UCLouvain

École polytechnique de Louvain

Transformers for EEG classification: architectures, pre-training, and applications to epileptic seizure forecasting

Author: **Tim Bary**
Supervisor: **Benoît Macq**
Readers: **Gauthier Rotsart de Hertaing, Riëm El Tahry, Dany Rimez**
Academic year 2022–2023
Master [120] in Biomedical Engineering

Abstract

Used at first in the fields of Natural Language Processing (NLP) and Computer Vision (CV), the transformer neural network recently became popular for electroencephalogram (EEG) signals analysis [1–3]. However, despite being appreciated for their ability to understand long-term dependencies and their great training speed derived from their parallelizability, transformers require a large amount of labelled data to train effectively [1, 4, 5]. Such data is often scarce in the medical field, as annotations made by experts are costly. This is why self-supervised training, using unlabelled data, has to be performed beforehand [6, 7].

On top of providing a non-exhaustive list of transformer architectures used for EEG classification, we present a way to design several datasets from unlabeled EEG data, which can then be used to pre-train transformers to learn representations of EEG signals. We tested this method on an epileptic seizure forecasting task on the TUSZ dataset [8] using a Multi-channel Vision Transformer (MViT) [3]. Our results suggest that models pre-trained with this approach not only train significantly faster, but also yield better performances than models that are not pre-trained.

The code produced during this thesis is available on this repository: `https://github.com/tbary/MasterThesis.git`.

Keywords: Transformer neural network • Electroencephalogram • Epileptic seizure forecasting • Pre-training • Self-supervised learning.

Acknowledgements

I would like to express my gratitude to Pr. Benoît Macq, thanks to whom I got the opportunity to work on this master thesis and to improve my knowledge on machine learning in general and on transformer networks in particular.

I would also like to heartily thank Dany Rimez and Gauthier Rotsart de Hertaing, who were always available to answer my questions and provide me with good feedback on my progression.

I am grateful to all the people that were keen to read my work and give me their opinion, helping me to improve the thesis with each reading. Thank you, Isabelle Ly, Isabelle Camerlynck, Vincent Bary, Catherine Echezuria, Marko Jovanovic, and Xin Zi Zhang.

Finally, I am grateful to the Pr. Riëm El Tahry, who accepted right away to become one of my readers, despite the fact that I never had the opportunity to work with her before.

Contents

Acronyms

AdaM Adaptive Moment estimation

ANN Artificial Neural Network

AUC Area Under the ROC Curve

BERT Bidirectional Encoder Representations from Transformers

CNN Convolutional Neural Network

CV Computer Vision

CWT Continuous Wavelet Transform

DALY Disability-Adjusted Life Years

DL Deep Learning

ECoG Electrocorticogram

EEG Electroencephalogram

EO/EC Eyes open/eyes closed

EOC Epoch Of Convergence

EP Evoked Potential

EPSP Excitatory Post-Synaptic Potential

ERP Event Related Potential

ETST EEG Temporal Spatial Transformer

FPR False Positive Rate

GELU Gaussian Error Linear Unit

GPT Generative Pre-training Transformer

GRU Gated Recurrent Unit

GWN Gaussian White Noise

ICA Independent Components Analysis

iid Independent and identically distributed

IPSP Inhibitory Post-Synaptic Potential

LSTM Long Short-Term Memory

MEG Magnetoencephalogram

MHA Multi-Head Attention

MI Motor Imagery

ML Machine Learning

MLP Multi Layer Perceptron

MViT Multichannel Vision Transformer

NLP Natural Language Processing

PDR Posterior Dominant Rhythm

PE Positional Encoding/Embedding

PSD Power Spectral Density

R^2 Coefficient of determination

rEEG Resting state electroencephalogram

ReLU Rectifier Linear Unit

RNN Recurrent Neural Network

ROC Receiver Operating characteristic Curve

SGD Stochastic Gradient Descent

SL Supervised Learning

SSL Self-Supervised Learning

STFT Short-Term Fourier Transform

TPR True Positive Rate

TUSZ Temple University hospital SeiZure detection corpus

ViT Vision Transformer

Introduction

The transformer neural network is a deep learning (DL) model introduced by Ashish Vaswani and his team in 2017 as an alternative to Recurrent Neural Networks (RNN). Thanks to a mechanism called self-attention, which allows the network to focus on different parts of the input sequence as it processes it, the transformer can capture long-range dependencies and contextual relationships in the input in a more efficient way than RNNs [1]. On top of this, this type of Artificial Neural Network (ANN) also operates on the entire sequence of input data at once, allowing for parallelization and improved efficiency [9, 10].

The qualities of the transformer quickly made it popular for Natural Language Processing (NLP) and Computer Vision (CV) tasks [1, 2], but its ability to correlate elements located far away in a sequence also allows it to build models suitable for signal analysis. In particular, researchers have recently started to design transformer-based architectures that would process and classify electroencephalograms (EEG) for diverse applications, including emotion recognition [11], motor imagery (MI, i.e., the mentalisation of a physical action) [12, 13], and epileptic seizure forecasting [3, 14].

However, unlike NLP and CV, which have a lot of labelled data available to train models, quality annotated data in the medical field is scarce and costly to obtain since it requires some expertise only a few people have. Self-Supervised Learning (SSL) therefore makes use of all the available unlabelled data to pre-train a model in a specific field such that the latter can be fine tuned on a small sample of labelled data while achieving good performances [6, 7, 15, 16].

The objective of this thesis is therefore double. On the one hand, this work gathers and centralizes knowledge about the transformer neural network and the different architectures proposed for EEG classification tasks. On the other hand, it tries to design a few simple pre-training tasks to utilize the huge amount of unlabelled EEG data available. The text also ensures to provide the proper context to any engineer with no background in the biomedical field as well as any medical practitioner with little knowledge about DL, although a few notions of Machine Learning (ML) are required. To guide the reader through this work, a paragraph at the beginning of each chapter summarizes what is developed in the following pages as well as how it is related to the rest of the content.

This thesis is divided into two parts of three chapters each. The first part defines the context of this field of research by introducing EEGs and transformers working principles.

It also presents the different ML algorithms used for EEG classification of neurological diseases, as well as the different types of transformer architectures used for other tasks of EEG classification. A quick overview of the epilepsy and of how epileptic seizures appear on a patient's EEG is also provided, as this pathology is employed as a practical case in the second half of the thesis.

This second part describes several pre-training tasks designed to help a model improve its representation of EEG signals. To benchmark the pre-trainings, a transformer architecture from the literature is built, with some adjustments made to its signal processing pipeline, and is first trained using a simple eyes open/eyes closed classification. The efficiency of the best pre-training is then tested on an epileptic seizure forecasting task using the Temple University hospital Seizure detection corpus (TUSZ) dataset [8]. The last chapter of this part focuses on the limitations of the work produced and on the future research perspectives by presenting potential improvements and how these should be implemented.

Part I

Context and state of the art

Chapter 1

Basics of electroencephalography

This first chapter focuses on the electroencephalogram (EEG) operational principle, how it is used in modern medicine, and its main qualities and limitations. It also provides all the biological background necessary for its proper understanding, such as the way neurons work and propagate information, as well as the different signals recordable by an EEG.

The content of this chapter is aimed at readers with little to no knowledge about EEGs. The objective is to help understand where this multichannel signal comes from and what are its properties so that the strategies for model and pre-training designs approached in sections 3.3, 4.2, and 4.4 feel more logical and intuitive.

1.1 Physiology of the neuron

Neurons are specialized cells considered as the basic units of the nervous system. They are composed of three main parts (fig. 1.1). The soma, or cell body, comprehends the nucleus as well as all the organelles necessary for the cell survival and function. Dendrites are fibrous structures branching from the latter that aim at reaching the terminal ends of the neighboring cells' axons. Axons are long, cylindrical tails that connect to the cell body at the axon hillock. They are intermittently covered in myelin sheaths produced by glial cells. The number of dendrites and axons, as well as their position with regards to the soma determine the type of the neuron [17, 18].

The main ability that differentiates neurons from other specialized cells is their capacity to receive, process, and send signals. The way a signal propagates inside a neuron is via the particular electrochemical properties of this cell. At resting state, there is a negative potential difference of $60 - 70mV$ [19, 20] between the intracellular and extracellular media, which is due to a concentration difference in K^+, Na^+ and Cl^- ions between the two sides of the membrane. This concentration gradient is actively maintained by ions transporters which move ions against their natural diffusion gradient.

The information moving along the neuron is translated to a local inversion of the membrane polarity triggered by the opening of Na^+ gated channels that leads to the quick diffusion of these ions from the extracellular into the intracellular medium. Because these

channels are voltage-gated, which means they open or close depending on the voltage, this depolarization induces the opening of nearby channels, propagating the influx.

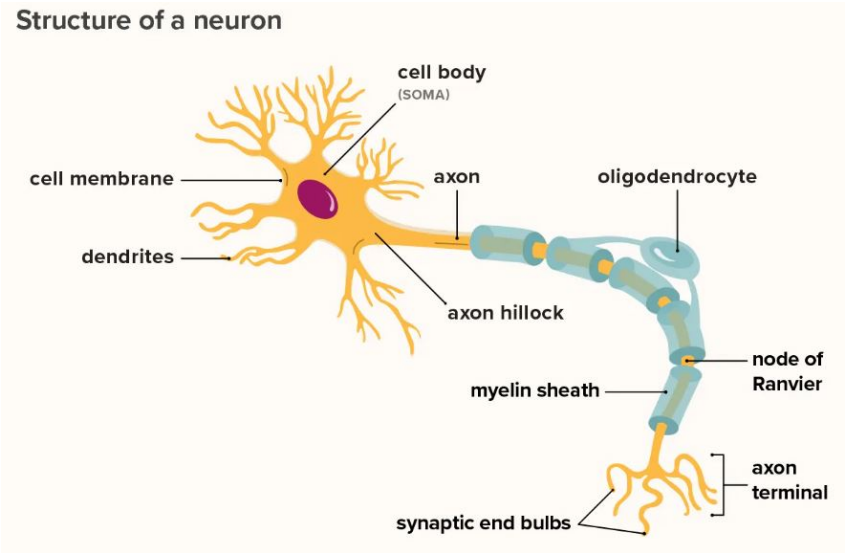

Figure 1.1: Representation of the neuron key elements, adapted from [18]

To reinstate its initial potential, the membrane will, as a second step, open its K^+ gated channels, allowing these positive ions to escape the cell. The cell potential will even overshoot its basal state which, along with the Na^+ gated channels being momentaneously locked, prevents the signal from propagating in the wrong direction – the good direction usually being from the dendrites to the axon. This overshoot also limits the maximum number of influx that can go through the neuron in a given time, as the potential needs to be brought back to its basal level first. This is known as the refractory period [19, 20].

The propagation of the influx inside a neuron is represented on figure 1.2. For the neurons with myelinated axons, the influx does not propagate gradually but instead hops from one unmyelinated section – known as a Ranvier node, visible on figure 1.1 – to another. This allows a faster propagation of the signal [21].

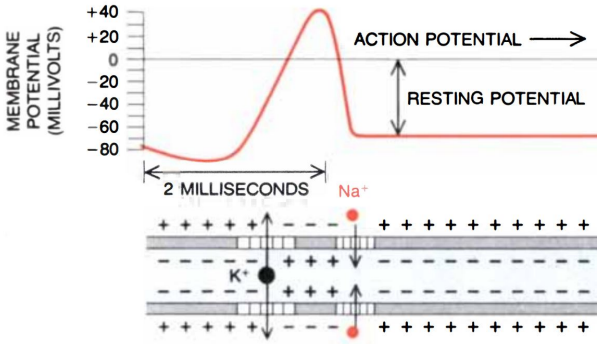

Figure 1.2: Propagation of the action potential in an unmyelinated axon, adapted from [20].

Neurons connect to one another via the synapse, a small gap between the axon of a pre-synaptic cell and the dendrites (axo-dendritic), body (axo-somatic), or axon trunk (axo-axonic synapses) of the post-synaptic cell [22]. To transmit a signal through the synapse, the terminal end of the pre-synaptic neuron releases neurotransmitters, which are signalling molecules that bind to gated ions channels present on the post-synaptic membrane, activating them.

Depending on the inhibitory or excitatory nature of the synapse, which determines the nature of the ions diffusing through the membrane, this will trigger respectively a local hyperpolarisation or depolarisation in the neuron [20, 23]. If the depolarisation reaches a certain threshold, typically between -55 and $-50mV$, it generates an action potential that will propagate inside the neuron [19, 20]. Nothing will happen if the threshold is not reached, which is know as the all-or-none law [24].

The process by which a neuron integrates the information received and outputs an action potential or not is called summation. As explained in the previous paragraph, the depolarisation in the post-synaptic neuron has to reach a certain threshold in order for the neuron to fire and the excitation by one pre-synaptic neuron will not necessarily trigger this response. One of the two existing summation types is the spatial summation, where the neuron sums up the excitatory post-synaptic potentials (EPSP) and the inhibitory post-synaptic (IPSP) potentials of different neurons. If the sum of these potentials reaches the threshold, the neuron will fire.

The other summation type, called temporal summation, benefits from the amount of time necessary for the EPSPs to fade away. As a matter of fact, if a post-synaptic neuron repeatedly receives excitatory signal from the same pre-synaptic axon in a short time span, the EPSPs generated will accumulate until the threshold is reached [25, 26]. The difference between both summations is shown on figure 1.3.

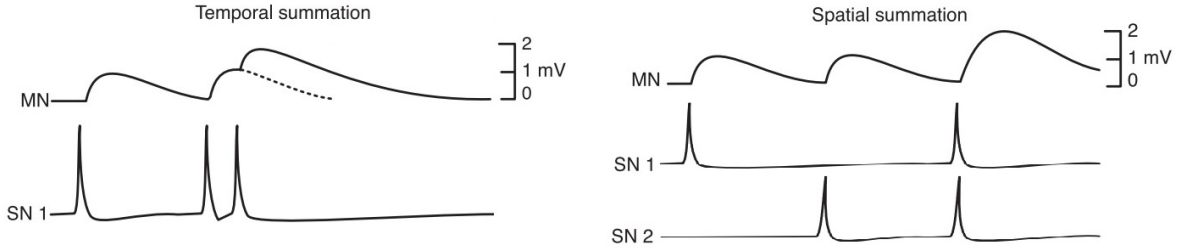

Figure 1.3: Difference between temporal summation (left) and spatial summation (right), SN1 and SN2 are the sensory, pres-synaptic neurons and MN is the motor, post-synaptic neuron. Adapted from [25].

The last key principle of the neuron physiology relevant to this thesis is the way the neuron translates the intensity of the signal received. As explained previously, the all-or-none law prevents the neuron from firing if the EPSP amplitude is too low, but this law also encompasses that, whether the threshold is mildly or strongly surpassed, the amplitude of the action potential will remain the same. The signal strength is instead

modeled by the firing frequency of the neuron. This frequency coding maps a high level of excitation to a high firing frequency and a lower level to a lower frequency [24, 27, 28].

1.2 Measure of the cortical activity by the EEG

EEG is a cortical activity measurement tool and technique that was used on humans for the first time in July 1924 by the psychiatrist Hans Berger − following the work of other scientists such as Adolf Beck and Richard Caton [29, 30]. It records the electric fields induced by neurons activity, translated into a potential difference between electrodes placed on the patient's scalp. This electrophysiological technique allows a non-invasive way to understand dynamic cerebral functioning [31].

Due to the thin membrane of the neuron, the potential difference between extracellular and intracellular media generates an electric field of around $100kV/cm$ at rest, which fluctuates when the cell conveys signals [20]. It is the sum of the electric fields generated by spatially close neurons firing simultaneously that is recorded by an EEG electrode [32].

The action potential travelling through the neuron are too fleeting and therefore cannot sum up to be detected at the scalp surface, but this is not the case of the post-synaptic potentials that can last tens of milliseconds [33, 34]. In particular, the pyramidal neurons, located in the layers III and V of the cerebral cortex [35], tend to contribute the most to the signal captured at the surface of the skull due to their structure and orientation − with a long dendrite aligned perpendicularly to the scalp − giving them the behavior of an electric dipole [33].

The EEG is composed of multiples electrodes distributed uniformly on the patient's scalp. In the most commonly used *International 10-20 system* (figure 1.4), 19 electrodes − plus eventually one or two as reference − are placed on the skull such that the distance between two adjacent probes represent 20% of the skull distance from left to right or from nasion (front) to inion (back). Each electrode is given a location and a tag consisting of a letter and a number. The letter depends on the closest brain area (table 1.1) and the number increases as the electrode location moves away from the sagittal plane − with the letter z meaning zero. Even numbers are located on the right hemisphere and odd numbers on the left [36, 37].

Table 1.1: Lobe nomenclature signification in the *International 10-20 system* [36].

Fp	Frontopolar	F	Frontal	P	Parietal
C	Central	T	Temporal	O	Occipital

Extensions of the 10-20 system exist to achieve higher resolution, such as the 10-10 system, with 74 electrodes, and the 10-5 system having 142. Other placement methods use 64 − typically used for event related potential (ERP) recording, explained later in this chapter −, 128 or even 256 electrodes placed between the ones of the initial 10-20 system [38]. Systems with a high number of electrodes such as the 10-5 system are in

practice used in research settings rather than clinical settings, as they only add minimal information for a much higher computational cost [37].

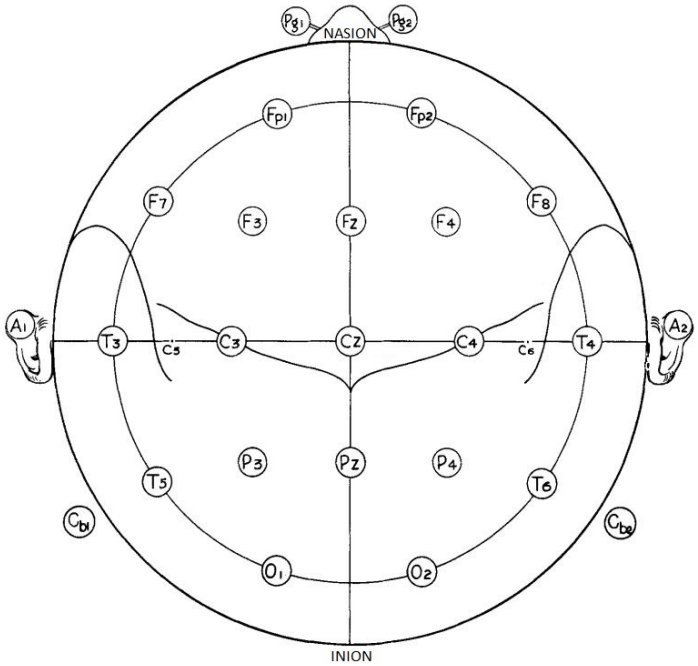

Figure 1.4: Placement of the electrodes and nomenclature of the 10-20 system. Electrodes $A_{1,2}$, $C_{b1,2}$ and $P_{g1,2}$ are reference electrodes and only a subset of them is added to the montage [36].

To record a *difference* of potential, electrodes need a reference which will depend on the montage used. For so-called referential montages, one or multiple electrodes determined *a priori* (e.g. $A_{1,2}$, $C_{b1,2}$ and $P_{g1,2}$ on figure 1.4) play this role. The sub-case of common average montage use the average of all channels output as the reference value. These types of montages, called unipolar, are very efficient at capturing phenomena spreading through the whole cortex, but the operator must ensure that the reference suffers from a minimal amount of artifacts or it will contaminate the whole measure.

In bipolar montages, each probe compares its obtained value with the adjacent ones, either longitudinally or transversely. Laplacian montages use a weighted sum of the surrounding channels as a reference. Bipolar and Laplacian montages are used to study local phenomena [31, 37, 39].

1.3 Two main types of EEG experiments

The patient can be subject to two main types of EEG examinations. The first one, called resting state EEG (rEEG), collects signals emitted by a brain not carrying out any task in particular. To do so, the patient's neural activity is recorded during a few minutes, either with the eyes closed or open in an environment that will not trigger any specific brain reaction (i.e. dim light, low sound, comfortable position) [40]. For the detection of

certain pathologies such as epilepsy, provocation techniques may be added as a way to induce sought abnormalities. These maneuvers include hyperventilation and stimulation of optic nerves via strobing lights [31].

During rEEG, the recorded periodic signal is called Posterior Dominant Rhythm (PDR). It is a mix of several sine waves with a frequency ranging from 1 to 70Hz [31, 41]. This range is divided into 5 frequency bands, which are reflecting the age and the wakefulness of the subject and whose unusual amplitude can reflect certain neurological pathologies. The main bands names and ranges are delta $(1-4\text{Hz})$, theta $(4-8\text{Hz})$, alpha $(8-13\text{Hz})$, beta $(13-30\text{Hz})$ and gamma $(>30\text{Hz})$. An example of EEG decomposition into these frequency intervals is shown on figure 1.5.

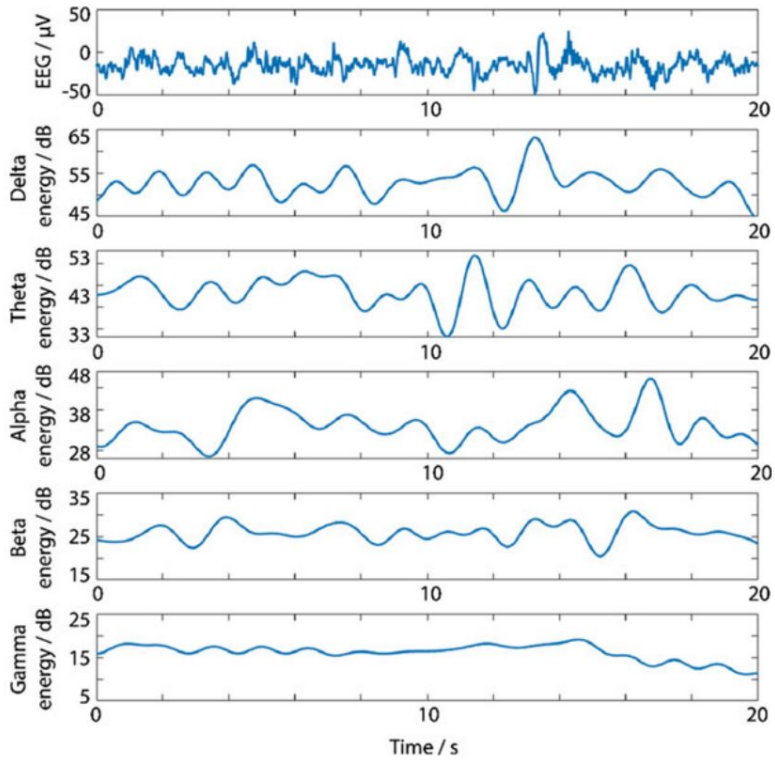

Figure 1.5: Representation of the different frequency bands isolated from a raw rEEG [42].

There is no consensus in the literature regarding the bounds of each bands, but the variability between the sources stays rather low [40, 42, 43]. Lower frequency signals are related to a relaxed and passive brain whereas higher frequencies, such as in beta and gamma waves are the sign of a more intense activity, like attention, concentration, or memorization [43, 44]. Other types of waves can be observed in rEEG, such as the Mu rhythm, overlapping the higher alpha rhythms and related to mirror neurons ; sleep spindles and K complex during sleep ; or ripples (frequency $>100\text{Hz}$) which are a marker for epileptic activity [31, 44].

The other protocol using EEG consists in submitting a stimulus to the subject and capturing the reaction happening inside its brain. That reaction is either called an Evoked

Potential (EP) when the stimulus is basic − such as a light flashing, a sound or an electric shock − or an Event Related Potential (ERP) when the stimulus requires a complex processing by the brain − like memorization, attention and expectation tasks [44, 45].

The acquired waves have a small amplitude with regard to the ambient background noise and it is therefore required to repeat the stimulus multiple times in order to average the trials and suppress that noise. This method, called time-domain averaging, is used under the assumption that the response stays stationary between trials. When it is not the case, other techniques need to be used to retrieve the signal [46, 47].

Nomenclature for EP and ERP is as follow: a letter indicates the polarity of the peak (P for positive, N for negative) and a value is added afterward, being either the delay in milliseconds between the event onset and the peak, or the number of the latter (since EP and ERP are composed of a series of positive and negative peaks). For instance, P50 is a positive peak happening 50ms after the stimulus. If it is the first positive peak of the reaction, it can also be called P1 [45, 48].

1.4 Strengths and limitations of EEG

The main reasons why EEGs are still used in medical settings nowadays are their low cost − with non-medical EEG hardware costing less than 1000$ being effectively used in scientific experiments [49] −, their portability − allowing to conduct ambulatory EEGs, where the patient is recorded doing normal activities during several days without being impaired by the device [50] − and their non-invasiveness [37, 42, 43]. The previous criteria, added to the possibility to correct artefacts related to head movements [51] and the non-necessity for the patient to be actively participating in the protocols leads to the usage of EEG in a lot of medical cares involving newborns [52] and patients in a coma [53] or unable to communicate [54].

Furthermore, EEG temporal resolution, ranging from one millisecond to one second, is one of the best for non-invasive procedures, along with magnetoencephalography (MEG). Other cortical activity measurement techniques, qualified as "metabolic" in opposition to the electrophysiological-based EEG and MEG, have a lower temporal resolution. This is due to the indirect observation methods they use, that relies on the metabolic consequences of a neuron activity, which takes a few seconds to occur [55–57].

Three main types of limitations arise when using EEG for research and diagnosis. One of them occurs when the goal is to localize the sources of the signals captured at the scalp level. With around 256 electrodes for the biggest montages [38] against billions of neurons potentially generating fields [18], the number of recording probes is clearly limited with regards to the number of sources. The resulting problem, known as inverse problem, is ill-posed and unstable as there is no unique solution and these are very sensitive to the noise. It therefore needs to be constrained, for instance by limiting the sources to a finite number of dipoles [42, 58].

However, this harms the spatial resolution, which becomes much lower than for the metabolic-based techniques [55, 56]. A comparison of temporal and spatial resolution of electrophysiological- and metabolic-based techniques is shown on figure 1.6.

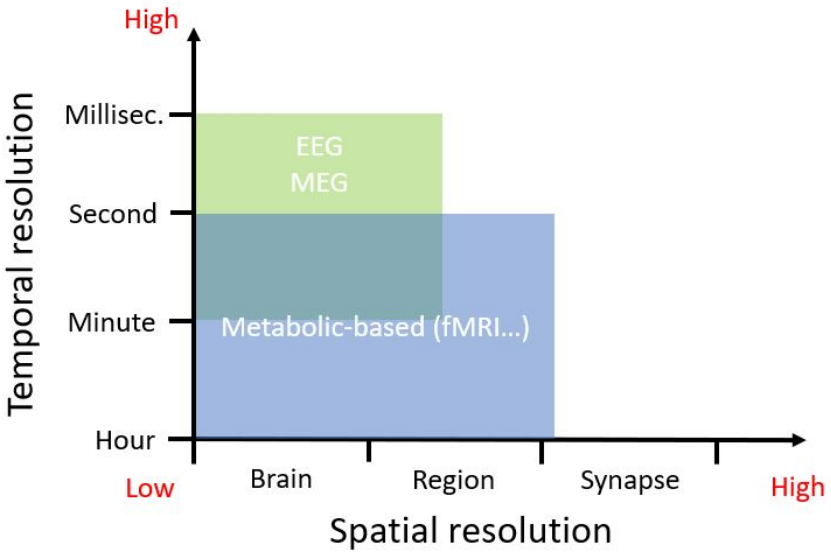

Figure 1.6: Areas of resolution for non invasive techniques. [55, 56]

Secondly, many elements in the brain do not have a significant contribution to the EEG signal. As mentioned earlier, it is believed that most of said signal is provided by the activity of pyramidal cells, whose dendrites being perpendicular to the scalp help form a dipole with an optimal orientation for the electric field to be captured [32, 33].

Another reason why these neurons are more visible to the EEG is because they are arranged in an open field configuration. This means that the equivalent dipoles representing theses neurons do not cancel because they are aligned with one another (figure 1.7-A). The field resulting from a closed field configuration (i.e. when the neurons are arranged in a globular manner with equivalent dipoles randomly oriented) is almost null and can therefore not be captured at the scalp level [59, 60]. Illustration of open and closed fields is shown on figure 1.7-B.

Another important factor for a field to be visible is temporal synchrony. If the activity of the neurons is asynchronous, the resulting summed EEG will be negligible. It has been shown that cortical areas of at least 6 cm^2 must achieve synchronicity to be observed [31, 32]. Lastly, the brain tissues tend to not conduct the field generated by the neurons very well and therefore the dipoles located near the surface of the brain tend to contribute more to the signal than the ones located deeper [32, 61].

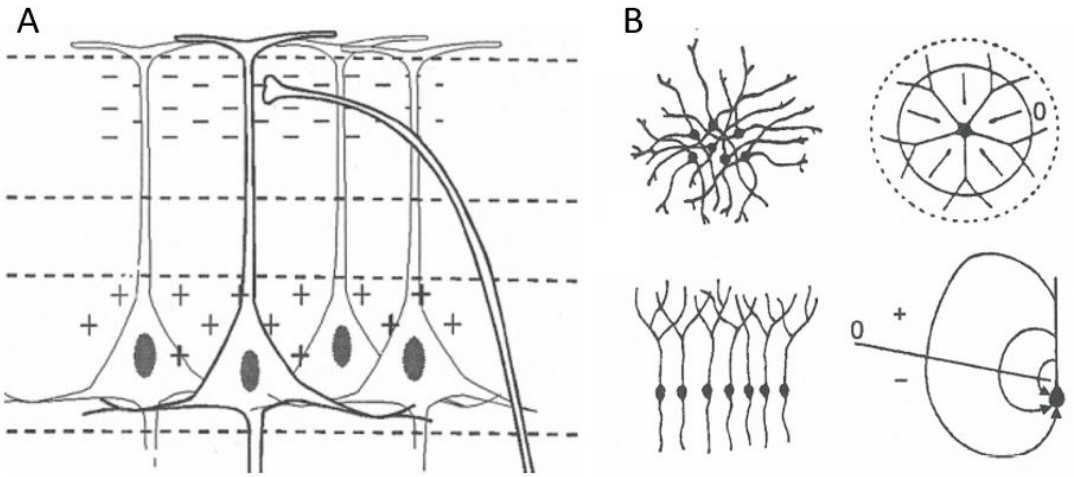

Figure 1.7: (A) Pyramidal neurons aligned in an open field configuration. (B) Neurons in closed field (top) and open field (bottom) configurations. Adapted from [59].

