FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Diagnosis of epilepsy in EEGs using limited supervision

Joana Sofia Mendes Ramos

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Dr Luís Filipe Teixeira

Supervisor: Prof. Dr Michel van Putten

Co-Supervisor: Eng.ª Catarina da Silva Lourenço

July 21, 2021

Diagnosis of epilepsy in EEGs using limited supervision

Joana Sofia Mendes Ramos

Mestrado Integrado em Engenharia Informática e Computação

July 21, 2021

Abstract

Electroencephalographies (EEGs) are exams that monitor brain activity that are widely used as an auxiliary tool in the diagnosis of epilepsy, a neurological disease. EEGs can be ictal (i.e., during a seizure) or interictal. In interictal EEGs, specialists look for the presence of patterns that seem to be highly correlated with epilepsy. These patterns are called interictal epileptiform discharges (IEDs) and are useful to distinguish epilepsy from other conditions.

This diagnosis is not trivial since it is time-consuming and requires trained specialists. For this reason, a reliable automated diagnosis is of importance. Using machine learning it is possible to accomplish such task but a lot of data is needed to train the models, especially using deep learning paradigms. Yet, although there is plenty of data available for problems such as this one, most of the time that data is not annotated. With self-supervised learning, it is possible to train with limited annotations, making it possible to use all the available data.

We explored self-supervised learning as a method to learn the model representation efficiently and extract physiologically meaningful features from the EEG data. This is possible through the use of pretext tasks, which are regression or classification tasks for which an algorithm can autonomously generate the true labels. Ultimately, these tasks provide knowledge on the data.

In this approach, we designed two different pretext tasks, Relative Positioning (literature-inspired) and Channel Correlation. The models developed to perform such tasks were then used to initialize a model that performed IED detection. The results of such weight initialization were compared against a well-established baseline with random initialization. It was proven that this SSL approach could make up for the lack of positive samples in low-labelled data regimes.

Keywords: Epilepsy, EEG, machine learning, self-supervised learning, IED, automated diagnosis

Resumo

Eletroencefalogramas (EEGs) são exames que monitorizam a atividade cerebral, frequentemente usados para auxiliar no diagnóstico da epilepsia, uma doença do foro neurológico. EEGs podem ser ictais (i.e., durante uma crise convulsiva) ou interictais. Nos EEGs interictais, especialistas clínicos procuram por certos padrões que parecem estar altamente relacionados com a presença de epilepsia. Estes padrões chamam-se Descargas Epileptiformes Interictais (DEIs), e são úteis para distinguir epilepsia de outras condições.

Este diagnóstico está longe de ser trivial, uma vez que leva bastante tempo e requer especialistas treinados. Por este motivo, seria útil e importante ter uma ferramenta fiável de diagnóstico automático. Fazendo uso de *machine learning* é possível atingir este objetivo mas são necessários muitos dados para treinar os modelos, especialmente utilizando paradigmas de *deep learning*. Mas, embora existam muitos dados disponíveis para problemas como este, na maioria das vezes esses dados não estão anotados. Com *self-supervised learning*, é possível treinar com uma quantidade limitada de anotações, tornando possível o uso de todos os dados disponíveis.

Exploramos *self-supervised learning* como um método para aprender a representação do modelo e extrair *features* com um significado fisiólogico. Isto é possível através de tarefas de pretexto, que são tarefas de classificação ou regressão para as quais um algoritmo consegue gerar automaticamente as *labels*. Em última instância, estas tarefas dotam o modelo de conhecimento sobre os dados.

Nesta abordagem, criámos duas tarefas de pretexto, *Relative Positioning* (inspirada na literatura) e *Channel Correlation*. Os modelos desenvolvidos para levar a cabo essas tarefas foram usados posteriormente para inicializar um modelo que detetasse DEIs. Os resultados dessa inicialização de pesos foram comparados com uma *baseline* bem estabelecida com inicialização aleatória. Foi provado que esta abordagem consegue compensar a falta de exemplos positivos num regime de dados fracamente anotados.

Keywords: Epilepsia, EEG, machine learning, self-supervised learning, DEI, diagnóstico automático

Acknowledgements

First and foremost, I would like to thank my supervisors Luís Teixeira, Catarina Lourenço and Michel van Putten for their invaluable efforts in helping me throughout this work. Your feedback and ideas were much appreciated.

I wish to thank everyone in the CNPH group who pitched in. This was the most encouraging environment I could ask for to carry out this project.

To the incredible *Tugas in Enschede*, especially my amazing housemates Catarina, Manuel and Mariana, it has been great exploring the Netherlands with you. Nonetheless, a big thank you to my friends from home, who were always with me.

I have to mention Joana and Maggie, with whom I have shared more than I can recall; you are the best.

To the people I have missed the most these past five months, my family, thank you for giving me the freedom to be whoever I want to be and for being proud of me for it. Also, thank you for taking care of my cat while I was away.

Finally, a special thank you to Koen; you are definitely my favourite Dutch person.

Joana Sofia

Contents

List of Figures

List of Tables

Abbreviations

EEG	Electroencephalography
IED	Interictal Epileptiform Discharge
PWE	People With Epilepsy
SSL	Self-Supervised Learning
CNN	Convolutional Neural Network
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
AE	Autoencoder
SVM	Support Vector Machine
DL	Deep Learning
RNN	Recurrent Neural Network
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
SGD	Stochastic Gradient Descent
ANN	Artificial Neural Network
RP	Relative Positioning
CC	Channel Correlation

Chapter 1

Introduction

Epilepsy is a chronic neurological condition characterised by the recurrent occurrence of unprovoked seizures [1]. As of 2007, this disorder was the most common neurological disorder in humans and, according to World Health Organization data, affected approximately 50 million people in the world [2]. However, it is important to note that the presence of seizures does not mean that a patient has epilepsy since these can occur due to other conditions. As such, the clinical diagnosis takes into account other factors besides ictal periods (which are not a necessary condition for an epilepsy diagnosis), such as electroencephalography (EEG) recordings [3].

1.1 Context and Motivation

Electroencephalography is a fundamental tool in diagnosing and monitoring people with epilepsy (PWE). Non-invasive EEG is a procedure that records cerebral electrical activity through the electrodes placed on a patient's scalp. Currently, the gold standard for diagnosing this disease lies in the visual analysis of EEG recordings. These exams can record ictal activity if performed during a seizure, or interictal activity if performed while the patient is not having a seizure. An ictal EEG is the most efficient way of distinguishing an epileptic seizure from a non-epileptic one. However, it is rare to have an ictal EEG. In interictal EEGs, lab technicians look for interictal epileptiform discharges (IEDs), which are patterns that seem to be highly correlated with epilepsy [3, 4].

This visual analysis is not trivial since it is time-consuming and requires trained specialists. More so, it might not be possible at all for many patients since 85% of the people affected by this disorder live in developing countries [4]. For these reasons, a reliable automated diagnosis is of importance.

Machine learning models require many data to be trained, validated and tested, especially in deep learning paradigms where the architecture is more layered and complex. However, although there is plenty of data available for problems such as this one, most of the time, that data is not annotated.

Self-supervised learning (SSL) tackles the problem of limited annotated data by leveraging large amounts of unlabelled data to uncover its structure. This is possible through the use of pretext tasks. These are machine learning tasks that intend to provide knowledge on the data. This means that the model needs to extract meaningful patterns from the input to correctly predict the output.

1.2 Objectives

Ideally, the representations created by the pretext tasks are meaningful enough that the model that performs the classification task at hands (in this case, detect the presence of IEDs in EEGs) will need fewer annotated examples to achieve state-of-the-art performance. As such, this work aims to prove that SSL can make up for the lack of annotated EEG data.

1.3 Document Structure

This document is divided into 6 chapters. A brief introduction is made in Chapter 1, contextualizing the problem, the motivation and the objectives of this approach. In Chapter 2 the state of the art is assessed, clarifying some background terms. The pretext tasks designed for this work, as well as the self-supervised learning pipeline, are presented in Chapter 3. The obtained results are shown in Chapter 4 and discussed in Chapter 5. Finally, the conclusions that can be drawn from this dissertation can be read in Chapter 6, as well as the future work.

Chapter 2

State of the Art

2.1 Background

2.1.1 Electroencephalography

The electroencephalogram is normally a non-invasive recording of cerebral electrical potentials by means of electrodes placed on the scalp [5]. The first EEG performed on a human was done in 1924 by Hans Berger, a German physiologist and psychiatrist [6]. This method is in practice to this day and is used to diagnose and monitor epilepsy, among other brain conditions and clinical purposes. In the machine learning field, it has been used for clinical domains such as the detection/prediction of Alzheimer's disease, epilepsy, schizophrenia and sleep stages.

During an EEG, electrodes are pasted onto the scalp of the patient. Electrodes are patches made of metal conductors that detect electrical potentials which result from neural activity. These charges are recorded either digitally or printed out on paper and analysed by clinical experts. The location of each electrode on the scalp depends on the electrode placement system used. One of the most common is the International 10-20 system, seen in Figure 2.1. In the figure, it is possible to see the electrodes represented by circles. The labels on the circles represent the lobe of the brain that is most close to that electrode. For example, the electrode *Fp1* is one of the electrodes most close to the prefrontal cortex, while *T4* is one of the electrodes that tracks activity from the temporal lobe. In Figure 2.2, a fragment of an EEG recording from a patient with epilepsy can be seen. It consists of pairs of electrodes (that can also be called channels, although a channel does not necessarily need to be associated with a pair of electrodes) and the voltage between them over time.

The flow of acquiring and analysing EEG signals usually goes as follows: firstly, the signals are acquired from the patient and passed through an amplifier. Then, the signals are filtered according to their frequency to exclude potential biological noise. Some artefacts may also be present in the signal, due to co-occurring physiologic activity like muscle and ocular movements, or environmental noise such as that from electronic equipment. For that reason, artefact removal is also carried out. After that, the signals are divided into time windows of the same size (the size

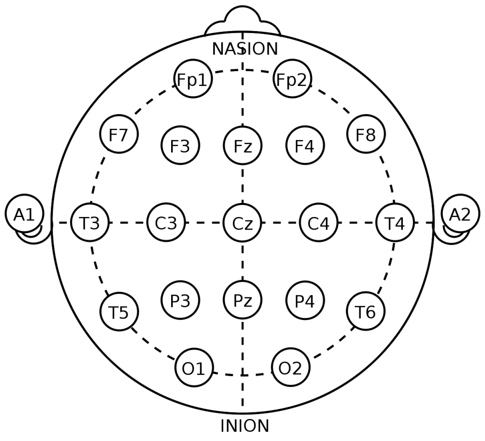


Figure 2.1: The international 10-20 system for electrode placement. The nasion is the point between the forehead and nose while the inion is the bump at back of skull.

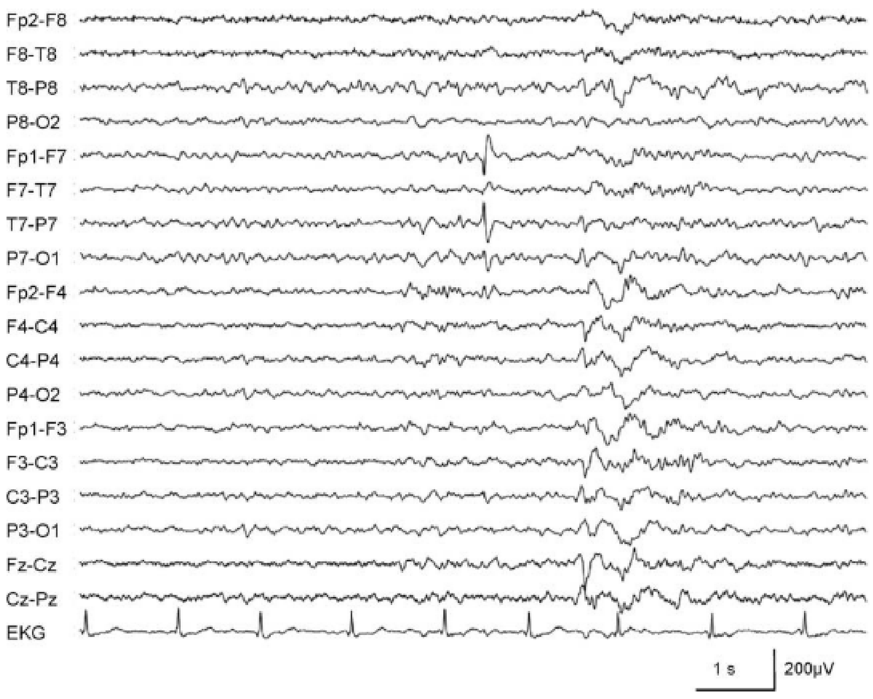


Figure 2.2: Fragment of an EEG recording of a 32-year-old patient with left temporal lobe epilepsy [7]

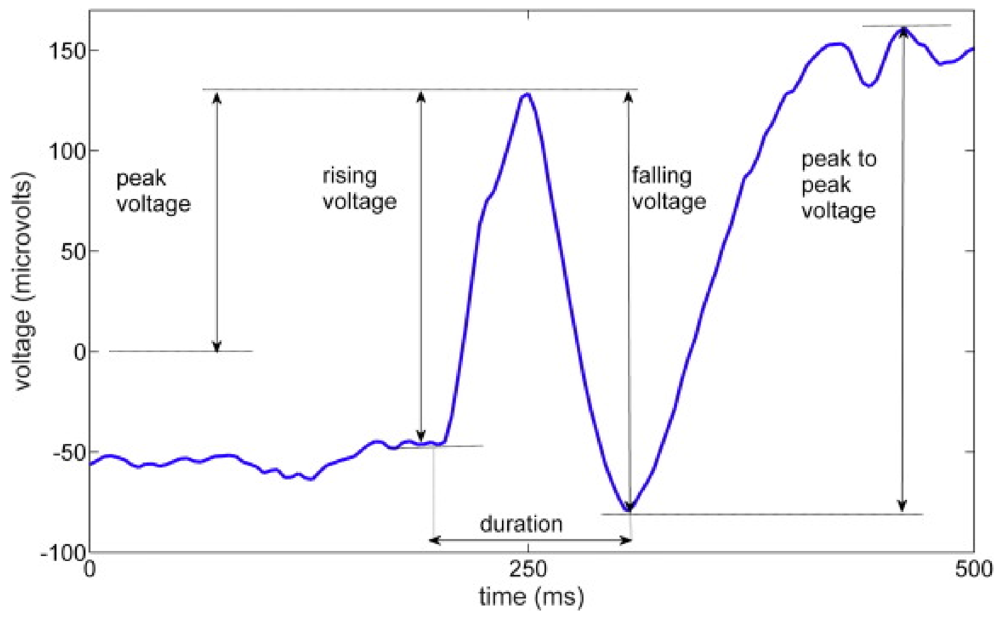


Figure 2.3: Examples of morphological features of an IED [9]

of these windows depends on the condition that is being assessed) called epochs. These epochs can then be inspected for specific electrical patterns depending on the domain.

2.1.1.1 The role of EEG in epilepsy

EEGs can be ictal (if acquired during a seizure) or interictal (if done in a so-called normal state). Interictal EEG is the most common and easy to obtain. Interictal epileptiform discharges (IEDs) are electrical patterns that can usually be seen in interictal EEGs of patients with epilepsy, and only rarely in healthy subjects (see Figure 2.3 for an example of an IED). These patterns are sudden spikes/waves often followed by a slow-wave complex [7]. IEDs also help discriminate the epilepsy condition into different syndromes, particularly, generalized epilepsy (when they arise in both brain hemispheres) or focal epilepsy (when certain regions of the brain are responsible for the epileptic activity) [8].

