UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

HENRIQUE INDALENCIO VALCANAIA

TS2Image: A software to convert EEG time series into images for training Brain-Computer Interface Convolutional Neural Networks

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Dante Augusto Couto Barone
Coadvisor: MSc. Jaime A. Riascos Salas

Porto Alegre
December 2021

"If I had an hour to solve a problem
I'd spend 55 minutes thinking about the problem
and five minutes thinking about solutions."

— ALBERT EINSTEIN

ACKNOWLEDGMENTS

Agradeço aos meus pais, por todo o suporte ao longo da vida e por sempre buscarem oferecer a melhor educação possível. Educação e conhecimento sempre se pagam, sempre valem a pena, e dessa vez não foi diferente. Obrigado, mãe e pai!

Agradeço aos meus avós, que estiveram sempre preocupados com o netinho deles. Infelizmente não os tenho mais ao lado hoje mas, vô Adolfo, vó Alba e vó Neta, estou me formando.

Eu não poderia encerrar esse capítulo da minha vida sem agradecer também aos meus amigos. Agradeço toda a parceria durante a graduação, que foi parte essencial para tornar esse momento realidade. Obrigados, amigos!

Ao Jaime Salas, por me introduzir e guiar na pesquisa desta área completamente nova para mim.

Ao professor Dante Barone, pela flexibilidade em me ajudar a encontrar uma área de interesse para realização desse trabalho.

Obrigado a todos!

ABSTRACT

The current advances in neuroscience, signal processing, and artificial intelligence allow scientists to explore new ways to communicate with computers continually. These efforts will drastically impact society, especially individuals with neuromuscular disorders that prevent them from using conventional communication and/or motor methods. Brain-Computer Interfaces (BCI) are the current state-of-the-art method that seeks to tackle the mentioned difficulties. Due to the lack of pre-existing software tooling, BCI research workflows and pipelines are currently developed on-demand, requiring a computer science background. Therefore, this work presents a new software tool and pipeline to make BCI research accessible to those lacking a computer science background for signal processing. The proposed software generates images from EEG signals to then be used to train a Deep Learning model, namely, a Convolutional Neural Network (CNN), which aims to extract and classify the BCI features automatically. The initial implementation comprises Python algorithms for generating the images from the time series (EEG signals) using the Gramian Angular Field (GAF) and Event-Related Spectral Dynamics (ERSP) techniques. This software aims to reduce time and effort in creating image data sets to train CNN models, using rich diversity of customizable settings, like the window size, channels, and method (GAF or ERSP), that can improve the classification rates. Thus, we expect the BCI research community to benefit from this tool, allowing the ability to explore the EEG classification problem using different images rather than the time series for explaining BCI signatures and improving the classification stage. Finally, we experiment with the software training a Deep Learning architecture (VGG-16) using the generated images from the well-known BCI Competition IV data set B.

Keywords: Brain-computer interface. gramian angular field. event-related spectral perturbation. convolutional neural networks. electroencephalography. machine learning. deep learning.

TS2Image: Um programa para converter séries temporais de EEG em imagens para treinamento de Redes Neurais Convolucionais de sistemas de Inteface Cérebro-Computador

RESUMO

Os recentes avanços em neurosciencia, processamento de sinais, e inteligência artificial permitem cientistas explorar novas formas de se comunicar com computador continuamente. Estes esforços impactarão drásticamente a sociedade, especialmente indivíduos com distúrbios neuromusculares que os impedem de utilizar métodos convencionais de comunicação e/ou motores. As Interfaces Cérebro-Computador (ICC) são o atual método estado da arte que busca contornar as dificuldades mencionadas. Devido à falta de ferramentas de software pré-existentes, os fluxos de trabalho e pipelines de pesquisa em ICC são desenvolvidos sob demanda, exigindo conhecimentos em ciência da computação. Portanto, este trabalho apresenta uma nova ferramenta de software e pipeline para tornar a pesquisa em ICC mais acessível aqueles que não possuem experiência em ciência da computação para processamento de sinais. O software proposto gera imagens a partir de sinais de EEG para, então, serem utilizadas no treinamento de um modelo de Deep Learning, mais especificamente, uma Rede Neural Convolucional (RNC), visando extrair e classificar as características da ICC de forma automática. A implementação inicial inclui algoritmos em Python para geração de imagens a partir de séries temporais (sinais EEG) usando as técnicas Gramian Angular Field (GAF) e Event-Related Spectral Dynamics (ERSP). Este software visa reduzir o tempo e esforço na criação de conjuntos de dados de imagem para treinar modelos CNN, utilizando uma rica diversidade de configurações personalizáveis, como o tamanho da janela, canais e método (GAF ou ERSP), que podem melhorar as taxas de classificação. Assim, esperamos que a comunidade de pesquisa em ICC se beneficie desta ferramenta, permitindo a capacidade de explorar o problema de classificação EEG usando imagens diferentes ao invés de séries temporais para explicar assinaturas de sinais de ICCs e melhorar o estágio de classificação. Finalmente, experimentamos o de treinamento uma arquitetura de Deep Learning (VGG-16) usando as imagens geradas pelo software a partir do conhecido conjunto de dados BCI Competition IV.

Palavras-chave: redes neurais, gramian angular field, event-related spectral perturbation.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
BCI	Brain-Computer Interface
CNN	Convolutional Neural Networks
DL	Deep Learning
EEG	Electroencephalography
ERSP	Event-Related Spectral Perturbation
GAF	Gramian Angular Field
IIC	Interface Cérebro-Computador
RNC	Redes Neurais Convolucionais
ML	Machine Learning

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Society is evolving and eager to find faster, more reliable, and accessible ways to communicate with machines and systems. Every advance in this area makes the technology cheaper, more portable, and open for the lower-income population. Communication between humans and machines also significantly impacts the accessibility realm, where people can rely on prostheses controlled by some commands sent to it by the impaired person.

A Brain-Computer Interface(BCI) is a hardware and software system that enables communicatation between humand and machines using brain signals. BCI is sometimes know as brain-machine interface(BMI). Besides the impressive progress in BCI research, finding practical and useful real-world application is still a challenge (CHAVARRIAGA et al., 2016).

Thus, this thesis seeks to develop a software able to translate the brain signals (time-series) into images to train deep learning models, offering a new alternative to create and configure the classification step in the BCI pipeline.

1.1 Motivation and scope

BCI applications could be more appropriately explored with advanced tools and software; therefore, more and more researchers without computer science backgrounds can investigate this area if the number of tools and software increases. The lack of software tooling to easily manipulate EEG data and posteriorly train Machine Learning models for BCI negatively impacts this research domain's development. Indeed, these difficulties have been a vital motivation factor for this work and experimenting with novel approaches to EEG signal classification like Convolutional Neural Networks using Gramian Angular Field (GAF) and Event-Related Spectral Perturbation (ERSP) images.

The scope of this work is to create an extensible open-source software with configurable parameters to convert time series files into GAF or ERSP images. To test the software's image generation EEG data sets are used, and then these images are used to train a CNN to classify other EEG signals.

1.2 Problem description

BCI devices can be seen as a signal pattern recognition system (Prashant; Joshi; Gandhi, 2015), hence it's vital for it to have great digital signal processing and well-trained machine learning models (CHAVARRIAGA et al., 2016).

While (PASCANU; MIKOLOV; BENGIO, 2013) wrote about the how hard is it to properly train recurrent neural networks(RNN), Convolutional Neural Networks have shown outstanding results in image classification in many different applications. As mentioned in 1.1, the lack of software tooling is also a primary factor here, many times requiring researchers to have some programming knowledge to manipulate the data they want to work with.

1.3 Objective and approach

This work aims to build software to convert EEG time-series into ERSP or GAF images, enabling researchers to generate images and train CNN models for BCI applications or whatever else these images might be helpful to them. To accomplish such objective, the following specific tasks are proposed:

1. Understanding EEG, BCI, time-series and Machine Learning fundamentals.
2. Exploring EEG data sets focusing on BCI ones.
3. Studying time series data set exploration tools.
4. Research existing libraries to analyze EEG data.
5. Software prototyping.
6. Software design, test, clean up, documentation and publication.
7. Machine Learning model training.

1.4 Outline

The following chapter presents a background in brain physiology and neuroimaging, Electroencephalography, BCI, time-series, and machine learning. Chapter 3 presents related work on BCI, time-series analysis, the usage of CNN with time series, and lastly, CNN usage for BCI systems. Section 4 describes the materials and methods to be used by this work. Lastly, chapter 5 presents the main findings and chapter 6 the respective conclusions.

2 BACKGROUND

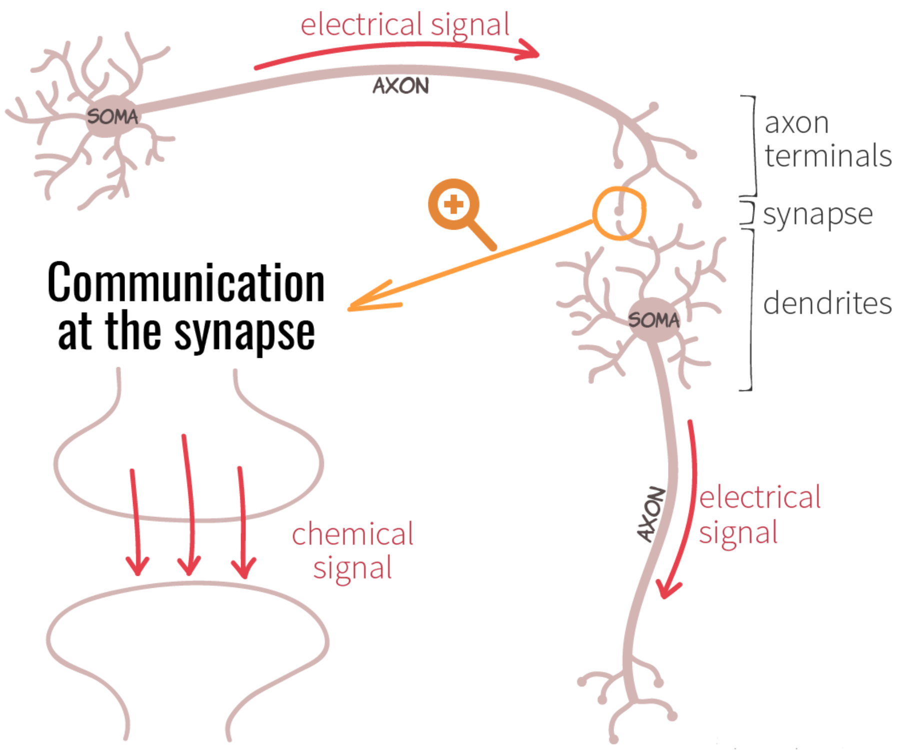
This section will introduce the reader to the main subjects related to this project: Brain and Neuroimaging, Electroencephalography (EEG), Brain-Computer Interface (BCI), Time Series, and Machine Learning.

2.1 Brain and Neuroimaging

The 86 billion neurons of the human brain are fundamentally building blocks, relying on chemical and also electric signals to transmit information (MORRISON, 2018). A neuron is composed of three essential parts: soma or cell body, axon, and dendrites. The soma is the cell body, where we can find the nucleus and the dendrites; dendrites are ramifications in the cell body; the axon is a conductor for electrical signals from the soma to its terminals; and finally, a synapse is a communication between one neuron's axon terminals and another neuron's dendrites. Figure 2.1 shows a simplified version of a neuron with these parts.