Finally, the potential recorded by the electrodes being in the order of $1 - 100\mu$V [62] makes the signal very sensitive to noise and artifacts. These are either physiological, such as blinking, eye movements, myographic activity in the heart, maxillary or scalp muscles, and sweat (which modifies the electrodes impedance and induce drifting) ; or non-physiological, like powerline contamination, loose wiring, electronic malfunction, low battery and wrong electrode placement [31, 43, 63].

These artifacts are very detrimental because they tend to drown the signal under background noise (especially for ERPs). They can also resemble pathological biomarkers such as epileptic seizures, requiring a specific training for EEG interpreters to differentiate them [31, 64].

Chapter 2

Exploration of the transformer network

This chapter contextualizes the apparition of the transformer as a machine learning model and describes its structure in depth, explaining how its parameters are optimized. It also explores how this tool, initially designed for Natural Language Processing (NLP), can be trained and pre-trained to deal with the lack of labelled data present in the field of electroencephalography.

The goal of this chapter is to teach the processes by which transformers are able to extract useful information from their inputs as this will be considered as familiar to the reader in the rest of the thesis. In particular, sections 2.1, 2.3, and 2.4 are aimed at readers with little background in deep learning, so that the concepts addressed during model design in part II (sections 4.2 and 4.3) are understood.

2.1 Definition of deep learning and history of transformers

Deep learning (DL) is a hierarchical machine learning field that models complex relationships between data by processing their features through different layers, using learnt low-level concepts to define higher level ones. The models issued by this paradigm allow great progresses in fields such as speech technologies, Computer Vision (CV), NLP and some medical applications [65–67].

The use of artificial neural networks (ANN) is almost inevitable in DL. ANNs, called this way to dissociate them from biological neural networks found in animals nervous systems, are composed of nodes − or neurons − arranged into jointly connected layers. These neurons receive inputs from the preceding layers and multiply each of them by a weight before applying a non-linear operation − also named activation function − to produce an output sent to the next layer. The weights' values are learnt by the network during a training phase and represent the strength of a connection between two neurons. On a trained neural network, the data enters via the input layer, is treated by the hidden layers and a result is provided on the output layer [68–70].

An example of a simple ANN is the Multi-Layer Perceptron (MLP, figure 2.1). It is a kind of neural network classified as fully connected and feed-forward, which means that each neuron of a layer is connected to, and only to, all neurons of the directly preceding and succeeding layers, without any retroactive connection. These models contain at least one hidden layer[1] and are able to solve problems that are not linearly separable [70, 71].

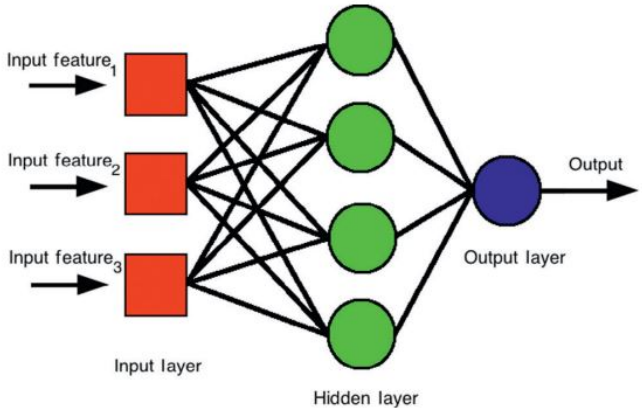

Figure 2.1: Representation of a MLP with one hidden layer [72].

The activation functions typically used by the neurons are either the sigmoid/softmax function (figure 2.2-A) or the rectifier linear unit (ReLU, figure 2.2-B) that outputs the input if it is positive and 0 if it is negative [73, 74].

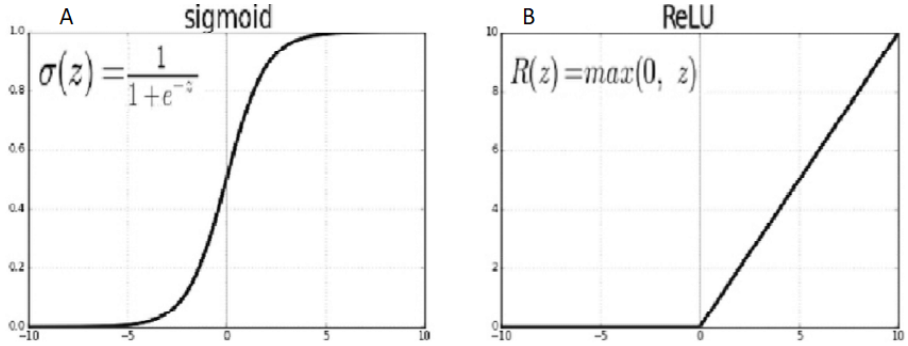

Figure 2.2: Sigmoid (A) and ReLU (B) activation functions. Adapted from [75].

Another type of ANN that was widely used to tackle NLP tasks before the apparition of the transformer is the recurrent neural network (RNN). RNNs are an extension of feed-forward neural networks that include loops in the architecture (recurrent connections). These retroactive connections allow the model to handle sequences of variable length instead of fixed-size vectors, which is not possible with MLP [76].

[1]The term MLP will be used in this thesis as an abuse of language for fully connected feed-forward networks, even when no hidden layer is present.

However, this simple RNN suffers from the vanishing gradient effect, which is the phenomenon by which the distant past elements in the input sequence do not influence the current element during the training phase. It is due to a distance-dependent contribution of the past inputs to the error function [76, 77].

To overcome this issue, the long short-term memory (LSTM) [77] and the gated recurrent unit (GRU) [78] networks were introduced. Both networks are similar in their idea and performances but the more recent GRU has less parameters to train. The idea of those architectures is to add a trainable memory state to the cells that allow a unit gain transfer between the neurons when activated, hence bypassing the vanishing gradient problem [76, 77].

The Transformer, introduced in 2017 by Vaswani *et al.* [1], replaces the neurons of the recurrent neural network by an attention mechanism. Attention allows the model to give a different weight to each input token depending on their relevance in the treatment on the current token using matrices of learnt weights. Self-attention refers to when the input is compared to itself, whereas two inputs can be compared to one another in regular − cross − attention. In the field of NLP, self-attention is similar to linking a word to the ones which are relevant to it in the sentence (figure 2.3) [1, 9, 10].

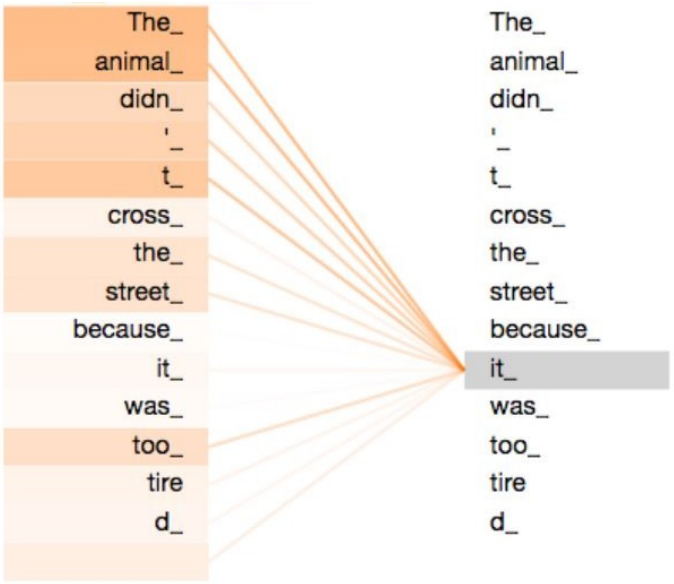

Figure 2.3: Illustration of the self-attention mechanism. The word “it” is linked to the words it may refer to with a higher weight depending on the relevance. Adapted from [9].

For its original purpose, the transformer was designed to set a new state of the art in language translation. This new model achieved higher performances than previous RNN and LSTM, while being significantly faster to train thanks to its parallelizability [1]. Furthermore, the trained transformer model is much more interpretable than a neural network since it shows how the weights are attributed to all the input tokens [10].

However, all these advantages come at the cost of a computational and memory complexity quadratic to the input length, which results in models that are hardly scalable and very demanding memory-wise [79, 80]. On top of this, the transformer-based models will overfit if not trained on enough data with proper regularization [4, 5, 81].

2.2 Structure of a transformer

The original architecture of the transformer proposed by Vaswani *et al.* [1] is given in figure 2.4. This architecture, composed of N encoder layers and N decoder layers, is used to process sequence to sequence tasks. In the field of deep learning, a sequence is an array of variable, undetermined length [68]. Classification tasks using a signal as an input are typically sequence to vector tasks (where a vector has a known length ; one in this case).

For such tasks, the encoder is used to project the input data into a new representation, called an embedding where the short and long-term dependencies are more visible. A second stage then interprets that embedding and classifies the data [3, 82–84]. Since the decoder is not used in these models, it will therefore not be further detailed in this thesis, especially since its constituents are similar to the ones of the encoder.

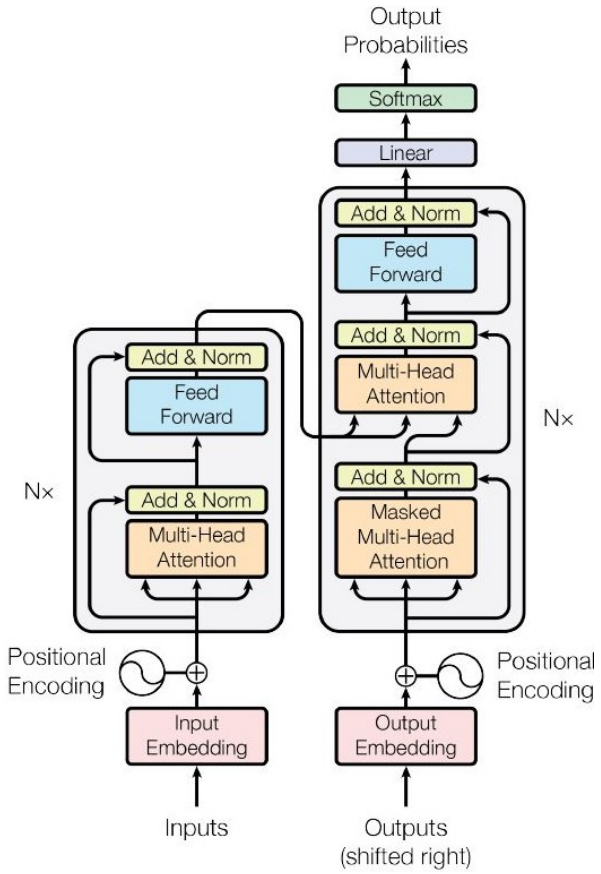

Figure 2.4: Architecture of the Vaswani *et al.* transformer [1]. The encoder (left) processes the inputs that are then used as a comparative attention element to generate the output sequence in the decoder (right).

Input embedding Before the data enters the transformer, it needs to be tokenized. A token is the smallest data sub-unit that the model can train on [68]. The tokenization in NLP is typically done using a tokenizer, which splits sentences into words or syllables and attributes an ID to each of these elements as well as to the punctuation and the [start] and [stop] tags that delimit the sentence [85, 86].

When the input data are images, the tokenization usually consists in patching said images [2, 87]. The same idea of subdividing the input into smaller − sometimes overlapping − parts is generally applied for all other kinds of input, including raw signals [82, 84].

Positional encoding Unlike RNNs whose structure already takes into account the arrangement of the input sequence, the transformer neural network is invariant to the input order. The information of the position of each token in the sequence, which is important in fields like NLP, CV, and signal processing, has therefore to be embedded directly in the input [88, 89].

To do so, positional encoding (PE), also named positional embedding, is applied. It consists in the element-wise addition of a stack of sines and cosines of different frequencies to the tokenized data, with the frequencies being dependent on the token position. This embedding method is used because waves of similar frequencies align well, allowing the encoded tokens that are close to one another to have a higher score when the sequence is compared to itself [1, 90].

Figure 2.5 shows the alignment of all the tokens of a sequence with the 1000^{th} token of the same sequence after using PE. Trainable PE may also be chosen and consists in the addition of trainable weights to the tokenized data. It yields similar performances as sine-wave PE, but requires more parameters to train [1].

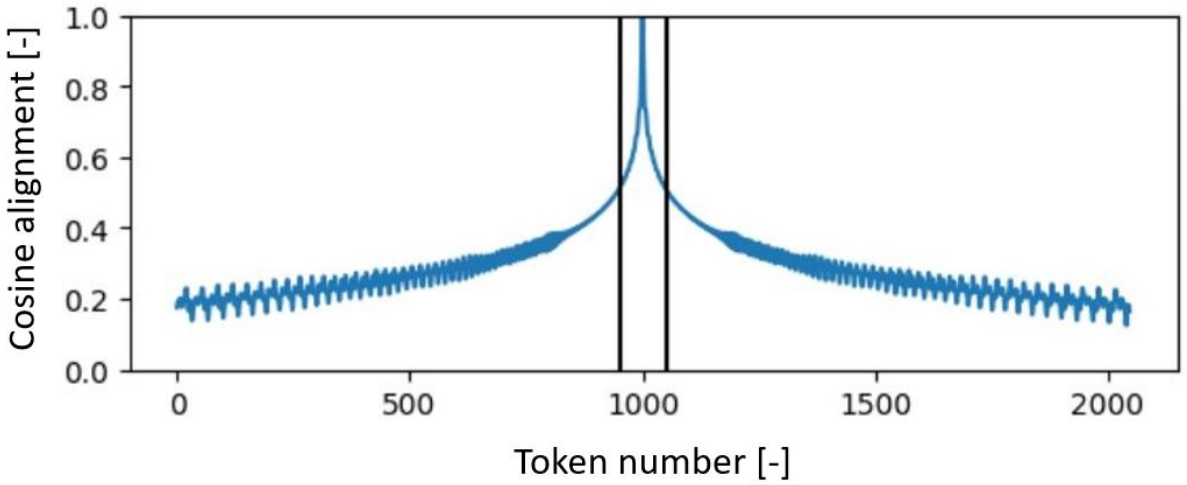

Figure 2.5: Cosine alignment of the thousandth token with the rest of the sequence after using positional embedding. Unit-value tokens, sequence length: 2048, number of sine waves for PE: 512. Adapted from [90].

Multi-head attention The attention mechanism consists in a succession of operations on three vectors called the keys K, queries Q and values V. Those vectors are generated by multiplying one (in the case of self-attention) or more embedded inputs with weight matrices W_K, W_Q and W_V. The attention output is then computed using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Where d_k is the dimension of the keys vector and is used for scaling, which stabilizes the gradients during training. The softmax function is the same as the sigmoid function (figure 2.2-A), but extended to multiple dimensions [1, 9, 90].

Transformers use multi-head attention (MHA), which means that multiple attention heads process the same input using different weight matrices, projecting the input into different representation sub-spaces. The outputs of all the attention heads are concatenated in a single vector. The computational cost of the operation remains the same because the Q-K-V vectors dimensions are divided by the number of heads. With this reduced dimension and since the heads are independent − which implies they can be processed in parallel − the input treatment is made much faster [1, 9].

Multi-head attention can be seen as analyzing the sentence through multiple prisms to retrieve different contexts [10, 90]. The most telling example is with NLP where, given a certain sentence, a first head could, for instance, analyze the semantics whereas another could link the words using a syntactic point of view.

Add and normalize The add and norm block is put after each sub-layer of the transformer. It sums the input (through what is called a residual connection) and the output of each sub-layer to avoid the gradient vanishing problem discussed earlier [9, 90]. The sum is then normalized using batch normalization, which improves the stability on initialization and the learning time by allowing the use of larger learning rates [91].

Feed forward network The feed forward network (see the paragraph on MLP, section 2.1), placed at the output of each encoder or decoder block is composed of one fully connected layer. The same network (with the same neuron weights) is applied to each attention vector separately, making it highly parallelizable. Its role is to embed the inputs to make them suitable to enter the next block [1, 9].

2.3 Deep learning models training strategies

To understand the training strategies that are developed for transformers in this thesis, it is first important to understand how Supervised Learning (SL) works. SL consists in providing data with the correct label attached to it to an untrained model, and letting it adjust its trainable parameters − its weights − in order to provide a similar answer as the label. The label is either a number in the case of regression problems, or a class in the case of classification problems [92, 93].

The main challenge with SL is the disponibility of quality annotated data as data annotation (i.e. the action of providing manually a data to a label, which has to be done by a human in most cases) is very costly, especially in medical settings where this has to be fulfilled by an expert. Attempts at reducing the annotation cost are detrimental to the data quality (which is very important to allow models to learn properly) [93, 94]. SL is the strategy used by Vaswani *et al.* to train the initial transformer model [1].

Unlike labelled data, which is sparse and costly to annotate, unlabelled data presents the advantages to exist in a much higher volume, to be much cheaper to produce, and to be less task specific (i.e. a labelled data is only useful in a specific context but the annotation could be useless if the task is different) [7, 95]. A possibility to exploit this type of data is to perform Self-Supervised Learning (SSL). Also called unsupervised learning by abuse of language[2], SSL first automatically creates labels for the data based on some of its intrinsic characteristics (for instance, the label could be the value of a specific token masked in the sequence, which can be generated using a simple code) and then trains a model using this "dummy-labeled" data, as in a SL task [6].

This technique is used for model pre-training, as it has been found that the weights resulting from aforementioned simple learning tasks constitute an excellent starting point for training in the same domain as the initial task [97]. This paradigm extends what is known as "transfer learning", which infers that a model performing well on a precise task would train faster and yield better performances earlier than a model initialized with random weights on a similar assignment [15, 16]. Once the model has been pre-trained on a large sample of dummy-labeled data, only a few labelled data are required to fine-tune the model to the desired task.

A metaphor of the above methodology is to give easy exercises to students so that they can understand a general subject before submitting them fewer, but more precise questions on a specific area in the same field. The students will use the knowledge they already gathered with the simpler questions to improve faster at answering the more specific ones.

An example of SSL for transformers is found in the field of NLP. BERT (standing for Bidirectional Encoder Representations from Transformers) is a pre-trained transformer encoder that can later be fine tuned to accomplish other tasks without having to choose the initial weights at random. The two dummy-tasks used for its training are the recovery of a masked word in a sentence and the classification of duets of sentences according to whether they were following one another. This pre-training allows BERT to already learn certain characteristics of the human language and the need for labelled data for a specific task becomes much smaller when starting from such a pre-trained model.

Nevertheless, this pre-training is very time expensive and still requires a high volume of unlabelled data [97, 98]. BERT methodology is transposable to other fields such as CV

[2]Unsupervised learning is usually used to discover patterns in the data but without performing classification nor regression [96].

[99–101] or signal processing [87, 102] that are both in the scope of this thesis. In the same vein, GPT (Generative Pre-training Transformer) is a pre-trained transformer decoder used notably in the recent and public-famous ChatGPT. However GPT-like architectures and applications are no interest to this thesis as they are decoder-oriented and working on already embedded data [103].

Another SSL strategy compatible with transformers is cross teaching, that is, training a less data-greedy architecture on a few labelled data and make it create pseudo-labels for the unlabelled data that are then used to train the transformer. This strategy allows and makes up for the lack of initial labelled data, but the annotation quality provided by the small model will not be as good as what a human can produce [104, 105]. This is why this method will not be further explored in this thesis.

2.4 Determining the weights of a transformer

A transformer's weights are found the same way as for any other type of ANN. To determine the correctness of the answer provided by the model, its outputs are compared with the data labels using a loss function that maps the closeness between the two values to a value between zero and infinity, with zero being a perfect match [106]. The loss function mainly used for classification is the categorical cross-entropy − even though many others are used in the literature, yielding similar or sometimes better results depending on the model architecture and the application [107, 108]. It is computed as followed for each data j in the train set:

$$\text{Loss}(t,s)_j = -\sum_{i}^{C} t_{ij} \log(s_{ij}) \tag{2.2}$$

Where C is the number of classes, t_{ij} is the true probability for data j to be in category i (equal to 1 or 0 whether the label is respectively equal to i or not) and s_{ij} is the softmax output of the predicted probability for data j to be classified in category i (bounded by 0 and 1) [109]. Cross entropy can be interpreted as how uncertain the model is about the real label of the data j [110]. For binary classifications ($C = 2$), the equation 2.2 becomes:

$$\text{Loss}(t,s)_j = \begin{cases} -\log(s_{0j}), & \text{if } i = 0 \\ -\log(s_{1j}), & \text{if } i = 1 \end{cases} \tag{2.3}$$

Which is called binary cross entropy. This loss function is used because the output value tends to explode when the predicted value is far from reality, which is useful later in the process. The total loss for both equations 2.2 and 2.3 is equal to either the sum or the average of the individual losses for each data j [109, 111].

Since the loss function translates the performances of the model, the bests weights are the ones that minimize said function. Weights optimization is therefore achieved using gradient descent. The idea is that since each trainable parameter is also a parameter of the loss function, these can be updated using the loss gradient. This means that each

weight is modified according to its partial derivative with regard to the loss function [112]:

$$w_n^{(k+1)} = w_n^{(k)} - l_r * \frac{\partial \text{Loss}(w_1^{(k)}, ..., w_N^{(k)})}{\partial w_n^{(k)}} \tag{2.4}$$

Where $w_n^{(k)}$ is the n^{th} weight of the model at iteration k, $\text{Loss}(w_1^{(k)}, ..., w_N^{(k)})$ is the loss function depending on all the weights, and l_r is the learning rate. A learning rate is a factor dampening the initial update value to reduce the step length and allow the gradient descent algorithm to converge properly (see example on figure 2.6) [113]. It can be constant or vary with the iterations, and this learning rate variation strategy is called scheduling. Ideally, the learning rate should start high to allow faster training and end low to ensure convergence [114].

To compute the loss function partial derivatives with regards to the weights, the backpropagation algorithm is used. Its main idea is to express the derivatives of the loss function at one layer in terms of the already computed derivatives of the above layer using the chain rule. This avoids the re-calculation of some terms that could occur using a more naive algorithm [114, 115].

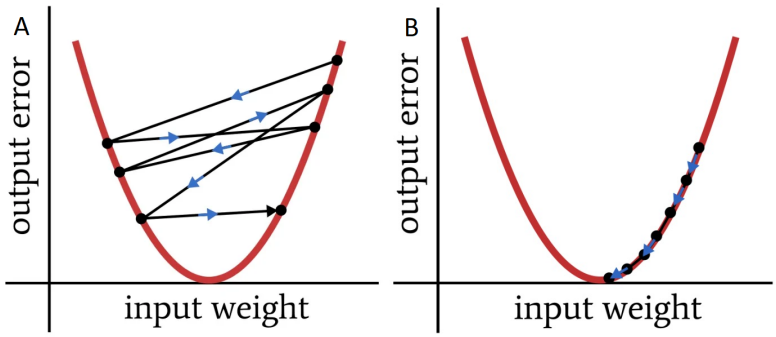

Figure 2.6: One-dimensional example of the weight optimization problem (A) High l_r, difficult convergence to minimum (B) Lower l_r facilitates convergence, but lengthens the training time. Adapted from [113].

The main problem with gradient descent is that the loss function is not convex and there are therefore many local minima or saddle points where the model can be stuck without reaching the global minimum. Furthermore, since gradient descent computes the loss function as a sum of the loss on each data j before making an update, this algorithm can be very time consuming for each iteration.

Stochastic gradient descent (SGD) and mini-batch gradient descent propose to mitigate these two drawbacks by dividing the data into random and fixed size − equal to one for SGD − batches, to compute the loss function, and update the weights on each of them [114]. Once all the batches have gone through the neural network, new batches are drawn and a new iteration − epoch − is started. This random sampling generates stochastic estimates of the loss function such that the local minima and saddle points are commuting, decreasing the chance for the model to converge to one (figure 2.7).

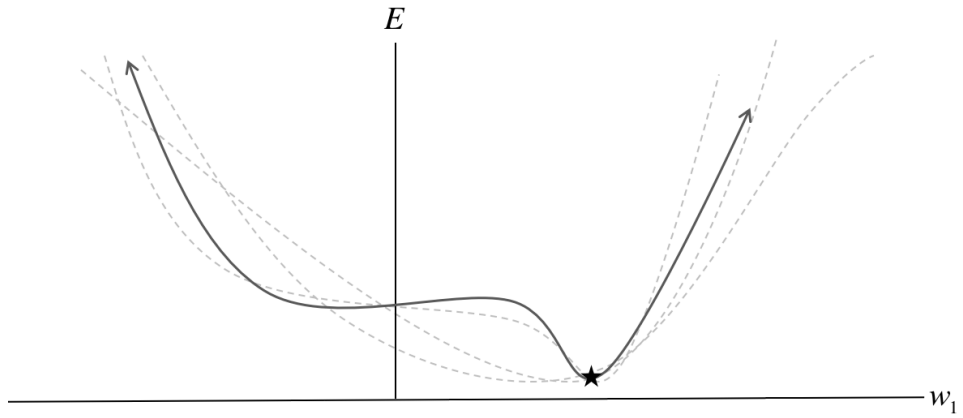

Figure 2.7: Representation of the multiple stochastic estimates of the loss function (discontinuous lines) using SGD or mini-batch gradient descent. The real loss function is shown in black, with the star being the global minimum [114].

To ensure selecting the best weights for generalization, the training data is split between a training and a validation set. The latter is constituted by batches randomly drawn at each epoch accounting for $x\%$ − often 10 to 20% − of the total data volume. For the current epoch, the model will not train on these data and will instead try to predict their label. The model performances on this validation set is used to determine the quality of the current weights [116].

Chapter 3

Detection of neurological disorders with EEG and DL

The final chapter of the first part of this thesis extends on the utility and efficiency of Machine Learning (ML) − and of transformers in particular − to detect neurological disorders in EEG. The first section defines a neurological disorder and provides some medical background on epilepsy, as it is the pathology studied in the use case. The second and third sections summarize the state of the art in the field of intelligent neurological pathology detection in EEG.

The content of this chapter was integrated to this work for two reasons. First, it allows the reader to have a good overview of the pathology tackled in part II, including how the seizures can be sensed via the EEG and why seizure forecasting matters in some treatments. Second, the study of the state of the art on ML and transformers allows to compare the different elements commonly employed to build performant models and pre-processings, which is important for the design of our model in sections 4.2 and 4.3.

3.1 Introduction to neurological disorders and epilepsy

Neurological disorders are any disorder affecting the nervous system (central or peripheral) [117]. The origin of these pathologies can be multiple and multi-factor, including genetic predisposition, bacteria and viruses, physical traumas, and gestational environment [117–119]. They are a global health issue as, in 2005 and in 2010, they accounted for respectively 92.4 [120] and 73.8 millions [121] of Disability-Adjusted Life Years (DALY), which is measured by summing the time of life lost by premature death with the number of years lived with a disability [121, 122]. That is between 2.97% and 6.3% of the total burden for all diseases depending on the year, study, and diseases taken into account [120–122].

One of the commonly known neurological disorders is epilepsy, where an abnormal synchronous firing of a set of neurons in the brain leads to epileptic seizures. The origin of the pathology is either idiopathic − with no known cause − but suspected to have a genetic origin, or linked to a trauma in the central nervous system, like infectious diseases, brain tumors, and strokes [119, 120, 122, 123].

Anti-epileptic drugs are used as a treatment to protect the patients against their seizures by lowering the neurons excitability to block their uncontrollable firing [124], but a significant part of the afflicted population do not use them [121, 123]. A reason for this is that many patients (22.5% according to [125]) have a pharmacoresistant epilepsy, where the effect of the medication is greatly reduced, if not non-existent. Alternative treatments for drug-resistant patients include ablative surgery (but only for candidates meeting some specific criteria) [126, 127] and neurostimulation [127–129].

In the frame of this thesis, one kind of neurostimulation interests us in particular. The Responsive Neurostimulation combines an electrocorticogram (ECoG), which can roughly be compared to an invasive EEG, and a few stimulating electrodes. The objective is to identify biomarkers of a seizure thanks to the ECoG and to neutralize it before the appearance of the clinical symptoms. This is done through the stimulation of the vagus nerve or of the brain nuclei that are believed to spawn the seizure[127–130]. Intelligent identification of biomarkers forecasting a seizure onset is therefore an essential feature of such devices, and research is still exploring the subject, trying to incorporate ML and DL to the field [131, 132].

The EEG signal of epileptic patients is labelled according to when it is recorded with regards to a seizure (figure 3.1). The epileptic seizure is called the ictal period and the periods that directly precede and follow it are respectively called pre-ictal and post-ictal. A period between two seizures is called inter-ictal [3, 133]. The start of an ictal period has many recognizable patterns (figure 3.2) [134], and it is possible to predict its onset due to a frequency synchronization in the delta and gamma bands in the same hemisphere as the seizure during the pre-ictal period [135, 136].

