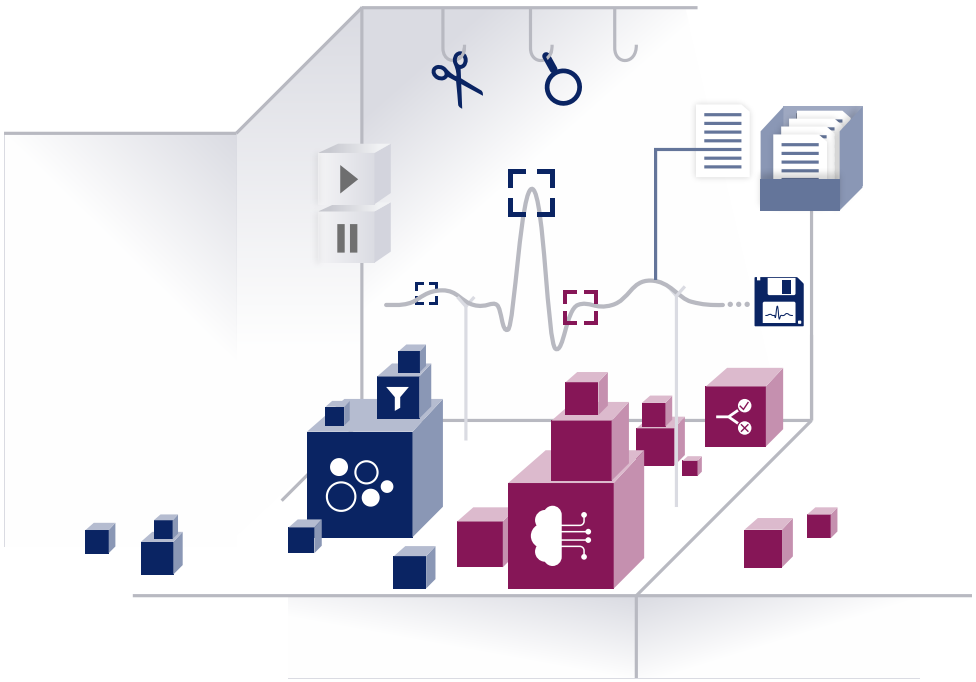
TÉCNICO LISBOA



Deep Residual Learning for Epileptic Seizure Prediction and Tools to Expedite Biosignal Research

João Miguel Areias Saraiva

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors

Prof. Hugo Humberto Plácido da Silva
Prof. Ana Luísa Nobre Fred

Examination Committee

Chairperson: Prof. José Alberto Rodrigues Pereira Sardinha
Supervisor: Prof. Hugo Humberto Plácido da Silva
Member of the Committee: Prof. Susana de Almeida Mendes Vinga Martins

October 2022

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Preface

The work presented in this Thesis was performed at Instituto de Telecomunicações (IT) (Lisbon, Portugal) and at the Departamento de Engenharia Informática at Instituto Superior Técnico (IST) (Lisbon, Portugal), during the period March – October 2022, in collaboration with the epilepsy monitoring unit of Hospital de Santa Maria (HSM) and Hospital de Egas Moniz (HEM) (Lisbon, Portugal), under the supervision of Prof. Dr. Ana Luísa Nobre Fred, PhD, and Prof. Dr. Hugo Plácido da Silva, PhD, from IT and IST, supervision support from Eng. Mariana Abreu and Eng. Ana Sofia Carmo, from IT and IST, and clinical advice of Dr. Carla Bentes, M.D., of HSM. This work was part of a collaborative project between IT, HSM and HEM named ”PreEpiSeizures” concerned with the development of devices to monitor patients with Epilepsy. This work was partially funded by the IST research grant BL88/2022, under the scope of project 1018P.06071.1.01.01 ”CardioLeather”, by the IT research grants BI16/2021 and 2022.12369.BD, and by the Fundação para a Ciência e Tecnologia (FCT) / Ministério da Ciência, Tecnologia e Ensino Superior (MCTES), through national funds and when applicable co-funded by EU funds under grants 2021.08297.BD and under the projects PCIF/SSO/0163/2019 ”SafeFire” and DSAIPA/AI/0122/2020 ”AIMHealth”.

Acknowledgments

Dedico esta Dissertação aos meus pais, Marta e João, que, sempre e incondicionalmente, me deixaram sonhar. Fico com uma dívida impagável para com eles por todo o sacrifício que fizeram para que pudesse estudar simultaneamente em dois mestrados e ter a oportunidade de me formar em dois ramos da engenharia em tão curto espaço de tempo. Pelo amor e apoio familiar que nunca faltou. Pela educação que me deram e por todos os valores que me incutiram e que se refletem neste trabalho até ao mais ínfimo detalhe. Sou, verdadeira e modestamente, alguém privilegiado graças a eles.

Este trabalho só foi possível graças aos meus orientadores Ana Fred, Hugo Silva, Mariana Abreu, Ana Sofia Carmo e Carla Bentes. Obrigado por terem acreditado nas minhas capacidades desde o início. Trabalhar convosco faz-se com um sorriso na cara todos os dias. Obrigado pelos conselhos, pelas gargalhadas e por sempre experimentarmos até a pior das ideias. Agradeço ainda à Dr. Carla Bentes pelos esclarecimetos clínicos e pela oportunidade de colaboração. A próxima interação multidisciplinar com os restantes colegas de investigação foi também ela indispensável para a conclusão deste trabalho. Agradeço especialmente ao André Gomes, Maria Brites, Joana Brito e Rafael Silva por todo o conhecimento que trocámos e apoio técnico que partilhámos.

Agradeço o apoio e a amizade dos meus companheiros dos últimos 6 anos, em especial à Carolina Cebola, Sara Franco, Miguel Mendes, Margarida Ferro, Pedro Santos, Mariana Tilley, Francisco Catarrinho e Pedro Silva. Agradeço também o apoio incansável dos meus avós, tios, primas e restante família. A todos vós, pela paciência que tiveram comigo e por compreenderem todas as vezes que não pude estar convosco por deveres académicos.

Agradeço ainda a todos os que participaram nos testes com utilizadores do LTBio. O vosso contributo foi de extrema importância para este trabalho. Todas as vossas opiniões foram ouvidas, estão e continuarão a ser muito úteis no desenvolvimento desta ferramenta que é feita para vocês.

Por último, agradeceço aos que vieram antes de mim. Por poder viver num país livre e com acesso democratizado à Educação. Àquelas e àqueles que acreditaram que hoje alguém como eu podia estudar. Àquelas e àqueles que acreditaram que a Educação não deve ter cor, género ou carteira.

João Saraiva

Resumo

Mundialmente, cerca de 50 milhões de pessoas sofrem de epilepsia, uma doença caracterizada por crises recorrentes e espontâneas de atividade cerebral abrupta. As crises epilépticas podem constituir um perigo considerável no quotidiano dos doentes e daqueles que os rodeiam. Constituiria um aumento significativo na qualidade de vida destes doentes se pudessem usar um dispositivo que previsse e os alertasse na iminência de novas crises. Neste trabalho, a desregulação do sistema autonómico nervoso (SAN) precedente a algumas crises epilépticas – e que pode estar patente no electrocardiograma (ECG) de alguns doentes – é explorada para prever estas crises. São propostos modelos de aprendizagem profunda por resíduos para discriminar entre segmentos de ECG prévios às crises – o período pré-ictal – dos restantes. Foram treinados modelos do tipo ResNet-34, personalizados para cada doente, que testaram para um F1-score de 0.728 mediano para 11 doentes. Dada a classificação dos segmentos, um algoritmo de decisão desencadeou alarmes com um F1-score de 0.774 e 4 falsos alarmes por dia. Para os doentes que responderam a este método, em média, foram previstas 78.4% das suas crises com uma antecipação mediana de 9.1 minutos. Este trabalho apresenta ainda a Long-Term Biosignals Framework (LTBio) para facilitar a investigação em biossinais e reduzir o tempo dispensado em programação repetitiva. O LTBio atingiu uma pontuação de 85.75 (em 100) na escala de usabilidade SUS e provou-se uma mais-valia no grupo de investigação em epilepsia, contribuindo para acelerar o estudo de modelos de aprendizagem automática para biossinais.

Palavras-chave: previsão de crises epilépticas, aprendizagem profunda por resíduos, electrocardiograma (ECG), investigação em biosinais, investigação biomédica, computação fisiológica, ferramentas para programação científica

Abstract

Around the world, about 50 million people suffer from epilepsy, a disease characterised by recurrent and unprovoked seizures of abrupt cerebral activity. Epileptic seizures can present a considerable danger to the daily life of patients and those around them. It would constitute a significant improvement in their quality of life if patients could wear some device that would predict their epileptic seizures and raise raise an alarm when a seizure is about to be elicited. In this work, the autonomic nervous system (ANS) deregulation before epileptic seizures – that can be reflected in the patients' electrocardiogram (ECG) – is exploited to predict epileptic seizures. Deep residual learning is proposed to discriminate between ECG segments of seizure eminence – the precital period – from the remaining. ResNet-34 models were train in a patient-specific manner for 11 patients and a cohort median F1-score of 0.728 was attained. A decision algorithm to raise alarms from the segment classifications attained a 0.774 F1-score with a median of 4 false alarms per day. On average, patients that responded to this method had 78.4% of their seizures predicted with a median anticipation of 9.1 minutes. Furthermore, a novel framework for the management and processing of long-term biosignals is proposed to help biosignal engineers to focus more on the research at hands. The LTBio framework achieved a score of 85.75/100 in the system usability scale (SUS) and has proved to be effective in expediting machine learning research with biosignals in the epileptic seizure prediction group.

Keywords: epileptic seizure prediction, deep residual learning, electrocardiogram (ECG), biosignal research, biomedical research, physiological computing, scientific programming tools

Contents

List of Tables

List of Figures

Abbreviations

1D 1-dimensional. 18

ACC accelerometer. 6, 10, 12, 22, 49, 64, 65

AI artificial intelligence. 9

ANN artificial neural network. xiii, 16–18

ANS autonomic nervous system. 14, 15, 76

ASM anti-seizure medication. 2, 10, 42, 54

AUC area under the curve. 20

CNN convolutional neural network. 8, 25

CNS central nervous system. 2, 12

CSI cardiac sympathetic index. 8, 22, 25, 43

CSMTS clusters of short myoclonic-tonic seizures. 14, 23

CV cross-validation. 36–39, 41, 42

DL deep learning. 11, 30, 34, 41, 77

ECG electrocardiogram. xvii, 1–8, 10, 11, 19, 21–23, 25, 26, 28, 30, 31, 33, 34, 38, 39, 72, 76

ECG electrocardiography. 6

ECG electrocardiography. 7–9, 12, 22–25, 31, 32, 35, 43, 45, 46, 49, 51, 54, 56, 59, 60, 64, 72, 73

EDA electrodermal activity. 1, 5, 10, 22, 45, 46, 65

EDF european data format. 51

EEG electroencephalography. 8

EEG electroencephalogram. xiii, 1, 3, 5, 6, 8–11, 15, 20–22, 25, 26, 33, 36, 42, 45, 46

EMG electromyogram. 5, 22, 45, 46, 60, 72, 73

EOG electrooculography. 5, 9

FAR false alarm rate. 3, 20, 22–25, 36, 41–43, 76

FAS focal aware seizure. 13, 15, 16, 22, 23, 25, 30

FBTCS focal to bilateral tonic-clonic seizure. 13, 15, 22–25, 30

FIAS focal with impaired awareness seizure. 13, 15, 16, 22–25, 30

FIR finite impulse response. 30

FN false negative. 20, 36, 94

FP false positive. 20, 29, 36, 94

FUAS focal with unknown awareness seizure. 13, 22, 25, 30

GAS generalized absence seizure. 15, 25

GMM gaussian mixture model. 25

GPU graphical processing unit. 37, 67

GTCS generalized tonic-clonic seizure. 14–16, 22, 23, 25

GTS generalized tonic seizure. 23

HEM Hospital de Egas Moniz. 9, 30, 51, 53

HR heart rate. xiii, 6, 8, 22–24

HRV heart rate variability. xiii, 8, 15, 22–25, 30, 43, 59, 66

HSM Hospital de Santa Maria. 3, 9, 30, 42, 51, 53

IDE integrated development environment. 71

ILAE International League Against Epilepsy. 12, 22

IT Instituto de Telecomunicações. 1–3, 9, 10, 51, 54, 57, 62, 64, 74–76

KSC kernel spectral clustering. 23, 25, 43

kSQI kurtosis signal quality index. 31, 32

LS-SVM least squares support vector machine. 25

LTBio Long-Term Biosignals Framework. xiv, 4, 44–51, 53, 62, 64, 65, 67, 68, 71–74, 76, 77

ML machine learning. 3, 19, 22, 23, 25, 33, 48, 68, 73, 76, 77

MLP multilayer perceptron. 24, 25, 43, 73

MPS Metal Performance Shaders. 37

MSE mean squared error. 99

OOP object-oriented paradigm. 49

PH prediction horizon. 33, 34

PPG photoplethysmogram. 1, 5, 8, 10, 12, 22, 24, 45, 46, 49, 65

PPV positive predictive value. 20, 36, 41, 43, 95

pSQI QRS-power signal quality index. 31, 32

QoL quality of life. 1, 2

QRS QRS complex. 31

RDF respiratory-derived feature. 23

ReLU rectified linear unit. 16, 26, 27

ResNet residual network. 26–29, 35, 38, 41, 67, 76, 94, 95

RESP respirogram. 1, 5, 9, 10, 45, 46, 64

RF random forest. 43

RNN recurrent neural network. 25

RRI R-R peak interval. 8, 59

Sen sensitivity. 43

Spe specificity. 43

SpO2 oxygen saturation. 5

SQI signal quality index. 31

SUDEP sudden unexpected death in epilepsy. 2, 15

SUS system usability scale. 73, 74, 77

SVM support vector machine. 23–25, 43

TC tonic-clonic. 23

TEMP temperature. 6, 65

TLE temporal lobe epilepsy. 13

TN true negative. 20, 36, 94

TP true positive. 20, 29, 36, 94

TRC packet trace. 55

UML unified modeling language. 51, 70

vEEG video-electroencephalography. 8–10, 30, 34, 54, 76

WHO World Health Organization. 2

XTLE extratemporal lobe epilepsy. 13

Chapter 1

Introduction

Wearable devices have been engineered for ambulatory unobtrusive continuous monitoring of biosignals. Biosignals are processes that measure the body internal state, such as the brain electrical activity or the cardiovascular processes. These biosignal wearable devices have opened opportunities in healthcare applications to improve the quality of life of quality of life (QoL) of patients. In this work, the topic of epileptic seizure prediction with wearable devices is explored. Figure 1.1 shows three wearable devices that have been proposed to predict epileptic seizures of patients with epilepsy. Left panel A shows Epilog, which predicts epileptic seizures directly from the source: brain electrical abnormalities. Middle panel B shows Empatica E4, which detects epileptic seizures from the wrist blood pulse. And right panel C shows the *PreEpiSeizures* band, developed at the research group in which this thesis was conducted, which aims at predicting seizures from the heart abnormalities induced by seizures.

The workflow of the *PreEpiSeizures* and other projects at Instituto de Telecomunicações (IT) was studied. From acquiring the biosignals, validating them, annotating them, incorporating clinical findings, storing them safely, and sharing them with other researchers, until pre-possessing and analysis are actually performed, months may already have been gone. Sometimes, pre-research work accumulates up to the point teams spend more time and resources on it rather than on the actual biosignal investigation. Hence, the second part of this work proposes software tools to expedite any type of biosignal research.

Figure 1.1: Wearable devices for epileptic seizure monitoring. A) Epilog measures EEG [1]. B) Empatica Embrace measures PPG and EDA [2]. C) PreEpiSeizures band measures ECG and RESP [3].

1.1 Prediction of Epileptic Seizures

Epilepsy is a chronic neurological disorder of the central nervous system (CNS), characterised by the enduring predisposition the brain of some individuals has to produce epileptic seizures and the associated cognitive, psychological and social consequences [4]. The World Health Organization (WHO) estimates that there are approximately 50 million people diagnosed with some form of epilepsy worldwide [5], from which 50 thousand (0.1%) live in Portugal [6]. In developing countries, the risk of premature death is 3 times higher for epilepsy patients [5].

Appropriate medication, termed anti-seizure medication (ASM), can control, reduce, or even cease most epileptic seizures of patients, allowing them to live normal personal and professional lives. However, it is estimated that ASMs only work effectively in approximately 2/3 (two thirds) of all epileptic patients [5, 7]. For the other 1/3 (one third) of patients, epileptic seizures are still recurrent in their daily lives, which is clinically designated as refractory epilepsy, or drug-resistant epilepsy [8]. Only 7–8% of these patients still have surgery as an option to cease seizures [9].

Epileptic seizures may represent serious danger situations in the daily life of patients with refractory epilepsy. Seizures can result in convulsions, falls, unawareness, unresponsiveness, memory impairment, urinary or fecal incontinence, among other events. After the outbreak, complications may also arise, such as physical injury of the patients or others around them, drowning in the sea or swimming pools, or car accidents when the patient is the driver [10]. In the worst case scenario, epileptic seizures can also lead to *status epilepticus*[1] [11], sudden unexpected death in epilepsy (SUDEP) [12], or death by injury. Besides the danger associated with some of these events, epilepsy can impact negatively patient's QoL and require dependence on family or professional caregivers [13]. Social stigma, fear and misconceptions about epilepsy also lead to patients with epilepsy to be discriminated in daily life situations [14] and to develop a low self esteem and mental health disorders [5, 15].

In the light of that, for patients with epilepsy, specially the ones that cannot control seizures with medication, it would be a major improvement in their safety and QoL if they could have some device that could predict seizures before they arise, giving them time to take action and to prepare adequately wherever they are. The *PreEpiSeizures* project developed at IT in collaboration with two hospitals in Lisbon, where this thesis was conducted, has been developing wearables and algorithms for this purpose. To continue that work, this thesis proposes the use of deep residual neural networks to predict epileptic seizures from the electrocardiogram (ECG), and compares that approach with the most recent state-of-the-art methods. This part of the work is presented in Chapter 3.

1.2 Expediting Biosignal Research

Other biosignal research topics follow similar workflows to that of epilepsy research. There is a general set of tasks common to nearly all biosignal research projects that acquire large volumes of

[1] *Status epilepticus* is a medical emergency involving one acute prolonged seizure for more than 5 minutes.

data. The pace of research can become hindered if all data is not organised properly. More than just organised, it should be organised in a way that accelerates the posterior analysis of biosignals. Sometimes, pre-research work accumulates up to the point teams spend more time and resources on it rather than on the actual biosignal investigation.

The problem continues downstream, when researchers want to study biosignals with machine learning (ML) techniques, but end up using software that was simply not designed around the workflow of processing biosignals and all the data associated with them, and that needs to be present when researching. Moreover, exploratory analysis is a big component of biosignal research, which can be slowed down if computer software is not facilitated towards manoeuvring and exploring long-term biosignals.

To mitigate the described situations, a complete set of tools – the LTBio Framework – was developed so that researchers focus more on the research at hands rather than the computer programming behind it. This framework facilitated, and continues to do so, the epilepsy research being conducted in the *PreEpiSeizures* project. Developing it to be general enough is a prime goal for it to be used in other biosignal research projects, and therefore to succeed in the community.

1.3 Contributions

The main scientific contributions of this dissertation are:

- A comprehensive review on the state-of-the-art algorithms to detect and predict epileptic seizures using both EEG and non-EEG (peripheral) biosignals, accessible in Section 3 of:

  Saraiva, João, "Detection and Anticipated Prediction of Epileptic Seizures on Continuous-Monitoring Wearable Devices: A Systematic Review and Research Proposal", 2022 [16].

- An ECG-based deep learning method to predict epileptic seizures with sufficiently satisfactory anticipation and low false alarm rate (FAR), up to the state-of-the-art standards.

- A Python framework, LTBio, to manage and process all aspects of biosignals, seamlessly integrated with industry-standard libraries, like BioSSPy, SciKit Learn, and PyTorch. Its source and documentation are open and publicly available in:

  `github.com/jomy-kk/IT-LongTermBiosignals`

  Stable releases to be used in research projects can be installed from PyPi:

  `pypi.org/project/LongTermBiosignals`

Additionally, a significant effort was also devoted to the following complementary activities inserted in the context of the IT research group:

- Acquisition and gathering of biosignals of patients with Epilepsy hospitalised at Hospital de Santa Maria (HSM), substituting the responsible engineer when absent, thus contributing to enlarge the epileptic seizure dataset already existent and being created at IT.

- Scientific validation and benchmarking of ScientISST hardware and software, contributing to storing acquired biosignals in a more efficient format, easy to read, analyse and share with peer researchers.

1.4 Outline

Chapter 2 introduces the necessary background about biosignal modalities, epileptic seizures and deep learning theory.

The former part of this thesis regards epilepsy research. It follows Chapter 3 that dives into the challenge of predicting epileptic seizures. State-of-the-art methods are compared and design heuristics are gathered regarding patient-specificity, and the performance of models for different types of seizures. After a revision in deep learning theory, an ECG-based deep residual network architecture is proposed to predict epileptic seizures. Its performance is evaluated with the cohort dataset being acquired at the collaborating hospitals and against clinical expectations.

The second part of this thesis regards LTBio. Chapter 4 starts with a review of software tools similar to LTBio, and the gathering of requirements for its design. The framework is introduced in the following Sections, and evaluated with tests and users. Moreover, its documentation and open-source ambition to be expanded by the community is discussed.

Chapter 5 concludes with a summary of the primary results and suggestions for future improvements.

Chapter 2

Background

This Chapter provides the necessary background to understand biosignal research and epilepsy research in particular. By the end of this Chapter, the reader should understand what a biosignal is, what biosignal modalities there are, in particular the electrocardiogram (ECG), and how biosignals are processed and studied. These topics are covered in Section 2.1. The fundamentals of epileptic seizures, explained in Section 2.2, should also be understood. Finally, Sections 2.3 and 2.4 present, respectively, a revision in deep learning theory and metrics to evaluate classification models, in particular epileptic seizure prediction models.

2.1 Biosignal Acquisition and Processing

A biosignal is any time-varying process that describes a biological variable produced by some human tissue or organ. The variable itself is some form of energy, such as electrical energy (e.g. brain and heart signals), mechanical energy (e.g. movement and speech), thermal energy (e.g. body temperature), or chemical energy (e.g. hormone concentration) [17]. These biological variables can be measured through time by analogue instruments, converted to the digital domain, and stored in computer files for posterior analysis. Examples of biosignal modalities and common body locations to acquire them are depicted in Figure 2.1. Here is informally listed what biological variable each modality measures [17, 18, 19]:

- Electrocardiogram (ECG) – the cardiac muscle electrical activity.
- Electroencephalogram (EEG) – the neural electrical activity.
- Respirogram (RESP) – the rib cage movement due to lung activity.
- Electromyogram (EMG) – the electrical activity of any muscle of interest.
- Electrooculography (EOG) – the eye electrical activity.
- Electrodermal activity (EDA) – the skin resistance changes due to sweat release.
- Photoplethysmogram (PPG) – the blood volume pulse.
- Oxygen saturation (SpO2) – the percentage of oxygen present in blood flow.

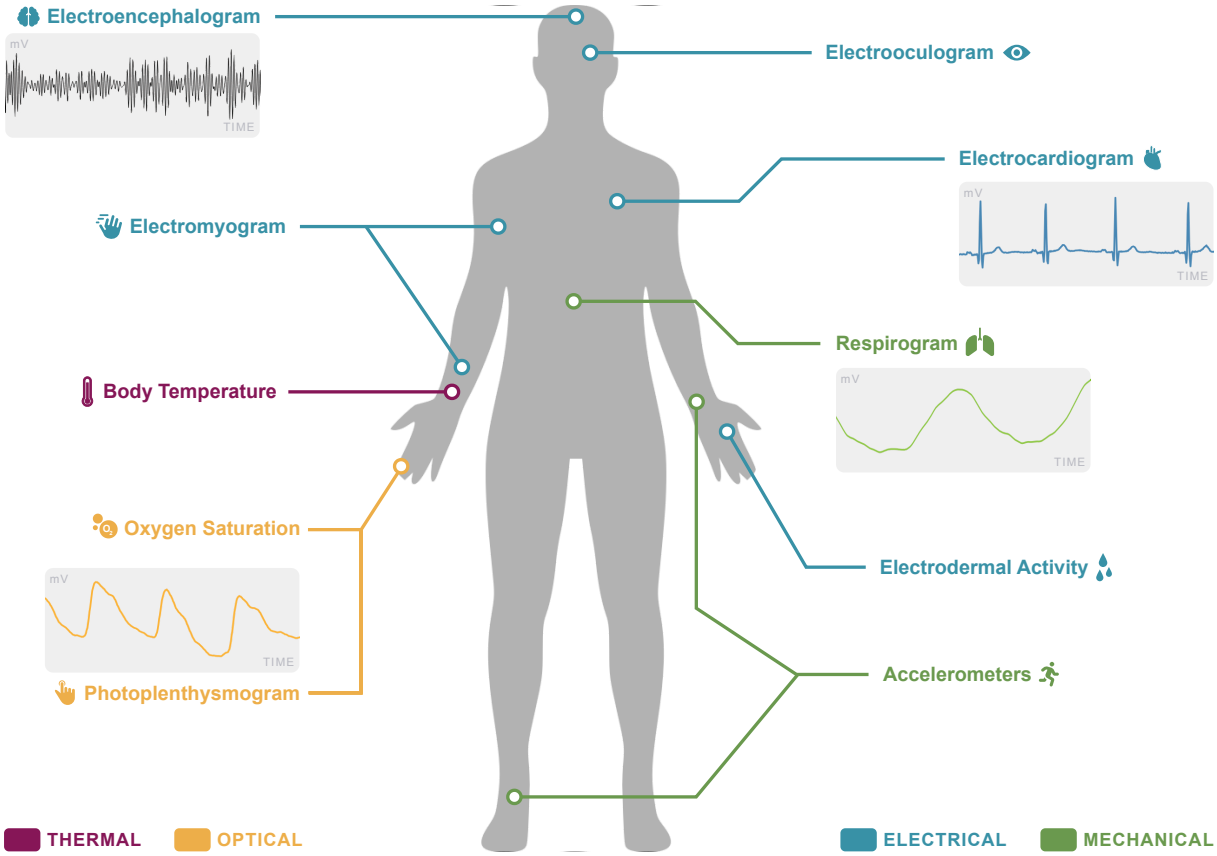


Figure 2.1: Multiple biosignal modalities and their common acquisition locations in the human body.

- Temperature (TEMP) – the temperature of any body region of interest.
- Accelerometer (ACC) – the movement of any body region of interest.

Some features extracted from biosignals can be interpreted by physicians or by specialised algorithms [17]. For instance, in clinical settings, the four common human vitals can be extracted: heart rate (HR), blood pressure, respiratory rate, and body temperature. On the other end, in engineering and research settings, more complex features can be extracted and algorithms can be devised to interpret and draw conclusions from them, a topic that will be approached in later Sections.

In this work, the ECG and EEG modalities will get the most attention in Chapter 3, although multimodality is addressed in Chapter 4. Hence, the following Subsections cover ECG and EEG in detail.

2.1.1 Electrocardiography (ECG)

The electrocardiography (ECG) technique measures the electrical activity of the heart at the body surface. The recorded biosignal corresponds to changes in the polarisation[1] of cardiac muscle tissue, which is responsible for the coordinated contraction of the heart. The repeated heart depolarisation and repolarisation gives rise to the different phases of the cardiac cycle. Each cardiac cycle is comprised of

[1] Depolarisation of the heart is the orderly passage of electrical current through sequential regions of the heart muscle, producing a contraction that makes blood flow through the body.

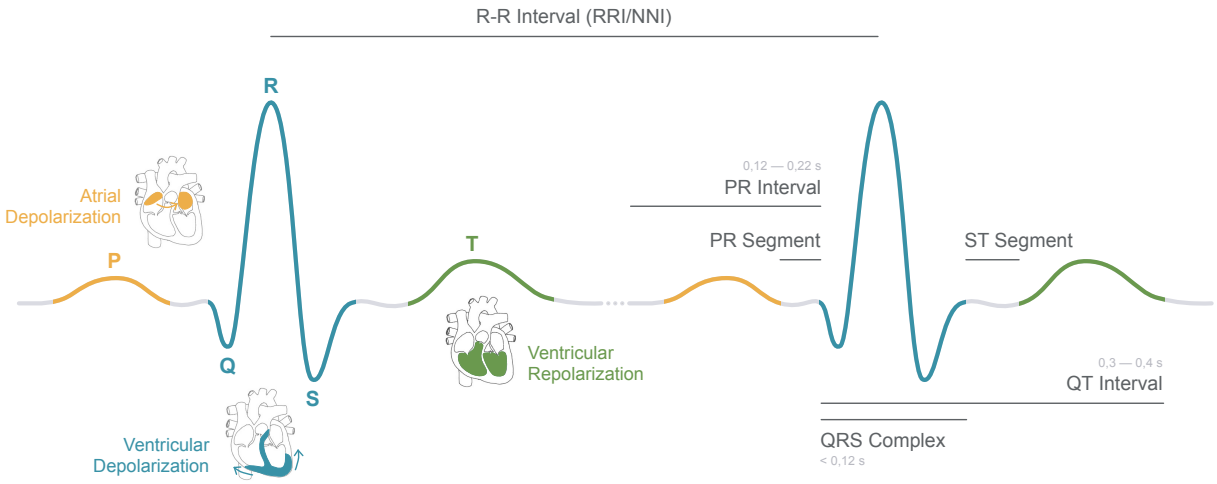


Figure 2.2: Illustration of a typical ECG waveform produced by the human heart as consequence of the depolarisation-repolarisation cycle of the heart chambers. Waves P, Q, R, S and T are the most important curves in each cycle. The QRS complex (in blue) refers to the Q, R and S curves together. The PR interval is the time between the first deflection of the P wave and the first deflection of the QRS. The ST interval is the time between the end of the QRS and the start of the T wave.

three stages: the atrial contraction, the inter-ventricular propagation, and the ventricular contraction [20]. In the absence of cardiac dysfunctions, the electrocardiogram (ECG) signal presents a characteristic pattern, depicted in Figure 2.2. This pattern comprises five main waves – P, Q, R, S and T – that occur in the same temporal order of, and are produced by, those three stages [20].

Acquisition. In clinical settings, an ECG would be acquired by attaching two to ten wet electrodes[2] on the chest and limbs (Figure 2.3A), which can provide up twelve angles of the heart depolarisation. In the digital time series, these are commonly called channels. However, ambulatory and wearable devices usually do not acquire signals with twelve channels, but rather with just one or two on the chest, abdomen or left upper limbs. The *PreEpiSeizures* chestband of Figure 1.1C is an example of such a wearable device. In either case, the analogue acquisition equipment samples a point of the variable being measured at equal intervals. This rate at which points are sampled is called the sampling frequency, and it is usually expressed in Hertz (Hz) [18]. Hence, the digital time series will have as many data points per second as the sampling frequency value.

Preprocessing. Noise is always present in the digital time series of any acquired biosignal. electrocardiography (ECG) signals are usually contaminated with noise coming from respiration [21], sweat release [18], muscle activity [22] and high-amplitude movements of the chest and upper limbs [23]. Wearable acquisitions are even more prone to noise and artifacts than clinical ones [24, 25], because wearables generally use dry electrodes, more prone to noise; whereas clinical units use wet electrodes, more resistant to noise. Effort should be dedicated in attenuating the majority of noise before any further analysis takes place, or the ECG might be wrongfully interpreted. In controlled environment acquisitions, a frequency-domain passband filter will suffice, but that is a naive strategy for non-stationary environ-

[2] An electrode is usually a metal piece that transduces electrochemical potentials to electrical current. It interfaces the body surface with an analogue-to-digital converter (ADC), allowing to record the biosignal in the digital domain [18].

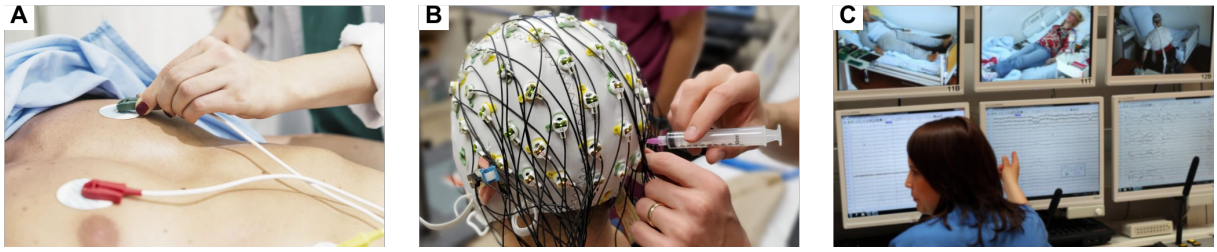


Figure 2.3: Clinical setup at epilepsy monitoring units. A) 1-channel ECG[5]. B) Multi-channel EEG[6]. C) Video monitoring room, where vEEG technicians monitor hospitalised patients.

ments [24, 26, 27].

HRV Features. The time difference (usually in milliseconds) between each consecutive pair of R peaks can create a new time series termed R-R peak interval (RRI) time series. The RRI can also be computed from the photoplethysmogram (PPG) pulse. From the RRI, a popular set of features, rich in physiological information, is usually extracted called the heart rate variability (HRV) features. These features only require one ECG lead, hence being the most appealing choice for wearable ambulatory monitoring algorithms [28]. Some authors consider the heart rate (HR) the most basic HRV feature. A more complex, non-linear feature, the cardiac sympathetic index (CSI), has drawn much attention in recent years, particularly in epileptic seizure monitoring [29]. Moreover, since the CSI computational cost can be somewhat expensive, convolutional neural networks (CNNs) have been proposed to extract approximations of it [30]. Many research groups have found that the values of several HRV features significantly *increase* or *decrease* when a seizure is about to start or has started [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41], which makes them valuable biomarkers for seizure detection and seizure prediction. As shall be seen, most reviewed methods of Section 3.1 use these features.

2.1.2 Electroencephalography (EEG)

Electroencephalography (EEG) techniques allow to *directly* measure the brain electrical activity, by means of electrodes placed on the scalp (non-invasive EEG) or inside the brain cortex[7] (invasive EEG) [42]. In the hospital, in epilepsy monitoring units, both scalp and intracranial signals are recorded (Figure 2.3B). Essentially, the amplitude of one electroencephalogram (EEG) channel is the sum of all the voltage potentials generated by neurons in the electrode vicinity [18, 43], hence being the gold-standard to document epileptic seizure episodes. The characteristic signatures like spikes, spindles, and bursts of epileptic seizures help epileptologists to uniquely identify their onset (beginning) on the EEG trace, as well as the brain region in which the seizure started. These concepts are clarified in Section 2.2.

Furthermore, in epilepsy monitoring units, usually the EEG is integrated in a larger system called the video-electroencephalography (vEEG) (Figure 2.3C), which additional captures video, 1-lead ECG,

[6]Retrieved from `www.mediclife.med.br/exame/eletrocardiograma`. Accessed in February 2022.
[6]Retrieved from `loonylabs.org/2021/06/03/intro-to-ica`. Accessed in October 2022.
[7]The cerebral cortex is the outer layer of neural tissue of the cerebrum.

EOG and RESP, to more accurately document seizures [44]. The *PreEpiSeizures* collaboration between IT, HSM and HEM allowed the gathering of vEEG data of hospitalised patients with epilepsy, from which the ECG signals used in this work were retrieved.

Multiple EEG-based algorithms and machine learning models have also been explored to predict epileptic seizures [45, 46, 47], once evidence was found of EEG changes before seizures arise [48]. It is worth mentioning the relevance of features like the Shannon entropy for seizure prediction, which significantly increases 30 minutes before some seizures [49], or the permutation entropy, which decreases transiently before some other types of seizures [50].

2.1.3 General Biosignal Research

A branch of biomedical research is concerned with the acquisition, storage, processing, and analysis of biosignals. This Section conveys what is this branch of research about, and how can computer engineers contribute to the field. To recapitulate, the previous Sections introduced that:

- A biosignal is any signal produced by the human body, that can be sampled at a fixed frequency by analogue instruments, and digitally converted to one time series per channel.
- Biosignals can be acquired by clinical instruments or by everyday-use wearable devices.
- Noise is always present when acquiring biosignals, even if negligible, which should be attenuated.
- Features can be extracted from these time series to access the body state and to detect events.

As will become clear, several computer engineering disciplines come together to make biosignal research a reality from large research institutes to the Master's student in the library. These include hardware engineering to create components to build biosignal acquisition instruments; network engineering to establish wired and non-wired protocols that transmit the acquired data from these instruments to the designated storage locations; data engineering to make ways to storage large volumes of biosignals in compact files, universally accessible, but yet private if necessary; software engineering to create development environments and research tools to facilitate the analysis of biosignals; and others such as computer graphics to provide powerful visualisation tools, and artificial intelligence (AI) to lead the way on computational methods to learn from biosignals. Here focuses will be on software, data files, and AI.

Acquisition and Storage of Biosignals

Let us start from the beginning: acquiring data. To better illustrate how biosignals can be acquired in clinical settings, yet for research purposes, the *PreEpiSeizures* weekly routine at Hospital de Santa Maria (HSM) will be used as an example.

