distributed Deep Learning Library

In this chapter, we will briefly introduce the software that will be used in this work.

3.1 EDDL: European Distributed Deep Learning Library

As we introduced in Chapter 1, this work has been carried out under the development of the European Distributed Deep learning Library (EDDL). It is part of the DeepHealth project, and it is an open-source library for Distributed Deep Learning and Tensor operations for CPU, GPU and FPGAs.

The library is written in C++, and it also provides a Python wrapper called `pyEDDL`, which offers the possibility for the programmer to use Python. This library is being developed by a team of the Pattern Recognition and Human Language Technology (PRHLT) research centre, and the source code and documentation are publicly available on the GitHub repository of the project¹.

EDDL provides multiple layers in order to build state-of-the-art Deep Learning architectures, from Dense layers to 2D and 3D convolutional layers. It also provides many loss functions and metrics. Optimizers like SGD, Adam, Adagrad, RMSProp and others are also implemented.

Not only layers, but the EDDL also provides many Tensor operators, which let the user decide how to use the library. It has different levels of abstraction, and the user can work from a high level of abstraction with built-in high-level functions to a lower level with Tensor manipulation and atomic functions. For example, to train a neural network, the user can use the `fit()` function, which passes the entire dataset through the network and trains it for a specific number of epochs. Nonetheless, another possibility is to use the `train_batch()` function to perform the training (forward and backward) of a batch of data. However, if the user prefers to have more control over the execution, there are low-level functions such as

¹EDDL GitHub: <https://github.com/deephealthproject/eddl>

forward(), *backward()* to perform the training of the network in a more controlled way, in combination with Tensor operations.

Regarding distributed training, EDDL can work on different GPUs in the same machine and different machines, but an extra infrastructure is needed for the latter case. When a model is created, it can be compiled for different devices. The user can select if the model will work on CPU, GPU, or FPGA. For the GPU option, there are two compilations of the library, the EDDL implementation for GPUs and the CUDNN implementation, which uses Nvidia cuDNN accelerated library for Deep Learning.

EDDL library provides some pre-trained models that can be loaded in a line of code. It also can import and export models. The standard for these models is ONNX [4], a widely known neural network standard to define models.

All of these features make EDDL a complete library for Deep Learning, adjustable for different types of users and perfect for High Performance Computing, thanks to the possibility of distributed training. All of the experiments that are carried out in this work are developed using the EDDL toolkit. Although the library can train in distributed environments, all the experiments carried out for this work were performed on a single machine, using a single GPU, as the purpose of these use cases was to test the libraries and validate that the DeepHealth toolkit is suitable for solving real-world problems.

3.2 ECVL: European Computer Vision Library

In this work, we will also use the ECVL library, which is a part of the DeepHealth toolkit as well. It is a Computer Vision library that provides many operators for classic Computer Vision and provides a lot of Data Augmentation techniques implemented that will be used in this work.

ECVL library is also written in C++, and it provides a Python wrapper called pyECVL for Python users. It can work with different image types, and it is a powerful tool for working with images. In order to support the EDDL, it has implemented a data loader called DLDataset, which can load lots of images very fast, using parallel workers. The source code and documentation can be found on the GitHub repository².

²ECVL GitHub <https://github.com/deephealthproject/ecvl>

CHAPTER 4

Use Case 13: Introduction and Problem Analysis

In this chapter, we will introduce Use Case 13 and give a detailed description of the dataset and the different proposed solutions to solve the task.

4.1 Problem description

Epilepsy is a chronic neurological disorder characterised by recurrent seizures. Epileptic seizures can vary from brief and nearly undetectable periods to long periods of vigorous shaking due to abnormal electrical activity in the brain. The Use Case 13 of the DeepHealth project is focused on automatic epileptic seizure detection through the analysis of Electroencephalogram signals (EEG).

4.1.1. Epilepsy

Epilepsy is a chronic non-communicable disease of the brain that affects people of all ages. This disorder is characterized by recurrent seizures, episodes when the affected subject has involuntary movement that can involve parts of the body or the whole body. Seizures can vary from short lapses of attention to prolonged convulsions. Around 50 million people around the world are diagnosed with epilepsy. It is one of the most common neurological diseases, and it is estimated that up to 70% of the cases could live seizure-free if they had a proper diagnosis and treatment [2].

An Electroencephalography is a method to record the electrical activity of the brain. An Electroencephalogram (EEG) is the resulting recording, which is a signal formed by different channels. These channels are collected by computing the difference between potentials measured at two electrodes. There are two main ways of recording Electroencephalogram signals: the intracranial EEG and the scalp EEG. The intracranial EEG collects the brain activity by electrodes placed on the brain surface, while the scalp EEG signals are collected by placing electrodes on the scalp. This second method is noninvasive and easier to perform, but the signals can be corrupted by artefacts and noises. Meanwhile, intracranial EEG is more accurate, but it has important risks for the patient.

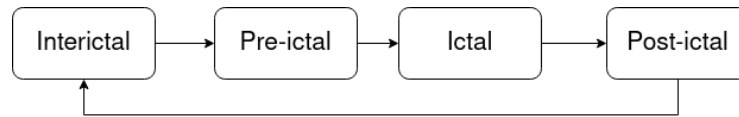


Figure 4.1: Epilepsy brain stages.

An epileptic seizure has commonly four stages. The interictal stage is a period with regular brain activity, where the patient is not having any epileptic symptoms. The preictal stage is the period between the interictal stage and the seizure onset. It is the time before the epileptic seizure begins. This period can vary from minutes to hours, so it is challenging to determine it. There could be some strange brain activity alerting that a seizure is about to happen in this stage. The ictal stage is when the patient suffers the seizure, and the brain activity in this period is quite abnormal. The patient has obvious physical symptoms at this stage. Finally, after the seizure ends, there is a postictal period, when the patient is recovering, and the brain activity starts to be normal again. It usually lasts several hours. Then, the brain activity returns to the interictal stage. We can see the brain stages cycle in Figure 4.1.

The characteristics of EEG vary significantly across patients. In fact, the patterns associated with a seizure onset for one patient may be similar to a normal stage of another patient. This causes patient-independent classifiers to have poor accuracy or long delays in detecting seizures. However, patient-specific seizure detection and prediction remain challenging. Patients with epilepsy have considerable overlap in the EEG signal associated with normal and seizure periods. The EEG is also constantly transitioning between stages in a non-stationary process. Additionally, since seizures are rare events, algorithm designers must create solutions that work with limited seizure data. These particularities complicate the task of detecting and also predicting epileptic seizures.

4.1.2. Patient-Specific *vs.* Patient-Independent approaches

Epileptic seizure detection or prediction can be solved using two approaches: a patient-specific or a patient-independent approach. On the one hand, patient-specific approaches construct a classifier for each patient, considering the high inter-subject variability. In these studies, a single architecture is defined, and it is fine-tuned for each subject. The used data is not mixed between the classifiers of different patients. The performance is measured by averaging the accuracy for all of the patients.

On the other hand, patient-independent approaches aim to design a classifier that can detect or predict seizures of any patient. In these approaches, the entire dataset is used to train the classifier, which has to learn a global function able to work with data from any patient.

4.1.3. Seizure detection *vs.* Seizure prediction task

EEG signals are being used for solving different problems, such as seizure detection or seizure prediction. The seizure detection task aims to detect where the

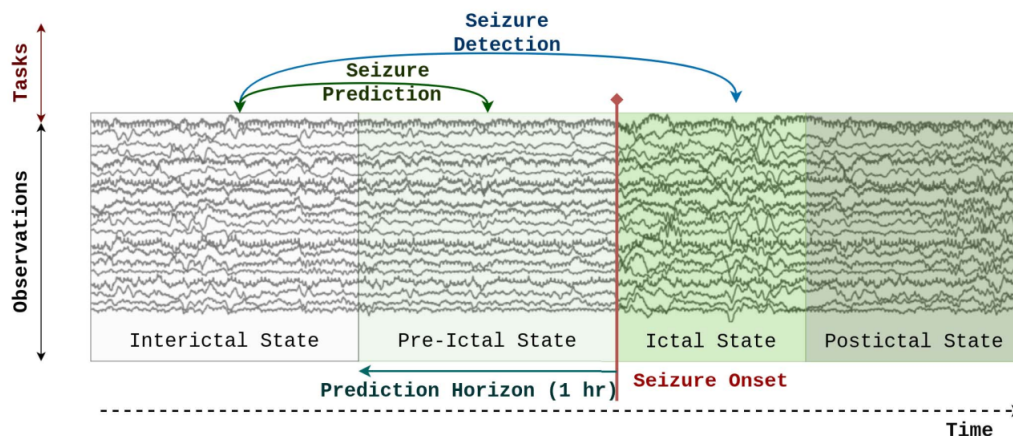


Figure 4.2: Seizure detection *vs* seizure prediction task. The seizure prediction aims to differentiate between interictal and preictal stages. The detection aims to distinguish interictal and ictal stages. Image extracted from [16].

seizures happen in a recording. It is a binary classification task between the ictal periods and the rest.

The main objective of the seizure prediction task is to distinguish between the interictal periods and the preictal ones. The ictal and postictal stages are not involved in this task because there is no need to detect them to predict a seizure. In fact, we only need to determine when the signal has a preictal period, which is the first stage after a long normal activity period that would become in a seizure event. The differences between both tasks are illustrated in Figure 4.2.

In this use case, we are focusing on the detection task. Early detection and immediate warning of the seizures could significantly improve the quality of life of the patients. In addition, detecting these seizures with high reliability could help better understand the disease as well as open the door to improve existing treatments or computer-aided diagnosis to help neurologists.

4.2 Description of the data

In this use case, we will work with the CHB-MIT scalp EEG database, published in Physionet [18]. The dataset comprises 24 sessions of EEG recordings from 23 pediatric subjects from the Children’s Hospital of Boston. Subjects were monitored for up to several days following withdrawal of anti-seizure medication in order to characterize their seizures and assess their candidacy for surgical intervention. Each session was recorded on one subject, and the additional one was captured from subject *chb01* a year and a half after the first recording, so it is treated as a different case, resulting in 24 cases (or patients).

Each session contains between 9 and 42 continuous signal files. Most of the files last 1 hour, but there are recordings of 4 hours in a single file for some patients. Due to hardware limitations, there are gaps (of around 10 seconds) be-

Table 4.1: Summary of the available recordings.

<i>Patient id</i>	# seizures	# interictal hours	# ictal hours	# seizures per hour
chb01	7	40.43	0.12	0.17
chb02	3	35.22	0.05	0.09
chb03	7	37.89	0.11	0.18
chb04	3	155.96	0.11	0.02
chb05	5	38.85	0.16	0.13
chb06	8	66.69	0.04	0.12
chb07	3	66.96	0.09	0.04
chb08	5	19.75	0.26	0.25
chb09	3	67.79	0.08	0.04
chb10	7	49.90	0.12	0.14
chb11	3	34.57	0.22	0.09
chb12	16	20.41	0.28	0.77
chb13	8	10.88	0.12	0.73
chb14	7	25.95	0.05	0.27
chb15	17	38.46	0.55	0.44
chb16	6	16.98	0.02	0.35
chb17	3	19.93	0.08	0.15
chb18	6	34.54	0.09	0.17
chb19	3	28.86	0.07	0.10
chb20	6	27.52	0.08	0.22
chb21	4	32.77	0.06	0.12
chb22	3	30.95	0.06	0.10
chb23	4	26.44	0.12	0.15
chb24	13	21.15	0.14	0.61
Total	150	948.85	3.08	5.46

tween files. Table 4.1 summarises the number of hours recorded for every patient and the corresponding number of seizures.

The dataset contains digitized scalp EEG recordings, and the signals were captured at 256hz with a 16-bit resolution. The format of the files is EDF (European Data Format), a standard file format designed for the exchange and storage of medical time series. The signals are composed of many channels, each one related to a pair of electrodes located on the scalp. The electrodes were located following the international 10-20 system, as shown in Figure 4.3. For this task, 23 channels were used, which are the ones that repeat the most in the dataset, ignoring some extra channels in some cases and files with fewer channels. Figure 4.4 is a 10 seconds extract from a signal of the dataset.

The labels for the files are indicated in summary files, one per patient. On the summary file, we can find the name of the recorded channels, the start and end time of each recording, the number of seizures that occur and the start and end time of the corresponding seizures. A summary file for a patient looks like in Figure 4.5.