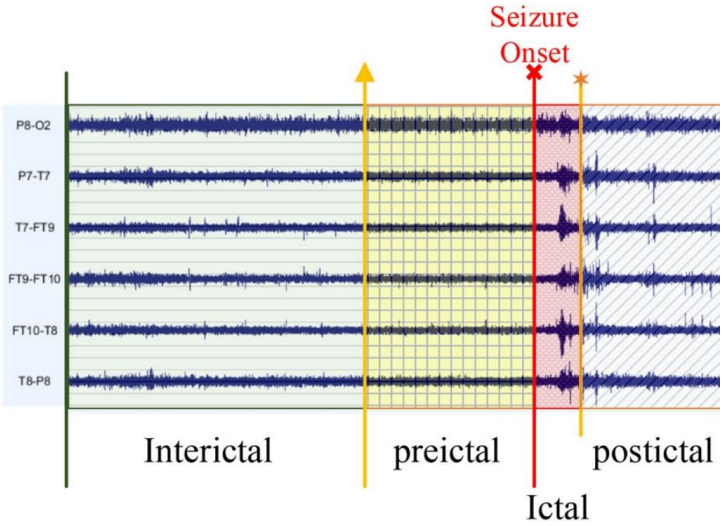

Figure 3.1: Different periods of an epileptic EEG. Adapted from [133].

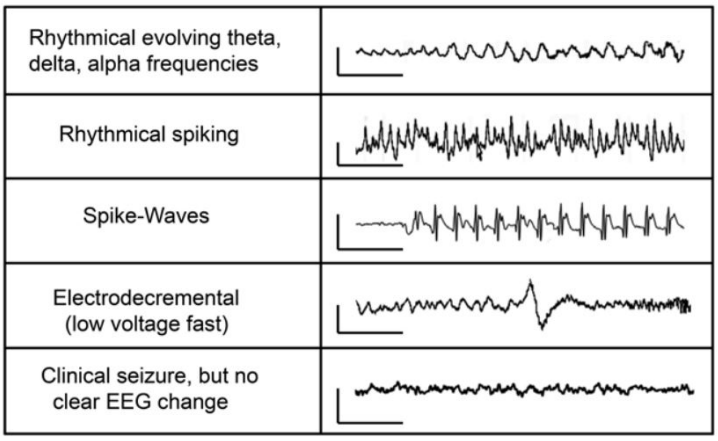

Figure 3.2: Different EEG patterns at the beginning of the seizure [134]. Sometimes, there is a clinical seizure recorded but no change on the EEG (last row). This may come from the fact that all the cerebral activity is not reachable by EEG (see section 1.4).

3.2 Neurological disorders detection in EEG using conventional ML

To efficiently determine the state of the art in the field of machine learning for the detection of neurological disorders, seven elements of the scientific literature have been gathered, including five reviews [137–141], and two articles [142, 143]. Reviews have been sampled either in the 2017-2020 period [137–139] or since the beginning of the year 2023 [140, 141] to notice any evolution present in the recent years. Transformers-based studies were discarded as they are the focus point of the next section.

Since the field of disorders detection using ML and EEG does not seem to make a distinction between neurological disorders and any other brain disorders, and since the strategies used are similar for both, some pathologies like schizophrenia or autism spectrum disorder were included in the state of the art as well.

The first element standing out is the popularity of epilepsy as a case study for EEG classification, as this pathology appeared in every consulted paper, either for seizure prediction [137–139, 141] or detection of positive patients [139, 140, 142, 143]. Other disorders encountered were schizophrenia [140, 142], Alzheimer's disease [138, 142], autism spectrum disorder [139, 143], multiple sclerosis [139], Parkinson [141], and alcohol use disorder [139]. Neurological disorders are not the most popular EEG classification tasks as they only account for a minority of the studies compiled by the reviews [137, 139] and as many reviews in the field simply ignore them.

To pre-process the data, an inescapable step is band-pass filtering, with a low variation in the lower bound among studies (between 0 and 0.1 Hz) and a much higher variation in the upper bound ($40 - 70$ Hz, with a majority of studies between $60 - 70$ Hz) [137, 140, 143]. This is because the physiologic range of EEG signals is located between $0.5 - 50$ Hz

[42, 144]. In EEG, high pass filters are used to remove the baseline drift, low pass filters to remove the higher frequencies noise, and notch filters at 50 (Europe) or 60 (America) Hz to remove the powerline noise [145].

Artifact removal is mostly achieved using Independent Component Analysis (ICA) [137, 140, 142, 143]. This technique assumes that the signal of interest and the artifacts are mixed together in a linear and additive way, and that it is therefore possible to unmix them (up to a permutation and a scaling factor) by finding the inverse of the mixing matrix [146, 147]. However, one review [137] pointed out that older studies tend to have the artefacts removed manually, although this operation takes time and creates variability.

The input of the ML models is one of the three following: pre-processed signal, extracted features, or images. Pre-processed signals, are usually cut into smaller segments (min. duration found: 1s, max.: 50s) to increase their quantity, with the trade-off that smaller segments are less informative. They thus either require more data to train the model or reduce its accuracy. This kind of data is only fed into DL models, as only this type of ML model is able to extract features from raw data [148].

For simpler ML models, computed features like band Power Spectral Density (PSD) [140, 143], or statistical metrics such as the mean, median, variance or kurtosis (i.e., the "tailedness" of the distribution, interpretable as the quantity of outliers [149]) provide excellent results. As a matter of fact, it was demonstrated that there is a significant difference between sane patients and those suffering from certain pathologies regarding these features [142, 143]. Interestingly enough, some papers [137, 143] provided extracted features to ANN instead of feeding them raw data, which made them perform poorly with regards to simpler ML techniques. This policy seems however to have disappeared in recent years.

Finally, images are used as input for a type of ANN called Convolutional Neural Network (CNN), which uses an alternation of learnt convolutional masks to find the image features and pooling layers to downsample these[1] [150]. For these images, the EEG scalograms [137, 140, 141] are preferred over their spectrograms [140], although no reason are given in the reviewed literature.

A spectrogram is the transposition of a signal into an image using the Short-Term Fourier Transform (STFT). This consists in cutting the input into N overlapping segments and performing a Fourier transform on each one of them. The results are then concatenated together to form a heatmap where the x-axis is the time, the y-axis are the frequencies and the pixel brightness is the signal amplitude at a particular time and a particular frequency [151].

Spectrograms are an excellent input for sound-related deep learning tasks as they mimic the way the human cochlea encodes sound [151]. Mel-spectrograms even replace the frequency scale by the Mel scale, which adapts the frequencies such that the pitch is

[1] CNNs also work on one-dimensional signals.

perceived as linear by the human ear [152].

Scalograms are used in a similar fashion to transpose signals into images but using Continuous Wavelet Transform (CWT) instead of STFT. The CWT is the application of the following operation on a signal $x(t)$ for different scalings a and locations b:

$$X_w(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{+\infty} x(t)\bar{\psi}\left(\frac{t-b}{a}\right) dt \tag{3.1}$$

The signal $\bar{\psi}(t,a,b)$ is called a wavelet, which is a signal of finite energy chosen among multiple candidates − which have to follow some prerequisites − for its affinity with $x(t)$. The squared value of $X_w(a,b)$ is then reported on a heatmap at position $(b,\ a_{max} - a)$, forming an image. The parameters a and b encode respectively for the frequency (with larger a capturing the lower frequencies of the signal with a non-linear scaling) and for the time location of the signal [153, 154].

This method differs from the scalogram as its frequency resolution is improved in the lower frequencies at the expense of the resolution in higher frequencies [154, 155]. A comparison of a spectrogram and a scalogram applied to the same signal is shown in figure 3.3.

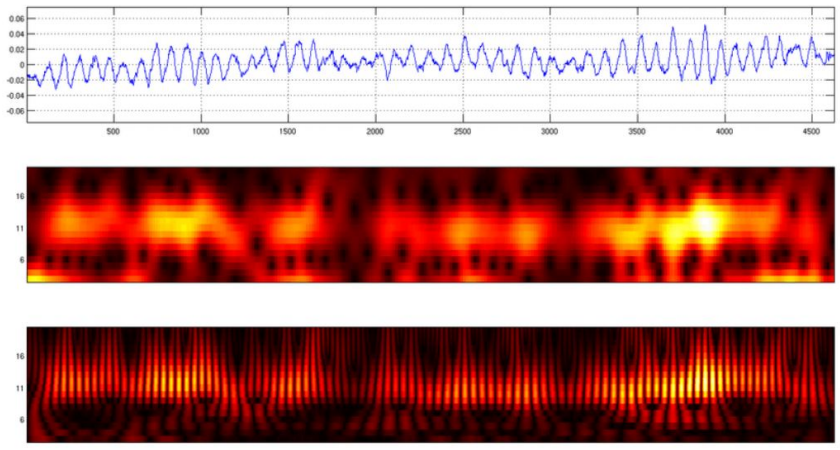

Figure 3.3: Comparison of a spectrogram (middle row) and a scalogram using a Morlet wavelet (bottom row) applied to the same signal (top row). Adapted from [156]. The scalogram is recognizable by the flare at the lower frequencies, indicating a higher frequency resolution and a lower time resolution [154, 155].

Finally, it is interesting to notice how smaller ML models are still used nowadays with the important advances in deep learning. This may be due to the very high amount of data required to train an ANN (For instance, Alwosheel *et al.* [157] proposes to multiply the number of parameters by 50 to know how many data are required), whereas hyperplane separation models only require $N_{features} + 1$ points to fit linearly separable data [158]. However, the highest performances with shorter signals are achieved by neural networks, in particular with CNN and RNN.

3.3 State of the art on EEG classification using transformers

To find inspiration about the design of the model that will be built in part II, literature was browsed to discover transformer-based architectures in the field of EEG classification. This allows for an establishment of the common processing pipelines and architectures. Eight articles [3, 11–14, 82, 159, 160] were kept and their data pre-processing, architectures and limitations are described below. As it can be deduced by the date of publication of these articles (one dating from 2021 and the other seven from 2022), this precise field is still very young and expanding.

The main elements of signal pre-processing for each article are compiled on table 3.1, and we can note several similarities between the different pipelines. For instance, similarly to the other DL models studied in the previous section, the raw signals from each used dataset are split into segments of less than $10s$ (with the exception of Kostas *et al.* [160]). The goal of this step is to have a higher volume of training data, which is highly correlated to the performances of the model [82, 161]. On top of this, filters, when applied, also tend to have the similar physiological bandwidth as for the conventional ML applications studied in the previous section.

Half the papers used standardisation in their pipeline, which is known to reduce the effect of the outliers and to accelerate the training [91]. The two main types of standardisation are minmax, which scales all the data down to the [0,1] interval, and z-score, that scales the data to a zero mean and a unit variance [162]. Three articles also used ICA for artifact removal.

Table 3.1: Pre-processing steps common to almost each transformer application.

Article	Segment duration [s]	Bandwidth [Hz]	Standardization	ICA
Du *et al.* [82]	1 to 5	0.5-42	Z-score	Yes
Hussein *et al.* [3]	10	No filter	No	No
Xie *et al.* [12]	3 or 6	No filter	Z-score	No
Yan *et al.* [14]	5	$\emptyset$: 57-63 $\cup$ 117-123	No	No
Siddhad *et al.* [159]	10	Unspecified	No	Yes
Wang *et al.* [11]	6	4-45	No	Yes
Song *et al.* [13]	2 to 7	Unspecified	Z-score	No
Kostas *et al.* [160]	60	No filter	Minmax	No

The architectures from the selected articles have been clustered in three categories: the Vision Transformer (ViT) inspired models, the models using raw EEGs, and the combinations of transformers with CNNs.

A comparison of the different architectures performances would be counterproductive, as the models are trained on different tasks − with some easier than others − and on datasets differing in size and quality. Comparison would therefore require to train all the

models on the same tasks and datasets to benchmark their performances and notice an eventual trend, which alone could be the subject of a thesis!

Vision transformers-based models

Dosovitskiy *et al.* [2] designed a transformer for image classification that splits an image into non-overlapping patches before flattening them out and feeding them through an encoder after positional embedding. The output is processed by a MLP (see section 2.1) to provide a class to the image. This is the Vision Transformer.

Among the three articles inspired by ViTs, Hussein *et al.* [3] has the closest implementation to the original idea. The architecture (figure 3.4) of its model is an array of independent transformer encoders that each take the patched and embedded scalogram (see previous section) of one EEG channel as input. The outputs of all the encoders are concatenated together before the MLP classifier.

This model, called multichannel ViT (MViT) is used for epileptic seizure prediction, achieving between 91.15 and 99.8% accuracy − depending on the dataset size and quality − with a False Positive Rate (FPR) of 0.004 false alert per hour, uplifting the state of the art for the three datasets it has been tested on. However, the authors warned about the necessary computational resources of such an implementation as well as the non-interpretability of the model decision.

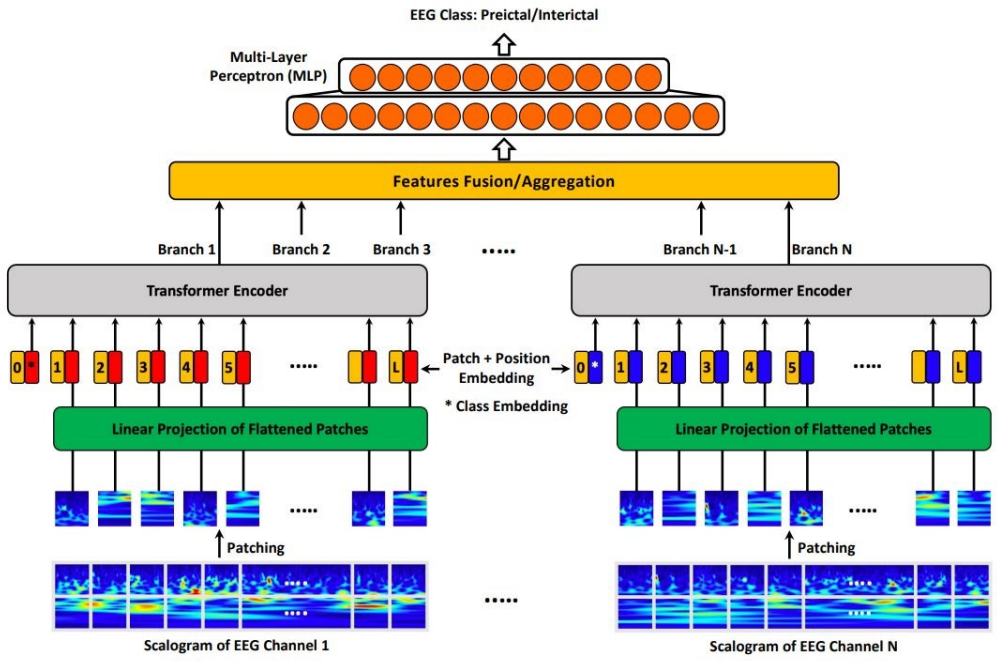

Figure 3.4: Architecture of a MViT. Each channel is encoded independently as a scalogram image before being aggregated for classification. Adapted from [3].

Yan *et al.* [14] also tries to predict epileptic seizures and postulate that the input EEG data, once transformed into an image by STFT (see previous section), is three-dimensional

as there are as many 2D images as there are channels. It therefore proposes to lower the dimensional space to two by sequentially flattening the data along the frequency, time and channel axis. Each subspace is encoded by its own independent transformer and the outputs are concatenated to a MLP classifier (figure 3.5).

With an accuracy of over 90% for every patient and a FPR of 0.047 false alerts per hour, this model also achieves state of the art on its dataset. The architecture however still needs to demonstrate its generalization ability, as the models from this article are patient specific (i.e., the train and test sets are issued from the same patient). Since each patient has seizures with varying characteristics (localization, onset patterns...), a model trained solely on one person will perform poorly with other subjects. However, it may be arguable that such personalized models should be preferred as they result in better performances on the patient they are trained [163, 164].

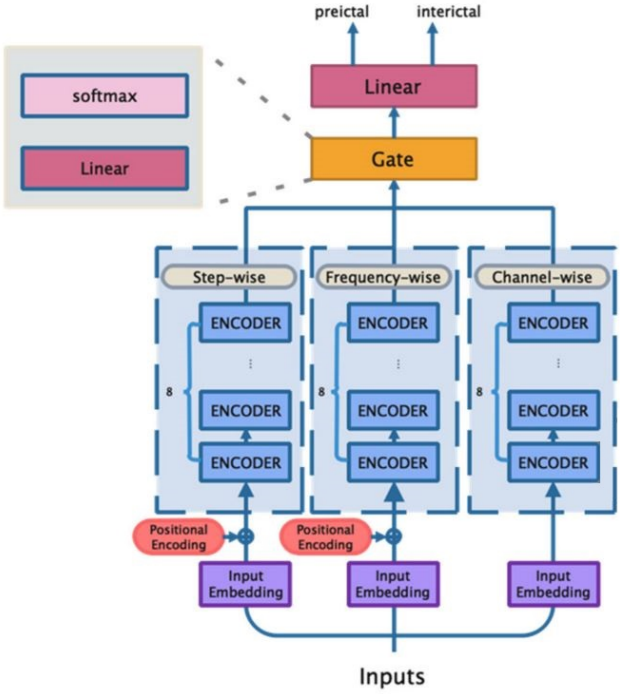

Figure 3.5: Architecture of the transformer proposed by Yan *et al.* Each ViT encodes and image composed of one main dimension and the concatenation of two other. Adapted from [14].

The third ViT-inspired architecture (Wang *et al.* [11], figure 3.6) is the only one of the selected corpus to require extracted features as input, as for each electrode the Power Spectral Density (PSD) of each of the five main physiological bands is computed like in a non-deep learning ML pipeline. The electrodes are clustered together depending on the lobe they are located on, and each cluster is studied by an independent transformer encoder. Electrodes receive Positional Encoding (PE, see section 2.2) within a cluster. The outputs of each encoder are embedded together and fed to another encoder that will determine the class of the data (in this case an emotional state).

Unlike with Yan *et al.*, this model was made subject independent by training the dataset on all patients' EEGs but one, used as a test set (with cross-validation). The model only obtained an accuracy ranging between 65.75 and 66.63% on a binary emotion dataset, but this was sufficient to improve the state of the art for this dataset.

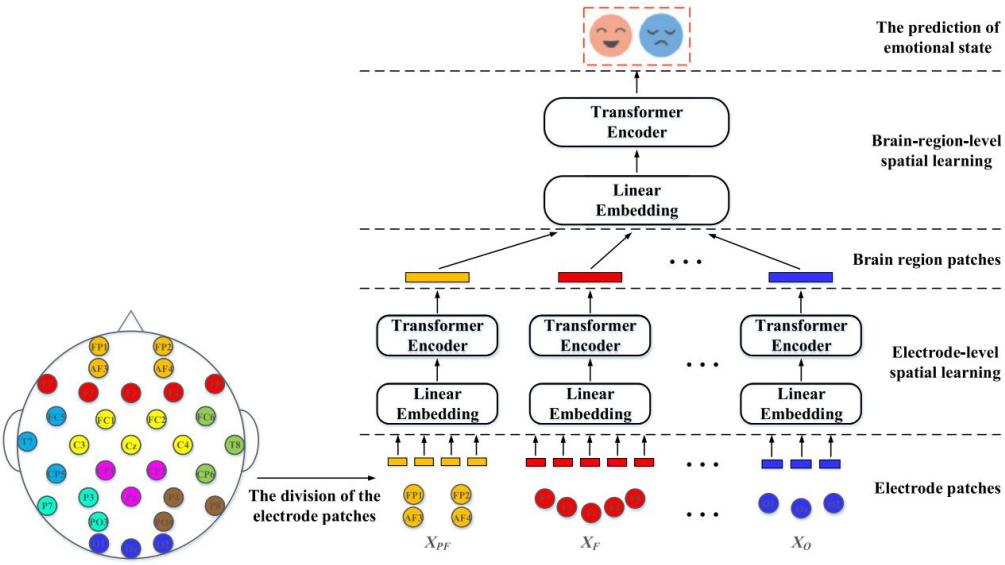

Figure 3.6: Architecture of the transformer proposed by Wang *et al.* The electrodes are clustered depending on their spatial proximity and their extracted features are encoded by two layers of transformers [11].

Models with raw EEG input

The simplest architecture of the corpus, proposed by Siddhad *et al.* [159], solely consists in a MLP stacked on top of a transformer encoder that reads the input after a 32-dimensional embedding and PE addition along the temporal axis.

Nevertheless, this simplistic model scores a state of the art-comparable accuracy on an age-gender prediction dataset (94.7% in gender accuracy and 87.6% in age category accuracy, versus respectively 97.5% and 93.7% for the best model), and beats the best models on another task prediction dataset, achieving 95.3% accuracy in binary classification and 89.0% in ternary classification.

A more complex architecture is the EEG Temporal Spatial Transformer (ETST). Du *et al.* [82] took advantage of the multichannel output of the EEG and fed the data through two encoders in series, varying the encoding axis to be temporal at first and then spatial (i.e. on channels interaction) before classification by a MLP (figure 3.7).

The model aims at identifying people based on their EEG and manages 100% accuracy on a single state dataset (i.e. the subject to identify was recorded multiple times in the same settings) and 97.29% accuracy on a multiple state dataset (the subject was given a different task for each record).

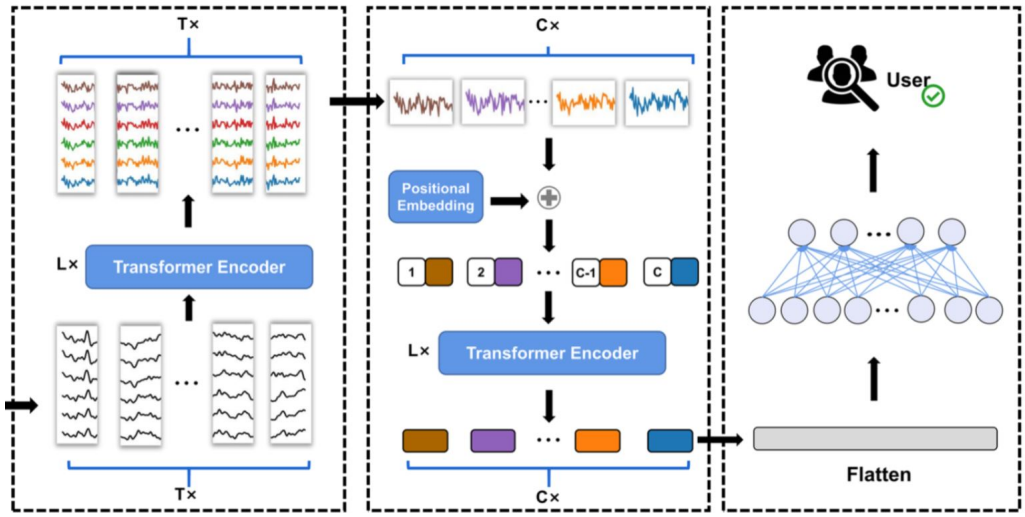

Figure 3.7: Architecture of the ETST. This transformer uses two encoders to encode data following the temporal axis (T tokens of size C) before encoding the output following the spatial axis, after positional embedding (C tokens). Adapted from [82].

This article further studied the effect of PE and discovered, using an ablation study (i.e., some part of the full model are deleted and the consequences on the performances are observed), that the encoding of the spatial axis improved the performances whereas the encoding of the temporal axis strongly harmed them. These results are in opposition with was is proposed by Yan *et al.* [14] and Siddhad *et al.* [159], as the authors of both papers encoded the temporal axis. Since they did not provide any justification for this choice, and before the introduction of any proof, the PE of only the spatial axis should be more trusted as a strategy.

CNN-transformers hybrids

Two limitations of CNNs (which, as described in the previous section, are ANNs able to extract features from images) are their rough gradient and their low ability to discover long distance dependencies in images [12, 13, 161]. Researchers therefore combine them with transformers to exploit their local features extraction ability while mitigating their flaws [12, 13, 160].

For instance, Xie *et al.* [12] stacked a transformer encoder on top of either one or two CNNs, adding PE between the first and second level (figure 3.8-A,B,C). This family of models achieves state of the art on a multi-class motor imagery (MI) dataset (88.0% accuracy on binary classification, and 64.2% on a 4 class classification). In MI tasks, the subject is asked to imagine doing a precise movement while its EEG is being recorded.

On top of these results, the article compares the performances of an CNN-transformer hybrid encoding along the temporal axis, the spatial axis, and both axis. It however did not find any significant difference, as each strategy performs better on different datasets. The authors also compare these hybrid architectures with a purely transformer-based

architecture (similar to the one from Siddhad *et al.* [159]), and observed that the hybrid models were systematically better at MI classification tasks.

Finally, different PE methods were compared and no significant differences in performances were found between sine wave PE and learnable PE, which confirms the results of Vaswani *et al.* [1] (see the paragraph about positional encoding in section 2.2).

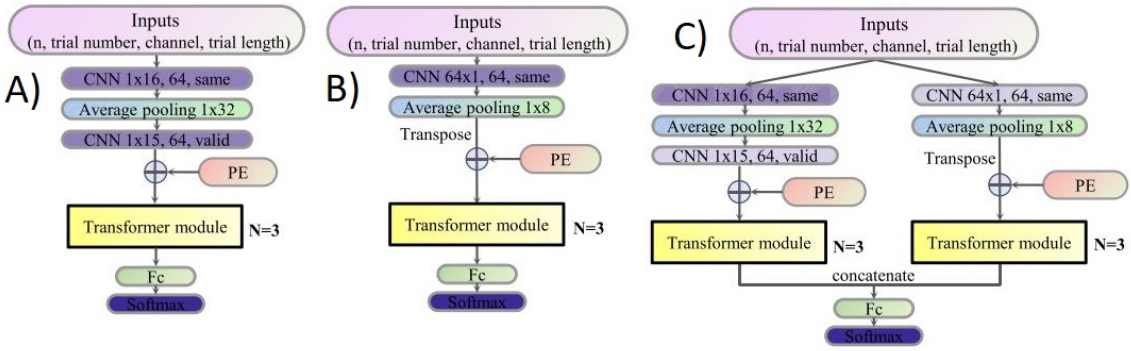

Figure 3.8: Architecture of the hybrid models proposed by Xie *et al.* (A) CNN-transformer encoding on spatial axis (B) CNN-transformer encoding on temporal axis (C) CNN-transformer encoding on both spatial and temporal axis. Adapted from [12].

Similarly to Xie *et al.*, Song *et al.* also uses a CNN as a feature extractor and adds a second level to the model as well. The difference is that this second level only contains self-attention layers instead of a complete transformer encoder. On top of these two stages, a MLP decision layer is added for classification (figure 3.9). Both articles also differ in the input format, as the convolutionnal layer extracts the features from an image formed by the EEG channels stacked on one another in the case of Song *et al.*

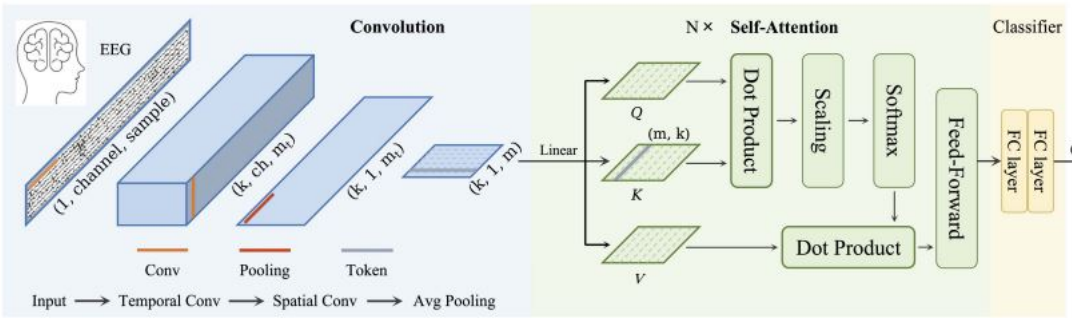

Figure 3.9: Architecture of the Conformer. A first level of CNN extracts the features from the EEG, a second level of self-attention extracts the long distance dependencies in the data and the third level MLP classifies the data based on the second level output [13].

This so-called "Conformer" improves the state of the art on two MI datasets (with accuracies of 78.7% and 84.6%) as well as in one emotion recognition dataset (95.3% accuracy). The ablation experiment conducted by this article allows to show that with a CNN as feature extractor, the depth of the attention layer (i.e. the number of attention

layers in series) and the number of attention heads can be reduced to 1 without significantly harming the accuracy.

Finally, Kostas *et al.* [160] also uses a CNN-transformer stack on an image made by the superposition of EEG channels (figure 3.10), but with the objective to learn EEG data representations using Self-Supervised Learning (SSL).

The task implemented to achieve this goal is a contrastive learning task inspired from another article aiming at building speech representations using the same architecture [165], itself inspired by the methodology used to train BERT (see section 2.3). In this task, a part of the EEG signal is masked after embedding by the CNN, and the model is proposed different segments to fill the gap and must choose which one is the most likely to be correct.

Although it is not mentioned in the article, this contrastive pre-training lacks interpretability, as the transformer learns by recovering segments of already embedded data. In other words, we do not know if the data representations of the model are the same as the ones a human could have.