The research group is acquiring biosignals from patients with epilepsy hospitalised at the HSM's "Laboratório de EEG e do Sono". This collaboration for data gathering has been approved by the hospital

ethical commission. The laboratory is part of the hospital's neurology service, and one of its purposes is to serve as internment facilities for patients to stay while clinicians study their epilepsy condition, do pre-surgical evaluations, or study the effect of ASMs on them. Eng. Mariana Abreu, from *PreEpiSeizures*, is currently responsible for the acquisition of biosignals using the chest-band of Figure 1.1C. When Eng. Mariana is absent, I substitute her. For each patient arriving at the Laboratory, they are invited to participate in the biosignal collection program. Patients are explained the general purpose of the investigation, they are shown the wearable that was developed, and they are asked what data they are willing to consent. If the patient agrees to participate, they or their legal guardians are asked to read and sign an informed consent that legitimises the use of data, naturally complying with all data protection laws, hospital rules and good practices. The study lasts for five business days, which is typically the period the patient stays hospitalised. From herein this will be called the monitoring week.

During the monitoring week, ECG, respirogram (RESP), and accelerometer (ACC) biosignals are continuously acquired by the chestband, and stored in EpiBOX [51], an acquisition system developed by Eng. Ana Sofia Carmo. EpiBOX is placed in a corner of the patient's room, and it connects via Bluetooth to the wearable to store the acquired data; it also provides an intranet for a smartphone to connect and visualise the data being acquired. EpiBOX is capable of processing and interpreting this biosignals, so seizure detection algorithms can be tested on it. In the future, it is planed to move the processing unit to inside the wearable itself. The patient is explained how to put on the wearable and that they can move or remove it at any time that suits them. The patient is also connected to the hospital vEEG equipment with one lead ECG. As part of clinical protocol, all seizures are registered by the nurse on duty in a video surveillance room on the same floor, and self-registered by the patient, when awareness is not impaired during seizures. These approximate onsets are given to EEG expert technicians, who later review the vEEG and identify the precise onset and comment on the seizure. Later, a neurologist reviews again the vEEG to confirm the seizure onsets, and to classify the seizures in a report, to which the research team has access. Besides the chest-band biosignals, the team also has access to the vEEG data from the hospital service computers, for gold-standard comparison. These computers allow to visualise the EEG and select the time periods to export. At the end of every monitoring week, the vEEG, ECG, EpiBox data and clinical reports are collected at the hospital and brought to IT to be stored offline.

Non-clinical studies usually follow similar procedures, although with less restrictions. For instance, another project at IT called *EmotiphAI* [52] acquires electrodermal activity (EDA) and PPG signals from wristbands, to assess the emotional response of subjects while watching movies. In each session, the same set of steps is required, from signing informed consents, explaining to the subjects how to wear the devices, establishing synchronous WiFi connection between all devices, and store the biosignals safely for posterior analysis.

Preprocessing Biosignals

After biosignals have been acquired and stored, researchers tend to organise them by cohorts. A cohort is the set of subjects who participated in some acquisition and that usually have an event in com-

mon, e.g. experienced seizures, or watched the same movie. Biosignals then go through a sequence of preprocessing steps, many of them that are commonly known in the community [17]:

- **Resampling:** Increase or decrease the sampling frequency, to have, respectively, more or less data points per second.
- **Filtering:** Apply digital filters to attenuate noise or to smooth the signal waveforms.
- **Normalising:** Normalise the signal in amplitude boundaries or by removing the signal's mean.
- **Segmenting:** Partition the time series in blocks or segments.
- **Excluding Segments:** Discard portions of time series for presenting low quality, or because the subject has removed the wearable, or any other circumstance that makes some period non-useful.

The order in which this steps were presented is not mandatory, nor all of them are mandatory. Depending on the project, there can be multiple combinations and orders by which researchers conduct the preprocessing stage.

Moreover, there is an uncountable number of processes that researchers can further apply on biosignals, in order to prepare them adequately to subsequent analysis. This includes adding and subtracting channels to obtain new leads, adding more noise on purpose to evaluate some denoising algorithm being proposed, applying any mathematical function to correct the biosignal waveforms, etc. And these can be applied selectively on some intervals of time deepening on which events occurred at those. Depending on the research needs, preprossessing can be an automatic simple sequence of steps, or a more complex pipeline that requires human attention.

Researchers also spend a great deal of time in annotating the biosignals. Biosignal annotation is the task of giving names to intervals of time or timepoints. These points of interest can be called events. An event can start at its onset and end at its offset, and have an associated name, e.g. an epileptic seizure. Or, instead of a time interval, it can be just a timepoint, e.g. when the subject raised a hand. Annotation can be accomplished by visual inspection of a trained specialist on the biosignal modality being annotated. For instance, in *PreEpiSeizures* project, seizure events are annotated by the neurologist, as previously described. Or if it is not a physiological nor clinical event, it can be annotated by the researchers conducting the experiment, e.g. the start and end of the movie in *EmotiphAI* sessions.

Processing and Interpretation of Biosignals

Finally, as exemplified for ECG and EEG, features can be extracted and selected – a stage known as feature engineering – from biosignals separately or together. These can be statistical features, structural features, quality assessment features, etc. Features can be computed by well-defined mathematical functions, however feature selection can be a very cumbersome and error-prone task. Hence, deep learning (DL) models have been proposed to automatically extract features from biosignals [53].

Furthermore, to explore and interpret biosignals, researchers can use unsupervised learning techniques, e.g. to cluster the cohort in groups of emotions while watching a movie. To offer medical decision

support solutions and healthcare applications to the general consumer, researchers usually design supervised learning models, e.g. run fitness trackers with ACC [54, 55], arrhythmia detection with ECG [56], driver drowsiness detection with PPG [57], or multimodal epileptic seizure detection [58, 59].

After acquisition and preprossessing, typically, the more downstream researchers get on their workflow, the more exploratory these analyses get, for they try to answer a research question, not always understood in full. Hence, from start to finish, these experiments need to be tweaked and repeated with as much easiness as possible, a topic that will be addressed in Chapter 4.

2.2 Fundamentals of Epilepsy

In this Section, the necessary knowledge for epileptic seizure prediction is introduced. There is not an unique way a patient can develop epilepsy, but rather it is a complex disease with a spectrum of diverse conditions and clinical manifestations[8] [60]. The causes of epilepsy can range, from genetic conditions, to neurological disorders, cerebrovascular diseases or systemic disorders [61]. Regardless of the cause, the International League Against Epilepsy (ILAE) considers that a patient can be diagnosed with epilepsy if they show unprovoked seizures in a recurrent way [4]. Therefore, when a patient experiences one single seizure, it does not necessarily mean they have developed epilepsy. Built upon decades of research, currently, the theory of how one develops epilepsy is based on the concept of epileptogenesis, which is the chronic process by which a group of neurons become hyperexcitable, generating unprovoked (spontaneous) seizures, called epileptic seizures [4, 60].

An epileptic seizure is manifested by an abrupt change in behaviour, either semiologically[9] or not, caused by abnormally excessive and synchronous neuronal activity in the brain, that can last from a few seconds to a few minutes [62, 63]. In short, a seizure can be triggered whenever a disruption of the homeostasis[10] and correct functioning of the central nervous system (CNS) occurs [60], usually at the biochemical level [63, 64, 65, 66, 67, 68]. There are specific designations for the different periods or stages surrounding a seizure episode. The outbreak period of seizures is called the **ictal** period and starts at the so-called seizure onset [69]. The **preictal** and **postictal** periods are, respectively, the periods immediately before and immediately after the ictal period. When subjects are not experiencing seizures they are in **interictal** periods, which are the periods of rest between seizures [70]. From herein, with respect to (wrt) one seizure, negative time values will refer to periods before its onset, and positive time values will refer to periods after its onset, being zero ($t = 0$) the onset timepoint. Figure 2.4 illustrates the periods surrounding a seizure episode and this notation.

In 2017, the ILAE reviewed the classification of seizure types and, in this classification, seizures are first discriminated by their **onset** type: focal, generalised or unknown [71]:

[8] Clinical manifestations are perceptible, outward, visible expressions of a disease or illness, such as signs and symptoms, observed by a physician or the patients themselves.

[9] Semiology is the study of signs; in medical sciences, "semiologically" means manifesting visible signs.

[10] Homeostasis is the self-capacity of the body to maintain its stability through the regulation of heart rate, respiration, sweat release, digestion, reproduction, and others as such.

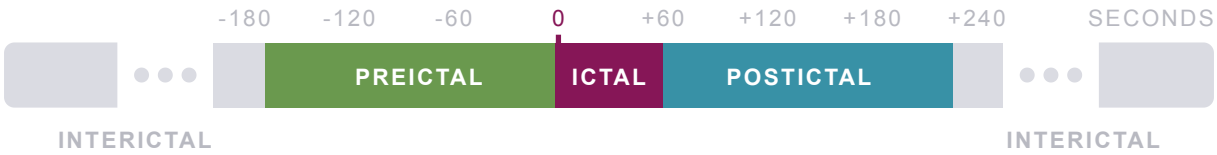


Figure 2.4: Time series representation of an epileptic seizure with onset at $t = 0s$. Seizure outbreak period represented in pink. Periods before and after seizure represented in green and blue, respectively.

- **Focal seizure**[11]**:** Originates in one or more localized regions of the brain.
- **Generalised seizure:** Originates from widespread regions on both hemispheres of the brain.

A focal seizure can be categorized according to the patient's awareness state[12] during the seizure. A **focal aware seizure (FAS)**[13] is the type of focal seizure that does not cause awareness impairment, and the subject is fully aware of it in the ictal and postictal periods. Symptoms may be subtle and last for more than 2 minutes. A **focal with impaired awareness seizure (FIAS)**[14] is the type of focal seizure that causes impairment of awareness and responsiveness, and the subject does not remember the seizure afterwards. This type of seizure may evolve to a generalised condition. If it propagates to both hemispheres, then it evolves to a secondary generalisation, and we call it **focal to bilateral tonic-clonic seizure (FBTCS)**, which presents a tonic-clonic component, described below. If physicians cannot classify the patient's awareness state, it is called a **focal with unknown awareness seizure (FUAS)** [71]. Focal seizures can also be categorised by the brain region they affect. Patients diagnosed with temporal lobe epilepsy (TLE) typically present with seizures on the temporal lobe. In contrast, the term extratemporal lobe epilepsy (XTLE) includes seizures on other brain regions such as frontal, occipital, parietal, or hippocampal, to name a few.

Seizures are also classified by semiology, based on the clinical manifestations patients present during seizure. These manifestations can occur alone or together in pairs or triplets, in focal or generalised seizures. They are divided into motor and nonmotor types [71]. Motor types are:

- **Tonic**: Characterised by sudden and continuous muscle contractions. Sometimes the subject cries or groans, and might bite the tongue or cheeks. The back gets arched, and it causes falling, often backwards [72].
- **Atonic**: Characterised by sudden muscle tone relaxation in head, trunk, jaw or limbs. It causes falling, often forwards. It lasts about 1–2 seconds [73].
- **Clonic**: Characterised by rapidly and rhythmic muscle contractions and relaxations. The period between each rhythmic discharge is usually 67-183 milliseconds [74], or 2-3 discharges per rhythmic cycle [71]. The eyes blink, the elbows, hips, knees also bend and relax, and sometimes there is even frothy saliva and urinary or fecal incontinence [72].

[11] Formerly known as partial seizure.
[12] Awareness is the level of alertness of one's personal state, surroundings, and external phenomena.
[13] Formerly known as simple partial seizure; also, it is still known as Jacksonian march.
[14] Formerly known as complex partial seizure.

- **Tonic-Clonic**: Characterised by a tonic (rigid) phase followed by a clonic (rhythmic) phase [73].
- **Myoclonic**: Presents with myoclonus: quick, involuntary muscle jerk. The seizure is composed of sudden and brief one/multiple muscle jerks over short time (1–2 seconds). There are brief body jerks, usually in facial muscles and limbs. If it occurs in clusters, we call them clusters of short myoclonic-tonic seizures (CSMTS) [73].
- **Epileptic Spasms**: Sudden flexion and extension of muscles of limbs and trunk, that is usually more sustained than myoclonus, but not as sustained as a tonic seizure [73].
- **Hyperkinetic**: Muscles producing sequential ballistic movements, such as pedaling or thrashing. Increase in rate of ongoing movements [73].
- **Automatisms**: Unconscious behaviors such as lip smacking, chewing motions, swallowing, eyelid flutters, unpurposeful walking, and hand fidgeting, picking or rubbing [72].

Nonmotor types are:

- **Autonomic**: Inadequate function of the autonomic nervous system (ANS) function, involving cardiovascular, gastrointestinal, sudomotor, vasomotor, and thermoregulatory functions. Symptoms vary from cardiorespiratory changes, sweating, piloerection, dilation of pupils, incontinence, and unusual feelings of nausea [73].
- **Behavior arrest**: Pause of ongoing activities, freezing, immobilization [73].
- **Cognitive**: Impairment of thinking, language, spatial perception, and memory capabilities, or the inability to plan movements. Speech can become difficult or even impossible [73].
- **Emotional**: Appearance of having an emotion such as fear, anger, sadness, joy, euphoria, laughing or crying; Feelings of derealisation (environment is not real) or depersonalization (dissociation from the environment or self); Feeling of déjà vu [73].
- **Sensory**: Unusual auditory, gustatory, tactile and olfactory sensations (e.g. smell of burnt rubber). If it occurs in the preictal period of another seizure, this component is called **aura** [73].

This is a non exhaustive list of the semiological components a seizure may present, which contains the necessary vocabulary for this work. A glossary compiling important epileptic seizure terms has been made available on Research Gate [75].

Additionally, there are two more types of generalised seizures that require detailed attention: GTCS and GAS. A seizure with generalised onset and tonic-clonic component is called **generalized tonic-clonic seizure (GTCS)**[15] and can become quite dangerous. It may be preceded by hallucinations and

[15]Formerly known as *grand mal* seizure, or primary generalised seizure.

dizziness in the preictal period (aura), and followed by sleepiness, amnesia, and paralysis in arms or legs for minutes to hours following the seizure. Too many frequent GTCSs can lead to SUDEP if failed to provide adequate antiepileptic treatment [76]. A seizure with generalised onset and absence component is called **generalized absence seizure (GAS)**[16] and can be difficult to detect. It is characterized by a sudden onset ceasing the preceding activity, a blank stare, a brief loss of awareness and responsiveness from a few seconds to half a minute. It ends abruptly or followed by automatisms [73]. They are most common in children, and they may lead to complications such as learning difficulties.

Finally, the terms **electrographic**, **subclinical**, or **infraclinical** seizure refer to seizures that have no clinically visible manifestations as the above described [73], making the only way to detect them the analysis of electrical brain activity by means of an EEG.

Changes in the Autonomic Nervous System (ANS)

The autonomic nervous system (ANS) plays a crucial role in maintaining the body homeostasis. Several evidence has shown that besides the abnormal neuronal brain activity that can be recorded in EEG, there are usually some autonomic dysregulation symptoms that accompany the seizure in the preictal, ictal or postictal stages [77]. These dysfunctions can be detected in non-EEG biosignals (also known as peripheral biosignals) and serve as biomarkers to detect or predict the occurrence of seizures. These symptoms include:

- Ictal tachycardia (the increase of heart rate over 100-120 bpm) in up to 80-100% of seizures [78, 79, 80, 81, 82, 83, 84, 85, 86], particularly in FASs, FIASs and FBTCSs in [86, 87, 88, 89, 90].
- Preictal tachycardia [91, 92], between 1 and 50 seconds before onset [78, 81, 86, 88, 90, 93, 94].
- Ictal bradycardia (the decrease of heart rate below 60 bpm) [81, 95, 96, 97, 98, 99], reported with less frequency than tachycardia, with approximately 1.3% to 28% incidence [86, 88, 90, 100, 101, 102] in FASs, FIASs and FBTCSs.
- Multiple dysfunctions of the heart depolarization cycle, such as atrial fibrillation, sinus arrhythmia, supraventricular tachycardia, atrial and ventricular premature depolarizations, bundle branch block and atrioventricular nodal escape rhythm in 39-42% of the patients [86, 103, 104], and various distortions of the electrocardiographic trace [103, 105, 106, 107], shown in Figure 2.2.
- Ictal and preictal parasympathetic-sympathetic unbalanced control, the two subsystems of the ANS, visible in heart rate variability (HRV) changes [108, 109, 110, 111, 112, 113, 114, 115].
- Ictal hyperventilation (rapid or deep breathing) [116, 117, 118].
- Ictal hypoxemia (low blood oxygen saturation), with oxygen saturation dropping to 70-80% lasting more than 70 seconds [119, 120, 121, 122, 123, 124].

[16] Formerly known as *petit mal* seizure.

- Ictal apnea (temporary cessation of breathing) [76, 118, 120, 125].
- Ictal sweating in FASs, FIASs, and GTCSs [126, 127, 128, 129, 130, 131].
- Piloreaction (erection of skin hair) [132, 133, 134, 135, 136].
- Ictal involuntary muscle activation in seizures with tonic, clonic, and myoclonic components [74, 137, 138]. Particularly, the tonic phase of tonic-clonic seizures shows higher amplitude and activity in higher frequencies than that of isolated tonic seizures [138].

2.3 Revision in Deep Learning Theory

Deep learning, a form of machine learning, can be used with biosignals in classification, regression, denoising, and feature extraction tasks [19, 139]. This Section revises core concepts regarding deep learning for biosignal feature extraction and classification, on top of which the proposed approach lies.

2.3.1 Artificial Neural Networks

An artificial neural network (ANN) is a function f that maps input tensors $\mathbf{x}$ into output tensors $\hat{\mathbf{y}}$, that is, $\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are trainable parameters. This function, f, is given by a composition of K other transformations [140]:

$$f(\mathbf{x}) = (f^{(K-1)} \circ f^{(K-2)} \circ \cdots \circ f^{(1)} \circ f^{(0)})(\mathbf{x}) \ . \tag{2.1}$$

In terms of the network architecture, each $f^{(k)}, k \in \{0, \ldots, K-1\}$ is represented by a parameterised (or weighted) layer. Hence, $\boldsymbol{\theta} = \cup_k \boldsymbol{\theta_k}$, where $\boldsymbol{\theta_k}$ is the set of parameters of each layer k. Each layer has a collection of units[17]. In the traditional fully-connected layers, the units of arbitrary layer k, $\mathbf{z}_k$, are computed by applying a linear transformation to the output of the previous layer, $\mathbf{x}_{k-1}$ [140]:

$$\mathbf{z}_k = \mathbf{w}_k^T \cdot \mathbf{x}_{k-1} + \mathbf{b}_k \ , \tag{2.2}$$

where $\mathbf{w}_k \in \boldsymbol{\theta_k}$ are the weights of layer k, and $\mathbf{b} \in \boldsymbol{\theta_k}$ is the intercept[18] of layer k, being $\mathbf{z}_k$ the output propagated forward. The values of these weights and intercepts can be readjusted iteratively, in a way that makes the model output tensors as similar as possible to the targets, as will be detailed next. Usually, at the end of each weighted layer, a nonlinear activation function, o, is applied to the units. This function is a design choice. To date, the most popular activation function in the community is the rectified linear unit (ReLU) function [139]:

[17] Also known as nodes or neurons.
[18] Also known as bias.

$$o(\mathbf{z}) \coloneqq \max(\mathbf{z}, 0) \ . \tag{2.3}$$

For the network's last weighted layer, usually the identify function is used for regression tasks, whereas the Sigmoid and Softmax functions are used, respectively, for binary and multi-class classification tasks. The Sigmoid function is defined as [139]:

$$\sigma(\mathbf{z}) \coloneqq \frac{1}{1 + \exp(\mathbf{z})} \ . \tag{2.4}$$

2.3.2 Training ANN Models

To assert the mapping f is correctly learned, ANNs are trained to minimise a loss function, $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, where $\hat{\mathbf{y}}$ is an output of $f(\mathbf{x})$, and $\mathbf{y}$ is the expected target, for any given input $\mathbf{x}$. A dataset with a significant number of examples $(\mathbf{x}, \mathbf{y})$ should be given to train the network. Traditionally, the model parameters, $\boldsymbol{\theta}$, are updated for each example, following the stochastic gradient descent algorithm. However, it is common practice to update $\boldsymbol{\theta}$ with multiple examples at a time, i.e., in mini-batches. The steps of the back-propagation algorithm that update $\boldsymbol{\theta}$ are summarised below [140]:

1. Input an example object, $\mathbf{x}$, or a batch of them to the model.
2. Perform forward propagation with Equations 2.2 and 2.3.
3. Compute the loss, $\mathcal{L}$, between outputs, $\hat{\mathbf{y}}$, and the respective targets, $\mathbf{y}$.
4. Perform backpropagation to get the gradient loss, $\partial\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})/\partial\mathbf{w}$.
5. Update the model parameters, $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \gamma \cdot \partial\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})/\partial\mathbf{w}$.

Given the functional composition of Equation 2.1, the back-propagation algorithm can efficiently compute at once the partial derivatives of Step 4 $\forall\theta_{\mathbf{k}} \in \theta$ using the chain rule of Calculus [141]. The term γ in Step 5 is the learning rate, which is useful to slow down or speed up the rate at which parameters are updated by what was learnt from each example or batch [140].

To evaluate the performance of a model, the dataset examples are commonly divided in two disjoint sets: the train set and the test set. The train set is used in the described learning process, whereas the test set is reserved to evaluate the model after training [140], with the metrics described in Section 2.4. The train set is further divided to reserve a portion of examples to what is termed the validation set. The learning process described occurs in epochs, and at the end of each epoch, the model's generalisation error – the validation loss – is evaluated with the validation examples, and some hyperparameters can be updated by an algorithm called the optimizer. Hyperparameters cannot be trained as the model's parameters are, but they can be optimised between epochs [139]. For instance, the learning rate is an hyperparameter that can be fixed by design or be automatically adapted. The optimizer used in this work, the Adam optimizer [142], adaptively optimises the learning rate per parameter.

A model is said to generalise well if the test loss is similar to that of the training loss. That can be probed during the training process by looking at the evolution of the validation loss. The iterative learning process only stops when some criterion is satisfied, usually when the validation loss being probed is below a satisfactory threshold, or the maximum number of epochs defined has been executed [139].

Conversely, a model does not generalise well if it cannot predict accurately the unseen examples of the test set after being trained [140]. This can occur in two phenomena: (i) overfitting, when the average test or validation loss is much higher than the train loss; or (ii) underfitting, when the train loss is high and, consequently, so is the validation or test loss [139]. Both of these scenarios can be avoided with careful design choices, regularisation techniques, and by gathering datasets with enough statistically representative examples.

2.3.3 Convolutional Layers

A common layer traditionally used in computer vision applications is the convolution layer. A convolutional layer is one that uses convolution operations to find the values for its units, instead of dense multiplication of weights and biases (Equation 2.2). An ANN is said to be convolutional if it has at least one convolutional layer. The convolution operation, $*$, is defined as [139]:

$$s(t) \coloneqq (x * K)(t) = \int x(a) \cdot K(t-a)\, da\ , \tag{2.5}$$

where $\mathbf{x}$ is an arbitrary input, $\mathbf{K}$ a chosen kernel, t any given time, and a traverses the neighbourhood of t. Essentially, for any given t, a weighted average around $\mathbf{x}(t)$ is obtained with its neighbourhood points. For a 1-dimensional (1D) input, like a 1-channel biosignal, a 1D kernel is usually used, and the convolution operation in its discrete form is given as:

$$S[i] \coloneqq (x * K)[i] = \sum_m x[m] \cdot K[i-m]\ . \tag{2.6}$$

There have been established variants to this equation, such as including stride and padding parameters. Stride allows to skip a number of positions of input, so that kernel multiplications do not overlap, at the cost of not extracting features from every point. This is equivalent to downsampling the input of each layer [139]. Padding the input allows the output not to shrink after every layer, since without padding the output reduces $|K|-1$ at every layer. Hence, if $|K|-1$ padding is added, the output after each layer will remain the same, if no other architectural changes take place. A padding larger than the kernel size can also be added, so that the points in the kernel border can receive as much attention as the rest [139]. These design choices can be defined empirically to increase the model performance.

Convolutional layers satisfy three useful properties [139]:

- **Sparse weights:** When the size of $\mathbf{K}$ is smaller than the size of $\mathbf{x}$, richer and more detailed features from small portions of the input can be extracted and stored in fewer parameters than those of traditional dense layers. Besides the advantage of improving statistical efficiency, it also reduces time and space complexity.
- **Parameter sharing:** Since each element of $\mathbf{K}$ is used at every input position, they can be reused, contrarily to what occurs in dense multiplication. Hence, learned parameters share computational space. This property further reduces spacial complexity.
- **Equivariant representations:** The convolution operation is naturally invariant to the input translations. For example, applying $S(\mathbf{x}-5,\mathbf{K})$, where $\mathbf{x}$ is shifted by 5 points, yields the same output as $S(\mathbf{x},\mathbf{K})$. Hence, convolution is said to be equivariant to shifted representations. The equivariant property will be leveraged in the proposed method, to extract features from ECG blindly segmented. However, convolution is not equivariant to rescaling or rotation.

In general, like traditional layers, each convolution layer applies its linear transformation that is passed through a nonlinear activation function, o (Equation 2.3). Additionally, convolutional layers usually have a third stage associated called pooling, that *summarises* the layer outputs with a statistic. This statistic can be, for instance, to keep only the maximum values or the average of values within rectangular neighborhoods, or other [139]. When taking the maximum or the average of each pair of units of the *current* layer, the *next* layer will have half the units, improving computational efficiency over dense layers. This also makes models invariant to translations on inputs, but at a rather local level [139].

2.4 Metrics to Evaluate Seizure Detection or Prediction Algorithms

Now that a full overview has been given about biosignals, in particular the ECG and its relationship with epileptic seizures, as well as the theory behind deep learning classifiers, this Section introduces the common metrics used to evaluate seizure monitoring algorithms.

On the topic of epileptic seizure identification, biosignals can be segmented in a variety of ways, and usually a model classifies each segment as preictal, ictal or interictal period. This can be a ML model or not. In turn, these classifications are analysed by an algorithm that triggers the alarms, based on some decision strategy. Throughout the review process of Section 3.1, it was found that this distinction was not always clear and, sometimes, was even confused. Being these two parts of a together-working system, they must be evaluated in separate, each with the adequate metrics [143]. Here, a clear distinction is made about the object of evaluation: the segment classification model and the decision algorithm.

When evaluating segment classification models, commonly accepted classification metrics by the community should be employed, such as sensitivity and specificity [144]:

$$Sensitivity\ (\%) = \frac{TP}{TP+FN} \tag{2.7}$$

$$Specificity\ (\%) = \frac{TN}{TN + FP} \tag{2.8}$$

where TP is the amount of true positives, TN is the amount of true negatives, FP is the amount of false positives, and FN is the amount of false negatives. The area under the curve (AUC) of the sensitivity-specificity plot is a metric relating both, also commonly employed. Furthermore, other metrics report other aspects of classification results, such as the positive predictive value (PPV) and the F1-score:

$$PPV\ (\%) = \frac{TP}{TP + FP} \tag{2.9}$$

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.10}$$

The goal is to design a model that achieves the highest values possible for the above metrics.

When evaluating the decision algorithm, the same metrics can also be applied for the decision itself, if the result is similar to a classification, e.g. issue an alert (binary decision). In the context of epileptic seizure detection and prediction, it is also of importance to evaluate the false alarm rate (FAR) and the latency of correct seizure alarms. The FAR is usually expressed in the number of false alarms per day (FAR/d or FAR/24h). The latency is the time elapsed since the alarm is issued and the seizure onset marked on the EEG:

$$Latency = t_{alarm} - t_{onset}\ \ (minutes) \tag{2.11}$$

It is usually expressed in seconds or minutes. Negative values are used in case of predictions, like illustrated in Figure 2.4, conveying an anticipation result.

Chapter 3

Epileptic Seizure Prediction

People think that epilepsy is divine simply because they don't have any idea what causes epilepsy. But I believe that someday we will understand what causes epilepsy, and, at that moment, we will cease to believe that it's divine. And so it is with everything in the universe.

– Hippocrates, 460–370 BC

As introduced in the previous Chapters, patients with refractory epilepsy would benefit from a wearable device capable of acquiring and analysing peripheral biosignals to look for seizure activity. Current opinion among neurologists is that there is a need for automated seizure prediction systems, if possible using non-EEG wearable devices, and, if they show a good performance, they could in the future be brought to the clinical practice [101, 145, 146]. Adequate wearable devices have already been engineered to continuously acquire peripheral biosignals, such as head-bands, chest-bands, and wrist-bands [147]. The next step is to develop algorithms that predict epileptic seizures from the non-EEG biosignals, so that alerts can be issued by these wearables whenever a seizure has been or is about to be elicited.

A systematic review on this topic is presented in Section 3.1. An ECG-based deep learning approach is proposed in Section 3.2. The preparation of datasets with ECG being acquired at both collaborating hospitals is presented in Section 3.3, as well as data augmentation techniques to extend them. How the networks were trained and their hyperparameters tuned is presented, respectively, in Sections 3.4 and 3.5. Finally, the cohort results of the proposed method are presented and discussed in Section 3.6.

3.1 State-of-the-Art

A PRISMA[1] systematic review was conducted, which is known for being a systematic and reproducible method that can be repeated by other researchers. The method allowed for the identification and screening of relevant and eligible clinical studies that proposed an original algorithm for automatic

[1] Visit `prisma-statement.org` for details.

human epileptic seizure detection or prediction, implemented it in a piece of software, and tested its performance. The end-goal of the reviewed algorithms is to issue an alarm when recognizing the onset of the ictal stage of a seizure (detection alarm) or its preictal stage (prediction alarm). The review included only non-EEG-based methods, that is, only peripheral biosignals, like the ECG, EDA, EMG, PPG, ACC, temperature and sound, in order to summarise the most suitable methods to be implemented in discreet wearable devices. The full review can be found in [16]. In this Section, primary concern will be on the ECG-based methods, since this is the biosignal modality chosen for the proposed method of Section 3.2.

To be included in this review, studies had to include at least 10 patients with at least 15 recorded epileptic seizures. An appropriate description about the type of seizures according to the previous or new ILAE vocabulary also needed to be provided. Studies since 2005 were included, hence, for coherence, it was necessary to translate any deprecated vocabulary into the currently used – the one introduced in Section 2.2. A standard terminology enables the proper comparison between studies performed under the same conditions. The complete set of criteria, which is further detailed in [16], guarantees the studies included in the review were at least phase 2 studies, according to the five phases of development and clinical validation of seizure detection devices (Beniczky and Ryvlin, ILAE, 2018 [148]). From 10524 records found on public databases, only 91 met the eligibility criteria to be part of this review, from which only about 27% represent ECG-based methods. Today, the majority of algorithms proposed are still EEG-based.

To open up ground, methods based on HR and HRV features are firstly introduced. It follows a summary with the most prominent research questions already answered by the community, and later how the reviewed methods perform differently for different seizure types. Finally, some EEG-based methods are presented for comparison with the ECG-based.

3.1.1 HR and HRV Features as Seizure Biomarkers

Generally, detection and prediction algorithms partition the signals into segments of seconds or minutes (e.g. 15s or 15m), from which features can be extracted or the ECG time series can be used as is. These segments, or their features, are given as input to binary or multi-class ML or non-ML models, that classify them as interictal, preictal or ictal segments. Most of the algorithms here reviewed issue an alarm on the presence of successive preictal or ictal segments. This Subsection focuses only on methods relying on one or multiple HRV features.

Regarding chest-abdominal ECG wearables, a small one called ePatch [149] was used to extract the cardiac sympathetic index (CSI) to be used on a threshold model meant to detect GTCSs, FBTCSs, FIASs, FASs, and FUASs. The model obtained a sensitivity of 87.0%, a FAR of 0.9/day and 0.22/night, and a median latency of 35 seconds for 11 patients marked as *reponders*[2] (57.9%) [150]. In another

[2]In cohort clinical studies, *responders* are the patients that respond to a treatment, diagnosis or other proposed method. One of the eligibility criteria of this review was that authors could not cherry-pick and report only the results of patients that responded to the proposed method. The full cohort results had to be disclosed even if the main results were those of the *responders*.

work, thresholds on the HR showed 60% sensitivity and 4.3 FAR/night in detecting 59 GTCSs and generalized tonic seizures (GTSs), as well as 27 CSMTSs and hypermotor seizures [151].

Regarding 1-lead clinical ECG, focal motor non-specific seizures were detected with time feature thresholds, with 86% sensitivity, but huge 172.8 FAR/day [152]. Temporal and frontal FASs, FIASs and FBTCSs were predicted with 78.59% sensitivity, mean anticipation of 4.5 minutes, and 5.04 FAR/day, using HRV feature thresholds [153]. The HR increase was shown to detect 36 FIASs and 24 tonic-clonic (TC) seizures with 88% sensitivity, 18.07 FAR/day and a mean latency of 11.9 seconds [154].

Regarding ML approaches, a variety of features extracted from ECG can effectively be used by support vector machine (SVM) models to detect seizures. SVM models trained with HRV and other more complex features can detect FIASs and FBTCSs with sensitivities of 81.89–94.1%, FARs of 9.84–11.76/day, and 6–18s mean latencies [155, 156, 157, 158]. Instead of classifying segments into *preictal* and *non-preictal* classes, Smirnov *et al.* (2017) trained a SVM only with *non-preictal* segments, while testing it with both. This approach, called anomaly detection, detects every anomaly for being out of the ”normal” pattern, including seizure preictal periods. Its main advantage is being ample and inclusive of pattern heterogeneity of different seizures, but, as a disadvantage, it detects every anomaly falling out of the ”normal” pattern, such as arrhythmias. Eight ECG time and frequency features were extracted, from which only the most significant was selected, and the model achieved a 100% and 92% prediction sensitivity and specificity, respectively [159].

3.1.2 Answering Research Questions

Now that a broad sense of the approaches being proposed has been introduced, this Subsection answers some of the most important questions that were tackled in the published review.

Are ML better than non-ML methods? Only one group has directly answered this question with the same dataset. Varon *et al.* (2013) [160] have shown that clustering models perform better than traditional threshold algorithms at detecting and predicting seizures with HRV features. On 90 focal and generalised seizures from 35 pediatric patients, a kernel spectral clustering (KSC) model achieved higher sensitivities (57.3–84.7%) than predefined thresholds (50.0–77.0%) on HRV features and respiratory-derived features (RDFs) extracted from clinical ECG signals. No other work has addressed this question.

Should a patient-specific approach be followed? Patient-specific strategies include training one classification model for each patient only with its own seizures as examples, or separately selecting significant features for each patient, if applicable. Patient-independent approaches do not follow any of these strategies. On 285 FASs, FIASs, FBTCSs and subclinical seizures, included in 3751 hours of clinical ECG, De Cooman *et al.* (2018) HRV-based SVM models achieved a 157% reduction in FAR when using a patient-specific approach, compared to the patient-independent pair [161]. On 227 temporal FIASs, included in 2172 hours of clinical ECG, the same group showed that patient-specific HRV-based SVM models achieved 2/3 of the FARs the patient-independent models showed. However, the detection sensitivity of the patient-specific model was 73%, while for the patient-independent model

it was 76%. The mean detection latency of the patient-specific models were of 21s [162]. There seems to be always a benefit from following a patient-specific approach, but to do so with machine learning models requires a dataset with a sufficient number of seizures per patient, that allows for good intra-patient generalisation of the learning process.

How much should be the prediction horizon? When predicting seizures, the duration of what the algorithm considers to *look* for patterns in the preictal period matters. Billeci *et al.* (2019) [163] studied the influence of the duration that models consider to be the preictal period, called the prediction horizon. Their proposed SVM model achieved higher prediction sensitivities and specificities when the preictal period was considered to be larger. Concretely, sensitivities and specificities were, respectively, $\approx$ 64% and $\approx$ 48% analysing the 5 minutes before onset, $\approx$ 72% and $\approx$ 51% analysing the 15 minutes before onset, and 84% and 73% analysing the 30 minutes before onset. This later model could predict seizures 18.2 minutes (median) before onset, however the FAR was severely high (358/day). The models were tested on 38 focal seizures of 12 patients, with patient-specific HRV features extracted from clinical ECG.

Why ECG and not PPG? PPG sensors can be worn at much less pervasive body locations, like the wrist (Figure 1.1B), whereas ECG sensors can be worn much more discreetly in the chest, below the clothes (Figure 1.1C). So, if equivalent HRV features can be extracted from both, which modality is more suitable for epilepsy monitoring? In Vandecasteele *et al.* (2017) [164], the detection sensitivity and specificity using chest ECG were 70.0% and 50.6%, respectively, whereas using the wrist PPG were 32.0% and 43.2%, respectively. A total of 47 fronto-temporal FIASs were tested, all signals were acquired with non-clinical instruments and an SVM model was used. The lower PPG performance is most likely attributed to its propensity to noise and sensor saturation [57].

3.1.3 Studies Comparing Different Types of Seizures

Cooman *et al.* (2018) [165] tested 107 seizures from 28 pediatric patients on a clustering and SVM patient-specific algorithm. The algorithm was based on the HR and HR increase of clinical ECG. The detected seizure types, sorted by the one with highest sensitivity to one with the lowest sensitivity were: tonic-clonic seizures (96.0%), generalised non-specific seizures (83.3%), subtle seizures (82.8%), focal temporal seizures (82.1%), hyperkinetic seizures (72.5%), focal frontal seizures (71.4%), and tonic seizures and clonic seizures (42.2% for both types). Focal temporal seizures exhibited a higher FAR of 4.24/night, relatively to focal frontal seizures, which exhibited a FAR of 2.32/night. The lowest FAR occurred in generalised seizures (1.60/night). The mean detection latency was also lower for generalised seizures (15.0s), followed by frontal seizures (16.9s), followed by temporal seizures (26.9s). Moreover, tonic-clonic and hyperkinetic seizures were detected on average in half of the time than tonic, clonic, and subtle seizures.