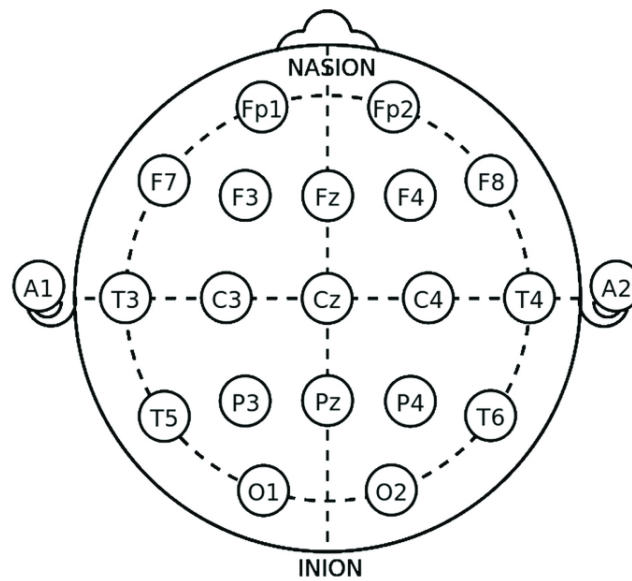


Figure 4.3: The 10-20 International system of EEG electrode placement. Image extracted from [29].

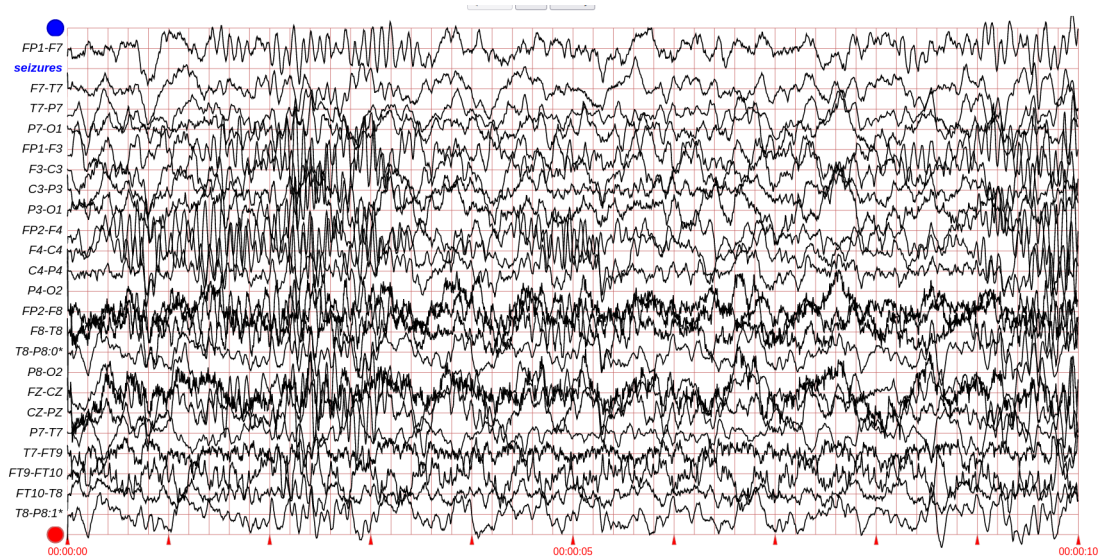


Figure 4.4: This is a 10 seconds extract of an EEG recording from file 3 of patient chb01. On the left side we can observe the labels for the recorded channels. Snapshot taken from the LightWave visualizer of Physionet [18].

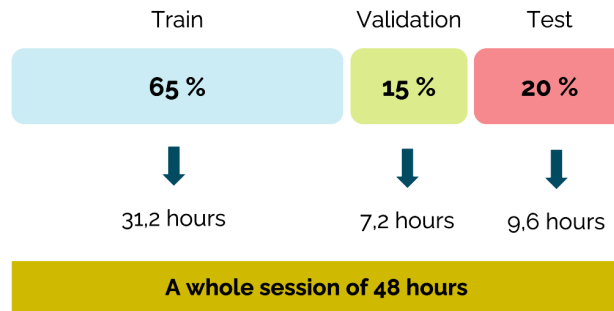


Figure 4.6: Dataset splits.

```

File Name: chb01_03.edf
File Start Time: 13:43:04
File End Time: 14:43:04
Number of Seizures in File: 1
Seizure Start Time: 2996 seconds
Seizure End Time: 3036 seconds

```

Figure 4.5: Extract of a summary file of patient chb01.

4.2.1. Splits

The dataset was divided into three splits: training, validation and test. The training split will be used for training the neural networks, and the validation set will be used to check how the model is learning and stop the training process before overfitting the network. Finally, the test set will be used by the model with the best hyperparameters.

For this purpose, the dataset was split using 65% of the recordings for training, a 15% for validation and the 20% left for testing. This was done for each subject independently, so, for instance, if the whole session of a subject has a total of 48 hours, there will be 31.2 hours in the training set, 7.2 hours in the validation set and 9.6 hours in the test set.

As the data is split into recordings (of one-hour-long mostly), the recordings were divided into two groups: recordings with seizures and recordings without seizures. Then, the splits were generated, trying to keep the same proportion of each class. Figure 4.6 shows a summary of the splits.

4.2.2. Class imbalance

One of the most critical problems when approaching such a task is the high class imbalance. As seizures usually last a few seconds, the number of seizure samples compared to non-seizure samples (interictal class) is excessively low. If we look at Table 4.1, we can see the difference between the number of interictal hours and

the number of ictal hours. This high imbalance makes this task difficult because it leads to overfitting in the training phase.

In order to deal with the class imbalance, we applied undersampling to the interictal class and oversampling to the ictal samples, in order to obtain a balanced dataset (only for the training process). When generating each batch, we used half of the samples of one class and the other half of the other. The interictal samples (the majority class) were selected randomly at each epoch, so some of them were excluded. The ictal samples were repeated to balance the dataset. In order to deal with overfitting problems, we used some techniques such as Dropout layers in the neural network models or Gaussian Noise added to the input data in the way of data augmentation.

4.3 Solution proposal

Our solution for this task involves a classifier based on neural networks and a post inference process that will act as a seizure detector. For each one of the patients, a classifier will be trained only with data from the same patient, following a patient-specific approach. The first classification part will be tackled with two different architectures: recurrent neural networks and convolutional neural networks.

4.3.1. Data preparation

Both the approaches work with raw signals as input, but the signals were processed before using them. This process transforms EDF files into Python dictionaries containing the signals and the information about the seizures. It also homogenizes the data. As we are using 23 channels and there are files with more or fewer channels, this process only keeps the desired channels and excludes the ones that will not be used. At the end of this process, we have a standardised dataset with the same channels on every recording and easy to load by the data loaders to feed our neural networks.

In both approaches, a custom data loader was built. This data loader is an object that loads the data and prepares it to fit the input of the networks. These data loaders will also perform a Z-score normalization, in order to have zero mean and a standard deviation of one. This normalization is essential because the values of the data have high variability. For instance, the range of values can vary between -2200.73 and 2403.13 for subject *chb01*.

4.3.2. Recurrent approach

This first approach is based on recurrent neural networks. These types of networks can handle temporal data and find temporal dependencies. The input of the network will be a sequence of data from the signal. On each step, the network will output the class of the last sample of the sequence, so in each time step, we will get a prediction.

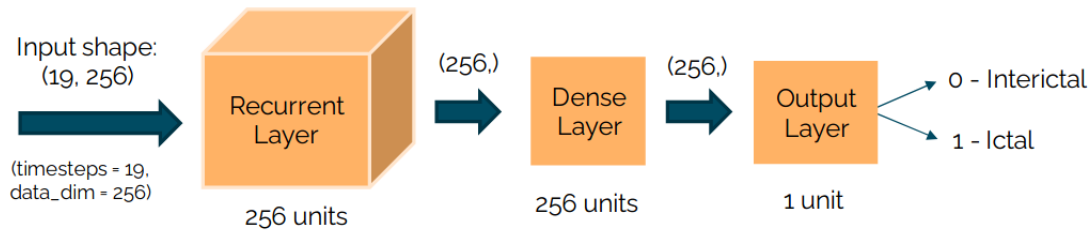


Figure 4.7: Recurrent model architecture. The recurrent layer type could be either LSTM or GRU. The input sequence has 19 timesteps, and the data dimension is 256 because we are using one-second-long periods. Each of the 23 channels that compose the signal is processed independently.

Model architecture

In this approach, two recurrent models were tested: one based on Long Short Term Memory (LSTM) layers and one based on Gated Recurrent Units (GRU) layers. The architecture of the models is the same, except for the recurrent layer type. The inputs for this approach are sequences of one-second-long periods extracted from raw signals. This input data goes through a recurrent layer (LSTM or GRU) of 256 units, followed by a fully-connected part with a Dense layer of 256 units and the output layer with one unit and a Sigmoid activation function. The output layer will provide an approximation of the posterior probability of the input to be ictal. In Figure 4.7, we can observe a diagram of the recurrent model architecture. As we can see, it is a noncomplex model, composed of a single recurrent layer and a Dense layer afterwards.

Data Loader: Sequences

Before feeding the recurrent neural networks, the data was prepared in sequences. For this purpose, a data loader was built for loading the data and preparing batches of sequences. To generate each sequence, a one-second-long sliding window is shifted through the signal every 500ms, in such a way that every single sample corresponds to a period of one second of the signal, which is overlapped 50% with its surrounding previous and next samples. Each time step, the network looks at the last 10 seconds of the signal in order to predict the current state. In this approach, the channels that compose the signal are processed independently. They are passed one by one by the same network, and the results are combined to make the prediction at each step.

The data loader applies the oversampling and undersampling techniques, providing balanced batches when training the model, so each batch has the same proportion of samples of each class.

Training vs Inference

When the network is fed with data, two processes can be distinguished: the training and inference processes. The training process is when the neural network is fed with the training subset, providing pairs of [sample, label] so the network

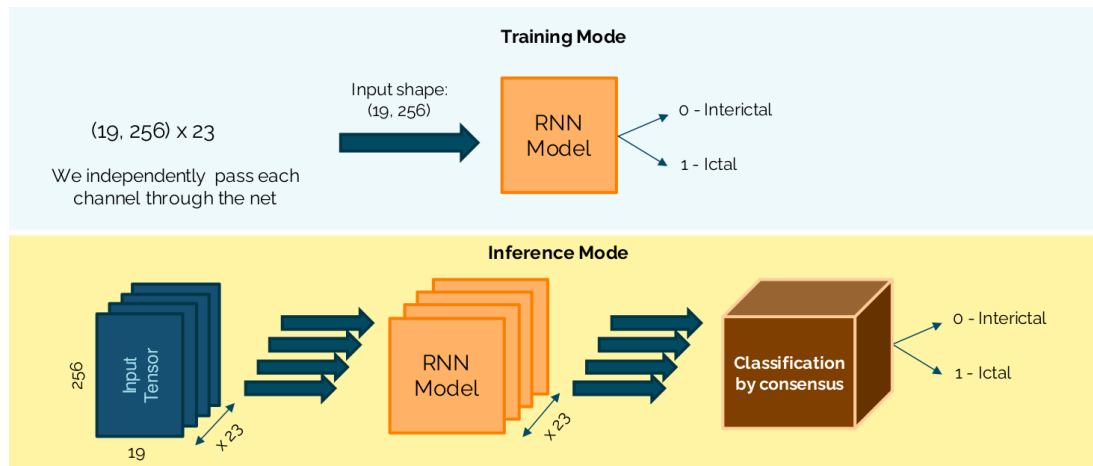


Figure 4.8: Training vs Inference process in Use Case 13. In the training phase, the 23 channels that compose the signal are passed independently and in the inference phase the results are combined.

weights are modified to associate the input sample with the label. In the inference process, only the input samples are provided, so the validation or test set samples are passed through the network to "inference" the class they correspond to (in a classification scheme). In this recurrent approach, as the data is a signal composed of many channels, these two processes are done as follows.

In the training process, the network is trained with each channel independently, so at each time step, the channels are passed one by one. In the inference process, all signal channels are passed one by one, but the results are combined, and the classification of that sample is done with the combination of the channels. As the output of the network can be interpreted as the probability of the sample belonging to class ictal, the outputs of the channels are added and normalized in order to get a single output. A summary of these two processes can be observed in Figure 4.8.

4.3.3. Convolutional approach

The second approach to solve this task is based on convolutional neural networks. These networks use convolutional layers, which can learn spatial and temporal dependencies from the input data. They are usually used with images for tasks such as classification or segmentation. In this task, we treat periods of the signal as "images" and the convolutional layers perform the feature extraction process to classify these periods posteriorly in a fully-connected part.

As we extract periods to create samples, our data has two dimensions: a dimension with the different channels that compose the signal, where it is possible to extract spatial features, and a temporal dimension, where it is possible to extract temporal features.

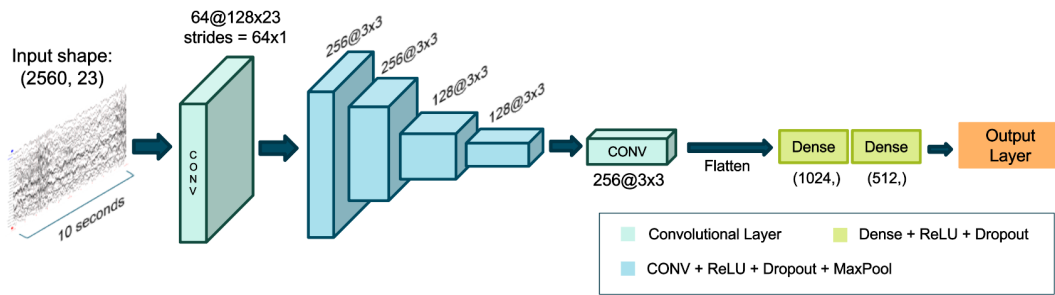


Figure 4.9: Convolutional approach model architecture.