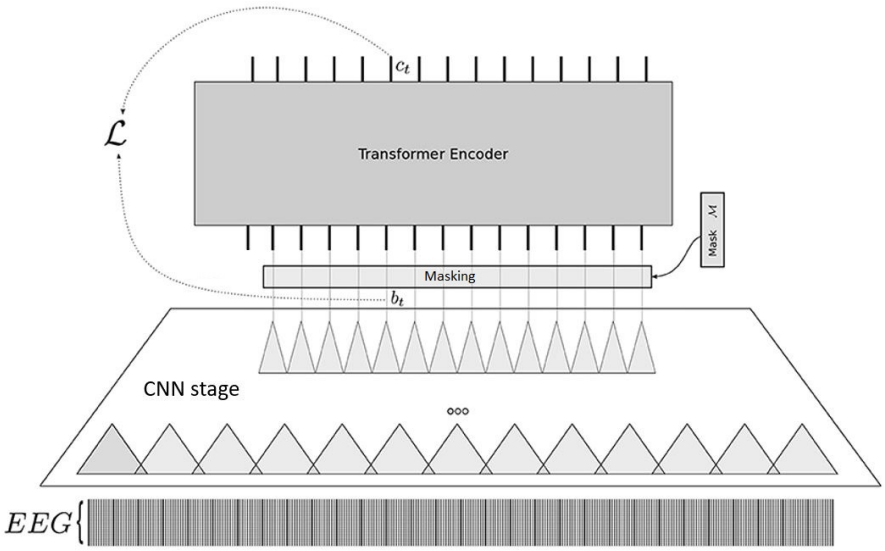

Figure 3.10: Architecture used by Kostas *et al.*. The CNN stage embeds the EEG channels stacked on one another before that some elements of these representations are masked. The encoder stage is then trained to recover the proper embedding between different propositions. Adapted from [160].

Conclusions on the transformer models for EEG classification

The observations resulting from the overview of the transformers architectures used in EEG classification are summarizable in three points. Firstly, it is observed that the multi-dimensional properties of the multichannel EEG were mostly dealt with by transforming the array of signals into an image [3, 13, 14, 160]. This transformation is either achieved using CWT [3] or STFT [14], but the most common method consists in stacking the EEG

channels into an image where the x-axis is time, the y-axis is the channel number and the pixel brightness is the signal intensity at a given time and channel [13, 160]. This latter transformation is the one used with architectures integrating a CNN.

Other solutions to deal with the multi-axis problem inherent to EEGs are either the successive [82] or parallel [12] embeddings of the different axis of the signal by different encoders, or the extraction of one-dimensional features or embeddings to input to the model [11, 159].

In second, the global similarities between all the proposed architectures should be highlighted. All but one studied models are composed of three stages: a first embedding, a transformer encoder, and a decision layer.

The first embedding is either a simple fully connected linear layer (identical to a MLP, but without hidden layers nor activation function) [3, 11, 14, 82, 159] or a CNN [12, 13, 160]. The second level is composed of transformer blocks that can either be unique [12, 13, 159, 160], in series [82] or in parallel [3, 11, 12, 14] and that allow to generate representations from the embedding from the previous layers using attention. Then, a last stage feeds on the encoded representations to take a decision on the class of the input data. This layer is almost always a MLP [3, 13, 82, 159] or a fully connected linear layer [12, 14], but one article used a second transformer encoder [11].

Kostas *et al.* [160] does not have a third stage because its objective is to pre-train the first two layers so that they require less training time for further applications. A majority of the architectures comprehends PE; either after stage I [3, 11, 12, 14, 159], or in the middle of stage II [82].

Finally, it should be noted that all articles but Kostas *et al.* [160] used Supervised Learning (SL, see section 2.3) to train their model, meaning they had to use fully annotated datasets. Table 3.2 (next page) summarizes the conclusions drawn from the state of the art by comparing the different architectures according to the aforementioned points.

Table 3.2: Comparison of the transformer-based architectures.

Article	Hussein *et al.* [3]	Yan *et al.* [14]	Wang *et al.* [11]	Siddhad *et al.* [159]
Category	ViT	ViT	ViT	Raw signal
Input type	Image	Image	Extracted features	EEG signal
Stage I	Linear layer	Linear layer	Linear layer	Linear layer
Stage II	Parallel	Parallel	Parallel	Unique
Stage III	MLP	Linear layer	Encoder	MLP
PE	Stage I	Stage I	Stage I	Stage I
Learning type	SL	SL	SL	SL
Article	Du *et al.* [82]	Xie *et al.* [12]	Song *et al.* [13]	Kostas *et al.* [160]
Category	Raw signal	CNN	CNN	CNN
Input type	EEG signal	EEG signal	Image	Image
Stage I	Linear layer	CNN	CNN	CNN
Stage II	Series	Parallel or unique	Unique	Unique
Stage III	MLP	Linear layer	MLP	NA
PE	Stage II	Stage I	No	No
Learning type	SL	SL	SL	SSL

Part II

Classification of pre-ictal and inter-ictal EEG segments using a transformer network

Chapter 4

Methods

This chapter first explains how we elaborated a transformer-based model able to distinguish pre-ictal from inter-ictal EEG segments, and to therefore forecast epileptic seizures. In particular, the used datasets and their specifications are detailed, as well as how the data is pre-processed to be suitable for the model. The architecture of the latter is also explained, with emphasis on components that were not overviewed in the first part of this thesis.

Afterwards, we present a walkthrough of the design of different pre-training strategies that are based on the intrinsic properties of EEG. A last section is dedicated to the metrics that are used to assess the performances of the models.

This chapters only focuses on what are the features implemented to the pre-processing or the model. For more details on how these are implemented, the reader is invited to consult annex A, giving a rough overview of how the `python` code is designed. The link to the documented GitHub repository, where further details are provided, is also available both in the abstract and annex A.

4.1 The datasets

Two datasets were used for the experimental part of this thesis, each related to a different EEG classification task and serving different purposes in the model elaboration.

The eyes open/eyes closed (EO/EC) dataset

This dataset [166] was initially linked to a master course project in biomedical engineering at UCLouvain. The subject is a male in the $20 - 25$ age range with no previous abnormal neurological history. The data consists in a single, approximately 11 minutes-long, 32 channels EEG signal in unipolar montage, recorded at a 1 kHz frequency. It is annotated with three labels; 0 when the eyes are open, 1 when they are closed and NaN in case the state is unknown (at the beginning and end of the experiment).

This data was recorded in the UCLouvain Woluwe's laboratory using the following

protocol : the subject was asked to open or close his eyes by an operator that activated a switch to automatically switch the label from 0 to 1 or from 1 to 0 in the annotations. However, since there is a reaction time between the operator's injunction and the patient's reaction, and since the label for the eye state is changed as soon as the order is given, there is a period of time after each change of eye state where the label provided does not correspond to the ground truth. Both the data and the label have been stored in `.csv` files as respectively a (t x ch) matrix and a (t x 1) vector, where t represents the time steps and ch the channels.

Given the small size of this dataset (only 11 minutes of recordings from a single subject) and the simplicity of the task (as a dominant alpha rhythm should be expected during eyes closed, and a decrease in all bands should be observed when eyes are opened [167, 168]), it was solely used to experiment on the different pre-training elaborated and to select the most promising one.

The Temple University Hospital EEG Seizure Corpus (TUSZ)

This database [8] is part of a bigger corpus created to stimulate the development of EEG processing tools [169]. The TUSZ contains EEG signals recorded on 579 patients over 1175 sessions, 352 of which comprehend seizures.

The recorded signal for each session is stored in an `.edf` file[1] whereas information on − the absence of − seizure events is joint in two `.csv` files having the same name as the data file they annotate and containing two different levels of details. For this thesis, only the less detailed annotation file is exploited as the other one provides context beyond what is required for seizure prediction (such as elaborations on the type of seizure and what channel it affected for instance). The simpler annotation file solely contains the beginning and end of each seizure event occurring during the session.

One of four possible unipolar EEG montages were used to record each of the signals. Two of them use an average reference (see section 1.2) while the two others use two reference electrodes located on both ear lobes and linked together. On top of this, half the montages use 19 electrodes and the other half use 21. The two supplementary electrodes are located on the left and right preauricular areas (electrodes A_1 and A_2 on figure 1.4, section 1.2). Besides that difference, all electrodes are placed on the skull following the 10-20 system [171].

Due to the abundance of data the TUSZ offers, this corpus is employed in this thesis to train a bigger version of the model designed for the EO/EC dataset. This is achieved both to obtain a model completing a task often presented in the literature, and to test the hypothesis that pre-training such model with one of our datasets leads to better performances than no pre-training in realistic settings.

[1] EDF stands for European Data Format, and this type of file often chosen to store multichannel medical data such as EEG [170].

4.2 Pre-processing

Since the architecture of the model created is greatly inspired by the MViT of Hussein *et al.* [3] (see section 3.3), it is necessary to transform the raw data from each dataset into an image, which is processable by this architecture. Furthermore, while the TUSZ does not explicitly label the data as "pre-ictal" (i.e. right before a seizure) or "inter-ictal" (i.e. with no imminent seizure) that are necessary for epileptic seizure prediction, these states can be inferred from the existing labels. This re-labelization must also be achieved during pre-processing. The pre-processing pipeline for both datasets is shown on figure 4.1 and explained in the following paragraphs.

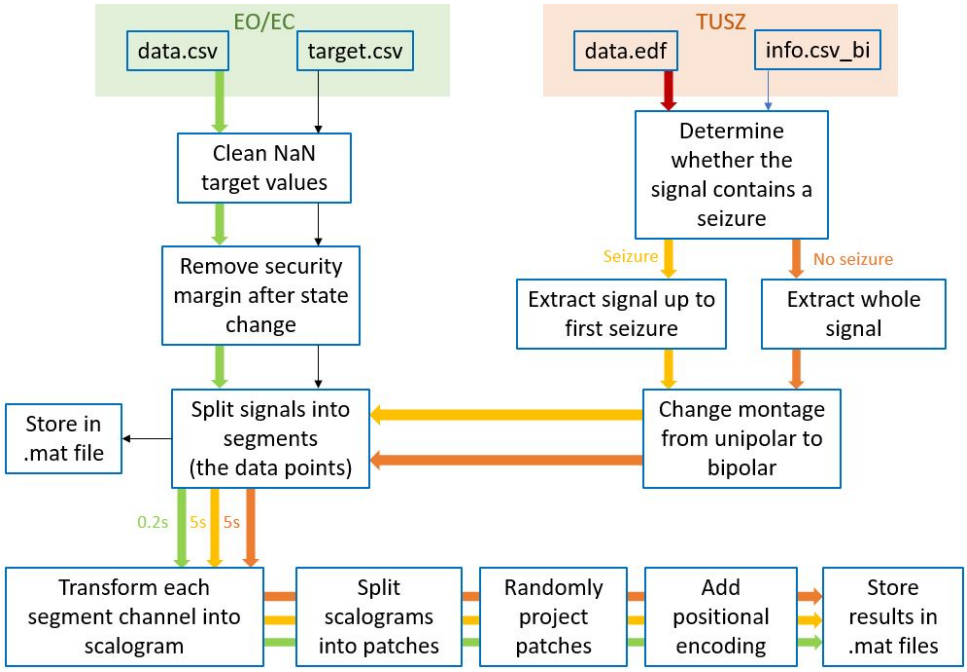

Figure 4.1: Data pre-processing pipeline for the TUSZ and EO/EC datasets. The TUSZ pipeline is repeated over all files present in the corpus.

Data extraction from the EO/EC dataset Since the data constituting this dataset is stored in a single `.csv` file, the whole file as well as the target are loaded in two numpy arrays. The first step consists in trimming the parts of the EEG that correspond to NaN values in the target vector, since these are not exploitable as training or test data. As NaN values are only present at the beginning and end of the session, this trimming does not generate any discontinuity in the signal.

Another pruning operation is the removal of a certain amount of data after a state change in the target vector. This is done because, as stated in the dataset description, there is a delay between the moment when the target changes and the moment when the subject effectively close its eyes. This lag has been determined to be of approximately one second, but as a safety measure to avoid miss-annotated data in the ground truth, 1.5 second worth of data is removed after each state change.

Data extraction from the TUSZ dataset The major objective of the first part of the TUSZ extraction pipeline is to identify which EEGs can be labelled as pre-ictal and which can be labeled as inter-ictal (those terms are defined in section 3.1). To do so, the names of all `.edf` files in the tree are separated in two lists depending on whether the file contains a seizure or not. This is done by searching for seizure events in the `.csv` file linked to each data file and sort according to whether any is found.

For the files stored in the "seizure" list, the data is extracted from the beginning of the file up to the first seizure event (minus a certain security margin, here 5 seconds). This data is labeled as pre-ictal as it has been extracted little time before a seizure, meaning that a model able to detect pre-ictal EEG is able to predict an imminent seizure. For the "non-seizure" or "background" list, the whole signal is extracted and labeled as inter-ictal since no epileptic seizure happened during the sessions.

As pre-ictal signal is very scarce in the TUSZ, the number of pre-ictal data obtained is used to stop the inter-ictal data extraction such that both labels are present in the same quantity after this step. This allows a balanced dataset, which is important to avoid the effects of class imbalance that lower the accuracy of the minority class (here the pre-ictal data), which is the most critical one in our case [172]. However, undersampling the majority class this way drastically reduces the total amount of training data as it now is limited by the minority class.

Also, in opposition to the EO/EC dataset that already has its labels shuffled and kept in a target vector, the pre-ictal and inter-ictal labelled data are pre-processed separately and then stored in two different directories. They are mixed together and a track of their label is kept only after they are called for model training.

Montage conversion from unipolar to bipolar As stated before, the EEGs from the TUSZ can originate from four different montages. To use signals collected with each of them in the training set, it is therefore necessary to convert them all into a common montage. Theoretical information about montages can be retrieved from section 1.2

Firstly, the electrodes used are similar between all montages and placed at the same position, with the exception of the A_1 and A_2 which are sometimes missing. The solution to uniformize montages with and without these electrodes is therefore to discard them when they are present. The information loss due to this operation is supposed to be less detrimental to the model than dropping half the available data.

To solve the disparity in reference across the montages, it is necessary to convert them from unipolar (where the difference of potential is computed with regards to the reference) to bipolar (where the difference is computed with regards to a surrounding electrode). To do so, the recommendations of the report accompanying the data are followed [171], subtracting electrodes to get rid of the reference. The conversion between unipolar and bipolar montages is reported on table 4.1. After conversion, the EEG is constituted of 20 channels.

Table 4.1: Conversion from unipolar to bipolar montage as recommended by [171]. REF stands for the reference, whether it is an average reference or electrode reference.

Channel	Montage	Channel	Montage
1	F_{p1}-REF − F_7-REF	11	C_Z-REF − F_4-REF
2	F_7-REF − T_3-REF	12	C_4-REF − T_4-REF
3	T_3-REF − T_5-REF	13	F_{p1}-REF − F_3-REF
4	T_5-REF − O_1-REF	14	F_3-REF − C_3-REF
5	F_{p2}-REF − F_8-REF	15	C_3-REF − P_3-REF
6	F_8-REF − T_4-REF	16	P_3-REF − O_1-REF
7	T_4-REF − T_6-REF	17	F_{p2}-REF − F_4-REF
8	T_6-REF − O_2-REF	18	F_4-REF − C_4-REF
9	T_3-REF − C_3-REF	19	C_4-REF − P_4-REF
10	C_3-REF − C_Z-REF	20	P_4-REF − O_2-REF

Splitting signals into segments As seen in section 2.2, one downside of transformer networks is that they require a big training data volume. To multiply the amount of data available for training, the extracted signals must be split into segments of determined length. This operation comes at a cost, as all information on phenomenons longer than the segment length is lost in the process.

The task of the EO/EC dataset (i.e. classified EEG based on whether the subject's eyes are open or closed) being rather simple and the total signal duration being short, a $200ms$ segment duration was chosen. This blocks the observation of phenomenons with a frequency below $5Hz$, which corresponds to the delta waves physiologic band. However, this is not a problem as these waves should mainly be observed when the subject is asleep.

For the signals from the TUSZ, a length of $5s$ has been chosen as being a proper trade-off between information per data point and number of data points. This decision has also been guided by the segment duration for transformer networks tasks in the state of the art (section 3.3), ranging between $1 - 10$ seconds. This splitting step allows to obtain 2 216 data points for the EO/EC dataset and 54 589 data points for TUSZ.

Transformation into scalograms The MViT architecture is based on the Vision Transformer (ViT) that requires images as input. Between the two signal-to-image transformations described earlier in this thesis (see section 3.2), Continuous Wavelet Transform (CWT) was arbitrarily chosen over Short Term Fourier Transform (STFT), simply because it was the transformation used by Hussein *et al.* [3].

An interesting experiment would be to compare the performances of two identical models, but having one trained on spectrograms (STFT output) and the other one on scalograms (CWT output). This idea has however not been explored in this thesis as both transformations are very time consuming and duplicating the dataset would therefore have taken much time.

The wavelet used for the CWT is from the Morlet family (figure 4.2) as these are the most commonly used in EEG analysis. Morlet wavelets are defined as complex sine waves tapered by Gaussian windows [170]. To make the resulting scalogram a square, which facilitates patching, as many scaling parameters a were used as location parameters b. Locations correspond to a translation of the wavelet to every sample of the signal (i.e. a range between 0 and 200 for EO/EC and between 0 and 1250 for TUSZ, with steps of 1 for both).

To determine the value of the scaling parameters, we first generate an array ranging from the lowest recordable frequency in the signal (i.e. $5Hz$ for EO/EC and $0.2Hz$ for TUSZ) to an upper frequency. The obtained frequencies are then converted into the appropriate scales using the fact that the scaling of a wavelet shifts its main frequency in the Fourier domain, and that it is therefore possible to map the first to the second [173]. The upper frequency has been arbitrarily fixed to $80Hz$, as this allows to capture the physiological frequency range. An example of scalogram for the EO/EC segment is shown on figure 4.3.

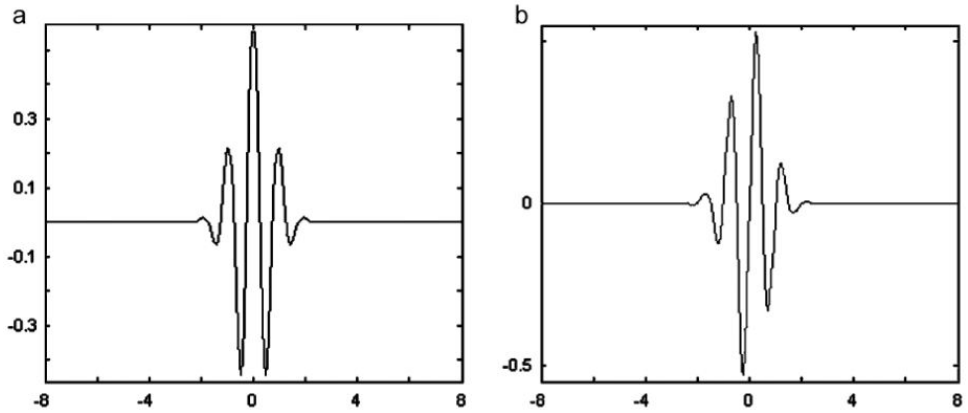

Figure 4.2: Real (a) and imaginary (b) parts of the Morlet mother wavelet, from [174].

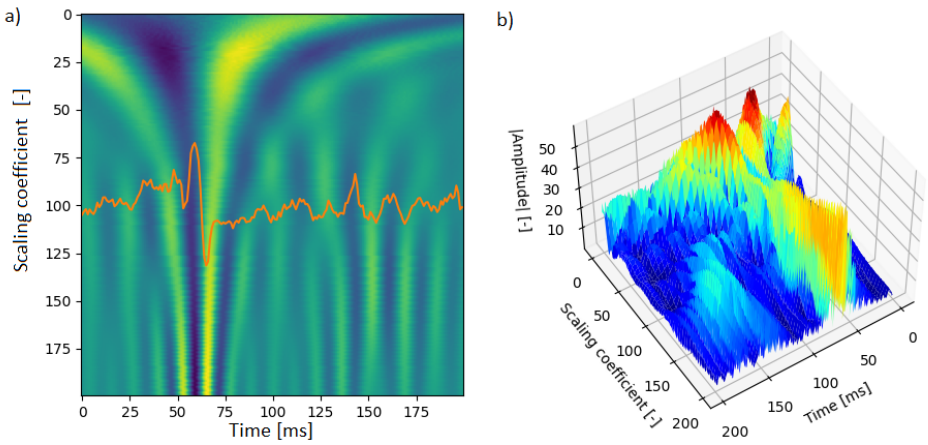

Figure 4.3: Scalogram of the $F_{p1} - REF$ channel for one segment of the EO/EC dataset. (a) is the heatmap with the original signal superimposed for comparison, (b) is the 3D plot of the heatmap. Lower scaling coefficients correspond to lower frequencies.

The transformation to scalograms is very time consuming, especially when the demanded resolution is high. It should therefore be noted that a smaller number of a and b parameters could significantly reduce the pre-processing time without curbing the quantity of information present in the scalogram (mainly since the images are compressed with loss in the next step).

Unfortunately, the pre-processing was already completed when this was noticed, so the data was kept. If pre-processing needed to be realized again, adjustments would have been made such that the resolution of the scalogram does not scale with the resolution of the signals (i.e. the number of pixels in the image would not be equal to the square of the number of samples in the segment), but would instead be fixed to a reasonable value. This modification would lead to a controllable and standardized size of the images across the EO/EC and TUSZ datasets. Readers are invited to consult section 6.1 for more details on how this step could be improved.

Patching and random projection In order for a ViT to process an image efficiently, it has to be split into several patches [2, 3, 14]. For this specific application, the number of patches has been set to 25, as scalograms from both the EO/EC and the TUSZ dataset could be split in that number of sub-images without requiring padding nor overlapping.

Due to the square shape of the scalograms, this amounts to splitting the images in 5 vertically and dividing each of the resulting rectangle in 5 horizontally (figure 4.4). The ensuing patches are 40x40 pixels for EO/EC and 250x250 pixels for TUSZ. This image size is a direct consequence of the scalograms being dependent on the signal resolution, which limitations are discussed in the previous step.

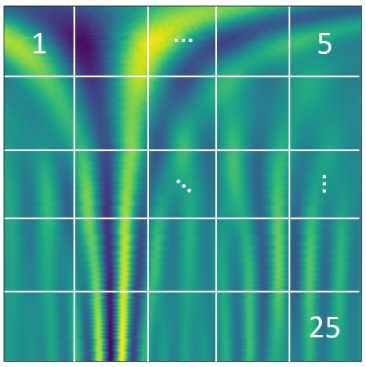

Figure 4.4: Division of the scalogram in 25 equal patches. The patch number follows in what order it is embedded.

Dosovitskiy *et al.* [2] and Hussein *et al.* [3] both used a trainable linear layer to embed the patches to a lower dimensional space. The problem with that layer is that, since it needs to be trained along with the rest of the model, the high dimensional patches should be stored before being embedded and would therefore occupy much more disk space.

For instance, the set of patches for one data segment dumped into an uncompressed `.mat` file respectively takes 1.28 Mo and 31.25 Mo of disk space for the EO/EC and the

TUSZ settings, which adds up to 2.84 Go and 1.7 To to store the whole pre-processed datasets.

Not storing the scalograms and instead producing and destructing them when required would also be inefficient since, as stated before, the CWT is a computationally expensive operation (the generation of a scalograms takes 0.53s in the EO/EC pipeline and 4.7s in the TUSZ pipeline[2]).

The alternative embedding exploited in this thesis that does not require any kind of training is called random projection. This method simply uses a (m x p) matrix with random weights which multiplies the data to be embedded (of dimension p) to project it to a m dimensional space. This matrix can even be made sparse to reduce computation and memory costs [175].

The loss of information imputed to this transformation is quantifiable using the Johnson-Lindenstrauss lemma [176, 177]. This lemma states that, given a distortion $0 < \epsilon < 1$, there exists a projection $f : \mathbb{R}^p \to \mathbb{R}^{m<p}$ mapping any set of n points in p dimensions to a $m \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n$ dimensional space such that the ratio of distances between any pair of points after embedding over before embedding is in $[1-\epsilon,\ 1+\epsilon]$ [177].

In our case, the embedding is made from a space $A \in \mathbb{R}^{n_{channels} \times n_{patches} \times n_{pixels/patch}}$ to a space $B \in \mathbb{R}^{n_{channels} \times n_{patches} \times n_{embedding}}$. If the three axis of the input tensor are flattened, this yields theoretical distortions ϵ of 0.102 ($n_{embedding} = 8$) and 0.068 ($n_{embedding} = 40$) for datasets EO/EC and TUSZ respectively. The $n_{embedding}$ values are justified at the end of this section as this requires some context that has not been set yet.

After projection, a data segment dumped in a `.mat` file weights 38 Ko for the EO/EC dataset (84.21 Mo for all segments) and 118 Ko for the TUSZ (6.44 Go in total). This corresponds to compression ratios[3] of 33.7 and 264.8 respectively. This kind of operation does however not provide the best embedding for the data as a trained linear layer would be adjusted to a better encoding through backpropagation (see section 2.4).

This method of compression is far from ideal, as the random projection from a high dimensional space to a low dimensional space destroys the original image, removing a lot of the model's interpretability. Furthermore, since the distortion ϵ scales with the number of points n in the set, the random embedding aggravates the information loss if the selected dataset is bigger. Alternatives to this pre-processing step are discussed in section 6.1, along with ways to improve the transformation of signals into scalograms − therefore requiring less compression downstream.

Positional encoding As detailed in section 2.2, Positional Encoding (PE) is added to the embedded data to help the model keeping track with the tokens locations. Dosovitskiy *et al.* [2] (and probably Hussein *et al.* [3] too, although not specifying it) used trainable

[2]On a single Intel Core i5-7300HQ CPU working at 2.5 GHz

[3]Compression Ratio = $\frac{\text{Size of uncompressed data}}{\text{Size of compressed data}}$ [178]

weights for PE. However, this requires a trainable layer that could potentially be replaced by fixed weights with a similar performance. For instance, fixed weights in Vaswani *et al.* [1] follow:

$$PE_{(pos,2i)} = \sin(pos * 10000^{-2i/d_{depth}}) \tag{4.1}$$

$$PE_{(pos,2i+1)} = \cos(pos * 10000^{-2i/d_{depth}}) \tag{4.2}$$

Where pos is the position of the token in the 1D sequence, i is an index representing the i^{th} element of the token, and d_{depth} is the number of elements in the token.

However, these weight can only encode position in 1D, whereas the patches constituting the scalogram need 2D PE. To achieve this, Wang & Liu [179] extend equations 4.1 and 4.2 by changing the parameter pos by either the vertical or the horizontal position. This idea was used by us as a starting point to form:

$$PE_{(x,y,2i)} = \sin(x * 10^{-2i/d_{depth}}) * \sin(y * 10^{-2i/d_{depth}}) \tag{4.3}$$

$$PE_{(x,y,2i+1)} = \cos(x * 10^{-2i/d_{depth}}) * \cos(y * 10^{-2i/d_{depth}}) \tag{4.4}$$

Where x and y are the horizontal and vertical positions of the patch (with the origin being the top left corner of the image). A value of 10 instead of 10000 was chosen for the base of the exponential as it allows a greater range of cosine alignment distances in the case of a short and fix-sized sequence (here, 25 tokens).

The cosine alignment of each patch with regards to the rest of the image is represented on figure 4.5 for both trained (from Dosovitskiy *et al.* [2]) and fixed weights. On that image, similar quality of PE can be observed for both embeddings.

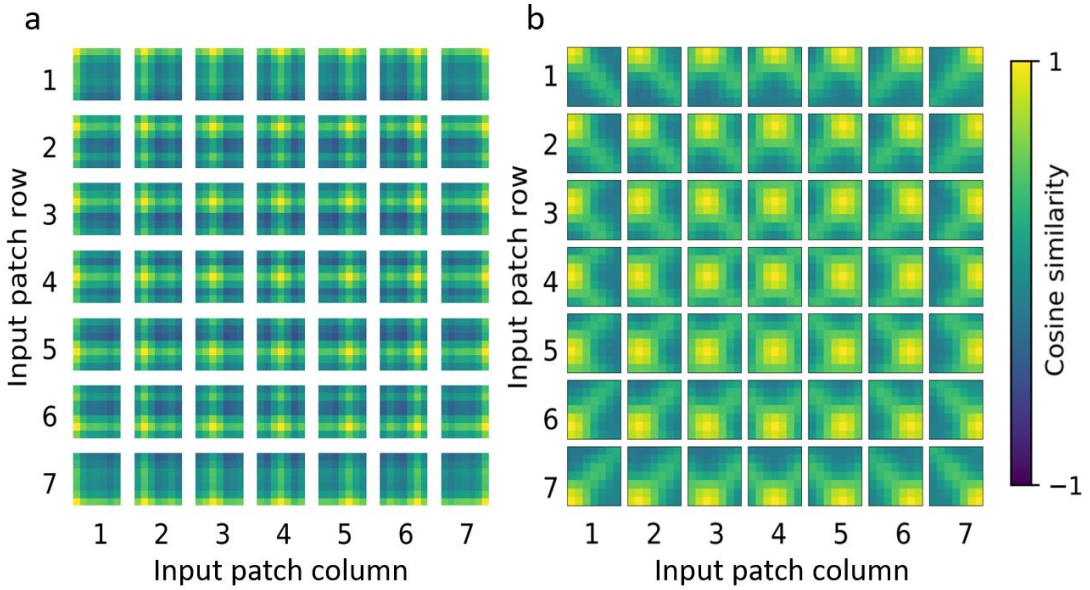

Figure 4.5: Cosine similarity between each patch and the rest of the image after PE. (a) Trained weights from Dosovitskiy *et al.* [2]. (b) Fixed weights obtained with equations 4.3 and 4.4. It is observed that the similarity between two patches increases when these are close to one another for both encodings.

Data storage The pre-processed data is stored in `.mat` files, with one sample per file. For the EO/EC dataset, the labels are stored in a single target file whereas for TUSZ, the data is stored in one of two different directories depending on its label.

To wrap this section up, table 4.2 compares the different parameters of the input pre-processing for the EO/EC and TUSZ datasets. The values of 8 and 40 for the embedding space sizes of the datasets ensue from a trade-off between the distortion ϵ and the PE on one side, which both benefit from a larger dimension, and the model complexity on the other side. Since the number of parameters in the encoders depends on the embedding depths, these have been fixed to the aforementioned values, knowing that the EO/EC model must be designed with less trainable weights, and that the task of the TUSZ model is more challenging.