Behbahani *et al.* (2014) [166] found that HRV features of FBTCSs delivered more detection sensitivity (86.66%) and less specificity (90.00%) to a multilayer perceptron (MLP) classifier, than HRV features

of FIASs (sensitivity of 83.33% and specificity of 96.11%). The dataset of 206 focal seizures was reasonably balanced and HRV features were extracted from clinical ECG of 15 patients. No patient-specific approach was followed.

In Jeppesen *et al.* (2019) [167], with a CSI threshold, all (100%) of tonic-clonic seizures with generalisation (FBTCSs and GTCSs) were accurately detected, while focal seizures without generalisation (FASs, FIAS and FUASs) showed a lower sensitivity of 90.5%. These results regard only 53% of their cohort, which they considered the *responders* to the algorithm. The tested seizures could be detected with a median latency of 30 seconds, while exhibiting FARs of 1.0/day and 0.11/night on 3180 hours.

The KSC model proposed by Varon *et al.* (2013), previously reviewed in Subsection 3.1.1, detects and predicts focal seizures (frontal and temporal) with 27.4% higher sensitivity and 2.2 seconds faster than generalised seizures (tonic, tonic-clonic, and myoclonic) in [160]. This was again confirmed in another work [168], where focal and generalised seizures were detected, respectively with 100% and 93.1% sensitivities. Another work [169] with the KSC model reconfirmed that focal seizures (temporal and frontal) were better detected and predicted than generalised seizures (tonic, tonic-clonic, and absence) of 37 pediatric patients, with similar results. The mean alarm latency was 20s.

There are multiple aspects that make a machine learning model suitable to detect/predict a specific type of seizure, from the input it takes to the design of its architecture and trainable parameters. Some designs better detect seizures of focal onset, others are more accurate for generalised ones. It depends very much on how the model was trained and on the semiology expressed.

3.1.4 EEG-based Methods

Some studies have shown that the performance of EEG-based models increases when joining ECG features to the learning process [170, 171, 172]. Nevertheless, the use of scalp EEG alone is still the gold-standard and most studied technique to detect all types of seizures, including subclinical seizures. Threshold detection models can achieve 83–100% sensitivities, some with a FAR as low as 0.48/day, others with an unacceptable FAR of 261/day (1 false alarm every 3 minutes) [173, 174, 175, 176, 177, 178, 179, 180]. ML models, such as MLPs and CNNs were also proposed for seizure detection, achieving sensitivities of 74–89% and FARs of 4.8–13.0/day [181, 182, 183, 184]. Absence seizures (GASs) are a niche of research with very high detection sensitivities of 98–99.1% to SVM, least squares support vector machine (LS-SVM), and CNN models [185, 186, 187, 188, 189].

For both tasks of detection and prediction, models such as SVMs, recurrent neural networks (RNNs), and gaussian mixture models (GMMs) have been proposed, yielding sensitivities of 85–100%, FARs of 0.55–38/day, and mean anticipations of 51s–22.5m [190, 191, 192]. Although SVM models detect seizures quite well, CNN models seem to outperform them in sensitivity, in faster detections and in less false alarms. A 3D-CNN trained with frequency features was able to predict seizures, on average, 14.1 minutes before onset [193]. In order to ECG-based approaches to be accepted by patients and in the market, these need to perform at least as good as EEG-based solutions.

3.2 Proposed Approach

In this work, deep residual networks (ResNets) are proposed to predict epileptic seizures from the ECG. ResNets have been very successful in a variety of computer vision tasks [194]. Inclusively, ResNets have been explored to detect epileptic seizures from the EEG [195]. However, up to the author's knowledge, ResNets have not yet been experimented to detect epileptic seizures from the ECG. As introduced before, there is strong motivation to monitor epileptic seizures from non-EEG biosignals, since these can be acquired by wearables at more discreet body locations. This section explains the proposed approach to predict epileptic seizures with ResNet models and a decision algorithm.

3.2.1 Proposed Network Architecture

The proposed architecture can be divided in 3 parts: the input block, a sequence of R residual blocks, and a classification block. Herein, the network's depth will be considered to be the number of weighted layers, i.e. convolutional layers. To extract the first set of features, the input block starts with a convolutional layer, followed by a batch normalisation layer, followed by a ReLU activation layer.

Next follows a sequence of residual blocks. All residual blocks have similar structure, except some of them which downsample the input (explained ahead). For networks with depth inferior to 50 layers, each residual block contains two sub-blocks with the following ordered layers: Batch normalisation, ReLU activation, Dropout, Convolutional. For networks with depth equal or superior to 50 layers, each residual block contains three sub-blocks of these, to prevent the vanishing gradient problem [196]. Figure 3.1 schematically illustrates the full architecture, where the residual block is represented only with 2 sub-block repetitions. In either case, dropout layers zero elements of the input with probability 0.5, as suggested in [197]. Also, convolutional layers have a kernel size of 16, as suggested in [198], apply SAME padding to the input, have increasing number of filters, and stride by

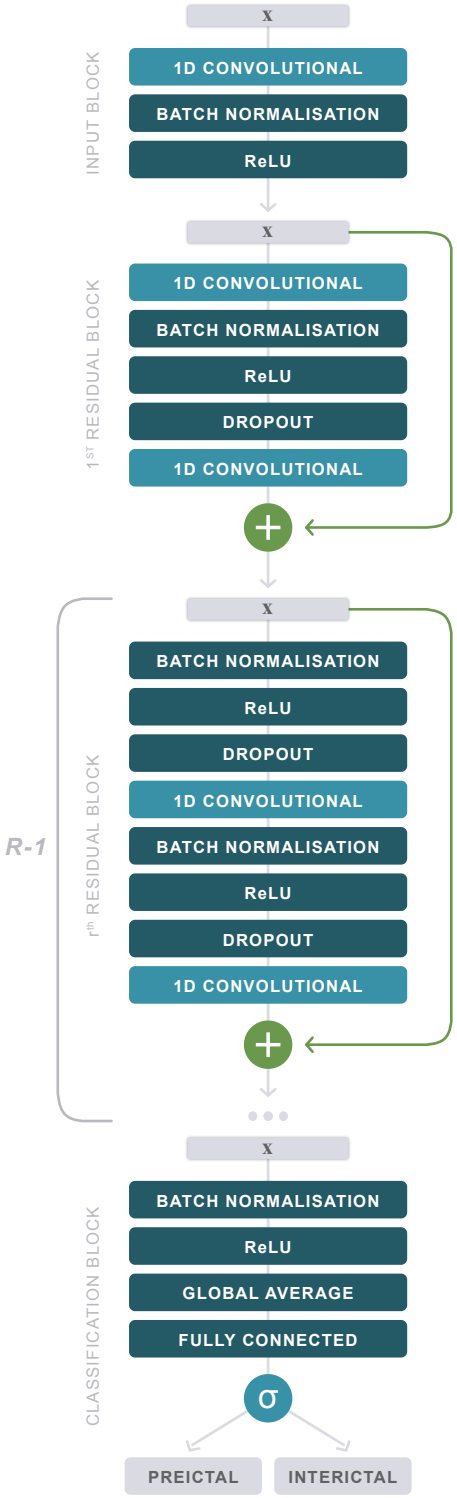


Figure 3.1: General ResNet architecture proposed. The first residual block is smaller, and from the second onwards the same structure can be repeated $R-1$ times. Green connections represent the shortcut connections. '+' represents addition. 'σ' represents the Sigmoid function.

1, except the ones that downsample the input (explained ahead). Additionally, the input of each residual block is added to its output, by what is termed a *shortcut connection*, or an *identity shortcut*, in order to learn the *residue* between the two, and propagating that forward (explained ahead). Theoretically, there can be $R > 0$ residual blocks in between the input and the classification block.

Lastly, the classification block contains a batch normalisation, followed by a ReLU activation, followed by a global average of each extracted feature, followed by a fully connected transformation onto one unit, activated by a Sigmoid layer. The output should represent the probability of the input segment being a preictal segment. The complete forward procedure is detailed in Algorithm 1. The next subsections detail particular processes of this architecture and this forward procedure.

Algorithm 1: Forward Procedure of Proposed ResNets

input : $\mathbf{x}$, R, D, F, F_0
output: $\mathbf{x}$

begin

$filters \leftarrow F_0$

/* Input Block */

$\mathbf{x} \leftarrow \texttt{Convolution}(\mathbf{x}, filters, 1)$
$\mathbf{x} \leftarrow \texttt{BatchNormalisation}(\mathbf{x})$
$\mathbf{x} \leftarrow \texttt{ReLU}(\mathbf{x})$

/* Residual Blocks */

for $r \leftarrow 0$ **to** R **do**
  $\mathbf{identity} \leftarrow \mathbf{x}$
  if $r \neq 0$ **then**
    $\mathbf{x} \leftarrow \texttt{BatchNormalisation}(\mathbf{x})$
    $\mathbf{x} \leftarrow \texttt{ReLU}(\mathbf{x})$
    $\mathbf{x} \leftarrow \texttt{Dropout}(\mathbf{x})$
  if $r \mod F = 1$ **then** (UPFILTER)
    $filters \leftarrow 2 \times filters$
  if $r \mod P = 1$ **then** (DOWNSAMPLE)
    $\mathbf{x} \leftarrow \texttt{Convolution}(\mathbf{x}, filters, 2)$
    $\mathbf{identity} \leftarrow \texttt{MaxPooling}(\mathbf{identity})$
  else
    $\mathbf{x} \leftarrow \texttt{Convolution}(\mathbf{x}, filters, 1)$
  $\mathbf{x} \leftarrow \texttt{BatchNormalisation}(\mathbf{x})$
  $\mathbf{x} \leftarrow \texttt{ReLU}(\mathbf{x})$
  $\mathbf{x} \leftarrow \texttt{Dropout}(\mathbf{x})$
  $\mathbf{x} \leftarrow \texttt{Convolution}(\mathbf{x}, filters, 1)$
  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{identity}$ (SHORTCUT)

/* Classification Block */

$\mathbf{x} \leftarrow \texttt{BatchNormalisation}(\mathbf{x})$
$\mathbf{x} \leftarrow \texttt{ReLU}(\mathbf{x})$
$\mathbf{x} \leftarrow \texttt{GlobalAverage}(\mathbf{x})$
$\mathbf{x} \leftarrow \texttt{FullyConnected}(\mathbf{x}, 1)$
$\mathbf{x} \leftarrow \texttt{Sigmoid}(\mathbf{x})$

Identity Shortcuts

Shortcut identity connections are the central property of residual networks, since they allow to compute learning residues along the network. Let $\mathcal{G}_r$ be the mapping to be learned at each residual block r,

through its combination of convolutional, activation, normalisation, and dropout layers. Then, the output of each residual block r can be expressed as:

$$\mathbf{y} = \mathcal{G}_r(\mathbf{x}) + \mathbf{x} \,, \tag{3.1}$$

where the addition is accomplished by a shortcut that connects the initial identity of $\mathbf{x}$ to the output mapping, $\mathcal{G}_r$. Figure 3.1 illustrates these shortcut connections in green arrows. Also, this connection is explicit in line *SHORTCUT* of Algorithm 1. Note that, the addition of both tensors is performed element-wise for each channel (or feature).

Continuous Downsampling

As aforementioned, it is important the networks are *forced* to prioritize which features most contribute to accurate classifications. The convolutional layers of this architecture sequentially extract features from the input ECG, and the most relevant should be selected as depth increases on the way to the classification block. To that end, every D residual blocks, the first convolutional layer of a block strides by 2, hence downsampling the input. This is explicitly stated in line *DOWNSAMPLE* of Algorithm 1. Since this downsampling occurs every D residual blocks, the input gets continuously downsampled through the network. The length of inputs reaching the classification blocks has influence on the classification performance, and it depends on the gap D and the network's depth, $\propto R$. Therefore, the optimal pair of values for hyperparameters D and R should be investigated.

Continuous Increase of Filters

Every F residual blocks, the number of filters of both convolutional layers of each residual block can also duplicate. In this way, more features get extracted from the input as it is downsampled. This is explicitly stated in line *UPFILTER* of Algorithm 1. Extracting more features can help to identify features that are more meaningful and that can help the model to generalise better. The value of F can equal the value of D, in which case the time complexity is kept the same for every residual block, as originally proposed in [196], but that is not mandatory. Therefore, the optimal pair of values for hyperparameters F and the number of filters in the input block, F_0, should also be investigated.

3.2.2 Decision Algorithm: Triggering Alarms

Given a stream of segment ResNet classifications, the goal is to have a second algorithm that decides if and when an alarm should be raised. A simple algorithm, based on the ones the works reviewed in Section 3.1 already use, would be to count the number of consecutive positive (preictal) segments and, if that exceeds a threshold, an alarm is raised. A modification of this strategy has been devised, and here it is proposed a similar decision rule that allows interruptions on this counting. That is, for instance, if the threshold was to count 10 consecutive positive segments, now that counter is not reset if isolated negative segments are found. Figure 3.2 illustrates this strategy in four examples. From left to right,

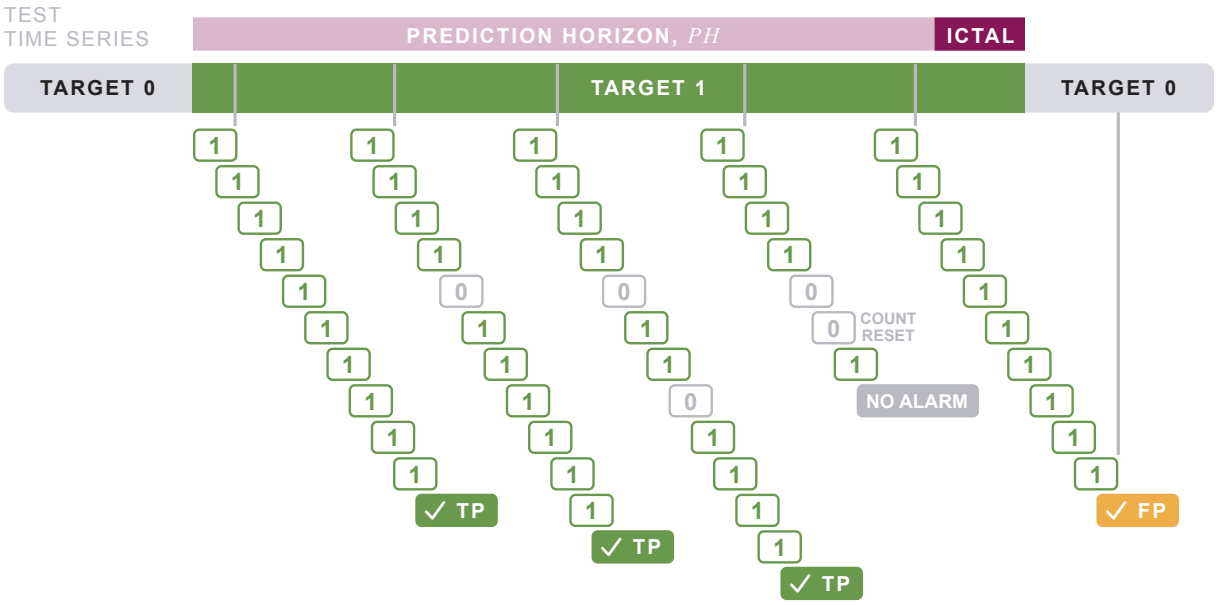


Figure 3.2: Examples of five decision scenarios. In the three left-most examples, true positive (TP) alarms are raised. In the fourth example no alarm is raised. In the right-most example, a sequence of 10 consecutive positively classified segments occurs, hence an alarm is raised, however only when the seizure has ended, so it is a false positive (FP) alarm.

in the first example, a sequence of 10 consecutive positively classified segments occurs, and a true positive (TP) alarm is raised. In the second and third examples, a sequence of 10 positively classified segments occurs, although with isolated interruptions of negatively classified segments, hence a true positive (TP) alarm is raised. In the fourth example, a sequence of positively classified segments occurs but it is interrupted by two negatively classified examples, so the count is reset, and no alarm is raised. Notice the potential this strategy has in mitigating the impact of false negatives coming from the ResNet classification. Algorithm 2 formally defines this decision rule, where output 1 represents an alarm.

Algorithm 2: Proposed Decision Rule to Raise Preictal Alarms

input : $\mathbf{c}$, segment classifications
output: $1 \vee 0$

begin
$nConsequitivePositives \leftarrow 0$
$lastOneWasNegative \leftarrow 0$
for $s \in \mathbf{c}$ **do**
  if $s = 1$ **then**
    $nConsequitivePositives \leftarrow nConsequitivePositives + 1$
    $lastOneWasNegative \leftarrow 0$
  else
    if $lastOneWasNegative = 1$ **then**
      $nConsequitivePositives \leftarrow 0$
    $lastOneWasNegative \leftarrow 1$
  if $nConsequitivePositives = 10$ **then**
    return 1
return 0

3.3 Datasets for Evaluation

The ECG signals, retrieved from the vEEG system, of epilepsy patients hospitalised at HSM and HEM were used to evaluate the proposed method. A cohort of 11 adult patients with a total of 107 recorded seizures was selected. From these, only 88 seizures where clinical seizures, that is, seizures that can potentially be predicted or detected from peripheral biosignals. These patients were selected due to preictal HRV changes found in previous works, and because these patients present some of the highest number of seizures – a necessary condition to design patient-specific DL models with good generalisation. Also, this cohort conforms with the minimum number of patients and seizures demanded in the systematic review of Section 3.1. Table 3.1 summarises this cohort and their types of seizures that were recorded during the monitoring week.

Preprocessing Methods

All ECG time series used were preprossessed with a finite impulse response (FIR) bandpass filter, with a [1, 40] Hz passing band, and order 200. This design allowed to attenuate most of the noise described in Section 2.1.1, although not all noise like motion artifacts due to the patient normal movement of the chest and limbs. The first block of Figure 3.3 represents this filter.

Table 3.1: Cohort of anonymised patients with epilepsy and their seizures recorded during the monitoring week. '#' represents number of seizures. Only the clinical seizures (88) were included in the datasets.

Code	Sex	#	Description of Seizures	Total Hours
2XF9	F	7	1 FAS and 6 FUAS, all in the right temporal lobe. All with automatisms and one with myoclonus.	92
I1HP	M	9	1 FBTCS and 8 FIAS, all starting in the left temporal lobe. All with automatisms and 3 of them with preictal aura.	71
6QXD	F	11	7 FIAS and 3 FUAS with automatisms and preictal aura. 1 Sensory FAS with preictal aura. All in left temporal lobe.	91
Z410	F	9	All FUAS with automatisms and preceded by preictal aura. All in the right fronto-temporal lobe.	44
YD2L	F	10	All Sensory FAS in the right temporal lobe.	41
S8SG	F	6	All FIAS with automatisms, in the right temporal lobe.	69
I204	M	7	5 FAS of the right temporal lobe, three of them non-specific non-motor and two emotional and sensorial. 2 FIAS with behaviour arrest on both temporal lobes.	98
IFW2	F	10	All FUAS while asleep, so consciousness was not tested.	97
58QF	F	14	6 FAS and 8 electrographic.	100
RR6Z	F	18	5 FAS, 2 FUAS and 11 electrographic.	97
RE38	M	6	2 of them are non-specific motor. 3 of them had impaired awareness. Onset types not declared.	93
11	3:8	107	88 clinical seizures; 19 electrographic seizures	893

Figure 3.3: Pipeline of preprocessing steps applied on the acquired ECG. From left to right is a passband filter, a resampling operation, a normalisation in amplitude, a segmentation, and the exclusion of the segments with poorer quality. The remaining segments proceeded to dataset preparation.

All filtered ECG time series were then resampled to 80 Hz – second block of Figure 3.3 – to decrease the total size of data. This new sampling frequency allows for maximum data volume reduction, based on the Nyquist theorem. Reduction to 80 samples per second ensures no aliasing, albeit it cannot be guaranteed potentially seizure biomarkers are not suppressed. Nonetheless, as reviewed in Section 3.1, most authors agree that meaningful ECG features lie between the [1, 40] Hz band [31]. Since these time series were originally sampled at 250 Hz, total data volume was reduced approximately 3.1x (times).

All time series were also normalised in amplitude between 0 and 1 – third block of Figure 3.3 – to facilitate convergence of convolutional layers' parameters. The commonly known "minimum-maximum" method was followed:

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \tag{3.2}$$

After normalisation, all time series were segmented in windows of N duration – fourth block of Figure 3.3. Each segment could later come to constitute a dataset example. The concrete duration of these segments, N, and their overlap, N_O, are studied ahead in Section 3.5.

Discarding Low-Quality Segments

Well-annotated datasets with reasonable quality examples are crucial to attain acceptable performances. However, real data acquired in non-controlled environments is far from perfect. Moreover, it is impractical to access the quality of long-term biosignals solely by visual inspection. Hence, venturing to automate this process, two ECG signal quality indexes (SQIs) were used as heuristics of segment quality [199, 200, 201]. The first is the QRS-power signal quality index (pSQI), which evaluates the QRS complexes quality, by taking their relative power. It is computed as:

$$pSQI = \frac{\int_5^{15} P_{\mathbf{x}}(f)\ df}{\int_5^{40} P_{\mathbf{x}}(f)\ df} \tag{3.3}$$

where $P(\cdot)$ is the power spectrum distribution function, $\mathbf{x}$ is any ECG signal, and f is an arbitrary frequency. The majority of QRS power is around 5 and 15 Hz [202], hence the ratio being of that frequency band over the reaming ECG characteristic band. An ECG segment with optimal QRS complex (QRS) quality shows $0.5 \leq$ pSQI ≤ 0.8, otherwise it can be considered to have unacceptable quality. The second is the kurtosis signal quality index (kSQI), which evaluates the ECG absence of noise in general. It is computed with the fourth statistical moment (kurtosis) of the signal:

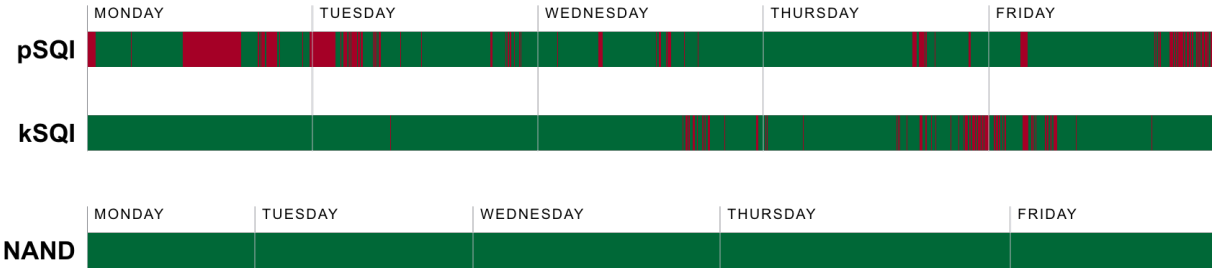


Figure 3.4: Application example of rule 3.5. All red segments were discarded for presenting poor quality.

$$kSQI = \frac{E\{(\mathbf{x} - \bar{\mathbf{x}})^4\}}{\sigma_{\mathbf{x}}^4} \tag{3.4}$$

where $\mathbf{x}$ is any ECG signal, $\bar{\mathbf{x}}$ its mean, and $\sigma_{\mathbf{x}}$ its standard deviation. Here the expected value, $E(\cdot)$, is the average. Segments with kSQI > 5 present low levels of noise, whereas segments with kSQI ≤ 5 present with unsatisfactory quality.

Using pSQI and kSQI, the following discarding rule was devised:

$$\neg\left[(0.5 \leq pSQI(\mathbf{x}) \leq 0.8) \wedge (kSQI(\mathbf{x}) > 5.0)\right] \tag{3.5}$$

If the above NAND rule was true for any segment $\mathbf{x}$, that segment was discarded. Figure 3.4 visually illustrates this procedure applied to one of the cohort patients. In this example, every segment not conforming with the quality boundaries of pSQI is highlighted in red in the top timeline; every segment not conforming with the quality boundary of kSQI is highlighted in *red* in the middle timeline; the result of discarding all of the *red* segments is illustrated in the bottom timeline. For each patient, only the *green* segments would constitute the final useful time series. Table 3.2 indicates the percentage kept of each patient's time series – the useful percentage.

Seizure Cross-Validation Datasets

Models were trained and tested using a leave-one-seizure-out cross-validation approach. That is, each patient time series with S recorded seizures was split in two time series, one including all $S - 1$ seizures, from which the train dataset would be created, and another including the remaining seizure, from which the test dataset would be created. In each fold, the test time series started 3 hours before the testing seizure onset and ended 3 hours after its offset, hence lasting 6 hours plus the seizure duration. Figure 3.5 schematically illustrates how a patient time series is split for a fold, in a monitoring week with 4 seizures. In each fold, the metrics introduced before were computed as detailed in Section 3.4.

Table 3.2: Useful percentage (%) of each patient time series, after discarding poor quality segments.

2XF9	I1HP	6QXD	Z410	YD2L	S8SG	I204	IFW2	58QF	RR6Z	RE38
82.7	73.1	42.7	76.7	54.4	63.1	91.2	88.0	87.2	82.4	79.5

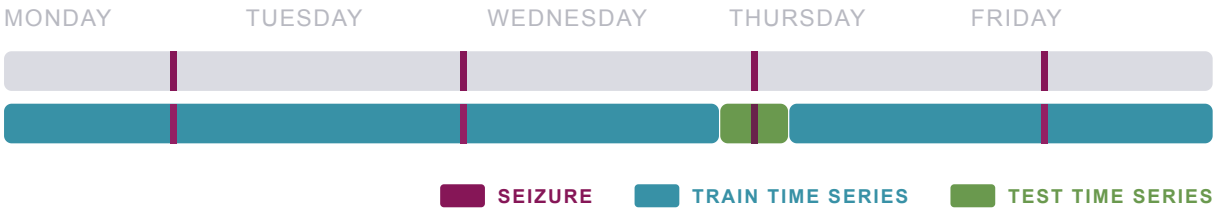


Figure 3.5: Example of time series split for train and test datasets. In this case, the third cross-validation fold (S3) is exemplified, i.e., the segments of the third seizure and its surroundings are reserved for testing (green), whereas the remaining will constitute the train dataset (blue).

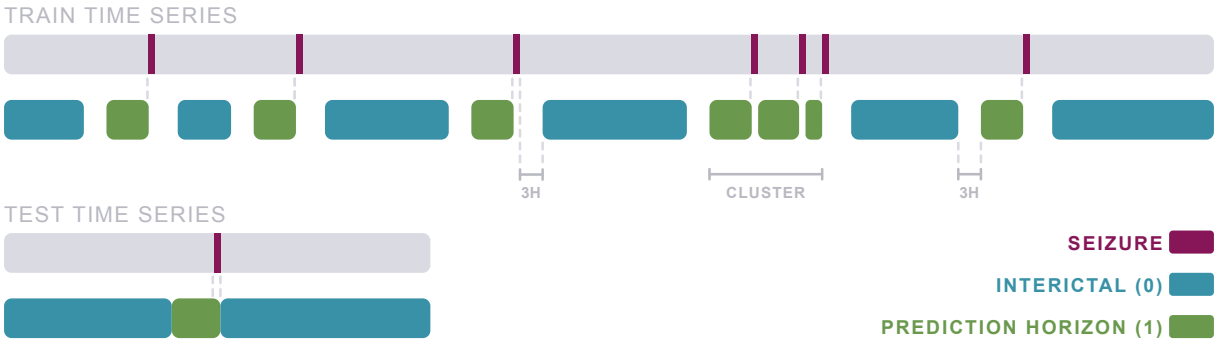


Figure 3.6: Example of train and test timeseries target annotation. In this case, the train time series includes 7 seizures, from which 3 are a cluster. The interval defined as the preictal horizon is annotated with target 1 (green), whereas the remaining segments are annotated with target 0 (blue). In the absence of clusters, the 3h transition intervals between classes were discarded in train time series.

Annotating Segments

Each segment was annotated with a target. Since this is a binary classification task, segments in the preictal period of seizures were annotated with 1, and the remaining segments, by exclusion assumed to be the interictal period, were annotated with 0. But the preictal period of seizures lacks a clear definition. Many of the authors reviewed in Section 3.1 have recognised this problem, and a straightforward answer is yet to be found, if any [143]. This doubt is further aggravated when one asks if the preictal period is specific of each seizure type, or even patient-specific. Additionally, to define preictal period one must be aware that this can have somewhat different meanings depending on the application: describing preictal clinical signs or symptoms, identifying EEG preictal spikes, or labeling segments for ML training. Hence, for this later application, authors have adopted the term Prediction Horizon (PH) to refer *to what will the model assume to be* the preictal period, which does not have to necessarily intersect entirely the clinical or EEG preictal periods. This definition can be motivated by the features observed in the biosignal being used for prediction, and by how soon models are designed to anticipate seizures. A common understanding in ML related work is that the prediction horizon (PH) should not be longer than 1 hour. In this work multiple PH values were evaluated in Section 3.5.

In the training datasets, the ictal segments, well-defined by each seizure onset and offset, were excluded from the dataset, since it is not the models' goal to identity the ictal period. As introduced in Section 2.2, ictal ECG can be quite different than interictal ECG, and, moreover, several motion artifacts were visually identified in ictal ECG, presumably attributed to electrode displacement due to high-amplitude

chest and upper limb movements, identifiable in vEEG video recordings (not shown). Furthermore, in preliminary experiments, it was found that excluding the interictal-preictal transition periods and ictal-interictal transition periods from the training datasets decreased models' generalisation error. These excluded segments are represented by the gaps between green and blue periods in Figure 3.6.

None of the above mentioned exclusions was applied in the testing datasets, in order to simulate online prediction, where every segment is fed to the system, and to correctly access the number of false alarms in the time series. This is also explicit in Figure 3.6 as the absence of gaps between green and blue periods. Besides that, in testing datasets, the ictal segments were also annotated as 1, in case detection takes place instead of prediction. Despite ictal segments not being what the models were trained to discriminate, positive predictions (1) in these segments should not be considered false positives, but rather late true positives. Seizure detection is also of value for patients and clinicians, and, as reviewed in Section 3.1, some models that were trained to predict seizures showed to only detect them, and vice-versa, mostly because, in some patients, preictal and ictal semiologies are similar.

Automation of segment annotation is essential due to the total volume of data being used (893h). However, not all time series are fit for a straightforward division, as that presented so far. For instance, some patients experienced clusters of seizures within 3-hour periods, invalidating the described division. Figure 3.6 illustrates one of these clusters. In these cases, it was found prudent to annotate all segments in the PH of seizures, even if not complete, and to not discard transition periods. The ictal segments continue to be discarded.

Data Augmentation

As can be drawn from Table 3.1, even if a PH of 1 hour is considered, there will be many more interictal segments than preictal ones. Unbalanced datasets are one of the main problems in DL, for which normalising the loss with class weights can come as a naive strategy, because more and different examples are indeed needed to learn. In previous work, ECG augmentation techniques have proved to be effective in minimising generalisation error, and, in some cases, even to allow convergence of validation loss [203, 204]. In this work, the same augmentation techniques were applied on the preictal segments, to balance them with interictal ones, only on the train datasets. These techniques are [205]:

- **Scale:** Contraction or expansion, in amplitude, by a multiplier, M (Figure 3.7A).
- **Shift:** Left or right translation, in time, by $D \times$ number of samples (Figure 3.7B).
- **Sine:** Addition of a sinusoidal wave of random frequency, f, and amplitude A (Figure 3.7C).
- **Randomness:** Addition of gaussian noise of amplitude A (Figure 3.7D).
- **Drop:** Multiplication of each sample by zero with probability p (Figure 3.7E).

Parameter M, in Scale, should be between $[0.25, 1[$ for contraction or between $]1, 4]$ for dilation. Parameter D, in Shift, should be between $]0, 1[$. The maximum displacement is achieved when $D = 0.5$. The shift direction (left or right) is random. Parameter p, in Drop, should be between $]0, 1[$, since it

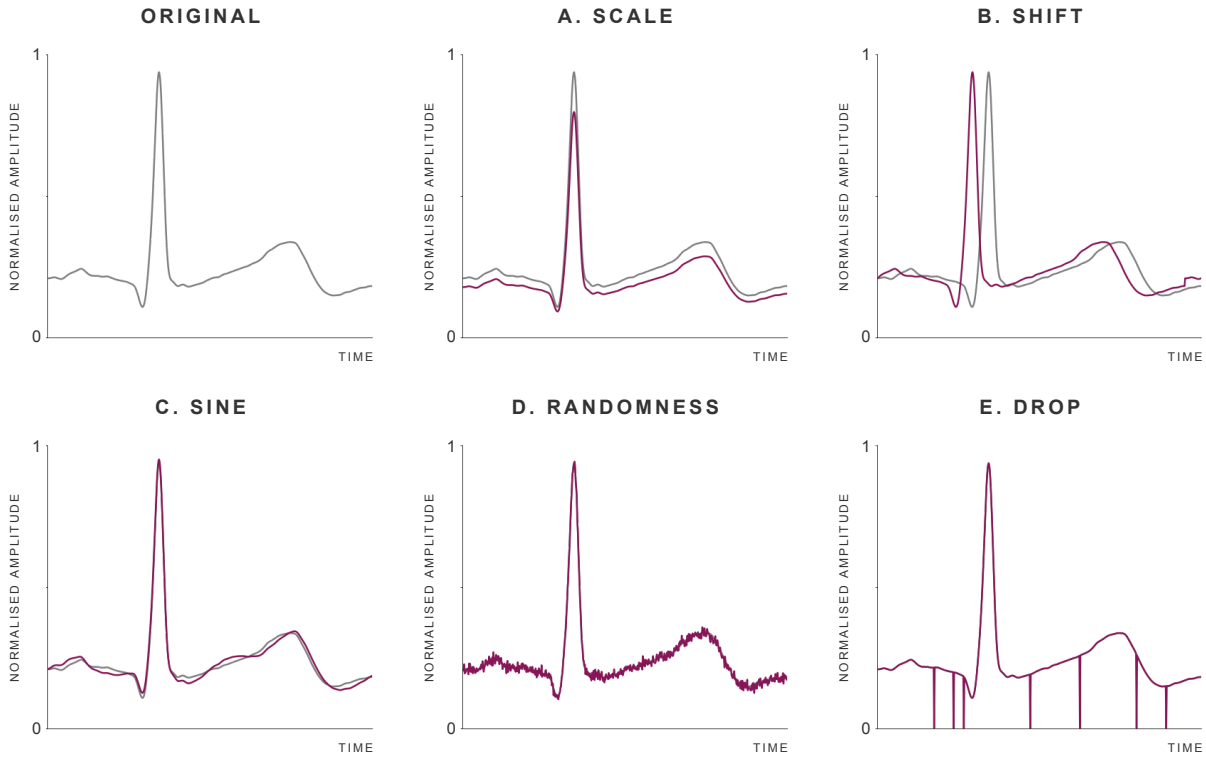


Figure 3.7: Examples of derived ECG segments when applying data augmentation techniques, in pink. Original time series in grey. Scale $M = 0.85$. Shift $D = 0.05$. Sine $A = 0.02$. Randomness $A = 0.01$. Drop $p = 0.02$. Example heartbeat waveforms with duration of 700 ms.

is a probability. Parameter A should be between $]0, 1]$ in Sine, $]0, 0.02]$ in Square and Randomness, so that the natural biosignal morphology does not get significantly altered. Frequency f is random between $[0.001, 0.02]$ in Sine, and between $[0.001, 0.1]$ in Square. These were the recommended values in [205]. In this work, the set of values explicit in Figure 3.7 were used. Notice how the derived segments could have very well been acquired in a real scenario, due to environmental variability. Using these five transformations, the preictal segments were augmented up to the number of interictal segments.

3.4 Experimental Methodology and Empirical Evaluation

This Section firstly explains how the networks were trained and tested. In second, it explains how the decision algorithm was evaluated. Finally, details regarding the computer hardware used are provided.

ResNet Classification Models

All models were trained with the Adam optimiser [142], as suggested in [196, 206], using the default implementation[3] of PyTorch, with $betas = (0.970, 0.999)$. The initial learning rate, L_0, and batch size, B, were considered training hyperparemeters, and are studied in Section 3.5. Additionally, an hard-coded 10^{-1} decrease of learning rate was implemented if no validation loss improvement occurs in 15 epochs. Also, the early stopping mechanism interrupted training after 20 epochs with no validation loss improvement.

[3]Available in full at `https://pytorch.org/docs/stable/generated/torch.optim.Adam`. Accessed in June 2022.