Currently, the analysis of EEG signals in search for IEDs is the gold standard for the clinical diagnosis of epilepsy. In order to improve the chances of interictal spikes occurring during EEG, activation procedures can be used. These procedures include: putting the patient to sleep, inducing hyperventilation by having the patient breathe very quickly and performing photic stimulation [10].

2.1.2 Self-Supervised Learning

Self-supervised learning (SSL) is a learning technique for which the goal is to uncover meaningful structures in data and use them to make predictions. This type of learning has been emerging recently because it seems to mimic the way humans learn. It has gained a lot of attention, especially in the computer vision field [11].

There are two main concepts in SSL: pretext tasks and downstream tasks. The pretext task is a machine learning task that is meant to provide a meaningful representation of the data that can later be used to classify the data according to the downstream task (the actual classification problem we intend to tackle).

The pretext tasks must have the following characteristics:

- The labels can be generated automatically from the dataset;
- The task requires understanding, i.e. the model has to extract meaningful patterns from the data in order to predict the output correctly.

Usually, these are the main steps to carry out a pretext task:

1. *Data generation:* create new examples from the original ones;
2. *Self-labelling:* associate the new examples with the auxiliary values related to their creation, these are their labels;
3. *Classify/predict:* develop a model to classify/predict all examples;

Here is an example of this framework on a pretext task developed for a dataset consisting of images:

1. *Data generation:* rotate the original image X_i by an arbitrary angle of θ_i°. This generates a new example $X_{i'}$;
2. *Self-labelling:* pair $X_{i'}$ with its label which is the rotation angle θ_i;
3. *Classify/predict:* develop a model that, given an image X predicts the angle θ by which that image was rotated. In the case of images that were not altered, θ should be 0.

A general and theoretically grounded framework to SSL was recently formalized in [12] from the perspective of nonlinear independent components analysis (ICA). Under that framework, variables are statistically dependent on a random variable u (e.g., θ from the previous example) and u is used to perform data augmentation. Then, a classifier is trained to predict whether a sample is paired correctly with its auxiliary variable u ($\theta = 0^\circ$) or a perturbed (random) one, u' ($\theta \neq 0^\circ$). In the case of EEG data, this auxiliary variable u can be time and, as such, a lot of the pretext tasks from the literature consist of temporal shuffling, which will be discussed later.

2.2 Deep Learning for EEG Analysis

Deep learning (DL) has been trending in this last decade in machine learning applications because of its completeness: it makes it possible to skip steps such as data pre-processing and feature extraction. It does this by using deep neural networks (DNNs), where deep refers to the number of layers of the network (even though there is not a consensus on how many layers a network should have in order to be considered deep). This method has been widely used in the computer vision field because of the hierarchical nature of the features present in an image. The first layers seem to extract low-level features like edges while the last layers deal with more complex features, for example, a nose in a picture of a person.

DL has also been adopted in many other areas, and EEG analysis is one of them. As was said before, algorithmically analysing an EEG requires some pre-processing. Because of its automatic end-to-end learning, DL is believed to reduce the number of pre-processing operations necessary to obtain good results.

In [13], 154 papers consisting of different deep learning approaches to EEG analysis are reviewed. These approaches are inserted in several domains, from the detection of epilepsy, Alzheimer's disease and schizophrenia to cognitive and affective monitoring. Regarding common phases for machine learning tasks, here is the general outlook:

- **Dataset:** half of the datasets contained epochs from less than 13 subjects, although this results in different input sizes, depending on the sampling rate (most common was 250Hz) and the epoch size;

- **Data augmentation:** few studies carried out data augmentation, having demonstrated better results when that technique was employed. Multiple studies used overlapping epochs as a way to augment their data;

- **Artefact removal:** almost half the papers did not perform artefact removal, although, according to the authors, it may be essential for achieving good performance;

- **Feature extraction:** the studies that performed feature extraction usually did so with unsupervised methods, namely restricted Boltzmann machines (RBMs), deep belief networks (DBNs) and autoencoders (AEs). However, most studies leveraged deep learning to perform both feature extraction and classification/regression;

- **Architecture design:** convolutional neural networks (CNNs) were the most common architecture type followed by recurrent neural networks (RNNs). Most of the architectures had no more than 10 layers;

- **Network regularization:** 79 papers used at least one regularization method;

- **Optimizer:** most papers did not mention the optimizer that was used, but Adam and stochastic gradient descent (SGD) were the most common optimizers for those that did;

- **Hyperparameter search:** some studies performed hyperparameter search using either grid search or Bayesian methods;

- **Model inspection:** most solutions employed at least one model inspection technique.

Regarding CNNs and RNNs in particular, their popularity seems natural given their ability to deal with time-series data. CNNs, in particular, can be used for EEG data in two ways: the conventional one using 2D convolutions (one dimension is the time and another is the location/channels) or simply 1D convolutions, having only one dimension which is time.

In what concerns the results, it is hard to compare the different solutions: this field lacks benchmark datasets and baseline models. It also remains unclear whether deep learning methods are more efficient than traditional machine learning algorithms in these domains. However, the difference in accuracy between each proposed model and corresponding baseline (defined by the authors) per domain type was assessed, and the median gain in accuracy with DL was of 5.4% [13]. It is worth noting that the lack of labelled data in clinical settings was specifically pointed out as an obstacle.

2.3 Self-Supervised Learning in EEG

Self-supervised learning is a recent trend in the machine learning field. For that reason, the literature on this topic is scarce. Recently, it has been used in some approaches for automatic EEG analysis, namely, in 3 studies (one in 2019, one in 2020 and one in 2021). These will be presented next.

In [14], the authors present a self-supervised solution for the multiclass problem of identifying sleep stages using EEG data. Experiments were performed with two pretext tasks (seen in Figure 2.4): relative positioning (RP) and temporal shuffling (TS). This solution was framed under the framework discussed before and the auxiliary variable used, u, was the time index, under the assumption that an accurate representation of the EEG data should depend and evolve according to time.

For the RP task, two values were considered, τ_{pos} and τ_{neg}, which are the durations of two contexts around time windows, namely the positive context and the negative context, correspondingly. In order to create the self-labelled dataset, pairs $(x_t, x_{t'})$ of time windows are sampled. The goal is to classify each pair with 1, if $|t - t'| \leq \tau_{pos}$ or -1, if $|t - t'| > \tau_{neg}$. Similarly, the TS task samples tuples $(x_t, x_{t'}, x_{t''})$, where $x_{t''}$ is in the positive context of x_t. The goal is now to classify the tuple as temporally ordered ($t < t' < t''$) or shuffled.

In both tasks, firstly, a feature extractor h was used. This feature extractor is an embedder that maps a time window to its representation on the feature space. The architecture used for this feature extractor was adapted from a convolutional neural network (CNN) previously used in the literature that had shown to perform well on this problem (sleep staging). A contrastive module is then used in order to combine all the representations together, by computing an elementwise absolute difference. Finally, logistic regression is used to predict the labels of each pair or tuple.

The authors conducted experiments on two public datasets of EEG sleep data: the Physionet Sleep EDF expanded dataset [15] and the MASS dataset session 3 [16]. The datasets comprised, in total, 189,510 (Physionet Sleep EDF dataset) and 57,445 (MASS dataset) time windows from 83 and 62 patients, correspondingly. The results were compared with 3 different baselines for h:

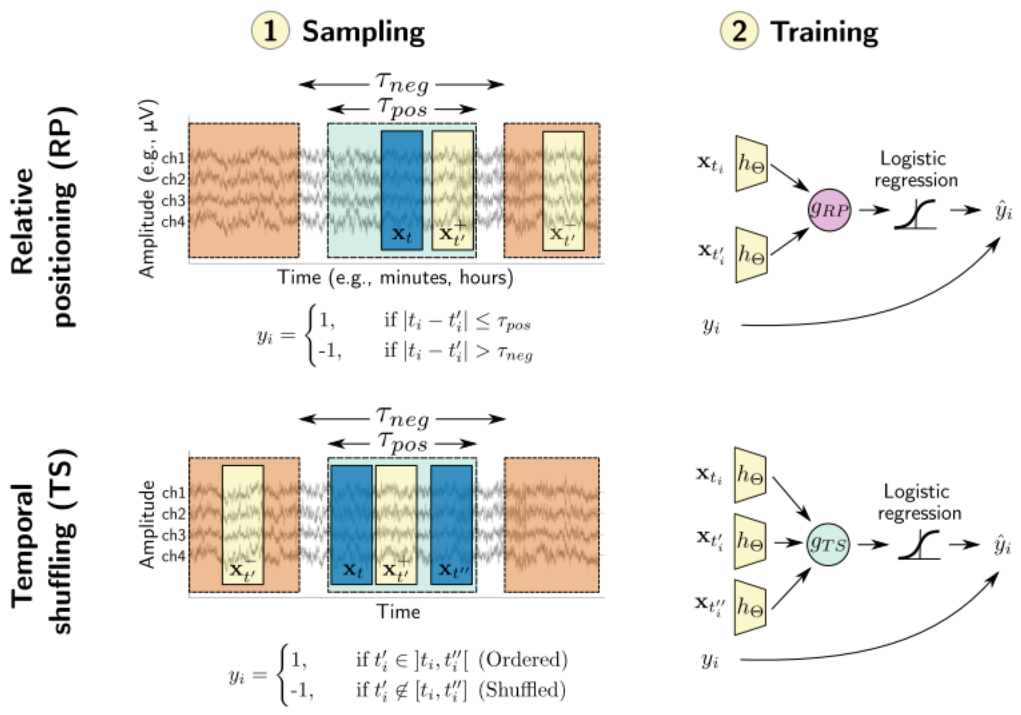
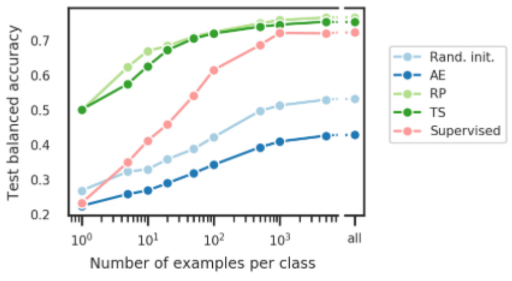

1 Sampling
2 Training
Relative positioning (RP)
Temporal shuffling (TS)
Amplitude (e.g., µV)
Amplitude
Time (e.g., minutes, hours)
Time
Logistic regression
$y_i = \begin{cases} 1, & \text{if } |t_i - t'_i| \le \tau_{pos} \\ -1, & \text{if } |t_i - t'_i| > \tau_{neg} \end{cases}$
$y_i = \begin{cases} 1, & \text{if } t'_i \in]t_i, t''_i[\text{ (Ordered)} \\ -1, & \text{if } t'_i \notin [t_i, t''_i] \text{ (Shuffled)} \end{cases}$

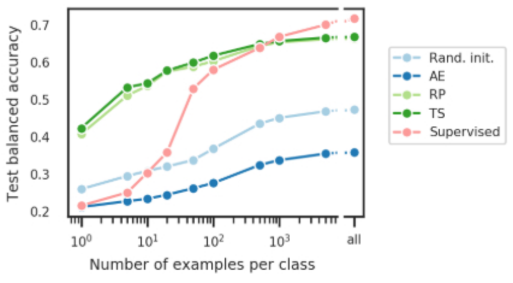
Figure 2.4: Visual representation of the TS and RP pretext tasks [14]

an artificial neural network (ANN) with random initialization and frozen weights, a convolutional autoencoder (AE) and a purely supervised model (by adding an additional softmax layer). Balanced accuracy was used as an indicator of performance and comparison was made by varying the number of labelled examples per class (see Figure 2.5).

On MASS, the SSL approach outperformed all baselines, for all data regimes (number of labelled examples). The same happens on the Physionet Sleep EDF dataset, except for when there are more than 500 examples per class available, where the purely supervised model outperformed all the other ones. When it comes to comparing the pretext tasks (RP and TS) between themselves, RP was slightly better than TS on the MASS dataset, well the opposite happened on the Physionet Sleep EDF dataset.



(a) Mass dataset



(b) Physionet Sleep EDF dataset

Figure 2.5: Impact of varying the number of examples per class on each dataset [14]

Additionally, in order to explore the features learned with SSL, the embeddings (representations) obtained on the Physionet Sleep EDF dataset were projected to two dimensions using UMAP [17]. This projection showed that the learned features were physiologically meaningful, which is the primary goal of pretext tasks (discover meaningful patterns in data) and SSL in general.

In [18], SSL is used in a slightly different manner: the pretext task is enough to fulfil the downstream task. The authors formulate an anomaly detection problem where the goal is to score the degree of abnormality in EEG data. The EEG data included short sequences (time windows of 1 second) of signal values from multiple electrodes and patients.

For this purpose, one pretext task was developed based on the fact that abnormal EEG data often include signals of higher frequencies. This task performs a scaling transformation on the data by interpolating the EEG values in order to create a new sequence of values, with lower frequency. The resulting sequence has $s_k \times d$ values where d is the number of values in the original sequences, and s_k is the scaling factor (the values used for this factor were 1.0, 2.0 and 3.0). Finally, d contiguous values from the scaled sequence are extracted from either the beginning or the centre of the sequence, so that all transformed sequences have the same length. All transformed sequences together construct a self-labelled dataset, where each label in the labels vector is the same as the scaling factor s_k applied to the sequence.

These sequences are then fed into a deep convolutional neural network (CNN) that reuses the backbone of a well known CNN, ResNet34 [19]. In the convolutional layers of the ResNet network, the kernels have a size of 3×3. Initially, the kernels were to be altered to a 1×3 size. However, it was experimentally verified that 3×3 kernels performed better, which is interesting because it suggests that some relationship was captured between the two dimensions (brain region and time). The classifier was trained using only healthy EEG data, i.e., EEG from patients without any disease or condition. This way, one could expect new normal EEG data to be classified correctly, while abnormal EEG data would ideally be misclassified, and the differences between the predicted values and the ground truth could be used to measure the degree of abnormality.

This solution was compared against well-known anomaly detection methods, namely the one-class support vector machine (OC-SVM), the statistical kernel density estimation (KDE) method, the autoencoder (AE) and the variational autoencoder (VAE). The self-supervised solution outperformed every one of these methods, as can be seen in Figure 2.6 (the performance metric used to this end was the AUC).

The robustness of the developed model was also tested by varying some settings:

- The CNN structure (VGG19 [20], ResNet18, ResNet34, ResNet50, and DenseNet121 [21]);
- Number of scaling transformations (2, 3, 4 and 5);
- The scaling range ($[1-2]$, $[1-2.5]$, $[1-3]$, $[1-3.5]$ and $[1-4]$);
- The sampling position, i.e., the heuristic used to sample the d values from the scaled sequence (starting from the beginning position, the one-third position, or around the centre of the scaled sequence).

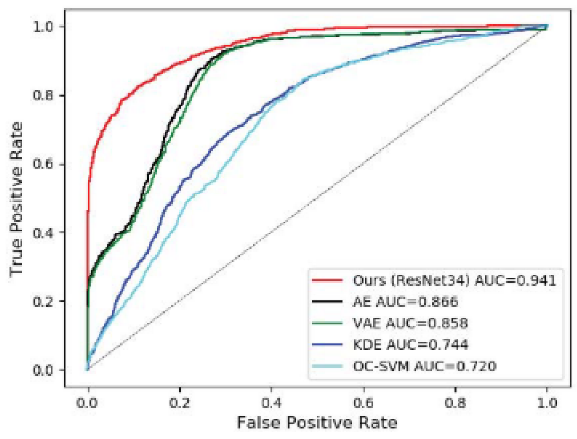


Figure 2.6: Comparison of the different approaches for anomaly detection [18]

The solution proved to be robust to these changes, demonstrating relatively small differences in performance. Despite these apparently excellent results, it is worth noting that the data used for experimental evaluation consisted of EEG data from 4 dogs. It would be important to evaluate the same framework with a more representative dataset as well as human patients.