Table 4.2: Pre-processing parameters for the EO/EC and the TUSZ datasets. The elements in bold are tunable parameters. The other values presented depend directly from them or are datasets properties.

Parameter name	EO/EC	TUSZ
Segment duration [s]	**0.2**	**5**
Sampling frequency [Hz]	1000	250
# samples per segment	200	1250
Scalogram size [px]	200x200	1250x1250
# patches	**25**	**25**
Patch size [px]	40x40	250x250
Embedding space size	**8**	**40**

4.3 Architecture and regularization

As stated in the previous section, the architecture chosen is the MViT from Hussein *et al.* [3] (figure 3.4, section 3.3), with the exception of the "Linear Projection" and the "Position embedding" layers that do not use trainable weights in this thesis (see previous section). A representation of it is shown on figure 4.6.

This architecture was selected arbitrarily over the other ones proposed in the state of the art (section 3.3), mostly because the design was intuitive without being overly simple. On the other hand, this choice is more parameter extensive than other approaches as the design requires one transformer encoder per EEG channel. The model therefore occupies more memory space and requires more training time under the hypothesis that each of its encoders is as bulky as the ones designed for the other models studied.

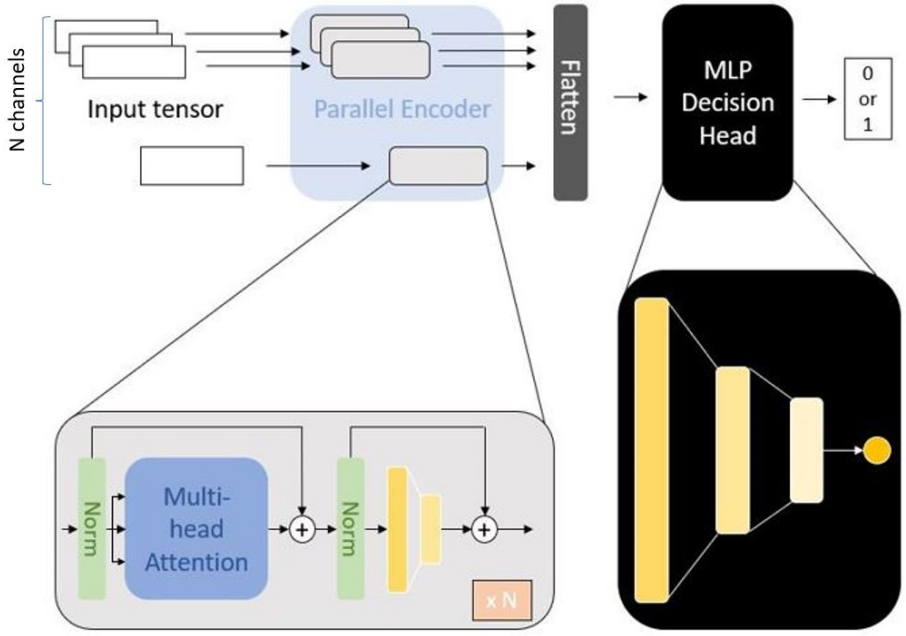

Figure 4.6: Architecture used in this thesis. This is similar to the MViT of Hussein *et al.* [3] with the first two layers removed.

From this architecture, two models types are built; one for each dataset. Since the task on EO/EC is used only to determine which pre-training is the most performant, the model tailored for this dataset is chosen to be much smaller than the one used for TUSZ, and the hyper-parameters are tuned to attain this objective. This allows a faster training and thus less time overall to repeat experiments in order to obtain significant results.

The hyper-parameters for the EO/EC and TUSZ models are resumed on table 4.3 along with the number of trainable weights. All of these are chosen with some level of arbitrarity, as it is very time consuming to build, train, and compare models to optimize all the variables.

Table 4.3: Hyper-parameters related to the model architecture and the number of weights this configuration yields.

Hyper-parameter	EO/EC	TUSZ
Input tensor shape	(32, 25, 8)	(20, 25, 40)
# encoders in parallel	32	20
# transformer layers in encoder	1	8
# attention heads per layer	2	4
Size of hidden and output layers (Transformer MLP)	[16, 8]	[80, 40]
Size of hidden layers (Decision head)	[128, 64]	[512, 256]
Weights	**EO/EC**	**TUSZ**
# trainable weights in parallel encoder	28 160	5 248 000
# trainable weights in decision head	827 665	10 372 177
# total trainable weights	855 825	15 620 177

Rather than the Rectifier Linear Unit (ReLU, figure 4.7, left) used in Vaswani *et al.* [1], a Gaussian Error Linear Unit (GELU, figure 4.7, right) was chosen as the activation function for the MLPs present in the encoders and in the decision head. This function is used in many transformer architectures such as BERT [98] and the ViT [2], mainly because it is non-linear for positive inputs − unlike ReLU − which leads to better performances overall [180]. The GELU is characterized by [180]:

$$\text{GELU}(x) = xP(X \leq x) \text{ with } X \sim \mathcal{N}(0, 1) \tag{4.5}$$

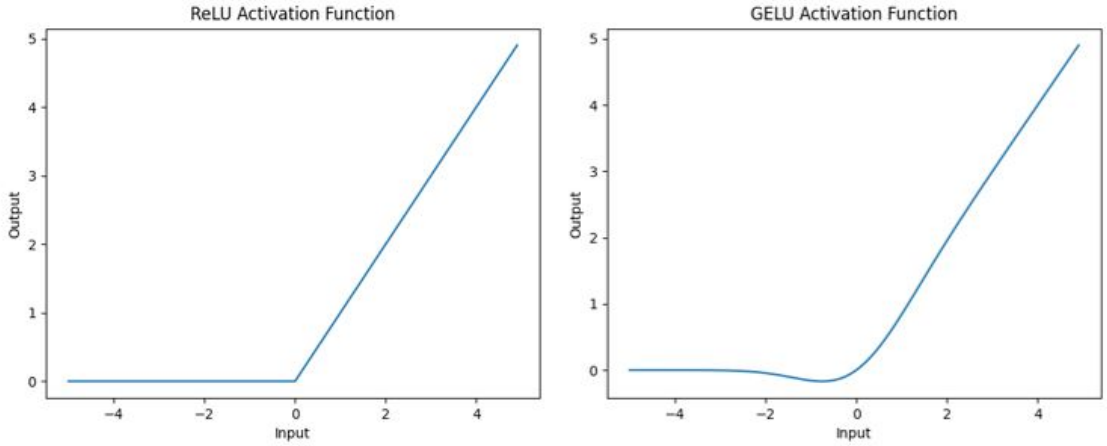

Figure 4.7: Representation of the ReLU (left) and the GELU (right)

Regularization

Overfitting expresses a too important match with the training data to a point where it hurts the accuracy on new data prediction (figure 4.8) [68, 181]. The generalization ability of a model to new data is believed to be an equilibrium between its complexity and the information of the training data [182, 183]. To avoid overfitting without modifying the neural network, more training data must be provided. If such data is not available anymore, regularization must be used.

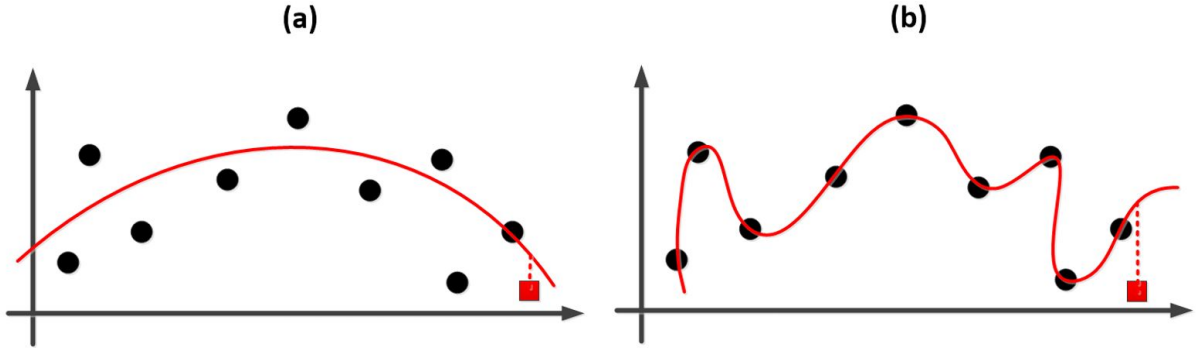

Figure 4.8: (A) Optimal model. (B) Overfitting model. The error with regards to the red dot representing a data to predict is much more important in (B). Adapted from [181].

As a first regularization technique, weight decay is implemented, which penalizes the high weights in the model by adding a squared weight-dependent term to the loss function (the latter being detailed in section 2.4):

$$\text{Loss}_{decay} = \text{Loss} + \lambda \sum_{n=1}^{N} w_n^2 \tag{4.6}$$

With w_n the n^{th} trainable weight in the model and λ the weight decay hyperparameter [182]. The value of λ has been set to 10^{-4} for both models after that trials with $\lambda = 10^{-2}$ and $\lambda = 10^{-3}$ led to underfitting with the TUSZ model.

The second regularization adopted is dropout (illustrated on figure 4.9), which consists in shutting down a fraction of the neurons in the model for one training iteration. This population of deactivated neurons changes after each epoch and no neuron is absent when the model is tested. This is a way to combine different models − i.e. different subset of the original ANN − which is known to improve performances [114, 184].

For both models, a dropout rate of 0.5 is selected for the MLP in the decision head and this rate is lowered to 0.1 for the MLP in the transformer encoders as well as in the Multi-Head Attention (MHA) layers. These values are the ones used in the TensorFlow[4] tutorial about the ViT [186].

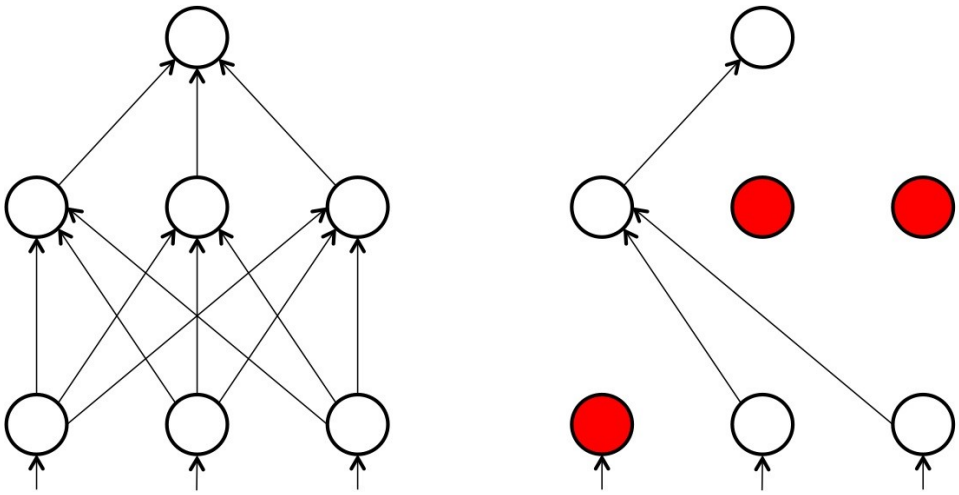

Figure 4.9: Representation of dropout regularization. The neurons have a certain probability to disappear for one iteration and doing so removes all connections to them [114].

Finally, early stopping is implemented for the TUSZ model[5]. This is a method where if a monitored value stops improving for a given number of epochs, the training is interrupted and the best weights according to that monitored value are retrieved [187, 188].

[4]TensorFlow is a `python` package allowing to build, train, and test deep learning neural networks [185]. It is the package that was exploited to build and regularize the model.

[5]It is not implemented for the EO/EC model as the behavior on a fixed number of epochs is required to select the best pre-training method.

In this case, the monitored value is the loss function on the validation set and the interruption occurs if no improvement is detected within 5 epochs after the last minimum value (i.e., the patience of the early stopping is set to 5). Such methodology might not always be the most adequate, as the validation error has a lot of variability in neural networks (figure 4.10) and early stopping might prevent from finding the global minimum.

Adaptive patience, such as proposed by Orr *et al.* [187], can be implemented to find a trade-off between an abrupt early stopping and none at all, and could be considered as a source of improvement for the model training, as it was not implemented in this thesis. Our simpler method however still leads to very good results (see the following chapter).

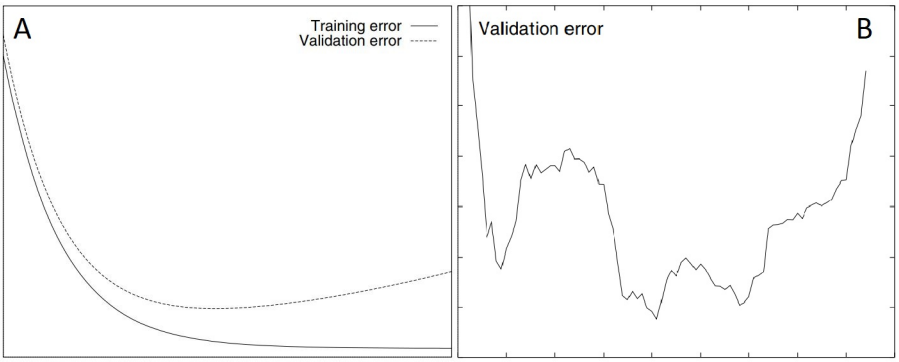

Figure 4.10: (A) Theoretical curves for the training error and validation error, with a minimum before overfitting. (B) Validation error behavior in a deep learning model. Adapted from [187].

To optimize the stochastic gradient descent (SGD, see section 2.4), the AdaM (for Adaptive Moment estimation) algorithm computes a different learning rate for each weight in the neural network instead of using the same value everywhere. To do so, the parameters of the model are adapted using an exponentially decaying sum of the previous values of the loss gradient and gradient squared, multiplied by a learning rate l_r.

On top of this, a bias correction term is included to correct the fact that the first and second moments of the gradient are initialized to zero [189]. The l_r hyper-parameter is set to 10^{-4} for all EO/EC model experiments, as well as for the TUSZ model. This choice was made by iteratively reducing l_r by a factor of 10 until finding an optimum in the test set accuracy, starting from 10^{-3}.

To implement the AdaM optimizer together with weight decay regularization, both must be decoupled. This is achieved by selecting a particular step in the AdaM algorithm where the penalty on the weight is applied [190]. This modified algorithm, called AdaMW, is the one used in this thesis.

4.4 Pre-training

Pre-training relies on transfer learning, which uses a model trained for one task as a starting point to solve a different one (see section 2.3). The principle is that the model keeps track of the features learned during its first training and does therefore not need to re-discover them. This translates to having a starting point on the loss function which is already close to a minimum instead of starting from a random configuration, thus converging in less epochs and leading to better performances overall [191].

The pre-training task proposed in this thesis is the binary classification of multi-channel signals between "EEG" and "non-EEG", following the intuition that the model could learn features inherent to electroencephalograms in the process. Such features are, for instance, the correlation of channels located close to one another, or the physiological frequency ranges.

To achieve this goal, three types of non-EEG multi-channel signals were implemented, all three of them using unmodified EEG signals as a basis (figure 4.11). The main advantage of such methodology is that unlabelled EEG can be harvested to constitute a pre-training dataset, therefore exploiting otherwise useless data. This strategy thus refers to Self Supervised Learning (SSL).

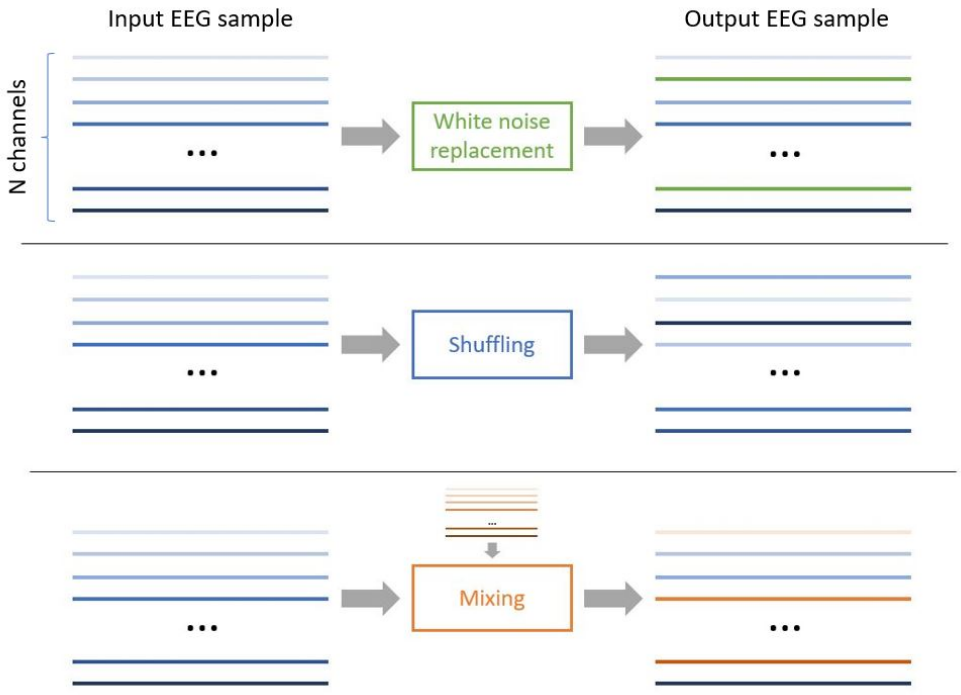

Figure 4.11: Illustration of how normal EEG signals are being modified for EEG recognition pre-training tasks.

White noise replacement In this first type of modification, n random channels of the original EEG are replaced with Gaussian White Noise (GWN). The number of replaced channels n is a random integer varying uniformly between 1 and hyperparameter N for each sample generated.

It is expected that a higher number of channels affected leads to more ease for the model at recognizing the "non-EEG" samples. N therefore controls the "difficulty" of the task (the closer N is to 1, the harder the classification becomes), and a value of 5 has been arbitrarily selected.

The intuition behind this modification is that, unlike with EEG signals where the Power Spectral Density PSD tends to decrease with the frequency (figure 4.12) [192], GWN has a constant PSD across all frequencies. The use of pink noise, where the PSD is inversely proportional to the frequency, instead of white noise could be considered for a more complex classification task, as this frequency behavior matches the one of the EEG better [193]. This has however not been further explored in this thesis.

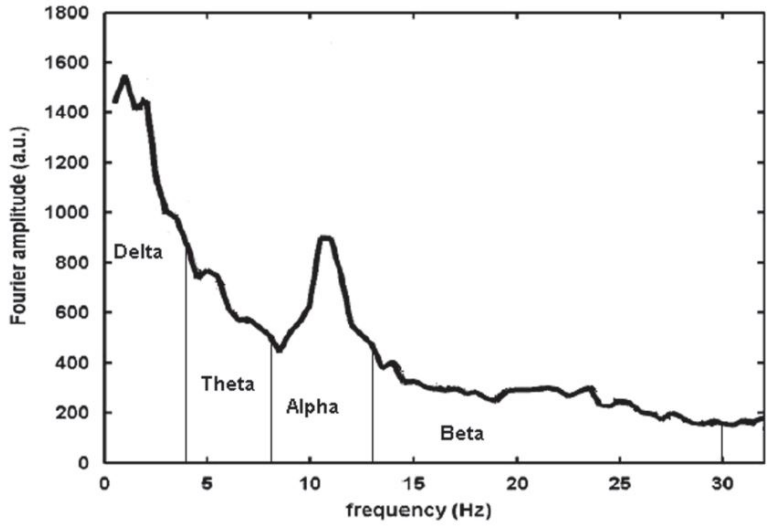

Figure 4.12: Average power spectrum of the EEG at rest. From [192], figure 7.

Channels shuffling In this modification, all the channels are conserved, but their position in the array are permuted. This operation amounts to shuffling the location of the different EEG probes for the model, and it therefore has to learn the inter-dependencies between the channels to classify the data as shuffled or not. Channels in EEG are correlated proportionally to the distance between them. That is, the further they are located from one another, the less correlated they are [194, 195].

Mixing of two EEG samples This third method takes two EEG samples as input and replaces $n \in [1, N]$ channels in each input by the same channels from the other input. The objective is to have channels that are not correlated to any other channel, although having the same behavior as normal EEG signals.

Like with channel shuffling, this classification task aims at teaching the channels correlation to the model. Once again, the hyperparameter N has arbitrarily been set to 5 (in this case, the further away N is from $n_{channels}/2 = 16$ for EO/EC and 10 for TUSZ, the more difficult the task becomes, as less uncorrelated channels are potentially identifiable).

The segments at the basis of the three "altered EEG" sample sets could not be used as real EEGs in the pre-training tasks, nor could they be run in the fine tuning of the model on the real task, as this would introduce biases. The original datasets therefore need to be separated into multiple subsets.

For both EO/EC and TUSZ, 70% of the available data points are stripped from their label to simulate a scarcity in labelled data. This unlabelled data is exploited for pre-training whereas the remaining 30% are used for model fine tuning. Of course, using the totality of the labelled data directly on the final task would have been much more efficient and lead to better performances overall, but the objective here is to show the impact of pre-training on a specific task, as well as how unlabelled data can still be exploited to improve a model. The data separation is interleaved after the "Split signals into segments" step of the pre-processing (retrievable on figure 4.1).

The unlabelled fractions of the datasets are further split in two; one half of the segments is not modified and directly transformed into scalograms, while the other half is duplicated twice, such that the three possible alterations are applied once on each segment. The three resulting sets of altered EEGs are then also sent down the rest of the pipeline (figure 4.13).

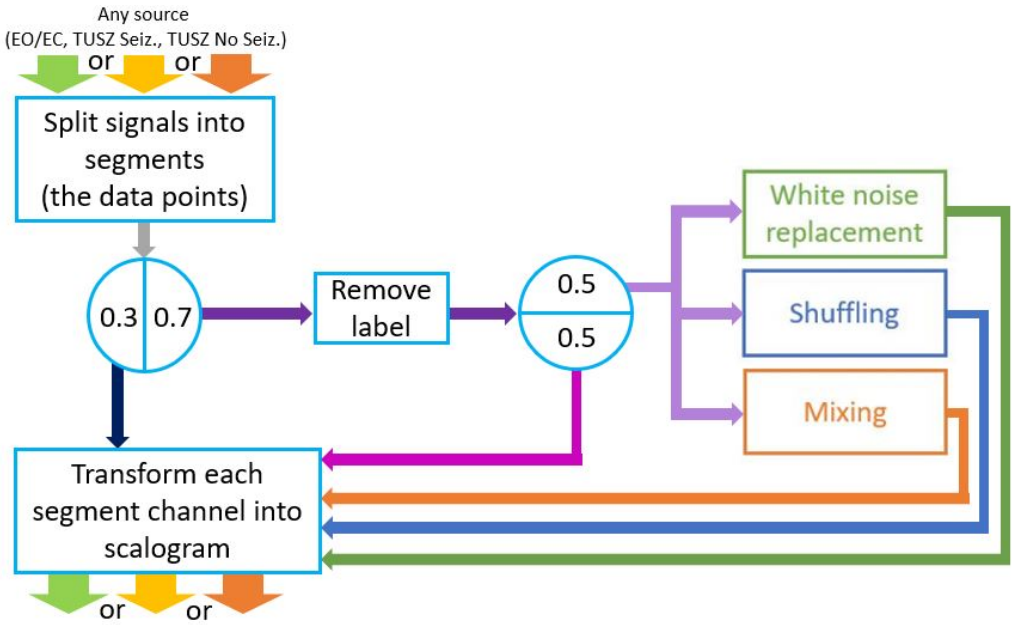

Figure 4.13: Modifications to the data pre-processing pipeline from figure 4.1 to add the division into pre-training and fine tuning data as well as the alterations brought to some of the pre-training signals.

To keep track of what fractions of the total dataset were used to what purposes, see table 4.4. One can notice that the fractions of the TUSZ are not exactly equal to the ones announced. This is because the data was split not at the segment level, but at the file level, and that the content of some files was longer than others, introducing some variations.

Table 4.4: Amount of data used for each subset in the dataset (portion of the total dataset in %)

Subset	EO/EC	TUSZ
Fine tuning	**665 (30.0)**	**18 106 (33.2)**
Label: positive	341 (15.4)	8 842 (16.2)
Label: negative	324 (14.6)	9 264 (17.0)
Pre-training	**1 551 (70.0)**	**36 483 (66.8)**
Modified	776 (35.0)	16 029 (29.4)
Unmodified	775 (35.0)	20 454 (37.4)
Total	**2 216 (100)**	**54 589 (100)**

4.5 Performances assessment metrics of the models

To assess a model performances, it must be tested on data that have not been encountered at any point in the training process. This is the test set, a random draw of $y\%$ of the initial data that will be left aside during the parameters fitting. The remaining $(100 - y)\%$ are the training set, later divided again between the actual training set and the validation set (see section 2.4) [114].

In this thesis, the test set always accounts for 10% of the total dataset regardless of whether we are working with the EO/EC or the TUSZ. The validation set represents 10% of the remaining 90%, or 9% of the total dataset. The remaining 81% of the data are used for training. The reason why such low shares of the initial dataset are selected for validation and test sets is because of the limited amount of data initially available. It is therefore preferred to maximize the training data under the constraint to keep a decent fraction for the model evaluation.

Once a classifier model is trained, each data from the test set is submitted to it and the predicted classes are recorded in the confusion matrix (table 4.5) [196].

Table 4.5: Confusion matrix for binary classifier [196].

Numbers of objects	Classified as positive	Classified as negative
Positive	A	B
Negative	C	D

In confusion matrices, the outputs are arranged along two axis; whether the model classified the data as positive (A, C on table 4.5) or negative (B, D), and whether the data was actually labelled positive (A, B) or negative (C, D). The elements on the diagonal are properly classified (called true positives in A, true negatives in D) and the other positions are missclasifications (false positives in C and false negative in B) [196, 197].

The metrics used to measure a classifier performances are issued from the confusion matrix. This thesis use accuracy and Area Under the ROC Curve (AUC) to benchmark the models. Accuracy measures the fraction of data categorized properly using the formula (A+D)/(A+B+C+D) [196]. It is important to note that a high accuracy is not necessarily a sign of a performant model, as it may be due to an imbalance between the data present in each category − called class imbalance.

For instance, if a dataset has 95% of its data labelled as class 1 and only 5% labelled as class 2, a model that classifies every data as being class 1 would have a 95% accuracy. This is why accuracy is generally computed with regards to a baseline model issued from the state of the art in the literature or a random classifier (which provides a random label to each points) [198].

The second metric requires the computation of auxiliary values. To measure what fraction of the positive labelled data are classified as positive, recall − also named true positive rate (TPR) − is computed using A/(A+B). On the opposite, the false positive rate (FPR), given by C/(C+D), shows what part of the "real" negative data are wrongly classified.

These two metrics are used to compute the Receiver Operating Characteristic (ROC) curve, as they are dependant from one another. The ROC curve (figure 4.14) is the plot of the TPR over the (FPR), which shows the trade-off between true positives and false negatives. It is built by changing the threshold at which a data is classified as positive by the model and report the repercussions on the TPR and FPR on the graph [197, 199].

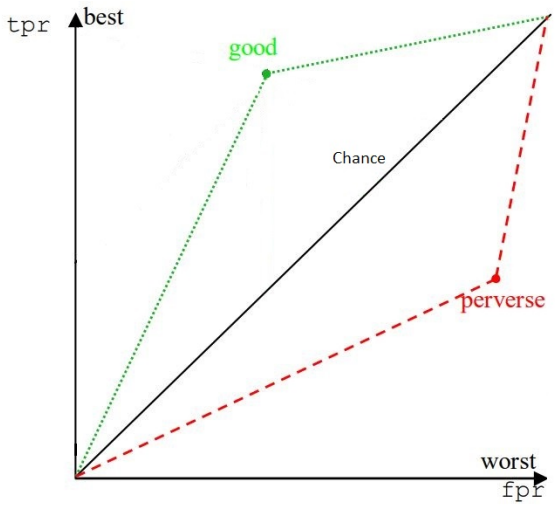

Figure 4.14: Plot showing the ROC curve of a random, a good and a perverse model [197].

On figure 4.14, it is observed that a random classifier has a ROC curve represented by the function FPR = TPR (chance line). To do better than random, a model must therefore have a curve drawn above the chance line. Another interesting element is that if the model trains on a dataset whose labels have been swapped (positive cases become negative and *vice-versa*), the resulting curve (perverse curve) will be the orthogonal symmetry of the unswapped model with regard to the 1:1 line [197, 199].

The AUC is determined by computing the area laying under the ROC. A value close to 1 means a very good model and an area around 1/2 is a model with similar performances as drawing labels at random [197, 199]. To avoid the construction of a ROC curve to compute the AUC, a simple three points trapezoid model can be used (the points being (0,0), (1,1) and the (FPR,TPR) working point of the model) and the area then simplifies to [197]:

$$AUC \approx \frac{TPR - FPR + 1}{2} \tag{4.7}$$

Chapter 5

Results and discussion

This chapter elaborates on the results of the conducted experiments aimed at assessing the performances of the model designed in the previous chapter (section 4.3), as well as the added value of the proposed pre trainings (section 4.4). The first section details how the best pre-training was selected for the epileptic seizure forecasting task, while the second section depicts the performances of the pre-trained model on said task and compares them to a similar model lacking pre-training.

Some supplementary graphs are available at annex B, along with their interpretations. Readers are invited to consult them after viewing this chapter, although their consultation is not compulsory for the proper understanding of the conclusions of this thesis.

5.1 Selection of the pre-training

To determine which data alteration among the ones discussed in section 4.4 would be the best to select as pre-training positive case, each of them has been exploited to pre-train a model on the EO/EC dataset. This smaller dataset was selected because it allows to repeat the experiments multiple times (here, 17) to assert the significance of the results using statistical tests without requiring too much total training time.