Since this is a binary classification task, the binary cross entropy loss was minimised between the targets, $\mathbf{y} = \{y_0, y_1, ..., y_{B-1}\}$, and the output probabilities, $\hat{\mathbf{y}} = \{\hat{y}_0, \hat{y}_1, ..., \hat{y}_{B-1}\}$. The average train and validation loss of each batch is given by:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{B}\sum_{b=0}^{B}\left[-[y_b \cdot \log \hat{y}_b + (1-y_b)\cdot \log(1-\hat{y}_b)]\right] \ . \tag{3.6}$$

The average test loss is given by a weighted version of the loss function:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{B}\sum_{b=0}^{B}\left[-w^{y_b} \cdot [y_b \cdot \log \hat{y}_b + (1-y_b)\cdot \log(1-\hat{y}_b)]\right] \ . \tag{3.7}$$

where w^1 is the preictal class weight and w^0 the non-preictal class weight. This weighted version is only used to compute the average test loss, because the train datasets are balanced with data augmentation techniques, whereas the test datasets are not. The class weights are given by Equation 3.8.

$$w^0 = 1 - \frac{number\ of\ non\ preictal\ examples}{total\ number\ of\ examples} \ \wedge \ w^1 = 1 - \frac{number\ of\ preictal\ examples}{total\ number\ of\ examples} \tag{3.8}$$

Furthermore, the models were evaluated on the test datasets with the sensitivity, specificity, PPV and F1-Score, introduced in Equations 2.7-2.10, in each CV fold.

Decision Algorithm

The decision algorithm is only evaluated on the test sets (unseen seizures and respective surroundings). It was evaluated using the F1-score (Equation 2.10), where a true positive (TP) is an alarm raised in the prediction horizon or ictal periods, and a false positive (FP) is an alarm raised out of the prediction horizon or ictal periods. The true negatives (TNs) and false negatives (FNs) are defined in opposition. Figure 3.2 illustrates this classification, where the last timepoint of each segment is the *current time*, i.e., when the decision would be made. Notice that the last example is classified as a FP.

Moreover, the number of FPs in each test time series was counted, and the FAR was estimated as:

$$\text{FAR} = \frac{24}{6} \times \text{FP} \ \ \text{(per day)} \ , \tag{3.9}$$

since each test time series had 6 hours of non-preictal and non-ictal period. The FAR of each patient was estimated by averaging the FARs of all CV folds.

The latency of each prediction or detection was computed with the difference between the alarm timepoint and the seizure onset marked on the EEG, following Equation 2.11.

Computer Hardware

The majority of the experiments were conducted in an Apple M1 Pro machine with 16 GB of unified memory. The training of models used the chip's Neural Engine and graphical processing unit (GPU) structures, each of these with another 16 GB of memory each. To that end, the training acceleration was implemented with the PyTorch library [207] and trained in its Metal Performance Shaders (MPS) backend. Other two machines were used to train the networks, with NVidea GPU acceleration also offered by PyTorch. These had both 32 GB of memory and 16 GB of dedicated graphics memory.

3.5 Tuning Hyperparameters

In this section, the way to the find adequate values for the described hyperparameters is reported. Firstly, the architectural dimensions were investigated: R, D, F and F_0. Secondly, how examples are created was investigated, in particular values for N, N_O and PH. Thirdly, a favourable initial learning rate, L_0, and batch size, B, were found. Once the most favourable values were determined in each of these stages, they were fixed for the following one.

Architectural Dimensions

The hyperparemeters of the proposed architecture were investigated, namely:

- R, the number of residual blocks (proportional to the network's depth);
- D, the downsample gap between residual blocks;
- F, the upfilter gap between residual blocks;
- F_0, the initial number of filters.

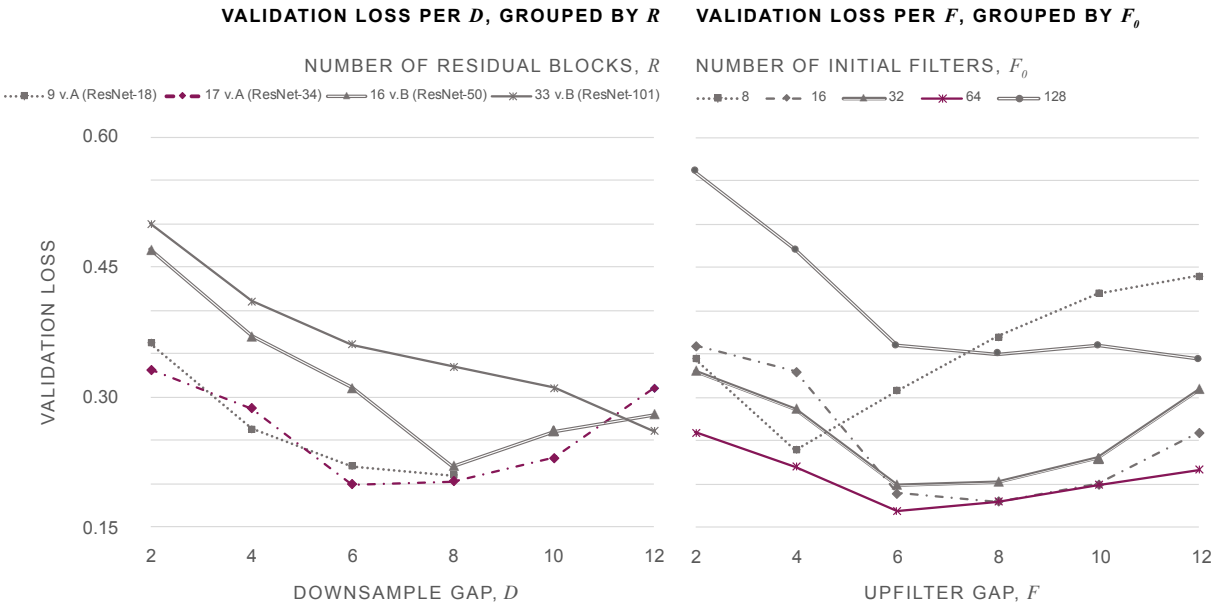


Figure 3.8: Cohort median of CV average validation loss for different architectural hyperparameters. Left: Different values for D and R, while fixing $F = D$ and $F_0 = 32$. Right: Different values for F and F_0, while fixing $R = 17$ and $D = F$. Groups containing the lowest loss value highlighted in pink.

In the first stage, favourable values for the pair (R, D) were found. The R values $\{9, 17\}$ lead to a network with less than 50 weighted layers, hence residual blocks had two repetitions. The R values $\{16, 33\}$ lead to a network with more than 50 weighted layers, hence residual blocks had three repetitions. These values of R correspond to the common variants ResNet-18, ResNet-34, ResNet-50, and ResNet - 101 [196]. Simultaneously, the values $D = \{2, 4, 6, 8, 10\}$ were experimented. The set of values chosen for D is in the range of those already proposed in the literature for ECG classification [198]. The value of F was fixed to D ($F = D$), and $F_0 = 32$ was also fixed[3]. The left panel in Figure 3.8 plots the cohort median of average CV validation losses. The pair with lowest validation loss was $(R, D) = (17, 6)$ in a ResNet-34 architecture; as such, the following experiments were conducted with this architecture.

In the second stage, favourable values for the pair (F, F_0) were found. The values $F_0 = \{8, 16, 32, 64, 128\}$ were experimented, while varying $F = \{2, 4, 6, 8, 10\}$. These sets of values are in the range of those suggested in [198]. The D value was fixed to F ($D = F$). The right panel of Figure 3.8 plots the cohort median of average CV validation losses. The pair with lowest validation loss was found to be $(F, F_0) = (6, 64)$ in the ResNet-34 architecture[3]. Therefore, the architectural design $(R, D, F, F_0) = (17, 6, 6, 64)$ was chosen and fixed for the following experiments.

Formulation of Examples

In the third stage, how dataset examples are created was investigated, in particular:

- N, the ECG segment length (in seconds).
- N_O, the overlap between segments (in percentage).
- PH, the prediction horizon, i.e. preictal interval considered as a true positive target (in minutes).

[3]The remaining hyperparameters were fixed: $N = 15$ seconds, $N_O = 0\%$, $PH = 15$ minutes, $L_0 = 10^{-3}$, $B = 128$.

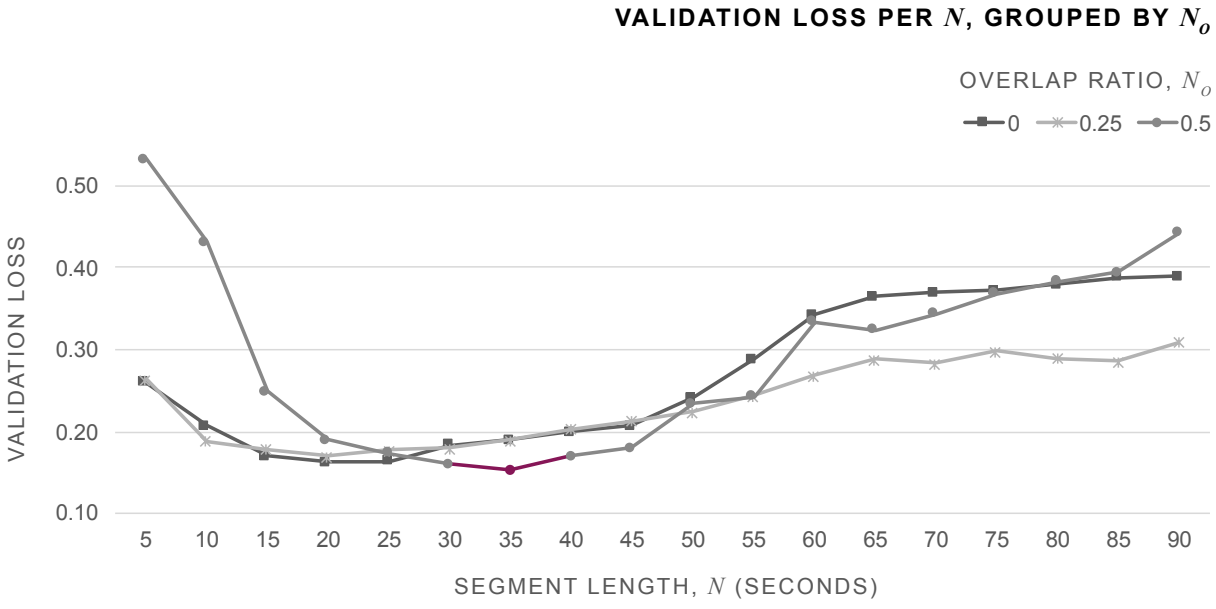


Figure 3.9: Cohort median of CV average validation loss for different ECG segment lengths, group by segment overlap. Lowest loss values highlighted in pink.

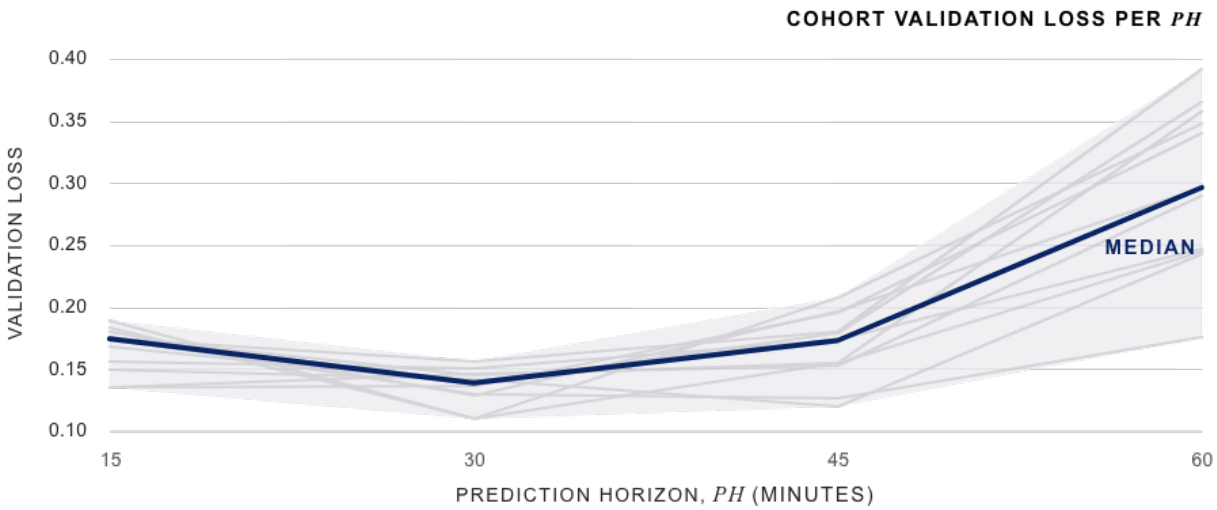


Figure 3.10: CV average validation loss for different seizure prediction horizons. Average intra-patient losses in grey. Cohort median loss in blue.

Using the $(R, D, F, F_0) = (17, 6, 6, 64)$ architecture[4], the most favourable values for the pair (N, N_O) were found. Segmenting the ECG in windows of $N = [5, 90]$ seconds was experimented with no overlap, 25% overlap and 50% overlap. Given that the time series were resampled at 80 Hz, this corresponds to $N = [400, 7200]$ samples and $N_O = [0, 3600]$ samples. Figure 3.9 plots the cohort median of average CV validation losses. The lowest validation loss was found for $(N, N_O) = (35, 0.5)$, that is, 35-second segments with 50% overlap, although the losses in the $N = 35$ neighborhood were very similar. Validation loss in the interval $N = [15, 20]$ seconds with no overlap was also satisfactory, which goes in hand with [197]. However, as previously discussed, in order to extract inter-beat variability features, a longer segment seemed more prudent, hence all ECG time series were segmented in 35-second segments with 50% overlap. Moreover, Figure 3.9 shows longer segments ($>$ 1 minute) hinder the models' performance, presumably due to a higher volume of information from which to learn a pattern.

In [208] and [197], the authors segmented the ECG by heartbeats (≈600ms) and 16-second segments, respectively. However their proposed models are to classify ECG conduction disorders, such as arrhythmias or atrial fibrillation, that can be diagnosed observing solely features from one heartbeat at a time. In this work, it is hypothesised that inter-beat features are to be extracted by the weighted layers, hence feature extraction from 35-second segments is a more suitable configuration.

On a fourth stage, different prediction horizon values, PH, were investigated. This matter of target annotation not only influences test performance metrics, but, more importantly, it has the ability to *augment or restrict* the extension of the pattern to be learned. The values $PH = \{15, 30, 45, 60\}$ minutes were evaluated[5], and the average CV validation losses of the cohort are plotted in Figure 3.10. The value $PH = 30$ minutes attained the lowest median validation loss, although for some patients validation loss was lower for $PH = 45$ minutes. It would be understandable and desirable to have patient-specific prediction horizons, however, for cohort evaluation homogeneity, the PH was fixed at 30 minutes.

[4]The remaining hyperparameters were fixed: $PH = 15$ minutes, $L_0 = 10^{-3}$, $B = 128$.
[5]The remaining hyperparameters were fixed: $L_0 = 10^{-3}$, $B = 128$.

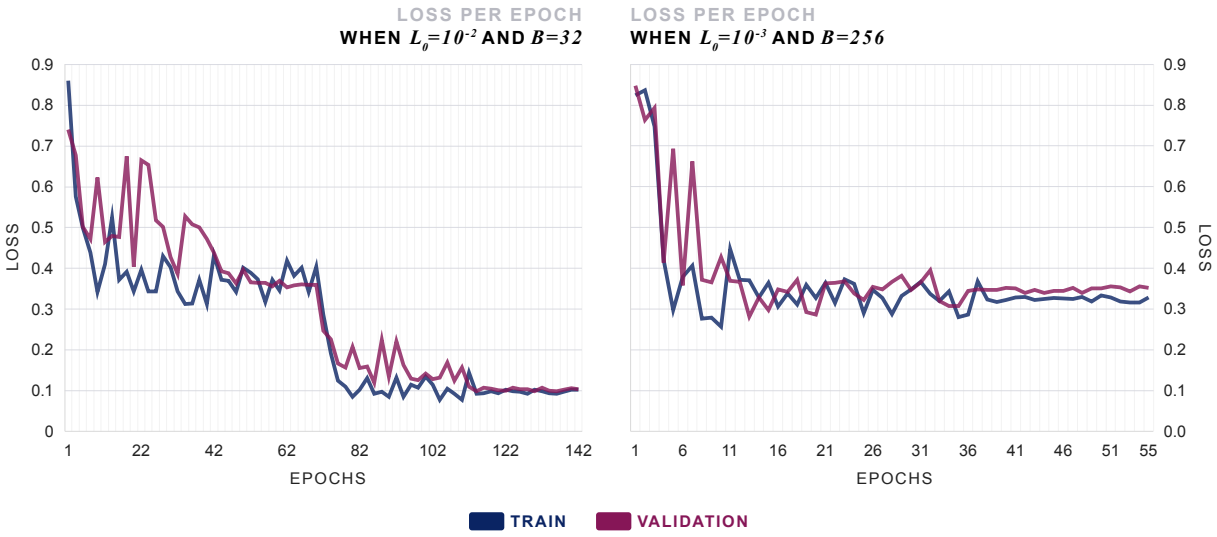


Figure 3.11: Exemplar train-validation loss curves for two different initial learning rate and batch size configurations. Left: $L_0 = 1 \times 10^{-3}$ and $B = 256$. Right: $L_0 = 1 \times 10^{-2}$ and $B = 32$.

Training Conditions

Some techniques like the dropout and batch normalisation layers in each residual block and the augmentation of datasets have already been introduced, that are commonly known in the community to act as regularisation techniques[6] [139]. Besides that, one might ask if the train and validation losses of these models is converging to a sub-optimal local minimum. If that would be the case, the models' parameters would overfit the training time series, and the models would not generalise well in the test time series, or any other unseen data. Hence, it was investigated if the trained models were converging to such local sub-optimal minimum losses, by varying the initial learning rate, L_0, and the batch size, B.

Figure 3.11 shows the train-validation loss curves of model S6 of patient I204. On the left panel, $L_0 = 1 \times 10^{-2}$ and $B = 32$, and, in contrast, on the right panel $L_0 = 1 \times 10^{-3}$ and $B = 256$. The former train conditions resulted in the loss *missing* the local minima in the first epochs, and finding a broader, flatter minimum in the later epochs. Recall that, as aforementioned in Section 3.4, the models were trained with adaptive learning rate. An adaptive learning rate together with a higher initial learning rate allowed the losses to converge to more favourable minima in between epochs 62 and 82. Moreover, a smaller batch size, $B = 32$, implies more variability from batch to batch, further contributing the rate and direction of convergence to be more variable in the first epochs, hence preventing the stabilisation at local minima. Although it cannot be guaranteed that the final validation loss (0.102) is an absolute minimum, it surely surpasses that of epoch 62 (0.361). Conversely, in the right panel, the final validation loss was 0.351, most likely a sub-optimal validation loss. Therefore, a higher $L_0 = 1 \times 10^{-2}$ and a lower $B = 32$ were chosen as implicit regularisation factors, even though such low B increased the training times from 0.12h/epoch to 0.26h/epoch on the specified main machine. Furthermore, notice the lower difference between final train and validation losses on the left panel configuration.

[6] In machine learning, regularization techniques help reduce overfitting and, consequently, generalisation error of a model.

Table 3.3: Final Configuration of Hyperparameters used to train-test the cohort.

R	D	F	F_0	N	N_O	PH	L_0	B
17	6	6	64	35s	50%	30m	10^{-2}	32

3.6 Final Cohort Evaluation

The final architecture and training conditions were fixed for all patients. This configuration is summarised in Table 3.3. Different models were trained for each patient, leaving one seizure out at a time, which constituted the unseen seizure. Recall that the neighborhood 6 hours of each unseen seizure constitute the test dataset in each cross-validation (CV) fold. The average test losses per patient for all seizure folds are presented in Table 3.4.

The lowest test losses were attained in the models of patients I204 and 58QF. For these patients it is safe to conclude that every set of $S-1$ seizures shows a preictal pattern that, when recognized, can predict the other seizure. As depicted in Table 3.6, every model except one of patient I204 was able to predict the unseen seizure, and every model of patient 58QF could predict or detect the unseen seizure. The highest test losses were attained in the models of patients 6QXD and YD2L, which have a performance comparable to that of a random classifier. Table 3.6 shows no seizure could be predicted nor detected for these patients. Table 3.4 also reports on the metrics introduced in Section 2.4. The cohort median sensitivity and specificity were 0.665 and 0.798, respectively, for segment classification. The cohort median positive predictive value (PPV) and F1-score were 0.782 and 0.728, respectively. This performance falls short on DL standards, however F1-scores above 0.8 were attained in models of patients RE38 and 58QF. Appendix A details the sensitivity, specificity, PPV and F1-score of each fold. In at least one fold of all patients except 6QXD, YD2L, S8SG, and RR6Z, the F1-score was above 0.8.

In testing, the classification of each segment was fed to the decision algorithm, which would ultimately decide if an alarm would be triggered. Table 3.5 reports on the decision algorithm performance. The cohort median F1-Score was 0.774, with the highest scores ($>$0.9) being those of patients I204, 2XF9 and Z410. These reflect on their low false alarm rates (FARs): 1.71, 0.57, and virtually 0.00, per day,

Table 3.4: Performance of patient-specific ResNet models. Average for all CV test datasets of each patient is shown. Cohort median on the last column.

Patient	2XF9	I1HP	6QXD	Z410	YD2L	S8SG	I204	IFW2	58QF	RR6Z	RE38	**Median**
Test Loss	0.231	0.305	0.786	0.242	0.740	0.320	0.194	0.371	0.165	0.723	0.221	**0.305**
Sensitivity	0.645	0.709	0.195	0.708	0.249	0.665	0.709	0.535	0.845	0.244	0.776	**0.665**
Specificity	0.886	0.806	0.680	0.868	0.552	0.784	0.858	0.798	0.779	0.672	0.853	**0.798**
PPV	0.848	0.782	0.372	0.829	0.348	0.771	0.844	0.712	0.794	0.414	0.840	**0.782**
F1-Score	0.728	0.736	0.254	0.747	0.289	0.698	0.760	0.600	0.818	0.305	0.802	**0.728**

Table 3.5: Performance of the decision algorithm. Average for all CV test datasets of each patient is shown. ⊗ means no seizure was predicted. Cohort median in the last column.

Patient	2XF9	I1HP	6QXD	Z410	YD2L	S8SG	I204	IFW2	58QF	RR6Z	RE38	**Median**
F1-Score	0.944	0.908	0.512	0.947	0.281	0.733	0.954	0.883	0.757	0.774	0.724	**0.774**
Latency (min)	-4.6	-14.6	⊗	-13.6	⊗	-7.2	-25.8	-9.1	-0.8	-1.5	-17.1	**-9.1**
FAR (/day)	0.57	4.00	12.00	0.00	92.00	5.33	1.71	4.00	17.33	15.43	0.00	**4.00**

respectively. The lowest scores were those of patients 6QXD and YD2L, in which, as expected from the respective models' performance, no seizure was predicted nor detected. Moreover, the decision models of patient YD2L attained the highest FAR – 92/day – which is unconceivable for real-world implementation. The cohort median FAR was 4/day, which, according to clinical expectations it is still too high. Hardly a patient would accept a system that raises a false alarm every 6 hours, for the trust in the system would be undermined. Tables A.6 and A.5 of Appendix A detail, respectively, the F1-score and the number of false alarms raised in each CV fold by the decision algorithm.

Table 3.6 details the alarm latency of the seizures that were indeed predicted or detected, and highlights the ones that were not. Seizures from patient I204 could be detected on average 25.8 minutes before EEG onset. Prediction happened at least 2 minutes before the onset in all seizures that were indeed predicted, except S6 from patient 2XF9, and the ones of patient 58QF, which were mostly detected after the onset. The supervisor of the epilepsy monitoring unit at HSM, Dr. Carla Bentes, says that a seizure prediction algorithm must detect seizures at least 2 minutes before seizure onset, so that proper action can be taken, such as the use of ASMs. That is the average case for models of 77% of patients from which seizures could be predicted. Moreover, Dr. Carla believes that predictions with an anticipation of 5 minutes would be the ideal, and that more than that is really not necessary. That is the average case for models of 67% of patients from which seizures could be predicted. Therefore, for *responders*, this is a method that meets clinical expectations.

Table 3.6: Latency of the decision algorithm for each cross-validated (unseen) seizure. Values in minutes. ⊗ means seizure was not predicted.

Patient	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**Average**
2XF9	⊗	-5.7	-6.3	-2.2	⊗	-1.1	-7.5					-4.6
I1HP	-14.3	-19.2	-10.1	⊗	⊗	⊗	-15.6	-15.7	-12.7			-14.6
6QXD	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗
Z410	-18.1	-10.0	-9.3	-14.7	-12.9	-16.6	⊗	⊗	⊗			-13.6
YD2L	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗		⊗
S8SG	-5.2	⊗	-7.4	-6.8	-8.0	-8.5						-7.2
I204	-28.7	-22.6	-29.3	⊗	-29.3	-21.2	-23.9					-25.8
IFW2	-2.0	-16.5	⊗	⊗	-3.2	-2.5	-14.2	⊗	-10.1	-14.9		-9.1
58QF	2.0	-1.5	1.1	0.1	-4.2	-2.1						-0.8
RR6Z	-1.5	⊗	⊗	⊗	⊗	⊗	⊗					-1.5
RE38	-16.7	-16.1	-18.2	-19.1	⊗	-15.3						-17.1

Table 3.7: Comparing performances of this work (in bold) with the reviewed ECG-based state-of-the-art. Grouped by with-prediction works (top section) and detection-only works (bottom section). Entries in descending order of number of seizures tested. Sensitivity, specificity and PPV are relative to the segment classifications. FAR and latency are relative to the alarms. Average values, except $\tilde{x}$ medians; $\downarrow$ indicates best anticipated alarm; – means not defined by the authors.

	Sen (%)	Spe (%)	PPV(%)	FAR (/day)	Latency	PS	D	P
Threshold CSI [209]	27–43	–	–	0.4–0.7 /d	$\widetilde{19}$ s ($\downarrow$ -22 s)		•	•
Threshold HRV [153]	78.6	–	–	5.04/d	-4.5 m			•
KSC [169]	90.0–100.0	77.5–90.5	–	–	20 s		•	•
ResNet [This Work]	$\widetilde{\textbf{65.8}}$	$\widetilde{\textbf{53.5}}$	$\widetilde{\textbf{28.6}}$	$\widetilde{\textbf{0.72/d}}$	$\widetilde{\textbf{-9.1 m ($\downarrow$ -25 m)}}$	•		•
KSC [160]	57.3–84.7	–	–	–	7.4 s ($\downarrow$ -11 s)		•	•
SVM [159]	100	92	92	–	–			•
SVM [155]	89.1	89.3	–	9.84/d	–	•		•
SVM [163]	83.8	72.8	–	358/d	-18.2 m	•		•
SVM [156]	94.1	–	–	11.76/d	–		•	•
SVM [158]	75.5	75.5	–	–	–			•
SVM + RF [170]	52–74	–	–	24.0 /d	–	•	•	
RF [59]	31.4–39	–	–	1.2–0.4 /d	–		•	
SVM [161]	76.5	–	3.67	26.16 /d	18 s	•	•	
Threshold [152]	86	–	–	172.8 /d	–		•	
SVM [162]	73	–	–	48 /d	21 s	•	•	
MLP [166]	83.3–86.7	90.0–96.1	–	–	–		•	
SVM [157]	81.9	47.28	5.43	–	17.8 s		•	
Threshold CSI [167]	90.5–100	–	–	1.0/d 0.11/n	$\widetilde{30}$ s		•	
Clustering + SVM [165]	42.2–96.0	–	30.7	1.60–2.32/n	15–28 s	•	•	
KSC [168]	93.1–100.0	–	86.9	–	–		•	
SVM [154]	88	–	–	18/d	11.9s		•	
Threshold CSI [150]	87.0	–	–	0.9/d 0.22/n	$\widetilde{35}$ s		•	
SVM [164]	70	50.6	–	–	–		•	

On average, each *responder* patient had 78.4% of the seizures in the monitoring week predicted, by the respective fold model. This is excluding patients 6QXD, YD2L, and RR6Z, which can be considered *non-responders* of this method. Note that, it would be expected that no seizure of patient RR6Z could be predicted, due to the unsatisfactory classification models' test losses (0.723 on average) but in fact, the decision algorithm predicted one (S1), however this is most likely attributed to random chance, since the sensitivity of the S1 model was inferior to 0.5 (Appendix A).

In Table 3.7, the performance of the proposed method is compared with the state-of-the-art methods reviewed of Section 3.1. The segment classification sensitivities, specificities and PPVs of all works is depicted in this table, although, as discussed in [16], it is not clear in some works whether the authors are reporting the segment classification performance or the decision algorithm performance. Table 3.7 is divided in methods with prediction and only with detection, in descending order, from the work with more seizures tested to the work with less seizures tested. As previously discussed, the number of seizures tested is an important factor to validate the method on the remaining population of patients with epilepsy.

Chapter 4

The LTBio Framework

Make things as simple as possible, but not any simpler.

— Albert Einstein

Up until a few years ago, biosignals would be acquired almost exclusively in healthcare units [210]. Today, biosignals are acquired by our watches, phones, and fitness devices [211], making biosignals and biosignal-based algorithms available to the general audience. This has been possible because biosignal research has evolved at a rapid pace, due to more accessible research opportunities [210]. Single-board computers and development boards that acquire biosignals with minimal effort have been commercialised for education and research purposes [212]. Hardware devices like BITalino, ScientISST, Arduino and Raspberry Pi have made biosignal research available to students and engineers more than ever before. However, software for research has not developed at the same pace.

As biosignal research continues to grow, so does the amount of data stored to support it. Our computers, phones and wearables continuously produce biological data everyday, that can be used for research and to get healthcare advice. Research conducted in clinical settings, like the one of *PreEpiSeizures*, continuously collect large volumes of data every week. It can become difficult to know what to do with such large volumes of data, in particular if it is dispersed, heterogeneous, and decoupled from each other. It can become impractical to analyse and extract knowledge from all collected data without automating the process; this is known as the *curse of data*. If all data is not analysed properly, important information might go unnoticed and research might not meet its full potential [210].

The set of operations described in Section 2.1.3 can be automated in a set of singular, yet personalised, tools. Tools that alleviate biosignal researchers – the users – from the programming overhead of automation and allow them to focus solely on the research at hand. Section 4.1 distils why the current tools do not offer the necessary automation and abstraction described. Section 4.2 gathers the requirements for the idea being proposed, and Sections 4.3, 4.4 and 4.5 present the top-level architecture of the

proposed solution – LTBio – and the essential tools to handle biosignals it offers. Moreover, Section 4.6 proposes a format to share processed biosignals between research members and teams.

Additionally, as conveyed in Section 2.1.3, Machine Learning (ML) and Deep Learning (DL) techniques have been widely employed in biosignal event detection, medical decision support, and fitness tracking. Such tools have been deployed at an ever growing pace because they make people's jobs easier and improve consumer's well-being [213]. For research, a wide variety of tools has been offered to expedite learning from images and audio, from more low-level libraries, to high production-level technologies [210]. However, it remains to be offered a tool to expedite learning from biosignals, with all the embedded knowledge about the Domain[1] required to make research easier. A standard tool that prevents users from needing to adapt gold-standard libraries every time they pivot the investigation methods or collect different biosignals. Section 4.7 presents how LTBio meets this need, and the advanced tools for automation it offers. How LTBio is being tested and how it was evaluated with users is presented in Section 4.8. Future plans for LTBio to receive community contributions are discussed in Section 4.9.

4.1 Related Work

This section reviews some open-source biosignal processing software used today by researchers.

BioSPPy — One of the most popular biosignal processing toolboxes was created in 2015, and has been continually growing ever since[215]. BioSPPy is written in Python, it is publicly available under the BSD-3-Clause license[2], and it is currently downloaded approximately 7000 times a month[3]. The library follows a procedural programming paradigm, offering multiple procedures to be used as the user needs them. These procedures are organized in modules for the common biosignal modalities (EEG, ECG, EDA, EMG, RESP, PPG), and modules for *plotting*, *clustering*, and *biometrics*. In this paradigm, data is loosely kept in common language structures and procedures are highly coupled[4], creating opportunity for errors when manipulating data. Moreover, modules show low cohesion[5], as their elements do not take part in the same tasks. For example, to open an ECG from a file and plot it to check its content, one would have to use modules *storage*, *ecg*, and *plotting*, demonstrating a lack of functional, sequential and procedural cohesion [214]. Regarding quality assurance, the library shows no automatic test. Nevertheless, in terms of usability, once users become familiar with it, BioSPPy allows for rapid prototyping and has been employed as a standard tool in many projects world-wide.

NeuroKit — Another community-driven biosignal toolbox is NeuroKit 2 [216], with more than 6000 downloads a month[6]. It was also written in Python and it is publicly available under the MIT license[7].

[1] In Software Engineering, the Domain is the sphere of knowledge of a subject around which a program's logic revolves [214].
[2] BioSPPy is available at `github.com/PIA-Group/BioSPPy`
[3] Accessed in October 2022 at `pepy.tech/project/BioSPPy`
[4] In Software Engineering, coupling measures of how interdependent are system components or modules [214].
[5] In Software Engineering, cohesion measures of how well elements of a system component or module function together [214].
[6] Accessed in October 2022 at `pepy.tech/project/neurokit2`
[7] NeuroKit 2 is available at `github.com/neuropsychology/NeuroKit`

Although its name can be misleading, this library offers elements to process a variety of biosignal modalities (ECG, PPG, EDA, RESP, EMG). Like BioSPPy, it follows a procedural programming paradigm, organising procedures in modules for each biosignal modality, event management, segmentation in epochs, statistical analysis, Markov chains, among other more specific tools. Data is also stored in general-purpose structures (Pandas' Dataframes) and is loosely coupled to procedures. The NeuroKit 2 architecture shows the same low cohesion issues BioSPPy does. Besides the processing procedures, it also offers public datasets and synthetic biosignal generators, helping projects with shortage of data. Regarding quality assurance, the library shows automatic tests with a 54% code coverage.

EEG Toolboxes — MNE and EEGLAB are the most popular libraries for EEG-specific analysis. The first runs on Python and follows a procedural programming paradigm, whereas the second runs on MATLAB and follows a functional programming paradigm and offers a GUI.

The general-purpose libraries described above (BioSPPy and NeuroKit) are the gold-standard biosignal analysis tools freely available. They provide state-of-the-art biosignal processing functions, but they do not abstract automation. Despite of being implemented in a high-level programming language, these libraries provide low-level tools with respect to the Domain of biosignals. For instance, although the internal management of data structures and visualisation algorithms are abstracted when drawing plots, the user still needs to prepare the data structures for x-axis and y-axis, instruct how to draw the plot curves, scale them, and deal with potential interruptions in the time series. This can be time-consuming for long-term biosignals and large cohorts. The user should be able to just instruct for a plot from timepoint *a* to timepoint *b*. LTBio lies on the premise that this and other workflows can be further abstracted without loosing fine control over these tools. Therefore, the primary goal of LTBio is allowing researchers to focus more on the research task at hands, rather than on the coding technicalities. The following section gathers the requirements for such a system.

4.2 System Requirements

The process of eliciting the requirements for a software is perhaps the most challenging and error-prone in the software development life cycle. So, effort was put into understanding the Domain of Section 2, and eliciting the requirements that conform with the users' needs. Over the course of six months, the system requirements were gathered with the help of 6 biosignal researchers in early-career, that already use the software introduced in the previous section, mainly through the following methods:

- Individual and group interviews with users that use similar software in their everyday work.
- Brainstorming sessions with the users to propose tools that can help them.
- Direct observation of how users currently use similar software in their everyday work.
- Reading published work on biosignal analysis to gather which features are essential.
- Analysing similar software to identify and classify features to include, to improve and to avoid.

The following subsections characterise the typical user and the requirements that were gathered.

4.2.1 The Users, their Environment, and their Tasks

The typical user of this framework is a researcher that conducts biosignal analysis in their everyday work. This is often someone older than 23 years, with an engineering degree, highly familiarised with computer programming, and competent in biosignal acquisition, processing, and subsequent analysis. Engineering students are also potential users, although these should be considered as novice users, since they have not yet completed the necessary education to be autonomous researchers.

As conveyed in Section 2.1.3, some users acquire biosignals at hospitals or laboratories with third-party instruments, while others acquire biosignals from volunteers outside clinical context with their own devices. Users usually conduct the posterior biosignal analysis sat at a desk, using a UNIX-based or Windows computer with Internet connection.

Common tasks performed on these biosignals are quality assessment, filtering, extraction of features, and traditional or machine learning analyses. This is currently done using software libraries like Pandas, BioSPPy, NeuroKit, SciKit Learn, PyTorch and Tensorflow. To conduct these analyses, the programming language most used is Python. This is not an individual preference of each user, but it is rather the language the world-wide biosignal processing community employs, along with MATLAB. Besides, most interviewed users revealed that Python is the programming language they are most fluent at, when not the only. The user has usually learnt these technologies, through specific education and work experience, by reading the documentation, and by networking with work colleagues.

Besides computer programs users write, they also make use of annotations and reports to get subject metadata, instrument specifications, event annotations, among other information, important to support and direct their research. These pieces of information sometimes come from partner institutions, and other times are created by themselves. Examples of these are hospital discharge notes, medical records and reports, consent forms, and hand-written annotations. Often users organise all data in Excel sheets.

4.2.2 Functional Requirements

With the methods previously described, it was gathered that LTBio had to offer the following features:

Data and Metadata

1. Encapsulate multiple channels of a biosignal and all their relevant properties.
2. Associate title names, channel names, sampling frequency, and anatomical region with biosignals.
3. Associate patient data and clinical history with biosignals.
4. Trace back which instrument acquired the biosignals and where did they came from.
5. Read biosignals from multiple devices and collaborating institutions.
6. Read metadata and annotations from multiple devices and collaborating institutions.
7. Read biosignals from public databases.
8. Associate and dissociate multiple events and annotations to biosignals.
9. Associate units to time series and allow for possible conversions; e.g., mV, μS, g, etc.
10. Retrieve when biosignals were acquired, when they were interrupted, and their useful duration.
11. Serialise and deserialise biosignals and their processing so far.