Electroencephalography (EEG) is a neuroimaging technique that measures electric potentials, but this is not the only type of signal that can be measured. Magnetic and metabolic brain activity can also be measured by Magnetoencephalography (MEG) and Functional Magnetic Resonance Imaging (fMRI), being these some of the most representative recording methods in BCI literature (Prashant; Joshi; Gandhi, 2015). In this work, we will focus on EEG.

Figure 2.1: A simplified version of the neuron with its fundamental parts.



Adapted from https://sapiensoup.com/serotonin

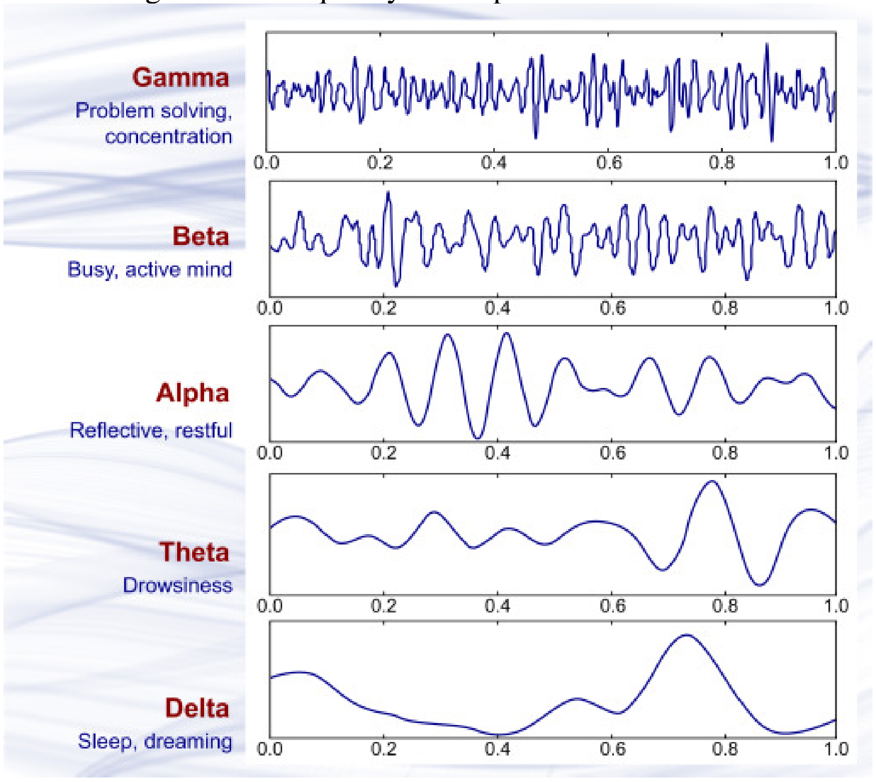
2.2 Electroencephalography

Due to the expressive number of neurons generating currents, some of the electric potentials reach the scalp surface, and we can measure these electric potentials with electrodes in a subject's head. Hans Berger, a German psychiatrist, first introduced this neuroimaging technique usage in humans (BERGER, 1929), (LOUIS et al., 2016) that is called Electroencephalography (EEG) (BRONZINO, 1970). It records the electric potential changes over time, creating a time series(more about time series later). Figure 2.2 shows the main frequency bands present in brain waves.

2.2.1 Acquisition and type of signals

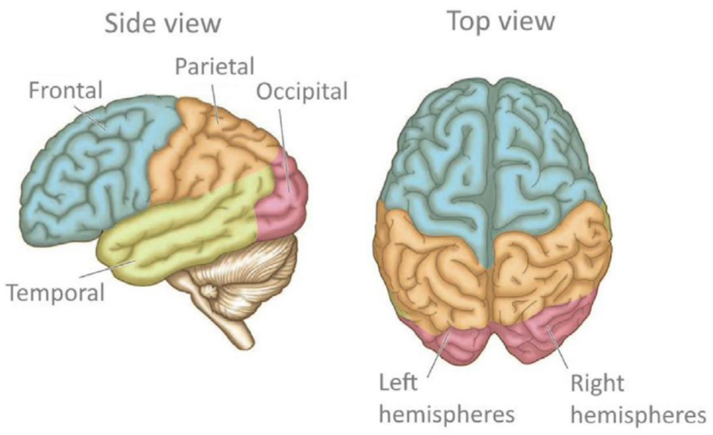
The brain is divided into regions. It has a left and right hemisphere (MORRISON, 2018), and Figure 2.3 shows such brain divisions - the frontal, temporal, parietal, and occipital lobes.

Figure 2.2: Frequency bands produced in the brain



Source: https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/brain-waves

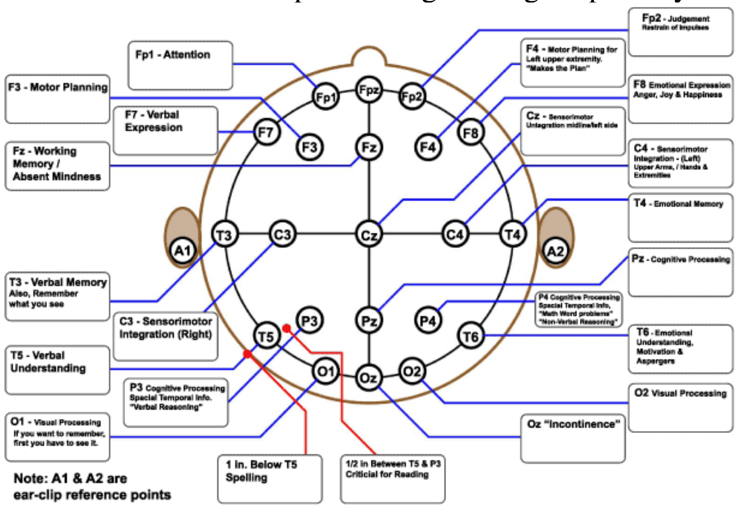
Figure 2.3: Brain divisions.



Source: (LIM et al., 2018)

This division is important to consolidate a standardized system for EEG electrode placement, so the International Federation of Societies for Electroencephalography and Clinical Neurophysiology created the 10-20 system (THE. . . , 1961), defining the standard for positioning the electrodes. Figure 2.4 shows the placement system.

Figure 2.4: 10-20 electrode positioning and region primary function.


Fp1 - Attention
F3 - Motor Planning
F7 - Verbal Expression
Fz - Working Memory / Absent Mindness
Fp2 - Judgement Restrain of Impulses
F4 - Motor Planning for Left upper extremity. "Makes the Plan"
F8 Emotional Expression Anger, Joy & Happiness
Cz - Sensorimotor Untegration midline/left side
C4 - Sensorimotor Integration - (Left) Upper Arms, / Hands & Extremities
T4 - Emotional Memory
Pz - Cognitive Processing
P4 Cognitive Processing Spacial Temporal Info. "Math Word problems" "Non-Verbal Reasoning"
T6 - Emotional Understanding, Motivation & Aspergers
O2 Visual Processing
Oz "Incontinence"
T3 - Verbal Memory Also, Remember what you see
C3 - Sensorimotor Integration (Right)
T5 - Verbal Understanding
P3 Cognitive Processing Spacial Temporal Info. "Verbal Reasoning"
O1 - Visual Processing If you want to remember, first you have to see it.
1 in. Below T5 Spelling
1/2 in Between T5 & P3 Criticial for Reading
Note: A1 & A2 are ear-clip reference points
Fp1
Fpz
Fp2
F7
F3
Fz
F4
F8
A1
T3
C3
Cz
C4
T4
A2
T5
P3
Pz
P4
T6
O1
Oz
O2

Source: <http://www.edmontonneurotherapy.com/>

From its conception, EEG was thought of as a neurological and psychiatric diagnostic tool, with numerous applications in health and medicine areas. These areas can be used in surgical interventions, diagnosis of neurophysiological disorders, psychology, and neuroscience. For being non-invasive, portable, and relatively cheap, EEG presents a good fit for being used, including in mass scale, as an interface with computers - this specific use case is known as Brain-Computer Interface(BCI).

2.3 Brain-Computer Interface

Humans have been seeking new ways to communicate with machines, and one of these is called Brain-Computer Interfaces(BCI). This subject is particularly interested in impaired people, who can benefit from BCI devices by controlling prostheses or communicating, for example, and neuroscientists interested in leveraging the computational advances to expand this area of knowledge.

BCI applications can range from medical to consumer entertainment, particularly controlling robotic prostheses to help motor-impaired people. It has historically been out of reach due to its complexity by several factors, like limited resolution and reliability, high information variability, real-time systems pricing, or simply because the technology did not exist yet (Prashant; Joshi; Gandhi, 2015).

The last 30 years of evolution in hardware, leading to price reduction; and in software, leading to increased availability, has made a significant impact in the BCI research community. In recent times, machine learning evolved and became very accessible to lots of research areas, including BCI, making deep learning an up-and-coming tool for BCI systems.

BCI enables the execution of a command in a machine by recording brain signals from specific tasks and then training classifiers to identify these commands. These five main steps make a typical BCI system structure:

1. **signal acquisition**: signals can be recorded from the brain using many different ways. This work will focus on EEG.
2. **signal processing**: apply filters to increase signal-to-noise ratio.
3. **feature extraction**: reduction from a high dimensional space to a feature space aiming to improve the classification (Prashant; Joshi; Gandhi, 2015).
4. **classification**: a feature vector is the input to a classifier to identify a mental state.
5. **feedback application**: the user receives feedback on how well the task is being performed.

In summary, for this work, BCI can be seen as a pattern recognition system whose specific job is to classify signals into mental states (Prashant; Joshi; Gandhi, 2015).

2.4 Time series

In studying a phenomenon, we often encounter data sets where the observations are captured over time. This time-ordered sequence of observations is called a time series, and its fundamental characteristic is that its observations are correlated (LITTLE, 2013). A time series can have one or more variables over time, called univariate and multivariate time series, respectively. EEG data sets usually have recordings of multiple channels. Therefore, EEG recordings are multivariate time series.

2.4.1 Time series analysis techniques

Time series analysis has many possible applications like predicting stock prices, hourly energy demand, country census data, disease incidence, and time domain and frequency domain approaches (LITTLE, 2013). Fourier analysis (FOURIER, 1878) is a usual technique used in time and frequency domain analysis.

Time series could become images for visual analysis, and the most common way we see it is in the form of 2D charts, where, usually, the horizontal axis represents time, while the vertical axis represents the observed value on a specific time point. With the advances and results presented in the latest years in machine learning and computer vision research, these tools became a well-suited tool for time series analysis and forecasting, offering acceptable accuracy in real and artificial data sets in a variety of domains (GONZALEZ; BARONE, 2018).

2.4.2 Gramian Angular Field (GAF)

To generate the images from time series, one of the methods we propose is Gramian Angular Field (WANG; OATES, 2015). GAF was used in data sets from different domains presenting highly competitive classification performance. Since time series correlate in time, we need a method that preserves the temporal dependency to explore it. GAF does so by encoding the time into the geometry of the matrix, from top-left to bottom-right - this diagonal also preserves the original time series values so one can also explore a reconstruction of the original time series from generated images.

Given a time series $X = x_1, x_2, ..., x_n$, we normalize it by linearly rescaling X to fit all values in the interval $[-1, 1]$, creating $\tilde{X}$.