Recently, another self-supervised learning approach to EEG was developed [22], inspired by `wav2vec 2.0` [23], which is a framework for self-supervised speech recognition. Recognizing the impact of ImageNet [24] pre-training in the computer vision field, the authors proposed that DNN transfer learning in brain-computer interface (BCI) and neuroimaging analysis could follow a similar line, with encephalography models. To test this hypothesis, the authors compiled a dataset with data from various datasets, namely, TUEG [25], MMI [26], BCIC [27], ERN [28], P300 [29] and SSC [30]. However, this raises uniformity problems. These were solved by:

- focusing on the 19 channels from the 10/20 channel set (all additional channels were ignored, while missing ones were set to 0);
- over- or under-sampling the signals to achieve a sampling frequency of 256 Hz.

The model architecture resembles that of `wav2vec 2.0`, comprised of two stages. The first one was titled by the authors as BENDR (BErt-inspired Neural Data Representations). BENDR takes raw EEG data and downsamples it to a new sequence of vectors. The second stage uses a transformer encoder to map the previous representations to a new sequence for the downstream task. The main difference between BENDR and the original architecture is that BENDR treats several channels, while `wav2vec 2.0` only 1 (raw audio).

In accordance with their results, the authors believed this pre-training step to be suitable for all EEG data and every domain of the target task (e.g. sleep stage classification, pathology detection, affective monitoring, etc.). If accurate, this can be paradigm-changing in the field of automatic EEG analysis.

In Table 2.1, a summary of these studies and their results can be seen. Since SSL is an increasing trend, we expect more solutions and approaches to come out during our work.

Study	**Task**	**Results**
H. Banville, G. Moffat, I. Albuquerque, D. A. Engemann, A. Hyvarinen, and A. Gramfort [14]	Sleep stage classification	SSL outperformed all baselines in low-labelled data regimes.
J. Xu, Y. Zheng, Y. Mao, R. Wang, and W. S. Zheng [18]	EEG anomaly detection	SSL outperformed other anomaly detection methods considered by the authors (AE, VAE, KDE and OC-SVM), with an AUC of 0.941.
D. Kostas, S. Aroca-Ouellette, and F. Rudzicz [22]	Pre-training a model for EEG tasks	SSL keeps up with other approaches on the same datasets used, however, authors claim that with fine-tuning for specific domains the BENDR system can achieve better performance.

Table 2.1: Summary of the studies presented

Chapter 3

Methods

The purpose of this Chapter is to explain the development pipeline used in this project. In Section 3.1, the dataset is presented, and the data preparation and understanding processes are carried out. Section 3.3 comprehends a brief description of the employed self-supervised learning framework, while Sections 3.2 and 3.4 build on this by describing, respectively, the downstream task and the pretext tasks of this work.

3.1 EEG Data

The EEG data was kindly provided by the Clinical Neurophysiology (CNPH) group [31] of the University of Twente [32]. It contained EEGs from 166 patients between 4 and 72 years of age, randomly selected from the digital database of the Medisch Spectrum Twente in the Netherlands. The data was anonymized before use. The recordings were made with twenty-one silver/silver chloride cup electrodes placed on the scalp according to the international 10-20 system. These EEGs are distributed in the following manner:

- 50 recordings from patients with focal epilepsy (from now on referred to as set **F**);
- 49 recordings from patients with generalized epilepsy (from now on referred to as set **G**);
- 66 recordings from people without epilepsy (from now on referred to as set **N**).

Sets F and G contained IEDs besides normal brain activity, while set N contained no EEG abnormalities.

This data was provided in the form of `.mat` files, with some pre-processing already employed, namely:

- Filtering operations in the 0.5–35 Hz range in order to reduce artefacts;
- The raw signal was discretized using a sampling frequency of 125 Hz to reduce input size;

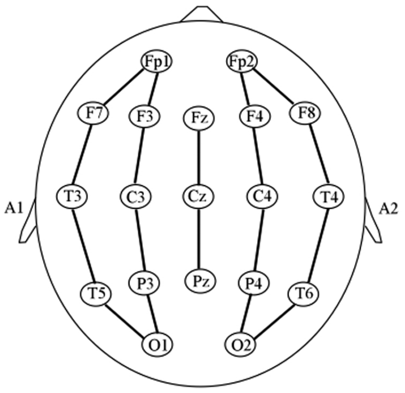


Figure 3.1: The longitudinal bipolar montage, also known as the double banana montage because of the shape formed by the electrodes on the left and right sides of the head.

- Each recording was split into 2s non-overlapping time windows, commonly called *epochs*. Each epoch contained $125 \times 2 = 250$ values. These values are potential differences between 2 electrodes that form a channel;
- Signals were re-referenced to a longitudinal bipolar montage. This montage consists of 19 electrodes placed on the scalp in the disposition seen in Figure 3.1. A list of the resulting 18 channels is presented in Table 3.1;
- Epochs were annotated by clinical experts concerning the presence of IEDs (0 if no IED was present, 1 otherwise).

In each EEG file, the following information was provided:

- The discretized signal in the form of an array with the shape: number of epochs (depends on the duration of the EEG) $\times$ number of samples (250) $\times$ number of channels (18). This structure can better be understood in Figure 3.2;
- The labels (list of 1 and 0 values). Labelling is done at the epoch level, so this list has a size equal to the number of epochs in the signal;
- The bandpass filter applied to the signal during artefact removal;
- The beginning and end time marks for each epoch.

As shown in Table 3.2, set G was the largest set, although set F contained more positive samples (IEDs). Set N was the smallest set, and, naturally, it did not contain any positive samples. The dispersion in IEDs in sets F and G can best be seen in the box plot of Figure 3.3.

Fp1-F7	Fp1-F3	Fp2-F4	Fp2-F8	F7-T3	F3-C3
Fz-Cz	F4-C4	F8-T4	T3-T5	C3-P3	Cz-Pz
C4-P4	T4-T6	T5-O1	P3-O1	P4-O2	T6-O2

Table 3.1: All 18 channels of the longitudinal bipolar montage.

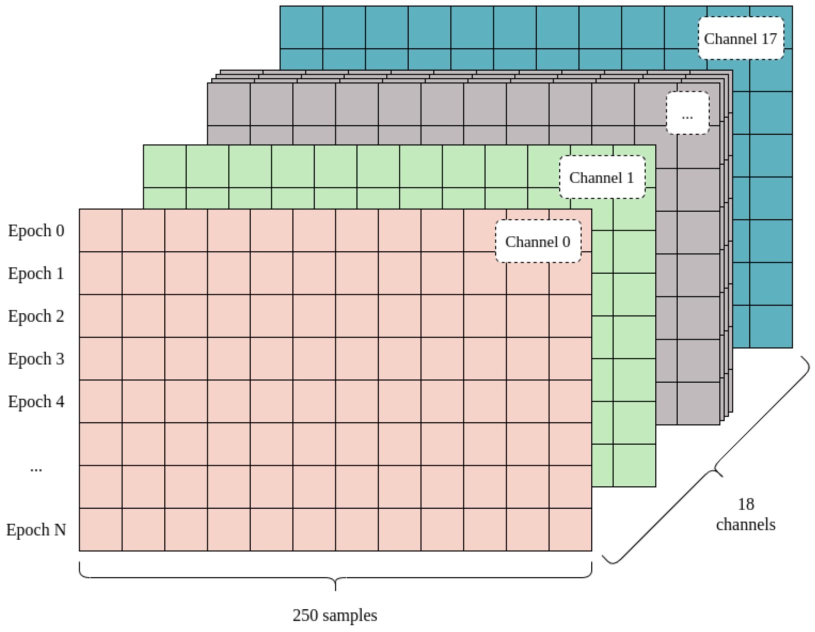


Figure 3.2: Structure of an array containing the discretized information from an EEG signal.

Set	No. of epochs	No. of minutes	No. of IEDs
F	86 250	2 875 (∼ 47.92 hours)	1 752
G	120 692	4 023.07 (∼ 67.05 hours)	912
N	57 989	1 932.97 (∼ 32.22 hours)	0
Total	264 931	8 831.04 (∼ 147.18 hours)	2 664

Table 3.2: Distribution of epochs, recording duration and IEDs per set.

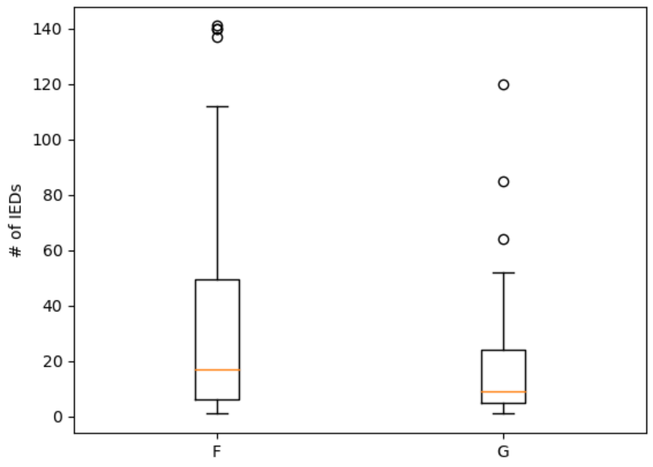


Figure 3.3: Box plot of the number of IEDs in sets F and G. Note that the dispersion in set F is much larger than the dispersion in set G, even though the first quartile is very similar in both.

	Label		
Set	0	1	Total
Training	13 383	1 487	14 870 ($\sim$ 39%)
Validation	6 597	733	7 330 ($\sim$ 19%)
Test	15 414	452	15 866 ($\sim$ 42%)
Total	35 394 ($\sim$ 93%)	2 672 ($\sim$ 7%)	38 066

Table 3.3: Data distribution for the downstream task.

3.2 Downstream Task (IED Detection)

As discussed before (see Section 2.1.2), in the context of self-supervised learning, a downstream task is a supervised learning task that can benefit from pre-trained models, especially when the available data is limited. In this project, the downstream task is a binary classification problem in which the inputs (X) are EEG epochs, and the output (y) is 0 if the epoch does not contain an IED and 1 if the epoch contains an IED. This classification task can be helpful for the clinical diagnosis of epilepsy, as we have already discussed.

3.2.1 Baseline

A baseline was established in order to compare a fully supervised learning approach to the self-supervised learning method presented in this work. In [33], an adaptation of the VGG16 model with remarkable performance (refer to the original paper) was proposed for the task at hand. The authors provided the code for that model and we used it as a basis for the baseline.

3.2.1.1 Data

The mentioned paper was also developed in the CNPH [31] group, and as such, the EEG data available included the data described in Section 3.1 as well as some additional EEGs. The data was split into training/validation/test sets with approximate proportions of 40%/20%/40%, making sure that data from one patient was only present in a single set. Not all negative samples were used for the sake of increasing the proportion of positive samples present in the data. However, the true proportion was kept in the test set (1:34). The resulting distribution of positive and negative samples throughout these sets can be seen in Table 3.3. As can be expected in clinical problems such as this one, the resulting dataset is highly unbalanced, with only 7% positive samples.

3.2.1.2 Model

Due to compatibility issues, the architecture from [33] was slightly modified. The architecture in place is essentially the same as the VGG16, with some adaptations due to the input and output shape. The convolutional blocks are the same, as well as the stride and padding. ReLu activation was also used in hidden layers. The dense layers are also the same as in the original VGG16, except for the output layer. The model architecture can be seen in Figure 3.4 while the main differences

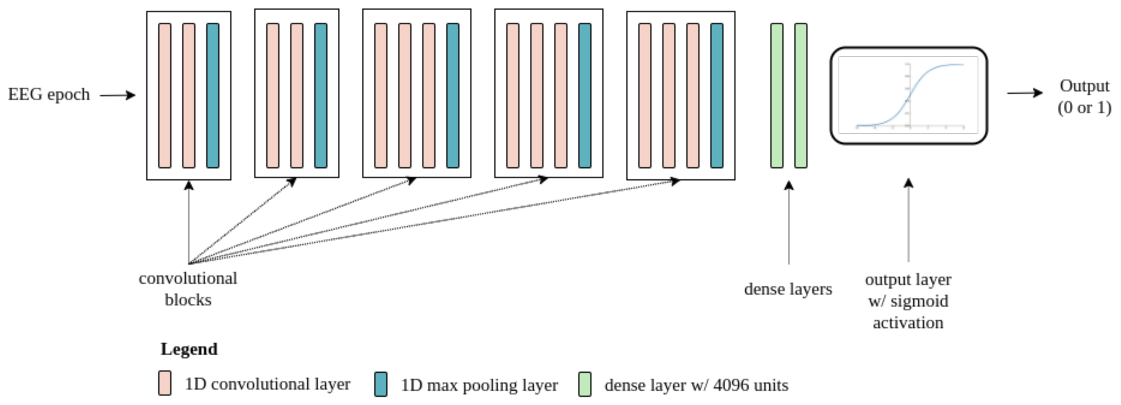


Figure 3.4: The VGG16-var model architecture. This architecture is similar to the original VGG16, except for the convolutional and max-pooling layers' dimensions and the output activation function.

between the original VGG16, the architecture from [33] and ours (referred to as **VGG16-var**) are highlighted in Table 3.4.

The model was trained for 20 epochs (not to be confused with EEG epochs) with a batch size of 64. The Adam optimizer was used with a learning rate of 2×10^{-5}, $\beta_1 = 0.91$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The loss function employed was binary cross-entropy, and class weights of 100:1 (100 corresponding to the positive class) were used to compensate the class imbalance problem.

AUC, sensitivity and specificity were monitored to evaluate model performance. The points where either the sensitivity and specificity were the same or the specificity was 99.00% were recorded.

3.3 Self-Supervised Learning Approach

In Section 2.1.2, the theory behind the SSL framework was briefly explained. Now that we have dived further into the topic, we can explore its practical side, applied to our context.

Self-supervised learning aims to take advantage of large quantities of unannotated data, and use that leverage to benefit a supervised model. Usually, in clinical settings, that is the case —

	VGG16	**VGG16-C**	**VGG16-var**
Input dimensions	$224 \times 224 \times 3$	$250 \times 18 \times 1$	250×18
Kernel size in convolutional layers	3×3	3×3	3
Kernel size in max pooling layers	2×2	2×2	2
No. of units in output layer	1 000	2	1
Activation function in output layer	Softmax	Softmax	Sigmoid

Table 3.4: Main differences between the original VGG16, the VGG16 from [33] (VGG16-C) and our adaptation, called VGG16-var.

there is a vast amount of unannotated data at our disposal.

The first step is to design a pretext task. This pretext task should take as input our data, in some shape, and the output must be automatically verifiable (no need for human annotations). As such, the next step is to automatically label the data according to our newly designed pretext task. Then, a model, m_1, can be trained to perform that classication/regression task.

Subsequently, the pre-trained model m_1 can be used to perform the downstream task (IED classification) in one of two ways:

- Continue the training of the pre-trained model m_1 to perform the downstream task, this time in a supervised manner (on the human-annotated dataset). This will be the case for one of our pretext tasks, where the model is exactly the same for the pretext and downstream tasks;
- Partially initialize a second model m_2 with model m_1's weights and train m_2 to perform the downstream task. This will be the case for our other pretext task, where the model architecture is not exactly the same and as such the m_2 model's weights are only partially initialized.