This choice comes at the cost of making a very strong hypothesis, which is that the conclusions drawn for the experiment on this dataset and this model size extend to other datasets and model sizes, as long as the task is about EEGs and the architecture is a MViT.

For each experiment, the same model − with the same initial weights − is pre-trained independently on each of the three datasets containing one different data alteration (see section 4.4) for 40 epochs before being fine-tuned on the eyes open/eyes closed (EO/EC) classification task for another 40 epochs[1]. A control model with no pre-training (i.e., only 40 epochs of fine tuning, starting with the same initial weights as the other models) is added to quantify the global effect of pre-training.

[1]The early stopping regularization (see section 4.3) was removed to observe the whole training process, but the weights at best validation loss were kept at the end of the pre-training

A hybrid pre-training consisting in 20 epochs of gradient descent on the dataset with the white noise alteration, followed by 20 epochs of gradient descent on the dataset with the shuffling alteration is also used to train one of the models. This type of pre-training is *a priori* believed to be able to teach more features to the model, as the white noise replacement modification is tailored to teach the frequential content of an EEG to the model, whereas the channel shuffling could teach the channels correlations.

The elements that vary between the experiments are the initial weights and the splitting of the dataset between training and validation sets at each epoch (which also implies a shuffling of the dataset). Four performances metrics are extracted at each experiment after that the model is fine-tuned. These are the epoch of convergence (i.e., the epoch of the minimum validation loss, EOC in short), the minimum validation loss, the validation accuracy at the EOC, and the validation Area Under the ROC Curve (AUC) at the EOC. A schematic recap of the pre-training performances assessment methodology is shown on figure 5.1.

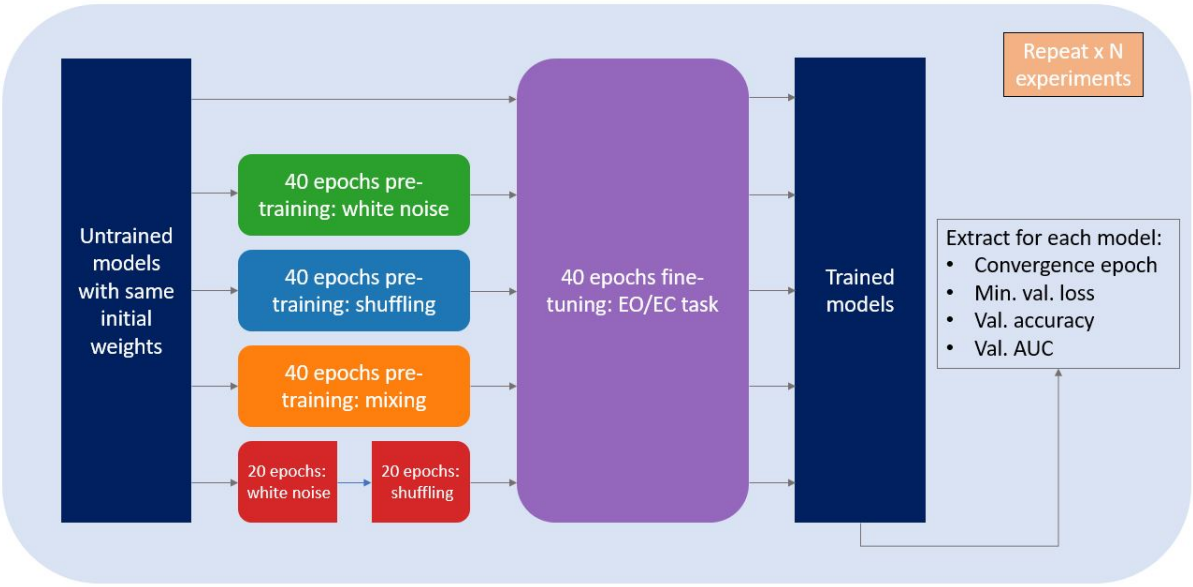

Figure 5.1: Pre-training performances assessment methodology on different pre-training datasets (named by there alteration to the normal EEG). In this thesis, $N = 17$.

A minimum validation loss to convergence epoch scatter plot with the theoretical distributions of the metrics along each axis is shown on figure 5.2, and the means and standard deviations of the performances of each pre-training, as well as their pooled values are shown on table 5.1.

To test whether the differences in performances are statistically significant, two-tailed Welch's t tests are conducted to compare each couple of pre-training with regards to each performance metric. This type of test was selected because the results of the experiments can be considered as independent and identically distributed (iid) random variables [200], and because the true variances of the samples are assumed to be different [201, 202].

Table 5.1: Mean performances for each type of pre-training on the EO/EC dataset and model, as well as pooled performances. Values in parentheses are the standard deviations. Best result per metric in bold, but without evidence of significance yet.

Pre-training	EOC	Min val. loss	Val. acc. [%]	Val. AUC
White noise	27.4 (9.58)	.497 (.053)	76.8 (4.92)	**.847 (.036)**
Shuffling	**17.0 (12.0)**	.500 (.029)	77.4 (2.74)	.839 (.026)
Mixing	31.4 (5.37)	**.496 (.040)**	77.8 (4.78)	.843 (.035)
Hybrid	17.2 (10.9)	.507 (.050)	**78.3 (4.35)**	.828 (.032)
Pooled	23.2 (11.6)	.500 (.044)	77.6 (4.32)	.839 (.033)
No pre-training	36.1 (4.27)	.520 (.025)	75.7 (4.36)	.825 (.025)

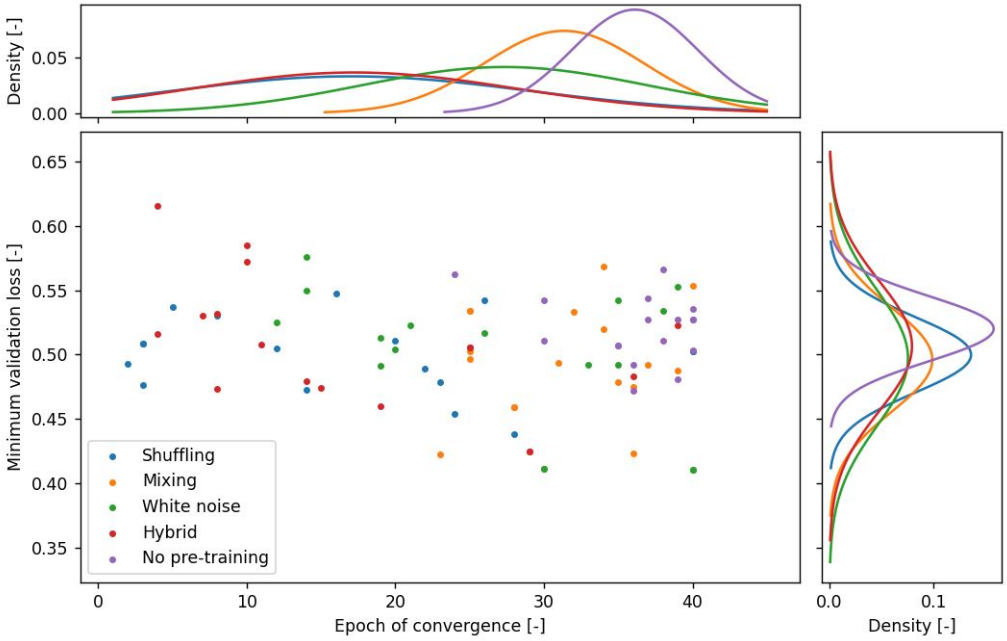

Figure 5.2: Scatter plot of the minimum validation loss against EOC for each of the pre-training methods. Smaller plots show the theoretical probability density function of each axis, according to the central limit theorem (since the results are iid) [202].

When comparing the pooled pre-training performances with the ones of the model without pre-training, it appears that there are no significant improvement with pre-training in general with regards to validation accuracy and validation AUC at the EOC (p-values equal to 0.1419 and 0.0661 respectively, which is greater than the significance threshold at 0.05). However, the difference is highly significant with regards to the EOC (p-value $< 10^{-5}$) and significant with regards to the minimum validation loss (p-value = 0.0197).

When it comes to the performances of each type of pre-training in particular, only the shuffling alteration manages to show a significant improvement with regards to no pre-training at all when comparing the minimum validation loss (p-value = 0.0441), even though the statistical tests show a trend for the other types of alterations (p-value

= 0.0530 for the channels mixing and p-value = 0.1329 for the white noise replacement). The hybrid pre-training has the least significant difference for this metric, with a p-value equal to 0.3474.

If the epochs of convergence are compared, both the shuffling alteration and the hybrid pre-training show a highly significant improvement with regards to no pre training at all (p-value $< 10^{-5}$ for both), but also with regards to channel mixing (p-values of 0.0024 for shuffling and 0.0011 for hybrid) as well as to white noise replacement (p-values of 0.0132 for shuffling and 0.0088 for hybrid). Nevertheless, as seen on table 5.1 and on figure 5.2, the standard deviation of the EOC for these two pre-trainings is much higher than for the other alterations.

As a last step, a linear regression is fit through all the data to verify if there is any correlation between how early the model converges on fine tuning and the minimum validation accuracy at that convergence epoch. If this correlation exists, this would mean that there would be a trade-off between convergence speed and loss and that both metrics could not be minimised at the same time.

Fortunately, the results of this test (table 5.2) show that only a small amount of the variance on the minimum validation loss is explained the EOC, as the highest coefficient of determination (R^2) among all the different pre-trainings is just above 0.25 [203–205]. Furthermore, even if the R^2-score was higher, some of the pre-trainings would not suffer from early convergence as the slope of their linear regression is negligible (like with the shuffling alteration), or even positive (with the mixing alteration). It is therefore possible to conclude that earlier convergence in pre-training is not achieved at the detriment of validation loss.

Table 5.2: Slopes and R^2 scores of the linear regressions fitting the EOC-minimum validation loss relation. The pre-trainings are ranked from the lowest to the highest R^2.

Pre-training	Slope [loss increase/epoch]	R^2
Mixing	$1.049 * 10^{-3}$	0.0194
Shuffling	$-0.488 * 10^{-3}$	0.0399
No pre-training	$-1.637 * 10^{-3}$	0.0763
White noise	$-2.459 * 10^{-3}$	0.1985
Hybrid	$-2.351 * 10^{-3}$	0.2581

In light of the results previously gathered, the datasets with the shuffling alteration seems to be the best to use as pre-training, since it significantly improves both the EOC and the minimum validation loss. A dataset using this alteration will therefore be generated and used as pre-training for the seizure prediction task on the TUSZ dataset.

The hybrid pre-training, although also significantly reducing the EOC, does not show any significant improvement of the minimal validation loss function, which makes it the second best one to use. It is possible that other methods of hybrid pre-training, such as applying multiple alterations at once on some signals of the dataset, or mixing multiples

datasets having each one alteration, could yield better results. These new strategies are however not covered in this thesis and are left for future researches.

On top of the strong hypothesis stating that the results obtained with a single dataset and model size can be extrapolated to any other dataset and model sizes (as long as the dataset is with EEG signals and the model architecture is a MViT), another limitation may mitigate the conclusions of this section. This limitation is that the number of experiments achieved is rather low, as the realization time for one experiment (i.e. the time required to pre-train and fine-tune all models once) is, on average, 56 minutes. This is an issue because the Welch's t test relies on the central limit theorem, which assumes a large enough sample size [201, 202].

Although "large enough" is a vague criterion, the literature states that a population of 30 can be considered as large, even though a sample size higher than 8 is already sufficient [206–208]. Increasing the number of experiments to 30 may thus provide more confidence in the statistical tests.

5.2 Performances of the model

This section asserts the performances of the architecture and strategies used for a pre-ictal/inter-ictal classification (or seizure forecasting) task on the TUSZ dataset. Based on the results of the previous section, the model is first pre-trained on a EEG/non-EEG task using the shuffling alteration on some of the TUSZ data with removed label before being fine tuned on the seizure prediction task.

As a way to further test the usefulness of pre-training, another model is trained on the same fine-tuning set, but starting with random weights (i.e. without pre-training phase). All models are trained until the minimum validation loss remains the same for 5 epochs and the weights from that optimal epoch are selected as the best weights (see early stopping in section 4.3).

As with the experiments on the EO/EC dataset, the performances are monitored using a validation set, which is changing at each training epoch. On top of this, a 10% test set is also used to harvest metrics once the model is trained. The latter are − as in the previous section − the validation loss, the validation accuracy, and the validation AUC (respectively, test loss, test accuracy and test AUC for the test set).

When comparing the time necessary for the models to converge to a local validation loss minimum (table 5.3), it clearly appears that pre-training allows for a much faster fine tuning, as the number of epochs required to find a minimum is divided by two when the only element changing is whether the starting weights are random or issued from pre-training (going down from 8 epochs to 4).

It might be argued that the pre-training itself is very time-consuming, as it requires training a model on a big amount of data (which is why the average training time/epoch

in table 5.3 is much higher than in the fine-tuning phase), but its advantage is that one pre-trained model can be used as a starting point for many different tasks in the same area, curbing their fine-tuning time.

Table 5.3: Compilation time of the different training phases on a single Intel Core i5-7300HQ CPU working at 2.5 GHz with no GPU set up. The values are the ones necessary to reach the minimum in the validation loss (even though the training continue for 5 more epochs afterwards before the early stopping is triggered).

Training phase	Time/epoch (avg.) [s]	EOC	Tot. train. time [h]
Pre-training	11 117	9	27.8
Fine-tuning (with pre-train.)	5 288	4	5.88
Fine-tuning (no pre-train.)	5 698	8	12.66

Regarding the performances of the pre-trained model against the non pre-trained model (figure 5.3 and table 5.4), results clearly show an improvement on all metrics when using the model that has been pre-trained before fine-tuning.

In particular, it allows the test loss to decrease from .1939 when no pre-training is done to .1748 when starting with pre-trained weights, which has a beneficial effect on the test accuracy and AUC. Interestingly enough, this effect was not demonstrated on the model trained on the EO/EC dataset in the previous section, which is hypothetically due to the lack of data. The literature, however, concurs with the fact that pre-training a model improves its performances [209, 210].

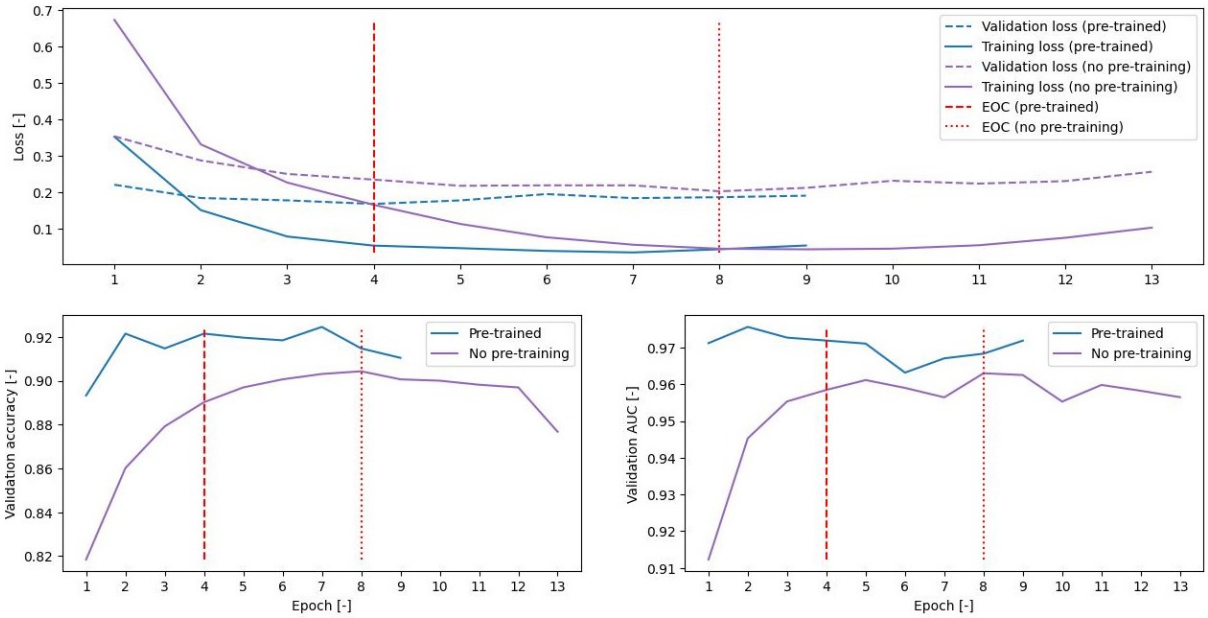

Figure 5.3: Comparison of the evolutions of the performance metrics of the pre-trained and non pre-trained models with regards to the training epoch.

Table 5.4: Comparison of the model performances after fine-tuning depending on whether it was pre-trained or not.

Metric	With pre-training	Without pre-training
Val. loss at EOC	.1689	.2035
Val. acc. at EOC [%]	92.15	90.43
Val. AUC at EOC	.9719	.9630
Test loss	.1748	.1939
Test acc. [%]	92.16	90.93
Test AUC	.9702	.9648

The above results suggest that the pre-training datasets designed in this thesis not only allow a faster convergence of the models during the training on the real task, but also lead to models with better classification performances than models that were not pre-trained. Further researches should focus on the comparison between these datasets and the pre-trainings already existing in the state of the art (such as, for instance, the one proposed by Kostas *et al.* [160], summarized in section 3.3).

Unfortunately, since the TUSZ dataset was, to our knowledge, never used before for seizure prediction (i.e. pre-ictal/inter-ictal segment classification)[2], the performances of the model trained on this dataset cannot be properly compared to the ones of other models. This is because the selection of the dataset itself plays a major role on the resulting performances, due to either its size or characteristics (e.g., the sampling frequency, electrodes placement and number, or conditions of recording in the case of EEG) [211, 212].

To properly compare this thesis model to another transformer model, it is therefore required to train both models on the same dataset, while also using the pre-processing pipelines used with the models. This comparison was not realized in this thesis as determining the added value of the designed pre-trainings was considered as more interesting than bench-marking the architecture that is very similar to the one of Hussein *et al.* [3].

[2]This dataset is initially designed for seizure type classification and is used as such in the literature.

Chapter 6

Perspectives and potential improvements of the model

This final chapter considers some of the model limitations raised throughout the second part of this thesis and proposes solutions that could be implemented in future works. It also introduces suggestions for further experimentation and bench-marking on the model architecture and pre-training.

6.1 Improvement of the pre-processing

As discussed in the paragraph about the conversion of EEG data into scalograms in section 4.2, the current pre-processing pipeline suffers from the fact that the algorithm exploiting the CWT (see section 3.2 for a reminder on this transformation) is designed to output squared scalograms with a side length equal to the size of the input array. This makes the whole pre-processing dependent on the input dimension, both in terms of execution time (as the CWT is the most expensive step of the pipeline) and of required memory.

Currently, the memory requirements are dampened by randomly projecting the output image into a lower dimensional space. According to the Johnson-Lindenstrauss lemma (section 4.2), this distorts the distances between each point of the dataset by a factor bounded by $[1 - \epsilon, 1 + \epsilon]$, where ϵ depends, inter alia, on the size of the dataset [176, 177]. With that kind compression, the pre-processing pipeline suffers once again from scalability issues, as a higher volume of data worsens the distortion.

To resolve both previously mentioned problems at once, the scalogram has to become invariable in terms of the input length. This is achievable by anchoring the number of values that the location parameter b of the CWT may take, although still covering the same range.

In other words, instead of a unit step size that is implement at the moment, the parameter b could be replaced by n equidistant values ranging from the start to the end of the input signal. Since the parameter a is already designed to take as many values as the parameter b, this modification results in scalograms with a n x n size, with n controllable

as an hyper-parameter. The behaviors of the old and the new parameter b are reported on figure 6.1.

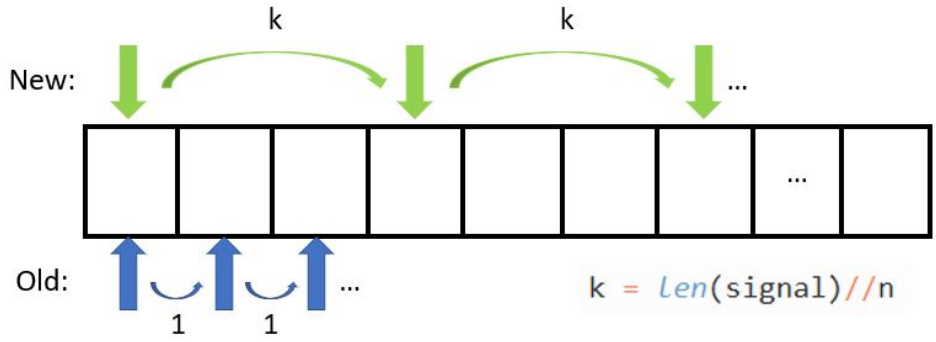

Figure 6.1: Movement of the old and new b parameters along the signal for the transformation into scalograms. The new method allows to set the output resolution to any desired value.

The redesign of the CWT step takes away the need for any compression method, as the pre-processed data can be rendered as compact as desired by tweaking n. This also allows to reinstate the abandoned learnable linear embedding layer to the model, as the random embedding becomes unnecessary. Such learnable layer is preferred because, although adding parameters to the model (the number of added weights being equal to $n_{channels} \times n_{patches} \times n \times m$, with m the layer output dimension), it produces a better embedding via the adjustments made by the backpropagation algorithm.

If we desire to allocate approximately the same memory for the output of this alternative pre-processing pipeline as we did for its predecessor, the hyper-parameter n has to be set to 15 for the EO/EC dataset and to 35 for the TUSZ dataset. This is because without random compression, the length of the third axis of the pre-processing output tensor (equal to $n_{emebedding}$ in the current version of this model) becomes equal to the length of a flattened scalogram patch.

With the values of this third axis being currently equal to 8 and 40 for respectively the EO/EC and the TUSZ dataset, it is simply needed to choose a close perfect square value (i.e., 9 and 49), and to multiply it by the number of patches to obtain the number of pixels present in the full scalogram. Taking the square root of this result provides the side length of the image, which is equal to n.

A comparison between the scalograms with resolutions of 400x400 (i.e., the current scalogram dimensions for the EO/EC pipeline), 35x35 ($n = 35$), and 15x15 ($n = 15$) pixels is provided on figure 6.2. On that figure, it is possible to remark that the main features of the image are still distinguishable when the resolution is lower, making $n = 35$ a good candidate for this hyper-parameter.

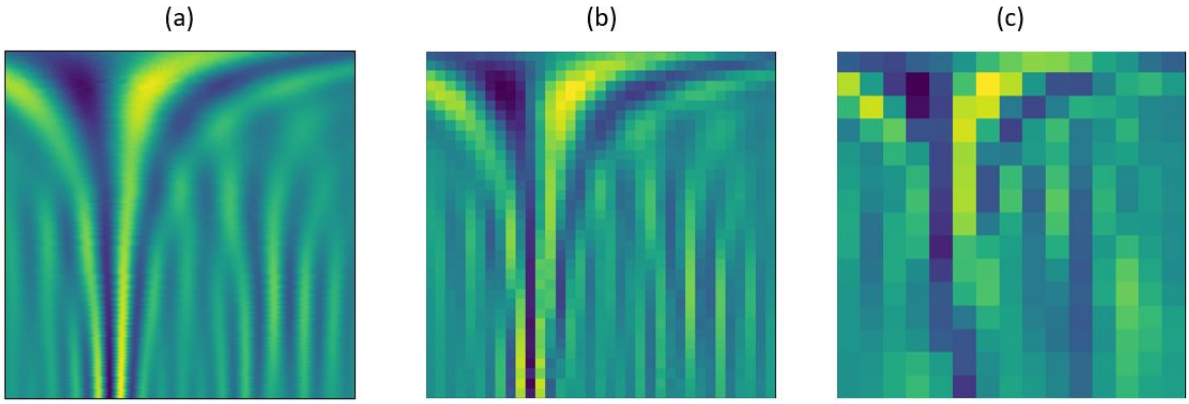

Figure 6.2: Comparison of the scalogram resolutions. (a) current scalogram for the EO/EC dataset (b) scalogram when $n = 35$. (c) scalogram when $n = 15$.

Once the proposed improvement is incorporated to the pre-processing pipeline, the main experiments to realize to assess the effect of the added modification must target two elements in particular. These are the pre-processing time, which is expected to curb significantly, and the model performances, which are expected to raise with regards to their current values.

6.2 Working with other datasets

A limitation intrinsic to the MViT (see section 3.3) is that each of the model implementing this architecture is designed for EEG signals similar in channel number and locations. Although there are some guidelines, such as the International 10-20 system (see section 1.2), which try to uniformize the number and the placement of the EEG electrodes, a great disparity in terms of these two parameters is observed throughout the different datasets available. It is therefore necessary to find a way to prevent their variation from generating incompatibilities between the data and the model.

The MViT is set up for a precise number of channels, and trained such that each branch of its architecture can process a the output of a probe located at a rather precise spot on the scalp. One solution could thus be to map the channels configuration of the dataset the model was build for to the configuration of any new dataset that requires compatibility for training or testing. This process potentially includes both the suppression and the interpolation of some probes to fit the expected placement. Fortunately, some EEG processing `python` packages, such as `DN3` [213] and `MNE-Python` [214] include a feature for bad channel interpolation that could be repurposed to this aim.

A corollary to the action of reformatting the initial dataset configuration through channel mapping is the possibility to experiment on the variation of the model performances with regards to the number of channels. As a matter of fact, there exists a correlation between this number and both the model size in memory and required training time. This is because the quantity of trainable parameters in the model increases with the number of transformer encoders required to process all the channels (see the discussion in section

4.3). Henceforth, the finding of a "threshold" number of channels, ensuring both a proper model quality and a decent quantity of parameters, would be valuable.

6.3 Further assessment of the pre-training performances

Although this thesis thoroughly demonstrate the added value of the developed pre-training datasets with regards to their absence, the designed methodology still needs to be compared to an existing one. A candidate for this would be the BERT-inspired pre-training adapted to EEGs by Kostas *et al.* [160] and described in section 3.3.

The experiment assessing the performances of both pre-trainings would follow the protocol described now. Three identical models would be built with the objective to perform at least two different EEG classification tasks and training on at least two different datasets. For each task, two of the three models would be pre-trained with the two compared methods, using the same amount of unlabelled data and starting from the same random weights. The third model is kept as control and is not pre-trained.

The three models are then all fine-tuned with the same training set on the downstream task and their performances are evaluated using the same test set. These are reported on a table and the eventual improvements of the candidates with regards to the control model are compared. If one pre-training systematically yields better results across the different tasks, it might therefore be considered as more efficient than the other. For completeness, this experiment should be duplicated with another architecture to ensure that the results are not model specific.

Conclusion

The objective of this thesis was to explain the principle of transformers neural network and to give an overview of the different architectures used for EEG classification, as well as to add a personal contribution to the field with the design and testing of pre-training datasets from otherwise useless unlabelled data.

The browsing of the literature and the comparison of the different architectures yielded similarities both in the pre-processing pipelines (that are sometimes even shared with other ML designs) as well as in the three-stage paradigm inferred from the study of the models. This paradigm hypothesizes that every transformer-based design for EEG classification is composed of three levels, which are a primary embedding of the multidimensional data, followed by a composition of one or more encoder blocks in series or in parallel, and ended by a decision head being almost always a fully connected linear layer or a MLP.

The pre-training proposed in the second half of this thesis consists in the classification of multichannel signals depending on whether they have the properties of an EEG or not. The datasets generated to this end make use of unlabeled EEG data that may be transformed by different means. The three alterations proposed in this context, when incorporated into the pre-training of a classification model, show a significant difference in terms of the earliness of the convergence to the best weights compared to a model without pre-training.

On top of this, using one of these alterations significantly improves the minimum validation loss. On a larger dataset, this same pre-training methodology leads to a training time reduced by half on the downstream task as well as an improvement on the model loss, accuracy, and AUC on a test set when compared to a similar model that did not benefit from our method.

Further studies should be directed towards duplicating the experiment done on the larger dataset to confirm the significance of the improvement, but also towards benchmarking our method against other pre-trainings from the state of the art. The proposed strategy however already has the advantage to be more intuitive than the existing SSL as it makes use of the intrinsic properties of the EEG instead of adapting a methodology imported from other machine learning fields.

Appendix A

Explanations of the GitHub repository

All the code developed in the frame of this thesis is available at the following address: `https://github.com/tbary/MasterThesis`. The present appendix briefly explains how to navigate through the repository and what are the purposes of the different `python` scripts located in the `src` directory.

A.1 Structure of the repository

Besides the `README.md` and the `LICENSE` files, the main branch of the repository is composed of four directories. The content of these is to be visualized on figure A.1. The `src` directory contains the `python` scripts that pre-process the data, build and train the models, and record their performances. It also holds one package `library.py`, that encloses the different classes and functions useful to the program, as well as an `init.py` file, which sets up the global variables and imports the `library`. This second file is imported by all the scripts.

The `data` directory stores the `.csv` files that constitute the EO/EC dataset[1]. On top of this, it also contains five more folders. Each of these receives a part the pre-processed data from the EO/EC dataset. `scalograms_small` has the fine tuning set, `scalograms_pt_untouched` has the unaltered part of the pre-training set, and `scalograms _intra`, `_inter`, and `_noise` respectively store the data with the swapping, shuffling, and white noise alterations. The files containing the data are `.mat` files; one per data point.

The `results` folder gathers the recorded performances from all the trained models. For instance, the comparison of the different pre-trainings via their performance metrics is stored in the `pre_trainings_performances.pickle` file. The performances of the bigger models trained on the TUSZ dataset are located in files with the following structure: `raw_performances_XXX.pickle`, where `XXX` is replaced by an identification of the model.

[1]These files are actually not present on the public repository because we were uncertain whether this would represent a violation of the intellectual property of UCLouvain.