GAF uses a polar coordinate system to represent time series to preserve absolute temporal relations by exploring the trigonometric sum between two angles generated from values at different times. The polar angle is defined as:

$$\theta_i = arccos(\tilde{x_i}) \tag{2.1}$$

$$r_i = \frac{t_i}{normalization\ value} \tag{2.2}$$

A Gram matrix is a matrix where each element is an inner product between two vectors. So a Gram matrix of the new rescaled time series $\tilde{X}$ can be defined as:

$$G = \begin{bmatrix} < \tilde{x}_1, \tilde{x}_1 > & \dots & < \tilde{x}_1, \tilde{x}_1 > \\ < \tilde{x}_2, \tilde{x}_1 > & \dots & < \tilde{x}_2, \tilde{x}_n > \\ \vdots & \ddots & \dots \\ < \tilde{x}_1, \tilde{x}_1 > & \dots & < \tilde{x}_n, \tilde{x}_n > \end{bmatrix} \tag{2.3}$$

By defining *I* as the unit row vector $[1, 1, ..., 1]$, and the inner product as 2.4, the final Gram matrix used to generate the image becomes 2.5.

$$< x, y >= x.y - \sqrt{1-x^2}.\sqrt{1-y^2} \tag{2.4}$$

$$GAFImage = \begin{bmatrix} cos(\phi_1 + \phi_1) & \dots & cos(\phi_1 + \phi_n) \\ cos(\phi_2 + \phi_1) & \dots & cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \dots \\ cos(\phi_n + \phi_1) & \dots & cos(\phi_n + \phi_n) \end{bmatrix} \tag{2.5}$$

$$GAFImage = \tilde{X^T}.\tilde{X} - \sqrt{I - \tilde{X^2}}^T.\sqrt{I - \tilde{X^2}} \tag{2.6}$$

Since it uses a matrix, GAF will increase the problem dimension from N samples to N^2, so to reduce the final size (WANG; OATES, 2015) proposes the usage of Piecewise Aggregation Approximation(PAA). This method is only applicable for univariate time series, so for multivariate time series, multiple GAF images can be stacked vertically as (YANG CHEN-YI YANG, 2019).

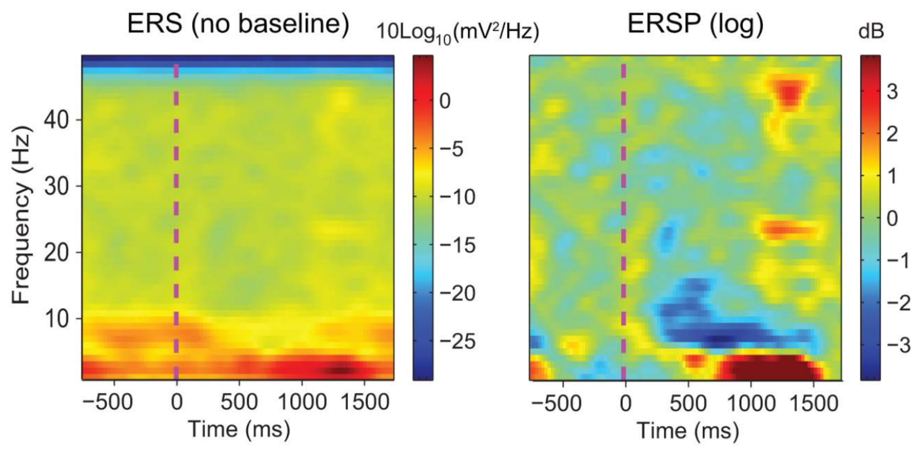
2.4.3 Event-Related Spectral Perturbation (ERSP)

When modeling event-related responses in an EEG experiment, amplitude and phase effects may be considered separately or in combination (MAKEIG, 1993).

The time course of event-related attenuation in alpha-band EEG was first quantified in (PFURTSCHELLER; ARANIBAR, 1977) and called event-related desynchronization (ERD). In contrast, the opposite phenomenon, event-related synchronization (ERS), describes the phasic and regional increase of brain activity (PFURTSCHELLER, 1992).

The Event-Related Spectral Perturbation can be viewed as a generalization of the ERD/ERS patterns (MAKEIG, 1993). First, a windowed baseline is calculated by averaging out trials and frequency bands. Then this average is subtracted from the original event-related spectrum. Figure 2.5 shows how we can see variations way more clearly in the ERSP image when compared to the absolute spectrum.

Figure 2.5: Raw event-related spectrum (absolute log-ERS) on the left versus baseline corrected ERSP (log-ERSP) on the right for scalp EEG data trials.



Source (GRANDCHAMP; DELORME, 2011)

2.5 Machine Learning

Historically, software tools have been developed to perform specific tasks imperatively, specifying the steps to complete a particular task, similar to a recipe. Machine Learning (ML) brings the concept of building software to create models that can change their performance over time by being exposed to data. A brief and more precise definition of Machine Learning(ML) by (MITCHELL, 1997) is:

> A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.

2.5.1 Learning methods

In machine learning, there are three main learning methods when categorizing by the task feedback type: *supervised learning*, *unsupervised learning* and *reinforcement learning*.

1. **Supervised learning**: algorithms work with full feedback, relying on input data that has the expected output labeled (RUSSELL; NORVIG, 2009), like classification and regression.
2. **Unsupervised learning**: algorithms have no feedback; it groups data and find relations between attributes by learning the data features. It is suited for descriptive tasks and exploratory analysis to investigate connections and associations between data instances by looking for data set intrinsic properties.
3. **Reinforcement learning**: algorithms get partial feedback about the action taken towards the task goal, as a positive or negative reward, for example.

2.5.2 Deep Learning

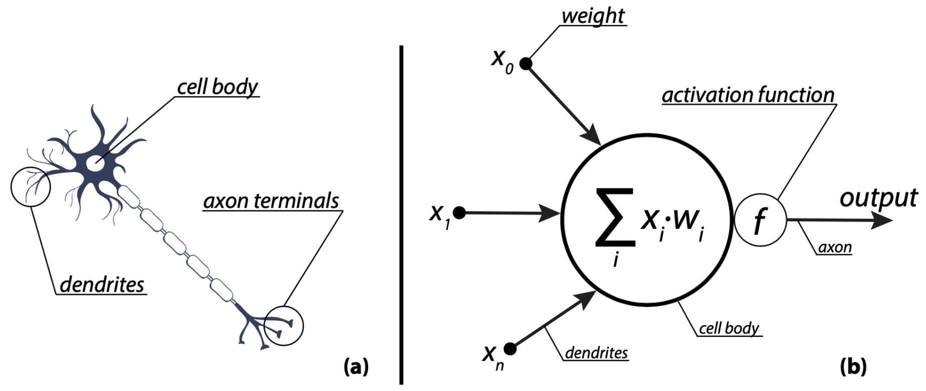
A Neural Network(NN) is a data processing structure widely used and heavily inspired in the inner working of the brain's neurons. Each node of the network represents a neuron, and the connections between the nodes represent the axons. Each node can connect or receive connections from many other nodes, assembling the network. The basic mathematical model for a neuron used in an Artificial Neural Network(ANN) is composed of:

1. **input values**: can be represented as the multiple dendrites the biological neuron has
2. **multiply and accumulate(MAC)**: the neuron cell body encapsulates this idea
3. **activation function**: non-linear function to map the MAC to an output value
4. **output value**: the output is sent through the axon

This model is illustrated in Figure 2.6 as follows:

The concept of deep learning comes from a neural network having multiple layers and can classify and extract features due to its dense architecture.

Figure 2.6: A neuron *(a)* and it's basic mathematical model(b).



Adapted from (SANTANA, 2019).

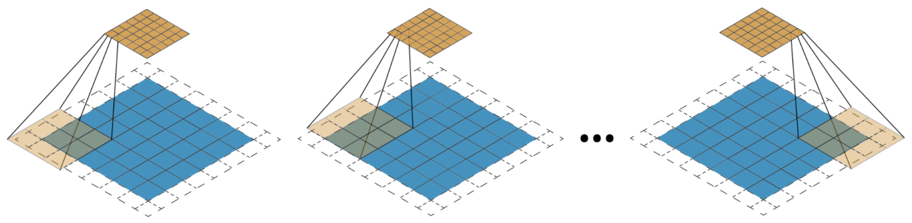
2.5.3 CNN

While identifying objects can be a simple task for a human, writing image recognition code is not a simple task. Image recognition software is hard to create due to countless possibilities in the high dimensional space of an image like color channels, shapes, different view angles. An example of a classical image recognition problem is to classify handwritten digits using Convolutional Neural Networks(CNN), explicitly designed to handle 2D shapes, as explored by (LECUN Y., 1998) with outstanding results outperforming all other techniques at the time (1998).

The convolution operation consists in a *filter*(or *kernel*) matrix, sliding *stride* number of rows and columns, over another matrix, considering a *padding* number of extra rows and columns added to the original input, applying a mathematical operation on each matrix region, mapping the initial matrix to an *activation map*.

CNNs are heavily used on image recognition tasks, mainly due to the characteristics of the convolution operation. The convolution operation makes feature extraction happens naturally by using a filter that can identify increasingly complex patterns. In a multi-layered architecture, the first convolution layer can identify edges. The second can identify textures, and a third may recognize objects, and so on. The benchmark for CNNs has been the ImageNet (RUSSAKOVSKY et al., 2015) competition, which has given CNNs first place every year since the breakthrough results presented by AlexNet in 2012.

Figure 2.7: A convolutional layer full pass using a 3x3 filter, stride = 1, padding = 1, on a 6x6 input matrix. The resulting activation map is a 6x6 matrix.



Adapted from (SANTANA, 2019).

3 RELATED WORK

This section describes some of the related work around the basics of this work proposal: BCI, time series analysis, CNN for time series, and CNN for BCI.

3.1 BCI

BCI systems can has been used in many applications from a variety of domain areas. Smart home environments controlled by physiological human state (LIN et al., 2014) relying on Mahalanobis distance from an alert mental state reference signal as classification method; insight extraction from TV commercials (VECCHIATO et al., 2009); and even as a game controller in a game to reduce stress (HJELM; BROWALL, 2000) using closed source software to classify and interpret the received signals, to name a few. We can notice that even though all data sets used in the previous works cited are time series they all had different approaches on how to work with it. Since at least 2007, the classification step on BCI systems have been using and exploring Machine Learning techniques showing and are currently the state of the art (LOTTE et al., 2007; LOTTE et al., 2018).

From a market perspective, as of 2021, Emotiv, Nuerosky, and Neuralink, are three companies having BCI as its core business:

> Emotiv was founded in 2011 by tech entrepreneurs Tan Le (CEO) and Dr. Geoff Mackellar (CTO), the company is headquartered in San Francisco, U.S.A. with facilities in Sydney, Hanoi and Ho Chi Minh.
>
> Founded in 2004, NeuroSky is a privately held, Silicon Valley-based company with offices throughout Asia and Europe.
>
> Neuralink is a team of exceptionally talented people. We are creating the future of brain interfaces: building devices now that will help people with paralysis and inventing new technologies that will expand our abilities, our community, and our world.

While Neurosky and Emotiv focus is portable, non-invasive, and head-sized devices, Neuralink's solution focuses on an implanted device, bringing it to 23mm x 8mm, while having 1024 electrodes.

3.2 Time series analysis

Time series analysis and forecasting can explore the time, and frequency domain approach with Fourier Analysis (FOURIER, 1878) for example. Multiple knowledge areas have time series as a data structure that can be useful in extracting experiments' results, like characterizing the principal neocortical cells by analyzing network interactions that have time correlation (BARTHó et al., 2004).