Note that in any case there are two classification/regression tasks (the pretext task and the downstream task) and two model training processes. These training processes make use of two different datasets - one self-labelled and one annotated by humans - even though the data itself is the same (in our case, EEG data).

Having a baseline that was trained in a fully supervised manner, it is then possible to compare the gain from this transfer learning process. However, since our baseline model was trained with a decent amount of annotated data and its performance was quite good, we performed an additional step: we trained the baseline model on different data regimes to monitor the impact of withdrawing human-annotated data. This was done in two ways:

Varying the amount of data Reducing the number of samples in both the training and validation sets to 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%, 9%, 8%, 7%, 6%, 5%, 4%, 3%, 2% and finally 1%;

Varying the number of positive samples Reducing the number of positive samples in both the training and validation sets to 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, 10%, 9%, 8%, 7%, 6%, 5%, 4%, 3%, 2% and finally 1%, while maintaining the number of negative samples to 100%.

The percentage interval changes from 10% to 1% in the [10-1%] range because the SSL approach should be beneficial especially in such low-labelled data regimes.

The difference on the training and validation sets with these variations can be seen in Figures 3.5 and 3.6 (to see the original training/validation sets refer to Table 3.3). This is further detailed in Tables B.1 and B.2.

The pretext tasks created for this problem will be explained in the next sections.

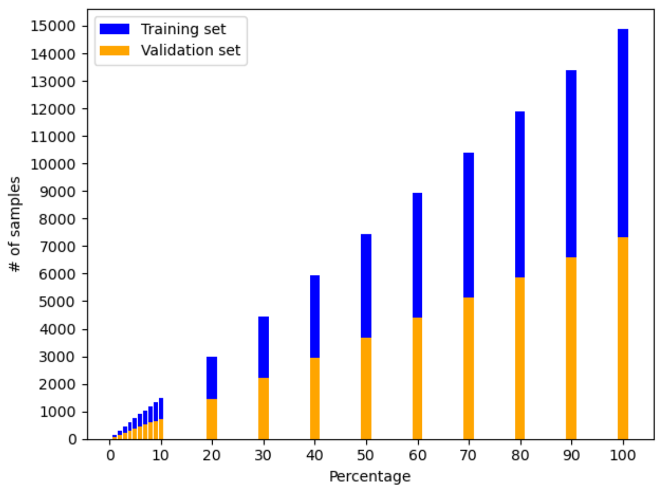


Figure 3.5: Number of training/validation samples obtained by linearly decreasing the amount of available data used.

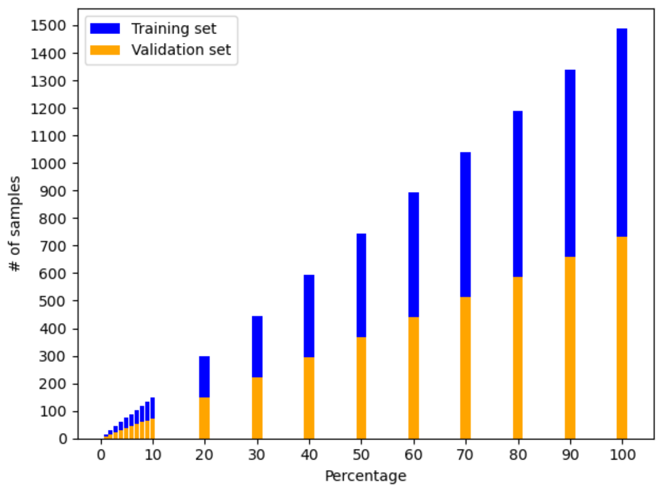


Figure 3.6: Number of training/validation samples obtained by linearly decreasing the amount of positive samples used.

Summary of the SSL framework
1. Design a pretext task that makes use of the data at your disposal;
2. Self-label the data according to the pretext task;
3. Use the self-labelled dataset to train/validate/test a model m_1 to test;
4. Leverage model m_1 to perform the downstream task in different data regimes;
5. Compare the performances of the fully supervised baseline model and the initialized model.

Table 3.5: Summary of the steps described in Section 3.3.

3.4 Pretext Tasks

Two pretext tasks were experimented in this work: one adapted from the literature [14] and one fully designed by us. The first one is named *Relative Positioning*, as in its original work, and the second is *Channel Correlation*. For both pretext tasks we will go through the steps listed in Table 3.5.

3.4.1 Relative Positioning

This pretext task explores the temporal dimension of the EEG data by defining a positive and a negative context around an EEG epoch, assuming that the signal evolves slowly over time and that epochs closer in time will be closer in value as well. The original downstream task for this pretext task was sleep stage classification, which encompasses neurophysiological phenomena at a greater scale than epilepsy-related abnormalities (a sleep stage lasts much longer than an IED). For that reason, the duration of the positive context was adjusted. We defined the positive context's length, τ_{pos}, as 6s (equal to 3 EEG epochs). The negative context includes every epoch that is not included in the positive context.

3.4.1.1 Task Definition

The goal is to have as input 2 epochs (a pair) and predict whether they belong in the same positive context ($y = 1$) or not ($y = 0$). Thus, this pretext task is a binary classification problem.

3.4.1.2 Dataset Labelling

In order to self-label the data, every epoch of every EEG was iterated. At each iteration, the epoch in question would be an anchor, x_t, which means the positive context was generated considering the anchor the central point. The positive and negative context around an anchor can be seen in Figure 3.7. All epochs belonging to the positive context of the anchor would be retrieved. These are its consecutive epochs (x_{t-1} and x_{t+1}), therefore, a maximum of two epochs in the positive context would be retrieved (only one if the anchor was the first or the last epoch in the EEG). The pairs (x_t, x_{t+1}) and (x_{t-1}, x_t) would be added to the input set (X), while the labels 1 would be added to the output set (y). Following this, 1 to 3 other epochs belonging to the anchor's negative context would be retrieved randomly. For every epoch $x_{t'}$ retrieved, a pair (x_t, $x_{t'}$) or (x_t, $x_{t'}$) (depending

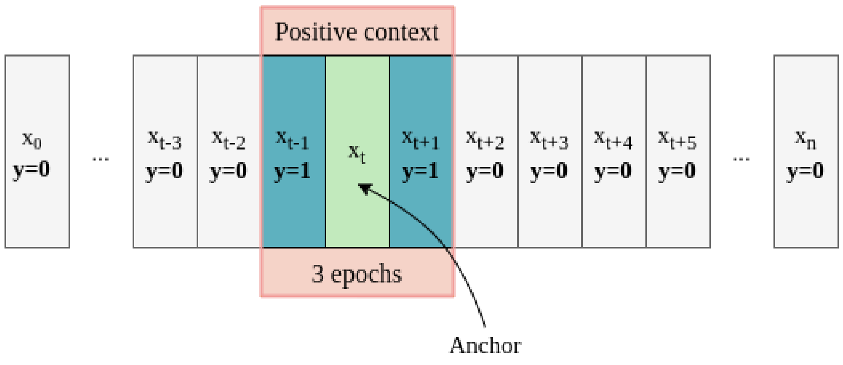


Figure 3.7: The positive context around an anchor epoch, x_t. The y value in each epoch represents the resulting label of paring the anchor with that epoch.

on the correct order) would be added to X whereas the label 0 would be added to y. After going through every epoch and every EEG, the sets X and y were saved in `.npy` files for posterior use.

Some problems arose due to the large volume of pairs created (in total, 72GB). There was not enough RAM to deal with this amount of data, and the same applies to the GPU card (even with reduced batch sizes). Some experiments were done in order to determine how much of the data we had could be used. These experiments are detailed in Appendix A.

Eventually, it was assessed that 150 EEGs represented a good balance: $\sim 91\%$ of the total EEGs and $\sim 93\%$ of the total IEDs were being used, while the memory savings were about 50%. Still, the batch size had to be 16, and each training epoch took, on average, 1291 seconds (21.52 min).

The resulting dataset was balanced (due to the pair generation algorithm). The samples were split into training/validation/test sets with approximate proportions of 60%/20%/20%, making sure data from one patient was only present in a single set. The distribution of samples per set and category is presented in Table 3.6.

3.4.1.3 Model

To perform this task, we used Siamese Neural Networks, which are networks that consist of two identical subnetworks. These subnetworks share the same weights, so whenever one is updated, both of them are updated with the same values. They are widely used in tasks where one would

	Label		
Set	0	1	Total
Training	172 900	172 878	345 778 (~ 61%)
Validation	57 280	57 196	114 476 (~ 20%)
Test	51 288	51 274	102 562 (~ 18%)
Total	281 468 (~ 50%)	281 348 (~ 50%)	562 816

Table 3.6: Data distribution for the Relative Positioning pretext task.

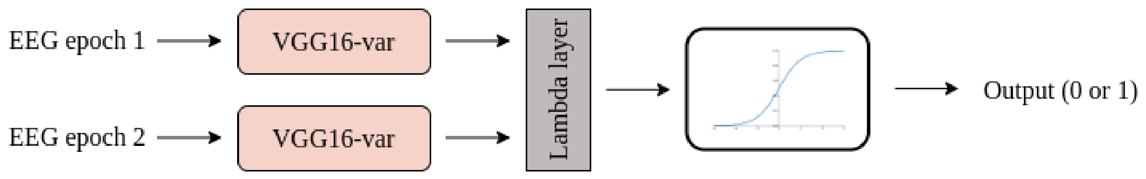


Figure 3.8: Siamese model used to perform the Relative Positioning task.

want to match two samples because of their similarity, e.g. signature verification or facial recognition.

A siamese neural network with two identical VGG16-var subnetworks (already described in Section 3.2) was used. Each VGG receives as input one of the epochs of the input pair, and a lambda layer connects the output of the two subnetworks by computing the euclidean distance between the output features of each subnetwork. Again, sigmoid activation was used in the output layer to produce a binary output. Figure 3.8 portrays a depiction of this model.

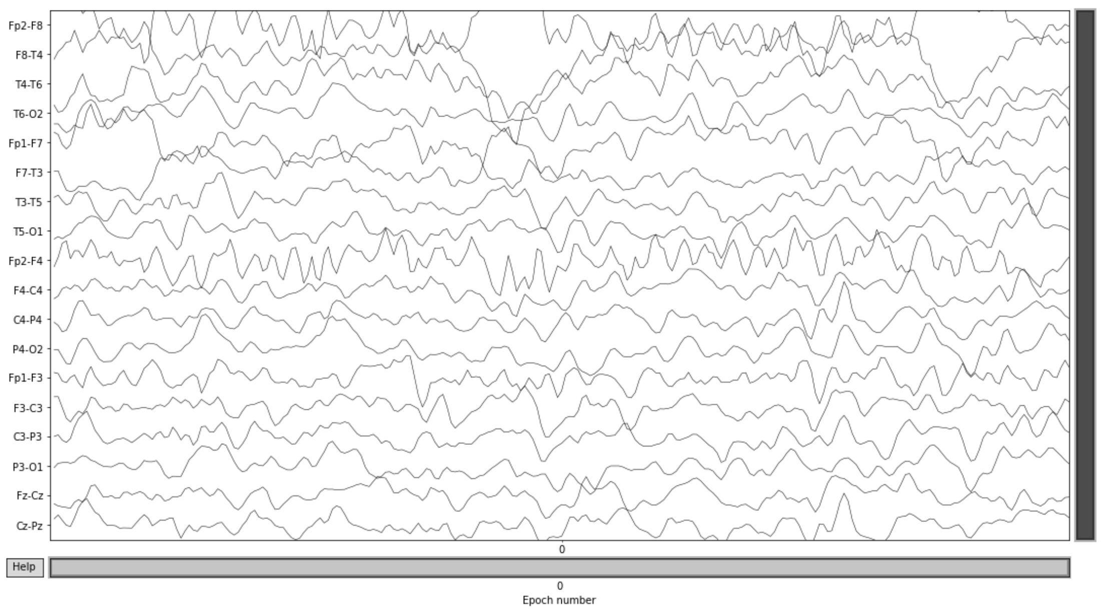
The model was trained for 24 epochs (which took approximately 19 hours due to the problems already mentioned) with a batch size of 16. The Adam optimizer was used with a learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$. The loss function employed was binary cross-entropy.

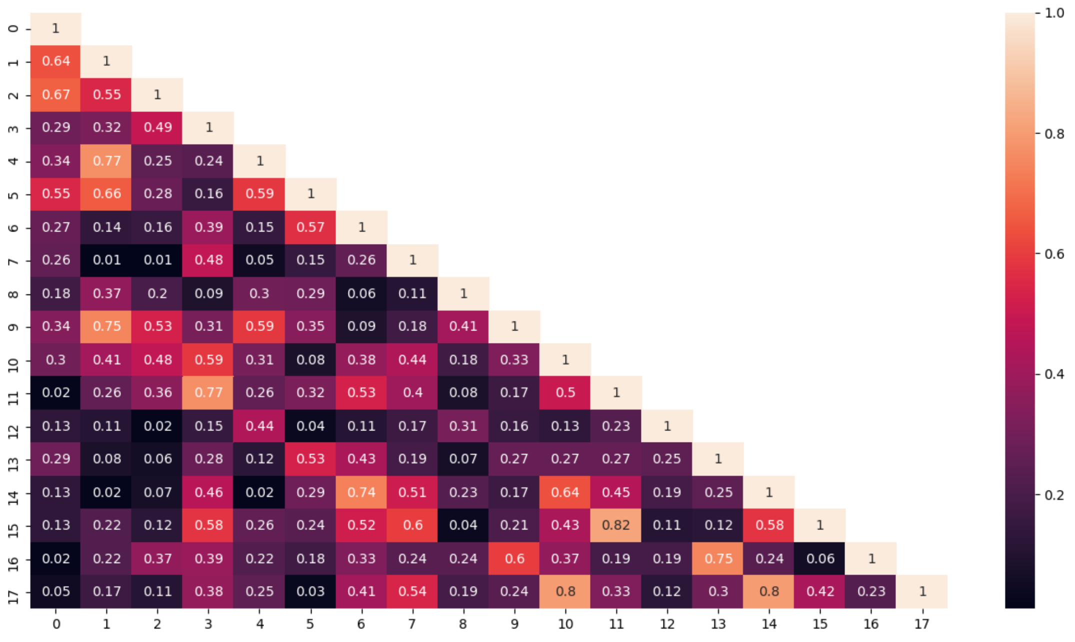
3.4.2 Channel Correlation

In the previous pretext task, we explored the temporal dimension of the EEG data. In this task, we aimed to explore the spatial dimension by analysing the correlation between channels. We expected to see a higher channel correlation in controls and patients with generalized epilepsy in comparison to patients with focal epilepsy, where IEDs occur in a few channels only. Only a subset of F and G were used for this task: the epochs containing IEDs. These subsets were named F_select and G_select, respectively. This decision was made after analysing the results from the previous pretext task, which will be discussed later, that strongly hinted that the baseline model was highly dependent on positive samples.

We started by computing the average Pearson correlation between channels for each set of EEGs (F_select, G_select and N). We did this by computing the correlation matrix for every epoch. The absolute values were considered because we were only interested in the magnitude of the correlation. This matrix is symmetric, so only half was saved, excluding the diagonal, which is only made up of 1's (an example of an EEG epoch and its corresponding correlation values can be seen in Figure 3.9). The mean of these values was taken as the correlation mean for that epoch. The EEG mean was, in turn, the average of all the epoch means. We obtained the following results:

As can be seen in Figure 3.10 the values were significantly overlapped. However, F_select and G_select seem to have less overlapping, so we decided to base the task off of these two sets.