Finally, the `tmp` folder contains the weights of the models trained on the TUSZ dataset. The five saved weights sets are the ones of the pre-trained model, the ones of the fine tuned models with learning rates of 10^{-4} and 10^{-3}, and the ones of the models without pre-training using the same two learning rates. These weights can be loaded into the MViT, either to make seizure forecasting, or to use them as pre-trained parameters for downstream tasks.

One can notice that the TUSZ dataset is absent from this repository. This is because this dataset, even after pre-processing, cannot be shared to a third party. To fully test the code produced for this thesis, the reader can fill the form located at `https://isip.piconepress.com/projects/tuh_eeg/html/downloads.shtml` to retrieve a copy of the dataset. This copy does not need to be incorporated to the repository once obtained, as the `src/init.py` file contains a global variable to fill with the path to the dataset.

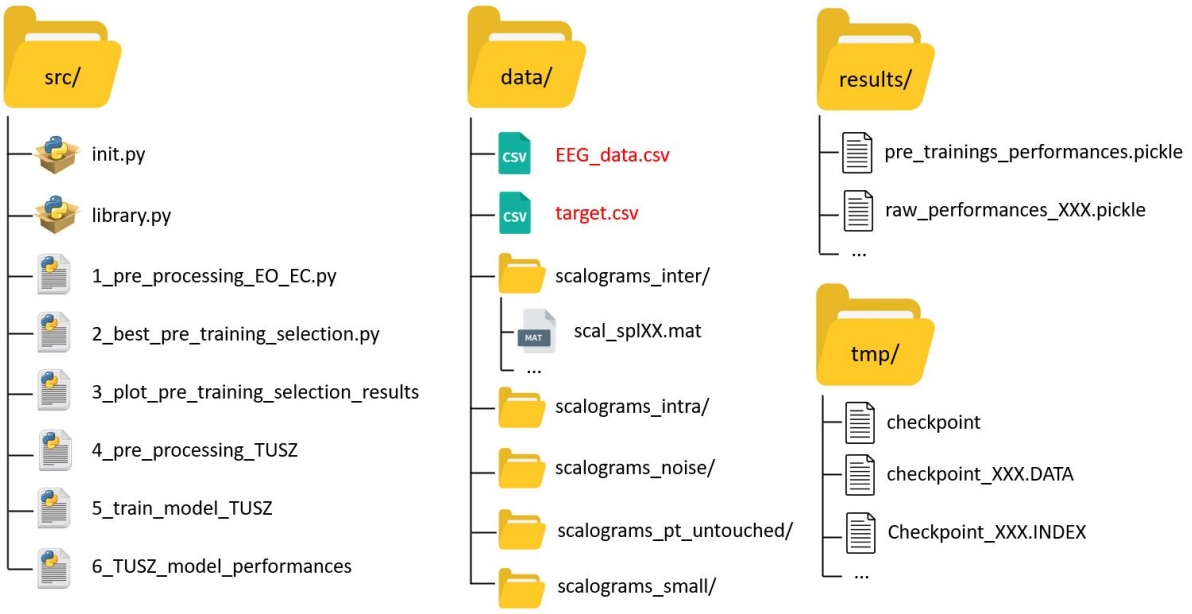

Figure A.1: Representation of the content of each of the main directories on the GitHub repository. Files in red are absent from the public repository.

A.2 Executing the code

As mentioned in the previous section, the `src` folder contains six `python` scripts, one `init` file and one `library` package. The scripts are numbered according to in what order they should be executed. The first three scripts act on the EO/EC dataset and allow to pre-process the data, to train some light models, and to compare the performances of the different pre-trainings. Scripts four to six have the same effects on the TUSZ dataset, with the difference that the trained models are scarcer but bigger.

Each of the six scripts reads files as input and either write some new ones or outputs graphs and string to report performances. These input/output relationships are repre-

sented on figure A.2. For reasons stated in the previous section, the raw data from the EO/EC dataset, as well as the raw and pre-processed data of the TUSZ dataset are not accessible on this repository and some of the programs can therefore not be executed. Scripts 2, 3, and 6 can nonetheless be run, which allows to verify the performances of the pre-training datasets and the big models. The classes and functions from `library.py` can also serve as a basis for further experimentation on transformers for EEG classification.

The execution of each `python` file in order yields the following results:

`1_pre_processing_EO_EC.py` Reads the EO/EC dataset contained in `EEG_data.csv` and `target.csv`. Pre-processes the data, splits the dataset into pre-training and fine-tuning, and applies the alterations on the pre-training data that requires it. Writes the embedded scalogram of each data points in a `.mat` file in the folder corresponding to its dataset (fine-tuning, pre-training...)

`2_best_pre_training_selection.py` Pre-trains small MViT models on the different pre-training datasets. Fine-tunes them all on the same dataset. Saves the performances of the models during fine-tuning in the `results` folder. Repeats this procedure multiple times to account for variability.

`3_plot_pre_training_selection_results.py` Reads the performances of the fine tuned models and extracts statistics out of it. Plots the best performances of each pre-training and performs significance testing to highlight whether one of the sets leads to better results or not.

`4_pre_processing_TUSZ.py` Reads the TUSZ dataset and re-labels the data between pre-ictal and inter-ictal. Pre-processes the data, splits the dataset into pre-training and fine-tuning and applies the shuffling alteration (as it is determined to yield the best performances). Saves `.mat` files of the embedded scalograms in the appropriate folders.

`5_train_model_TUSZ.py` Trains one model on one of the available pre-processed datasets, eventually starting from pre-trained weights. Saves the weights in the `tmp` folder and the model performances evolution during training in the `results` folder. To obtain all the desired models, it is necessary to re-execute this script with different input datasets and weights.

`6_TUSZ_model_performances` Plots the performances of the model during pre training. Then, compares the performances of the fine-tuned model with the ones of the model that did no go through pre-training.

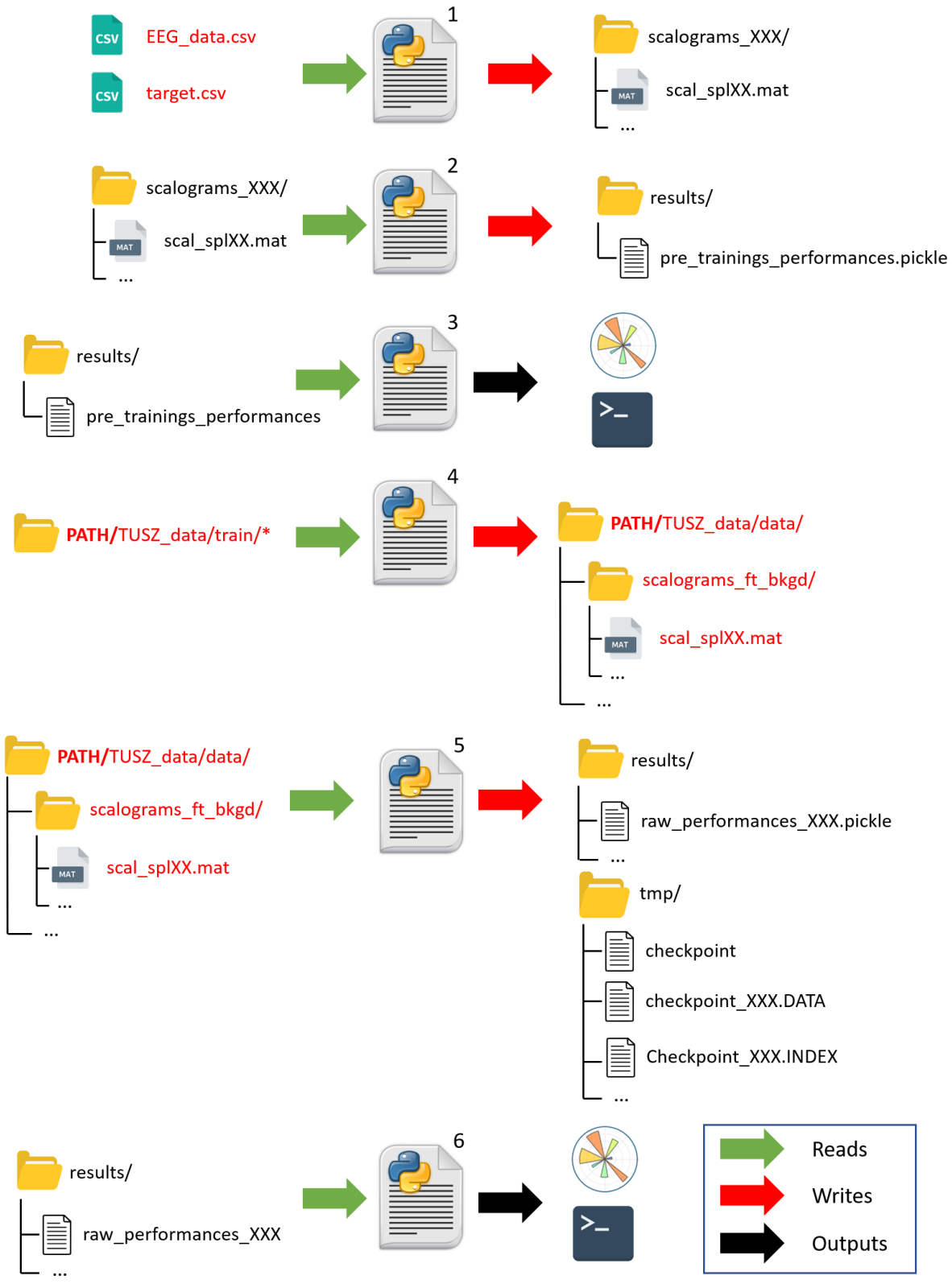

Figure A.2: Inputs and outputs of the different `python` scripts.

Appendix B

Supplementary graphs

This annex contains some graphs that were produced during the development of this thesis, but that were not included in the final text. This is either because their content is not pertinent with regards to the rest of the work, or because they illustrate results that are already displayed using other figures. These graphs are nevertheless presented here for the sake of completeness and transparency and are accompanied by some observations.

B.1 Complements to section 5.1

Figure B.1 shows how the models pre-trained on the different datasets improve during their fine tuning by displaying their scores in training loss, validation, accuracy and AUC after each epoch. On these graphs, we can notice that the performance curves separate in a spindle, never crossing one another until they reach their respective epoch of convergence (EOC, see section 5.1).

An analysis of the position of each curve in the spindles shows that the shuffle and hybrid datasets yield models with better performances than the other datasets for a same number of fine tuning epochs. This behavior is observed as long as their EOC is not reached, which happens earlier than with the other pre-trainings. Inversely, the model that did not go through pre-training always has the poorest performances.

These elements lead to the same conclusions as the ones drawn in the section about pre-training selection (i.e., section 5.1). This also shows that even if the fine tuning is stopped at a sub-optimal epoch (i.e., before the EOC), the pre-trainings that yield the best model performances remain the same. The reason why figure 5.2 was kept in the main text instead of B.1 is because the latter does not clearly indicate whether the performances reached at EOC by some of the models are better than the others.

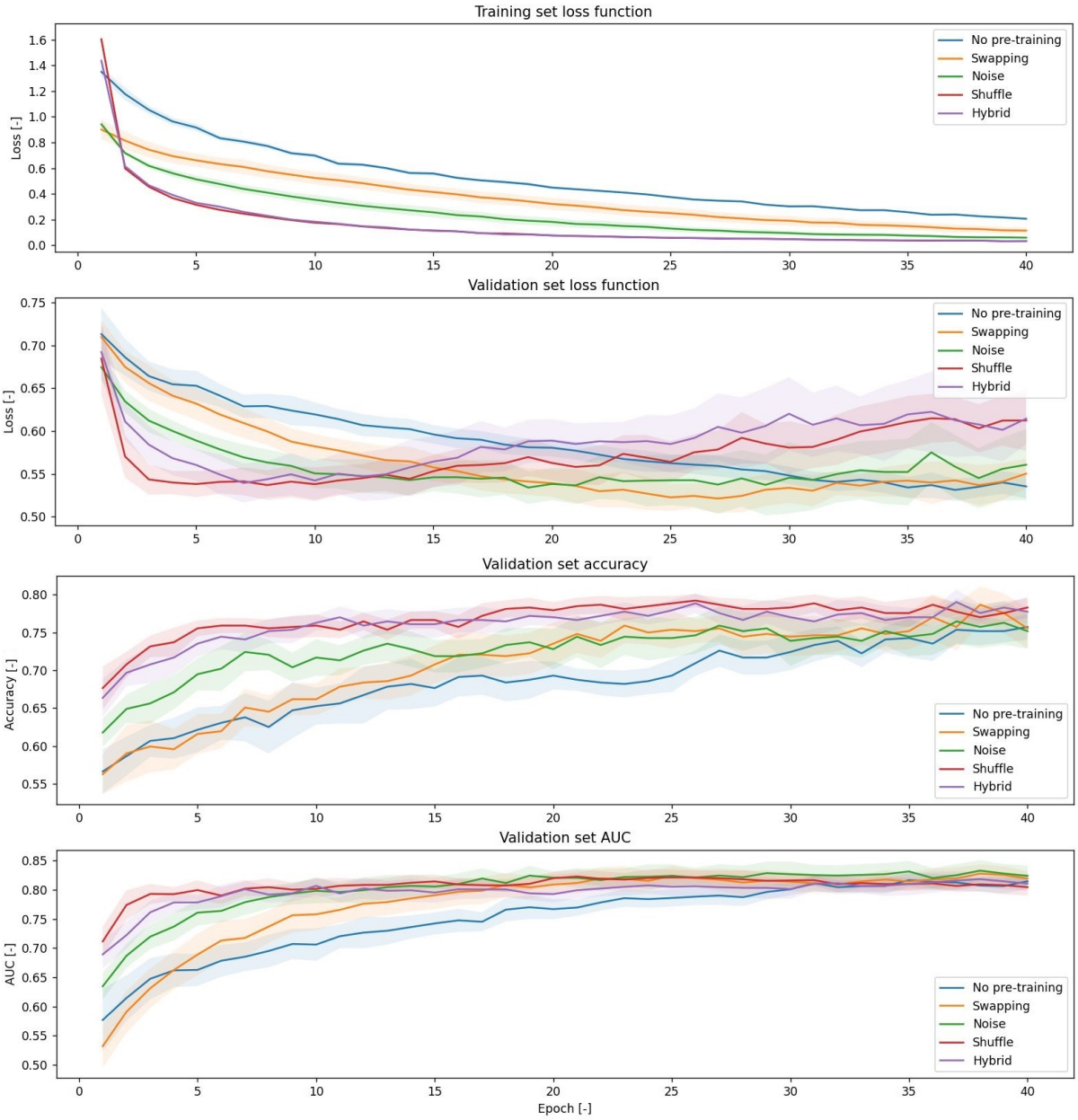

Figure B.1: Average evolution of the performances of the models during fine tuning on the EO/EC dataset. The shaded area corresponds to the 95% confidence interval. Each model has been pre-trained on a different dataset (or none at all).

Figure B.2 takes the same cloud of points as figure 5.2, but plots the linear interpolations of the EOC against minimum validation loss relationship for each of the models represented. This figure is not shown in the main work because the low coefficient of determination R^2 suggests that the aforementioned relationship is poorly explained by a linear trend and that providing such graph might therefore be misleading.

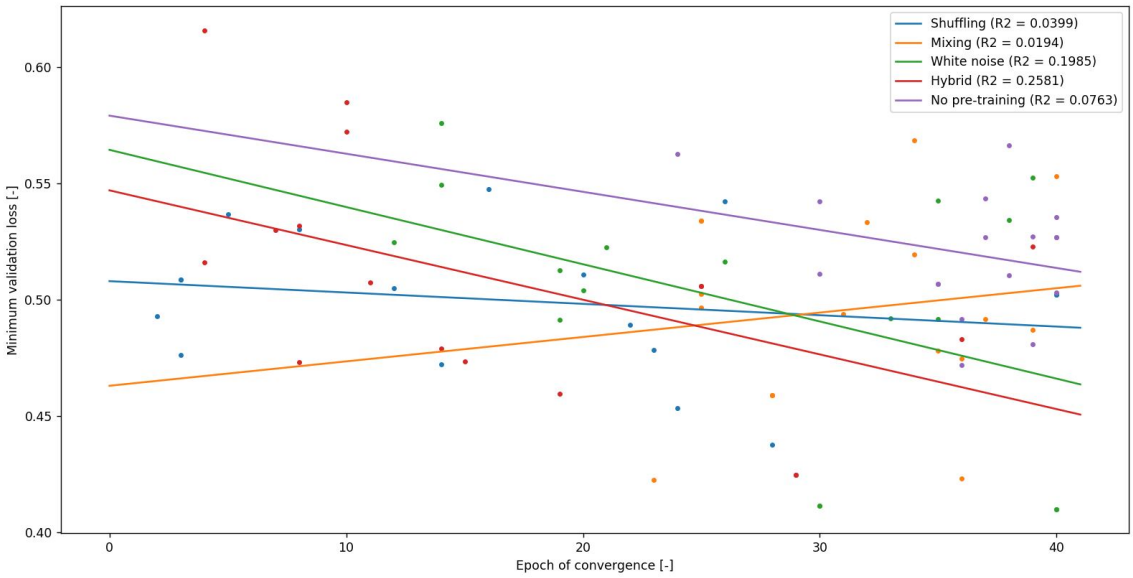

Figure B.2: Linear interpolation of the EOC to minimum validation loss relationship for the different models on the EO/EC dataset. R^2 coefficients are indicated in the legend.

B.2 Complements to section 5.2

Figure B.3 shows the evolution of the model trained on the TUSZ dataset during its pre-training phase. As a reminder, the pre-training task consists in classifying multichannel signals based on whether they are EEGs or not with, in this case, shuffled channels as the alteration for the negative case. This figure does not appear in the main work because such graph only allows to verify that the model handles the pre-training task as expected.

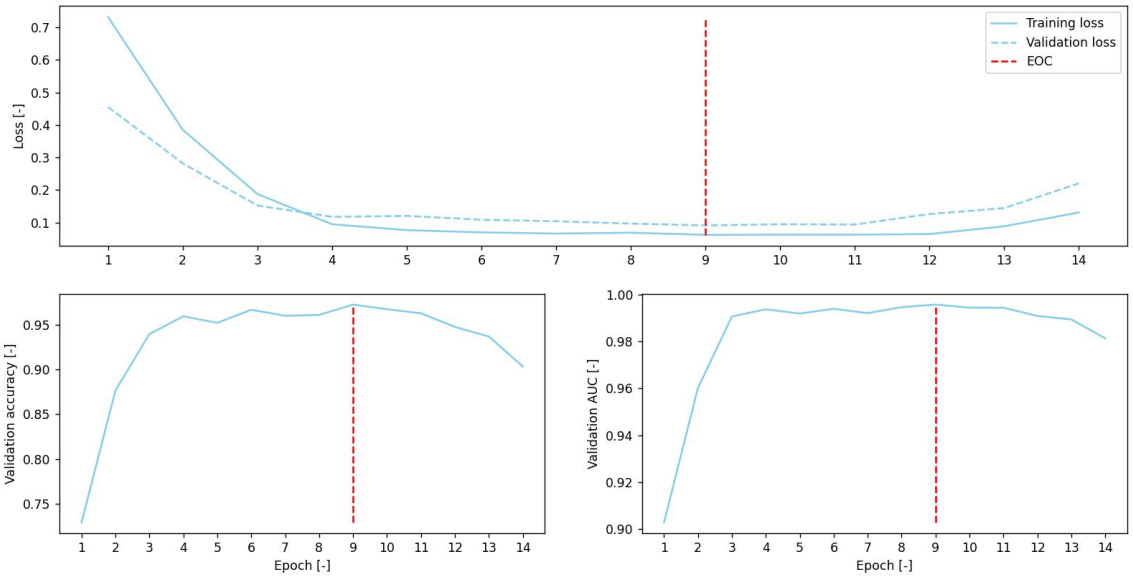

Figure B.3: Evolution of the performances of the big model during pre-training on the TUSZ dataset.

It is however interesting to observe that the training loss stays higher than the validation loss during the first few epochs, and that the validation loss starts to increase after a certain time. The first phenomenon is explained by the multiple regularization techniques involved in the model training to avoid overfitting, and by the fact that the training loss is computed trough the epoch whereas the validation loss is computed at the end of it [215].

The increase of the validation loss is due to the overfitting that, although slowed down by regularization, ends up occurring if the model is trained for too long (see section 4.3 for more details). This is why the (pre)-training is stopped if the validation loss keeps rising for a certain number of epochs, which is called early stopping.

Figure B.4 displays the same type of results as figure 5.3 (i.e. the comparison of the performances of the pre-trained and non pre-trained model during fine tuning on the TUSZ dataset), but is issued from an experiment with a learning rate equal to $l_r = 10^{-5}$ instead of $l_r = 10^{-4}$. This lower value, although showing a much higher performance gap between the pre-trained and non pre-trained models than the higher one, leads the best model to a higher validation and test loss (equal to respectively 0.1842 and 0.1925). This latter metric being the main selection criterion for the hyper-parameters, the value of 10^{-4} was conserved for the learning rate.

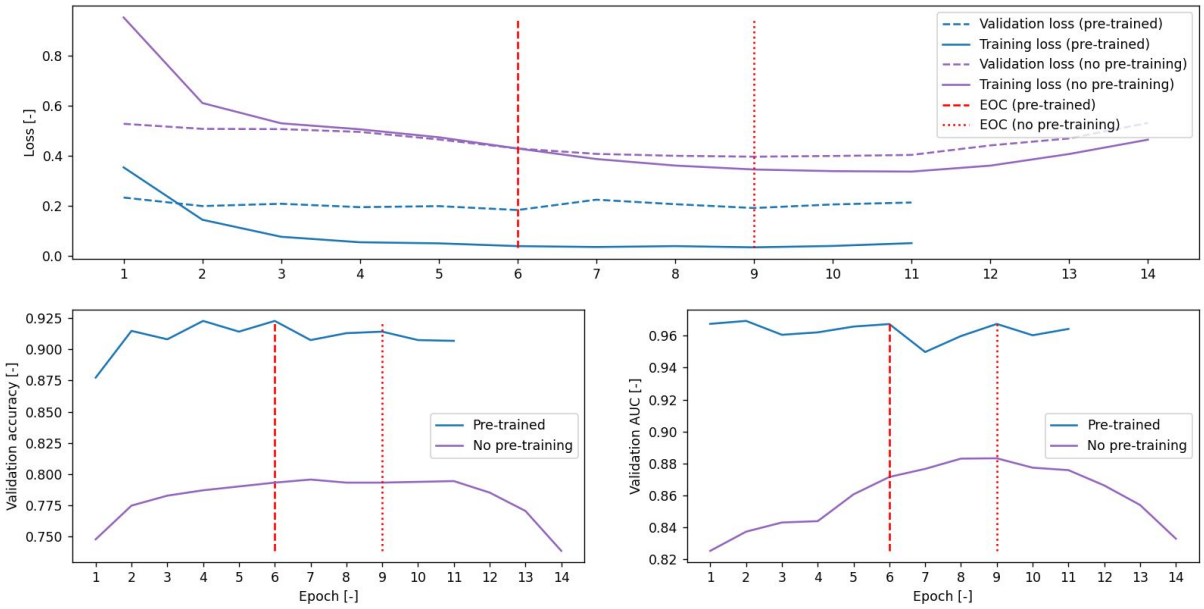

Figure B.4: Comparison of the evolutions of the performance metrics of the pre-trained and non pre-trained models during fine tuning using a sub-optimal learning rate ($l_r = 10^{-5}$).

Interesting observations can nevertheless be made from figure B.4, especially when compared to figure 5.3. First, the EOC of the pre-trained and non pre-trained models are respectively 6 and 9 for the models with $l_r = 10^{-5}$, versus 4 and 8 when $l_r = 10^{-4}$. This is because the learning rate represents the size of the step in the gradient descent algorithm and a smaller step size therefore requires more epochs to cover the same distance [113, 114].

In second, the literature was browsed to try to explain why a smaller learning rate was so detrimental to the non pre-trained model performances. The reason found is that starting the training of a model by using a small learning rate increases the chances for it to converge to a sub-optimal local minimum [216, 217], which is what might have happened here. We hypothesize that the pre-trained model escaped this fate because the pre-training shifts the gradient to a region near the global minimum. This assumption must still be verified, as no element in the literature was found to affirm nor deny it.

Bibliography

[1] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[2] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[3] Ramy Hussein, Soojin Lee, and Rabab Ward. "Multi-channel vision transformer for epileptic seizure prediction". In: *Biomedicines* 10.7 (2022), p. 1551.

[4] Martin Popel and Ondřej Bojar. "Training tips for the transformer model". In: *arXiv preprint arXiv:1804.00247* (2018).

[5] Dušan Variš and Ondřej Bojar. "Sequence length is a domain: Length-based overfitting in transformer models". In: *arXiv preprint arXiv:2109.07276* (2021).

[6] Carl Doersch and Andrew Zisserman. "Multi-task self-supervised visual learning". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2051–2060.

[7] Tong Zhang and F Oles. "The value of unlabeled data for classification problems". In: *Proceedings of the Seventeenth International Conference on Machine Learning,(Langley, P., ed.)* Vol. 20. Citeseer. 2000.

[8] Vinit Shah et al. "The temple university hospital seizure detection corpus". In: *Frontiers in neuroinformatics* 12 (2018), p. 83.

[9] Jay Alammar. *The Illustrated Transformer*. 2018. URL: `http://jalammar.github.io/illustrated-transformer/` (visited on 11/07/2022).

[10] Jesse Vig. "A multiscale visualization of attention in the transformer model". In: *arXiv preprint arXiv:1906.05714* (2019).

[11] Zhe Wang et al. "Transformers for EEG-based emotion recognition: A hierarchical spatial information learning model". In: *IEEE Sensors Journal* 22.5 (2022), pp. 4359–4368.

[12] Jin Xie et al. "A Transformer-based approach combining deep learning network and spatial-temporal information for raw EEG classification". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 30 (2022), pp. 2126–2136.

[13] Yonghao Song et al. "EEG Conformer: Convolutional Transformer for EEG Decoding and Visualization". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (2022).

[14] Jianzhuo Yan et al. “Seizure prediction based on transformer using scalp electroencephalogram”. In: *Applied Sciences* 12.9 (2022), p. 4158.

[15] Lisa Torrey and Jude Shavlik. “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[16] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[17] NIH. *What are the parts of the nervous system?* 2018. URL: https://www.nichd.nih.gov/health/topics/neuro/conditioninfo/parts (visited on 02/15/2023).

[18] C. Vandergriendt and R. Zimlich. *An Easy Guide to Neuron Anatomy with Diagrams*. 2022. URL: https://www.healthline.com/health/neurons (visited on 02/15/2023).

[19] J. Seifter, D. Sloane, and A. Ratner. “An Overview of Nerve Cell Physiology and the Autonomic Nervous System”. In: *Concepts in Medical Physiology*. Lippincott Williams & Wilkins, 2005. Chap. 5, pp. 50–59.

[20] Charles F Stevens. “The neuron”. In: *Scientific American* 241.3 (1979), pp. 54–65.

[21] Daniel K. Hartline. “What is myelin?” In: *Neuron Glia Biology* 4.2 (2008), pp. 153–163. DOI: 10.1017/S1740925X09990263.

[22] P Michael Conn. “Cytology of the Central Nervous System”. In: *Conn’s Translational Neuroscience*. Academic Press, 2017. Chap. 1, pp. 1–10. DOI: https://doi.org/10.1016/B978-0-12-802381-5.00001-4.

[23] Alan Peters and Sanford L Palay. “The morphology of synapses”. In: *Journal of neurocytology* 25 (1996), pp. 687–700.

[24] Keith Lucas. “The ‘all or none’ contraction of the amphibian skeletal muscle fibre”. In: *The Journal of Physiology* 38.2-3 (1909), pp. 113–133. DOI: https://doi.org/10.1113/jphysiol.1909.sp001298. eprint: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1909.sp001298. URL: https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1909.sp001298.

[25] John H Byrne. “Postsynaptic potentials and synaptic integration”. In: *From molecules to networks*. Elsevier, 2014, pp. 489–507. DOI: 10.1016/b978-0-12-397179-1.00016-6.

[26] Jahangir Moini, Nicholas Avgeropoulos, and Mohtashem Samsam. “Cytology of the nervous system”. In: Academic Press, 2021. DOI: 10.1016/B978-0-12-821736-8.00012-1.

[27] Germana Pareti. “The" all-or-none" law in skeletal muscle and nerve fibres.” In: *Archives italiennes de biologie* 145.1 (2007), pp. 39–54. DOI: 10.4449/aib.v145i1.865.

[28] E Florey. “Frequency and amplitude codes of neuronal signals”. In: *From Neuron to Action: An Appraisal of Fundamental and Clinical Research*. 1990, pp. 413–419. DOI: 10.1007/978-3-662-02601-4_47.

[29] Rümeysa İnce, Saliha Seda Adanır, and Fatma Sevmez. "The inventor of electroencephalography (EEG): Hans Berger (1873–1941)". In: *Child's Nervous System* 37 (2021), pp. 2723–2724.

[30] Anton Coenen and Oksana Zayachkivska. "Adolf Beck: A pioneer in electroencephalography in between Richard Caton and Hans Berger". In: *Advances in cognitive psychology* 9.4 (2013), p. 216.

[31] Erik K St Louis et al. *Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults*. 2016, pp. 1–95.

[32] Ray Cooper et al. "Comparison of subcortical, cortical and scalp activity using chronically indwelling electrodes in man". In: *Electroencephalography and clinical neurophysiology* 18.3 (1965), pp. 217–228.

[33] Timo Kirschstein and Rüdiger Köhling. "What is the source of the EEG?" In: *Clinical EEG and neuroscience* 40.3 (2009), pp. 146–149.