Data Loader

In order to load the data for this approach, another data loader was built. In this approach, the samples are generated by sliding a temporal window of 10 seconds through the signal, shifting it each 250 milliseconds. It extracts periods of 10 seconds that are highly overlapped. As the data was acquired at 256 Hz, the shape of the samples will be 2560x23, corresponding to a period of 10 seconds long and the 23 channels. In this case, no sequence is generated, and the label for each sample is the label of the last value inside the window. This data loader also provides balanced batches when training.

Model architecture

The model of this approach starts with a convolutional layer that performs the convolution along the temporal axis, similar to a Time Delay Neural Network (TDNN). This first convolution layer has a kernel size of 128x23, and it has strides of 64x1. These values for kernel size and strides make the convolution operation to be applied horizontally through the input data, in the temporal axis, taking the information of all of the 23 channels at once.

After the first convolution, there are more convolutional layers, with 3x3 kernels and strides of 1x1, that extract more relevant features and reduce the dimensionality. These latter convolutions are followed by ReLU activations, Dropout layers and MaxPooling layers. The ReLU activation is essential because it can avoid the gradient vanishing problem; the Dropout layer helps the model not overfit because of the problems derived from the data imbalance, and the Max-Pooling layer reduces the size of the feature maps. These groups of layers form convolutional blocks, which are repeated four times. After these blocks, there is a last convolutional layer, followed by fully-connected part with two Dense layers of 1024 and 512 units, followed by Dropout layers. Finally, the 512 units layer connects with the output layer of 1 unit, followed by a Sigmoid activation function. A summary of the model architecture can be observed in Figure 4.9.

In order to prevent overfitting, an L2 regularization with a factor of $1e-5$ is applied on each convolutional and Dense layer, so the weights of the network are penalized. Additionally, the input data is modified with Gaussian Noise in the training phase, acting as a data augmentation technique.

4.3.4. Post Inference process

After the neural network models do the classification part, a post inference process is done to act as a detector of seizures. It is a function that analyses the consecutive outputs of the network and raises alarms when it detects a seizure.

The post inference function is based on a finite-state machine of two states: interictal and ictal. As the outputs of the classifier are ones or zeros (one for Ictal, zero for Interictal), this function analyses the predictions and makes transitions between the states, so it can raise alarms when a seizure onset is detected. It works by sliding a window over the predictions, so it can analyse the last predictions in order to know the current state of the brain (interictal or ictal). There are two possible transitions between the states:

- If the current state is Interictal and the proportion of ones inside the window is greater than a parameter α_{pos} , then it makes a transition from Interictal to Ictal.
- If the current state is Ictal and the proportion of ones inside the window is lower than a parameter α_{neg} , then it makes a transition from Ictal to Interictal.

In addition, there are two more intra-state transitions, so in the case that none of the previous conditions is met, the function keeps transitioning in the same state in a loop.

The function has four parameters that can be tuned to perform the detection: the size of the analysis window, given in time steps; α_{pos} , the minimum ratio of positive predictions required inside the window to trigger a transition from Interictal to Ictal; α_{neg} , the maximum ratio of positive predictions required inside the window to trigger a transition from Ictal to Interictal; and the detection threshold, which is the number of seconds that the model will be allowed to detect a seizure, so if it detects it outside the detection threshold it will be counted as a false alarm because it is a late detection.

To sum up, this function performs as a seizure detector, which raises an alarm when it detects a new seizure. This process is done after the inference of the test samples of each subject.

4.3.5. Evaluation Metrics

In this use case, two groups of metrics will be evaluated. One group is associated with the neural network classifier, and the other is associated with the post inference process.

When training and testing the neural network models, the evaluated metrics are accuracy, macro f1-score and balanced accuracy.

Accuracy

The accuracy is the proportion of predictions that the model did correctly among the total number of predictions and can be defined in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Where TP, FP, TN and FN refer to True Positives, False Positives, True Negatives and False Negatives, respectively.

Macro F1-score

The F1-score is a metric that considers both precision and recall, so it is a very helpful metric to evaluate a classifier. It can be defined as follows:

$$\text{F1-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

This metric is calculated per class. Therefore, to compute the Macro average F1-score, it is as simple as calculating the arithmetic mean (unweighted mean) of the metric for each class.

This Macro averaging was selected for this task because it treats all classes as equal independently of the number of samples they have. A metric that gives the same contribution to each class is handy for unbalanced data when the aim is to perform well in each class.

Balanced Accuracy

Balanced Accuracy avoids inflated performance estimates on unbalanced datasets. It is the accuracy where each sample is weighted according to the inverse prevalence of its true class, or the average recall obtained on each class [3]. For the binary case, it can be defined as follows:

$$\text{Balanced Accuracy} = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$$

Post Inference Metrics

When the neural network has been trained, and the test set samples have been already predicted, the post inference process is performed, as it has been described. In this process, a set of event-based metrics is calculated to evaluate the detection performance. These metrics are the following:

- **Percentage of correctly detected seizures:** Proportion of the seizures that were correctly detected, similar to the recall associated to the detection of events.
- **Average latency of detection:** Average time in seconds that the model needs to detect the seizures.
- **False Alarms per Hour:** Number of false alarms raised by the model per hour.

CHAPTER 5

Use Case 13: Experiments and results

This chapter presents the experiments that were carried out on Use Case 13. After the experiments section, there is a discussion of the presented results.

To begin with the experiments, a subset of eight patients was selected, in order to find which one of the approaches performed better, as well as to find the best training configurations. This was done because of resources and time limits, as each configuration had to be trained and tested for each patient specifically. This subset was composed by patients *chb01*, *chb03*, *chb05*, *chb08*, *chb12*, *chb14*, *chb15* and *chb24*. The group of patients was selected trying to have a very diverse set of patients, taking patients with different number of seizures per hour.

After the best model and configuration were selected, it was tested with all of the subjects in the second experiment. In the end, an extra experiment was performed to analyze different configurations of the post inference process parameters.

Regarding the hardware used, the experiments were performed on different machines from the Pattern Recognition and Human Language Technology research centre. The available hardware was:

- 1 Machine with 2 Nvidia GTX 1080 8Gb GPUs
- 3 Machines with 2 Nvidia RTX 2080 8Gb GPUs each.

As it has been explained, the experiments were performed in a sequential environment, only using one machine and GPU at a time.

5.1 First experiment: Selection of Best Model

In this first experiment, both approaches were trained and tested with different configurations using the subset of eight patients.

5.1.1. Recurrent Approach

In the recurrent approach, two architectures were tested: the LSTM based and the GRU based. Optimizers Adam and SGD were tested, with an initial learning rate ranging from $1e-4$ to $1e-6$. The best results in the test set for the GRU architecture were obtained using SGD as the optimizer and an initial learning rate of $1e-5$. The best results for the LSTM architecture were obtained using Adam as optimizer and an initial learning rate of $1e-4$. Tables 5.1 and 5.2 summarize these results.

Table 5.1: Results in the test set with the GRU architecture, SGD optimizer and initial learning rate of $1e-5$.

Patient id	Model	Accuracy	Macro F1-Score	Balanced Accuracy
<i>chb01</i>	GRU	96.34%	0.6039	95.98%
<i>chb03</i>	GRU	92.10%	0.5206	94.97%
<i>chb05</i>	GRU	95.42%	0.5618	96.21%
<i>chb08</i>	GRU	82.18%	0.5153	86.29%
<i>chb12</i>	GRU	65.60%	0.4274	67.43%
<i>chb14</i>	GRU	87.28%	0.4760	86.41%
<i>chb15</i>	GRU	89.50%	0.5907	89.96%
<i>chb24</i>	GRU	97.49%	0.6701	92.32%
Average		88.24%	0.5457	88.70%

Table 5.2: Results in the test set with the LSTM architecture, Adam optimizer and initial learning rate of $1e-4$.

Patient id	Model	Accuracy	Macro F1-Score	Balanced Accuracy
<i>chb01</i>	LSTM	99.91%	0.9573	92.40%
<i>chb03</i>	LSTM	99.64%	0.5034	50.21%
<i>chb05</i>	LSTM	99.89%	0.9234	86.75%
<i>chb08</i>	LSTM	99.38%	0.8714	80.84%
<i>chb12</i>	LSTM	97.92%	0.6142	59.06%
<i>chb14</i>	LSTM	99.86%	0.5728	53.95%
<i>chb15</i>	LSTM	97.73%	0.6165	59.75%
<i>chb24</i>	LSTM	99.62%	0.8422	76.12%
Average		99.24%	0.7377	69.89%

As we can see in Tables 5.1 and 5.2, the models achieve high scores for Accuracy, especially the LSTM based model. This score is very optimistic because the dataset is highly unbalanced. In the balanced accuracy column, it is observed that the GRU model performs better than the LSTM for all of the subjects. For subjects *chb01*, *chb03*, *chb05* and *chb24*, the GRU model achieved more than 90% of balanced accuracy. Regarding the F1-score, for the GRU model, the scores are lower than in the LSTM in general, but the LSTM scores could be a bit inflated because of the high performance in the majority class observed in the experiments. Therefore, the balanced accuracy score is the best to look at in this case.

After inferring the test set samples with the GRU model, which has performed better in terms of balanced accuracy, it achieves the results shown in Table 5.3. In

general, the model detected 80.76% of the seizures, with an average latency of 8.11 seconds. For patients *chb01*, *chb03*, *chb05*, *chb08* and *chb24*, the model detected all of the seizures of the test set. The main drawback observed in these results is the high rate of false alarms per hour, with an average of approximately 26 false alarms per hour. This means that the model detects most of the seizures, but it has many false alarms.

Table 5.3: Results in the test set with the best GRU model in terms of balanced accuracy. This model was trained with SGD as optimizer and an initial learning rate of $1e-5$.

Patient id	No. Seizures	Detected Seizures	Average Latency (s)	False Alarms per Hour	Test Hours
<i>chb01</i>	2	100.00%	8.25	12.06	9.62
<i>chb03</i>	2	100.00%	4.50	20.73	8.97
<i>chb05</i>	1	100.00%	1.50	13.04	7.98
<i>chb08</i>	1	100.00%	14.00	48.27	4.99
<i>chb12</i>	11	81.82%	7.72	51.81	5.98
<i>chb14</i>	2	50.00%	7.00	38.82	6.98
<i>chb15</i>	7	14.29%	17.00	23.06	8.98
<i>chb24</i>	4	100.00%	4.88	4.14	6.28
Average		80.76%	8.11	26.49	7.47

After testing another GRU based model, it was found that it had better results in the test set after the post inference process for some patients, even though the neural network scores were lower. It was the GRU model trained with Adam as optimizer and an initial learning rate of $1e-5$. The results after the post inference process can be seen in Table 5.4. As it can be observed, this model detects fewer seizures in the test set (61.12%) with four more seconds of average latency, but it has a big difference in terms of false alarms. This model has an average of 1.32 false alarms per hour compared to the 26.49 false alarms of the other one. These results are definitively better than the ones shown in Table 5.3, even though the model could not detect any seizure for patients *chb03* and *chb05* in the test set. The results generally show a significant reduction in the number of false alarms, with a slight increase in the latency and a lower detection rate.

5.1.2. Convolutional Approach

In the convolutional approach, only one architecture has been tested, using different configurations in order to find the best results. The experiments have been testing optimizers as Adam or SGD, and initial learning rates from $1e-3$ to $1e-6$. As in the recurrent approach, the model has been tested first with a subset of patients. After training the neural network classifier with the different subjects and configurations, the best results achieved by the network in the test set can be seen in Table 5.5.

In these experiments with the convolutional approach, something similar happens in the results as with the recurrent models. In Table 5.5, the accuracy column shows very high scores, but as it has been mentioned, these are very optimistic. In the balanced accuracy column, it is observed that the model achieves more

Table 5.4: Results in the test set with the GRU model trained with Adam as optimizer and an initial learning rate of $1e-5$.

Patient id	No. Seizures	Detected Seizures	Average Latency (s)	False Alarms per Hour	Test Hours
<i>chb01</i>	2	100.00%	11.25	0.10	9.62
<i>chb03</i>	2	0.00%	-	0.11	8.97
<i>chb05</i>	1	0.00%	-	0.13	7.98
<i>chb08</i>	1	100.00%	16.00	1.00	4.99
<i>chb12</i>	11	81.82%	10.83	7.02	5.98
<i>chb14</i>	2	50.00%	19.00	0.00	6.98
<i>chb15</i>	7	57.14%	7,63	2.01	8.98
<i>chb24</i>	4	100.00%	8,38	0.16	6.28
Average		61.12%	12.18	1.32	7.47

Table 5.5: Results in the test set by the convolutional architecture. The optimizer used was Adam with an initial learning rate of $1e-6$.