	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1																	
1	0.64	1																
2	0.67	0.55	1															
3	0.29	0.32	0.49	1														
4	0.34	0.77	0.25	0.24	1													
5	0.55	0.66	0.28	0.16	0.59	1												
6	0.27	0.14	0.16	0.39	0.15	0.57	1											
7	0.26	0.01	0.01	0.48	0.05	0.15	0.26	1										
8	0.18	0.37	0.2	0.09	0.3	0.29	0.06	0.11	1									
9	0.34	0.75	0.53	0.31	0.59	0.35	0.09	0.18	0.41	1								
10	0.3	0.41	0.48	0.59	0.31	0.08	0.38	0.44	0.18	0.33	1							
11	0.02	0.26	0.36	0.77	0.26	0.32	0.53	0.4	0.08	0.17	0.5	1						
12	0.13	0.11	0.02	0.15	0.44	0.04	0.11	0.17	0.31	0.16	0.13	0.23	1					
13	0.29	0.08	0.06	0.28	0.12	0.53	0.43	0.19	0.07	0.27	0.27	0.27	0.25	1				
14	0.13	0.02	0.07	0.46	0.02	0.29	0.74	0.51	0.23	0.17	0.64	0.45	0.19	0.25	1			
15	0.13	0.22	0.12	0.58	0.26	0.24	0.52	0.6	0.04	0.21	0.43	0.82	0.11	0.12	0.58	1		
16	0.02	0.22	0.37	0.39	0.22	0.18	0.33	0.24	0.24	0.6	0.37	0.19	0.19	0.75	0.24	0.06	1	
17	0.05	0.17	0.11	0.38	0.25	0.03	0.41	0.54	0.19	0.24	0.8	0.33	0.12	0.3	0.8	0.42	0.23	1

Figure 3.9: An epoch from set F (top) and its corresponding lower triangle of the correlation matrix (bottom).

EEG set	Average correlation
F_select	0.3449 ± 0.0584
G_select	0.4393 ± 0.0683
N	0.3805 ± 0.0615

Table 3.7: Average channel correlation for each EEG set.

3.4.2.1 Task Definition

We used the value of the intersection of both probability density functions (see Figure 3.11) as a cutoff value for a binary classification task. Thus, the input for this classification task is an EEG epoch, and the output should be 1 if the channel correlation mean is greater or equal to the cutoff value (0.382) and 0 otherwise.

3.4.2.2 Dataset Labelling

Labelling the EEG data according to this pretext task was only a matter of iterating through all EEG epochs from F_select and G_select, computing their Pearson correlation mean (as mentioned before with the absolute values of the correlation matrix) and labelling them accordingly.

The self-labelled data was split into training/validation/test sets with approximate proportions of 60%/20%/20% ensuring data from one patient was only present in a single set. The resulting dataset is presented in Table 3.8.

3.4.2.3 Model

The model architecture used was the same as the baseline model already presented in Section 3.2 (VGG16-var). The model was trained for 10 epochs with a batch size of 64. The Adam optimizer was used with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-7}$, and a learning rate of 2×10^{-5} for the first 7 epochs and 1×10^{-5} for the last 3 epochs since the performance was plateauing. The loss function employed was binary cross-entropy.

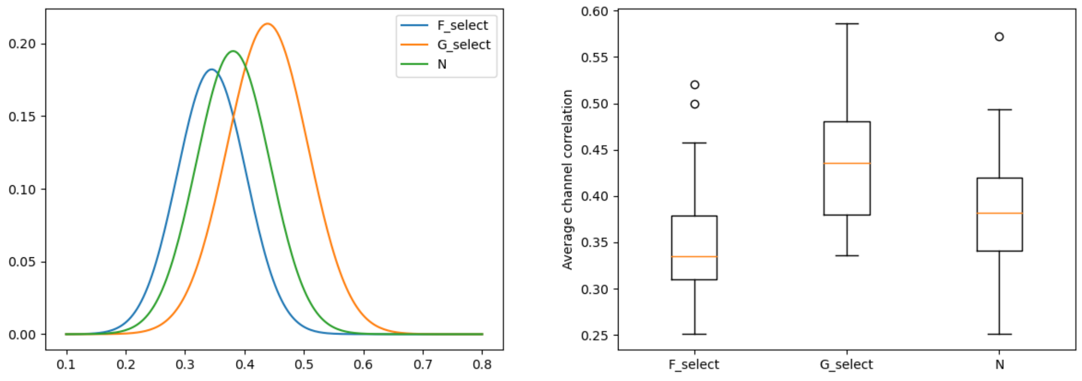


Figure 3.10: Gaussian distribution and box plots of the EEG correlation means for each set.

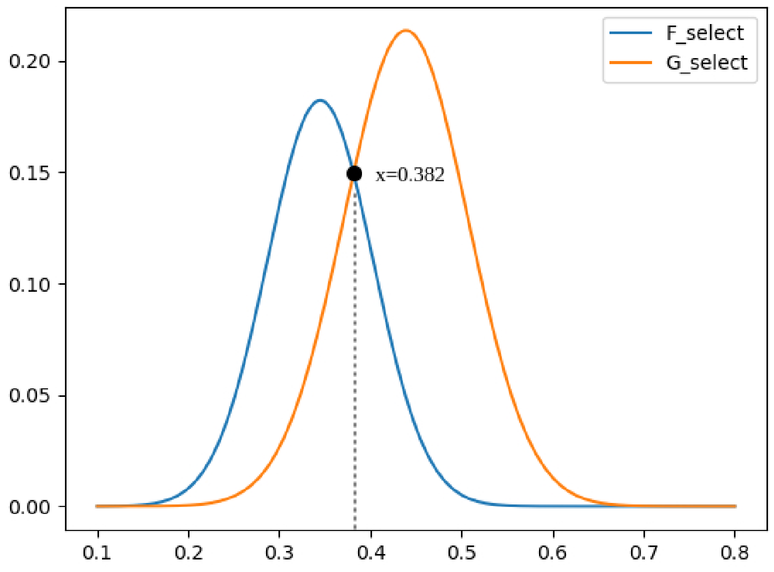


Figure 3.11: Gaussian distribution of the EEG correlation means for the F_select and G_select sets. The intersection of both was used as a cutoff value in order to create a binary classification task.

	Label		
Set	0	1	Total
Training	826	536	1 362 ($\sim$ 49%)
Validation	311	202	513 ($\sim$ 19%)
Test	521	362	883 ($\sim$ 32%)
Total	1 658 ($\sim$ 60%)	1 100 ($\sim$ 40%)	2 758

Table 3.8: Data distribution for the Channel Correlation pretext task.

Chapter 4

Results

In the previous chapter, we described the methods developed in this work. The purpose of the current chapter is to present the results obtained by carrying out said methods. All the sensitivity/specificity pairs presented are the ones where $specificity = 99\%$ or $sensitivity = specificity$.

4.1 Pretext Tasks

In Table 4.1 a performance summary of the siamese model for the Relative Positioning task on the training/validation/test sets can be seen. Additionally, the ROC curve on the test set is shown in Figure 4.1. The same information for the VGG16-var model on the Channel Correlation task is presented in Table 4.2 and Figure 4.2.

4.2 Downstream Task

Similarly, Table 4.3 and Figure 4.3 provide an overview of the baseline VGG16-var model's performance on the downstream task (IED detection).

As mentioned before (refer to Section 3.3), the baseline model was trained on different data regimes in order to assess the contribution of the SSL approach when using a limited amount of annotated data. The following sections showcase the results of each of the two experiments done.

Set	AUC	Sensitivity	Specificity
Training	0.97	45.80%	99.00%
		90.90%	90.90%
Validation	0.84	4.70%	99.00%
		75.80%	75.80%
Test	0.83	5.00%	99.00%
		74.70%	74.70%

Table 4.1: Model performance for the Relative Positioning task.

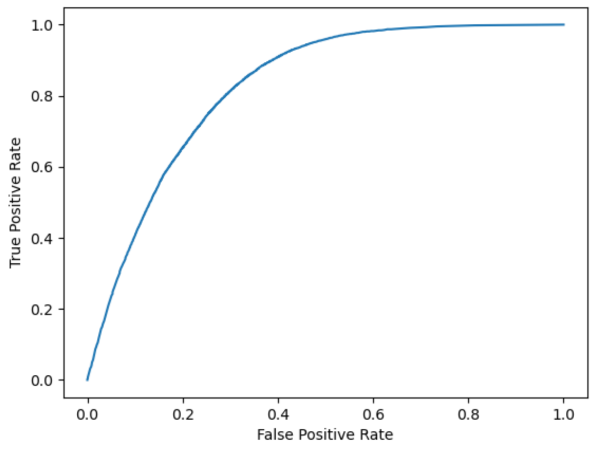


Figure 4.1: ROC curve of the siamese model (RP task) on the test set.

Set	**AUC**	**Sensitivity**	**Specificity**
Training	0.97	76.00%	99.00%
		90.00%	90.00%
Validation	0.83	35.60%	99.00%
		76.00%	76.00%
Test	0.92	42.80%	99.00%
		83.70%	83.70%

Table 4.2: Model performance for the Channel Correlation task.

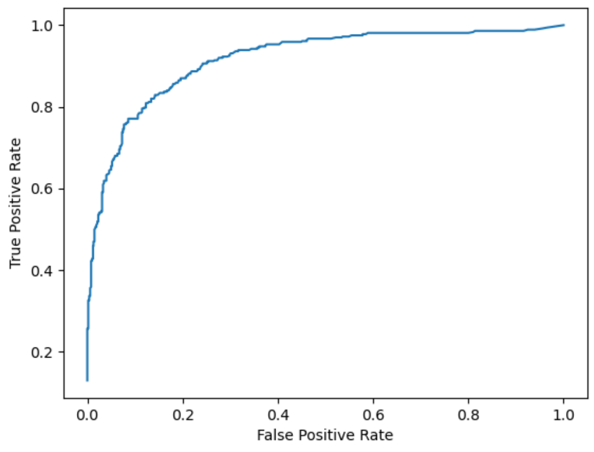


Figure 4.2: ROC curve of the VGG16-var model (CC task) on the test set.

Set	**AUC**	**Sensitivity**	**Specificity**
Training	1.00	100.00%	99.00%
		100.00%	100.00%
Validation	0.99	89.09%	99.00%
		96.45%	96.82%
Test	0.96	65.00%	99.00%
		91.00%	90.00%

Table 4.3: Model performance for the downstream task. Note that for the validation and test sets it was not possible to find a threshold for which $sensitivity = specificity$. Instead, the closest two values were used.

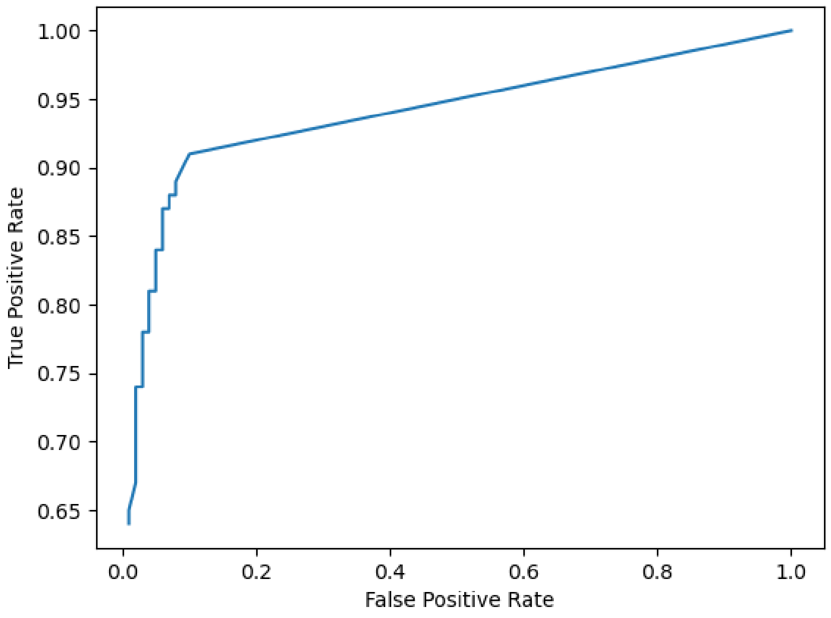


Figure 4.3: ROC curve of the baseline VGG16-var model (downstream task) on the test set.

Percentage of available data used	**Random weights**	**Initialized weights (RP)**	**Initialized weights (CC)**
100%	0.96	0.92	0.97
90%	0.96	0.92	0.97
80%	0.94	0.91	0.97
70%	0.95	0.91	0.96
60%	0.94	0.90	0.95
50%	0.93	0.91	0.94
40%	0.93	0.88	0.93
30%	0.88	0.77	0.89
20%	0.87	0.75	0.84
10%	0.91	0.81	0.86
9%	0.82	0.80	0.84
8%	0.84	0.74	0.90
7%	0.81	0.71	0.88
6%	0.80	0.66	0.80
5%	0.78	0.63	0.77
4%	0.82	0.54	0.80
3%	0.80	0.56	0.80
2%	0.67	0.54	0.65
1%	0.69	0.52	0.65

Table 4.4: AUC obtained by the baseline model on the test set by varying the amount of data available for training/validation and the weight initialization method. Note that the values from 1 to 10% are average values.

4.2.1 Varying the Amount of Data

For this experiment, the baseline VGG16-var model was trained with different settings and weights. The model's weights were initialized either randomly, with the RP model's weights or with the CC model's weights. The training/validation procedure was repeated for all percentages enumerated before and for all weight initialization methods. The results (AUC on the test set) are illustrated in Table 4.4 and Figure 4.4. The training/validation cycle was repeated 10 times for the percentages 1–10% and, as such, the AUC values presented in the aforementioned table in that range are then the mean of all 10 runs. Refer to Figure 4.5 for the standard deviations. The full results of this experiment (including the AUC on the training/validation sets and the test AUC for all 10 runs) as well as the next one are included in Appendix C.

4.2.2 Varying the Number of Positive Samples

Likewise, for this experiment, the baseline model was trained with different settings and weights. The model's weights were initialized either randomly, with the RP model's weights or with the CC model's weights. Again, the training/validation procedure was repeated for all percentages enumerated before and for all weight initialization methods, with the addition of the RP + CC

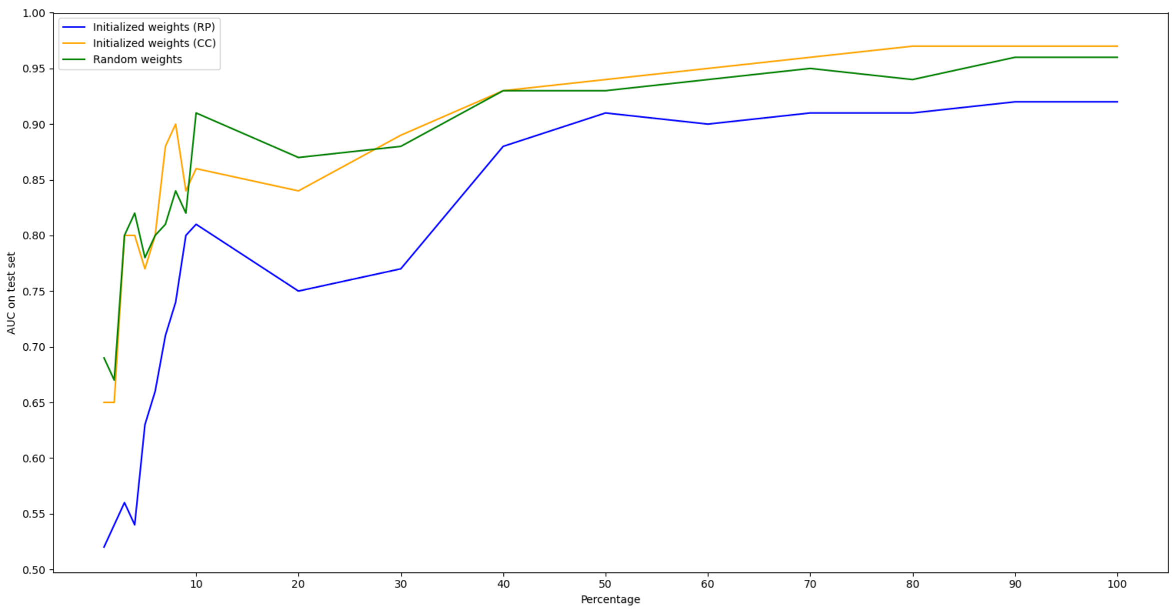


Figure 4.4: AUC of the baseline model on the test set for every percentage of available data used. Note that the values from 1 to 10% are average values.