Operations

12. Perform arithmetic operations to multi-channel biosignals.
13. Format time series; e.g., split, trim, invert, normalise, and segment them.
14. Resample multi-channel biosignals.
15. Filter multi-channel biosignals.
16. Apply any digital signal processing operation to all channels of a biosignal.
17. Automate and repeat any signal processing operation to a collection of biosignals.
18. Undo any operation applied to biosignals.
19. Extract any (novel or commonly known) feature from multi-channel biosignals.

Visualisation

20. Plot multi-channel biosignals, with date, time, events and with patient identification, if any.
21. Plot short random excerpts of multi-channel biosignals, for quick inspections.
22. Clearly visualise ML datasets.
23. Visually summarise the results of training ML models.
24. Open project-specific summaries of acquisition sessions on a Web browser, for quick inspections.

Automation

25. Automate pre-processing of multi-channel biosignals.
26. Automate feature selection processes.
27. Automate the creation and preparation of n-dimensional ML datasets.
28. Automate training multiple ML models in multiple conditions.
29. Automate the evaluation of ML models, with state-of-the-art and project-specific metrics.
30. Automate medical decision support, or other automatic decisions.
31. Automate specific biosignal processing steps with reusable and reproducible blocks.

4.2.3 Non-Functional Requirements

It was gathered that LTBio had to offer the following non-functional requirements:

1. **Usability:** Users should be able to use the framework capabilities through the command line, as well as in written programs of their own, in a programming language that is familiar to them.
2. **Familiar Language:** Structures, procedures, and other constructs should be named after the Domain concepts, so that users immediately identify what a construct is or does using solely their background knowledge.
3. **Prevent User-Error:** Semantic errors, both programming-related and Domain-related, should be minimised by offering usable interfaces.
4. **Expandability:** Most processing functionalities the framework offers should be easily extendable by the users (e.g. to add features specific to a project's Domain), in order to accomplish their research goals.
5. **Reusability:** Most processing components the framework offers should be reusable repeatedly with minimal effort.

6. **Reproducibility:** Users should be able to run the same biosignal processing steps on their datasets, using analysis scripts shared by other teams.
7. **Portability:** The framework must provide easy ways of sharing biosignals and their processing history among users, in a small and recognisable file format by everyone that uses the framework. Biosignal files and their processing history need to be easily transferred between computers, data storage, and locations (e.g. research institutes, hospitals, etc.).
8. **Flexibility:** The framework should not feel limiting; users should be able to easy step outside of it and use other Python libraries in separate or in collaboration.
9. **Time Efficiency:** Long-term biosignals should be quickly loaded to computer memory or segments of them should be quickly accessed, in order to speed up the research process. Unfortunately, there is no way of quantifying how fast this should be, since it is highly dependent on the type of files and their size, how many days the acquisition lasted, how many interruptions occurred, among other heterogeneities. Nonetheless, users with their current methods may take up to hours to load long-term biosignals to memory, and they have to partition them after the first loading. Hence, reducing the current loading time, however much, is a priority.
10. **Space Efficiency:** Biosignal files should occupy a small space in storage. Currently, external hard drives are used to store large long-term biosignals, and cohort analyses on them are performed with the files on those external memories, which hinders the processing time. Unfortunately, for the same reasons, there is also no way of quantifying how small this should be. Nonetheless, reducing the current file sizes, however much, is a priority.

4.3 Top-Level Organisation of the Proposed Framework

Having gathered the requirements, the LTBio framework was developed with three main components: the `ltbio` Python library, the `.biosignal` file format, and the Biosignal Summary for the browser. Figure 4.1 illustrates how these three components interact, as well as the main actors. The `ltbio` Python library is the main aggregating component of the framework, and it is presented in the following sections, until the end of this chapter. The Biosignal Summary for the browser is introduced in Section 4.4. The `.biosignal` file format is covered in Section 4.6.

Contrarily to BioSPPy and NeuroKit, the LTBio Python library was designed following an object-oriented paradigm (OOP), where biosignals, filters, models, and all other constructs are objects. Although Python does not quite harness all OOP features, it serves the purpose, namely in allowing structural Domain-specific inheritance. For example, ECG, PPG and ACC signals are all biosignals that are manipulated in time in the same way and are filtered using similar methods, hence, although with some specific properties, they all inherit a set of common properties every biosignal does. Abstraction in OOP is also a great property for this Domain. For example, a plot of a multi-channel biosignal is often drawn in the same way, in terms of axes and drawing the amplitude points over time, hence a biosignal object should be auto-contained enough *to know how to plot itself*. The same applies to *how to trim*

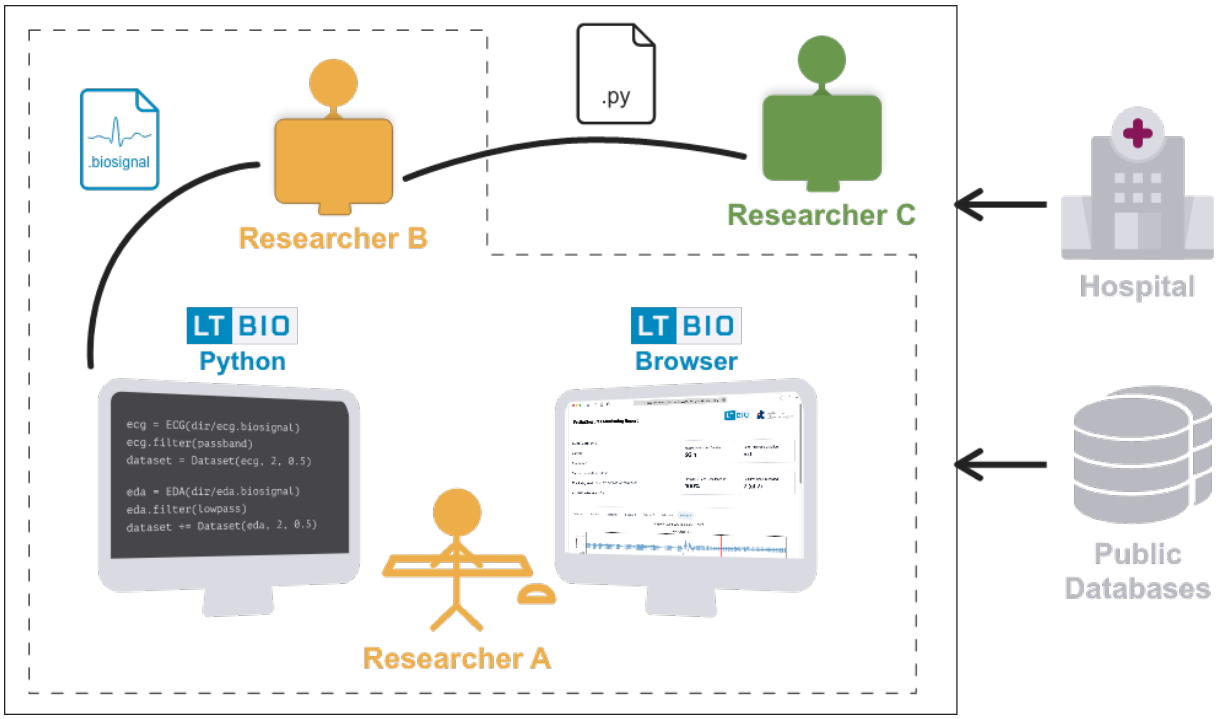


Figure 4.1: Use case of the three LTBio components. Solid-line box separates the system actors from the exterior. Dashed-line area defines a research team. Researchers A and B are exchanging biosignal files, as well as Python scripts with Researcher C from another team (represented by solid bi-directional lines). Researcher A can write analyses scripts while, on a browser, consulting the cohort summaries.

itself, e.g., around the date and time a seizure event occurred. This level of abstraction is essential to keep users focused on the research at hands, rather than remembering how to program these tasks. Despite the disadvantages of BioSPPy and NeuroKit, users are familiar with the routines these libraries offer. Therefore, rather than re-implementing them, LTBio builds on top of BioSPPy. That is, LTBio uses BioSPPy as its *functional module* regarding biosignal processing, e.g. apply filters, extract common features, etc. BioSPPy is a benchmarked and peer-reviewed library in which users trust, so it seems prudent to delegate the logic behind biosignal processing to it.

The LTBio Python library is divided into 7 packages: (i) *biosignals*, (ii) *clinical*, (iii) *processing*, (iv) *features*, (v) *ml*, (vi) *decision*, and (vii) *pipeline*. This division provides structural organisation in the usual workflow of biosignal analysis, introduced in Chapter 2. That is, users will most likely use packages (i) to (vi) in this order. As introduced in Subsection 2.1.3, since the usual workflow of biosignal analysis resembles very much a pipeline, users have package (vii) dedicated to automating this workflow with building blocks in the order that is most suitable to their project. The main idea is users getting the components they need from each package, and for that a workflow-driven organisation requires less recalling of where each component is. The following subsections introduce the main features of these packages.

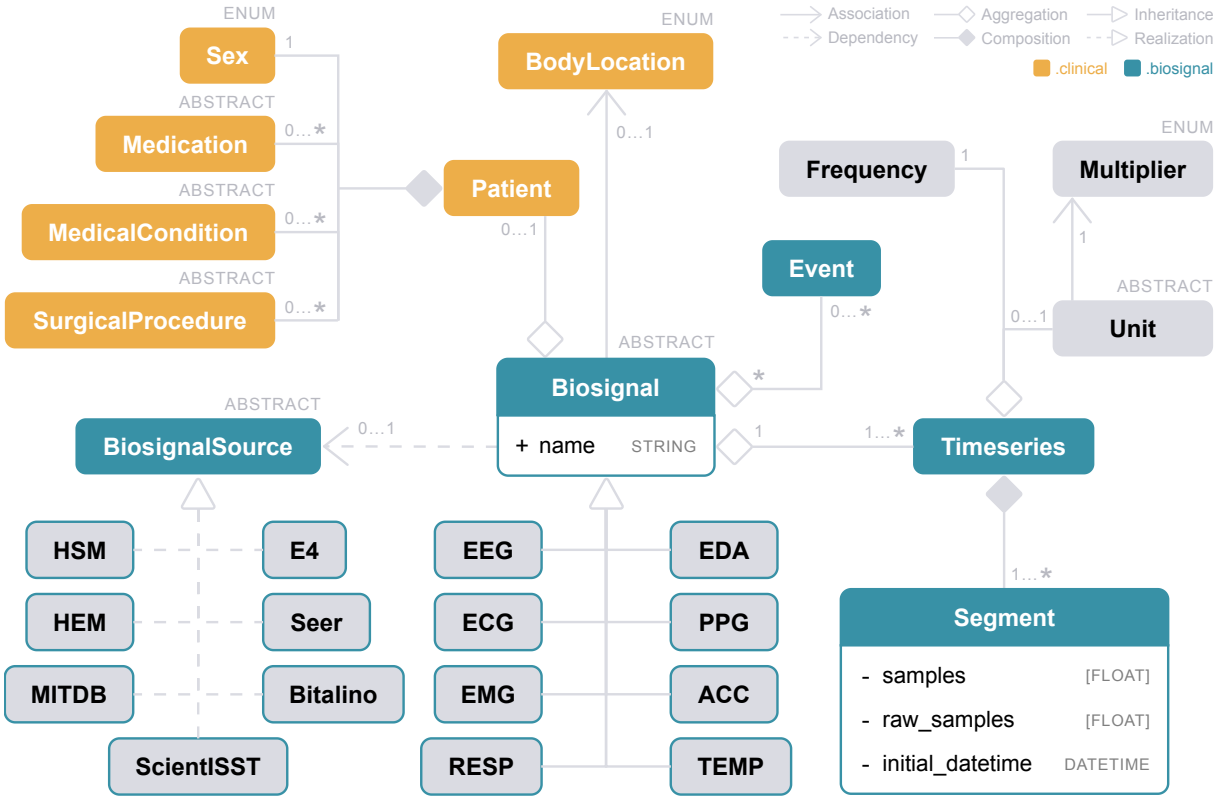


Figure 4.2: Top-level UML diagram of the *.biosignals* (blue) and *.clinical* (yellow) packages of LTBio.

4.4 Gathering, Inspecting and Annotating Biosignals

The central class of the framework is the ***Biosignal*** abstract class, present in the *.biosignals* package. Every time a user reads a biosignal data to memory they have to instantiate a *Biosignal* subclass. Multiple subclasses were made available for the most common biosignal modalities (Figure 4.2). For example, to instantiate an ECG from a file, one could use the following instruction:

```
ecg = ECG('pathToFile', HSM, name='My First Biosignal')
```

In the above example, `HSM` is the ***BiosignalSource*** representing "Hospital de Santa Maria". This class knows how to read biosignals from european data format (EDF) files collected at HSM. In fact, the biosignals used in Chapter 3 were read with this simple instruction. As depicted in Figure 4.2, there can be as many *BiosignalSource* subclasses as the user needs. The ones most used at IT are already provided, like *ScientISST* and *BITalino* devices, public databases like *MITDB* and *Seer*, and the *HEM* hospital. A *BiosignalSource* is an entity with knowledge about where (devices, hospitals, databases, etc.) and how biosignals are acquired. It has static procedures to ease the reading of biosignals from files of that source, and the respective patient metadata, clinical records, and event annotations. These have their own classes as well, as shall be described ahead. Other sources can be easily implemented by deriving *BiosignalSource*. This scalable property is of vital importance for the successful use of this framework, since biosignal researchers get data from a large variety of sources that increases by the day. Hence, the possibility of working with data from new sources only by creating their own *BiosignalSource* bounds to a single action the personalisation of the framework to their needs.

4.4.1 *Biosignal* as the Atomic Unit

Biosignal objects hold the acquired time series and all the associated metadata in properties. In this way, biosignals can be reused, passed around, and serialised, without loosing the data necessary for their interpretation. Moreover, holding theses properties allow *Biosignals* to modify themselves, without users having to remember them. A *Biosignal* object is a non-empty set of channels measuring one biological or physiological variable. Each channel is represented by a *Timeseries* object. Optionally, it may also have an associated *Patient*, an associated *BodyLocation*, an associated *BiosignalSource*, and a name. Figure 4.2 shows these relations. Associated *Patient* and *BodyLocation* objects allow to access metadata from whom the biosignal belongs and from where it was acquired, respectively. A *BiosignalSource* allows to know from which device or institution the biosignal came from, as aforementioned. The name of a *Biosignal* can be get and set, but other properties cannot, unless by the appropriate methods, in order to maintain a stable internal state and keep its abstraction level.

A ***Timeseries*** object is a discreet sequence of data points that occur in successive order over some period of time. In a *Biosignal*, the data points of one *Timeseries* are the measurement of a biological or physiological variable, in some unit, taken from a sensor. These data points are often called samples and are acquired at a fixed sampling frequency. To each timepoint of a *Timeseries*' domain corresponds one and only one sample, like a mathematical function. However, a *Timeseries* might be contiguous if a sample was acquired at every sampling timepoint, or discontiguous if there were interruptions. Each interval of contiguous samples is called a *Segment*, but those are managed internally by *Timeseries*. Hence, the internal structure of a *Timeseries* contains a sequence of *Segment* objects, a sampling frequency (*Frequency*), and optionally it may have associated units (*Unit*) and a name (*string*). Figure 4.2 shows these relations.

A ***Segment*** is an interrupted sequence of samples. It is an internal and private class of *Timeseries*, used for internal management of samples and interruptions. Internally, it holds one array of samples, an initial date and time, and some control variables for processing features, namely a boolean flag stating if the samples have been filtered and, if so, a reference to the raw samples.

A ***Unit*** represents a unit of measure of some variable. The common units of biological variables are Volt (*Volt*), Siemens (*Siemens*), G (*G*), Celsius degree (*DegreeCelsius*), decibels (*Decibels*), or beats per minute (*BeatsPerMinute*), for which concrete classes were implemented. This increases the level of abstraction when converting *Timeseries* from one unit to another.

A *Biosignal* can also have a set of associated events (***Event*** objects). As introduced before, events are common in biosignal analysis to know when did a certain occurrence of interest happened in time, so that such segment can be processed and analysed accordingly. An *Event* is an occurrence in time or within a period of time that has relevance to a specific problem. It has a name, and it must have at least an onset, an offset, or both. The onset and offset can be reset after instantiation, but the name cannot for identification stability reasons.

4.4.2 Standardising Clinical Concepts

Depending on the project, biosignal researchers can work or not with clinical concepts. So, the *.clinical* package is dedicated to these concepts. The user that needs can import them into their project, whereas the user that does not can leave them out and entirely work with biosignals with no clinical information. This package includes the class *Patient*, the enumeration *BodyLocation*, medical conditions, surgical procedures, and medications.

A ***BodyLocation*** is a region in the human body. These are useful, for example, to name *Biosignal* channels after the location where each electrode was placed, or to describe where did a surgical procedure take place. These can range from common names like chest, wrist, or scalp, to more technical standardised locations like where ECG electrodes are placed on the chest (V1, V2, V3, ...), or where EEG electrodes are placed on the scalp (F1, FP1, Cz, ...). A total of 47 *BodyLocation*s and their names were already defined, and more can be added in the future. *BodyLocation*s should be specifically defined to restrict the values the user can give, thus reducing human error and facilitating comparison between their values.

A ***Patient*** is a subject with a name, an age, a biological sex, a collection of medical conditions, of surgical procedures, and of medications, and a set of textual notes. All these properties are optional. Additionally, a *Patient* has an alphanumeric code, which is a mandatory property. This code is important in situations where a *Patient* needs to be unequivocally identified, or in a cohort where each *Patient* must have a unique code. When instantiating a *Biosignal*, if a *Patient* is associated, these properties can be accessed and be useful throughout the biosignal processing and analysis. Multiple *Biosignal* objects can be associated to one *Patient* object, as illustrated in Figure 4.2, facilitating the sorting of biosignals by patient.

A ***MedicalCondition*** is any condition given by a medical diagnosis. Its internal structure contains the years since diagnosis, and it should be extended in each subclass accordingly to which information is useful to maintain organised. It is an abstract class so that multiple medical conditions can be created given each project's needs. For now, the framework offers two examples: *Epilepsy* and *COVID19*. Inclusion polymorphism can be used to keep information compartmentalised regarding types and subtypes of conditions.

A ***Medication*** is any prescribed therapeutic. It has a name (*string*), a dose (*float*), a unit (*Unit*), and a frequency (*string*). *Medication* objects can be associated to *Patient* at instantiation, so it is always respective of a patient. A ***SurgicalProcedure*** describes a surgical procedure a patient has underwent and its outcome. It has a name, a date, a time and an outcome.

4.4.3 Case Study: Managing Epilepsy Data

The work conducted in Chapter 3 was made easier by using LTBio. Here it is exemplified how the *.biosignals* and *.clinical* packages were used to manage the incoming data from HSM and HEM. As

clinical reports and discharge notes were becoming available at IT, they had to be human interpreted to extract the information needed. Let us say arbitrary patient 4H9A experienced three seizures during the monitoring week. These would be translated into code as:

```
from datetime import *
from ltbio.clinical import BodyLocation, Semiology
from ltbio.clinical.conditions.Epilepsy import *

seizures = (
    Seizure(datetime(day=19, month=2, year=2019, hour=23, minute=11, second=2), duration=
    timedelta(seconds=65), awake=True, onset_type=SeizureOnset.F, awareness=True, semiologies=(
    Semiology.AURA, Semiology.AUTOMATISMS), onset_location=BodyLocation.FT_R, description="
    Preictal vegetative aura."),

    Seizure(datetime(day=20, month=2, year=2019, hour=4, minute=55, second=36), duration=
    timedelta(seconds=61), awake=False, onset_type=SeizureOnset.F, awareness=None, semiologies=(
    Semiology.TC, ), onset_location=BodyLocation.FT_R, description="Rhythmic movements followed
    by dystonic posture, in elevation and extension."),

    Seizure(datetime(day=21, month=2, year=2019, hour=9, minute=48, second=4), duration=
    timedelta(seconds=72), awake=True, onset_type=SeizureOnset.F, awareness=None, semiologies=(
    Semiology.MOTOR, Semiology.HK, ), onset_location=BodyLocation.FT_R, description="Layed down
    and beating in the chest in extension."),
)
epilepsy = Epilepsy(years_since_diagnosis=12, seizures=seizures)
```

The reports also contained the anti-seizure medication (ASM) the patient took during the monitoring week. This would be translated as:

```
from ltbio.clinical.medications import *

medications = (
    TPM(200, Grams(Multiplier.m), 'Everyday at 7 am and 7 pm.'),
    ESL(400, Grams(Multiplier.m), 'Every day at 7 am. Reduce on day 2.'),
    Clobazam(10, Grams(Multiplier.m), 'Everyday at 10 am.'),
)
```

Finally, the Packet Trace (TRC) files retrieved from the vEEG system were read using the *HEM* source. Since research interest was only on the ECG channels, these were read to memory using the *ECG* class. When receiving a file for the samples to be stored in a particular *Biosignal* modality, *BiosignalSource* classes filter the channels known to be of that modality. This is the type of knowledge mentioned in Subsection 4.4 that *BiosignalSource* classes should have. Channel names, sampling frequency, and units are also read automatically by the *HEM* source. The corresponding piece of code is as follows:

```
from ltbio.clinical import Patient, Sex
from ltbio.biosignals.modalities import ECG
from ltbio.biosignals.sources import HEM

patient = Patient('4H9A', 'Joao Saraiva', 32, Sex.M, conditions=epilepsy, medications=
    medications)
ecg = ECG('pathToFile.trc', HEM, patient, name='4H9A Hospital vEEG')
```

Line 6 of the previous snippet, reads all samples of each ECG channel and stores them as discontiguous *Timeseries*, for they have interruptions. Each channel is given by a *Timeseries* and each contiguous segment of a channel is given by a *Segment*. Usually, users do not interact with these two classes, but rather only with the *Biosignal* object, `ecg`. Users can inspect this biosignal by printing it:

```
print(ecg)
```

```
Name: 4H9A Hospital vEEG
Type: ECG (mV)
Location: Chest
Number of Channels: 2
Channels: Chest Lead I, Chest Lead II
Sampling Frequency: 254 Hz
Useful Duration: 3 days, 4 hours and 27 minutes (76.45h)
Source: Hospital Egas Moniz
Events associated to Medical Conditions:
- seizure1:
    Focal Aware seizure (FAS)
    Semiologies: Pre-ical Aura, Automatisms
    Onset Location: Right Fronto-Temporal lobe
    Onset: 2019-02-19 23:11:02
    Duration: 0:01:05
    State: Awake/Vigilant
- seizure2:
    Focal with Unknown Awareness seizure (FUAS)
    Semiologies: Pre-ical Aura, Automatisms
    Onset Location: Right Fronto-Temporal lobe
    Onset: 2019-02-20 04:55:36
    Duration: 0:01:01
    State: Asleep
- seizure3:
    Focal with Unknown Awareness seizure (FUAS)
    Semiologies: Pre-ical Aura, Automatisms
    Onset Location: Right Fronto-Temporal lobe
    Onset: 2019-02-21 09:48:04
    Duration: 0:01:12
    State: Awake/Vigilant
```

Notice how the channel name, units, and sampling frequency were read from the packet trace (TRC) file without user intervention. Also notice that the units and sampling frequency are properties of each *Timeseries*, but when these properties agree in all channels – which is the most common – they are uniquely presented. Moreover, the useful duration, the duration for which samples were acquired simultaneously for all channels without interruptions, is also presented. Each channel can experience its own interruptions, and the *Biosignal* instance computes the intersection of the domain of all channels. The

domain of a channel are the intervals of time when samples points were acquired, by analogy of what is the domain of a function. Users can easily find a *Biosignal* domain by printing it:

```
ecg.domain
```

```
[2022-02-19 08:09:24, 2022-02-20 19:08:12[ U [2022-02-20 19:10:14, 2022-02-21 23:54:21[
```

Furthermore, notice that seizures associated with the epilepsy condition that, in turn, was associated to the patient, are presented on the biosignal textual representation, although these *Event*s were not directly associated to the *Biosignal* object via the `associate` method. That would be, for instance, when annotating the patient brush their teeth:

```
e = Event('brushing', datetime(day=20, month=2, year=2022, hour=7, minute=40))
ecg.associate(e)
print(ecg)
```

```
Name: 4H9A Hospital vEEG
[...]
Source: Hospital Egas Moniz
Events:
- brushing
    Onset 2022-02-20 07:40:21
Events associated to Medical Conditions:
- seizure1:
[...]
```

When annotated, Events can be very useful to speed up research. Let's say we want to inspect the ECG right when the second seizure toke place. One would have to do:

```
ecg['seizure2'].plot()
```

The user instructs to see the ECG at the second seizure and no other code was needed. LTBio performed three operations in the background: (i) `'seizure2'` was replaced by the slice [2022-02-20 04:55:36, 2022-02-20 04:56:37], which is the interval between onset and offset; (ii) the `ecg` object was trimmed, that is, the domain of all channels was shortened to that interval – much like a list is sliced – and a new *Biosignal* object is returned; (iii) the `plot` method was called on this *Biosignal* object, which plots all channels. Method `plot` is part of *Biosignal*, not of *ECG*, so it can be called on any biosignal modality and the same behaviour is expected.

If interested in seeing the preictal period, one would execute:

```
ecg[40:'seizure2'].plot()
```

which plots the seizure domain and 40 seconds before. In total, that are 23 possible combinations of indexing *Biosignal* with events, dates and times, which can be found in the full documentation. As will be shown ahead, having the events well annotated in *Biosignal* can be very useful while investigating patterns.

If the user is simply interested in having a quick look, they can use the `preview` method:

```
ecg.preview.plot()
```

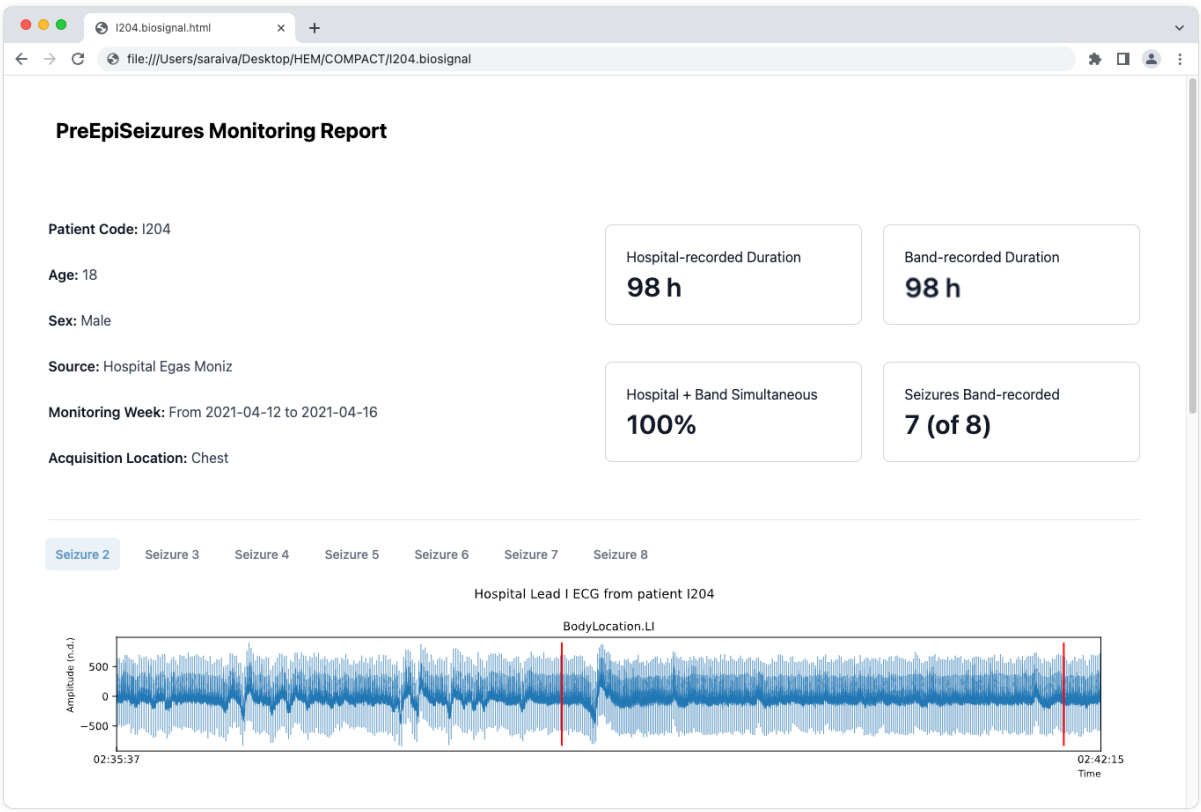


Figure 4.3: *Biosignal* summary on the browser of *PreEpiSeizures* patient I204.

The `preview` method looks exactly in the middle of the domain for 10 seconds simultaneously acquired by all channels, and trims the *Biosignal* on that interval, which is then plotted.

Furthermore, a summary of the *Biosignal* can be opened on a Web browser for specific research projects. For the case of the *PreEpiSeizures*, a class was derived from *BiosignalSummaryReport*, and other users can implement their own as well using Datapane package[8]. Summary HTML files are much smaller and eliminate the need to open biosignals on Python when the important information can be quickly consulted right on the browser. Figure 4.3 shows one of these *PreEpiSeizures* reports on the browser. In this summary, the key patient information is shown, statistics regarding the total recording duration, interruptions and seizures captured, as well as the list of seizures the patient experienced during the monitoring week. When clicking on each seizure tab, a plot of the preictal and ictal periods is shown for rapid inspection. Summary reports of the full *PreEpiSeizures* dataset were archived at IT.

4.5 Processing and Operating on Biosignals

As previously described in Chapter 2, digital signal processing is a central part of the biosignal analysis workflow, whether to attenuate noise, to detect outliers, or to reformat the signal as needed for further processing stages. The reviewed related software is mostly dedicated to this stage of the workflow, providing procedures to execute these kind of tasks. Hence, as aforementioned, the *.processing*

[8]Visit `datapane.com`. Accessed in October 2022.

package encapsulates many of these third-party procedures, providing a silent interface between them and *Biosignal* objects. The processing package is divided in *filters*, *formaters*, and *noises* modules.

4.5.1 Filters

Abstract class ***Filter*** represents a digital filter design that can be used to filter many *Biosignal* objects. Two concrete subclasses are the *FrequencyDomainFilter* and the *TimeDomainFilter*. *FrequencyDomainFilter* allows to design 6 types of FIR and IIR filters in 4 types of bands. These are commonly known and used in the signal processing community. For example, here is a 40 Hz lowpass filter design:

```
from ltbio.processing.filters.FrequencyDomainFilter import *
my_passband = FrequencyDomainFilter(FIR, LOWPASS, cutoff=40, order=200)
```

A *TimeDomainFilter* allows to design convolutional filters with 6 different operations. Here is an example of a 5-second window median filter design:

```
from ltbio.processing.filters.TimeDomainFilter import *
my_median = TimeDomainFilter(MEDIAN, timedelta(seconds=5))
```

All the uppercase options in the pieces of code above are offered in Python enumerations, to minimise user-error. These can be extended in the future. These two created instances are now designs that can be reused to filter many *Biosignal*s. To apply one filter to the `ecg` instance, one would execute:

```
ecg.filter(my_passband)
```

A Visitor design pattern [214] was implemented to apply `my_passband` to all channels of `ecg`. In fact, the pattern was implemented to be general for filtering and all other processing operations. Essentially, a *Filter* object is passed through each *Timeseries* and through each of their *Segment*s, these last ones calling *Filter.visit*, requesting the filtered samples to *Filter*, and saving them in-place. That is, *Filter* objects act as visitors, their method *visit* acts as the visitor method, and *Segment* objects act as the visited. The functional logic behind filtering is delegated to BioSPPy. LTBio only acts as an interface and silently automates the process for all channels.

4.5.2 Formaters and Extensions

Resampling biosignals is a common action taken when pre-processing them. A *Timeseries* object can *resample itself*, i.e., all its *Segment*s, because it holds its sampling frequency (Figure 4.2). The public method `resample` of *Biosignal* automates this for all channels. For example, to resample our `ecg` to 100 Hz, this simple instruction would be used:

```
ecg.resample(100)
```

To make data uniform, biosignals usually are reformatted in several ways. Just like *Filter*s are individual agents that modify *Biosignal*s, agents that format *Biosignal*s can also be designed and reused. Users can normalise or standardise *Biosignal*s using an agent like:

```
from ltbio.processing.formaters import Normalizer
normalizer = Normalizer(method='min-max')
```

Moreover, *Biosignal*s can be segmented by an agent like:

```
from ltbio.processing.formaters import Segmenter
segmenter = Segmenter(timedelta(seconds=10), overlap=timedelta(seconds=2))
```

Let us say we have a collection of biosignals: `ecg`, `eda`, `acc`. One could easily format these in the same way with the piece of code below:

```
for biosignal in (ecg, eda, acc):
    for formater in (normalizer, segmenter):
        biosignal.apply(formater)
```

The first formater normalises the samples of each *Segment*, whereas the second partitions a *Time-series* into fixed-size segments. Formating is performed on all channels of these biosignals. With method `apply`, formaters visit *Biosignal* objects in a similar way filters do.

The segmentation shown above is blind to the biosignals' waveform. However, quasi-periodic signals are often segmented according to their period, e.g. ECG signals are commonly segmented by heartbeats. One of the advantages of each biosignal modality having its own class is that modality-specific functionalities, e.g. heartbeat segmentation, can be available for that specific modality. LTBio calls modality extensions to this functionalities. To segment an ECG by heartbeats, one would execute:

```
heartbeats = ecg.heartbeats(method='hamilton')
```

To obtain the RRI time series from `ecg`, the following instruction can be used:

```
nni = ecg.nni(method='christov')
```

To just get when the R peaks occurred, one can use:

```
rpeaks = ecg.r_timepoints(method='ssf')
```

The `peaks()` extension is also available for `PPG` objects, in order to extract HRV features as well. Moreover, these extensions can be plotted as any *Biosignal* would:

```
nni.preview.plot()
```

The logic to compute these modality-specific extensions is also delegated to BioSPPy.

4.5.3 Adding Noise

If filters suppress noise, then there must be a way to add synthetic noise. Synthetic noise is part of the evaluation of many biosignal denoising algorithms. Since noise generation processes are quite standard, this functionality was included in LTBio. One could create a white noise process, $\mathcal{N}(0, 1)$, with the following piece of code:

```
from ltbio.processing.noises import GaussianNoise
white = GaussianNoise(0, 1)
```

Once again, this design can be reused whenever adequate. For instance, to instantiate a new `ECG` object consisting in gaussian noise added to the `ecg` we had before, one would use an alternative constructor:

```
noisy_ecg = ECG.withAdditiveNoise(ecg, white)
```

The constructor adds the white noise to every `ecg` channel, with the respective sampling frequency of each. This operation can be shortened with the + binary operator:

```
noisy_ecg = ecg + noise
```

or

```
ecg += noise
```

to have the operation executed in-place. Previously, users had to prepare arrays with noise in order to accomplish this operation. With this abstraction, researchers can focus more on the task at hands.

4.5.4 Arithmetic Operations

Picking up the previous example, *Biosignal* supports arithmetic operations. Let us say `emg` is an `EMG` signal loaded to memory. A common use case is to add electromyogram (EMG) noise to ECG signals to simulate myogenic artifacts. This could be easily achieved by:

```
noisy_ecg = ecg + emg
```

The EMG time series is added channel-wise to the ECG. LTBio ensures the user does not mistakenly add incorrect channels, by enforcing the same channel names on both *Biosignal*s. The user can make that true using `Biosignal.set_channel_name` adequately. Moreover, channels of different *Biosignal*s can be joined into one *Biosignal* with the `&` operator. The `&` operator should be read as "the first *Biosignal*'s channels and the second's". Given two `ECG` objects, the first with one channel at 254 Hz defined in [13h00, 14h00[and the second with two channels both at 100 Hz defined in [13h00, 15h00[, the following output would be produced when joining both:

```
print(ecg1)
print(ecg2)
print(ecg1 & ecg2)
```

```
Name: First
Type: ECG (mV)
Location: Chest
Number of Channels: 1
Channels: Chest Lead I
Sampling Frequency: 254 Hz
Useful Duration: 1 hour

Name: Second
Type: ECG (mV)
Location: Chest
Number of Channels: 2
Channels: Chest Lead VI, Modified Limb Lead II
```

```
Sampling Frequency: 100 Hz
Useful Duration: 2 hour

Name: First & Second
Type: ECG (mV)
Location: Chest
Number of Channels: 3
Channels: Chest Lead I (254 Hz), Chest Lead VI (100 Hz), Modified Limb Lead II (100 Hz)
Useful Duration: 2 hour
```

Two *Biosignal* objects can also be temporally concatenated using the >> operator, if one's domain comes after the other's. The >> operator should be read as "the first *Biosignal* comes after the second". Given two `ACC` objects, the first defined in [13h00, 14h00[and the second defined in [16h00, 18h00[, the following output would be produced when concatenating both:

```
print(acc1)
print(acc2)
print(acc1 >> acc2)
```

```
Name: First
Type: ACC (g)
Location: Left forearm
Number of Channels: 3
Channels: X, Y, Z
Sampling Frequency: 50 Hz
Useful Duration: 1 hour

Name: Second
Type: ACC (g)
Location: Left forearm
Number of Channels: 3
Channels: X, Y, Z
Sampling Frequency: 50 Hz
Useful Duration: 2 hours

Name: First >> Second
Type: ACC (g)
Location: Left forearm
Number of Channels: 3
Channels: X, Y, Z
Sampling Frequency: 50 Hz
Useful Duration: 3 hours
```

Furthermore, *Biosignal*s can be translated, contracted and expanded in amplitude and time, using the +, −, ∗, and / operators. Examples of these operations are:

```
ecg = ecg + 2
emg = emg * 0.5 - 3
eda /= 2
```

Minimum, maximum, mean and other statistics are also supported by *Biosignal*. For instance, one could make a *Biosignal*'s mean to be 0, using the following instruction:

```
ecg -= ecg.mean()
```

The purpose of having unit operations is to not limit the user in what they can do. It can be hard to find the balance between abstracting cumbersome operations and not restricting the users' capacity.