With all the advances in computer vision in recent years, novel frameworks and architectures for time series, especially multivariate, feature extraction and classification using deep learning techniques, were proposed, some of these being Multi-Layer Perceptron(MLP), Fully Convolutional Neural Network(FCN), Residual Network(ResNet), Encoder, Multi-scale Convolutional Neural Network(MCNN) and Multi-Channel Deep Convolution Neural Network(MCDCNN) (FAWAZ et al., 2019).

3.3 CNN for time series

FCNN and ResNet achieve state-of-the-art performance in time series classification (FAWAZ et al., 2019). Considering that time series can be multivariate, an adaptation can be made on the CNN architecture side, or the data set side (HATAMI TANN GAVET, 2017). On the CNN architecture, it is possible to modify the CNN architecture to receive 1D signals. On the other hand, on the data side, it is possible to convert signals from 1D to 2D and then use traditional CNN, which is the approach this work proposes by converting raw signals into 2D images.

3.4 CNN for BCI

Neural Networks have been a powerful tool in BCI studies (GONZALEZ et al., 2017). The usage of CNN in raw EEG signal, without domain-specific pre-processing, has shown state-of-the-art performance for motor imagery (MI) classification (Dose et al., 2018), making CNN a potential technique for further exploration seeking higher performance metrics.

CNN has been proved to be a robust machine learning architecture to solve a variety of problems, as well as in BCI systems, not only with raw signal data but also using 2D EEG images for signal classification, outperforming traditional methods (Gao et al., 2020).

4 MATERIALS AND METHODS

This work proposes to build software to export images to train machine learning models for BCI systems. The software can separate images into its respective class folder, a well-known pattern for organizing images in data sets for supervised training and allowing users to configure a set of parameters better to fit the images to their needs and exploration processes.

4.1 Data sets

For the experiments, this work used the existing BCI Competition IV 2008 Graz 2B data set (LEEB C. BRUNNER; PFURTSCHELLER, 2008) since there are classification performances available in the literature to compare to our classification methods. The following subsections explain more about the data set.

4.1.1 Description

The BCI competition IV 2B data set contains data from 9 right-handed subjects performing a motor imagery task. Each subject had 5 data collection sessions, 3 for training and 2 for evaluation, each with several runs as in figures 4.1 and 4.2.

Motor imagery EEG recording experiments can vary slightly depending on a number of factors, but are usually conducted as follows:

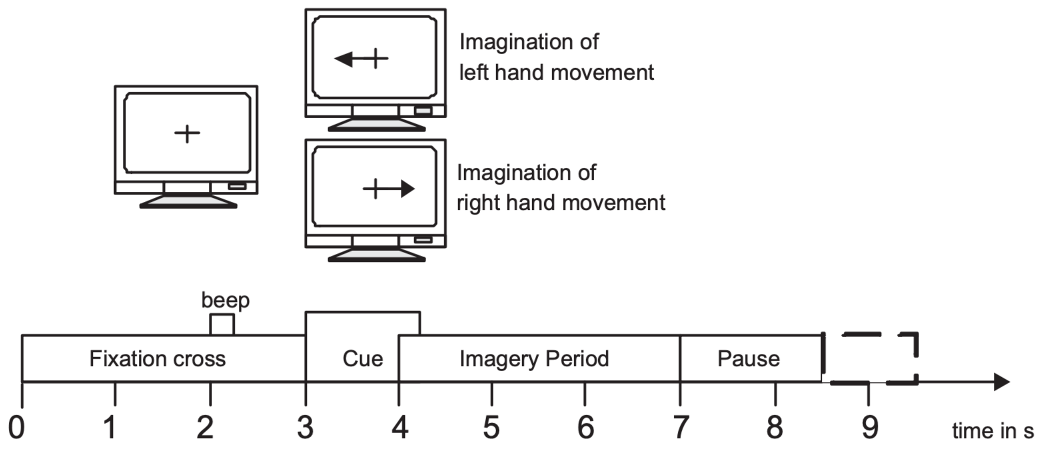
- Some interval to "reset" mental state
- A cue is presented to the subject for a period of time
- The subject performs the task indicated by the presented cue

Figures 4.1 and 4.2 presents the 2 schemes used in this data set for motor imagery recording sessions:

From this scheme, we can identify that it may be interesting to have pre and post-event time padding and event type selection to be a configurable setting - a researcher could, for example, use the "Pause" event to use as baseline brain activity; or add extra padding to an event to mitigate signal processing harsh border effects.

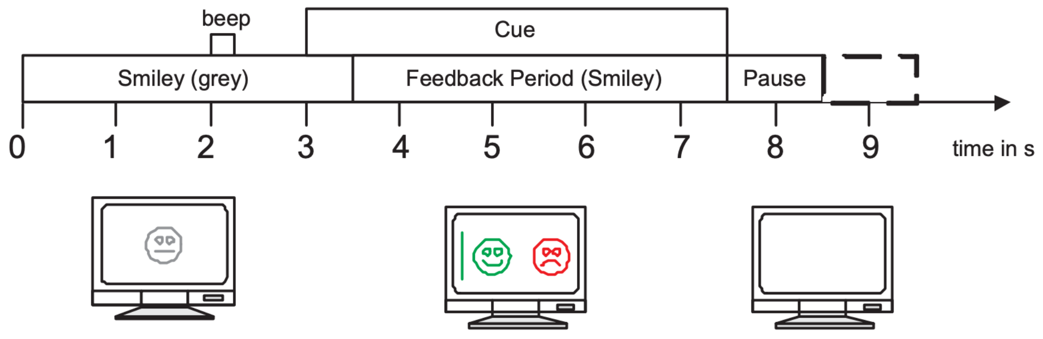
It is essential to highlight that during development was noticed that this scheme's

Figure 4.1: BCI competition 2b motor imagery task data set screening scheme.



Adapted from (LEEB C. BRUNNER; PFURTSCHELLER, 2008).

Figure 4.2: BCI competition 2b motor imagery task data set smiley feedback scheme.



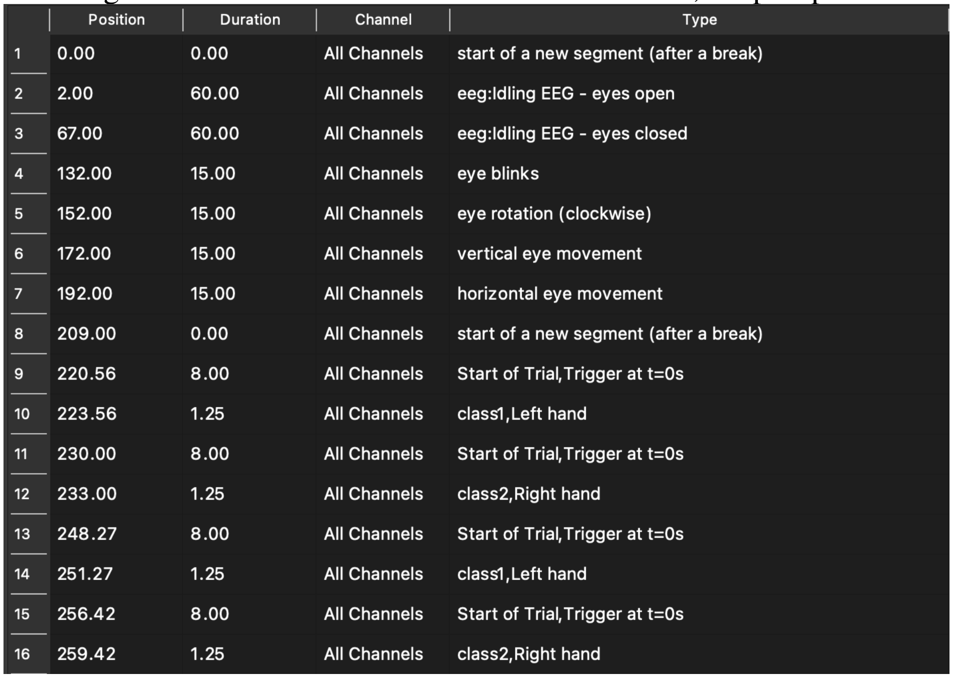
Adapted from (LEEB C. BRUNNER; PFURTSCHELLER, 2008).

data set does not provide events for all the boxes demonstrated in their scheme. For this reason, the **cue start** had to be used as a reference point and calculate the time sections to be processed according to the duration set up by the end-user of the software. Likewise, since we can only provide one reference point and duration settings, only data from the screening scheme was used in experiments, so files from smiley feedback sessions ending with *03T.gdf* were ignored, as well as the evaluation ones ending with *04E.gdf* and *05E.gdf*, since these do not have classification data.

4.1.2 Files

The BCI data sets found in our research have a few different file types to store EEG information. Besides the recorded EEG data, these files usually contain the list of channels available in the BCI device used to collect the data, event types to identify each type of task and some other control events, markers tracking when events occurred, and

Figure 4.3: No events to reference fixation cross, beep or pause



	Position	Duration	Channel	Type
1	0.00	0.00	All Channels	start of a new segment (after a break)
2	2.00	60.00	All Channels	eeg:Idling EEG - eyes open
3	67.00	60.00	All Channels	eeg:Idling EEG - eyes closed
4	132.00	15.00	All Channels	eye blinks
5	152.00	15.00	All Channels	eye rotation (clockwise)
6	172.00	15.00	All Channels	vertical eye movement
7	192.00	15.00	All Channels	horizontal eye movement
8	209.00	0.00	All Channels	start of a new segment (after a break)
9	220.56	8.00	All Channels	Start of Trial,Trigger at t=0s
10	223.56	1.25	All Channels	class1,Left hand
11	230.00	8.00	All Channels	Start of Trial,Trigger at t=0s
12	233.00	1.25	All Channels	class2,Right hand
13	248.27	8.00	All Channels	Start of Trial,Trigger at t=0s
14	251.27	1.25	All Channels	class1,Left hand
15	256.42	8.00	All Channels	Start of Trial,Trigger at t=0s
16	259.42	1.25	All Channels	class2,Right hand

sigviewer software listing events from B0101T.gdf

some other types of data as described on each data set.

Some of the files found are *.gdf, .edf and .xdf* binary files and *.csv* ASCII files. We highlight the binary GDF (SCHLöGL, 2013) and EDF/EDF+ (KEMP et al., 1992; KEMP; OLIVAN, 2003) files for being specific for biosignals. While XDF is not specific for biosignals, it is intended for scientific use, and it is used in some EEG data sets.

Our software loops through all the current events in these specific files to generate event images considering the chosen settings for the image generation method and pre and post-event time padding. More details about how the software works can be found in Section 5.1.

4.2 Hardware

The application was developed and tested using an Apple MacBook Pro (15-inch, 2018), 2.6 GHz 6-Core Intel Core i7-8850H CPU, 16 GB 2400 MHz DDR4 RAM, and a Radeon Pro 560X 4 GB graphics card, running macOS 11.5.2 (20G95). Also, Google Colab Pro was used for training the machine learning models.

4.3 Python

Python (ROSSUM; DRAKE, 2009) is the 4th most popular technology and the most popular programming language, according to (Stack Overflow, 2020). It has a lot of documentation and resources and a vast community, especially around data science and machine learning. For theses reasons this work is implemented using Python.