Patient id	Accuracy	Macro F1-Score	Balanced Accuracy
<i>chb01</i>	99.69%	0.8850	96.52%
<i>chb03</i>	99.75%	0.8719	99.13%
<i>chb05</i>	99.95%	0.9694	95.09%
<i>chb08</i>	99.41%	0.8926	87.03%
<i>chb12</i>	97.53%	0.6815	70.41%
<i>chb14</i>	96.50%	0.4959	53.84%
<i>chb15</i>	97.66%	0.7905	95.74%
<i>chb24</i>	99.66%	0.8805	84.73%
Average	98.77%	0.8084	85.31%

than 95% on patients *chb01*, *chb03*, *chb05* and *chb15*. It is remarkable that for patient *chb03* this model achieved a 99.13% of balanced accuracy, the highest score obtained for any patient thus far. If we compare these results with the ones obtained in the recurrent approach, this model achieves higher scores for five out of the eight patients.

After the post inference process, the obtained results are presented in Table 5.6. In this case, the average detection rate is 86.12%. The model has detected all of the seizures for five subjects. The average latency is 8.79 seconds, and a low false alarm rate of 2.33 on average is observed. In fact, for subject *chb05*, the model detected the only seizure in the test set with a latency of 6.50 seconds and no false alarms in a total of almost 8 hours. Even though this model achieved a lower balanced accuracy in the neural network classifier than the recurrent model, it achieved better results in the post inference process. Therefore, this was the model selected for further experiments.

Table 5.6: Results in the test set with the convolutional approach selected model.

Patient id	No. seizures	Detected Seizures	Average Latency (s)	False Alarms per Hour	Test Hours
<i>chb01</i>	2	100.00%	8.50	0.62	9.62
<i>chb03</i>	2	100.00%	2.88	0.56	8.98
<i>chb05</i>	1	100.00%	6.50	0.00	7.98
<i>chb08</i>	1	100.00%	17.00	1.20	4.99
<i>chb12</i>	11	81.82%	12.81	3.51	5.98
<i>chb14</i>	2	50.00%	9.50	9.31	6.98
<i>chb15</i>	7	57.14%	6.75	2.78	8.98
<i>chb24</i>	4	100.00%	6.38	0.64	6.28
Average		86.12%	8.79	2.33	7.47

5.2 Second Experiment: Test Selected Model

After testing both approaches with a subset of patients, the convolutional model achieved the best results. Thus, it was the model selected for testing again with all of the subjects of the dataset. The corresponding neural network classifier results after training and testing are presented in Table 5.7.

Regarding the neural network classifier results, the accuracy scores are very high, as has been happening with all the experiments so far. The model achieved an F1-score of 0.7371 and a balanced accuracy of 76.77% on average for the 24 subjects. For nine subjects, the model achieved more than 90% of balanced accuracy. However, poor results are observed in six patients, with balanced accuracy values of approximately 50%. Values around 50% of balanced accuracy suggest that the model is not learning anything or almost anything, as it could be predicting all the samples to the same class. Nonetheless, there are nine subjects with results between 60% and 90% of balanced accuracy that could perform well in the post inference process.

After the post inference process with all of the subjects, the obtained results are shown in Table 5.8. In general, the model has detected 56.67% of the seizures, with an average latency of 9.51 seconds and 2.15 false alarms per hour on average. As it can be seen in the table, most of the patients only have one or two seizures in many hours in the test subset, so the percentage of detected seizures is very sensitive, with extreme observable values of 0% and 100%, which makes the average not very meaningful, even though it is weighted to the number of seizures of each patient.

There are nine patients for whom the model could not detect any seizure in the test subset and ten patients with a 100% of detected seizures. If we look at the 15 patients for whom the model could detect at least one seizure, the average rate of detected seizures is 85.93, the average latency is 9.51 seconds, and the average rate of false alarms per hour is 2.31. Moreover, there are remarkable results for patients *chb02*, *chb05*, *chb09* and *chb10*, with a 100% of detected seizures and no false alarms. Finally, the best results in terms of accurate and fast detection were obtained with patients *chb03* and *chb10*: with 3.12 seconds of latency and 0.22 false alarms per hour in the case of patient *chb03*, and with 4.88 seconds of latency and

Table 5.7: Results of the neural network classifier for all of the patients in the dataset.

Patient id	Accuracy	Macro F1-Score	Balanced Accuracy
<i>chb01</i>	99.70%	0.8868	96.59%
<i>chb02</i>	99.98%	0.7985	77.02%
<i>chb03</i>	99.84%	0.9072	97.80%
<i>chb04</i>	95.74%	0.5097	74.45%
<i>chb05</i>	99.95%	0.9682	94.88%
<i>chb06</i>	97.90%	0.4947	48.98%
<i>chb07</i>	99.88%	0.7816	69.63%
<i>chb08</i>	99.40%	0.8909	87.16%
<i>chb09</i>	99.98%	0.9539	91.57%
<i>chb10</i>	99.98%	0.9795	97.99%
<i>chb011</i>	99.66%	0.9607	92.85%
<i>chb12</i>	97.61%	0.6860	70.54%
<i>chb13</i>	98.51%	0.7466	68.96%
<i>chb14</i>	96.46%	0.4955	53.49%
<i>chb15</i>	97.63%	0.7877	95.47%
<i>chb16</i>	99.82%	0.4996	50.00%
<i>chb17</i>	99.26%	0.5581	54.95%
<i>chb18</i>	96.17%	0.5575	90.14%
<i>chb19</i>	99.85%	0.8606	83.51%
<i>chb20</i>	98.64%	0.5746	72.21%
<i>chb21</i>	98.13%	0.4967	52.15%
<i>chb22</i>	99.89%	0.8797	81.48%
<i>chb23</i>	99.47%	0.4987	50.00%
<i>chb24</i>	99.74%	0.9162	90.73%
Average	98.88%	0.7371	76.77%

no false alarms in the case of *chb10*. All the seizures of these two patients in the test subset were detected (2 seizures each).

5.3 Third experiment: Post Inference Process

Finally, after selecting the best model and testing it with all of the patients, an extra experiment was done in order to analyse the performance of the post inference process function by varying some of its parameters.

First, the length of the analysis window used in the post-inference function was in the range of 10 to 80 predictions. This length indicates how many predictions from the neural network classifier has to analyze to predict the current state. As in the case of the convolutional approach, predictions are performed every 0.25 seconds, so a window length of 80 will analyze the last 20 seconds to decide on switching the state (from interictal to ictal or vice-versa). The hyperparameters α_{pos} and α_{neg} were set to 0.4, the same values used in all of the previous experiments. The results of this experiment can be observed in Figure 5.1. It can be noticed that as the analysis window is longer, the latency increases and the

Table 5.8: Results after the post-inference process with the selected model for all the subjects in the dataset.

Patient id	No. seizures	Detected Seizures	Average Latency (s)	False Alarms per Hour	Tested Hours
<i>chb01</i>	2	100.00%	8.38	0.62	9.62
<i>chb02</i>	1	100.00%	6.25	0.00	8.98
<i>chb03</i>	2	100.00%	3.12	0.22	8.98
<i>chb04</i>	2	0.00%		6.86	35.16
<i>chb05</i>	1	100.00%	7.00	0.00	7.98
<i>chb06</i>	2	0.00%		3.46	15.03
<i>chb07</i>	1	0.00%		0.05	19.34
<i>chb08</i>	1	100.00%	17.00	1.40	4.99
<i>chb09</i>	1	100.00%	9.50	0.00	13.62
<i>chb10</i>	2	100.00%	4.88	0.00	13.98
<i>chb11</i>	1	100.00%	5.00	0.91	8.77
<i>chb12</i>	11	81.82%	12.72	3.18	5.98
<i>chb13</i>	4	50.00%	11.75	1.00	2.99
<i>chb14</i>	2	50.00%	9.75	9.45	6.98
<i>chb15</i>	7	57.14%	7.06	2.78	8.98
<i>chb16</i>	4	0.00%		0.00	4.99
<i>chb17</i>	1	0.00%		1.80	4.99
<i>chb18</i>	2	100.00%	11.12	4.57	8.98
<i>chb19</i>	1	0.00%		0.25	7.90
<i>chb20</i>	2	50.00%	14.00	3.84	7.54
<i>chb21</i>	1	0.00%		6.77	7.98
<i>chb22</i>	1	0.00%		0.29	6.98
<i>chb23</i>	4	0.00%		0.00	12.88
<i>chb24</i>	4	100.00%	6.31	0.32	6.28
Average		56.67%	9.51	2.15	10.00

number of false alarms is reduced. This happens because the model can observe more predictions in order to predict if there is a seizure, so it is more confident in the predictions, but also slower because the latency is affected. The percentage of detected seizures is not much affected by these changes.

The second experiment was devoted to see the effect of the hyper-parameters α_{pos} and α_{neg} on the performance. The following configurations were tested:

- Configuration A:** $\alpha_{pos} = \alpha_{neg} = 0.2$
- Configuration B:** $\alpha_{pos} = \alpha_{neg} = 0.4$
- Configuration C:** $\alpha_{pos} = \alpha_{neg} = 0.8$
- Configuration D:** $\alpha_{pos} = 0.8, \alpha_{neg} = 0.2$

Figure 5.2 shows the results for configurations A, B, C and D with a fixed window length of 20 predictions. It can be observed that the configurations can vary the performance in terms of latency and false alarms. As higher are the values of α_{pos} and α_{neg} , the function needs more positive predictions inside the analysis window in order to raise an alarm, therefore the alarms are more delayed, and

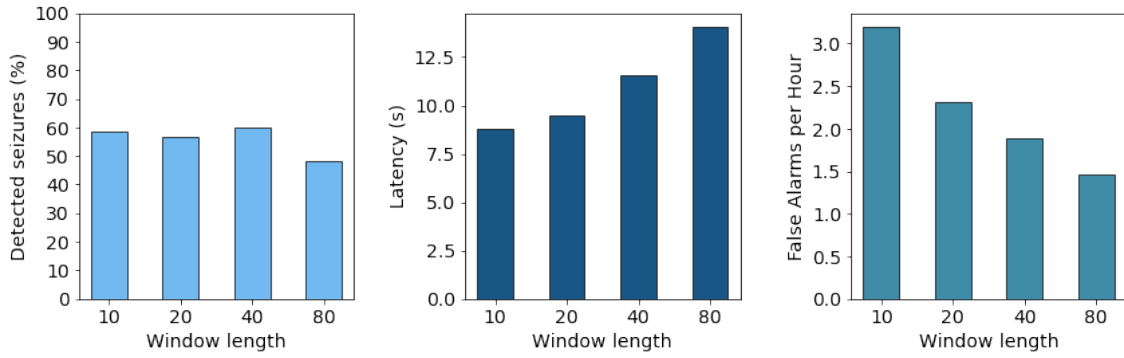


Figure 5.1: Experimentation with different window lengths in the post-inference function.

the false alarms are reduced because the model is more confident. Regarding the detection rate, it does not seem to be affected by these variations either.

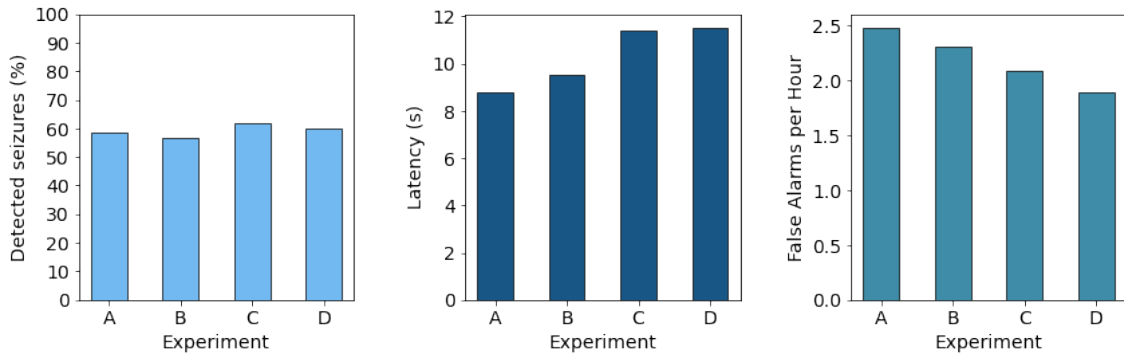


Figure 5.2: Experimentation with configurations A, B, C, and D for the hyper-parameters α_{pos} and α_{neg} in the post-inference function.

5.4 Discussion

The process of designing and implementing the proposed solution has had many steps. There was not a straightforward solution strategy at first, so different solutions were proposed, and after the phase of the experiments, the solution that has been presented is the one that performed better.