4.6 Storing and Sharing Biosignals

To fulfill the non-functional requirements 7, 9 and 10 gathered in Section 4.2, and to standardise how biosignals from different sources are stored and shared, a protocol was developed to serialise *Biosignal* and associated objects. What happens today at IT is that multiple external storage devices (in the order of Terabytes) are used to store biosignals and reports from hospitals and wearable devices, in the raw format they were provided as. Besides the large amount of storage they take, the long-term files can take hours do be read by a Python interpreter. Therefore, since users only use Python, a new format specific for Python memory loading was prototyped.

In order to serialize a *Biosignal* object and its associations, their content is reorganised in a frame of Python tuples, with the goal of minimising the number of bytes all content takes. A schematic of this frame can be found in Figure 4.4. Tuples are organised in a known ordered, so that each position unequivocally corresponds to a property. Classes *Timeseries*, *Segment*, *Event*, *Unit*, *Patient*, *MedicalCondition*, *SurgicalProcedure* and *Medication* also have their own frames, which are included in the dedicated positions inside the *Biosignal*'s frame. Since objects of all these classes are statefull, the content of each frame is populated with the state of each object. No more than what is needed to reset the objects' state at deserialisation is written in these frames, and no repeated nor redundant content is written, thus achieving the aimed size reduction. Repeated or redundant content is an issue in some formats, e.g. formats that indicate the date and time of each sample.

In the 0^{th} position of each class frame can be found a serial version number that identifies the class version that was serialised. This is used to verify that each loaded object and the serialised frame are compatible, and to ensure backwards compatibility with future class versions. In future LTBio releases, the serial version number of a class should increment by one if its state's properties change.

On serialisation, after getting the state of a *Biosignal* and its associations into one frame, its content is converted to binary code and compressed in the bz2 format for space reduction. *Biosignal* files can be identified by the symbolic ".biosignal" extension. On deserialisation, decompression takes place and

Figure 4.4: Serialised frame of a *Biosignal* object – top frame. The remaining frames are included in collections 5 and 6 of *Biosignal*, and 5, 6, and 7 of *Patient*.

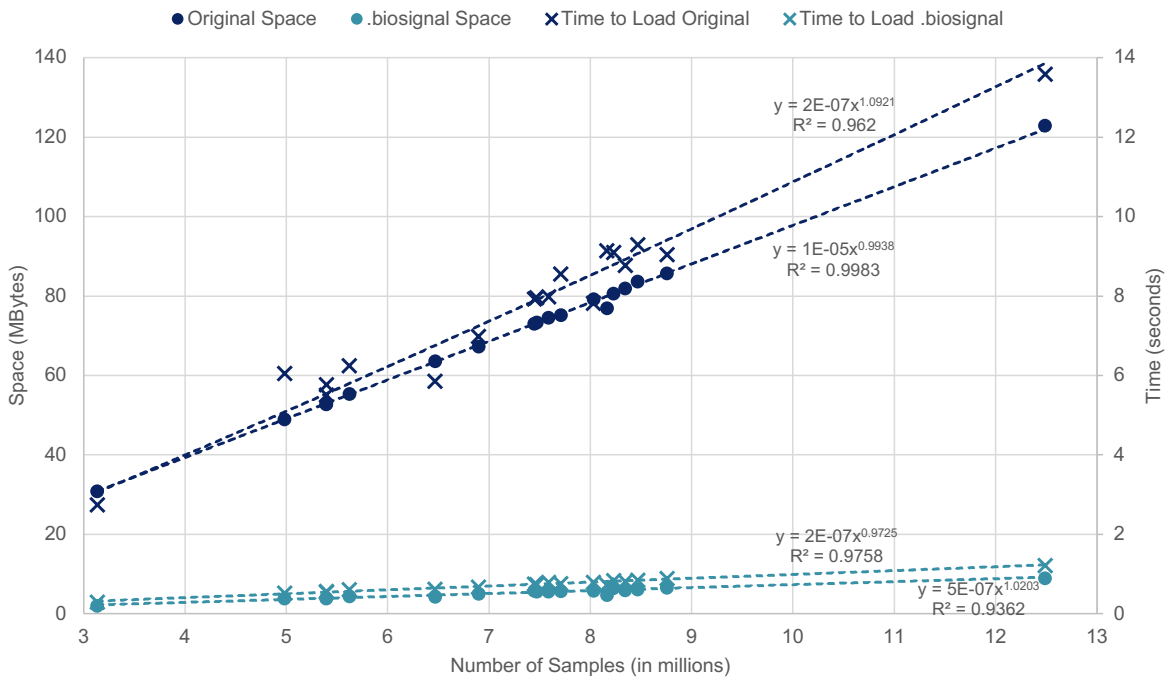


Figure 4.5: Space and loading time of ScientISST original files (dark blue) and .biosignal files (light blue) in respect to the number of samples. Tests with 158 files. Data points fitted to power laws.

the binary content is converted to Python tuples, from which empty objects get their state set, and the *Biosignal* gets reconstructed. Our `ecg` object can be serialised using the following instruction:

```
ecg.save('pathToFile.biosignal')
```

and recovered with the alternative constructor `Biosignal.load`:

```
ecg = ECG.load('pathToFile.biosignal')
```

4.6.1 Time and Space Empirical Complexity

The space `.biosignal` files occupy was compared with two current file formats being used at IT: files of ScientISST/BITalino devices, like those acquired from the *PreEpiSeizures* chestband (Figure 1.1C), and files acquired from the Empatica E4 wristband (Figure 1.1B). The elapsed time to load their content to memory was also studied and compared.

The ScientISST and BITalino devices allow to acquire multimodal biosignals and save them as ASCII text files (`.txt` or `.csv`) using appropriate software. These files are human-readable from start to end. The channels are divided by columns and the samples at each timepoint are given line by line, i.e., the sampling frequency has to be the same for all channels. ScientISST files include a header with at least 300 bytes, whereas BITalino files include a header with at least 600 bytes. A set of 158 ScientISST files were converted to `.biosignal` files to evaluate their space and loading time to LTBio *Biosignal* objects. These files included 2 ECG channels, 1 RESP channel, and 3 ACC channels. Hence 1 ScientISST text file originates 3 `.biosignal` files. Figure 4.5 plots the size of each pair of (1-ScientISST, 3-`.biosignal`) files in respect to the number of samples recorded. All files included the same metadata. In terms of

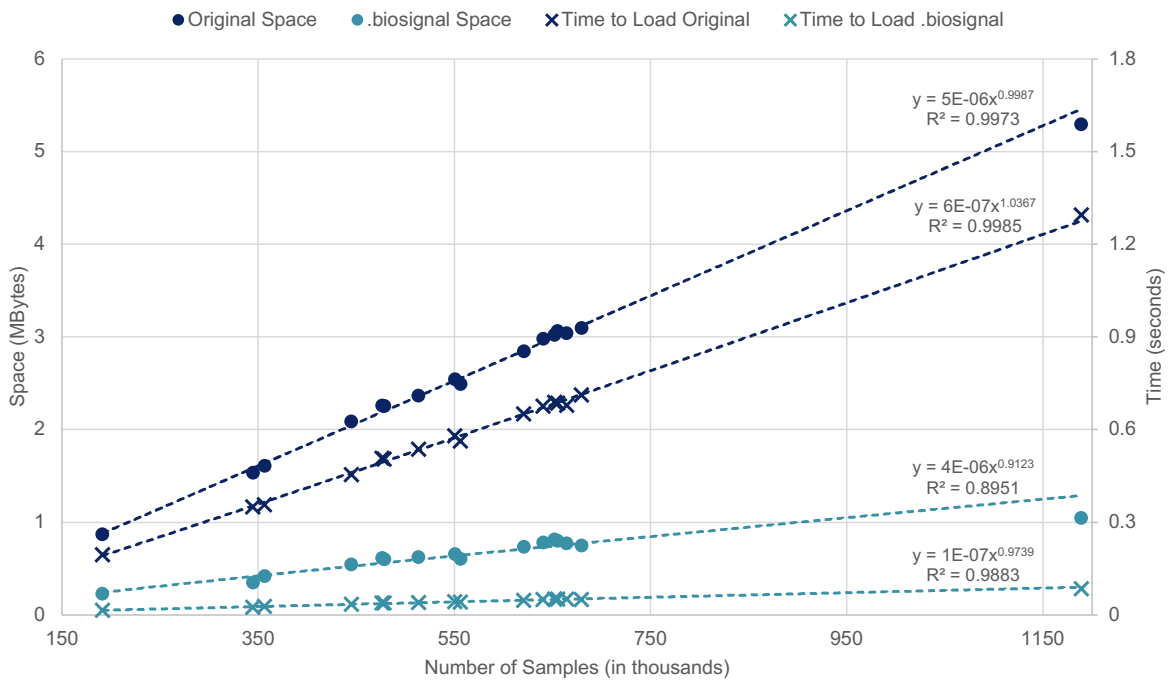


Figure 4.6: Space and loading time of Empatica E4 original files (dark blue) and .biosignal files (light blue) in respect to the number of samples. Tests with 96 files. Data points fitted to power laws.

space (circles), the computed power law is approximately linear in respect to the number of samples, either for the original files ($R^2 \approx 0.99$) and for the `.biosignal` files ($R^2 \approx 0.93$). However, the slope of the `.biosignal`'s is 2 orders of magnitude lower than the original's, representing in practice a median reduction in size of 13.2 times (1320%) the original size. In terms of loading time (crosses), the computed power is approximately 1.1 and 0.97 for the original files ($R^2 \approx 0.96$) and for the `.biosignal` files ($R^2 \approx 0.97$), respectively, which in practice represents a median reduction in loading time of 10.6 times (1060%) when instantiating from `.biosignal` files compared to using the original files.

The Empatica E4 device also produces files in `.csv` format, but with a rather different organisation of its data. Each biosignal modality is saved in a different `.csv` file, like in the LTBio proposed approach, and event onsets are saved in a separated file. The header of each modality file includes only the UNIX timestamp of the first sample. A common *header* is given in another file, which includes metadata and the necessary information to interpret the data, such as the sampling frequency, units of each channel, and the channel names. This common file accounts for 1543 bytes. A set of 96 Empatica E4 files were converted to `.biosignal` files to evaluate their space and loading time to LTBio *Biosignal* objects. These files included 1 EDA channel, 1 PPG channel, 1 temperature (TEMP) channel, and 3 ACC channels, separated by modality in four `.csv` files. Hence, four `.csv` files, a common header file, and an events file, originate four `.biosignal` files. Figure 4.6 plots the size of each pair of Empatica and `.biosignal` files in order to the number of samples recorded. Both file types include the same metadata. In terms of space (circles), the computed power laws in respect to the number of samples show 0.99 and a 0.91 powers for the original files ($R^2 \approx 0.99$) and for the `.biosignal` files ($R^2 \approx 0.89$), respectively. In practice, this represents a median reduction in size of 3.8 times (383%) the original size. In terms of

loading time (crosses), the computed power laws in respect to the number of samples show 1.03 and a 0.97 powers for the original files ($R^2 \approx 0.99$) and for the `.biosignal` files ($R^2 \approx 0.98$), respectively. The slope of the `.biosignal` curve is also half an order of magnitude smaller than the original's. All this accounts, in practice, to a median reduction in loading time of 13.3 times (1333%) when instantiating from `.biosignal` files compared to using the original files.

The space and time empirical evaluations were conducted on the same principal machine indicated in Chapter 3. All files were stored in the primary partition of the internal solid state drive. The machine was left overnight running the tests, plugged to the power wall, with no other significant process open rather than that running these tests. During the tests, at every instant, the Python process used $> 95\%$ CPU, $[12, 22]$ threads, and $[300, 900]$ MB of primary memory in its lifetime. All experiments were repeated 20 times and the average values were presented.

4.7 Learning from Biosignals and their Features

The exploratory process associated with feature engineering can become quite cumbersome and repetitive. So, extractor and selector agents were designed to expedite these tasks. Firstly, the process of extracting features is based on computations using the whole biosignal or segments of it. The task itself is invariable, but which computations are made are not. Secondly, the process of selecting features is also based on looking to all computed features and decide which get selected and which can be discarded. This task itself is also invariable, but how the decision is made is not. Hence, once again, agents were designed to know what to do – to extract and to select arbitrary features – but not how to compute them. The "how" is up to the user.

The user can design a *FeatureExtractor* by referring a procedure that, given an array of samples, computes a feature from them. This procedure is accepted by each *Segment* in a similar way a filter visits them, except that the computed features are not written in-place, but are rather outputted as a new *Timeseries*. The *.features* package comes with some examples of these procedures for statistical features (`TF.mean`, `TF.variance`, `TF.deviation`, etc.), and modality-specific procedures, such as HRV features (e.g. `HRV.csi`, `HRV.nn50`, etc.). Users can easily extend this collection by defining their own procedures, conforming with the mandatory signature.

The user can also design a *FeatureSelector* by referring a procedure that, given a set of features, selects some, none, or all. Formally, given a set of *Timeseries*, it decides which ones match certain conditions, and returns the subset that does. An independent binary decision takes place for each feature *Timeseries*. The following piece of code illustrates how three features can be extracted from our `ecg` and selected with a simple threshold:

```
from ltbio.features import *
extractor = FeatureExtractor(TF.mean, TF.variance, HRV.nn50)
selector = FeatureSelector(lambda x: x.max() > 90)
features = selector.apply(extractor.apply(ecg))
```

4.7.1 Automating the Training Supervised Models

In most cases, researchers create datasets of biosignal segments or features extracted from them to train machine learning models. These machine learning models are usually supervised or unsupervised. For now, LTBio supports automation in training only supervised models.

LTBio offers silent integration with machine learning Python libraries, currently SciKit Learn and PyTorch. Users are familiar with these libraries, and it is not the LTBio goal to replace them, but rather to provide an interface with them contextualised to the Domain of biosignals. In the *.ml* package, users can find useful components under the *datasets*, *metrics*, and *supervised* modules. Module *unsupervised* is planed to be implemented in the future.

A *SupervisedModel* is an abstract class representing a generic machine learning model that is trained in a supervised way, i.e., with labelled examples. It has a `design` and a history set of trained `versions`. Every *SupervisedModel* has a *train* and a *test* methods, which, respectively, train the model and test predictions with it. The logic of doing so is specific of each subclass. There is one subclass for each machine learning Python library: *SkLearnModel* and *TorchModel*. The user designs models as usual using the structures offered by these libraries, and then pass their designs to the `design` attribute on instantiation. For example, the ResNet design presented earlier in Algorithm 1 can implemented in a PyTorch module. This design would be encapsulated like this:

```
1 from torch import nn
2 from ltbio.ml.supervised import TorchModel
3
4 resnet = nn.Module(...)
5 model = TorchModel(resnet, name='My first model')
```

To train a model, a *BiosignalDataset* and a *SupervisedTrainConditions* objects should be given. In this example, one could have:

```
1 from torch import Adam, BCELoss
2 from ltbio.ml.supervised import SupervisedTrainConditions
3 from ltbio.ml.datasets import EventDetectionDataset
4
5 dataset = EventDetectionDataset(ecg, 'seizure2')
6 conditions = SupervisedTrainConditions(validation_ratio=0.2, optimizer=Adam(), loss=BCELoss(),
      batch_size=128, shuffle=True, epochs=500, patience=20)
7 train_results = model.train(dataset, conditions)
```

Line 7 above starts training the model using the PyTorch library and their latest recommendations, without the user intervention. The same `train` method is available for *SkLearnModel*, which interfaces with the SciKit Learn. Some good practices were adopted, without the user having to program them:

- When GPUs or other acceleration hardware is available, *SupervisedModel*s are automatically trained there.
- The examples are indexed from the indicated *Biosignal* objects on-the-fly[9].

[9] On-the-fly means prepared in "real-time" as they are needed.

- Multiple threads can be opened to pre-fetch examples, if computational power is available.
- When training in batches, if the dataset is large, memory is being continuously released.
- If data augmentation techniques are used, these are applied to examples on-the-fly.

In line 5 of the previous piece of code, an *EventDetectionDataset* was instantiated, which was actually what was used in Chapter 3 to accelerate research. Given a collection of segmented multi-channel *Biosignal*s, and a collection of event names, an *EventDetectionDataset* creates a binary dataset with examples of positive targets being the segments temporally corresponding to the specified events, and examples of negative targets with the remaining segments. Other parameters to customise the dataset can be found in the documentation. Since biosignal augmentation techniques have proved to be useful in learning tasks [203, 217], some techniques are also available to create new examples on-the-fly. The following code snippet exemplifies how the datasets of Subsection 3.3 were created:

```
from ltbio.ml.datasets.augmentation import *
dataset = EventDetectionDataset(ecg, ('seizure1', 'seizure2', ) name='Train Seizures')
dataset.augment((Sine(0.02), Randomness(0.01), Scale(0.85), Shift(0.05), Drop(0.02)))
```

Datasets can be inspected by printing them:

```
print(dataset)
```

```
Train Seizures
Negative Examples: 342035 (49%)
Positive Examples: 351900 (51%)
Total: 693935
```

The dataset object mimics the existence of examples, but in fact each is only indexed from `ecg` when about to constitute a batch. *EventDetectionDataset* is a subclass of the abstract *BiosignalDataset*. Currently, LTBio offers three subclasses to quickly arrange datasets from biosignals, which most reflect common tasks in the community:

- *SegmentToSegmentDataset*: in each example, the object and target are segments.
- *SegmentToValueDataset*: in each example, the object is a segment and the target is a value.
- *EventDectionDataset*: subclass of the previous, where segments within an event have target 1.

SegmentToSegmentDataset is useful when training ML models to denoise biosignals. *SegmentToValueDataset* is useful in regression tasks, e.g. predicting the probability of some genetic condition. These two options behave similarly to the *EventDetectionDataset*, but examples are formatted in a different way. Operations can also be applied to *BiosignalDataset*s, like splitting them:

```
a, b = dataset.split(0.8, 0.2)
```

or concatenating the datasets of multiple patients to form a `CohortDataset`:

```
cohort = dataset1 + dataset2 + dataset3
```

Another object required to train a model is a *SupervisedTrainConditions*, which carries information about how a training session should be conducted. This includes which solver, optimizer and loss functions to use, the train-test split proportions, the shuffling options, batching options, the number of iterations or epochs, among other conditions. They can also contain extra values to dynamically change the design hyperparameters at each training session.

In a similar way, the `test` method allows to test a *SupervisedModel*, i.e. make predictions and evaluate them. A *BiosignalDataset* should be given as well, and a collection of *Metric*s:

```
from ltbio.ml.metrics import *
dataset = EventDetectionDataset(ecg, 'seizure9')
metrics = (F1Score('weighted'), AUC(), TrainTestLossPlot(), )
test_results = model.test(dataset, metrics)
```

A *Metric* is a standard measure of efficacy or efficiency of a model. It can be computed as a real number (*ValueMetric*) or sequence of numbers that should be appreciated in a plot (*PlotMetric*). The framework comes already with the metrics mentioned in Section 2.4 as *ValueMetric*s, and others offered by SciKit Learn and PyTorch. An example of a *PlotMetric* is the train-validation loss plot, or the effect of feature permutation. Personalised metrics can be extended by the user.

When training a model, the loss and other metrics are returned in a *SupervisedTrainResults* object. When testing a model, the metrics the user asked for are returned in a *PredicitionResults* object. These can be analysed right after each session, or serialised and later revisited when convenient.

To alleviate the exploratory overhead often associated with experimenting different training conditions for a fixed model design, *SupervisingTrainer* agents were implemented. These agents repeat the train-test cycle for a *SupervisedModel* with a sequence of one or more *SupervisedTrainConditions* objects. If different model hyperparameters are defined in the given *SupervisedTrainConditions* objects, the model design can also change and be evaluated. The best *SupervisedTrainResults* and *PredictionResults* are returned:

```
from ltbio.ml.supervised import SupervisingTrainer
c2 = SupervisedTrainConditions(batch_size=64, shuffle=True)
c3 = SupervisedTrainConditions(batch_size=256, shuffle=True)
c4 = SupervisedTrainConditions(batch_size=256, shuffle=False)
trainer = SupervisingTrainer(model)
results = trainer.apply(dataset, (conditions, c2, c3, c4), metrics)
```

Moreover, each *PredictionResults* has a *name* describing which training conditions lead to those predictions and the model's version number they correspond to, so it is easier to recover that version, if needed. *SupervisingTrainer* agents also produce a PDF report with the results yielded by each set of conditions. An exemplar report is shown in Appendix B.

4.7.2 Advanced Automation with Pipelines

Finally, for advanced users – described in Section 4.8 – advanced automation tools were developed. The cunning reader has certainly noticed the steps involved in biosignal investigation resemble very

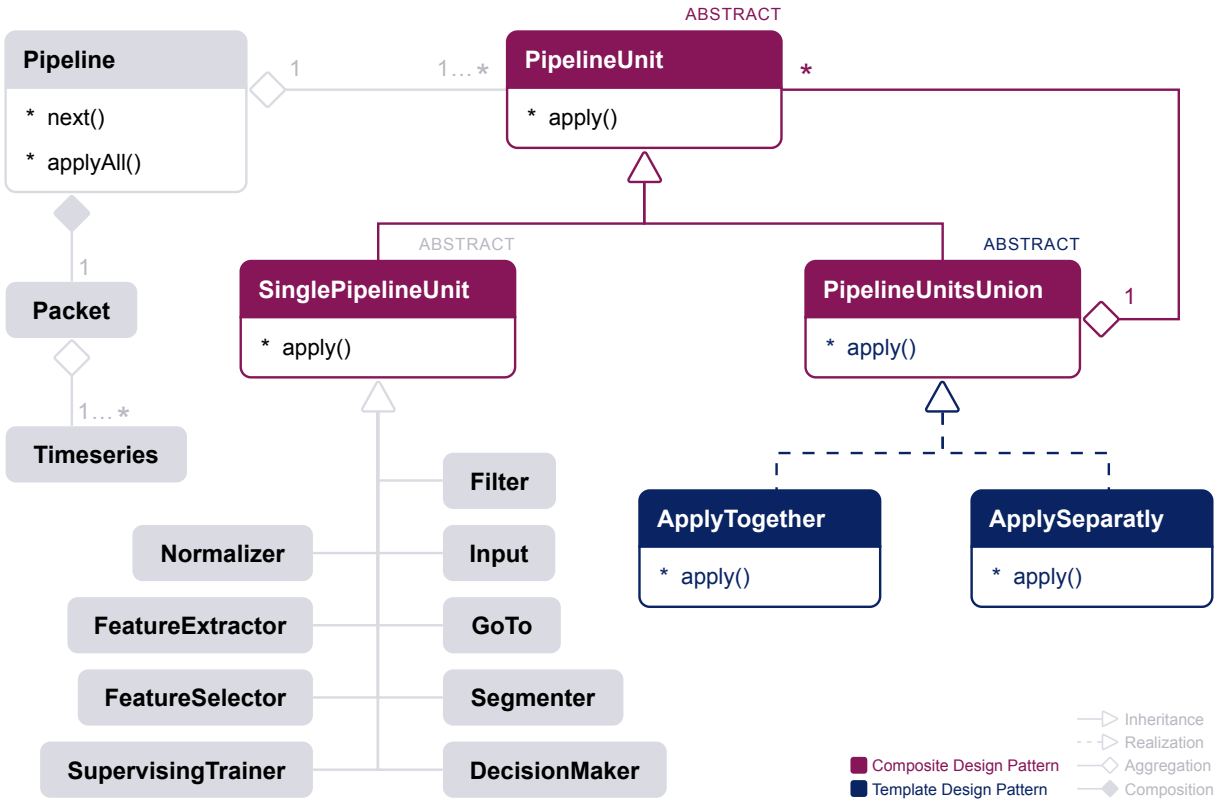


Figure 4.7: Top-level UML diagram of the *.pipeline* package of LTBio. The composite and template design patterns allow the building of infinite combinations of processing pipelines.

much a pipeline. The processing agents so far presented have been called ”agents” because all of them derive from a class named *PipelineUnit*, depicted in Figure 4.7. Every object that is a *PipelineUnit* (formaters, filters, feature extractors and selectors, and trainers) can be included in a *Piepline*. A *Pipeline* is a sequence of processing steps to be applied to *Biosignal*s. One could join all the processing steps used in the previous examples in a *Pipeline*, like this:

```
pipeline = my_passband >> normalizer >> segmenter >> trainer
print(pipeline)
```

```
Untitled Pipeline with 4 steps
1. Passband FIR Filter ([1, 40] Hz, order 200)
2. Normalizer (min-max method)
3. 10s Segmenter (2s overlap)
4. TorchModel ('My firt model')
```

And apply it to our `ecg` like this:

```
test_results = pipeline(ecg)
```

Pipeline objects are stateful, keeping track of the current step being executed, passing the output from step $i-1$ to the input of step i with a *Packet* object. A *Packet* is an object that carries content from one step to the next, including the time series being processed and other relevant data like results or control variables.

Although the so-far described workflow is generally a linear pipeline, in some projects it may branch, i.e. the output of one unit is fed to multiple units, or vice-versa. Also, when multiple *Biosignals* are given, these can be analysed together or separately by each unit. For that, *PipelineUnitsUnion* allows to group units that behave as sub-pipelines. The user must specify if multiple biosignals must be fed to the union together or in separate, using the realisations *ApplyTogether* or *ApplySeparatly*:

```
pipeline = ApplySeperatly(normalizer >> extractor >> selector) >> trainer
test_results = pipeline(ecg, emg, acc)
```

This flexible feature was implemented using the Composite Design Pattern – Figure 4.7. More advanced *PipelineUnit*s are offered, for instance to inject *Timeseries* in the middle of a pipeline, to jump between steps; and others shown in the published documentation.

4.8 Software Testing and Framework Evaluation

4.8.1 Automatic Tests

Unit and integration tests were designed prior to development of each package to serve as specification and later to ensure quality and stability of the framework over time. A total of 205 unit tests and 12 integration tests, with 63% code coverage, were developed[10]. The live status of each package is available in the GitHub project's page for transparency.

4.8.2 Evaluation with Users

Ten users were recruited to use and evaluate LTBio: 2 novice users, 4 advanced users and 4 expert users. Novice users have completed their engineering education, have conducted biosignal research in the past, they do not do so today, but they might in the future. Advanced users regularly conduct biosignal research in the context of a Master's or Doctoral thesis. Expert users practice biosignal investigation at some research institution for more than a year. Figure 4.8 further characterises these users. Users have evaluated LTBio using their own work computers and their integrated development environment (IDE) of choice. A small LTBio sheet-cheat was provided and all LTBio documentation was made available. Additionally, users could use the Internet with no restriction.

In each session, five tasks were proposed to the user. Each task consisted in resolving some familiar biosignal research questions. A task was considered completed once the user answered these questions, or uncompleted if the user requested to move on to the next task. A task was considered successfully completed if correct answers were given, or unsuccessfully completed otherwise. A verbal correct answer would suffice to consider the task successful, independently of what code was written to find it. Nonetheless, solution proposals to code the way into the correct answers of each task were

[10] Access `github.com/jomy-kk/IT-LongTermBiosignals/tree/main/tests`

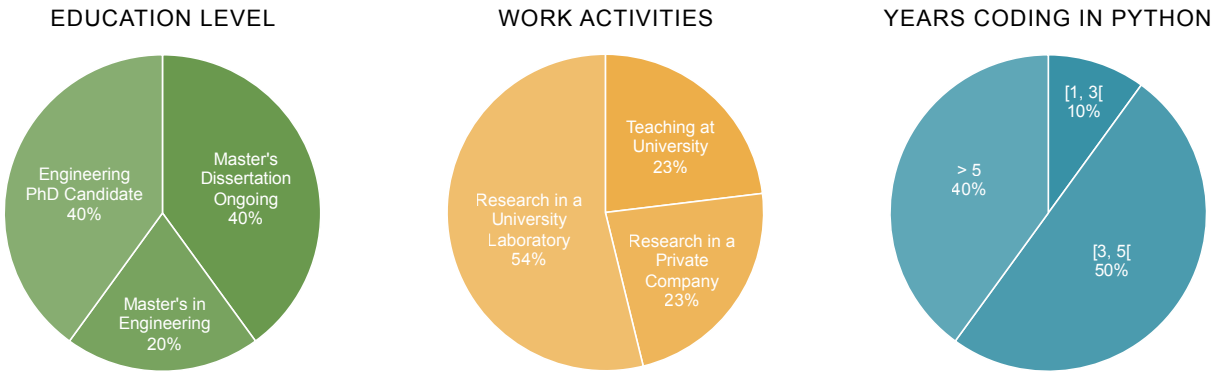


Figure 4.8: Demographic characterisation of the users that evaluated the LTBio Python library.

provided[11]. In case the user could not complete one of the tasks, they could proceed to the following one starting from these solutions.

In the first task, a `.biosignal` file, containing an ECG, was given and the user was asked how many channels did the biosignal had, what was its useful duration, and how many events were annotated. The correct answers were, respectively, 2 channels, 41.5 minutes, and 7 events. All users were able to effortlessly read the biosignal and to give the correct answers either by printing its textual description (30%) or/and by accessing properties individually (80%). One user took approximately 4 times more time than the average, because they computed and summed the duration of each interval in the *Biosignal*'s domain to answer the useful duration, instead of accessing the `Biosignal.duration` property.

In the second task, the user was asked to plot the ECG when the subject was running. To index that period of time, 80% of the users used the event name – something like `y = x['run']` – whereas 20% of users indexed using the initial and final timepoints. In either case, users recalled the event name or domain by accessing `Biosignal.events` or by printing the *Biosignal*'s textual description. The users were also asked in the beginning to indicate whether the electrodes had been correctly positioned, once they got to see the ECG visually. Only 20% of the users immediately used `.preview.plot()` after indexing, whereas the remaining firstly used `.plot()`, and only after realising they would not be able to answer the question by inspecting such a long period of time (30 minutes of running), they immediately used `.preview.plot()`. A visual inspection by a trained eye would give an answer like: "*Yes, the ECG leads were acquired upside down*". All users answered correctly. Finally, since the ECG leads were not right, when users were asked to invert them, 60% multiplied the *Biosignal* object by -1, whereas the remaining used `Biosignal.invert()`. One user tried to used NumPy to invert the time series, but, soon after the first error message, they realised there was an already built-in method for that.

In the third task, another `.biosignal` file, containing an EMG of the same subject, was given and the user was asked to add it as myogenic noise to the ECG. After reading the EMG, 60% of users tried straightforward to add both *Biosignal* objects with the + binary operator, completing the task. The remaining used the `Biosignal.withAdditiveNoise()` constructor to create the noisy signal, also com-

[11] Access `github.com/jomy-kk/IT-LongTermBiosignals/tree/main/research_journal/testsWithUsers/answers`

pleting the task. One expert user was skeptical about how would a 1-channel EMG add to a 2-channel ECG, and inspected the EMG biosignal first. All users felt compelled to visually check for myogenic noise in the created noisy signal, for which they used `.preview.plot()` again, successfully verifying noise in both ECG channels.

In the fourth task, the user was asked to resample, normalise, filter and segment the ECG. All users understood they had to use, respectively, `.resample()`, a *Normalizer*, a *FrequencyDomainFilter*, and a *Segmenter*. Difficulties were shown in filtering because only one user was familiarised with enumeration structures. A filter design equivalent to:

```
design = FrequencyDomainFilter(FIR, BANDPASS, cutoff=(1, 40), order=200)
```

was required, but erroneously users tried first:

```
design = FrequencyDomainFilter('fir', 'bandpass', cutoff=(1, 40), order=200)
```

Similarly, in *Segmenter*, users first tried to use:

```
segmenter = Segmenter(2)
```

but only after consulting the documentation used:

```
segmenter = Segmenter(timedelta(seconds=2))
```

except for the same expert user mentioned. After creating the signal processing agents, only one advanced user autonomously wanted to create a *Pipeline* to apply all processing steps at once. The remaining applied individually each agent on the ECG. Once again, all users felt compelled to visually check for the changes applied using `.preview.plot()`.

In the fifth and final task, the user was asked to train a MLP to detect when the subject was running. The MLP design, implemented with the PyTorch library, was already provided. All users created an *EventDetectionDataset* to target the segments within the '*run*' event interval with 1, and the remaining with 0. Users were also able to create a *SupervisedTrainingConditions* object with the training conditions indicated. The most common mistake was users calling `.train()` on the design, before encapsulating it in a *TorchModel*, which they did after consulting the documentation.

Overall, all users were able to complete all tasks successfully in 25 minutes on average. Given that the users had no previous contact with LTBio, this is a more than satisfactory duration to achieve what was proposed: reading and inspecting an unknown biosignal, and pre-processing it to later train a ML model with it. Without asking, once users learned how to use `.preview.plot()`, this became the most popular feature, therefore fulfilling functional requirement 21. Furthermore, these users evaluated LTBio with an average score of 85.75 $\pm$ 10.14 (out of 100) in the system usability scale (SUS) scale[12]. In 2014, BioSPPy received a similar score of 83.40 $\pm$ 11.88 with 25 undergraduate biomedical students, although these users did not explore BioSPPy with a fixed set of tasks [215]. Figure 4.9 summarises these results in an infographic. Useful comments and suggestions were also given in written, which will certainty be addressed in the next development iteration[12].

[12]Results made publicly available at `tinyurl.com/yxphcx24` until December 31, 2022.

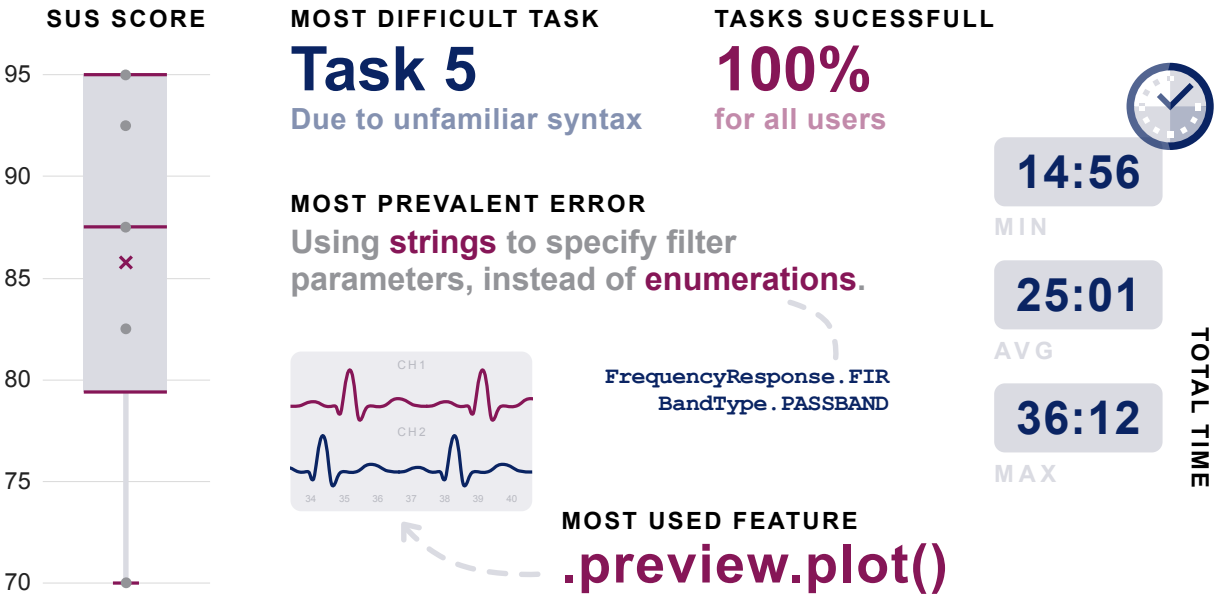


Figure 4.9: Infographic highlighting the main findings of the LTBio evaluation with users. SUS score evaluation on the left. Total time to complete the tasks on the right.

4.9 Community, Documentation and Expandability

The LTBio framework offers three levels of documentation: reference-oriented, learning-oriented, and internal documents. At `github.com/jomy-kk/IT-LongTermBiosignals/wiki/API-Reference` the full documentation can be found for familiarised users that look for specific API references. Learning-oriented quick-start guides, Q&As, and educational Jupyter notebooks can be found at `github.com/jomy-kk/IT-LongTermBiosignals/wiki` to facilitate the learning curve of novice users. The technical documentation (specification, UML diagrams, quality tests) is available only at IT, for maintainability.