4.4 Keras

Several libraries and frameworks can be used for training machine learning models such as PyTorch (PASZKE et al., 2019), Keras (CHOLLET et al., 2015), and Scikit-learn (PEDREGOSA et al., 2011). Keras uses Tensorflow as a backend in a Google Colab notebook to train the classification models for the GAF and ERSP images generated with TS2Image. Their creators describe it as follows:

> Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Both GAF and ERSP models were based on VGG model (SIMONYAN; ZISSERMAN, 2014) and used transfer learning. The entire Google Colab notebooks for GAF and ERSP are publicly available. The base code for models creation and training is presented in listing 4.1:

Listing 4.1 – Base code for models creation

```python
import tensorflow as tf
from keras import backend as K

K.set_learning_phase(1)
width = 256
height = 256
channels = 3
epochs =  10
batch_size = 32
n_classes = 3
class_type = "categorical"
model = 'VGG'
```

```
train_generator, validation_generator = train_validation_data(
    data_base,
    model,
    height,
    width,
    batch_size,
    class_type,
    val_split=0.2
)
pre_trained_model = model_TL(model, height, width, channels)

history = training_model(pre_trained_model,
                         epochs=epochs,
                         batch_size=batch_size,
                         learning_rate=0.0001,
                         training_data=train_generator,
                         validating_data=validation_generator,
                         patience=5,
                         stage='train',
                         model_name=model)

test_lost, test_acc = pre_trained_model.evaluate_generator(
   validation_generator)

predict_label = pre_trained_model.predict(validation_generator,
   batch_size)
```

4.5 MNE

MNE 0.22.0 (GRAMFORT et al., 2013) implements reading GDF, EDF+ and some other file types commonly used for neurophysiological data. It also has useful methods for processing the data, which will generate GAF and ERSP images. MNE is described as:

> Open-source Python package for exploring, visualizing, and analyzing human neurophysiological data: MEG, EEG, sEEG, ECoG, NIRS, and more.

4.6 pyts

pyts version 0.11.0 is used to generate GAF matrices, and it is describe by (FAOUZI; JANATI, 2020) as:

> **pyts** is a Python package dedicated to time series classification. It aims to make time series classification easily accessible by providing preprocessing and utility tools, and implementations of several time series classification algorithms. The package comes up with many unit tests and continuous integration ensures new code integration and backward compatibility. The package is distributed under the 3-clause BSD license.

5 RESULTS AND ANALYSIS

This chapter presents the final software product, generated images examples, and an analysis of these results. Initially, Section 5.1 describes the software created, Section 5.2 presents the results obtained for GAF images, while Section 5.3 presents the results obtained for ERSP images. Finally, Section 5.4 shows the deep learning results obtained from the generated images.

The proposed pipeline using the software created, leveraging its re-usability to multiple data sets and ease of use, saved a fair amount of time since there was no need to create custom processing tools for different data sets. It was just a matter of setting up a few parameters and running a single command. The domain experts can then use this time to focus on the task at hand rather than committing to a software development iteration cycle and eventual maintainability. Moreover, since software created is publicly available in TS2Image GitHub repository, anyone has full access to it and can propose new features, improvements, and bug fixes. Having the source code in a public repository makes worldwide collaboration seamless and probable.

The final generic pipeline demonstrated, for any time series application or use case and focusing on reducing friction when generating images to explore CNN classification capabilities, can then be summarized as follows:

1. Signal aquisition as GDF files
2. Optional signal pre processing
3. Use GDF files as input into TS2Image
4. Train CNN using GAF or ERSP images generated

5.1 TS2Image

TS2Image was created using the technologies described in chapter 4. Its implementation served as a proof of concept to automate the image generation process for CNN training and make this framework more accessible to researchers without a computer science background.

The usage of the software has been proved valuable and stable during testing. Exploring multiple files and data sets became as simple as configuring the software parameters to match each specific data set.

To those interested, the complete source code can be accessed in <https://github.com/hvsw/TS2Image>. The *README.md* file contains all information needed to set up and run the software. The source files have plenty of comments explaining how methods, parameters, and specific parts of the code work, to help users understand more about TS2Image.

The following subsections will present the setup and how to use it and some implementation details to those looking for expanding this work.

5.1.1 Setup and usage

The end-user entry point is *main.py*. This file was created to make it easier for researchers with no computer science background not to dive into how the main components of the software work, making the execution just a single command line in terminal: *python main.py (GAF|ERSP)*. To those with a computer science background, the exploration of the source is advised. Some features are not exposed through public APIs, like vertically stacking multiple channels into a single image for classification, similar to (YANG CHEN-YI YANG, 2019), and enabling the generation of both stacked and individual images, aiming to reduce iteration time when processing data sets so researchers can focus on optimizing models and experimenting with it.

TS2Image allow users to configure parameters to process the GDF recording files. These are the customizable parameters currently available for user configuration in *main.py* and used in *ts2i.generate_images(...)* invocation:

- **input_folder**: folder containing files to be processed
- **output_folder**: folder used by TS2Image to save image files
- **valid_events_descriptions**: events to generate images for
- **events_dictionary**: a dictionary specifying the all the data set events' identifier and description. This is optional and only needed by GAF class to get the description to create folders to save the generate images. This is not used by ERSP class.
- **t_start**: value summed to event start time to crop the data to generate the image
- **duration**: value summed to event start time and *t_start* to crop the data to generate the image

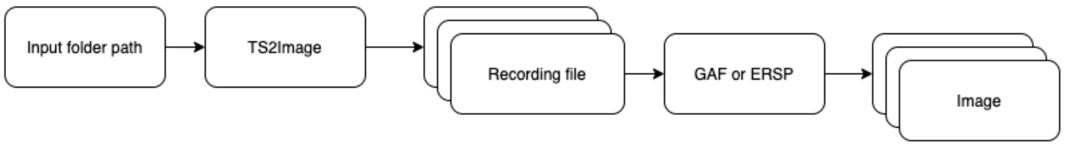
By setting up these parameters, TS2Image can process the GDF files specified and output the images accordingly.

5.1.2 Implementation details

This section is intended to share some high level implementation details to guide those interested in expanding TS2Image.

The software entry point is **TS2Image** class, with its constructor and **generate_images** method. This class holds logic to list files from an input folder path, iterate over this list of files, instantiates the proper class, **GAF** or **ERSP**, and then ask the instance to generate the images. An execution flow is presented in figure 5.1.

Figure 5.1: Execution flow and its main components



Source: The author

The **GAF** and **ERSP** classes were divided in a way to separate its responsibilities. It helped development speed in the early stages of the project, and due to the nature of the project, in which we iterate over files' events, it also caused some code duplication because each class implements its approach to go through events and can be improved.

This is a high level overview of the algorithm implemented in **TS2Image** class, and part of what is presented in execution flow in figure 5.1:

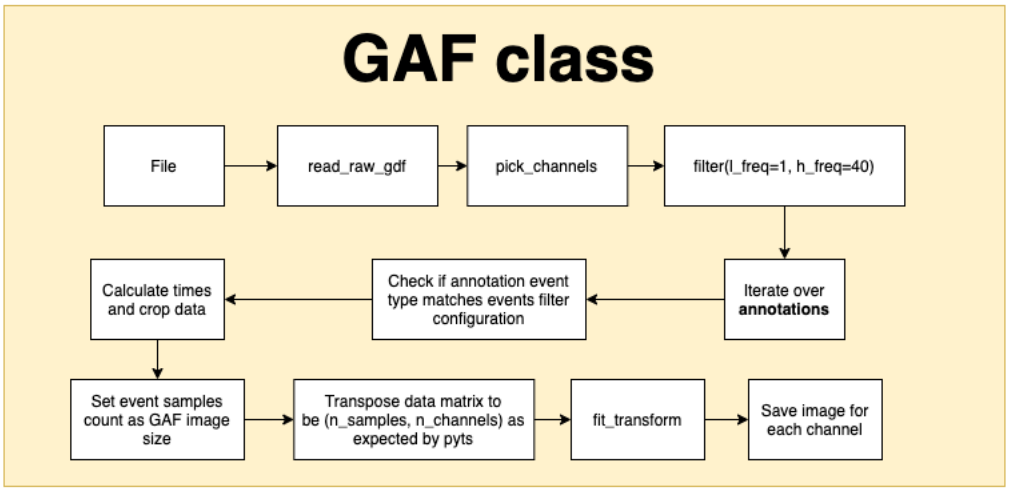
1. Get a list of files from input folder and iterate over it
2. Create GAF or ERSP instance according to **method** parameter for each file
3. Call **generate_images** method in the created instance

This is a high level overview on how **GAF** and **ERSP** classes work:

1. Band-pass filter, currently configured from 1 to 40Hz
2. List events for each file
3. List channels for each event
4. Generate image for each specific event channel

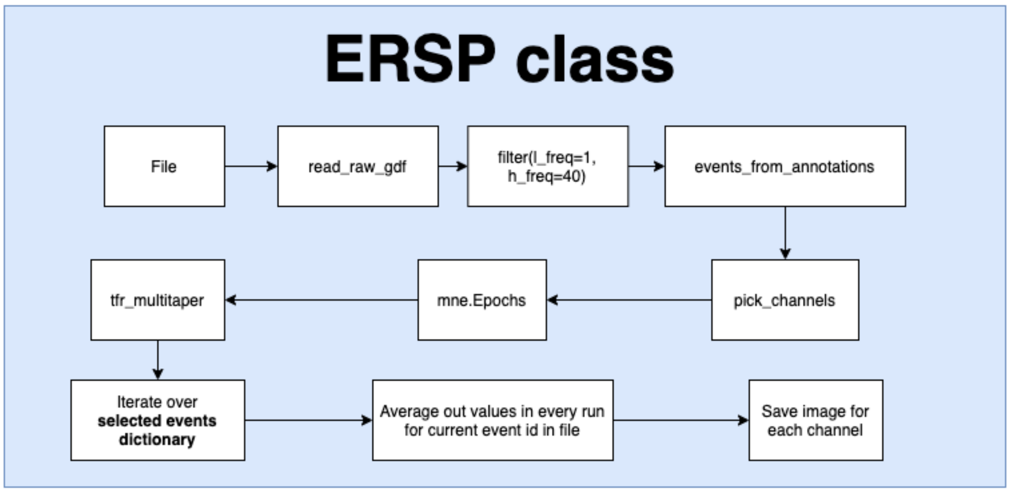
Figures 5.2 and 5.3 show some of the inner workings of GAF and ERSP classes, respectively.

Figure 5.2: GAF class image generation flow


GAF class
File
read_raw_gdf
pick_channels
filter(l_freq=1, h_freq=40)
Iterate over **annotations**
Check if annotation event type matches events filter configuration
Calculate times and crop data
Set event samples count as GAF image size
Transpose data matrix to be (n_samples, n_channels) as expected by pyts
fit_transform
Save image for each channel

Source: The author

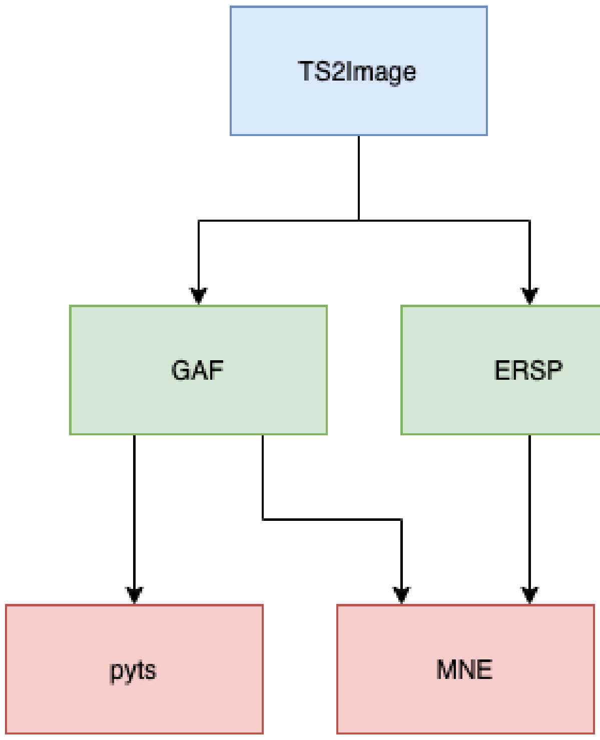
Figure 5.3: ERSP class image generation flow


ERSP class
File
read_raw_gdf
filter(l_freq=1, h_freq=40)
events_from_annotations
pick_channels
mne.Epochs
tfr_multitaper
Iterate over **selected events dictionary**
Average out values in every run for current event id in file
Save image for each channel

Source: The author

As mentioned in chapter 4, TS2Image uses third party libraries; figure 5.4 shows how the dependencies are organized in the source.