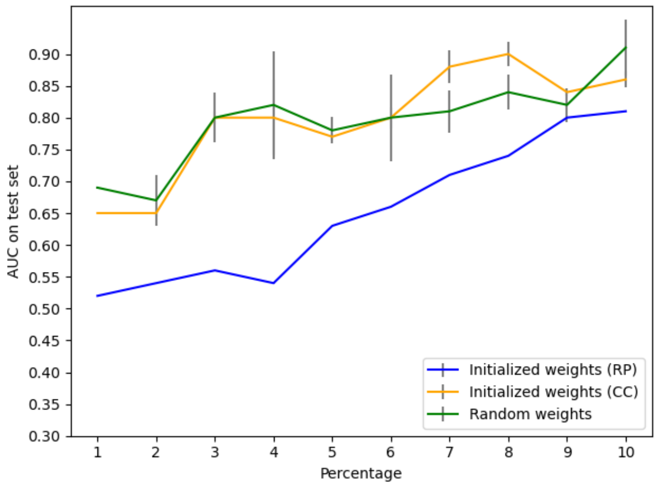


Figure 4.5: Mean AUC of the baseline model on the test set, using 1 to 10% of all available data, including the standard deviation (in grey) for every percentage. Random, RP and CC weight initialization were tried.

Percentage of positive samples used	**Random weights**	**Initialized weights (RP)**	**Initialized weights (CC)**	**Initialized weights (RP + CC)**
100%	0.96	0.92	0.97	0.90
90%	0.97	0.92	0.97	0.90
80%	0.97	0.92	0.97	0.90
70%	0.96	0.92	0.96	0.92
60%	0.96	0.92	0.96	0.92
50%	0.94	0.93	0.95	0.90
40%	0.92	0.91	0.93	0.86
30%	0.90	0.90	0.90	0.82
20%	0.86	0.87	0.88	0.84
10%	0.81	0.84	0.85	0.84
9%	0.82	0.85	0.82	0.83
8%	0.83	0.84	0.83	0.85
7%	0.80	0.83	0.82	0.85
6%	0.79	0.82	0.80	0.85
5%	0.78	0.82	0.78	0.82
4%	0.42	0.81	0.84	0.80
3%	0.77	0.80	0.80	0.71
2%	0.50	0.69	0.45	0.68
1%	0.50	0.63	0.46	0.60

Table 4.5: AUC obtained by the baseline model on the test set by varying the amount of positive samples available for training/validation and the weight initialization method. Note that the values from 1 to 10% are average values.

initialization. The results (AUC on the test set) are illustrated in Table 4.5 and Figure 4.6. Once more, the training/validation cycle was repeated 10 times for the percentages 1–10% and, as such, the AUC values presented in the aforementioned table in that range are the mean of all 10 runs. A more detailed look into the 1–10% range is shown in Figure 4.7.

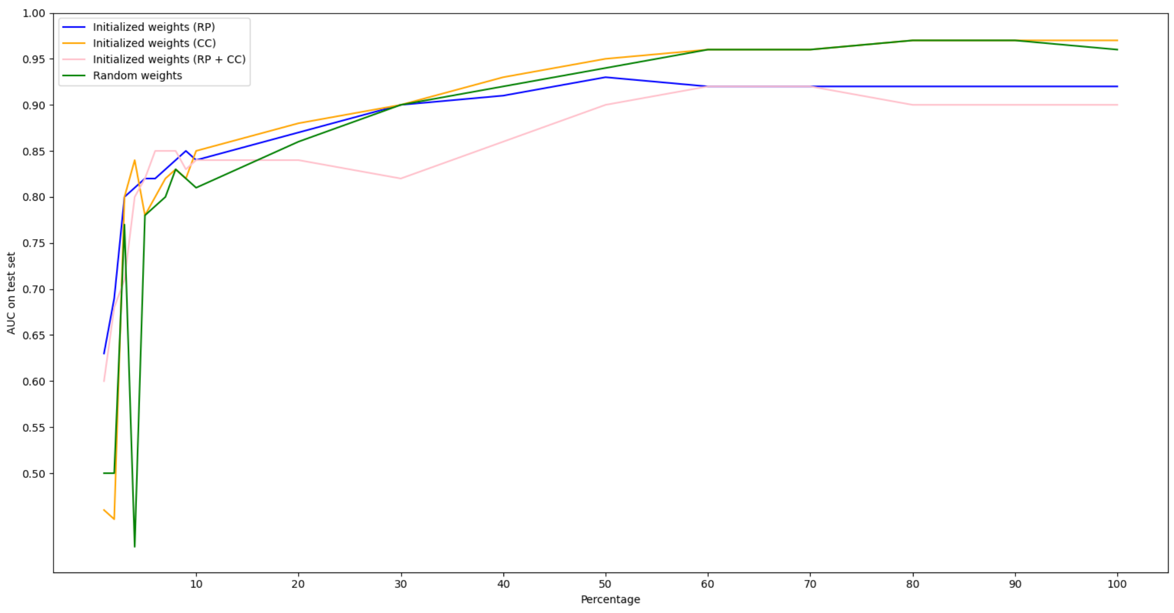


Figure 4.6: AUC of the baseline model on the test set for every percentage of positive samples used. Note that the values from 1 to 10% are average values.

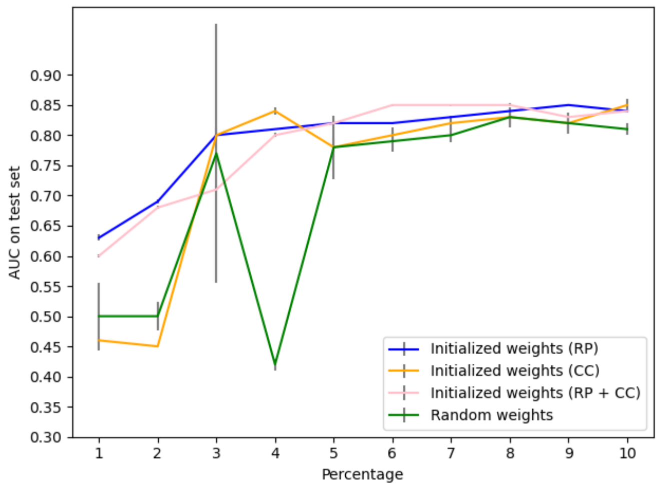


Figure 4.7: Mean AUC of the baseline model on the test set, using 1 to 10% of all available positive samples, including the standard deviation (in grey) for every percentage. Random, RP, CC and RP + CC weight initialization were tried.

Chapter 5

Discussion

The aim of this dissertation was to prove that SSL can make up for the lack of annotated EEG data. To that end, two pretext tasks were created: Relative Positioning (RP) and Channel Correlation (CC). The models developed to perform these tasks were then used to initialize the weights of a baseline model (VGG16-var). This baseline model was trained/validated/tested under different data regimes and with different weight initialization (random initialization, RP initialization and CC initialization) to assess the performance gain that pretext tasks could produce. In the present chapter, the results from both pretext tasks and the downstream tasks are discussed, with a greater focus on the latter.

5.1 Pretext Tasks

The siamese model (RP task) had decent performance. However, it could benefit from improvement. It is possible that the distance measure computed in the lambda layer (euclidean distance) is not the best fit for this task, since an IED usually lasts for less than an epoch, and its shape is very distinct from other EEG activity. Therefore, this sudden change could lead the model into misclassifying a pair of consecutive epochs as non-consecutive when one of the epochs contains an IED. It is worth mentioning that while in [14] the RP model performs better, the nature of the EEG data was quite different since the downstream task was sleep stage classification. Sleep stages last for much more than an epoch, and the EEG signal evolves slowly over time, assuming that the patient possesses no brain conditions. This may be one of the reasons this model did not do exceptionally well on this task. Nonetheless, the model performance on the test set was reasonably good.

The VGG16-var model for the CC task, however, was quite good. With good results on every set, this model learned very quickly (the training/validation cycle lasted for 10 epochs, with each epoch lasting roughly 10 seconds). When we compare it with the previous task, it is even more impressive to see that this model had better performance because there was far more data available for the RP task than for the CC task. However, since this task was designed in collaboration with a neurologist and inspired by what a real-life lab technician intuitively does (compare the activity in different channels), it may be that this domain knowledge may have been translated to a classification task more fit to the data we used.

5.2 Downstream Task

For simplicity purposes, we will refer to the baseline model with random weight initialization as BL-RW, to the baseline model with RP model's weights as BL-RP and to the baseline model with CC model's weights as BL-CC.

The results of the BL-RW in the IED detection task were very good across all sets. When we start reducing the amount of data (either globally or positive samples only), the performance is still quite stable (at 30% the model AUC on the test set was around 0.9 for both experiments). For that reason, the data regimes with less labels (1–10%) will be more deeply analysed, since it is the focus of this work as well as where there is more potential for a performance gain.

5.2.1 Varying the Amount of Data

The RP initialization actually worsened the baseline performance in this experiment. The BL-RP was outperformed by both the BL-RW and BL-CC at every point (as can be seen in Figure 4.4) and, most of the times, with a significant difference. The reason for this may be that the RP model was not good enough or that the features it learned for the pretext task were not relevant enough for the downstream task. It is also interesting to note that the standard deviation of the performance of the BL-RP model was ≈ 0 at every point.

From 30 to 100%, the BL-CC was either better or as good as the BL-RW. On the other hand, the BL-CC model had equal or worse performance than the BL-RW at 7 out of the 10 points in the low-labelled data regimes, as illustrated in Figure D.1d. Moreover, the standard deviation of the performance is distinctively higher at some points for both the BL-RW and the BL-CC.

In short, the BL-CC was marginally better than the BL-RW for higher-labelled data regimes, however, in low-labelled data regimes (1–10%), the BL-RW did not seem to benefit much from either pretext task. Nonetheless, the shape of the graphs was very irregular and inconsistent, which does not allow us to draw significant conclusions.

5.2.2 Varying the Number of Positive Samples

For this experiment, an additional initialization was considered. In the Channel Correlation task the VGG16-var model was initialized with the RP model's weights, and then trained like before. The resulting weights were then used to initialize the baseline model. We will call this model BL-RP-CC.

The BL-RW was consistently outperformed either by the BL-RP or the BL-CC. Its performance deteriorated much more with this type of data variation (positive samples) than with the previous one. This is possibly due to the already unbalanced nature of the IED detection task, which was aggravated with this variation.

In low-labelled data regimes, the BL-RW was consistently outperformed by the BL-RP (see Figure D.2c). Nonetheless, and repeating the same pattern from before, the RP initialization was the worst option in higher-labelled data regimes (from 30% on). Again, this can be due to the fact that the RP model does not adapt well to this type of data (it is in fact the one with the worst performance on its original task) or that the pretext itself does not fit our downstream task, and does not push the model to learn enough physiologically meaningful features. Moreover, it is especially interesting to note that in low-labelled data regimes the RP initialization outperformed the CC initialization while for higher percentages it was the other way around.

Regarding the BL-RP-CC model, although it had the highest performance in 4 out of 10 points from 1–10%, it is hard to say whether the RP initialization benefit the CC model or not because although the BL-RP-CC was generally better than the BL-CC in low-labelled data regimes, it was much worse in high-labelled data-regimes.

In both cases, it is hard to draw conclusions from higher percentages because the BL-RW has excellent and stable performance throughout most of those, and, as such, the potential performance gain is minimal. Besides that, all models converge to approximately the same AUC value, further making it impossible to make comparisons (which was to be expected for higher percentages).

5.2.3 Summary

The following table presents an overview of the appreciations made in this chapter.

	Low-labelled data regimes (1–10%)	**Higher-labelled data regimes (30–100%)**
Experiment 1	BL-RP far worse than the other models; BL-RW and BL-CC similar.	BL-RP worse than the other models; BL-RW and BL-CC very similar (maximum 0.03 difference).
Experiment 2	BL-RP better than BL-RW and BL-CC; BL-CC similar to BL-RW.	BL-RP worse than the other models; BL-RW and BL-CC very similar (maximum 0.01 difference).

Table 5.1: Summary of the appreciations made to the previously presented results. Note that *Experiment 1* refers to varying the amount of data while *Experiment 2* refers to varying the number of positive samples.

Chapter 6

Conclusions

This experimental approach hints that the self-supervised learning approach can indeed lead to a performance gain in scenarios where data is abundant but human annotations are scarce, and therefore the (labelled) dataset is limited.

However, we can also conclude that the context/domain is crucial. It is fundamental that the pretext tasks fit the downstream task. This can be done best with professionals from the respective area (e.g. in our Channel Correlation pretext task, we had the valuable help of Prof. Dr Michel Van Putten, a neurologist). It also seems to be essential that the models developed for the pretext tasks have good performance themselves. In fact, the Relative Positioning model seemed to be capped in higher-labelled data regimes, while in low-labelled data regimes it actually did very good in one of the experiments.

6.1 Future Work

Considering that in [14] it was concluded that "SSL pretext task hyper-parameters strongly influence downstream task performance", we believe that the following steps would include tuning the RP model. It was not possible to do that methodically due to the amount of time it took to train the network. However, it would be interesting to perform hyper-parameter tuning and then re-test it on the downstream task to see how it affects the performance gain. The idea of training the model with only IED epochs (as anchor epochs for generating the pairs) is also appealing since the CC model was trained only with IEDs and this would be much quicker to test.

Designing new pretext tasks and assessing their suitability is also another path for extending this work. One of those could build on the analysis of the channels symmetry (or lack of). That is to say, calculate the channel correlations between channels belonging to the left region of the brain and channels belonging to the right region of the brain. We could then analyse how correlated the EEG activity in both sides of the brain is and possibly design a pretext task like the CC task. For instance, when we think of focal epilepsy, it is possible that this correlation may be low if the origin of the epileptic activity lies in one of the temporal lobes, causing abnormal activity only on the channels of that lobe.

Another possible pretext task could be based on predicting the signal's autocorrelation or (serial correlation). This is the correlation of a signal or time-series data with a delayed copy of itself. Like the RP task, this explores the time dimension of the signal, however, it can be done in a much lower scale by adjusting how delayed the copy of the signal is, possibly mitigating the problem of IEDs being sudden.