Community Contributions

In the future, LTBio is planed to become an open-source project with community and third-party contributions, once guidelines are created. The LTBio architecture was designed to be easily expanded:

- More types of *Biosignal* and *BiosignalSource* can be created as users expand their collaborations and biosignal datasets.
- Depending on the project's needs, more types of medical conditions, surgical procedures, and medications can be created to enrich the analysis process and event associations.
- The set of standard body locations can be expanded whenever needed.
- More feature extraction and selection procedures are expected to be added with minimal effort as contributions from the community.
- Users can design machine learning models using their preferred libraries, which are projects from other groups in constant growing and expansion.

- Personalised metrics can be used according to the project's needs.
- Infinite different pipelines can be created, depending on the order units are piped together.
- If the offered workflow agents are insufficient, more agents can be implemented by the user and inserted on pipelines.

Furthermore, plans are in order constitute a team at IT to tackle bug reports and to further expand the framework. A priority-driven course of action has been discussed for the months to follow:

High priority

- Indexing portions of a *Biosignal* from disk, rather than loading it to primary memory.
- Benchmark the Biosignal File Format against EDF+ and HDF5 formats.
- Fetching from public biosignal databases.
- Implement whole-pipeline reports.

Low priority

- Interface with TensorFlow.
- Integrate with ScientISST Web.
- Developing a pedagogical playground GUI.
- Write more educational content using the framework.

Chapter 5

Conclusion

In this thesis, support was given to collect and store vEEG biosignals of patients with epilepsy, hospitalised at two *PreEpiSeizures* collaborating healthcare units. The ECG of 11 patients who experienced 88 clinical seizures, comprising a total of 893 hours, was used to prototype and evaluate a novel seizure prediction method. In this method, patient-specific deep residual learning of ECG features is proposed to classify preictal and non-preictal segments of patients with epilepsy. It is also proposed that consecutive preictal segments should raise an alarm, for which a cohort median 0.774 F1-score and 4.00 FAR/day were attained. Seizures were able to be predicted with a median latency of -9.1 minutes, not taking into account the 3 patients that did not respond to this method. These results go in hand with the state-of-the-art although the ResNet segment classification performance needs to improve. The following steps include testing this method online on the chestband being worn by patients with epilepsy, in clinical settings, and investigate weather an improved version of the method can be brought to the clinical practice.

Exploratory work was also conducted on different ResNet configurations for ECG classification, adjoining the contributions of the recent works [206, 198, 208, 218, 219, 220] that have been proposing this type of learning for the ECG. Contrasting with these, the present work lays the foundations for ResNet inter-heartbeat feature extraction for preictal recognition of epileptic seizures, which can be helpful for other works aiming to detect other clinical episodes that deregulate the ANS.

The complications gathered from the *PreEpiSeizures* project, and other biosignal research groups at IT motivated the development of a novel software framework. The Long-Term Biosignals Framework (LTBio) was designed specifically to mitigate the complications of managing and processing multimodal long-term biosignals. It includes a Python library which interfaces with industry-standard Python packages for biosignal processing and ML. LTBio can assist in creating reproducible processing pipelines and rapidly prototyping ML models that can learn from biosignals. Contextualisation of the biosignals domain of knowledge to the algorithms and libraries already used today is at the core of expediting this process. Moreover, a biosignal file format was devised to allow for fast Python serialisation of biosignals,

and to offer an easy way to share them among LTBio users. LTBio received an excellent 85.75 (out of 100) score in the SUS, and all users that evaluated it were able to complete multiple tasks from reading biosignals to training a DL model with them in an average of 25 minutes, with no previous experience with LTBio. It is believed that as the volume of data keeps increasing, the tools here offered will help researchers to focus more on the tasks at hand, rather than the programming technicalities behind them. The requirements elicited were all fulfilled by the main features LTBio offers:

- Biosignals can be quickly inspected and variables of interest can be rapidly identified.
- Exploratory analysis can be systematised in processing blocks, the order of which can be arranged and rearranged according to different research plans, and repeated effortlessly.
- Easily reproducible analysis scripts can be shared by different research teams, by abstracting standard methods in the community, e.g. filtering, formatting, extracting features, etc.
- Biosignal datasets can be easily prepared for machine learning (ML) tasks.
- Silent integration with industry-standard software, e.g. BioSPPy, SciKit Learn, and PyTorch, and file norms, e.g. TRC, HDF5 and EDF data files.

As a second goal, LTBio was proposed to be used in Education. Students with basic or minimal education in computer science can struggle in learning how to program and how to automate their analyses. Most biosignal processing and analysis courses at University focus more on passing the theoretical concepts about this Domain, and when it comes to evaluation projects, the overall pedagogical goal should be students know how to manipulate these concepts and know how to interpret results. That can be more easily accomplished with LTBio, because it abstracts coding technicalities and allows students to get to results faster, so that more time can be devoted to critical-thinking.

References

[1] "Epitel Epilog seizure monitoring device." [Online]. Available: https://www.epitel.com

[2] "Embrace2 Seizure Monitoring Device." [Online]. Available: https://www.empatica.com/embrace2

[3] A. F. Manso, A. L. N. Fred, R. C. Neves, and R. C. Ferreira, "Real-Time Pervasive Monitoring System for Ambulatory Patients," in *2018 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Jun. 2018, pp. 1–6.

[4] R. S. Fisher, C. Acevedo, A. Arzimanoglou, A. Bogacz, J. H. Cross, C. E. Elger, J. Engel Jr, L. Forsgren, J. A. French, M. Glynn, D. C. Hesdorffer, B. Lee, G. W. Mathern, S. L. Moshé, E. Perucca, I. E. Scheffer, T. Tomson, M. Watanabe, and S. Wiebe, "ILAE Official Report: A practical clinical definition of epilepsy," *Epilepsia*, vol. 55, no. 4, pp. 475–482, 2014.

[5] WHO, "Epilepsy." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/epilepsy

[6] J. Pimentel, "O doente epiléptico pode e deve ter uma vida normal," *Público*, Feb. 2020. [Online]. Available: https://www.publico.pt/2020/02/10/impar/opiniao/doente-epileptico-vida-normal-1902967

[7] P. Kwan and M. J. Brodie, "Early Identification of Refractory Epilepsy," *New England Journal of Medicine*, vol. 342, no. 5, pp. 314–319, Feb. 2000.

[8] P. Kwan, S. C. Schachter, and M. J. Brodie, "Drug-Resistant Epilepsy," *New England Journal of Medicine*, vol. 365, no. 10, pp. 919–926, Sep. 2011.

[9] B. Litt and J. Echauz, "Prediction of epileptic seizures," *The Lancet Neurology*, vol. 1, no. 1, pp. 22–30, May 2002.

[10] E. H. Al Zaid, "Prevalence of patients with epilepsy unfit to drive," *Journal of Family & Community Medicine*, vol. 26, no. 1, pp. 51–56, 2019 Jan-Apr.

[11] R. Sutter, P. W. Kaplan, and S. Rueegg, "Outcome predictors for status epilepticus-what really counts," *NATURE REVIEWS NEUROLOGY*, vol. 9, no. 9, pp. 525–534, Sep. 2013.

[12] C. M. DeGiorgio, A. Curtis, D. Hertling, and B. D. Moseley, "Sudden unexpected death in epilepsy: Risk factors, biomarkers, and prevention," *Acta Neurologica Scandinavica*, vol. 139, no. 3, pp. 220–230, Mar. 2019.

[13] M. Tombini, G. Assenza, L. Quintiliani, L. Ricci, J. Lanzone, and V. Di Lazzaro, "Epilepsy and quality of life: What does really matter?" *Neurological Sciences*, vol. 42, no. 9, pp. 3757–3765, Sep. 2021.

[14] F. M. C. Besag and M. J. Vasey, "Social cognition and psychopathology in childhood and adoles-

cence," *Epilepsy & Behavior: E&B*, vol. 100, no. Pt B, p. 106210, Nov. 2019.

[15] K. Eichenwald, *A Mind Unraveled: A Memoir*, 1st ed. New York: Ballantine Books, 2018.

[16] J. Saraiva, A. S. Carmo, M. Abreu, A. Fred, and H. Plácido da Silva, "Detection and Anticipated Prediction of Epileptic Seizures on Continuous-Monitoring Wearable Devices: A Systematic Review and Research Proposal," Jan. 2022.

[17] J. L. Semmlow and B. Griffel, *Biosignal and Medical Image Processing*, third edition ed. Boca Raton: CRC Press, Taylor & Francis Group, CRC Press is an imprint of the Taylor & Francis Group, an Informa business, 2014.

[18] M. Kutz, *Biomedical Engineering and Design Handbook, Volumes I and II (2nd Edition).* New York, USA: McGraw-Hill Professional Publishing, 2010. [Online]. Available: http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4657812

[19] Alejandro Torres Garcia, Carlos Reyes Garcia, Luis Villasenor-Pineda, and Omar Mendoza-Montoya, *Biosignal Processing and Classification Using Computational Learning and Intelligence*, 1st ed. Elsevier, 2022.

[20] E. A. Ashley and J. Niebauer, *Conquering the ECG*. Remedica, 2004. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK2214/

[21] R. Edelberg, "Biopotentials from the skin surface: The hydration effect," *Annals of the New York Academy of Sciences*, vol. 148, no. 1, pp. 252–262, Feb. 1968.

[22] F. Shi and X. Li, "Development of automatic motion artifact detection in mobile ECG monitor based on wavelet transform," in *2012 IEEE International Conference on Intelligent Control, Automatic Detection and High-End Equipment*, Jul. 2012, pp. 166–170.

[23] F. Mohaddes, R. L. da Silva, F. P. Akbulut, Y. Zhou, A. Tanneeru, E. Lobaton, B. Lee, and V. Misra, "A Pipeline for Adaptive Filtering and Transformation of Noisy Left-Arm ECG to Its Surrogate Chest Signal," *Electronics*, vol. 9, no. 5, p. 866, May 2020.

[24] J. Zhong, D. Hai, J. Cheng, C. Jiao, S. Gou, Y. Liu, H. Zhou, and W. Zhu, "Convolutional Autoencoding and Gaussian Mixture Clustering for Unsupervised Beat-to-Beat Heart Rate Estimation of Electrocardiograms from Wearable Sensors," *Sensors*, vol. 21, no. 21, p. 7163, Jan. 2021.

[25] A. Dabbaghian, T. Yousefi, S. Z. Fatmi, P. Shafia, and H. Kassiri, "A 9.2-g Fully-Flexible Wireless Ambulatory EEG Monitoring and Diagnostics Headband With Analog Motion Artifact Detection and Compensation," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1141–1151, Dec. 2019.

[26] V. P. Rachim and W.-Y. Chung, "Wearable Noncontact Armband for Mobile ECG Monitoring System," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 6, pp. 1112–1118, Dec. 2016.

[27] F. S. Butt, L. La Blunda, M. F. Wagner, J. Schäfer, I. Medina-Bulo, and D. Gómez-Ullate, "Fall Detection from Electrocardiogram (ECG) Signals and Classification by Deep Transfer Learning," *Information*, vol. 12, no. 2, p. 63, Feb. 2021.

[28] J. Lazaro, N. Reljin, M.-B. Hossain, Y. Noh, P. Laguna, and K. H. Chon, "Wearable Armband Device for Daily Life Electrocardiogram Monitoring," *Ieee Transactions on Biomedical Engineering*, vol. 67,

no. 12, pp. 3464–3473, Dec. 2020.

[29] S. Behbahani, "A Review of Significant Research on Epileptic Seizure Detection and Prediction using Heart Rate Variability," *Turk Kardiyoloji Dernegi Arsivi-Archives of the Turkish Society of Cardiology*, 2018.

[30] M. Kolodziej, A. Majkowski, P. Tarnowski, R. J. Rak, and A. Rysz, "A new method of cardiac sympathetic index estimation using a 1D-convolutional neural network," *BULLETIN OF THE POLISH ACADEMY OF SCIENCES-TECHNICAL SCIENCES*, vol. 69, no. 3, Jun. 2021.

[31] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, C. A. Teixeira, and A. Dourado, "Pre-ictal heart rate variability assessment of epileptic seizures by means of linear and non-linear analyses," *ANATOLIAN JOURNAL OF CARDIOLOGY*, vol. 13, no. 8, pp. 797–803, Dec. 2013.

[32] L. Gagliano, E. B. Assi, D. H. Toffa, D. K. Nguyen, and M. Sawan, "Unsupervised Clustering of HRV Features Reveals Preictal Changes in Human Epilepsy," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, Jul. 2020, pp. 698–701.

[33] G. Giannakakis, M. Tsiknakis, and P. Vorgia, "Focal epileptic seizures anticipation based on patterns of heart rate variability parameters," *COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE*, vol. 178, pp. 123–132, Sep. 2019.

[34] M. K. Moridani and H. Farhadi, "Heart rate variability as a biomarker for epilepsy seizure prediction," *BRATISLAVA MEDICAL JOURNAL-BRATISLAVSKE LEKARSKE LISTY*, vol. 118, no. 1, pp. 3–8, 2017.

[35] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, and A. Dourado, "Classification of ictal and seizure-free HRV signals with focus on lateralization of epilepsy," *Technology and Health Care*, vol. 24, no. 1, pp. 43–56, Jan. 2016.

[36] A. P. H., I. B. A., and E. J. A. P., "Ultra-short-term analysis of heart rate variability for epileptic seizures prediction," in *2021 IEEE 2nd International Congress of Biomedical Engineering and Bioengineering (CI-IB&BI)*. Bogota D.C., Colombia: IEEE, Oct. 2021, pp. 1–4.

[37] L. Billeci and M. Varanini, "Characterizing Electrocardiographic Changes During Pre-seizure Periods through Temporal and Spectral Features," in *2017 Computing in Cardiology Conference*, Sep. 2017.

[38] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, G. Attarodi, C. A. Teixeira, and A. Dourado, "Epileptic Seizure Behaviour from the Perspective of Heart Rate Variability," 2012.

[39] U. Işık, C. Ayabakan, K. Tokel, and M. M. Özek, "Ictal electrocardiographic changes in children presenting with seizures: ECG in seizures," *Pediatrics International*, vol. 54, no. 1, pp. 27–31, Feb. 2012.

[40] A. Jordan, M. Bausch, and R. Surges, "Semi-automatic quantification of seizure-related effects on heart activity," *Epilepsy Research*, vol. 157, p. 106187, Nov. 2019.

[41] V. Novak, A. L. Reeves, P. Novak, P. A. Low, and F. W. Sharbrough, "Time-frequency mapping of R–R interval during complex partial seizures of temporal lobe origin," *Journal of the Autonomic Nervous System*, vol. 77, no. 2-3, pp. 195–202, Sep. 1999.

[42] B. He, Ed., *Neural Engineering*, second editon ed. New York: Springer, 2013.

[43] D. Purves and S. M. Williams, Eds., *Neuroscience*, 2nd ed. Sunderland, Mass: Sinauer Associates, 2001.

[44] C. Baumgartner and S. Pirker, "Video-EEG," *Handbook of Clinical Neurology*, vol. 160, pp. 171–183, 2019.

[45] A. Sharmila and P. Geethanjali, "A review on the pattern detection methods for epilepsy seizure detection from EEG signals," *Biomedical Engineering / Biomedizinische Technik*, vol. 64, no. 5, pp. 507–517, Oct. 2019.

[46] S. Supriya, S. Siuly, H. Wang, and Y. Zhang, "Automated epilepsy detection techniques from electroencephalogram signals: A review study," *Health Information Science and Systems*, vol. 8, no. 1, p. 33, Oct. 2020.

[47] M. K. Siddiqui, R. Morales-Menendez, X. Huang, and N. Hussain, "A review of epileptic seizure detection using machine learning classifiers," *Brain Informatics*, vol. 7, no. 1, p. 5, May 2020.

[48] C. Alvarado-Rojas, M. Valderrama, A. Fouad-Ahmed, H. Feldwisch-Drentrup, M. Ihle, C. A. Teixeira, F. Sales, A. Schulze-Bonhage, C. Adam, A. Dourado, S. Charpier, V. Navarro, and M. Le Van Quyen, "Slow modulations of high-frequency activity (40–140 Hz) discriminate preictal changes in human focal epilepsy," *Scientific Reports*, vol. 4, no. 1, p. 4545, Apr. 2014.

[49] S. Blanco, A. Garay, and D. Coulombie, "Comparison of Frequency Bands Using Spectral Entropy for Epileptic Seizure Prediction," *ISRN Neurology*, vol. 2013, p. e287327, May 2013.

[50] J. Li, J. Yan, X. Liu, and G. Ouyang, "Using Permutation Entropy to Measure the Changes in EEG Signals During Absence Seizures," *Entropy*, vol. 16, no. 6, pp. 3049–3061, Jun. 2014.

[51] A. S. Carmo, "EpiBOX: A Novel Approach to Long-Term Data Acquisition with Automatic Seizure Detection in Epilepsy," MSc., Instituto Superior Técnico, Universidade de Lisboa, Jan. 2021. [Online]. Available: https://fenix.tecnico.ulisboa.pt/cursos/mebiom/dissertacao/846778572212828

[52] P. Bota, E. Flety, H. P. da Silva, and A. Fred, "EmotiphAI: A biocybernetic engine for real-time biosignals acquisition in a collective setting," *Neural Computing and Applications*, Mar. 2022.

[53] L. Montenegro, M. Abreu, A. Fred, and J. M. Machado, "Human-Assisted vs. Deep Learning Feature Extraction: An Evaluation of ECG Features Extraction Methods for Arrhythmia Classification Using Machine Learning," *Applied Sciences*, vol. 12, no. 15, p. 7404, Jan. 2022.

[54] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and Y. Huang, "Pedestrian Stride-Length Estimation Based on LSTM and Denoising Autoencoders," *Sensors*, vol. 19, no. 4, p. 840, Feb. 2019.

[55] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Zhang, "Locomotion Activity Recognition Using Stacked Denoising Autoencoders," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2085–2093, Jun. 2018.

[56] P. Wang, B. Hou, S. Shao, and R. Yan, "ECG Arrhythmias Detection Using Auxiliary Classifier Generative Adversarial Network and Residual Network," *IEEE ACCESS*, vol. 7, pp. 100 910–100 922, 2019.

[57] L. Rodrigues, "Driver Drowsiness Detection with Peripheral Cardiac Signals," MSc. Thesis, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, 2021.

[58] S. Kusmakar, C. K. Karmakar, B. Yan, T. J.O'Brien, R. Muthuganapathy, and M. Palaniswami,

"Automated Detection of Convulsive Seizures Using a Wearable Accelerometer Device," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 2, pp. 421–432, Feb. 2019.

[59] A. Jahanbekam, J. Baumann, R. D. Nass, C. Bauckhage, H. Hill, C. E. Elger, and R. Surges, "Performance of ECG-based seizure detection algorithms strongly depends on training and test conditions," *EPILEPSIA OPEN*, vol. 6, no. 3, pp. 597–606, Sep. 2021.

[60] H. E. Scharfman, "The Neurobiology of Epilepsy," *Current neurology and neuroscience reports*, vol. 7, no. 4, pp. 348–354, Jul. 2007. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2492886/

[61] A. Pitkänen, K. Lukasiuk, F. E. Dudek, and K. J. Staley, "Epileptogenesis," *Cold Spring Harbor Perspectives in Medicine*, vol. 5, no. 10, p. a022822, Oct. 2015.

[62] G. J. Tortora and B. H. Derrickson, *Introduction to the Human Body*, 10th ed. Hoboken, NJ: John Wiley & Sons, Apr. 2015.

[63] S. R. Cobb, E. H. Buhl, K. Halasy, O. Paulsen, and P. Somogyi, "Synchronization of neuronal activity in hippocampus by individual GABAergic interneurons," *Nature*, vol. 378, no. 6552, pp. 75–78, Nov. 1995.

[64] H. Matsumoto and C. A. Marsan, "CORTICAL CELLULAR PHENOMENA IN EXPERIMENTAL EPILEPSY: ICTAL MANIFESTATIONS," *Experimental Neurology*, vol. 9, pp. 305–326, Apr. 1964.

[65] D. Johnston and T. H. Brown, "The synaptic nature of the paroxysmal depolarizing shift in hippocampal neurons," *Annals of Neurology*, vol. 16 Suppl, pp. S65–71, 1984.

[66] R. D. Traub, H. Michelson-Law, A. E. J. Bibbig, E. H. Buhl, and M. A. Whittington, "Gap junctions, fast oscillations and the initiation of seizures," *Advances in Experimental Medicine and Biology*, vol. 548, pp. 110–122, 2004.

[67] T. P. Sutula and F. E. Dudek, "Unmasking recurrent excitation generated by mossy fiber sprouting in the epileptic dentate gyrus: An emergent property of a complex system," in *Progress in Brain Research*. Elsevier, 2007, vol. 163, pp. 541–563.

[68] R. S. Sloviter, C. A. Zappone, B. D. Harvey, and M. Frotscher, "Kainic acid-induced recurrent mossy fiber innervation of dentate gyrus inhibitory interneurons: Possible anatomical substrate of granule cell hyper-inhibition in chronically epileptic rats," *The Journal of Comparative Neurology*, vol. 494, no. 6, pp. 944–960, Feb. 2006.

[69] D. P. A. Pitkänen, Paul Buckmaster, A. S. Galanopoulou, and S. L. Mosh??, *Models of Seizures and Epilepsy*, 2017.

[70] S. Cui, L. Duan, Y. Qiao, and Y. Xiao, "Learning EEG synchronization patterns for epileptic seizure prediction using bag-of-wave features," *Journal of Ambient Intelligence and Humanized Computing*, Sep. 2018.

[71] R. S. Fisher, J. H. Cross, J. A. French, N. Higurashi, E. Hirsch, F. E. Jansen, L. Lagae, S. L. Moshé, J. Peltola, E. Roulet Perez, I. E. Scheffer, and S. M. Zuberi, "Operational classification of seizure types by the International League Against Epilepsy: Position Paper of the ILAE Commission for Classification and Terminology," *Epilepsia*, vol. 58, no. 4, pp. 522–530, 2017.

[72] O. Devinsky, A. Vezzani, T. J. O'Brien, N. Jette, I. E. Scheffer, M. de Curtis, and P. Perucca,

"Epilepsy," *Nature Reviews Disease Primers*, vol. 4, no. 1, pp. 1–24, May 2018.

[73] W. T. Blume, H. O. Lüders, E. Mizrahi, C. Tassinari, W. Van Emde Boas, and J. Engel Jr., Ex-officio, "Glossary of Descriptive Terminology for Ictal Semiology: Report of the ILAE Task Force on Classification and Terminology," *Epilepsia*, vol. 42, no. 9, pp. 1212–1218, 2001.

[74] I. Conradsen, M. Moldovan, P. Jennum, P. Wolf, D. Farina, and S. Beniczky, "Dynamics of muscle activation during tonic-clonic seizures," *Epilepsy Research*, vol. 104, no. 1-2, pp. 84–93, Mar. 2013.

[75] J. Saraiva, "Glossary of Epilepsy," Jan. 2022. [Online]. Available: http://dx.doi.org/10.13140/RG.2.2.12128.76801

[76] R. D. Thijs, P. Ryvlin, and R. Surges, "Autonomic manifestations of epilepsy: Emerging pathways to sudden death?" *Nature Reviews Neurology*, vol. 17, no. 12, pp. 774–788, Dec. 2021.

[77] C. Baumgartner, S. Lurger, and F. Leutmezer, "Autonomic symptoms during epileptic seizures," *Epileptic Disorders*, vol. 3, no. 3, pp. 103–16, Oct. 2001. [Online]. Available: http://www.jle.com/fr/revues/epd/e-docs/autonomic_symptoms_during_epileptic_seizures_110095/article.phtml?tab=texte

[78] F. Leutmezer, C. Schernthaner, S. Lurger, K. Pötzelberger, and C. Baumgartner, "Electrocardiographic changes at the onset of epileptic seizures," *Epilepsia*, vol. 44, no. 3, pp. 348–354, 2003.

[79] R. Massetani, G. Strata, R. Galli, S. Gori, C. Gneri, U. Limbruno, D. Santo, M. Mariani, and L. Murri, "Alteration of cardiac function in patients with temporal lobe epilepsy: Different roles of EEG-ECG monitoring and spectral analysis of RR variability," *Epilepsia*, vol. 38, no. 3, pp. 363–369, 1997.

[80] A. Afif, R. Bouvier, A. Buenerd, J. Trouillas, and P. Mertens, "Development of the human fetal insular cortex: Study of the gyration from 13 to 28 gestational weeks," *Brain Structure and Function*, vol. 212, no. 3, pp. 335–346, Dec. 2007.

[81] R. Freeman and S. C. Schachter, "Autonomic epilepsy," *Seminars in Neurology*, vol. 15, no. 2, pp. 158–166, Jun. 1995.

[82] C. A. Galimberti, E. Marchioni, F. Barzizza, R. Manni, I. Sartori, and A. Tartara, "Partial Epileptic Seizures of Different Origin Variably Affect Cardiac Rhythm," *Epilepsia*, vol. 37, no. 8, pp. 742–747, 1996.

[83] H. Mayer, F. Benninger, L. Urak, B. Plattner, J. Geldner, and M. Feucht, "EKG abnormalities in children and adolescents with symptomatic temporal lobe epilepsy," *Neurology*, vol. 63, no. 2, pp. 324–328, Jul. 2004.

[84] G. Di Gennaro, P. Quarato, F. Sebastiano, V. Esposito, P. Onorati, L. Grammaldo, G. Meldolesi, A. Mascia, C. Falco, C. Scoppetta, F. Eusebi, M. Manfredi, and G. Cantore, "Ictal heart rate increase precedes EEG discharge in drug-resistant mesial temporal lobe seizures," *Clinical Neurophysiology*, vol. 115, no. 5, pp. 1169–1177, 2004.

[85] M. van der Lende, R. Surges, J. W. Sander, and R. D. Thijs, "Cardiac arrhythmias during or after epileptic seizures," *Journal of Neurology, Neurosurgery, and Psychiatry*, vol. 87, no. 1, pp. 69–74, Jan. 2016.

[86] L. D. Blumhardt, P. E. Smith, and L. Owen, "Electrocardiographic accompaniments of temporal

lobe epileptic seizures," *Lancet (London, England)*, vol. 1, no. 8489, pp. 1051–1056, May 1986.

[87] M. J. Keilson, W. A. Hauser, and J. P. Magrill, "Electrocardiographic changes during electrographic seizures," *Archives of Neurology*, vol. 46, no. 11, pp. 1169–1170, Nov. 1989.

[88] C. Schernthaner, G. Lindinger, K. Pötzelberger, K. Zeiler, and C. Baumgartner, "Autonomic epilepsy–the influence of epileptic discharges on heart rate and rhythm," *Wiener Klinische Wochenschrift*, vol. 111, no. 10, pp. 392–401, May 1999.

[89] R. Surges, C. A. Scott, and M. C. Walker, "Enhanced QT shortening and persistent tachycardia after generalized seizures," *Neurology*, vol. 74, no. 5, pp. 421–426, Feb. 2010.

[90] W. Chen, C.-L. Guo, P.-S. Zhang, C. Liu, H. Qiao, J.-G. Zhang, and F.-G. Meng, "Heart rate changes in partial seizures: Analysis of influencing factors among refractory patients," *BMC Neurology*, vol. 14, no. 1, p. 135, Jun. 2014.

[91] S. M. Hilton and A. W. Zbrozyna, "Amygdaloid region for defence reactions and its efferent pathway to the brain stem," *The Journal of Physiology*, vol. 165, pp. 160–173, Jan. 1963.

[92] A. J. Gelsema, N. E. Copeland, G. Drolet, and H. Bachelard, "Cardiovascular effects of neuronal activation of the extended amygdala in rats," *Brain Research*, vol. 626, no. 1-2, pp. 156–166, Oct. 1993.

[93] M. J. Keilson, W. A. Hauser, J. P. Magrill, and M. Goldman, "ECG abnormalities in patients with epilepsy," *Neurology*, vol. 37, no. 10, pp. 1624–1626, Oct. 1987.

[94] K. Jansen and L. Lagae, "Cardiac changes in epilepsy," *Seizure*, vol. 19, no. 8, pp. 455–460, Oct. 2010.

[95] C. P. Monté, C. J. Monté, P. Boon, and J. Arends, "Epileptic seizures associated with syncope: Ictal bradycardia and ictal asystole," *Epilepsy & Behavior: E&B*, vol. 90, pp. 168–171, Jan. 2019.

[96] A. L. Reeves, "Autonomic activity in epilepsy: Diagnostic considerations and implications," *Journal of Epilepsy*, vol. 10, no. 3, pp. 111–116, May 1997.

[97] M. M. Mesulam and E. J. Mufson, "Insula of the old world monkey. I. Architectonics in the insulo-orbito-temporal component of the paralimbic brain," *The Journal of Comparative Neurology*, vol. 212, no. 1, pp. 1–22, Nov. 1982.

[98] C. Munari, L. Tassi, M. Di Leo, P. Kahane, D. Hoffmann, S. Francione, and P. Quarato, "Video-stereo-electroencephalographic investigation of orbitofrontal cortex. Ictal electroclinical patterns," *Advances in Neurology*, vol. 66, pp. 273–295, 1995.

[99] S. U. Schuele, A. C. Bermeo, A. V. Alexopoulos, E. R. Locatelli, R. C. Burgess, D. S. Dinner, and N. Foldvary-Schaefer, "Video-electrographic and clinical features in patients with ictal asystole," *Neurology*, vol. 69, no. 5, pp. 434–441, Jul. 2007.

[100] P. E. Smith, S. J. Howell, L. Owen, and L. D. Blumhardt, "Profiles of instant heart rate during partial seizures," *Electroencephalography and Clinical Neurophysiology*, vol. 72, no. 3, pp. 207–217, Mar. 1989.

[101] E. G. Gutierrez, N. E. Crone, J. Y. Kang, Y. I. Carmenate, and G. L. Krauss, "Strategies for non-EEG seizure detection and timing for alerting and interventions with tonic–clonic seizures," *Epilepsia*, vol. 59, no. S1, pp. 36–41, 2018.

[102] M. Nei, “Cardiac Effects of Seizures,” *Epilepsy Currents*, vol. 9, no. 4, pp. 91–95, 2009.

[103] M. Nei, R. T. Ho, and M. R. Sperling, “EKG abnormalities during partial seizures in refractory epilepsy,” *Epilepsia*, vol. 41, no. 5, pp. 542–548, May 2000.

[104] N. T. Mathew, G. M. Taori, K. V. Mathai, and J. Chandy, “Atrial fibrillation associated with seizure in a case of frontal meningioma,” *Neurology*, vol. 20, no. 7, pp. 725–728, Jul. 1970.

[105] B. Blum, N. Kauli, E. Liban, and P. Levy, “Paroxysms of T-wave alterations in the ECG in experimental epilepsy due to foci in the pseudosylvian gyrus,” *Life Sciences*, vol. 9, no. 4, pp. 219–225, Feb. 1970.

[106] S. Tigaran, V. Rasmussen, M. Dam, S. Pedersen, H. Høgenhaven, and B. Friberg, “ECG changes in epilepsy patients,” *Acta Neurologica Scandinavica*, vol. 96, no. 2, pp. 72–75, Aug. 1997.

[107] C. Opherk, J. Coromilas, and L. J. Hirsch, “Heart rate and EKG changes in 102 seizures: Analysis of influencing factors,” *Epilepsy Research*, vol. 52, no. 2, pp. 117–127, Dec. 2002.

[108] E. Kolsal, A. Serdaroğlu, E. Çilsal, S. Kula, A. Ş. Soysal, A. N. Ç. Kurt, and E. Arhan, “Can heart rate variability in children with epilepsy be used to predict seizures?” *Seizure*, vol. 23, no. 5, pp. 357–362, May 2014.

[109] T. Page and F. J. Rugg-Gunn, “Bitemporal seizure spread and its effect on autonomic dysfunction,” *Epilepsy & Behavior: E&B*, vol. 84, pp. 166–172, Jul. 2018.

[110] M. S. Berilgen, T. Sari, S. Bulut, and B. Mungen, “Effects of epilepsy on autonomic nervous system and respiratory function tests,” *Epilepsy & Behavior: E&B*, vol. 5, no. 4, pp. 513–516, Aug. 2004.

[111] R. El Atrache, E. Tamilia, F. Mohammadpour Touserkani, S. Hammond, C. Papadelis, K. Kapur, M. Jackson, B. Bucciarelli, M. Tsuboyama, R. A. Sarkis, and T. Loddenkemper, “Photoplethysmography: A measure for the function of the autonomic nervous system in focal impaired awareness seizures,” *Epilepsia*, vol. 61, no. 8, pp. 1617–1626, 2020.

[112] H. Evrengül, H. Tanriverdi, D. Dursunoglu, A. Kaftan, O. Kuru, U. Unlu, and M. Kilic, “Time and frequency domain analyses of heart rate variability in patients with epilepsy,” *Epilepsy Research*, vol. 63, no. 2, pp. 131–139, Feb. 2005.

[113] T. D. Pang, B. D. Nearing, K. B. Krishnamurthy, B. Olin, S. C. Schachter, and R. L. Verrier, “Cardiac electrical instability in newly diagnosed/chronic epilepsy tracked by Holter and ECG patch,” *Neurology*, vol. 93, no. 10, pp. 450–458, Sep. 2019.

[114] P. G. Kanmani Prince and R. Hemamalini, “Detection of Variations in HRV Using Discrete Wavelet Transform for Seizure Detection Application,” in *Computer Applications for Communication, Networking, and Digital Contents*, T.-h. Kim, D.-s. Ko, T. Vasilakos, A. Stoica, and J. Abawajy, Eds., vol. 350. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 342–348.

[115] E. Tobaldini, L. Nobili, S. Strada, K. R. Casali, A. Braghiroli, and N. Montano, “Heart rate variability in normal and pathological sleep,” *Frontiers in Physiology*, vol. 4, p. 294, Oct. 2013.

[116] N. Foldvary, N. Lee, G. Thwaites, E. Mascha, J. Hammel, H. Kim, A. H. Friedman, and R. A. Radtke, “Clinical and electrographic manifestations of lesional neocortical temporal lobe epilepsy,” *Neurology*, vol. 49, no. 3, pp. 757–763, Sep. 1997.

[117] A. S. Harvey, I. J. Hopkins, J. M. Bowe, D. J. Cook, L. K. Shield, and S. F. Berkovic, “Frontal

lobe epilepsy: Clinical seizure characteristics and localization with ictal 99mTc-HMPAO SPECT," *Neurology*, vol. 43, no. 10, pp. 1966–1980, Oct. 1993.

[118] S. Sivathamboo, T. N. Constantino, Z. Chen, P. B. Sparks, J. Goldin, D. Velakoulis, N. C. Jones, P. Kwan, V. G. Macefield, T. J. O'Brien, and P. Perucca, "Cardiorespiratory and autonomic function in epileptic seizures: A video-EEG monitoring study," *Epilepsy & Behavior: E&B*, vol. 111, p. 107271, Oct. 2020.

[119] A. S. Blum, J. R. Ives, A. L. Goldberger, I. C. Al-Aweel, K. B. Krishnamurthy, F. W. Drislane, and D. L. Schomer, "Oxygen desaturations triggered by partial seizures: Implications for cardiopulmonary instability in epilepsy," *Epilepsia*, vol. 41, no. 5, pp. 536–541, May 2000.

[120] B. J. Dlouhy, B. K. Gehlbach, C. J. Kreple, H. Kawasaki, H. Oya, C. Buzza, M. A. Granner, M. J. Welsh, M. A. Howard, J. A. Wemmie, and G. B. Richerson, "Breathing Inhibited When Seizures Spread to the Amygdala and upon Amygdala Stimulation," *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, vol. 35, no. 28, pp. 10 281–10 289, Jul. 2015.

[121] L. M. Bateman, M. Spitz, and M. Seyal, "Ictal hypoventilation contributes to cardiac arrhythmia and SUDEP: Report on two deaths in video-EEG–monitored patients," *Epilepsia*, vol. 51, no. 5, pp. 916–920, 2010.

[122] N. J. Azar, T. F. Tayah, L. Wang, Y. Song, and B. W. Abou-Khalil, "Postictal Breathing Pattern Distinguishes Epileptic from Nonepileptic Convulsive Seizures," *Epilepsia*, vol. 49, no. 1, pp. 132–137, 2008.

[123] M. Seyal, L. M. Bateman, and C.-S. Li, "Impact of periictal interventions on respiratory dysfunction, postictal EEG suppression, and postictal immobility: *Periictal Intervention and Hypoxemia*," *Epilepsia*, vol. 54, no. 2, pp. 377–382, Feb. 2013.

[124] B. D. Moseley, K. Nickels, J. Britton, and E. Wirrell, "How common is ictal hypoxemia and bradycardia in children with partial complex and generalized convulsive seizures?" *Epilepsia*, vol. 51, no. 7, pp. 1219–1224, Jul. 2010.

[125] C. A. Massey, L. P. Sowers, B. J. Dlouhy, and G. B. Richerson, "Mechanisms of sudden unexpected death in epilepsy: The pathway to prevention," *Nature Reviews. Neurology*, vol. 10, no. 5, pp. 271–282, May 2014.

[126] D. W. Mulder, D. Daly, and A. A. Bailey, "Visceral epilepsy," *A.M.A. Archives of Internal Medicine*, vol. 93, no. 4, pp. 481–493, Apr. 1954.

[127] J. M. Van Buren, "The abdominal aura. A study of abdominal sensations occurring in epilepsy and produced by depth stimulation," *Electroencephalography and Clinical Neurophysiology*, vol. 15, pp. 1–19, Feb. 1963.