First of all, the most considerable difficulty of this task, which consequences have been present during all the experiments, is the class imbalance of the samples. As it has been analysed, seizures are events that are short and occur occasionally. It means that if we want to classify between seizure periods and normal ones, we will have to deal with a small amount of seizure data, in the sense of the duration and number of periods available. When splitting the dataset into three subsets, ictal periods were distributed in the training, validation and test subsets without mixing samples from the same seizure in different splits. There were eight subjects with only three seizures in the whole session, so one seizure went to a different subset. In these cases, the difficulty of the task shows up when trying to train with only one seizure in the training set. Also, the results may

not be generalist when only considering one seizure in the test set. A different training and evaluation strategy such as performing K-fold cross-validation or leave-one-recording-out cross-validation could be a better approach. However, we could not consider these strategies for this work because of time limits.

The small number of ictal samples leads the training phase to problems like overfitting. There is a lot of interictal data but only a few samples of ictal periods. On each epoch, the data loaders provided the network with batches with half of the samples of each class. As there were less ictal samples, these were repeated various times during an epoch. These clones of the same samples lead to overfitting problems, even that Gaussian Noise was added to the input data in the convolutional approach to slightly modify the samples. Although the dataset was balanced thanks to oversampling and undersampling techniques, the models tended to overfit in a few epochs.

Another drawback observed was that the values coming from the EDF signal had a significant variance that makes the neural network training so challenging. Even though the values were normalized in order to have zero mean and unit variance, there were still some extreme values not too appropriate to use as input for the neural networks. At this point, a limit was set to the data, so the accepted values were between $[-20, 20]$. This limit was necessary in order to achieve some acceptable results.

When defining and implementing a detector, engineers experience with a tradeoff between good detection and fast detection. This tradeoff has been observed during the experimentation part of this presented work. A high-quality detection and a rapid alarm when detecting a seizure are needed to consider a classifier acceptable. The mentioned tradeoff can be reached when fitting the classifier and also in the parameters of the post inference function, the one that analyses the predictions of the network and raises alarms when it detects a seizure. As it has been seen, some parameters can be adjusted to focus on a quality detection or a fast detection. When focusing on a fast detection, the quality will be lower because more seizures will not be detected, and there will be more false alarms. When focusing on the quality of the detection, the number of detected seizures is higher and the false alarms are reduced, at the cost of having a higher latency. This concept is quite interesting in order to deploy the models in a real scenario, so that the parameters could be adjusted to improve the effectiveness.

Regarding the benefits of the presented work, the experimentation process in this use case has shown a model capable of detecting seizures correctly in the case of some patients. The classifier achieves high and good scores for some patients, e.g. patients *chb03* and *chb10*, showing that it can work well even with a small amount of training data of the ictal class. In general, the model detected a **56.67%** of the test seizures for all of the patients, with an average latency of **9.51** seconds and **2.15** false alarms per hour on average. In comparison with state-of-the-art works, these results are far from approaching the reported results in [35] and [9], with a 96% and a 90% of detected seizures in the CHB-MIT dataset, respectively. Despite that, in [9] they report an F1-score of 0.2947 on the CNN classifier, and the mean F1-score of our classifier has been 0.7371, which is quite better. Nevertheless, the followed training strategies are different, and the test splits are not the same. In fact, they use a leave-one-record-out cross-validation scheme and

train with all of the recordings except one recording with seizures that is left for testing every time. Therefore, the results may not be comparable.

Our solution is not a general model for every patient; in fact, it is a patient-specific model. However, the method for training and adjusting for each patient is a general method, which can be used to fit the models to any other patient. In the experiments, the idea was to select the best model in general for every patient, because when training in a patient-specific scheme, the model should be the same and should be fitted with the data of each patient. However, models with other training configurations achieved better results than the selected one for some single patients but worse in general, so their results were not shown. Furthermore, one considerable fact of the proposed solution of this work is that the classifiers work with raw EEG data, which is not preprocessed at all, and any feature extraction technique is applied. This is important because it makes the system easy to implement and faster for the potential use for processing signals in real-time.

Finally, due to the amount of work done in this use case, we want to mention other techniques and solutions that were tested to solve this task, as they could be interesting and helpful to understand better the task. These additional solutions tested are not presented because the results achieved so far are not significative. They are summarized in appendix [A](#), where an autoencoder approach and a feature extraction process are described, and a review of the seizure prediction task is presented.

CHAPTER 6

Use Case 15: Introduction and Problem Analysis

This chapter introduces the second task that will be addressed in this work and the proposed solution.

6.1 Problem description

Covid-19 is a contagious disease caused by the SARS-CoV-2 virus. Since the beginning of the pandemic, the virus spread out, reaching a total of 385 million cases in January 2022 and a total of 5.7 million deaths worldwide. Diagnosis of Covid-19 is associated with the symptoms of pneumonia; therefore, Chest X-ray scans play an essential role in the diagnosis of Covid-19. Nevertheless, the standard way to diagnose the disease is the use of Reverse Transcription-Polymerase Chain Reaction (RT-PCR) tests. These tests have become relatively faster, but they still cause a high risk to medical staff. Moreover, this kind of test is costly and it exists a limited number of test kits.

With the success of RT-PCR tests, there might be no reason to perform an X-ray scan on a potential Covid-19 patient. However, it is common to perform chest X-ray scans on patients presenting respiratory problems in medical practice. Medical imaging techniques such as X-ray or Computed Tomography are relatively safe, faster and easily accessible. In comparison with CT scans, X-ray scans require less imaging time, have a lower cost, and their scanners are available in practically every hospital. Despite that, the visual inspection of X-ray images by radiologists is time-consuming and may lead to an inaccurate diagnosis due to the lack of knowledge about the infected regions caused by the virus. Therefore, having an automated system for Covid-19 detection using X-ray images could save healthcare systems a lot of time and money.

The Use Case 15 of the DeepHealth project focuses on detecting Covid-19 and other pulmonary conditions from chest X-ray images. The solution for this task is based on Neural Networks as classifiers for these medical images.

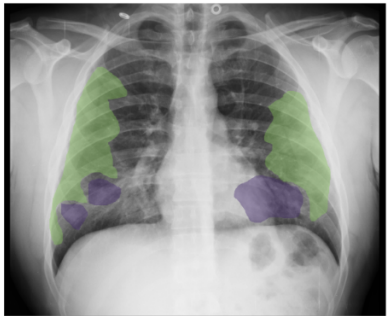
	<p>Report: opacidades de aspecto intersticioalveolar parcheadas y bilaterales que predominan en ambos lóbulos inferiores sospechosas de infección por covid19 . senos costofrenicos libres .</p> <p>Labels: COVID 19, alveolar pattern, interstitial pattern, pneumonia</p> <p>Locations: costophrenic angle, lobar, bilateral, lower lobe</p>																												
<table border="1"> <thead> <tr> <th>DICOM Fields</th> </tr> </thead> <tbody> <tr> <td>Study Date 20200317</td> </tr> <tr> <td>Patient's Sex M</td> </tr> <tr> <td>Patient's Birth Date 1986</td> </tr> <tr> <td>Modality CR</td> </tr> <tr> <td>Manufacturer Carestream Health</td> </tr> <tr> <td>...</td> </tr> </tbody> </table>	DICOM Fields	Study Date 20200317	Patient's Sex M	Patient's Birth Date 1986	Modality CR	Manufacturer Carestream Health	...	<table border="1"> <thead> <tr> <th>Date</th> <th>Test</th> <th>Result</th> </tr> </thead> <tbody> <tr> <td>17.03.2020</td> <td>PCR</td> <td>NEGATIVE</td> </tr> <tr> <td>18.03.2020</td> <td>PCR</td> <td>NEGATIVE</td> </tr> <tr> <td>19.03.2020</td> <td>IGG</td> <td>POSITIVE</td> </tr> <tr> <td>19.03.2020</td> <td>IGM</td> <td>POSITIVE</td> </tr> <tr> <td>20.03.2020</td> <td>PCR</td> <td>POSITIVE</td> </tr> <tr> <td>...</td> <td>...</td> <td>...</td> </tr> </tbody> </table>	Date	Test	Result	17.03.2020	PCR	NEGATIVE	18.03.2020	PCR	NEGATIVE	19.03.2020	IGG	POSITIVE	19.03.2020	IGM	POSITIVE	20.03.2020	PCR	POSITIVE
DICOM Fields																													
Study Date 20200317																													
Patient's Sex M																													
Patient's Birth Date 1986																													
Modality CR																													
Manufacturer Carestream Health																													
...																													
Date	Test	Result																											
17.03.2020	PCR	NEGATIVE																											
18.03.2020	PCR	NEGATIVE																											
19.03.2020	IGG	POSITIVE																											
19.03.2020	IGM	POSITIVE																											
20.03.2020	PCR	POSITIVE																											
...																											

Figure 6.1: BIMCV-COVID19+: Summary of data. Image extracted from the dataset publication [14].

6.1.1. Defined tasks

In this use case, two different tasks were defined. The first one was a binary classification task between “normal” versus “covid19” samples. In this task, the idea is to train classifiers to distinguish between an X-ray scan from a healthy patient and a scan from a patient with Covid-19. With this, we wanted to explore if the covid19 samples were separable from healthy lungs.

The second task was defined as a multi-label classification task. For this purpose, we used samples from four classes and tried to classify the images into one or more classes. With this task, the idea was to experiment with the detection of different pathologies at once with the same classifier.

6.2 Description of the dataset

The dataset that was used in this use case is the BIMCV-COVID19 dataset [14]. It is a large dataset composed of chest X-ray images (CXR) and Computed Tomography (CT) imaging of Covid-19 patients. It contains radiographic findings, Polymerase Chain Reaction (PCR) results, immunoglobulin G (IgG) and immunoglobulin M (IgM) diagnostic antibody tests and radiographic reports. Images were extracted from the DICOM format and stored as 16-bit PNG images. The scans come from the Medical Imaging Databank in Valencian Region Medical Image Bank (BIMCV). In Figure 6.1, we can observe a summary of the available information of the dataset.

The dataset is divided into two splits, the BIMCV-COVID19+, which contains the data of COVID-19 patients, and the BIMCV-COVID19-, which contains the data of non-COVID-19 patients. There are one or more sessions available for each patient in the dataset, with one or more X-ray images on each session. Every

patient has a list of pathologies that can be used as labels. There are 190 different labels distributed in an unbalanced way. We considered just four labels because they were the most repeated ones and they were related to Covid-19: “covid19”, “normal”, “pneumonia” and “infiltrates”. These four labels were used for preparing the two classification tasks: “normal” vs “covid19” on the one hand, and a multi-label task with the four labels on the other hand.

From all the available images, we selected just the ones with Anterior-Posterior or Posterior-Anterior views, excluding lateral views. We collected 2,083 samples and divided them into three partitions: 1,667 for training, 208 for validation, and 208 for testing. The partition was made at a patient level to avoid having images from the same patient in different splits. The distribution of samples inside each split is shown in Table 6.1. As the classes are not mutually exclusive, there are images with more than one label; for instance, an image can be labelled as “pneumonia” and “infiltrates”.

Table 6.1: Dataset partitions: Classes are not mutually exclusive.

Label	Train	Validation	Test
Covid	592 (33.95%)	77 (32.69%)	76 (35.10%)
Normal	219 (43.79%)	29 (43.27%)	29 (38.46%)
Pneumonia	730 (35.51%)	90 (37.02%)	80 (36.54%)
Infiltrates	566 (13.14%)	68 (13.94%)	73 (13.94%)
Total images	1667 (100%)	208 (100%)	208 (100%)

6.3 Solution proposal

For this image recognition task, we will use Convolutional Neural Networks (CNNs), which are outstanding in many image classification tasks. These CNNs will act as feature extractors and be combined with fully-connected layers at the end of the model to perform the classification.

6.3.1. Preprocess

Before feeding the neural networks, the images were padded and resized to the desired sizes. The images were prepared at two resolutions: 512×512 and 256×256 , as the original size was too large to manage (ranging from 2500 to 4000px, height and width). After that, the contrast was improved by applying the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm. This algorithm can improve the contrast of the image without amplifying or boosting the noise. An example of this is illustrated in Figure 6.2.

The preprocess was extended with a lung alignment method that was used to centre the lungs in the images. This preprocessing step was part of the experimental stage, so it will be described in the next chapter.

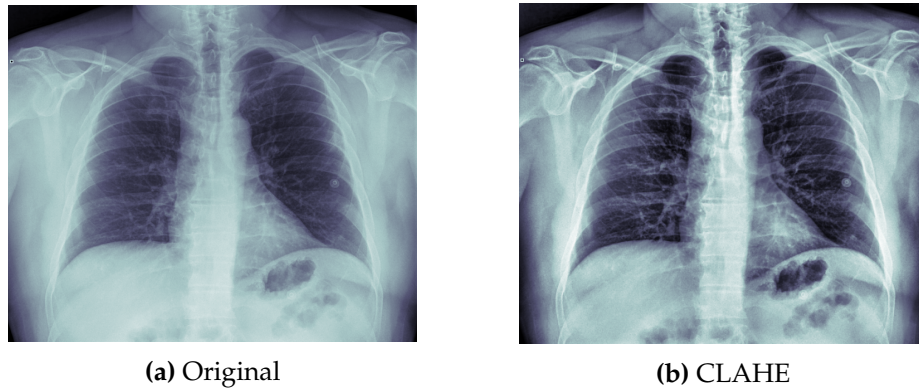


Figure 6.2: Image preprocessing: All images were padded, resized (lanczos) and normalized using the Contrast Limited Adaptive Histogram Equalization (CLAHE).