Figure 5.4: TS2Image dependency graph.



TS2Image class depends on both **GAF** and **ERSP** classes, which depends on *pyts* and *mne* libraries. Source: The author

5.2 GAF

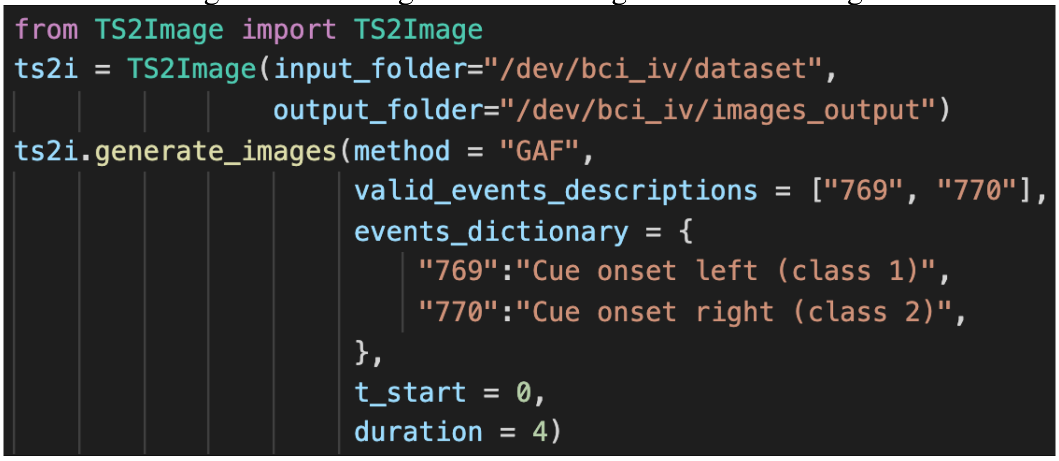
The GAF class is responsible for generating GAF images as described in section 2.4.2. More information about the arguments and how to use them can be found in source code comments. The public API is defined by its constructor and a method to generate the images, as follows:

```
gaf = GAF(file_path: str,
          valid_events_descriptions: list,
          cue_map: dict)
gaf.generate_images(output_folder: str,
                    t_start: float, duration: float,
                    generate_intermediate_images: bool,
                    generate_difference_images: bool,
                    desired_channels: list,
                    merge_channels: bool)
```

Figure 5.5 shows an example of configuration used to generate GAF images from

BCI competition IV 2b data set (LEEB C. BRUNNER; PFURTSCHELLER, 2008):

Figure 5.5: Configuration used to generate GAF images



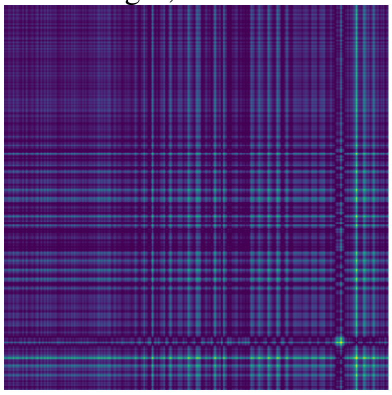
```python
from TS2Image import TS2Image
ts2i = TS2Image(input_folder="/dev/bci_iv/dataset",
                output_folder="/dev/bci_iv/images_output")
ts2i.generate_images(method = "GAF",
                     valid_events_descriptions = ["769", "770"],
                     events_dictionary = {
                         "769":"Cue onset left (class 1)",
                         "770":"Cue onset right (class 2)",
                     },
                     t_start = 0,
                     duration = 4)
```

Source: The author

5.2.1 Generated images samples

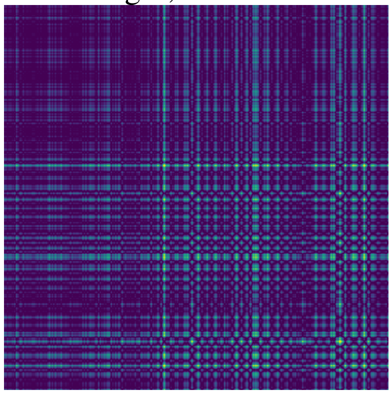
These are some example images of the first left and right-hand movements events, represented by 769 and 770, from the B0101T.gdf file, generated by using the configuration (Figure 5.5):

Figure 5.6: C3 channel from B0101T.gdf, first left hand event from t=223.56 to 224.81.



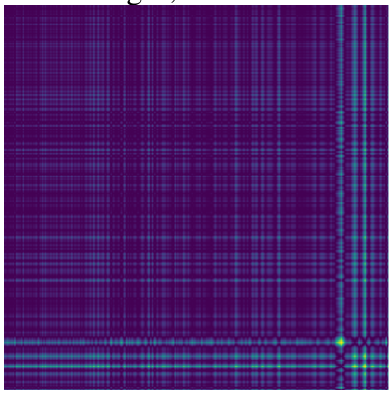
Source: The author

Figure 5.7: C4 channel from B0101T.gdf, first left hand event from t=223.56 to 224.81.



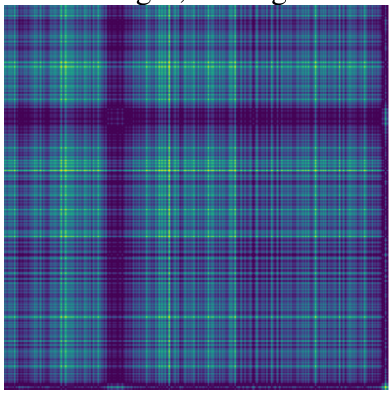
Source: The author

Figure 5.8: CZ channel from B0101T.gdf, first left hand event from t=223.56 to 224.81.



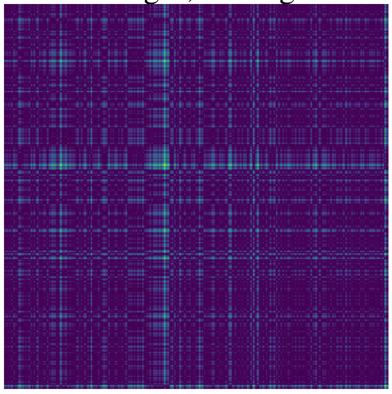
Source: TS2Image

Figure 5.9: C3 channel from B0101T.gdf, first right hand event from t=233 to 234.25.



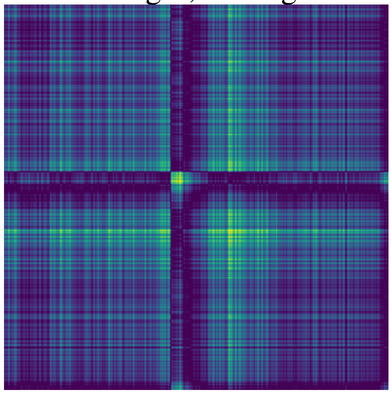
Source: The author

Figure 5.10: C4 channel from B0101T.gdf, first right hand event from t=233 to 234.25.



Source: The author

Figure 5.11: CZ channel from B0101T.gdf, first right hand event from t=233 to 234.25.



Source: The author

5.3 ERSP

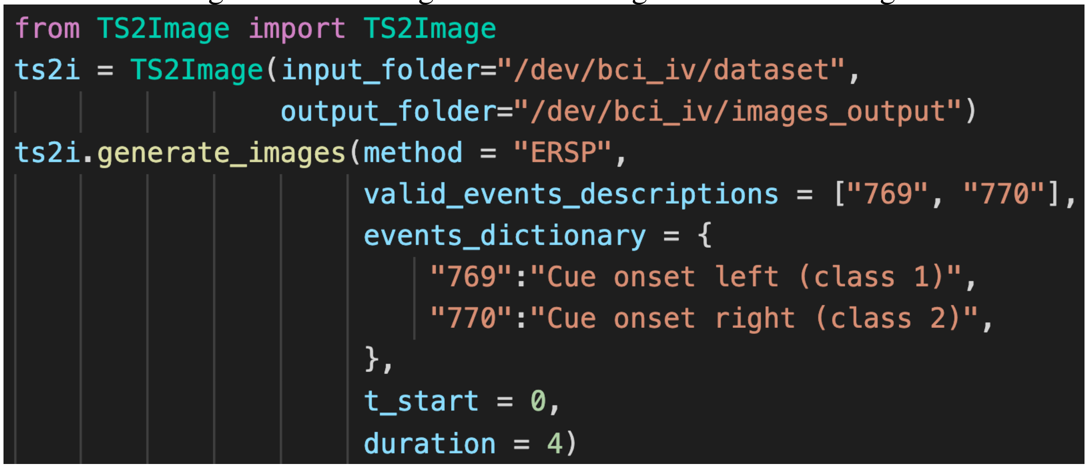
The ERSP class is responsible for generating ERSP images as described in section 2.4.3. Its public API is defined by its constructor and a method to generate the images, as follows:

```
ersp = ERSP(file_path: str)
ersp.generate_images(output_folder: str,
                     desired_events: list,
                     t_start: float,
                     t_end: float,
                     generate_intermediate_images: bool,
                     desired_channels: list,
                     merge_channels: bool)
```

5.3.1 Generated images samples

Using the configuration shown in figure 5.12, ERSP images 5.13, 5.14 and 5.15 were generate. Notice the only change, in comparison with generating GAF images as shown in figure 5.5, is the **method** parameter.

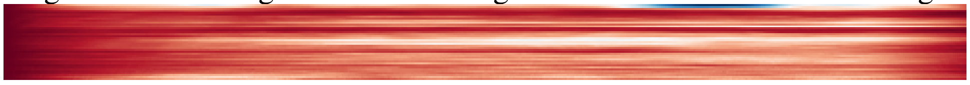
Figure 5.12: Configuration used to generate ERSP images



```
from TS2Image import TS2Image
ts2i = TS2Image(input_folder="/dev/bci_iv/dataset",
                output_folder="/dev/bci_iv/images_output")
ts2i.generate_images(method = "ERSP",
                     valid_events_descriptions = ["769", "770"],
                     events_dictionary = {
                         "769":"Cue onset left (class 1)",
                         "770":"Cue onset right (class 2)",
                     },
                     t_start = 0,
                     duration = 4)
```

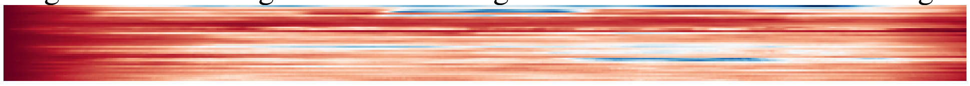
Source: The author

Figure 5.13: Averaged out ERSP image from channel C3 and B0101T.gdf



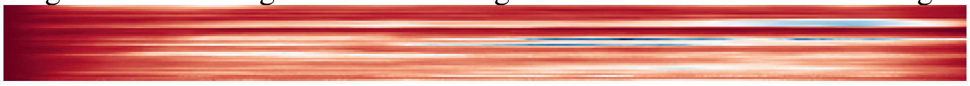
Source: The author

Figure 5.14: Averaged out ERSP image from channel C4 and B0101T.gdf



Source: The author

Figure 5.15: Averaged out ERSP image from channel Cz and B0101T.gdf



Source: The author

5.4 Classification

GAF and ERSP images were generated from the BCI competition data set 2B and used to train machine learning models and thus validate the pipeline. It is essential to highlight that achieving the state-of-the-art classification performance is not the goal of this work, yet our method, even having more classes, which is a known factor to degrade classification performance, did not rank the last.