[128] C. J. Klein, M. H. Silber, J. R. Halliwill, S. A. Schreiner, G. A. Suarez, and P. A. Low, "Basal forebrain malformation with hyperhidrosis and hypothermia: Variant of Shapiro's syndrome," *Neurology*, vol. 56, no. 2, pp. 254–256, Jan. 2001.

[129] M. H. Mendler, D. Sautereau, B. Pillegand, and R. Ravon, "[A case of digestive epilepsy with late diagnosis: a disease not to be disregarded]," *Gastroenterologie Clinique Et Biologique*, vol. 22, no. 2, pp. 235–239, Feb. 1998.

[130] V. Cortés, L. Landete, E. Gómez, and R. Blasco, "[Partial simple vegetative crisis: importance of electroencephalographic findings]," *Revista De Neurologia*, vol. 25, no. 148, pp. 1931–1933, Dec. 1997.

[131] S. Vieluf, M. Amengual-Gual, B. Zhang, R. El Atrache, C. Ufongene, M. C. Jackson, S. Branch, C. Reinsberger, and T. Loddenkemper, "Twenty-four-hour patterns in electrodermal activity recordings of patients with and without epileptic seizures," *Epilepsia*, vol. 62, no. 4, pp. 960–972, 2021.

[132] H. Yu, D. Yen, C. Yiu, W. Chung, J. Lirng, and M. Su, "Pilomotor seizures," *European Neurology*, vol. 40, no. 1, pp. 19–21, Jul. 1998.

[133] C. Scoppetta, C. Casali, S. D'Agostini, G. Amabile, and C. Morocutti, "Pilomotor epilepsy," *Functional Neurology*, vol. 4, no. 3, pp. 283–286, 1989 Jul-Sep.

[134] G. L. Ahern, G. F. Howard, and K. L. Weiss, "Posttraumatic pilomotor seizures: A case report," *Epilepsia*, vol. 29, no. 5, pp. 640–643, 1988 Sep-Oct.

[135] F. Tyndel, N. Bayer, and H. Preiksaitis, "An additional pilomotor seizure," *Neurology*, vol. 36, no. 2, p. 305, Feb. 1986.

[136] B. R. Tharp, "Orbital frontial seizures. An unique electroencephalographic and clinical syndrome," *Epilepsia*, vol. 13, no. 5, pp. 627–642, Sep. 1972.

[137] S. Beniczky, I. Conradsen, R. Pressler, and P. Wolf, "Quantitative analysis of surface electromyography: Biomarkers for convulsive seizures," *Clinical Neurophysiology*, vol. 127, no. 8, pp. 2900–2907, Aug. 2016.

[138] I. Conradsen, P. Wolf, T. Sams, H. B. D. Sorensen, and S. Beniczky, "Patterns of muscle activation during generalized tonic and tonic-clonic epileptic seizures," *Epilepsia*, vol. 52, no. 11, pp. 2125–2132, Nov. 2011.

[139] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016.

[140] S. J. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A Modern Approach*, 3rd ed., ser. Prentice Hall Series in Artificial Intelligence. Upper Saddle River: Prentice Hall, 2010.

[141] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[142] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.

[143] W. J. Bosl, A. Leviton, and T. Loddenkemper, "Prediction of Seizure Recurrence. A Note of Caution," *Frontiers in Neurology*, vol. 12, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fneur.2021.675728

[144] A. Zheng, *Evaluating Machine Learning Models*, 1st ed. O'Reilly Media, Inc., 2015.

[145] S. Beniczky and J. Jeppesen, "Non-electroencephalography-based seizure detection," *Current Opinion in Neurology*, vol. 32, no. 2, pp. 198–204, Apr. 2019.

[146] S. Rheims, "Wearable devices for seizure detection: Is it time to translate into our clinical practice?" *Revue Neurologique*, vol. 176, no. 6, pp. 480–484, Jun. 2020.

[147] M. Abreu, A. L. N. Fred, H. P. D. Silva, and C. Wang, "From Seizure Detection to Prediction: A Review of Wearables and Related Devices Applicable to Epilepsy via Peripheral Measurements,"

2020.

[148] S. Beniczky and P. Ryvlin, “Standards for testing and clinical validation of seizure detection devices,” *Epilepsia*, vol. 59, no. S1, pp. 9–13, 2018.

[149] “ePatch device.” [Online]. Available: https://www.myheartmonitor.com/device/epatch/

[150] J. Jeppesen, A. Fuglsang-Frederiksen, P. Johansen, J. Christensen, S. Wüstenhagen, H. Tankisi, E. Qerama, and S. Beniczky, “Seizure detection using heart rate variability: A prospective validation study,” *Epilepsia*, vol. 61, no. S1, pp. S41–S46, 2020.

[151] J. van Andel, C. Ungureanu, J. Arends, F. Tan, J. Van Dijk, G. Petkov, S. Kalitzin, T. Gutter, A. de Weerd, B. Vledder, R. Thijs, G. van Thiel, K. Roes, and F. Leijten, “Multimodal, automated detection of nocturnal motor seizures at home: Is a reliable seizure detector feasible?” *Epilepsia Open*, vol. 2, no. 4, pp. 424–431, 2017.

[152] I. Osorio, “AUTOMATED SEIZURE DETECTION USING EKG,” *INTERNATIONAL JOURNAL OF NEURAL SYSTEMS*, vol. 24, no. 2, Mar. 2014.

[153] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, and A. Dourado, “Prediction of epileptic seizures based on heart rate variability,” *Technology and Health Care*, vol. 24, no. 6, pp. 795–810, Jan. 2016.

[154] T. De Cooman, A. Van de Vel, B. Ceulemans, L. Lagae, W. Van Paesschen, B. Vanrumste, and S. Van Huffel, “Estimation of the Maximal Heart Rate to Improve Online Tonic-Clonic Seizure Detection using ECG,” in *2015 COMPUTING IN CARDIOLOGY CONFERENCE (CINC)*, A. Murray, Ed., vol. 42, 2015, pp. 977–980.

[155] L. Billeci, D. Marino, L. Insana, G. Vatti, and M. Varanini, “Patient-specific seizure prediction based on heart rate variability and recurrence quantification analysis,” *PLOS ONE*, vol. 13, no. 9, Sep. 2018.

[156] J. Pavei, R. G. Heinzen, B. Novakova, R. Walz, A. J. Serra, M. Reuber, A. Ponnusamy, and J. L. B. Marques, “Early Seizure Detection Based on Cardiac Autonomic Regulation Dynamics,” *FRONTIERS IN PHYSIOLOGY*, vol. 8, Oct. 2017.

[157] T. De Cooman, C. Varon, B. Hunyadi, W. Van Paesschen, L. Lagae, and S. Van Huffel, “Online Automated Seizure Detection in Temporal Lobe Epilepsy Patients Using Single-lead ECG,” *INTERNATIONAL JOURNAL OF NEURAL SYSTEMS*, vol. 27, no. 7, Nov. 2017.

[158] A. Popov, O. Panichev, Y. Karplyuk, Y. Smirnov, S. Zaunseder, V. Kharytonov, and IEEE, “Heart Beat-to-Beat Intervals Classification for Epileptic Seizure Prediction,” in *2017 SIGNAL PROCESSING SYMPOSIUM (SPSYMPO)*, 2017.

[159] Y. Smirnov, A. Popov, O. Panichev, Y. Karplyuk, V. Kharytonov, and IEEE, “Epileptic seizure prediction based on singular value decomposition of heart rate variability features,” in *2017 SIGNAL PROCESSING SYMPOSIUM (SPSYMPO)*, 2017.

[160] C. Varon, K. Jansen, L. Lagae, S. Van Huffel, and IEEE, “Detection of Epileptic Seizures by Means of Morphological Changes in the ECG,” in *2013 COMPUTING IN CARDIOLOGY CONFERENCE (CINC)*, vol. 40. Computing in Cardiology, 2013, pp. 863–866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6713514

[161] T. De Cooman, T. W. Kjaer, S. Van Huffel, and H. B. Sorensen, “Adaptive heart rate-based epileptic seizure detection using real-time user feedback,” *PHYSIOLOGICAL MEASUREMENT*, vol. 39, no. 1, Jan. 2018.

[162] T. De Cooman, K. Vandecasteele, C. Varon, B. Hunyadi, E. Cleeren, W. Van Paesschen, and S. Van Huffel, “Personalizing Heart Rate-Based Seizure Detection Using Supervised SVM Transfer Learning,” *FRONTIERS IN NEUROLOGY*, vol. 11, Feb. 2020.

[163] L. Billeci, A. Tonacci, D. Marino, L. Insana, G. Vatti, and M. Varanini, “Machine Learning Approach for Epileptic Seizure Prediction and Early Intervention,” in *CONVERGING CLINICAL AND ENGINEERING RESEARCH ON NEUROREHABILITATION III*, L. Masia, S. Micera, M. Akay, and J. Pons, Eds., vol. 21, 2019, pp. 972–976.

[164] K. Vandecasteele, T. De Cooman, Y. Gu, E. Cleeren, K. Claes, W. V. Paesschen, S. V. Huffel, and B. Hunyadi, “Automated Epileptic Seizure Detection Based on Wearable ECG and PPG in a Hospital Environment,” *Sensors*, vol. 17, no. 10, p. 2338, Oct. 2017.

[165] T. D. Cooman, C. Varon, A. V. de Vel, K. Jansen, B. Ceulemans, L. Lagae, and S. V. Huffel, “Adaptive nocturnal seizure detection using heart rate and low-complexity novelty detection,” *Seizure - European Journal of Epilepsy*, vol. 59, pp. 48–53, Jul. 2018.

[166] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, C. A. Teixeira, and A. Dourado, “A new algorithm for detection of epileptic seizures based on HRV signal,” *JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE*, vol. 26, no. 2, pp. 251–265, Apr. 2014.

[167] J. Jeppesen, A. Fuglsang-Frederiksen, P. Johansen, J. Christensen, S. Wüstenhagen, H. Tankisi, E. Qerama, A. Hess, and S. Beniczky, “Seizure detection based on heart rate variability using a wearable electrocardiography device,” *Epilepsia*, vol. 60, no. 10, pp. 2105–2113, 2019.

[168] C. Varon, A. Caicedo, K. Jansen, L. Lagae, S. Van Huffel, and IEEE, “Detection of epileptic seizures from single lead ECG by means of phase rectified signal averaging,” in *2014 36TH ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC)*, 2014, pp. 3789–3792.

[169] C. Varon, K. Jansen, L. Lagae, and S. Van Huffel, “Can ECG monitoring identify seizures?” *JOURNAL OF ELECTROCARDIOLOGY*, vol. 48, no. 6, pp. 1069–1074, 2015 NOV-DEC.

[170] K. Vandecasteele, T. D. Cooman, C. Chatzichristos, E. Cleeren, L. Swinnen, J. M. Ortiz, S. V. Huffel, M. Dümpelmann, A. Schulze-Bonhage, M. D. Vos, W. V. Paesschen, and B. Hunyadi, “The power of ECG in multimodal patient-specific seizure monitoring: Added value to an EEG-based detector using limited channels,” *Epilepsia*, vol. 62, no. 10, Jul. 2021.

[171] Y. Liu, S. Sivathamboo, P. Goodin, P. Bonnington, P. Kwan, L. Kuhlmann, T. O’Brien, P. Perucca, Z. Ge, and Assoc Comp Machinery, “Epileptic Seizure Detection Using Convolutional Neural Network: A Multi-Biosignal study,” in *PROCEEDINGS OF THE AUSTRALASIAN COMPUTER SCIENCE WEEK MULTICONFERENCE (ACSW 2020)*, 2020.

[172] F. Fuerbass, S. Kampusch, E. Kaniusas, J. Koren, S. Pirker, R. Hopfengaertner, H. Stefan, T. Kluge, and C. Baumgartner, “Automatic multimodal detection for long-term seizure documentation in epilepsy,” *CLINICAL NEUROPHYSIOLOGY*, vol. 128, no. 8, pp. 1466–1472, Aug. 2017.

[173] H. Khamis, A. Mohamed, and S. Simpson, “Seizure state detection of temporal lobe seizures by autoregressive spectral analysis of scalp EEG,” *CLINICAL NEUROPHYSIOLOGY*, vol. 120, no. 8, pp. 1479–1488, Aug. 2009.

[174] M. M. Hartmann, F. Fuerbass, H. Perko, A. Skupch, K. Lackmayer, C. Baumgartner, T. Kluge, and IEEE, “EpiScan: Online seizure detection for epilepsy monitoring units,” in *2011 ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC)*, 2011, pp. 6096–6099.

[175] H. Khamis, A. Mohamed, and S. Simpson, “Frequency-moment signatures: A method for automated seizure detection from scalp EEG,” *CLINICAL NEUROPHYSIOLOGY*, vol. 124, no. 12, pp. 2317–2327, Dec. 2013.

[176] R. Hopfengaertner, F. Kerling, V. Bauer, and H. Stefan, “An efficient, robust and fast method for the offline detection of epileptic seizures in long-term scalp EEG recordings,” *CLINICAL NEUROPHYSIOLOGY*, vol. 118, no. 11, pp. 2332–2343, Nov. 2007.

[177] S. Raghu, N. Sriraam, G. P. Kumar, and A. S. Hegde, “A Novel Approach for Real-Time Recognition of Epileptic Seizures Using Minimum Variance Modified Fuzzy Entropy,” *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 65, no. 11, pp. 2612–2621, Nov. 2018.

[178] M. Ruiz Marin, I. Villegas Martinez, G. Rodriguez Bermudez, and M. Porfiri, “Integrating old and new complexity measures toward automated seizure detection from long-term video EEG recordings,” *ISCIENCE*, vol. 24, no. 1, Jan. 2021.

[179] A. S. Zandi, M. Javidan, G. A. Dumont, and R. Tafreshi, “Automated Real-Time Epileptic Seizure Detection in Scalp EEG Recordings Using an Algorithm Based on Wavelet Packet Transform,” *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 57, no. 7, pp. 1639–1651, Jul. 2010.

[180] A. S. Zandi, G. A. Dumont, M. Javidan, and R. Tafreshi, “Detection of Epileptic Seizures in Scalp Electroencephalogram: An Automated Real-Time Wavelet-Based Approach,” *JOURNAL OF CLINICAL NEUROPHYSIOLOGY*, vol. 29, no. 1, pp. 1–16, Feb. 2012.

[181] A. Einizade, M. Mozafari, S. H. Sardouie, S. Nasiri, G. Clifford, and IEEE, “A Deep Learning-Based Method for Automatic Detection of Epileptic Seizure in a Dataset With Both Generalized and Focal Seizure Types,” in *2020 IEEE SIGNAL PROCESSING IN MEDICINE AND BIOLOGY SYMPOSIUM*, 2020.

[182] S. Wilson, “A neural network method for automatic and incremental learning applied to patient-dependent seizure detection,” *CLINICAL NEUROPHYSIOLOGY*, vol. 116, no. 8, pp. 1785–1795, Aug. 2005.

[183] A. Emami, N. Kunii, T. Matsuo, T. Shinozaki, K. Kawai, and H. Takahashi, “Seizure detection by convolutional neural network-based analysis of scalp electroencephalography plot images,” *NEUROIMAGE-CLINICAL*, vol. 22, 2019.

[184] M. Saqib, Y. Zhu, M. Wang, and B. Beaulieu-Jones, “Regularization of Deep Neural Networks for EEG Seizure Detection to Mitigate Overfitting,” in *2020 IEEE 44TH ANNUAL COMPUTERS, SOFTWARE, AND APPLICATIONS CONFERENCE (COMPSAC 2020)*, W. Chan, B. Claycomb,

H. Takakura, J. Yang, Y. Teranishi, D. Towey, S. Segura, H. Shahriar, S. Reisman, and S. Ahamed, Eds., 2020, pp. 664–673.

[185] E. B. Petersen, J. Duun-Henriksen, A. Mazzaretto, T. W. Kjaer, C. E. Thomsen, H. B. D. Sorensen, and IEEE, "Generic Single-Channel Detection of Absence Seizures," in *2011 ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC)*, 2011, pp. 4820–4823.

[186] S. Chen, X. Zhang, and IEEE, "An Automatic Seizure Detection Method using Multi-channel EEG Signals," in *2017 13TH IEEE CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING (CASE)*, 2017, pp. 94–95.

[187] S. Chen, X. Zhang, and Z. Yang, "Epileptic seizure detection by combining robust-principal component analysis and least square-support vector machine," *INTERNATIONAL JOURNAL OF IMAGING SYSTEMS AND TECHNOLOGY*, vol. 27, no. 4, pp. 368–375, Dec. 2017.

[188] T. Shoji, N. Yoshida, and T. Tanaka, "Automated detection of abnormalities from an EEG recording of epilepsy patients with a compact convolutional neural network," *BIOMEDICAL SIGNAL PROCESSING AND CONTROL*, vol. 70, Sep. 2021.

[189] Y. Yu, H. Qin, Y. Li, Z. Gao, Z. Gai, and IEEE, "EEG Absence Seizure Detection with Autocorrelation Function and Recurrent Neural Network," in *2019 IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE (IEEE SSCI 2019)*, 2019, pp. 3059–3064.

[190] R. Meier, H. Dittrich, A. Schulze-Bonhage, and A. Aertsen, "Detecting epileptic seizures in long-term human EEG: A new approach to automatic online and real-time detection and classification of polymorphic seizure patterns," *JOURNAL OF CLINICAL NEUROPHYSIOLOGY*, vol. 25, no. 3, pp. 119–131, Jun. 2008.

[191] A. S. Zandi, R. Tafreshi, M. Javidan, and G. A. Dumont, "Predicting Epileptic Seizures in Scalp EEG Based on a Variational Bayesian Gaussian Mixture Model of Zero-Crossing Intervals," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 60, no. 5, pp. 1401–1413, May 2013.

[192] G. R. Minasyan, J. B. Chatten, M. J. Chatten, and R. N. Harner, "Patient-Specific Early Seizure Detection From Scalp Electroencephalogram," *JOURNAL OF CLINICAL NEUROPHYSIOLOGY*, vol. 27, no. 3, pp. 163–178, Jun. 2010.

[193] P. Luckett, T. Watts, J. T. McDonald, L. Hively, R. Benton, and ASSOC COMP MACHINERY, "A Deep Learning Approach to Phase-Space Analysis for Seizure Detection," in *ACM-BCB'19: PROCEEDINGS OF THE 10TH ACM INTERNATIONAL CONFERENCE ON BIOINFORMATICS, COMPUTATIONAL BIOLOGY AND HEALTH INFORMATICS*, 2019, pp. 190–196.

[194] M. Shafiq and Z. Gu, "Deep Residual Learning for Image Recognition: A Survey," *Applied Sciences*, vol. 12, no. 18, p. 8972, Jan. 2022.

[195] Y. Gao, B. Gao, Q. Chen, J. Liu, and Y. Zhang, "Deep Convolutional Neural Network-Based Epileptic Electroencephalogram (EEG) Signal Classification," *FRONTIERS IN NEUROLOGY*, vol. 11, May 2020.

[196] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015.

[197] J. Li, S.-p. Pang, F. Xu, P. Ji, S. Zhou, and M. Shu, "Two-dimensional ECG-based cardiac arrhyth-

mia classification using DSE-ResNet," *Scientific Reports*, vol. 12, no. 1, p. 14485, Aug. 2022.

[198] S. Hong, Y. Xu, A. Khare, S. Priambada, K. Maher, A. Aljiffry, J. Sun, and A. Tumanov, "HOLMES: Health OnLine Model Ensemble Serving for Deep Learning Models in Intensive Care Units," Aug. 2020. [Online]. Available: http://arxiv.org/abs/2008.04063

[199] G. D. Clifford, J. Behar, Q. Li, and I. Rezek, "Signal quality indices and data fusion for determining clinical acceptability of electrocardiograms," *Physiological Measurement*, vol. 33, no. 9, pp. 1419–1433, Aug. 2012.

[200] Q. Li, R. G. Mark, and G. D. Clifford, "Robust heart rate estimation from multiple asynchronous noisy sources using signal quality indices and a Kalman filter," *Physiological Measurement*, vol. 29, no. 1, pp. 15–32, Dec. 2007.

[201] Q. Li, C. Rajagopalan, and G. D. Clifford, "A machine learning approach to multi-level ECG signal quality classification," *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, pp. 435–447, Dec. 2014.

[202] N. V. Thakor, J. G. Webster, and W. J. Tompkins, "Estimation of QRS Complex Power Spectra for Design of a QRS Filter," *IEEE Transactions on Biomedical Engineering*, vol. BME-31, no. 11, pp. 702–706, Nov. 1984.

[203] J. Saraiva, "Denoising and Artifact Removal of Peripheral Biosignals for Continuous Epileptic Seizure Monitoring on Wearable Devices," MSc. Thesis, Department of Bioengineering, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, 2022.

[204] F. Hatamian, N. Ravikumar, S. Vesal, F. Kemeth, M. Struck, A. Maier, and IEEE, "The Effect of Data Augmentation on Classification of Atrial Fibrillation in Short Single-Lead ECG Signals Using Deep Neural Networks," in *2020 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING*, 2020, pp. 1264–1268.

[205] N. Nonaka and J. Seita, "RandECG: Data Augmentation for Deep Neural Network Based ECG Classification," in *ADVANCES IN ARTIFICIAL INTELLIGENCE*, Y. Takama, N. Matsumura, K. Yada, M. Matsushita, D. Katagami, A. Abe, H. Kashima, T. Hiraoka, T. Uchiya, and R. Rzepka, Eds., vol. 1423, 2022, pp. 178–189.

[206] M. Cao, T. Zhao, Y. Li, W. Zhang, P. Benharash, and R. Ramezani, "ECG Heartbeat classification using deep transfer learning with Convolutional Neural Network and STFT technique," Jul. 2022. [Online]. Available: http://arxiv.org/abs/2206.14200

[207] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.01703

[208] H. Zhang, W. Zhao, and S. Liu, "SE-ECGNet: A Multi-scale Deep Residual Network with Squeeze-and-Excitation Module for ECG Signal Classification," Dec. 2020. [Online]. Available: http://arxiv.org/abs/2012.05510

[209] R. Brotherstone, A. McLellan, C. Graham, and K. Fisher, "A clinical evaluation of a novel algorithm in the reliable detection of epileptic seizures," *SEIZURE-EUROPEAN JOURNAL OF EPILEPSY*,

vol. 82, pp. 109–117, Nov. 2020.

[210] D. J. d. S. A. Belo, "Learning Biosignals with Deep Learning," doctoralThesis, Feb. 2021. [Online]. Available: https://run.unl.pt/handle/10362/126518

[211] T. Stuart, J. Hanna, and P. Gutruf, "Wearable devices for continuous monitoring of biosignals: Challenges and opportunities," *APL Bioengineering*, vol. 6, no. 2, p. 021502, Apr. 2022.

[212] H. P. da Silva, A. Fred, and R. Martins, "Biosignals for Everyone," *IEEE Pervasive Computing*, vol. 13, no. 4, pp. 64–71, Oct. 2014.

[213] QuantumBlack, "Global AI Survey 2021," Dec. 2021. [Online]. Available: http://ceros.mckinsey.com/global-ai-survey-2020-a-desktop-3-1

[214] I. Sommerville, *Software Engineering, 10th Edition*, 10th ed. Pearson India, 2018.

[215] H. P. da Silva, A. Lourenço, A. Fred, and R. Martins, "BIT: Biosignal Igniter Toolkit," *Computer Methods and Programs in Biomedicine*, vol. 115, no. 1, pp. 20–32, Jun. 2014.

[216] D. Makowski, T. Pham, Z. J. Lau, J. C. Brammer, F. Lespinasse, H. Pham, C. Schölzel, and S. H. A. Chen, "NeuroKit2: A Python toolbox for neurophysiological signal processing," *Behavior Research Methods*, vol. 53, no. 4, pp. 1689–1696, Aug. 2021.

[217] H. Stabenau, C. Bridge, and J. Waks, "ECGAug: A novel method of generating augmented annotated electrocardiogram QRST complexes and rhythm strips," *COMPUTERS IN BIOLOGY AND MEDICINE*, vol. 134, Jul. 2021.

[218] E. Adib, F. Afghah, and J. J. Prevost, "Arrhythmia Classification using CGAN-augmented ECG Signals," Aug. 2022. [Online]. Available: http://arxiv.org/abs/2202.00569

[219] H. Tung, C. Zheng, X. Mao, and D. Qian, "Multi-Lead ECG Classification via an Information-Based Attention Convolutional Neural Network," Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.12009

[220] N. Diamant, E. Reinertsen, S. Song, A. Aguirre, C. Stultz, and P. Batra, "Patient Contrastive Learning: A Performant, Expressive, and Practical Approach to ECG Modeling," *PLOS Computational Biology*, vol. 18, no. 2, p. e1009862, Feb. 2022.

Appendix A

ResNet and Decision Experiments

Here are presented the performance metrics Sensitivity, Specificity, Positive Predictive Value and F1-Score, in Tables A.1, A.2, A.3, and A.4, respectively, for each cross-validated test time series containing an unseen seizure. Figures A.1 and A.2 also show the TPs, TNs, FPs, and FNs of each test fold. Additionally, Tables A.5 and A.6 report the F1-Score and Number of False Alarms, respectively, of the decision algorithm on each test fold.

Table A.1: ResNet sensitivity on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0.4242	0.7674	0.7973	0.7274	0.4724	0.6575	0.6705	–	–	–	–
I1HP	0.7519	0.8589	0.7443	0.5142	0.4281	0.5348	0.8337	0.8387	0.8725	–	–
6QXD	0.1253	0.1452	0.2551	0.2967	0.1066	0.2553	0.2377	0.2818	0.1416	0.1851	0.1124
Z410	0.8612	0.8694	0.9056	0.8041	0.8789	0.8547	0.2438	0.4248	0.5324	–	–
YD2L	0.3845	0.2281	0.4222	0.2518	0.1828	0.2190	0.2074	0.1256	0.2933	0.1761	–
S8SG	0.7519	0.3542	0.7213	0.7149	0.6940	0.7532	–	–	–	–	–
I204	0.7139	0.7596	0.7841	0.3972	0.7915	0.7317	0.7829	–	–	–	–
IFW2	0.5636	0.4081	0.4248	0.2474	0.4381	0.7159	0.8672	0.3548	0.4859	0.8462	–
58QF	0.8498	0.8607	0.8619	0.8105	0.8275	0.8585	–	–	–	–	–
RR6Z	0.4124	0.2251	0.1645	0.2875	0.2672	0.1019	0.2464	–	–	–	–
RE38	0.8333	0.8476	0.8109	0.7259	0.5424	0.8945	–	–	–	–	–

Table A.2: ResNet specificity on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0.7919	0.9816	0.9310	0.8360	0.9158	0.7743	0.9736	–	–	–	–
I1HP	0.7082	0.8268	0.8446	0.7715	0.8904	0.7201	0.8762	0.7179	0.8989	–	–
6QXD	0.6567	0.6990	0.6196	0.7451	0.7053	0.6128	0.6517	0.7589	0.6639	0.5726	0.7912
Z410	0.8000	0.8952	0.8696	0.9139	0.8767	0.8334	0.9116	0.8081	0.9007	–	–
YD2L	0.5464	0.5138	0.5304	0.5041	0.5108	0.5876	0.5892	0.5835	0.5944	0.5600	–
S8SG	0.7357	0.9487	0.8392	0.8412	0.6870	0.6550	–	–	–	–	–
I204	0.8776	0.9692	0.7386	0.9520	0.8941	0.7140	0.8609	–	–	–	–
IFW2	0.8552	0.7413	0.8875	0.7469	0.8444	0.8771	0.7581	0.7254	0.8287	0.7171	–
58QF	0.7326	0.7984	0.7333	0.7552	0.7944	0.8595	–	–	–	–	–
RR6Z	0.6688	0.7381	0.6594	0.7330	0.6145	0.6664	0.6213	–	–	–	–
RE38	0.8676	0.8453	0.8511	0.8334	0.8938	0.8253	–	–	–	–	–

Table A.3: ResNet PPV on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0.6709	0.9766	0.9204	0.8160	0.8487	0.7445	0.9621	–	–	–	–
I1HP	0.7204	0.8322	0.8273	0.6923	0.7962	0.6564	0.8707	0.7483	0.8962	–	–
6QXD	0.2674	0.3254	0.4014	0.5379	0.2656	0.3974	0.4056	0.5389	0.2964	0.3022	0.3499
Z410	0.8115	0.8924	0.8741	0.9033	0.8770	0.8369	0.7339	0.6888	0.8428	–	–
YD2L	0.4588	0.3193	0.4734	0.3368	0.2720	0.3468	0.3355	0.2317	0.4197	0.2858	–
S8SG	0.7399	0.8735	0.8177	0.8182	0.6892	0.6858	–	–	–	–	–
I204	0.8536	0.9610	0.7500	0.8922	0.8820	0.7190	0.8491	–	–	–	–
IFW2	0.7956	0.6120	0.7906	0.4943	0.7379	0.8535	0.7819	0.5637	0.7393	0.7494	–
58QF	0.7607	0.8102	0.7637	0.7680	0.8010	0.8594	–	–	–	–	–
RR6Z	0.5546	0.4622	0.3257	0.5185	0.4094	0.2340	0.3942	–	–	–	–
RE38	0.8629	0.8457	0.8449	0.8133	0.8363	0.8366	–	–	–	–	–

Table A.4: ResNet F1-Score on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0.5198	0.8594	0.8544	0.7692	0.6070	0.6983	0.7903	–	–	–	–
I1HP	0.7358	0.8453	0.7836	0.5901	0.5568	0.5894	0.8518	0.7909	0.8842	–	–
6QXD	0.1706	0.2008	0.3120	0.3824	0.1521	0.3109	0.2997	0.3701	0.1916	0.2296	0.1701
Z410	0.8356	0.8808	0.8896	0.8508	0.8779	0.8457	0.3660	0.5255	0.6526	–	–
YD2L	0.4184	0.2661	0.4463	0.2882	0.2187	0.2685	0.2563	0.1629	0.3453	0.2179	–
S8SG	0.7459	0.5040	0.7665	0.7631	0.6916	0.7179	–	–	–	–	–
I204	0.7775	0.8485	0.7667	0.5497	0.8343	0.7253	0.8147	–	–	–	–
IFW2	0.6598	0.4897	0.5527	0.3298	0.5498	0.7787	0.8223	0.4355	0.5864	0.7949	–
58QF	0.8028	0.8347	0.8098	0.7887	0.8140	0.8589	–	–	–	–	–
RR6Z	0.4730	0.3028	0.2186	0.3699	0.3233	0.1420	0.3032	–	–	–	–
RE38	0.8478	0.8466	0.8275	0.7671	0.6580	0.8646	–	–	–	–	–

Figure A.1: TP, TN, FP, FN of ResNet evaluation on each test set (Part 1). Normalised values.

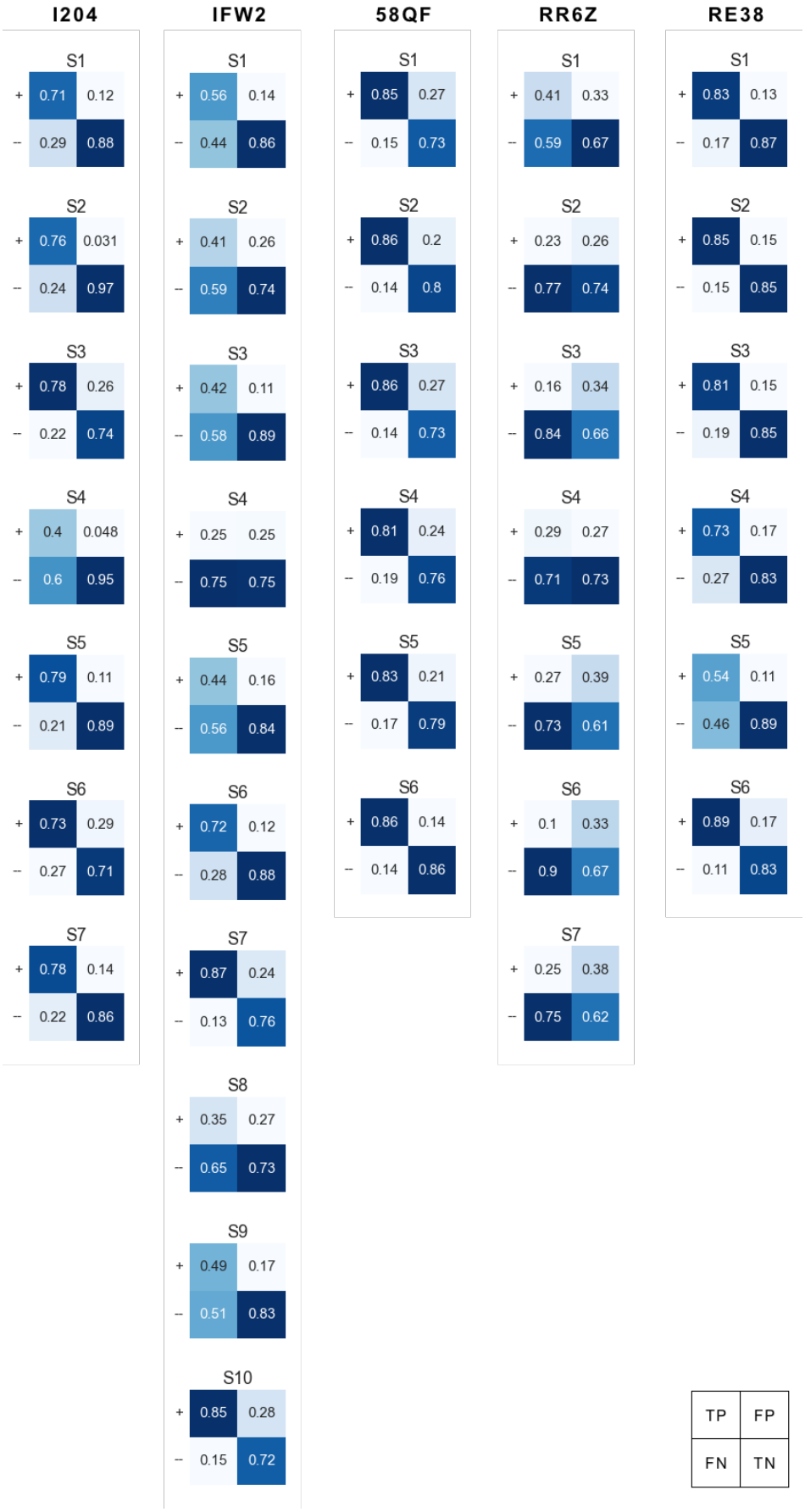


Figure A.2: TP, TN, FP, FN of ResNet evaluation on each test set (Part 2). Normalised values.

Table A.5: Decision algorithm F1-Score on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0.9485	0.9143	0.9818	0.9001	0.9654	0.9435	0.9512	–	–	–	–
I1HP	0.9851	0.9793	0.8188	0.9847	0.8143	0.8435	0.9548	0.8439	0.9501	–	–
6QXD	0.4462	0.5458	0.5045	0.5327	0.5263	0.5520	0.5489	0.5743	0.5229	0.4481	0.4300
Z410	0.9418	0.9440	0.9102	0.9208	0.9795	0.9643	0.9467	0.9614	0.9582	–	–
YD2L	0.3564	0.2527	0.1730	0.1633	0.3282	0.2101	0.3386	0.3998	0.3399	0.2447	–
S8SG	0.7693	0.7048	0.7217	0.7442	0.7132	0.7425	–	–	–	–	–
I204	0.9433	0.9867	0.9732	0.9587	0.9777	0.9116	0.9267	–	–	–	–
IFW2	0.9255	0.8602	0.8765	0.9496	0.8319	0.8612	0.9166	0.9114	0.8069	0.8927	–
58QF	0.7145	0.7572	0.7949	0.7943	0.7435	0.7404	–	–	–	–	–
RR6Z	0.7406	0.7585	0.8279	0.8228	0.7036	0.7281	0.8345	–	–	–	–
RE38	0.7744	0.7278	0.7807	0.7961	0.6444	0.6190	–	–	–	–	–

Table A.6: Number of false alarms of the decision algorithm on the test set of each seizure fold, grouped by patient.

Patient	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
2XF9	0	1	0	0	0	0	0	–	–	–	–
I1HP	1	1	1	1	1	1	1	1	1	–	–
6QXD	5	1	4	5	1	5	1	5	1	2	3
Z410	0	0	0	0	0	0	0	0	0	–	–
YD2L	30	28	29	5	18	26	22	24	18	30	–
S8SG	1	1	1	2	1	2	–	–	–	–	–
I204	0	1	1	0	1	0	0	–	–	–	–
IFW2	1	1	1	1	1	1	1	1	1	1	–
58QF	5	4	5	3	4	5	–	–	–	–	–
RR6Z	2	5	1	4	5	5	5	–	–	–	–
RE38	0	0	0	0	0	0	–	–	–	–	–

Appendix B

SupervisingTrainer Report Exemplar

Figure B.1 shows an example of multiple train-test sessions conducted by a *SupervisingTrainer* on a Gradient Boost Descent model, with architecture design with SciKit Learn. In this case the *Metric*s chosen were the mean squared error (MSE) (*ValueMetric*), and the losse curves, importance of each time series and their permutation (*PlotMetric*). Only the first two pages are shown. *SupervisingTrainer* produces the report and saves a version of it at the end of each training session. If the cycle is interrupted, the user can still have a semi-complete report.

Figure B.1: Exemplar of *SupervisingTrainer* report.