6.3.2. Neural Network Models

We trained Convolutional Neural Networks (CNNs) on this image recognition task, which are outstanding in many image classification tasks. Multiple neural network topologies were tested. Specifically, we made use of Imagenet pre-trained models such as VGG-{16, 19}, VGG-{16, 19} with Batch Normalization and ResNet-{18, 50, 101, 152}. These models were downloaded from the ONNX hub and loaded into the EDDL library for training and inference. From each neural model, we added on its top a new classifier with a dense layer with half the number of input features as units, followed by a ReLU activation, a Dropout layer, and finally, the output layer with the activation function to perform the classification on each task. For the binary classification task, the classifiers were prepared to experiment with a two units output layer followed by a Softmax activation function, and also to experiment with a single unit output layer followed by a Sigmoid activation function. For the multi-label classification task, the output layer had four units, and a Sigmoid activation function was applied element-wise after each unit.

Regarding the use of Transfer Learning, it is a common way to improve the classification results. The idea is to use a model that has been trained for another task with a large dataset of images, and fit it to perform on this task. For that purpose, the common way is to use the feature extraction part of the model and add new fully-connected layers in order to adapt the architecture for the current task. Then, the model is trained by freezing the feature extractor weights to train only the new part. After some epochs, the weights are unfrozen, and the model keeps training for some more epochs. Experiments on many tasks have proven that pre-trained weights improve training performance and stability.

Some of the models tested are Residual Networks, which are Convolutional Neural Networks with skip connections between some layers. These are called Residual connections, and a block of some convolutional layer blocks with a residual connection is called a Residual Block. This is the main idea of Residual nets. We can find a brief definition of these concepts in the following subsections. These architectures are the most popular ones in the recent studies on similar tasks, as we reviewed in Chapter 2.

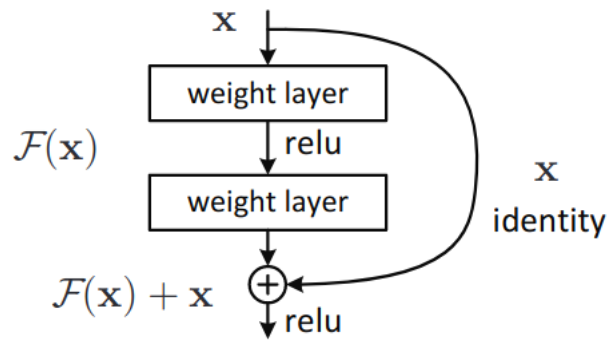


Figure 6.3: Residual connection. The weight layer is the convolutional block composed of a convolutional layer and a Batch Normalization layer in our case. Image extracted from [19]

Convolutional Block

A convolutional block is a stack of layers, containing a convolutional layer, a batch normalization layer and an activation function. There are different modifications of the convolutional block, such as using Batch Normalization and the activation before the convolutional layer. However, we used the following order: a convolutional layer, a Batch Normalization layer and the activation function. The activation function used was the Rectified Linear Unit, which helps to avoid the gradient vanishing problem and makes the models to learn faster and perform better.

Residual Block

A residual block is the composition of two or more convolutional blocks. A residual connection at the end connects the input of the first layer of the first convolutional block and the output of the last layer of the last convolutional block. At the connection, the residual part (the input) is added to the output of the residual block, as we can see in Figure 6.3.

Skip / Residual Connection

A skip connection also called a Residual connection, connects some convolutional blocks. There are different types of building this connection, depending on the order of the used layers. In this work, we used the original skip connection presented in [20], which adds the input of the residual block with the output of the batch normalization of the last convolutional block, right before the activation. After adding the residual part, the activation is performed, as shown in Figure 6.3.

Residual Network

A Residual Network usually starts with a first convolution with a bigger receptive field (7x7 for the original ResNet models). After this first convolutional layer,

some residual blocks are connected to form the feature extraction part of the network. In this part, the size of the feature maps is reduced, either using pooling layers or with convolutional layers with strides of 2x2. In our case, as we used the original ResNet models, the reduction of the feature maps in the residual blocks is done by using strides of 2x2 in the convolutional layers. Finally, after the feature extraction part, there is a fully connected part, which makes the classification.

The main benefit of these networks is the possibility to train much deeper neural networks, as the gradient can be propagated backwards without vanishing, thanks to the residual connections.

6.3.3. Data augmentation

Another technique that was applied is Data Augmentation. This technique helps the model be more generalist by slightly modifying the training images to have a larger dataset. We applied some basic augmentation techniques, such as small rotations and horizontal flips. These types of augmentations are very recommended for this task because, in real scenarios, we can find X-ray scans in Anterior-Posterior or Posterior-Anterior positions, and the lungs can have a small orientation. Additionally, more techniques were applied, such as Brightness augmentation and Gamma Contrast augmentation. As the brightness and contrast could be different in the images because they come from different hospitals and, therefore, different scanners, these Brightness and Gamma Contrast augmentations are also recommended, as they will add generalization power to the classifiers, and do not modify the structure or the morphology of the lungs. The techniques are applied in many of the previous works on similar tasks, as analysed in Chapter 2.

The parameters of the augmentation techniques are the following:

- **Horizontal Flip:** Applied with a probability of 0.5
- **Rotation:** A random degree in the range [-15, 15]
- **Brightness:** Beta value in the range [0, 70]
- **Gamma Contrast:** Gamma in the range [0.6, 1.4]

These augmentation techniques were applied using the ECVL library. This library can load many images simultaneously in parallel threads, and apply data augmentation techniques on-the-fly. It is straightforward to apply these techniques with this library by defining a container object with all the augmentations that will be applied. This container can be defined independently for each subset (training, validation and test).

CHAPTER 7

Use Case 15: Experiments and results

This chapter presents the experiments carried out with Use Case 15. First of all, we present a method for lung alignment that was explored and used to extend the preprocessing stage. After that, we present the experiments carried out regarding the detection of pulmonary conditions on both of the prepared tasks: binary classification and multi-label classification.

Regarding the hardware used, the experiments were performed on the same machines as in Use Case 13, and were performed in a sequential environment, using only one machine and GPU at a time.

7.1 Lung alignment

The idea of this first experiment came out because of the necessity of using the maximum resolution possible when processing our images with neural networks. The position and orientation of the lungs were quite variable, so in order to improve the quality of our input data, we decided to crop the lungs regions, excluding the information outside the region of interest and centring the lungs in our images. This process of lung alignment was done by detecting the lungs first and then cropping the bounding boxes containing the lungs centred.

The detection of the lungs was done in a semi-automatic process. We first manually segmented the lungs of 250 X-ray images using the CVAT [33] tool. After that, a neural network based on a U-Net architecture was trained with the manually annotated data in order to segment the remaining X-ray scans. This process followed a *human-in-the-loop* strategy to reduce human effort, and thanks to it, we could efficiently segment the lungs of 2,083 X-ray images, obtaining an IoU score of 0.95. The training strategy started with a *weak* segmentation model, trained on a small amount of manually labelled data. After training the first model, more remaining images were automatically segmented. The training set was extended with the new images and annotations, using only the properly segmented ones and excluding the images with unexpected results. After some iterations, we finally segmented the hard negatives manually to boost the performance of our model.

Regarding the neural network model, the architecture used was the U-Net presented in [30], since it is a well-known and widely used neural architecture for image segmentation tasks. As feature extractors, we experimented with different models pre-trained on ImageNet, such as VGG-16, ResNet34, ResNet50, ResNet101 and InceptionV3. However, since most of them offered similar results, we finally chose the ResNet34 architecture as it required fewer parameters than the larger models.

Finally, as this experiment had high-quality results, we included this method in the preprocessing of the images. Therefore, the original X-ray scans were first segmented and cropped into the region of interest, and after that, the histogram equalization and the resize were performed. An example of the results of this process is illustrated in Figure 7.1.

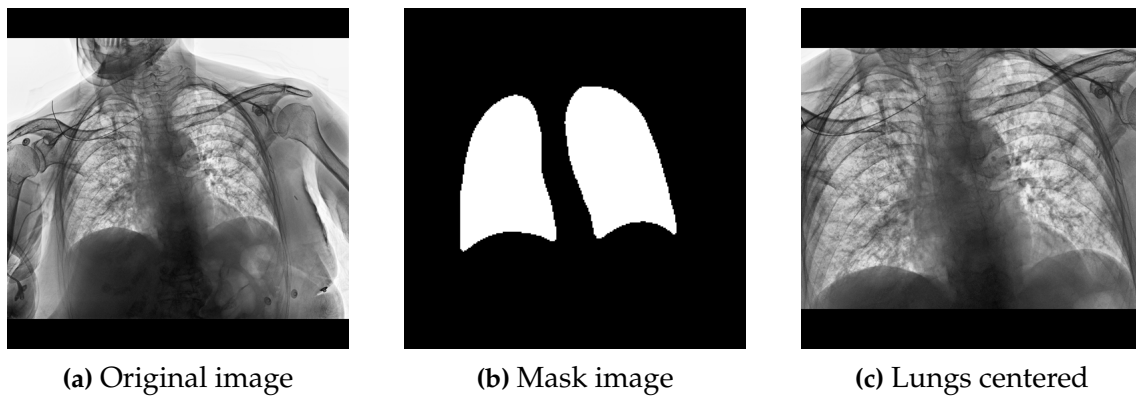


Figure 7.1: Lungs alignment: To align the lung images (cropping and centring), we first trained a U-net model to segment the lung regions. Then, we combined the bounding boxes of the segmented regions to extract the region of interest.

7.2 Detecting Pulmonary conditions

As it was explained in the previous chapter, we defined two classification tasks for this use case. The first task is a binary classification between healthy samples (“normal” class) and samples with Covid-19. The second task is a multi-label classification with four labels: “normal”, “covid19”, “pneumonia” and “infiltrates”. The preprocessing steps, data augmentation, and used models are the same for both tasks. The difference between them is in the loss functions used and the output layers of the models.

The loss functions used on the binary classification task were Binary Cross-Entropy (BCE) and regular Cross-Entropy (CE). We used BCE when the label was represented as 0 or 1, and we only used one output neuron in the last layer of the model. The CE was applied when the labels were encoded as one-hot vectors of length two. Regarding the multi-label task, we used one-hot vectors of length four to encode the labels, and the loss function used was BCE applied to each output unit independently.

Regarding the training process, it was done using pre-trained models as follows: first, the weights of the base model (pre-trained feature extractor) were

frozen so that the new extension on top of it could be trained for 50 epochs. This was the necessary number of iterations to determine where the loss function became flat. Then, we fine-tuned the entire model with a lower learning rate ($1e-5$) for more epochs until reaching a total of 200 epochs in some cases. On each epoch, the models and weights were stored when a lower loss or a higher accuracy was reached in the validation set. At the end of the training phase, we inferred the test set with the best model in terms of validation loss and the best one in terms of validation accuracy. To avoid overfitting during training, we applied weight decay (L2) with a penalty of $1e-5$, besides the data augmentation and the dropout layers. The initial learning rates tested were in the range from $1e-3$ to $1e-5$.

7.2.1. Covid-19 vs Healthy results

On this binary task, we first explored different training configurations and architectures. The best results are summarized in Table 7.1. As we can observe, the best topology has been a pre-trained Resnet18, with an **83.57%** of balanced accuracy when using an input image size of 512x512 pixels. Overall, good precision and recall scores are observed, with an F1-Score of **0.9231** on the “covid19” class in the case of the model trained with 512x512 images.

Table 7.1: Best results on the binary classification task.

Model	Input Size	Accuracy	Balanced Acc.	Precision		Recall		F1-Score	
				Covid	Normal	Covid	Normal	Covid	Normal
ResNet18	256x256	87.62%	81.85%	0.8889	0.8333	0.9474	0.6897	0.9172	0.7547
ResNet18	512x512	88.57%	83.57%	0.9	0.84	0.9474	0.7241	0.9231	0.7778

Regarding the size of the input images, the best results were obtained using 512x512 images. Nevertheless, the improvement of using this input size was not significant with respect to the smaller size. In table 7.2, we show the average results of the pre-trained ResNet18 model after five different training experiments with the same configuration. As we can see, we could replicate quite similar results using a smaller input size.

Table 7.2: Average results of five experiments obtained with the ResNet-18 model using different input sizes. Mean and standard deviation are shown.