[34] H Petsche, H Pockberger, and P Rappelsberger. "On the search for the sources of the electroencephalogram". In: *Neuroscience* 11.1 (1984), pp. 1–27.

[35] Michelle Tjia et al. "Pyramidal neurons in different cortical layers exhibit distinct dynamics and plasticity of apical dendritic spines". In: *Frontiers in neural circuits* 11 (2017), p. 43.

[36] Herbert H Jasper. "Report of the committee on methods of clinical examination in electroencephalography: 1957". In: *Electroencephalogr Clin Neurophysiol* 10 (1958), pp. 370–375.

[37] Paul L Nunez and Ramesh Srinivasan. "Electroencephalogram". In: *Scholarpedia* 2.2 (2007), p. 1348.

[38] Robert Oostenveld and Peter Praamstra. "The five percent electrode system for high-resolution EEG and ERP measurements". In: *Clinical neurophysiology* 112.4 (2001), pp. 713–719.

[39] Jayant N Acharya and Vinita J Acharya. "Overview of EEG montages and principles of localization". In: *Journal of Clinical Neurophysiology* 36.5 (2019), pp. 325–329.

[40] Claudio Babiloni et al. "International Federation of Clinical Neurophysiology (IFCN)–EEG research workgroup: Recommendations on frequency and topographic analysis of resting state EEG rhythms. Part 1: Applications in clinical research studies". In: *Clinical Neurophysiology* 131.1 (2020), pp. 285–307.

[41] Appaji Rayi and Najib Murr. "Electroencephalogram". In: *StatPearls* (2020). URL: `https://www.ncbi.nlm.nih.gov/books/NBK563295/`.

[42] Alexander J Casson et al. "Electroencephalogram". In: *Seamless healthcare monitoring: advancements in wearable, attachable, and invisible devices* (2018), pp. 45–81.

[43] Priyanka A Abhang, Bharti W Gawali, and Suresh C Mehrotra. "Technological basics of EEG recording and operation of apparatus". In: Academic Press Cambridge, MA, USA, 2016, pp. 19–50.

[44] Mahtab Roohi-Azizi et al. “Changes of the brain’s bioelectrical activity in cognition, consciousness, and some mental disorders”. In: *Medical journal of the Islamic Republic of Iran* 31 (2017), p. 53.

[45] Alberto Zani et al. “Evoked and event-related potentials”. In: *Encyclopedia of Sciences and Religions* (2013), pp. 787–792.

[46] André Mouraux and Gian Domenico Iannetti. “Across-trial averaging of event-related EEG responses and beyond”. In: *Magnetic resonance imaging* 26.7 (2008), pp. 1041–1054.

[47] George D Dawson. “A summation technique for detecting small signals in an irregular background”. In: *J Physiol (Lond)* 115 (1951), pp. 2–3.

[48] Peter J Molfese. “Imaging studies of reading disabilities in children”. In: *Reading, Writing, Mathematics and the Developing Brain: Listening to Many Voices* (2012), pp. 49–64.

[49] John LaRocco, Minh Dong Le, and Dong-Guk Paeng. “A systemic review of available low-cost EEG headsets used for drowsiness detection”. In: *Frontiers in neuroinformatics* (2020), p. 42.

[50] Udaya Seneviratne and Wendyl Jude D’Souza. “Ambulatory EEG”. In: *Handbook of clinical neurology* 160 (2019), pp. 161–170.

[51] Simon O’Regan, Stephen Faul, and William Marnane. “Automatic detection of EEG artefacts arising from head movements”. In: *2010 annual international conference of the ieee engineering in medicine and biology*. IEEE. 2010, pp. 6353–6356.

[52] Emilie Bourel-Ponchel and Marie-Dominique Lamblin. *EEG in premature newborns*. 2021.

[53] Nikesh I Ardeshna. “EEG and Coma”. In: *The Neurodiagnostic Journal* 56.1 (2016), pp. 1–16.

[54] Thilo Hinterberger et al. “A brain–computer interface (BCI) for the locked-in: comparison of different EEG classifications for the thought translation device”. In: *Clinical neurophysiology* 114.3 (2003), pp. 416–425.

[55] Alexander T Sack et al. “Optimizing functional accuracy of TMS in cognitive studies: a comparison of methods”. In: *Journal of cognitive neuroscience* 21.2 (2009), pp. 207–221.

[56] Boris Burle et al. “Spatial and temporal resolutions of EEG: Is it really black and white? A scalp current density view”. In: *International Journal of Psychophysiology* 97.3 (2015), pp. 210–220.

[57] João Jorge, Wietske Van der Zwaag, and Patricia Figueiredo. “EEG–fMRI integration for the study of human brain function”. In: *Neuroimage* 102 (2014), pp. 24–34.

[58] Roberta Grech et al. “Review on solving the inverse problem in EEG source analysis”. In: *Journal of neuroengineering and rehabilitation* 5.1 (2008), pp. 1–33.

[59] A Wróbel. “Zbiorcza aktywność elektryczna mózgu (ENG: Collective electrical activity of the brain).” In: *Kosmos* 46.3 (1997). [Translated from Polish using DeepL], pp. 317–326.

[60] André Mouraux. “Bioinstrumentation in neurology”. [Unpublished course notes from UCLouvain]. 2022.

[61] William Cobb and TA Sears. “A study of the transmission of potentials after hemispherectomy”. In: *Electroencephalography and clinical neurophysiology* 12.2 (1960), pp. 371–383.

[62] Khald Ali I Aboalayon et al. “Sleep stage classification using EEG signal analysis: a comprehensive survey and new investigation”. In: *Entropy* 18.9 (2016), p. 272.

[63] Li Hu and Zhiguo Zhang. “EEG signal processing and feature extraction”. In: Springer, 2019. Chap. 5, pp. 71–89.

[64] Jake H McKay and William O Tatum. “Artifact mimicking ictal epileptiform activity in EEG”. In: *Journal of Clinical Neurophysiology* 36.4 (2019), pp. 275–288.

[65] Li Deng and Dong Yu. “Deep Learning: Methods and Applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4 (2014), pp. 197–387. ISSN: 1932-8346. DOI: 10.1561/2000000039. URL: http://dx.doi.org/10.1561/2000000039.

[66] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[67] Charles C. Tappert. “Who Is the Father of Deep Learning?” In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2019, pp. 343–348. DOI: 10.1109/CSCI49370.2019.00067.

[68] *Machine Learning Glossary*. 2022. URL: https://developers.google.com/machine-learning/glossary (visited on 11/07/2022).

[69] Sonali B Maind, Priyanka Wankar, et al. “Research paper on basic of artificial neural network”. In: *International Journal on Recent and Innovation Trends in Computing and Communication* 2.1 (2014), pp. 96–100.

[70] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[71] S Abirami and P Chitra. “Energy-efficient edge based real-time healthcare support system”. In: *Advances in computers*. Vol. 117. 1. Elsevier, 2020, pp. 339–368.

[72] Tim Menzies et al. “Chapter 24 - Using Goals in Model-Based Reasoning”. In: *Sharing Data and Models in Software Engineering*. Ed. by Tim Menzies et al. Boston: Morgan Kaufmann, 2015, pp. 321–353. DOI: https://doi.org/10.1016/B978-0-12-417295-1.00024-2.

[73] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. PMLR, 2011, pp. 315–323. URL: https://proceedings.mlr.press/v15/glorot11a.html.

[74] *E-Learning Project SOGA: Statistics and Geospatial Data Analysis*. 2018. URL: https://www.geo.fu-berlin.de/en/v/soga/Basics-of-statistics/Logistic-Regression/The-Logit-Function/index.html (visited on 02/28/2023).

[75] Pinagadi Venkateswararao and S Murugavalli. "CTC token parsing algorithm using keyword spotting for BLSTM based unconstrained handwritten recognition". In: *Journal of Ambient Intelligence and Humanized Computing* (2019), pp. 1–8.

[76] S Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger. *Handbook of medical image computing and computer assisted intervention*. Academic Press, 2019.

[77] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[78] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[79] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. "On the computational complexity of self-attention". In: *International Conference on Algorithmic Learning Theory*. PMLR. 2023, pp. 597–619.

[80] Yi Tay et al. "Efficient transformers: A survey". In: *ACM Computing Surveys* 55.6 (2022), pp. 1–28.

[81] Zhen Wu et al. "Unidrop: A simple yet effective technique to improve transformer without extra cost". In: *arXiv preprint arXiv:2104.04946* (2021).

[82] Yang Du et al. "EEG temporal–spatial transformer for person identification". In: *Scientific Reports* 12.1 (2022), p. 14378.

[83] Chao Che et al. "Constrained transformer network for ECG signal processing and arrhythmia classification". In: *BMC Medical Informatics and Decision Making* 21.1 (2021), pp. 1–13.

[84] Zenghui Wang et al. "Transformer model for functional near-infrared spectroscopy classification". In: *IEEE Journal of Biomedical and Health Informatics* 26.6 (2022), pp. 2559–2569.

[85] *Tokenizing with TF Text*. 2022. URL: https://www.tensorflow.org/text/guide/tokenizers (visited on 11/07/2022).

[86] Gregory Grefenstette. "Tokenization". In: *Syntactic Wordclass Tagging* (1999), pp. 117–133.

[87] Kayan K Katrak et al. "Transformers for Speaker Recognition". In: *Machine Learning and Autonomous Systems: Proceedings of ICMLAS 2021*. Springer, 2022, pp. 49–62.

[88] Philipp Dufter, Martin Schmitt, and Hinrich Schütze. "Position information in transformers: An overview". In: *Computational Linguistics* 48.3 (2022), pp. 733–763.

[89] Mehreen Saeed. *A Gentle Introduction to Positional Encoding in Transformer Models, Part 1*. 2022. URL: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/ (visited on 05/16/2023).

[90] *Neural machine translation with a Transformer and Keras*. 2022. URL: `https://www.tensorflow.org/text/tutorials/transformer` (visited on 11/08/2022).

[91] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[92] Stuart J Russell. *Artificial intelligence: a modern approach*. Pearson Education, Inc., 2010.

[93] Happiness Ugochi Dike et al. "Unsupervised learning based on artificial neural network: A review". In: *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. IEEE. 2018, pp. 322–327.

[94] Omar Alonso. "Challenges with label quality for supervised learning". In: *Journal of Data and Information Quality (JDIQ)* 6.1 (2015), pp. 1–3.

[95] Meng Fang, Yuan Li, and Trevor Cohn. "Learning how to active learn: A deep reinforcement learning approach". In: *arXiv preprint arXiv:1708.02383* (2017).

[96] Khadija El Bouchefry and Rafael S de Souza. "Learning in big data: Introduction to machine learning". In: *Knowledge discovery in big data from astronomy and earth observation*. Elsevier, 2020, pp. 225–249.

[97] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. "A primer in BERTology: What we know about how BERT works". In: *Transactions of the Association for Computational Linguistics* 8 (2021), pp. 842–866.

[98] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[99] Hangbo Bao et al. "Beit: Bert pre-training of image transformers". In: *arXiv preprint arXiv:2106.08254* (2021).

[100] Zhigang Dai et al. "Up-detr: Unsupervised pre-training for object detection with transformers". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1601–1610.

[101] Sara Atito, Muhammad Awais, and Josef Kittler. "SiT: Self-supervised vision transformer". In: *arXiv preprint arXiv:2104.03602* (2021).

[102] Chuang Lin et al. "A BERT based method for continuous estimation of cross-subject hand kinematics from surface electromyographic signals". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* (2022).

[103] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[104] Zejia Weng et al. "Semi-supervised vision transformers". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX*. Springer. 2022, pp. 605–620.

[105] Xiangde Luo et al. "Semi-supervised medical image segmentation via cross teaching between cnn and transformer". In: *International Conference on Medical Imaging with Deep Learning*. PMLR. 2022, pp. 820–833.

[106] Lorenzo Rosasco et al. “Are loss functions all the same?” In: *Neural computation* 16.5 (2004), pp. 1063–1076.

[107] Hichame Yessou, Gencer Sumbul, and Begüm Demir. “A comparative study of deep learning loss functions for multi-label remote sensing image classification”. In: *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2020, pp. 1349–1352.

[108] Katarzyna Janocha and Wojciech Marian Czarnecki. “On loss functions for deep neural networks in classification”. In: *arXiv preprint arXiv:1702.05659* (2017).

[109] *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. 2018. URL: `https://gombru.github.io/2018/05/23/cross_entropy_loss/` (visited on 03/04/2023).

[110] Daniel Ramos et al. “Deconstructing cross-entropy for probabilistic binary classifiers”. In: *Entropy* 20.3 (2018), p. 208.

[111] Li Li, Miloš Doroslovački, and Murray H Loew. “Approximating the gradient of cross-entropy loss function”. In: *IEEE Access* 8 (2020), pp. 111626–111635.

[112] Augustin Cauchy et al. “Méthode générale pour la résolution des systemes d’équations simultanées”. In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[113] *Understanding Learning Rate in Neural Networks*. 2019. URL: `https://www.allaboutcircuits.com/technical-articles/understanding-learning-rate-in-neural-networks/` (visited on 03/05/2023).

[114] Nithin Buduma, Nikhil Buduma, and Joe Papa. “Training Feed-Forward Neural Networks”. In: *Fundamentals of deep learning*. " O’Reilly Media, Inc.", 2022. Chap. 2, pp. 17–37.

[115] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[116] James Gareth et al. “Statistical Learning”. In: *An introduction to statistical learning: with applications in R*. Spinger, 2013. Chap. 2, pp. 29–37.

[117] World Health Organization. *Mental health: neurological disorders*. 2016. URL: `https://www.who.int/news-room/questions-and-answers/item/mental-health-neurological-disorders` (visited on 03/08/2023).

[118] S Rees, Richard Harding, Terrie Inder, et al. “The developmental environment and the origins of neurological disorders.” In: *Developmental origins of health and disease* (2006), pp. 379–391.

[119] World Health Organization. Mental Health Evidence et al. *Disease control priorities related to mental, neurological, developmental and substance abuse disorders*. World Health Organization, 2006.

[120] World Health Organization. *Neurological disorders: public health challenges*. World Health Organization, 2006.

[121] Vikram Patel et al. *Disease control priorities, (volume 4): mental, neurological, and substance use disorders*. World Bank Publications, 2016.

[122] Christopher J Murray. "Quantifying the burden of disease: the technical basis for disability-adjusted life years." In: *Bulletin of the World health Organization* 72.3 (1994), p. 429.

[123] Hojjat Adeli and Samanwoy Ghosh-Dastidar. *Automated EEG-based diagnosis of neurological disorders: Inventing the future of neurology*. CRC press, 2010.

[124] Michael A Rogawski and Wolfgang Löscher. "The neurobiology of antiepileptic drugs". In: *Nature reviews neuroscience* 5.7 (2004), pp. 553–564.

[125] Marie-Christine Picot et al. "The prevalence of epilepsy and pharmacoresistant epilepsy in adults: a population-based study in a Western European country". In: *Epilepsia* 49.7 (2008), pp. 1230–1238.

[126] Panagiotis Kerezoudis et al. "Surgical outcomes of laser interstitial thermal therapy for temporal lobe epilepsy: systematic review and meta-analysis". In: *World neurosurgery* 143 (2020), pp. 527–536.

[127] Dorian M Kusyk et al. "Systematic review and meta-analysis of responsive neurostimulation in epilepsy". In: *World Neurosurgery* (2022).

[128] Barbara C Jobst et al. "Brain stimulation for the treatment of epilepsy". In: *Epilepsia* 51 (2010), pp. 88–92.

[129] Martha J Morrell. "Responsive cortical stimulation for the treatment of medically intractable partial epilepsy". In: *Neurology* 77.13 (2011), pp. 1295–1304.

[130] Jared Fridley et al. "Brain stimulation for the treatment of epilepsy". In: *Neurosurgical focus* 32.3 (2012), E13.

[131] Ujwal Boddeti et al. "Responsive Neurostimulation for Seizure Control: Current Status and Future Directions". In: *Biomedicines* 10.11 (2022), p. 2677.

[132] Victoria Peterson et al. "Deep net detection and onset prediction of electrographic seizure patterns in responsive neurostimulation". In: *Epilepsia* (2023).

[133] Song Cui et al. "Learning EEG synchronization patterns for epileptic seizure prediction using bag-of-wave features". In: *Journal of Ambient Intelligence and Humanized Computing* (2018), pp. 1–16.

[134] Robert S Fisher, Helen E Scharfman, and Marco DeCurtis. "How can we identify ictal and interictal abnormal activity?" In: *Issues in Clinical Epileptology: A View from the Bench* (2014), pp. 3–23.

[135] Attila Balogh. "O23 Characteristic slow gamma and delta frequency changes in the preictal EEG of partial epileptic patients". In: *Clinical Neurophysiology* 128.9 (2017), e187.

[136] Florian Mormann et al. "On the predictability of epileptic seizures". In: *Clinical neurophysiology* 116.3 (2005), pp. 569–587.

[137] Alexander Craik, Yongtian He, and Jose L Contreras-Vidal. "Deep learning for electroencephalogram (EEG) classification tasks: a review". In: *Journal of neural engineering* 16.3 (2019), p. 031001.

[138] Gen Li et al. “Deep learning for EEG data analytics: A survey”. In: *Concurrency and Computation: Practice and Experience* 32.18 (2020), e5199.

[139] Mohammad-Parsa Hosseini, Amin Hosseini, and Kiarash Ahi. “A review on machine learning for EEG signal processing in bioengineering”. In: *IEEE reviews in biomedical engineering* 14 (2020), pp. 204–218.

[140] Fatima Hassan and Syed Fawad Hussain. “Review of EEG Signals Classification Using Machine Learning and Deep-Learning Techniques”. In: *Advances in Non-Invasive Biomedical Signal Sensing and Processing with Machine Learning*. Springer, 2023, pp. 159–183.

[141] Khondoker Murad Hossain et al. “Status of deep learning for EEG-based brain–computer interface applications”. In: *UMBC Student Collection* (2023).

[142] Aarti Sharma, JK Rai, and RP Tewari. “Identification of Various Neurological Disorders Using EEG Signals”. In: *Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, April 12–13, 2019, Revised Selected Papers, Part I 3*. Springer. 2019, pp. 95–103.

[143] Fahd A Alturki et al. “EEG signal analysis for diagnosing neurological disorders using discrete wavelet transform and intelligent techniques”. In: *Sensors* 20.9 (2020), p. 2505.

[144] Sampsa Vanhatalo, Juha Voipio, and Kai Kaila. “Full-band EEG (FbEEG): an emerging standard in electroencephalography”. In: *Clinical Neurophysiology* 116.1 (2005), pp. 1–8.

[145] Jasjeet Kaur and Amanpreet Kaur. “A review on analysis of EEG signals”. In: *2015 International Conference on Advances in Computer Engineering and Applications*. IEEE. 2015, pp. 957–960.

[146] Christian Jutten and Jeanny Herault. “Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture”. In: *Signal processing* 24.1 (1991), pp. 1–10.

[147] Michel Verleysen. “Independent Component Analysis (ICA)”. [Unpublished course notes from UCLouvain]. 2022.

[148] Zoltán Tüske et al. “Acoustic modeling with deep neural networks using raw time signal for LVCSR”. In: *Fifteenth annual conference of the international speech communication association*. Citeseer. 2014.

[149] Peter H Westfall. “Kurtosis as peakedness, 1905–2014. RIP”. In: *The American Statistician* 68.3 (2014), pp. 191–195.

[150] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.

[151] Julius Orion Smith. “DFT Application”. In: *Mathematics of the discrete Fourier transform (DFT): with audio applications*. Julius Smith, 2008. Chap. 8, pp. 165–193.

[152] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. "A scale for the measurement of the psychological magnitude pitch". In: *The journal of the acoustical society of america* 8.3 (1937), pp. 185–190.

[153] Paul S Addison. "Wavelet transforms and the ECG: a review". In: *Physiological measurement* 26.5 (2005), R155.

[154] Michel Verleysen. "Wavelet transform". [Unpublished course notes from UCLouvain]. 2022.

[155] Stéphane Mallat. "Time meets frequency". In: *A wavelet tour of signal processing*. Elsevier, 1999. Chap. 4, pp. 102–115.

[156] Siyi Deng et al. "EEG classification of imagined syllable rhythm using Hilbert spectrum methods". In: *Journal of neural engineering* 7.4 (2010), p. 046006.

[157] Ahmad Alwosheel, Sander van Cranenburgh, and Caspar G Chorus. "Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis". In: *Journal of choice modelling* 28 (2018), pp. 167–182.

[158] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20 (1995), pp. 273–297.

[159] Gourav Siddhad et al. "Efficacy of transformer networks for classification of raw EEG data". In: *arXiv preprint arXiv:2202.05170* (2022).

[160] Demetres Kostas, Stephane Aroca-Ouellette, and Frank Rudzicz. "BENDR: using transformers and a contrastive self-supervised learning task to learn from massive amounts of EEG data". In: *Frontiers in Human Neuroscience* 15 (2021), p. 653659.

[161] Namuk Park and Songkuk Kim. "How do vision transformers work?" In: *arXiv preprint arXiv:2202.06709* (2022).

[162] SGOPAL Patro and Kishore Kumar Sahu. "Normalization: A preprocessing stage". In: *arXiv preprint arXiv:1503.06462* (2015).

[163] Hao Qu and Jean Gotman. "A patient-specific algorithm for the detection of seizure onset in long-term EEG monitoring: possible use as a warning device". In: *IEEE transactions on biomedical engineering* 44.2 (1997), pp. 115–122.

[164] Georgiy R Minasyan et al. "Patient-specific early seizure detection from scalp EEG". In: *Journal of clinical neurophysiology: official publication of the American Electroencephalographic Society* 27.3 (2010), p. 163.

[165] Alexei Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in neural information processing systems* 33 (2020), pp. 12449–12460.

[166] André Mouraux, Michel Verleysen, and Anne-Sophie Collin. "Project EEG classification : Open your eyes!" [Unpublished course notes from UCLouvain]. 2021.

[167] Robert J Barry et al. "EEG differences between eyes-closed and eyes-open resting conditions". In: *Clinical neurophysiology* 118.12 (2007), pp. 2765–2773.

[168] Robert J Barry and Frances M De Blasio. "EEG differences between eyes-closed and eyes-open resting remain in healthy ageing". In: *Biological psychology* 129 (2017), pp. 293–304.

[169] Iyad Obeid and Joseph Picone. “The temple university hospital EEG data corpus”. In: *Frontiers in neuroscience* 10 (2016), p. 196.

[170] Bob Kemp et al. “A simple format for exchange of digitized polygraphic recordings”. In: *Electroencephalography and clinical neurophysiology* 82.5 (1992), pp. 391–393.

[171] Sean Ferrell et al. “The temple university hospital EEG corpus: Electrode location and channel labels”. In: *Institute for Signal and Information Processing Report* 1.1 (2020).

[172] Haibo He and Edwardo A Garcia. “Learning from imbalanced data”. In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[173] Jonathan M Lilly and Sofia C Olhede. “Higher-order properties of analytic wavelets”. In: *IEEE Transactions on Signal Processing* 57.1 (2008), pp. 146–160.

[174] GRJ Cooper and DR Cowan. “Comparing time series using wavelet-based semblance analysis”. In: *Computers & Geosciences* 34.2 (2008), pp. 95–102.

[175] Ping Li, Trevor J Hastie, and Kenneth W Church. “Very sparse random projections”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 287–296.

[176] William B Johnson. “Extensions of Lipschitz mappings into a Hilbert space”. In: *Contemp. Math.* 26 (1984), pp. 189–206.

[177] Sanjoy Dasgupta and Anupam Gupta. “An elementary proof of a theorem of Johnson and Lindenstrauss”. In: *Random Structures & Algorithms* 22.1 (2003), pp. 60–65.

[178] Debra A Lelewer and Daniel S Hirschberg. “Data compression”. In: *ACM Computing Surveys (CSUR)* 19.3 (1987), pp. 261–296.

[179] Zelun Wang and Jyh-Charn Liu. “Translating math formula images to LaTeX sequences using deep neural networks with sequence-level training”. In: *International Journal on Document Analysis and Recognition (IJDAR)* 24.1-2 (2021), pp. 63–75.

[180] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[181] Benyamin Ghojogh and Mark Crowley. “The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial”. In: *arXiv preprint arXiv:1905.12787* (2019).

[182] Anders Krogh and John Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems* 4 (1991).

[183] Eric Baum and David Haussler. “What size net gives valid generalization?” In: *Advances in neural information processing systems* 1 (1988).

[184] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[185] Martin Abadi et al. “Tensorflow: a system for large-scale machine learning.” In: *Osdi*. Vol. 16. 2016. Savannah, GA, USA. 2016, pp. 265–283.

[186] Khalid Salama. *Image classification with Vision Transformer*. 2021. URL: https://keras.io/examples/vision/image_classification_with_vision_transformer/ (visited on 11/08/2022).

[187] Lutz Prechelt. "Early stopping—but when?" In: *Neural networks: tricks of the trade: second edition* (2012), pp. 53–67.

[188] Maren Mahsereci et al. "Early stopping without a validation set". In: *arXiv preprint arXiv:1703.09580* (2017).

[189] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[190] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[191] Stevo Bozinovski. "Reminder of the first paper on transfer learning in neural networks, 1976". In: *Informatica* 44.3 (2020).

[192] Hamidreza Namazi and Vladimir V Kulish. "Mathematical modeling of human brain neuronal activity in the absence of external stimuli". In: *Journal of Medical Imaging and Health Informatics* 2.4 (2012), pp. 400–407.

[193] Robert J Barry and Frances M De Blasio. "Characterizing pink and white noise in the human electroencephalogram". In: *Journal of Neural Engineering* 18.3 (2021), p. 034001.

[194] Behzad Hejrati, Abdolhossein Fathi, and Fardin Abdali-Mohammadi. "Efficient lossless multi-channel EEG compression based on channel clustering". In: *Biomedical Signal Processing and Control* 31 (2017), pp. 295–300.

[195] Ronakben Bhavsar et al. "The correlation between EEG signals as measured in different positions on scalp varying with distance". In: *Procedia computer science* 123 (2018), pp. 92–97.

[196] Mieczyslaw A Klopotek, Slawomir T Wierzchon, and Krzysztof Trojanowski. "Confusion Matrix Visualization". In: *Intelligent Information Processing and Web Mining*. Vol. 22. Springer Science & Business Media, 2013, pp. 107–117.

[197] David MW Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *arXiv preprint arXiv:2010.16061* (2020).

[198] Richard Dinga et al. "Beyond accuracy: measures for assessing machine learning models, pitfalls and guidelines". In: *BioRxiv* (2019), p. 743138.

[199] T Fawcett. "Introduction to receiver operator curves". In: *Pattern Recognit. Lett* 27 (2006), pp. 861–874.

[200] Stephanie Glenn. *IID Statistics: Independent and Identically Distributed Definition and Examples*. 2016. URL: https://www.statisticshowto.com/iid-statistics/ (visited on 04/26/2023).

[201] Bernard L Welch. "The generalization of 'STUDENT'S'problem when several different population varlances are involved". In: *Biometrika* 34.1-2 (1947), pp. 28–35.

[202] Hans Fischer. “Introduction”. In: *A history of the central limit theorem: from classical to modern probability theory*. Springer, 2011. Chap. 1, pp. 1–14.

[203] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation”. In: *PeerJ Computer Science* 7 (2021), e623.

[204] Stephen Allwright. *How to interpret R Squared (simply explained)*. 2022. URL: `https://stephenallwright.com/interpret-r-squared/` (visited on 04/26/2023).

[205] Inge S Helland. “On the interpretation and use of R2 in regression analysis”. In: *Biometrics* (1987), pp. 61–69.

[206] Tae Kyun Kim. “T test as a parametric statistic”. In: *Korean journal of anesthesiology* 68.6 (2015), pp. 540–546.

[207] Mosuk Chow. *Estimating Population Mean and Total under SRS*. [Plublic course notes from Penn State University]. 2023. URL: `https://online.stat.psu.edu/stat506/lesson/1/1.4` (visited on 04/26/2023).

[208] Sang Gyu Kwak and Jong Hae Kim. “Central limit theorem: the cornerstone of modern statistics”. In: *Korean journal of anesthesiology* 70.2 (2017), pp. 144–156.

[209] Huanru Henry Mao. “A survey on self-supervised pre-training for sequential transfer learning in neural networks”. In: *arXiv preprint arXiv:2007.00800* (2020).

[210] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[211] Ohbyung Kwon and Jae Mun Sim. “Effects of data set features on the performances of classification algorithms”. In: *Expert Systems with Applications* 40.5 (2013), pp. 1847–1857.

[212] Alhanoof Althnian et al. “Impact of dataset size on classification performance: an empirical evaluation in the medical domain”. In: *Applied Sciences* 11.2 (2021), p. 796.

[213] Demetres Kostas and Frank Rudzicz. “DN3: An open-source Python library for large-scale raw neurophysiology data assimilation for more flexible and standardized deep learning”. In: *bioRxiv* (2020), pp. 2020–12.

[214] Alexandre Gramfort et al. “MEG and EEG data analysis with MNE-Python”. In: *Frontiers in neuroscience* (2013), p. 267.

[215] Adrian Rosebrock. *Why is my validation loss lower than my training loss?* 2019. URL: `https://pyimagesearch.com/2019/10/14/why-is-my-validation-loss-lower-than-my-training-loss/` (visited on 05/29/2023).

[216] Amirkeivan Mohtashami, Martin Jaggi, and Sebastian Stich. “On Avoiding Local Minima Using Gradient Descent With Large Learning Rates”. In: *arXiv preprint arXiv:2205.15142* (2022).

[217] Kaichao You et al. “How does learning rate decay help modern neural networks?” In: *arXiv preprint arXiv:1908.01878* (2019).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**