Appendix A

Memory Experiments

As mentioned in Chapter 3, some problems arose due to the large volume of pairs created in the Relative Positioning task (in total, 72GB). There was not enough RAM to deal with this amount of data, and the same applies to the GPU card (even with reduced batch sizes). Three EEG files, in particular, contributed immensely to this problem because they were ambulatory recordings of approximately 16, 22 and 24 hours. The pairs/labels generated from these three files represented 41GB of data. Considering that they were not very relevant (they did not contain a significant amount of IEDs), they were removed. Even so, there were memory problems still, and some experiments were conducted.

For these experiments, we used a successively higher number of EEGs to generate the self-labelled dataset, from 10 up to 165 EEGs (total number of available EEGs). At each iteration, we would generate the self-labelled dataset for the RP task with the EEGs available at that point, split them into training/validation/test sets and run the siamese model for 5 epochs to assess 1) if it would run at all 2) how much time it would take per epoch, on average. The results of these experiments is summarized in Table A.1. The meaning of each table column is the following:

Number of EEGs used Number of EEG files used to generate the pairs;

Number of training pairs Number of resulting training pairs;

Number of validation pairs Number of resulting validation pairs;

Seconds per epoch Average duration of an epoch;

Epochs Total amount of epochs that the network trained for;

Batch size Batch size used in the training/validation cycle;

Execution time Total execution time in minutes;

Data preparation time Amount of time (in minutes) it took to prepare the pairs (load the array files, split them into training/validation/test and concatenate them);

Ran without problems? Whether the script completed without throwing memory errors.

No. of EEGs used	No. of train. pairs	No. of val. pairs	Seconds per epoch	Epochs	Batch size	Execution time	Data prep time	Ran w/o problems?
10	13 401	4 832	23	10	32	4.1	0.27	Yes
20	28 692	9 093	50	10	32	8.87	0.54	Yes
30	58 094	14 657	100	10	32	17.48	0.81	Yes
40	74 572	29 050	131	5	32	11.97	1.05	Yes
50	105 021	28 363	181	5	32	15.65	0.57	Yes
60	106 738	47 144	189	5	32	17.13	1.38	Yes
70	140 612	41 293	243	5	32	22.25	2.00	Yes
80	146 043	60 271	258	5	32	24.35	2.85	Yes
90	282 146	66 594	485	5	32	47.96	7.55	Yes
100	213 213	240 723	600	5	32	59.48	9.48	Yes
110	378 610	75 011	811	5	32	78.09	10.50	Yes
120	-	-	-	-	32	-	-	No
After deleting the 3 large EEGs								
120	288 053	69 617	496	5	32	49.36	7.78	Yes
130	274 919	124 596	500	5	32	50.83	8.76	Yes
140	310 829	102 256	561	5	32	55.91	8.93	Yes
150	-	-	-	-	32	-	-	No
150	345 778	114 476	1 291	5	16	118.34	10.44	Yes
155	351 680	132 870	1 515	5	16	137.95	11.38	Yes
160	385 145	103 147	-	1	16	-	11.88	No
165	347 854	113 376	-	-	10	-	11.52	No
165	347 854	113 376	3 108	1	5	63.56	11.53	Yes

Table A.1: Summary of the experiments made assessing memory issues.

Appendix B

Supplementary Tables of Chapter 3 - Methods

Percentage	**# of training samples**	**# of validation samples**	Total
100%	14 870	7 330	22 200
90%	13 383	6 597	19 980
80%	11 896	5 864	17 760
70%	10 409	5 131	15 540
60%	8 922	4 398	13 320
50%	7 435	3 665	11 100
40%	5 948	2 932	8 880
30%	4 461	2 199	6 660
20%	2 974	1 466	4 440
10%	1 487	733	2 220
9%	1 338	660	1 998
8%	1 190	586	1 776
7%	1 041	513	1 554
6%	892	440	1 332
5%	744	367	1 110
4%	595	293	888
3%	446	220	666
2%	297	147	444
1%	149	73	222

Table B.1: Number of training/validation samples obtained by varying the percentage of available data used.

Percentage	**# of positive samples (training)**	**# of positive samples (validation)**	Total
100%	1 487	733	2 220
90%	1 338	660	1 998
80%	1 190	586	1 776
70%	1 041	513	1 554
60%	892	440	1 332
50%	744	367	1 110
40%	595	293	888
30%	446	220	666
20%	297	147	444
10%	149	73	222
9%	134	66	200
8%	119	59	178
7%	104	51	155
6%	89	44	133
5%	74	37	111
4%	59	29	89
3%	45	22	67
2%	30	15	44
1%	15	7	22

Table B.2: Number of positive samples in training/validation obtained by varying the percentage of available positive samples used.

Appendix C

Supplementary Tables of Chapter 4 - Results

Percentage of available data used	Training	Validation	Test
100%	1.00	0.99	0.96
90%	1.00	0.99	0.96
80%	1.00	0.98	0.94
70%	1.00	0.98	0.95
60%	1.00	0.98	0.94
50%	1.00	0.98	0.93
40%	1.00	0.99	0.93
30%	1.00	0.97	0.88
20%	1.00	0.96	0.87
10%	1.00	0.95	0.91
9%	1.00	0.95	0.82
8%	1.00	0.99	0.84
7%	1.00	0.97	0.81
6%	1.00	0.95	0.80
5%	1.00	0.95	0.78
4%	1.00	0.96	0.82
3%	1.00	0.95	0.80
2%	1.00	0.80	0.67
1%	1.00	0.87	0.69

Table C.1: AUC of the BL-RW on the training, validation and test sets for experiment 1. Note that the values from 1 to 10% are average values.

Percentage of available data used	**Training**	**Validation**	**Test**
100%	1.00	0.97	0.92
90%	1.00	0.96	0.92
80%	1.00	0.97	0.91
70%	1.00	0.96	0.91
60%	1.00	0.95	0.90
50%	1.00	0.96	0.91
40%	1.00	0.96	0.88
30%	1.00	0.93	0.77
20%	1.00	0.92	0.75
10%	0.98	0.90	0.81
9%	0.98	0.93	0.80
8%	0.97	0.89	0.74
7%	0.96	0.85	0.71
6%	0.96	0.81	0.66
5%	0.95	0.78	0.63
4%	0.93	0.75	0.54
3%	0.94	0.64	0.56
2%	0.94	0.63	0.54
1%	0.96	0.56	0.52

Table C.2: AUC of the BL-RP on the training, validation and test sets for experiment 1. Note that the values from 1 to 10% are average values.

Percentage of available data used	**Training**	**Validation**	**Test**
100%	1.00	0.98	0.97
90%	1.00	0.98	0.97
80%	1.00	0.98	0.97
70%	1.00	0.98	0.96
60%	1.00	0.98	0.95
50%	1.00	0.97	0.94
40%	1.00	0.98	0.93
30%	1.00	0.97	0.89
20%	1.00	0.93	0.84
10%	1.00	0.95	0.86
9%	1.00	0.96	0.84
8%	1.00	0.99	0.90
7%	1.00	0.99	0.88
6%	1.00	0.97	0.80
5%	1.00	0.96	0.77
4%	1.00	0.95	0.80
3%	1.00	0.92	0.80
2%	1.00	0.82	0.65
1%	1.00	0.99	0.65

Table C.3: AUC of the BL-CC on the training, validation and test sets for experiment 1. Note that the values from 1 to 10% are average values.

Percentage of positive samples used	**Training**	**Validation**	**Test**
100%	1.00	0.99	0.96
90%	1.00	0.98	0.97
80%	1.00	0.98	0.97
70%	1.00	0.98	0.96
60%	1.00	0.98	0.96
50%	1.00	0.97	0.94
40%	1.00	0.97	0.92
30%	1.00	0.97	0.90
20%	1.00	0.99	0.86
10%	1.00	0.97	0.81
9%	1.00	0.97	0.82
8%	1.00	0.95	0.83
7%	1.00	0.94	0.80
6%	1.00	0.93	0.79
5%	1.00	0.92	0.78
4%	0.41	0.41	0.42
3%	1.00	0.91	0.77
2%	0.50	0.50	0.50
1%	0.50	0.50	0.50

Table C.4: AUC of the BL-RW on the training, validation and test sets for experiment 2. Note that the values from 1 to 10% are average values.

Percentage of positive samples used	**Training**	**Validation**	**Test**
100%	1.00	0.97	0.92
90%	1.00	0.97	0.92
80%	1.00	0.96	0.92
70%	1.00	0.98	0.92
60%	1.00	0.98	0.92
50%	1.00	0.97	0.93
40%	1.00	0.96	0.91
30%	1.00	0.95	0.90
20%	1.00	0.96	0.87
10%	1.00	0.95	0.84
9%	1.00	0.95	0.85
8%	1.00	0.95	0.84
7%	1.00	0.95	0.83
6%	1.00	0.92	0.82
5%	1.00	0.90	0.82
4%	1.00	0.91	0.81
3%	1.00	0.87	0.80
2%	1.00	0.77	0.69
1%	1.00	0.63	0.63

Table C.5: AUC of the BL-RP on the training, validation and test sets for experiment 2. Note that the values from 1 to 10% are average values.

Percentage of positive samples used	**Training**	**Validation**	**Test**
100%	1.00	0.98	0.97
90%	1.00	0.98	0.97
80%	1.00	0.98	0.97
70%	1.00	0.98	0.96
60%	1.00	0.98	0.96
50%	1.00	0.97	0.95
40%	1.00	0.97	0.93
30%	1.00	0.98	0.90
20%	1.00	0.99	0.88
10%	1.00	0.97	0.85
9%	1.00	0.96	0.82
8%	1.00	0.95	0.83
7%	1.00	0.95	0.82
6%	1.00	0.94	0.80
5%	1.00	0.92	0.78
4%	1.00	0.91	0.84
3%	1.00	0.92	0.80
2%	0.44	0.44	0.45
1%	0.44	0.45	0.46

Table C.6: AUC of the BL-CC on the training, validation and test sets for experiment 2. Note that the values from 1 to 10% are average values.

Percentage of positive samples used	**Training**	**Validation**	**Test**
100%	1.00	0.95	0.90
90%	1.00	0.95	0.90
80%	1.00	0.97	0.90
70%	1.00	0.98	0.92
60%	1.00	0.98	0.92
50%	1.00	0.96	0.90
40%	1.00	0.94	0.86
30%	1.00	0.90	0.82
20%	1.00	0.91	0.84
10%	1.00	0.90	0.84
9%	1.00	0.88	0.83
8%	1.00	0.93	0.85
7%	1.00	0.93	0.85
6%	1.00	0.91	0.85
5%	1.00	0.89	0.82
4%	1.00	0.84	0.80
3%	1.00	0.69	0.71
2%	1.00	0.73	0.68
1%	1.00	0.59	0.60

Table C.7: AUC of the BL-RP-CC on the training, validation and test sets for experiment 2. Note that the values from 1 to 10% are average values.

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Mean	SD
BL-RW												
10%	0.91	0.91	0.89	0.91	0.90	0.91	0.92	0.91	0.91	0.77	0.91	0.0439
9%	0.84	0.83	0.78	0.76	0.81	0.81	0.82	0.83	0.79	0.84	0.82	0.0268
8%	0.84	0.88	0.83	0.83	0.79	0.82	0.86	0.86	0.88	0.83	0.84	0.0274
7%	0.87	0.82	0.77	0.87	0.83	0.79	0.80	0.80	0.80	0.81	0.81	0.0331
6%	0.79	0.78	0.80	0.71	0.77	0.83	0.87	0.91	0.93	0.78	0.80	0.0684
5%	0.77	0.81	0.79	0.80	0.77	0.75	0.77	0.76	0.79	0.81	0.78	0.0210
4%	0.87	0.80	0.69	0.70	0.91	0.80	0.90	0.94	0.84	0.81	0.82	0.0848
3%	0.89	0.79	0.80	0.78	0.78	0.81	0.79	0.80	0.87	0.81	0.80	0.0390
2%	0.73	0.68	0.66	0.75	0.66	0.76	0.66	0.70	0.67	0.66	0.67	0.0398
1%	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.0016
BL-RP												
10%	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.81	0.0004
9%	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.0003
8%	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.74	0.0002
7%	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.0004
6%	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.0003
5%	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.0002
4%	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.0007
3%	0.56	0.55	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.0001
2%	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.0001
1%	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.52	0.0002
BL-CC												
10%	0.87	0.84	0.87	0.86	0.86	0.88	0.85	0.85	0.86	0.87	0.86	0.0129
9%	0.84	0.83	0.84	0.84	0.84	0.83	0.84	0.84	0.84	0.84	0.84	0.0053
8%	0.89	0.91	0.91	0.87	0.91	0.87	0.92	0.88	0.91	0.89	0.90	0.0184
7%	0.89	0.89	0.82	0.88	0.83	0.88	0.89	0.89	0.88	0.88	0.88	0.0260
6%	0.85	0.80	0.80	0.84	0.80	0.80	0.81	0.81	0.81	0.80	0.80	0.0168
5%	0.78	0.77	0.78	0.77	0.76	0.78	0.77	0.76	0.78	0.77	0.77	0.0049
4%	0.73	0.83	0.72	0.82	0.83	0.82	0.71	0.85	0.79	0.69	0.80	0.0592
3%	0.80	0.81	0.80	0.81	0.80	0.78	0.79	0.79	0.82	0.80	0.80	0.0099
2%	0.66	0.65	0.65	0.65	0.65	0.65	0.66	0.65	0.67	0.65	0.65	0.0049
1%	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.0003