Input Size	Accuracy %	Balanced Accuracy %
256x256	84.76 ± 1.4	76.89 ± 2.9
512x512	86.29 ± 1.1	79.94 ± 2.3

7.2.2. Multi-label task results

Regarding the second task, we experimented with different training configurations and topologies. After the experiments, the best results were obtained with ResNet101, trained with Adam as optimizer and a learning rate of $1e-5$ and Binary Cross-Entropy as the loss function applied on each output unit independently. The results are reported in Table 7.3. As it can be observed, the model

obtains a mean Accuracy of **0.7067** for the four classes and a mean Precision of **0.6253**. However, the obtained results for Recall indicate that this model is not working as well as in the binary classification task.

Table 7.3: Results with ResNet-101 on the multi-label task. The input image size used was 512x512.

Metric	Normal	Covid	Pneumonia	Infiltrates	Mean
Accuracy	0.8702	0.7067	0.5769	0.6731	0.7067
Precision	0.6667	0.6056	0.4512	0.7778	0,6253
Recall	0.1379	0.5658	0.4625	0.0959	0,3155
F1-Score	0.2286	0.585	0.4568	0.1707	0.3603

7.3 Discussion

The impact of the Covid-19 pandemic has been a hard experience for everyone. Our efforts were focused on helping in the pandemic from our perspective. Therefore, we approached this task as many researchers have done in the last two years. After the process of developing our solution and carrying out the experiments, we have found many important aspects to consider when approaching a similar task.

First of all, as this is a medical use case, it is crucial to build solutions robust enough to be applied in real scenarios. This starts from the data acquisition process. Medical personnel should be conscientious when collecting data for these purposes because errors in the data or misleading information may lead to problems in the classifiers and confusing results. In this dataset, we observed that samples from the COVID19 subset were not labelled as Covid19 by radiologists, but they had tested positive for Covid19 in a range of 5 days before or after they joined the hospital. These things make the task more difficult as researchers have to make decisions when encountering these issues. Our decision was to use as Covid19 samples only the ones with the “covid” label. These decisions result in a loss of potential helpful data. Nevertheless, we can only trust that the experts correctly labelled the images at this point.

The process of exploring the dataset and preparing the images has been such a complicated task. First, some of the X-ray images had the colors inverted, so we had to fix them manually. Additionally, we found images with no lungs at all, and some images with ‘Posterior-Anterior’ or ‘Anterior-Posterior’ positions labels but with lateral views. In the end, from the total of images provided on the dataset, we could only use 2,083, ending with a few more 100 samples in the test set.

Another factor that increases the difficulty of the task is the variability of the images, in particular the images labelled as Covid19. As the virus can affect the lungs differently depending on many factors, the X-ray scans of Covid19 patients can be very different. Furthermore, Covid19 X-rays usually have artifacts, that can introduce a bias in the data, so the classifier could tend to detect artifacts instead of the virus. An interesting concept to keep in mind in this task would be

model explainability, in order to see where the networks are looking at to make their decisions, but this concept is outside the scope of this work.

Despite the difficulties that have come out during the efforts on this task, we could finally obtain acceptable scores in distinguishing between healthy and Covid19 lungs. We achieved an **83.57%** of balanced accuracy, demonstrating that our solution could successfully separate Covid-19 samples from healthy ones. Nevertheless, our results are far from being used in real scenarios, as we would need a larger dataset for training and evaluating the models.

It is difficult to establish quantitative comparisons with state-of-the-art works as there are not any results published on the BIMCV-COVID19 dataset yet. Regarding the multi-label task, our results were not as good as in the first task. We think this is due to the increase in the task difficulty, as the models need to detect several pathologies at once and some of them are related. Additionally, the samples may not be enough representative to perform such a task because the two additional classes had fewer samples than "covid" and "normal" ones.

Finally, we also presented a method to extract the region of interest from X-ray images in this work. As we explained, we trained a segmentation model with a human-in-the-loop strategy, resulting in success. This process helped exclude the external areas that could introduce more biases to the dataset.

CHAPTER 8

Conclusions and future works

This chapter summarises the most relevant aspects of each use case and gives an overview of the work done. Additionally, some ideas for future works with these tasks are given.

8.1 Conclusions

This work presents two ways of working with the DeepHealth toolkit for real-world use cases related to health. Different deep learning approaches have been tested to solve two different use cases, one related to epilepsy detection with EEGs and one related to detecting pulmonary conditions with lung X-ray images.

In Use Case 13, a solution based on raw data as input and a neural network classifier has been presented. The classifiers followed two approaches, a recurrent and a convolutional approach. Despite the fact that recurrent neural networks can learn dependencies with data from the past, the convolutional approach achieved better results. As shown, the results are not consistent for every patient. In fact, the model works well for some patients but is not enough good for others. The task addressed is challenging mainly for two reasons: (i) A strong imbalance of samples of each class in the dataset. The target class (ictal) has much fewer samples than the other class (interictal). (ii) The high variability between patients.

Additionally, a post-inference process with a function acting as a seizure detection has been presented. The experimentation results show that it is possible to reach a trade-off between a fast detection and a low number of false alarms by adjusting the parameters of the function.

To sum up, the obtained results show that it is possible to achieve a good enough performance to detect seizures in the case of some patients, and suggest that, with more research and more data, the epilepsy detection task could be addressed with more reliable models; what could improve the quality of life of many people affected by Epilepsy.

Regarding Use Case 15 tasks, we first performed a segmentation process in order to obtain the region of interest from the X-rays and reduce the size of the

images without rescaling them too much, so the maximum amount of information was kept. This process had successful results, as has been described before.

Next, we trained several classifiers with different neural network architectures, obtaining the best results with pre-trained Residual Networks such as ResNet-18. We analysed different configurations for two tasks that required a large set of experiments. We observed that Covid-19 samples can be separated from healthy ones, as the models could achieve more than 80% of accuracy in our experiments. In addition, we experimented with a multi-label task and obtained worse results, also working with the same model architectures. Although the obtained results suggest that our solution could not be considered to be used in real scenarios due to the size of the evaluation set, we hope this work can help other researchers better understand this use case, and help in the global objective of finding a cheap and fast solution to detect Covid-19 or other pulmonary pathologies.

This work has shown how to deal with two different use cases related to health with deep learning based solutions. Even though the solutions presented are not robust enough to achieve the final objective of using them in real-world cases, we are proud of our work as this could help other scientists achieve better solutions. We believe that Deep Learning in particular, and Artificial Intelligence in general, are one of the most promising fields of research in the current era, and that these technologies will provide with solutions in next years for many fields such as the health sector.

We would like to highlight the ease of use of the DeepHealth Framework for developing advanced Deep Learning models for medical use cases. Our work suggests that the DeepHealth Framework is a robust framework for performing reliable medical research. However, given the limited amount of medical data available and the problems commented on each use case, more research is needed before these kinds of solutions can be safely deployed in production environments.

Finally, this work has provided us with more experience in the machine learning field, particularly in signal processing and medical imaging. Personally, it has not been an easy way, but it has been an enriching process, as the fact of developing a solution for medical use cases that could potentially help the health of people motivated us so much.

8.2 Future Work

Regarding Use Case 13, a patient-specific scheme was used in this work, so that each model was trained for each patient specifically, only with data from the patient. This strategy led to many experiments, and many more are still to be done, either following a patient-specific scheme or a patient-independent scheme. Nonetheless, our interests are in the following line of research:

- Test more models and configurations with all the subjects.
- Find a solvent feature extraction technique to improve results.

- Test our solution with other EEG datasets.
- Explore the seizure prediction task.

In the case of Use Case 15, our interests are the following ones:

- Test a solution based on a sliding window through a large image to improve the results.
- Enlarge the training dataset with samples from other datasets, train an autoencoder and then use the encoder part for classifying the samples of this dataset.
- Perform a k-fold cross-validation strategy to verify the robustness of our results.
- Test our solution with similar datasets, in order to compare it properly with state-of-the-art solutions.

Bibliography

- [1] Deep health – deep-learning and hpc to boost biomedical applications for health. <https://deephealth-project.eu/>.
- [2] Epilepsy. <https://www.who.int/en/news-room/fact-sheets/detail/epilepsy>.
- [3] Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.0.2 documentation. https://scikit-learn.org/stable/modules/model_evaluation.html#balanced-accuracy-score.
- [4] Onnx | open neural network exchange.
- [5] Prhlt research center – pattern recognition and human language technology. <https://www.prhlt.upv.es/wp/es/>.
- [6] Who director-general’s opening remarks at the media briefing on covid-19 - 11 march 2020. <https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020>.
- [7] Bayoumi Magdy Abdelhameed Ahmed. A deep learning approach for automatic seizure detection in children with epilepsy. *Frontiers in Computational Neuroscience*, 15, 2021.
- [8] Abhijit Bhattacharyya, Divyanshu Bhaik, Sunil Kumar, Prayas Thakur, Rahul Sharma, and Ram Bilas Pachori. A deep learning based approach for automatic detection of covid-19 cases using chest x-ray images. *Biomedical Signal Processing and Control*, 71:103182, 2022.
- [9] Poomipat Boonyakitanont, Apiwat Lek-uthai, and Jitkomut Songsiri. Automatic epileptic seizure onset-offset detection based on cnn in scalp eeg. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1225–1229, 2020.
- [10] Salvador Carrión, Álvaro López-Chilet, Javier Martínez-Bernia, Joan Coll-Alonso, Daniel Chorro-Juan, and Jon Ander Gómez. Automatic detection of epileptic seizures with recurrent and convolutional neural networks. In *Proceedings of the International Conference on Image Analysis and Processing 2022*, 6 2022.

-
- [11] Salvador Carrión, Álvaro López-Chilet, Javier Martínez-Bernia, Joan Coll-Alonso, Daniel Chorro-Juan, and Jon Ander Gómez. Detection of pulmonary conditions using the deephealth framework. In *Proceedings of the International Conference on Image Analysis and Processing 2022*, 6 2022.
- [12] Beatriz Garcia Santa Cruz, Jan Sölter, Matias Nicolas Bossa, and Andreas Dominik Husch. On the composition and limitations of publicly available covid-19 x-ray imaging datasets. *arXiv preprint arXiv:2008.11572*, 2020.
- [13] Hisham Daoud and Magdy A. Bayoumi. Efficient epileptic seizure prediction based on deep learning. *IEEE Transactions on Biomedical Circuits and Systems*, 13(5):804–813, 2019.
- [14] Maria de la Iglesia Vayá, Jose Manuel Saborit-Torres, Joaquim Angel Montell Serrano, Elena Oliver-Garcia, Antonio Pertusa, Aurelia Bustos, Miguel Cazorla, Joaquin Galant, Xavier Barber, Domingo Orozco-Beltrán, Francisco García-García, Marisa Caparrós, Germán González, and Jose María Salinas. Bimcv covid-19+: a large annotated dataset of rx and ct images from covid-19 patients, 2021.
- [15] Joaquim de Moura, Jorge Novo, and Marcos Ortega. Fully automatic deep convolutional approaches for the analysis of covid-19 using chest x-ray images. *Applied Soft Computing*, 115:108190, 2022.
- [16] Theekshana Dissanayake, Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep learning for patient-independent epileptic seizure prediction using scalp eeg signals. *IEEE Sensors Journal*, 21(7):9377–9388, 2021.
- [17] A. J. Gabor, R. R. Leach, and F. U. Dowla. Automated seizure detection using a self-organizing neural network. *Electroencephalography and Clinical Neurophysiology*, 99:257–266, 9 1996.
- [18] A L Goldberger, L A N Amaral, L Glass, J M Hausdorff, Ivanov PCh, R G Mark, J E Mietus, G B Moody, C.-K. Peng, and Stanley H E PhysioBank. Physiokit, and physionet: Components of a new research resource for complex physiologic signals.
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.
- [21] Bo Hjorth. Eeg analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, 29 3:306–10, 1970.
- [22] Saddam Hussain Khan, Anabia Sohail, Asifullah Khan, and Yeon-Soo Lee. Covid-19 detection in chest x-ray images using a new channel boosted cnn. *Diagnostics*, 12(2), 2022.