Transfer learning was used with the VGG16 model (SIMONYAN; ZISSERMAN, 2014). In both cases, GAF and ERSP, ten epochs were used. Since ERSP images were averaged out, a lower number of images was available for training. Therefore a batch size of 64 was chosen, in comparison to 32 chosen for GAF images. The GAF and ERSP generated images sizes were, respectively, 256x256 and 256x39. The input layer needs to accommodate the three channels (red, green, and blue) since these are colored images. The complete source code is available in Google Colab notebooks for GAF and ERSP.

Table 5.1: Model characteristics per image type

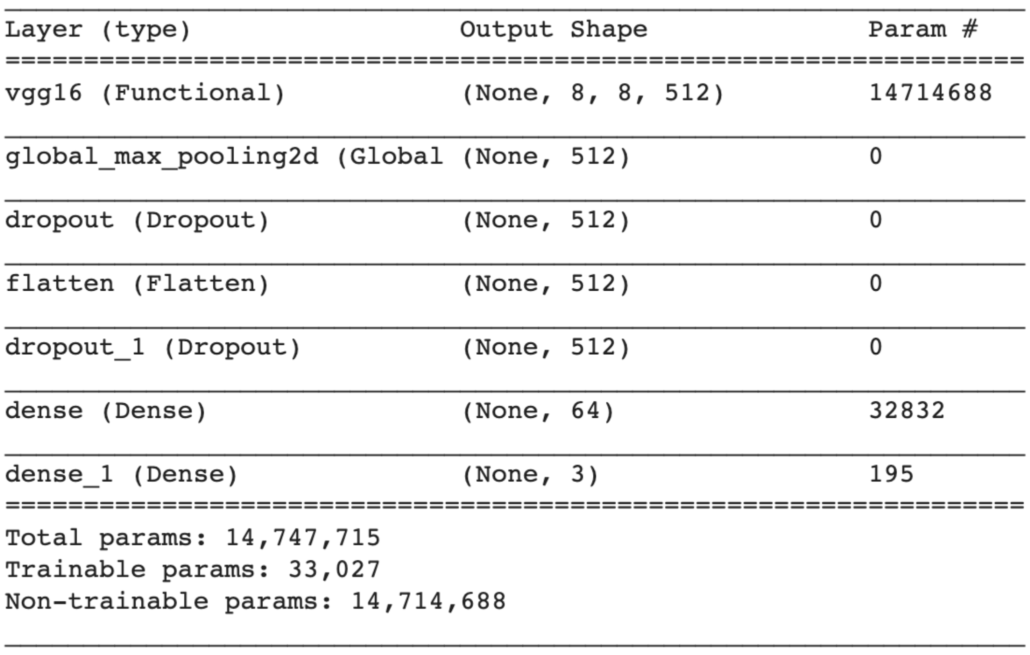
Image type	Input layer dimensions	Batch size
GAF	256 x 256 x 3	32
ERSP	256 x 39 x 3	64

Due to the lower number of ERSP images available a batch size of 64 was used

5.4.1 GAF

To classify GAF images, the following CNN architecture was used:

Figure 5.16: Model used to train with GAF images



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
global_max_pooling2d (Global (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 14,747,715
Trainable params: 33,027
Non-trainable params: 14,714,688
_________________________________________________________________
```

GAF with 3 classes: left and right hand, and pause. Source: The author.

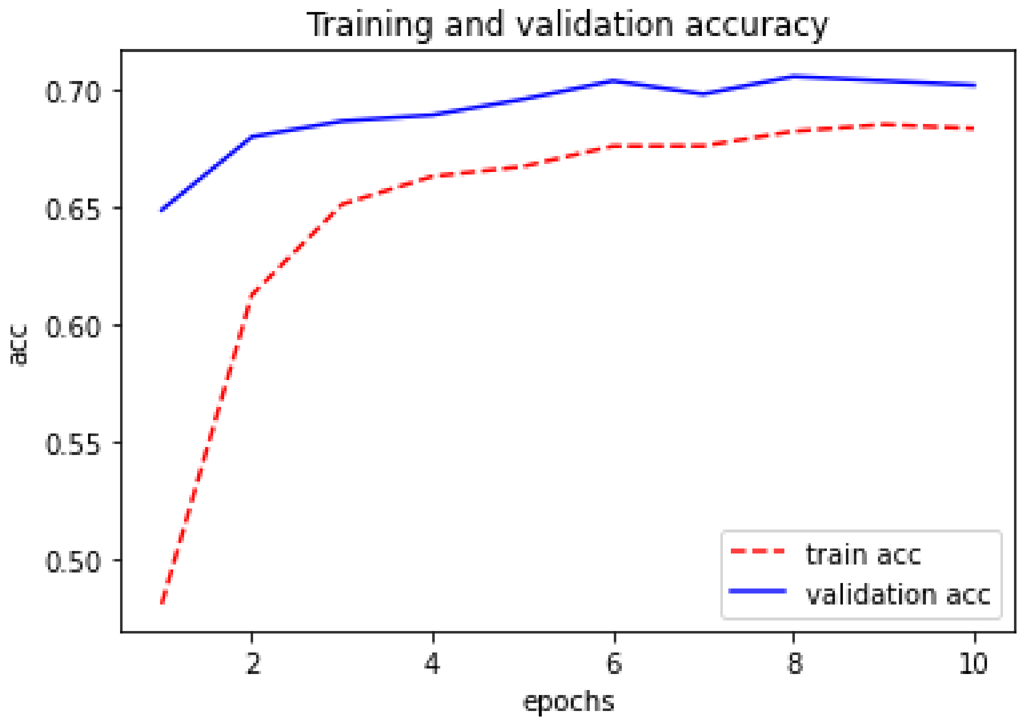
The classification performance obtained from GAF images converged to 70.15% accuracy and 59.02% loss. For reference, these are some state of the art classification methods:

Table 5.2: BCI Competition IV 2B data set classification performances

Method	Accuracy %	number of classes
(ALANSARI et al., 2018)	67.8	2
Ours	70.15	3
(WANG; FENG; LU, 2017)	81.2	2
(SILVA et al., 2017)	83.8	2

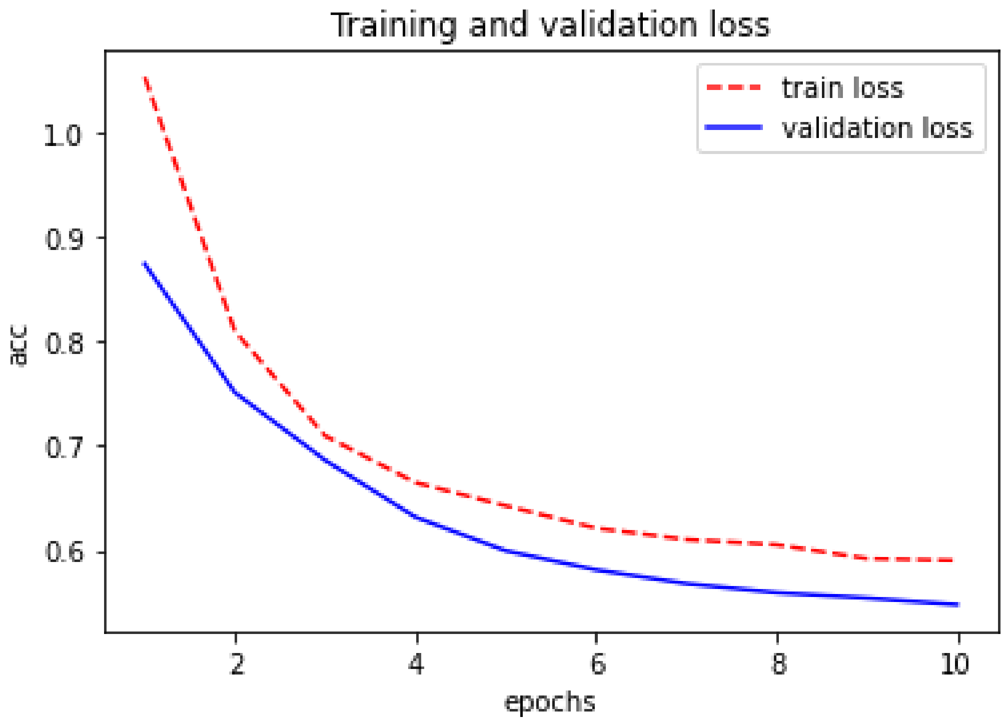
Even though this work is not focused on optimizing the model, and the used method is not the worst, there are room for improvement.

Figure 5.17: GAF training and validation accuracy evolution across epochs



Accuracy converging to final 70.15% accuracy result. Source: The Author.

Figure 5.18: GAF training and validation loss evolution across epochs

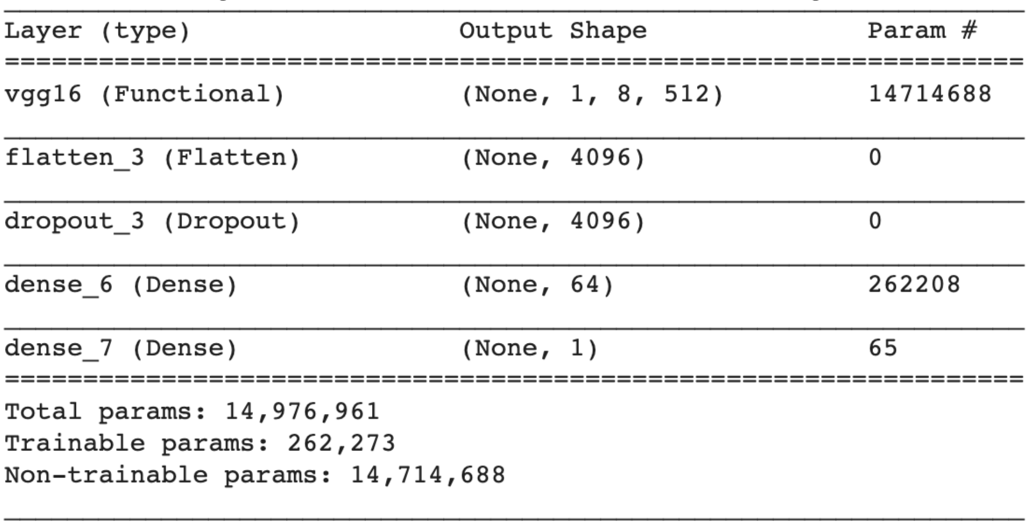


Accuracy converging to final 59.02% loss result. Source: The Author.

5.4.2 ERSP

To classify GAF images the following CNN architecture was used:

Figure 5.19: Model used to train with ERSP images



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 1, 8, 512)         14714688
_________________________________________________________________
flatten_3 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                262208
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 65
=================================================================
Total params: 14,976,961
Trainable params: 262,273
Non-trainable params: 14,714,688
_________________________________________________________________
```

ERSP classification was binary: left and right hand. Source: The author.