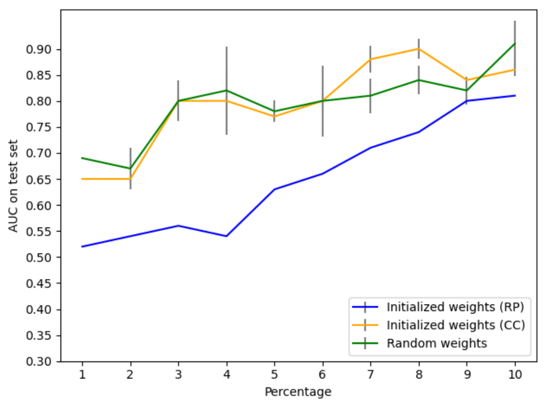
Table C.8: Performance (AUC) of the 3 models in the test set in experiment 1, with the results of each run individually shown as well as the resulting averages and standard deviations.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**	**Mean**	**SD**
BL-RW												
10%	0.84	0.81	0.82	0.83	0.81	0.82	0.81	0.82	0.81	0.81	0.81	0.0095
9%	0.82	0.83	0.83	0.80	0.81	0.83	0.83	0.81	0.81	0.78	0.82	0.0179
8%	0.83	0.84	0.84	0.80	0.84	0.80	0.84	0.83	0.83	0.82	0.83	0.0163
7%	0.79	0.80	0.78	0.78	0.80	0.80	0.80	0.82	0.79	0.80	0.80	0.0114
6%	0.80	0.78	0.80	0.78	0.79	0.82	0.79	0.79	0.80	0.75	0.79	0.0177
5%	0.79	0.79	0.77	0.76	0.78	0.79	0.78	0.77	0.78	0.61	0.78	0.0528
4%	0.43	0.43	0.42	0.42	0.40	0.42	0.43	0.42	0.42	0.41	0.42	0.0094
3%	0.78	0.28	0.77	0.77	0.81	0.80	0.79	0.28	0.80	0.59	0.77	0.2140
2%	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.42	0.50	0.0238
1%	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.32	0.50	0.0560
BL-RP												
10%	0.84	0.84	0.84	0.85	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.0011
9%	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.0010
8%	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.0010
7%	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.0016
6%	0.83	0.83	0.83	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.0014
5%	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.0008
4%	0.81	0.81	0.81	0.81	0.81	0.82	0.81	0.81	0.81	0.81	0.81	0.0008
3%	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.0023
2%	0.69	0.69	0.70	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.69	0.0031
1%	0.63	0.64	0.62	0.63	0.64	0.63	0.63	0.63	0.63	0.63	0.63	0.0051
BL-CC												
10%	0.84	0.85	0.84	0.86	0.83	0.85	0.86	0.84	0.86	0.86	0.85	0.0109
9%	0.84	0.85	0.83	0.83	0.82	0.82	0.82	0.81	0.83	0.82	0.82	0.0100
8%	0.82	0.82	0.85	0.83	0.83	0.83	0.82	0.82	0.82	0.85	0.83	0.0121
7%	0.82	0.82	0.82	0.80	0.81	0.82	0.82	0.80	0.80	0.82	0.82	0.0117
6%	0.80	0.79	0.80	0.82	0.78	0.82	0.81	0.80	0.79	0.80	0.80	0.0122
5%	0.78	0.77	0.80	0.79	0.77	0.76	0.78	0.75	0.79	0.79	0.78	0.0162
4%	0.84	0.83	0.83	0.84	0.84	0.86	0.84	0.84	0.84	0.85	0.84	0.0066
3%	0.80	0.81	0.81	0.79	0.80	0.79	0.78	0.82	0.78	0.80	0.80	0.0129
2%	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.0003
1%	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.44	0.46	0.46	0.0035
BL-RP-CC												
10%	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.84	0.0014
9%	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.0010
8%	0.85	0.85	0.85	0.85	0.85	0.85	0.86	0.85	0.85	0.85	0.85	0.0025
7%	0.85	0.85	0.86	0.85	0.86	0.85	0.86	0.85	0.86	0.85	0.85	0.0013
6%	0.85	0.84	0.85	0.85	0.85	0.84	0.85	0.85	0.85	0.85	0.85	0.0014
5%	0.83	0.82	0.82	0.82	0.82	0.82	0.82	0.82	0.83	0.82	0.82	0.0018
4%	0.80	0.80	0.80	0.79	0.80	0.79	0.80	0.80	0.80	0.79	0.80	0.0035
3%	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.72	0.71	0.71	0.71	0.0043
2%	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.0026
1%	0.60	0.60	0.60	0.60	0.60	0.59	0.60	0.59	0.60	0.60	0.60	0.0022

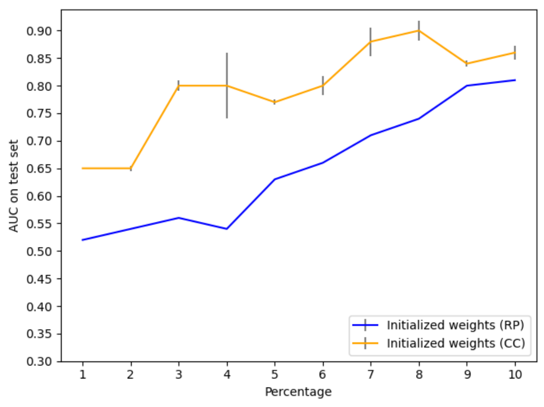
Table C.9: Performance (AUC) of the 4 models in the test set in experiment 2, with the results of each run individually shown as well as the resulting averages and standard deviations.

Appendix D

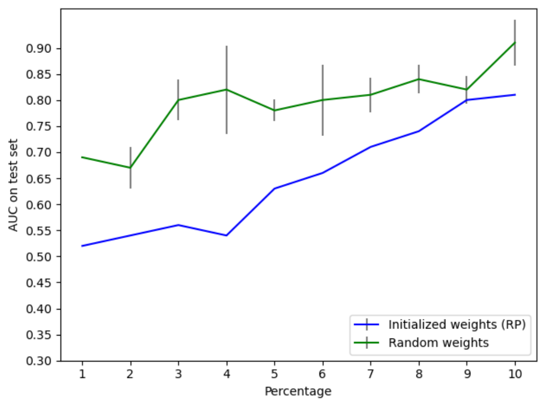
Supplementary Figures of Chapter 4 - Results



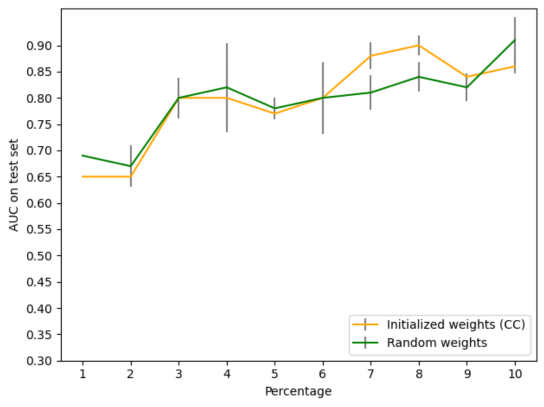
(a) With random, RP and CC initialization.



(b) With RP and CC initialization.

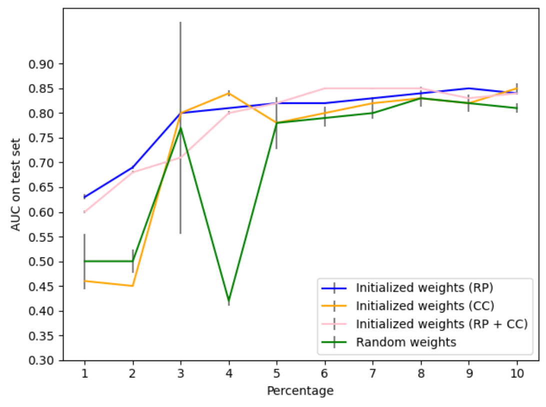


(c) With random and RP initialization.

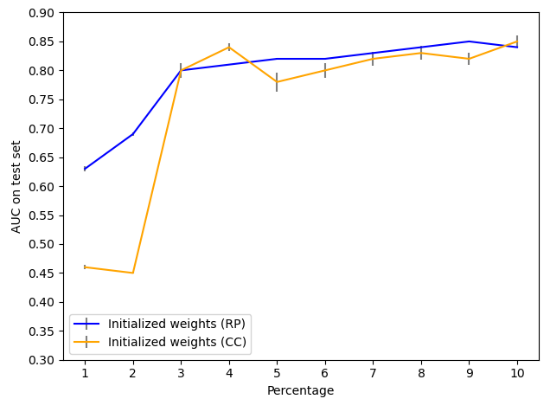


(d) With random and CC initialization.

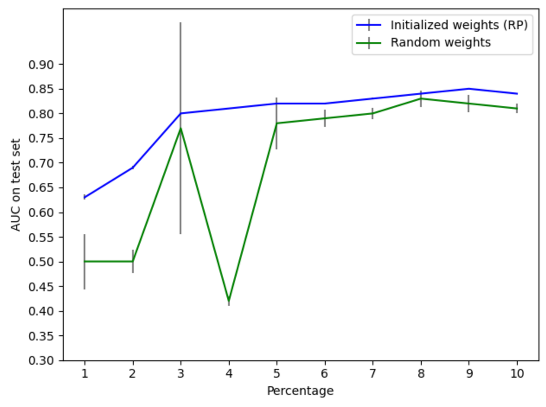
Figure D.1: Mean AUC of the baseline model on the test set, using 1 to 10% of all available data, including the standard deviation (in grey) for every percentage.



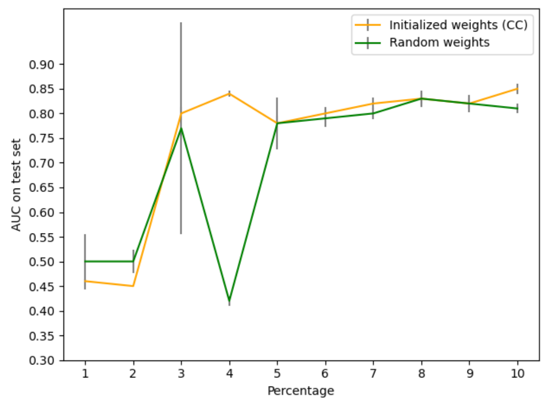
(a) With random, RP, CC and RP + CC initialization.



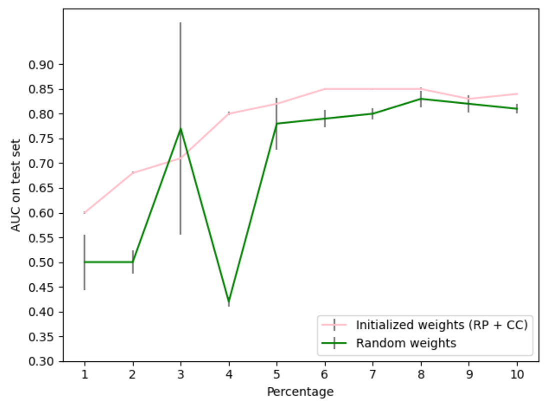
(b) With RP and CC initialization.



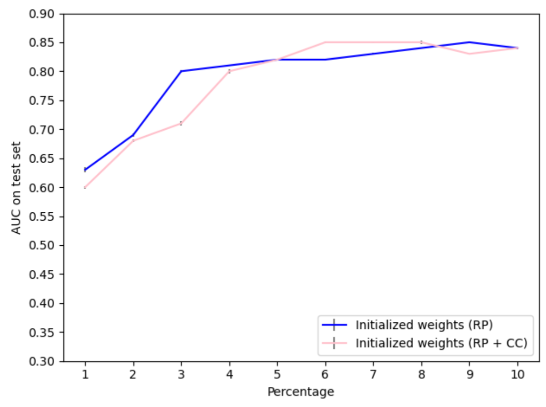
(c) With random and RP initialization.



(d) With random and CC initialization.



(e) With random and RP + CC initialization.



(f) With RP and RP + CC initialization.

Figure D.2: Mean AUC of the baseline model on the test set, from 1 to 10%, including the standard deviation (in grey) for every percentage.

References

[1] P. Kandula and C. Harden, "Epilepsy," in *Encyclopedia of Neuroscience*, pp. 1147–1149, Elsevier Ltd, jan 2009.

[2] L. Knutsen and M. Williams, "Epilepsy," in *Comprehensive Medicinal Chemistry II*, vol. 6, pp. 279–296, Elsevier, jan 2007.

[3] W. O. Tatum, G. Rubboli, P. W. Kaplan, S. M. Mirsatari, K. Radhakrishnan, D. Gloss, L. O. Caboclo, F. W. Drislane, M. Koutroumanidis, D. L. Schomer, D. Kastelijn-Nolst Trenite, M. Cook, and S. Beniczky, "Clinical utility of EEG in diagnosing and monitoring epilepsy in adults," in *Clinical Neurophysiology*, vol. 129, pp. 1056–1082, Elsevier Ireland Ltd, may 2018.

[4] U. R. Acharya, S. Vinitha Sree, G. Swapna, R. J. Martis, and J. S. Suri, "Automated EEG analysis of epilepsy: A review," *Knowledge-Based Systems*, vol. 45, pp. 147–165, jun 2013.

[5] C. D. Binnie and P. F. Prior, "Electroencephalography," nov 1994.

[6] L. F. Haas, "Hans Berger (1873-1941), Richard Caton (1842-1926), and electroencephalography.," *Journal of neurology, neurosurgery, and psychiatry*, vol. 74, no. 1, p. 9, 2003.

[7] S. Noachtar and J. Rémi, "The role of EEG in epilepsy: A critical review," *Epilepsy and Behavior*, vol. 15, pp. 22–33, may 2009.

[8] P. Gloor and R. G. Fariello, "Generalized epilepsy: some of its cellular mechanisms differ from those of focal epilepsy," *Trends in Neurosciences*, vol. 11, pp. 63–68, jan 1988.

[9] E. Bagheri, J. Dauwels, B. C. Dean, C. G. Waters, M. B. Westover, and J. J. Halford, "Interictal epileptiform discharge characteristics underlying expert interrater agreement," *Clinical Neurophysiology*, vol. 128, pp. 1994–2005, oct 2017.

[10] J. Pillai and M. R. Sperling, "Interictal EEG and the Diagnosis of Epilepsy," *Epilepsia*, vol. 47, pp. 14–22, oct 2006.

[11] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[12] A. Hyvärinen, H. Sasaki, and R. E. Turner, "Nonlinear ICA using auxiliary variables and generalized contrastive learning," in *arXiv*, pp. 859–868, PMLR, apr 2018.

[13] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: A systematic review," aug 2019.

[14] H. Banville, G. Moffat, I. Albuquerque, D. A. Engemann, A. Hyvarinen, and A. Gramfort, “Self-Supervised Representation Learning from Electroencephalography Signals,” in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, vol. 2019-October, IEEE Computer Society, oct 2019.

[15] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberyé, “Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG,” *IEEE Transactions on Biomedical Engineering*, vol. 47, pp. 1185–1194, sep 2000.

[16] C. O’Reilly, N. Gosselin, J. Carrier, and T. Nielsen, “Montreal Archive of Sleep Studies: an open-access resource for instrument benchmarking and exploratory research,” *Journal of Sleep Research*, vol. 23, pp. 628–635, dec 2014.

[17] L. McInnes, J. Healy, and J. Melville, “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction,” *arXiv e-prints*, p. arXiv:1802.03426, Feb. 2018.

[18] J. Xu, Y. Zheng, Y. Mao, R. Wang, and W. S. Zheng, “Anomaly Detection on Electroencephalography with Self-supervised Learning,” in *Proceedings - AISTATS*, pp. 363–368, 2020.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[20] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, sep 2015.

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, Institute of Electrical and Electronics Engineers Inc., aug 2016.

[22] D. Kostas, S. Aroca-Ouellette, and F. Rudzicz, “BENDR: using transformers and a contrastive self-supervised learning task to learn from massive amounts of EEG data,” *arXiv e-prints*, p. arXiv:2101.12037, Jan. 2021.

[23] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” *arXiv e-prints*, p. arXiv:2006.11477, June 2020.

[24] “ImageNet.” `http://image-net.org/`. Accessed: 2021-02-11.

[25] I. Obeid and J. Picone, “The Temple University Hospital EEG Data Corpus,” *Frontiers in Neuroscience*, vol. 10, p. 196, 2016.

[26] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley, “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.,” *Circulation*, vol. 101, jun 2000.

[27] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K. Miller, G. Mueller-Putz, G. Nolte, G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, and B. Blankertz, "Review of the BCI Competition IV," *Frontiers in Neuroscience*, vol. 6, p. 55, 2012.

[28] P. Margaux, M. Emmanuel, D. Sébastien, B. Olivier, and M. Jérémie, "Objective and subjective evaluation of online error correction during P300-based spelling," *Advances in Human-Computer Interaction*, vol. 2012, 2012.

[29] L. Citi, R. Poli, and C. Cinel, "Documenting, modelling and exploiting P300 amplitude changes due to variable target delays in Donchin{\textquotesingle}s speller," *Journal of Neural Engineering*, vol. 7, no. 5, p. 56006, 2010.

[30] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[31] "Home CNPH | Clinical Neurophysiology (CNPH)." `https://www.utwente.nl/en/tnw/cnph/`. Accessed: 2021-01-31.

[32] "Universiteit Twente (UT) | Enschede | High Tech Human Touch." `https://www.utwente.nl/en/`. Accessed: 2021-01-31.

[33] C. Lourenço, M. C. Tjepkema-Cloostermans, L. F. Teixeira, and M. J. van Putten, "Deep Learning for Interictal Epileptiform Discharge Detection from Scalp EEG Recordings," in *MEDICON 2019*, pp. 1984–1997, 2019.