- [23] Aayush Kumar, Ayush R Tripathi, Suresh Chandra Satapathy, and Yu-Dong Zhang. Sars-net: Covid-19 detection from chest x-rays by combining graph convolutional network and convolutional neural network. *Pattern Recognition*, 122:108255, 2022.
- [24] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.
- [25] Chien-Liang Liu, Bin Xiao, Wen-Hoar Hsaio, and Vincent S. Tseng. Epileptic seizure prediction with multi-view convolutional neural networks. *IEEE Access*, 7:170352–170361, 2019.
- [26] Soumya Ranjan Nayak, Deepak Ranjan Nayak, Utkarsh Sinha, Vaibhav Arora, and Ram Bilas Pachori. Application of deep learning techniques for detection of covid-19 cases using chest x-ray images: A comprehensive study. *Biomedical Signal Processing and Control*, 64:102365, 2021.
- [27] Ahmet Remzi Ozcan and Sarp Erturk. Seizure prediction in scalp eeg using 3d convolutional neural networks with an image-based approach. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(11):2284–2293, 2019.
- [28] Xiao Qi, David J. Foran, John L. Noshier, and Ilker Hacihaliloglu. Multi-Feature Semi-Supervised Learning for COVID-19 Diagnosis from Chest X-Ray Images. *Lecture Notes in Computer Science*, page 151–160, 2021.
- [29] Gonzalo Rojas, Carolina Alvarez, Carlos Montoya Moya, Maria de la Iglesia Vaya, Jaime Cisternas, and Marcelo Gálvez. Study of resting-state functional connectivity networks using eeg electrodes position as seed. *Frontiers in Neuroscience*, 12, 03 2018.
- [30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- [31] M. E. Saab and J. Gotman. A system to detect the onset of epileptic seizures in scalp eeg. *Clinical Neurophysiology*, 116:427–442, 2 2005.
- [32] Andreas Schulze-Bonhage, Francisco Sales, Kathrin Wagner, Rute Teotonio, Astrid Carius, Annette Schelle, and Matthias Ihle. Views of patients with epilepsy on seizure prediction devices. *Epilepsy and Behavior*, 18:388–396, 8 2010.
- [33] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOSmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. *opencv/cvat: v1.1.0*, August 2020.
- [34] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say

- that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.
- [35] Ali Shoeb and John Guttag. Application of machine learning to epileptic seizure detection. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10*, page 975–982, Madison, WI, USA, 2010. Omnipress.
- [36] Nhan Duy Truong, Anh Duy Nguyen, Levin Kuhlmann, Mohammad Reza Bonyadi, Jiawei Yang, Samuel Ippolito, and Omid Kavehei. Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram. *Neural Networks*, 105:104–111, 9 2018.
- [37] Shuhan Yang, Bo Li, Yinda Zhang, Meiyu Duan, Shuai Liu, Yexian Zhang, Xin Feng, Renbo Tan, Lan Huang, and Fengfeng Zhou. Selection of features for patient-independent detection of seizure events using scalp eeg signals. *Computers in Biology and Medicine*, 119:103671, 4 2020.

APPENDIX A

Beyond Use Case 13

In this appendix, we present the different solutions tested on Use Case 13. These different approaches were tested, but the results were not significant in order to be presented in this report. As the work done could be interesting, a description of the additional approaches is presented in the following sections.

A.1 Autoencoders

One solution that came out when trying to solve this task was the use of autoencoders. The idea was to train a neural network model to get a signal as input, extract the most relevant features while reducing dimensionality, and then reconstruct the input signal. Different neural network models were tested, from simple models with only Dense layers to models with convolutional neural networks similar to image segmentation models such as SegNet or UNet.

These autoencoders were trained with interictal data from at least one hour away from any ictal period. This was done in order to use interictal data the more similar to a normal state of the patient. The model was trained to try to minimize the mean squared error loss function. Once the model was trained with interictal data, it was tested with the original test signal, and for each sample, the difference between the output and the original input was measured with the Euclidean distance. We expected that the difference would be minimal when the input was an interictal sample, as the sample was not too much different from the ones that the model was trained on. When the input was a period close to or inside a seizure, we expected the distance to be higher, as the signal on these periods is different from the training set. With this in mind, we illustrated the distance for each sample on the test set, and we tried to find the behaviour described, but the model did not perform as expected. If the model performed as expected, we could study the use of a threshold to distinguish different periods in the signal, not only the ictal from the others.

A.2 Feature Extraction

This section presents a feature extraction method for EEG signals.

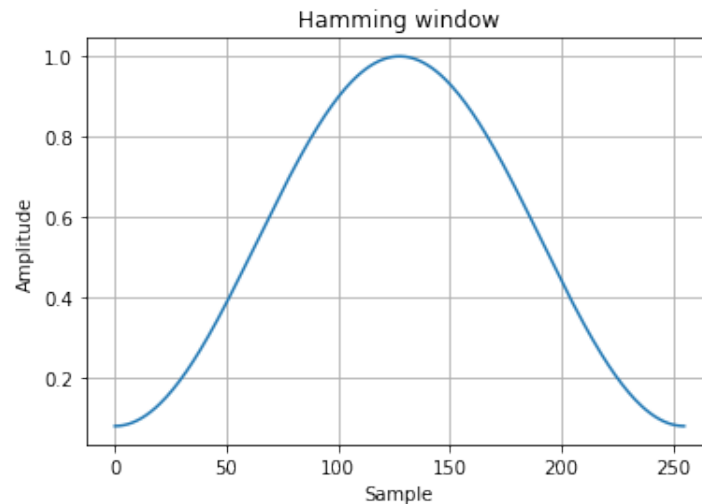


Figure A.1: Hamming window shape.

A.2.1. Feature extraction process

Before feeding the deep learning models, we are going to process the signals in order to extract the most important features. This preprocess has been done following the typical short-term analysis applied for Speech Recognition tasks and other signals. By applying this kind of analysis, we are assuming that we are processing a signal that contains information that does not change too fast. This means that, for a short period of time, the features we are interested to extract do not change. In Speech Recognition, the short-term considered period is 20-25ms. In the case of the EEG signals, and due to a large amount of data in this dataset, the period will be 4 seconds. In these cases, a sliding window is applied. For Speech Recognition the window slides ten milliseconds each time, but for this task, we are going to use 2 seconds, exactly half of the period we are analysing. In summary, we will extract features of the signal using a sliding window of 4 seconds, that we will shift through the signal every 2 seconds.

For each window extracted from the signal, we first compute the Discrete Fourier Transform (DFT) to obtain a distribution of the energy contained in the signal in the frequency domain. Additionally, we apply a filter bank in order to filter the frequency range of values into channels. After this, we would apply the Discrete Cosine Transform to obtain the Cepstral Coefficients and the energy in the case of Speech Recognition, but this last part is not done for EEG signals.

Hamming window

The Discrete Fourier Transform is sensitive to sharp changes in the signal, that are usually represented with a high amount of harmonics. In order to avoid these changes affecting the DFT, it is required to use a smoothing window. The most widely used window for smoothing is the Hamming Window, which shape can be seen in Figure A.1.

Filter bank

After applying the DFT, we use a filter bank to filter the frequency values into channels. In Speech Recognition, the Mel Scale filter bank is the most used. This filter bank consists of a set of overlapping passband filters of equal size with more resolution on low frequencies than high frequencies, trying to emulate the frequency discrimination of human's Cochlea. For the EEG signals, we are going to apply a linear filter bank instead of a logarithmic one. The filter will be restricted to a specific band of frequencies.

We will use the same filter bank as authors of [27]. The frequency domain EEG signal is separated into 5 spectral bands: δ , θ , α , β and γ . Considering that high-frequency sub-bands are effective in epileptic seizure tasks, the γ band is split into four sub-bands, resulting in a total of 8 spectral bands. The range of frequencies for each band is the following one:

- δ : 0.5 Hz to 4.0 Hz
- θ : 4.0 Hz to 8.0 Hz
- α : 8.0 Hz to 13.0 Hz
- β : 13.0 Hz to 30.0 Hz
- γ 1 : 30.0 Hz to 50.0 Hz
- γ 2 : 50.0 Hz to 75.0 Hz
- γ 3: 75.0 Hz to 100.0 Hz
- γ 4: 100.0 Hz to 128.0 Hz

After applying this filter bank, we get 8 values for each channel of the extracted window, corresponding to the 8 spectral bands.

Time-domain Statistics

Additionally to the feature extraction in the frequency domain, we are also extracting some features in the time domain in order to complement the feature vector for each window. These features will be some statistics, also inspired by the same paper as for the filter bank [27].

Statistical moments provide information about the amplitude distribution and shape of time series. This information could be helpful for detecting seizures, as the amplitude of some of the channels changes drastically depending on the stage of the seizure. The first four statistical features are mean, variance, skewness and kurtosis. The last three statistical moments are Hjorth parameters [21]. These are Activity, which represents the signal power, the variance of a time function;

Mobility, which represents the mean frequency of the proportion of standard deviation of the power spectrum; and Complexity, which represents the change in frequency. They are calculated with the following equations:

$$\text{Activity} = \text{var}(y(t)) \quad (\text{A.1})$$

$$\text{Mobility} = \sqrt{\frac{\text{var}\left(\frac{dy(t)}{dt}\right)}{\text{var}(y(t))}} \quad (\text{A.2})$$

$$\text{Complexity} = \frac{\text{Mobility}\left(\frac{dy(t)}{dt}\right)}{\text{Mobility}(\text{var}(y(t)))} \quad (\text{A.3})$$

As the Activity parameter is the variance of the function, we will have six statistical features in total. These will complement the feature vector, which will have 14 values: 8 in the frequency domain and 6 in the time domain.

After processing a signal to extract all of the features explained, an example of the resulting signal can be seen in Figure A.2. In it, we can see an extract of 5 seconds of the first record of patient *chb01*. It is the result of the process for the first channel of the signal.

A.3 Review on the prediction task

The other task associated with epilepsy is seizure prediction, which is more challenging and has been approached more recently. Many of the solutions from last years have been based on neural networks, and researchers have been exploiting different deep learning models, which are nowadays very powerful for many tasks, like processing speech signals for speech recognition. In [25], authors propose a multi-view convolutional neural network framework to predict seizures. They had two separated inputs: temporal and spectral features. The network had to acquire a shared representation of the signal. They achieved an Area Under the Curve of 0.82 and 0.89 on two patients of the CHB-MIT dataset. In [36], Truong et al. used the short-time Fourier transform on 30-s EEG windows to extract information. After that, they use convolutional neural networks to classify the samples. They achieved an average sensitivity of 81.2% in the CHB-MIT dataset and similar results in other datasets. In [27], they present a feature extraction that converts the signal into images by projecting the signal channels' position in the head to a 2D space and then computing the points between them by interpolation. They trained a CNN to classify the images and achieved a sensitivity of 85.7% in the CHB-MIT dataset. In [16], the authors propose two Deep Learning architectures that can learn a global function to predict seizures for any patient. They present two different strategies for training: a multiple loss function training and a siamese architecture with a Contrastive loss. Both of the models achieve state-of-the-art performance for the prediction in the CHB-MIT Dataset, with an accuracy of 88.81% and 91.54% for the multiple loss strategy and the siamese

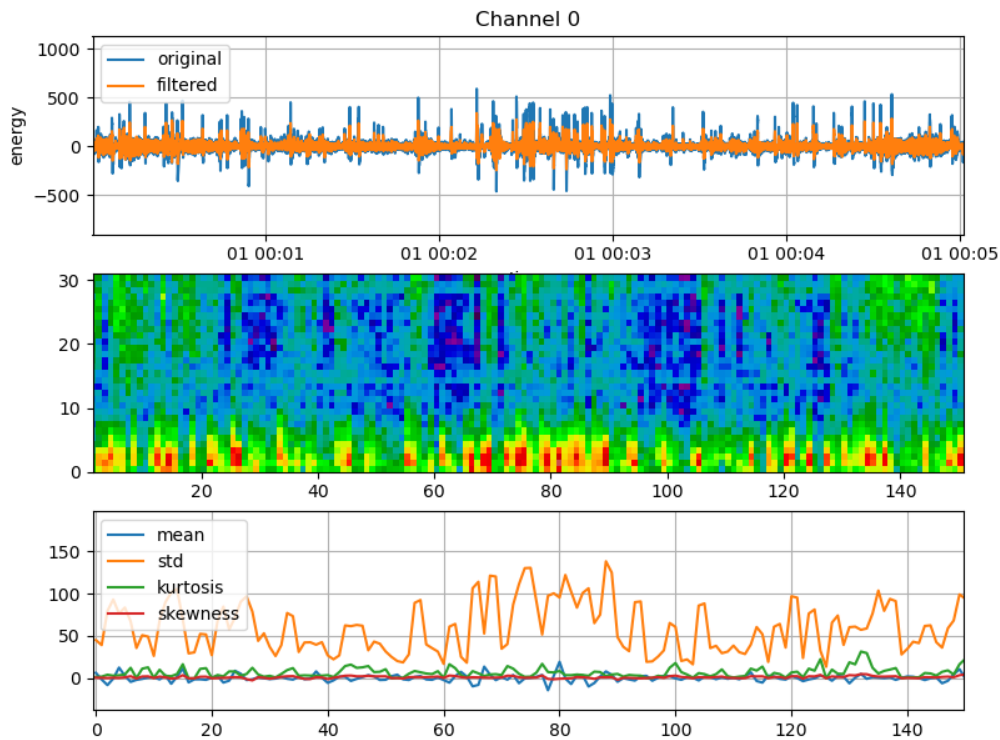


Figure A.2: Processed signal. The top figure shows the frequency of the original signal and the filtered signal. The figure in the middle shows the energy that falls on different filter banks. The bottom figure shows the time-domain statistics of the signal.

network, respectively. Finally, in [13], Daoud et al. propose four deep learning models (including convolutional neural networks, recurrent neural networks and autoencoders) and a channel selection algorithm for selecting the channels of the signal to use. They achieved an accuracy of 99.96% and a false alarm rate per hour of 0.004 on the CHB-MIT dataset.