While GAF classification results converged, ERSP images did not converge and topped around 65% accuracy and 70% loss. These are some of the reasons we believe could improve the results in future work:

1. The values used to generate the ERSP images were averaged per subject file. Therefore the number of images generated was lower, 54 per class(left and right) in a total of 108, compared to GAF, 3371 left, 3371 right, and 6760 pauses. Having more images to train the model is encouraged.
2. No pre-processing statistical filtering was applied to ERSP images. For future work, exploring image masks for areas with p values < 0.05 is encouraged.

Figure 5.20: ERSP training and validation accuracy evolution across epochs. Source: The Author.

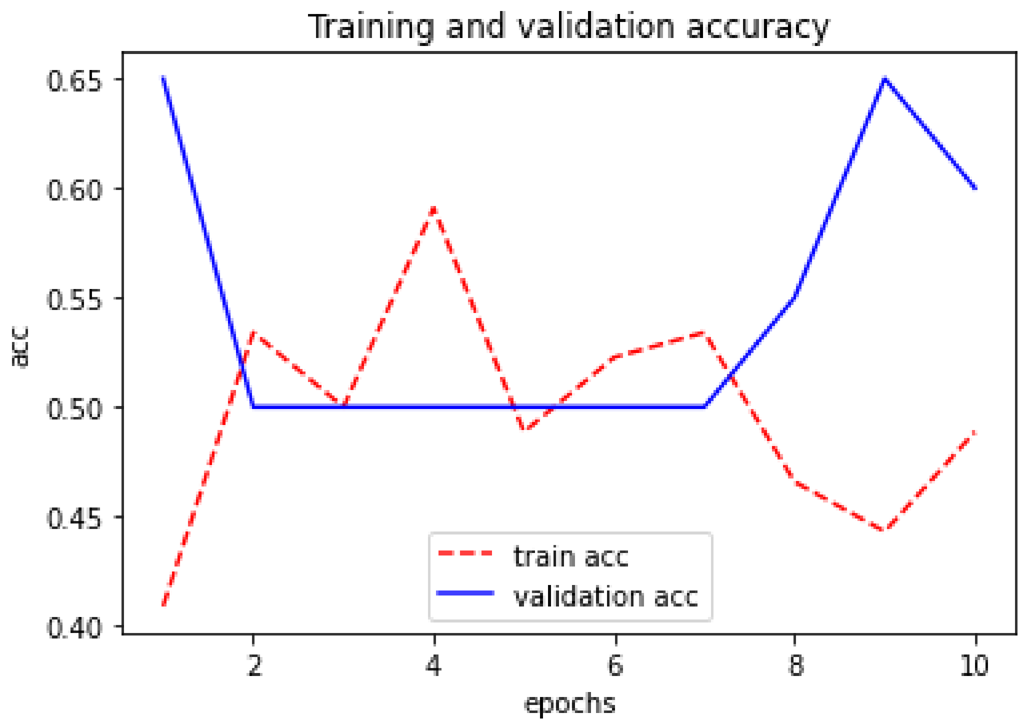
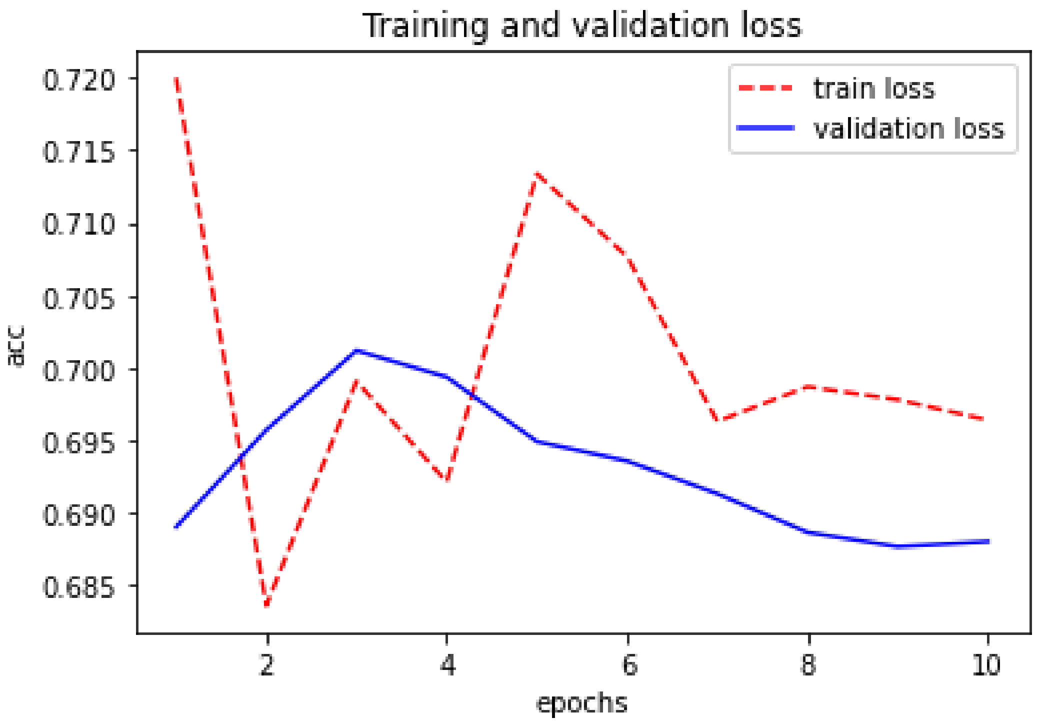


Figure 5.21: ERSP training and validation loss evolution across epochs. Source: The Author.



6 CONCLUSION AND FUTURE WORK

TS2Image performed its tasks as expected, images were generated according to our settings, so it can serve as a valid proof of concept for the idea of automating image generation for training CNNs. TS2Image offers a new way researchers without coding skills can explore time series.

During TS2Image development, was verified that the data sets have slightly different data collection characteristics, that sometimes do not match their descriptions. For example, while one data set would provide the start and end of events, others would mark the start, taking more time to find a more generic configuration solution that would work for both cases. Nonetheless, it is possible to achieve a more generic configuration with future work expanding on it.

Regarding the software implementation, these are some items to be improved in the long run:

- **Modularization:** it could introduce the architecture to extend TS2Image easily. A suggestion is to add plugin or components patterns and separate file reading, processing, and image generation.
- **Configuration:** as demonstrated in figure 4.3, the data sets available have a variable range of data collection characteristics. Therefore, every new data set to be studied to add support to TS2Image should bring more generalization to configuration settings.
- **Support more domains:** This work focuses on EEG, so expanding to support time series from different knowledge domains is advised and possible since GDF files can be used for EEG data.
- **Public contribution:** This software serves as a proof of concept, and starting point, of what could be implemented by the main libraries used if they want to. These libraries could give the option to the client to add default and custom filtering and data processing as we implemented here.
- **File types:** add support to more file types as described in 4.1.2.
- **Multithreading:**During the development, some multi-thread approaches were tested, but they did not make it to the final version of the software due to time constraints. It is essential to mention that a significant performance improvement was presented by implementing one file per process per CPU thread, using Python's *mul-*

tiprocessing library https://docs.python.org/3/library/multiprocessing.htmlmodule-multiprocessing, when processing multiple files.

- **Test coverage:** tests can contribute to increasing trust in TS2Image's output and make eventual refactors and corrections easier to verify.
- **Open to contributors:** having an open-source project that the community can verify is good practice to improve software quality since more interested people will lay eyes on it, and more use cases are tested every day.
- **Improve user interface:** currently, TS2Image only offers a simple command-line user interface. A Graphical User Interface can make it even more accessible to those with less familiarity with computer science skills.
- **Select frequency bands:** on EEG classification, a current use case is to split alpha(8-12Hz) and beta(12-35Hz) frequencies, hence adding support to generate individual images for each frequency band is advised.

As time passes, BCI tends to play an increasingly important role in society, specially for those impaired. Having proper tooling is essential to explore and help evolve every research area, and BCI is no exception. TS2Image comes as a tool to contribute by reducing the time experts invest into creating custom software for every use case, so they can focus on their expertise area, data acquisition and machine learning models training. From a software engineering standpoint, the current version has its limitations and it's ready to be expanded by the research community with public contribution through its GitHub repository.

REFERENCES

ALANSARI, M. et al. Study of wavelet-based performance enhancement for motor imagery brain-computer interface. In: . IEEE, 2018. Available from Internet: <https://doi.org/10.1109/iww-bci.2018.8311520>.

BARTHó, P. et al. Characterization of neocortical principal cells and interneurons by network interactions and extracellular features. **Journal of Neurophysiology**, v. 92, n. 1, p. 600–608, 2004. PMID: 15056678. Available from Internet: <https://doi.org/10.1152/jn.01170.2003>.

BERGER, H. Über das elektrenkephalogramm des menschen. **Archiv für Psychiatrie und Nervenkrankheiten**, v. 87, n. 1, p. 527–570, Dec 1929. ISSN 1433-8491. Available from Internet: <https://doi.org/10.1007/BF01797193>.

BRONZINO, J. D. Principles of electroencephalography. In: **Biomedical Engineering Fundamentals**. S.L.: CRC Press, 1970. chp. 24.

CHAVARRIAGA, R. et al. Heading for new shores! overcoming pitfalls in BCI design. **Brain-Computer Interfaces**, Informa UK Limited, v. 4, n. 1-2, p. 60–73, dec. 2016. Available from Internet: <https://doi.org/10.1080/2326263x.2016.1263916>.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

Dose, H. et al. A deep learning mi - eeg classification model for bcis. In: **2018 26th European Signal Processing Conference (EUSIPCO)**. [S.l.: s.n.], 2018. p. 1676–1679.

FAOUZI, J.; JANATI, H. pyts: A python package for time series classification. **Journal of Machine Learning Research**, v. 21, n. 46, p. 1–6, 2020. Available from Internet: <http://jmlr.org/papers/v21/19-763.html>.

FAWAZ, H. I. et al. Deep learning for time series classification: a review. **Data Mining and Knowledge Discovery**, Springer Science and Business Media LLC, v. 33, n. 4, p. 917–963, mar. 2019. Available from Internet: <https://doi.org/10.1007/s10618-019-00619-1>.

FOURIER, J. The analytical theory of heat. **Nature**, v. 18, n. 451, p. 192–192, Jun 1878. ISSN 1476-4687. Available from Internet: <https://doi.org/10.1038/018192a0>.

Gao, Z. et al. A deep learning method for improving the classification accuracy of ssmvep-based bci. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 67, n. 12, p. 3447–3451, 2020.

GONZALEZ, R.; BARONE, D. Using deep learning and evolutionary algorithms for time series forecasting. In: **ESANN**. [S.l.: s.n.], 2018.

GONZALEZ, R. et al. How artificial intelligence is supporting neuroscience research: A discussion about foundations, methods and applications. In: BARONE, D. A. C.; TELES, E. O.; BRACKMANN, C. P. (Ed.). **Computational Neuroscience**. [S.l.]: Springer International Publishing, 2017. p. 63–77. ISBN 978-3-319-71010-5.

GRAMFORT, A. et al. Meg and eeg data analysis with mne-python. **Frontiers in Neuroscience**, v. 7, p. 267, 2013. ISSN 1662-453X. Available from Internet: <https://www.frontiersin.org/article/10.3389/fnins.2013.00267>.

GRANDCHAMP, R.; DELORME, A. Single-trial normalization for event-related spectral decomposition reduces sensitivity to noisy trials. **Frontiers in Psychology**, Frontiers Media SA, v. 2, 2011. Available from Internet: <https://doi.org/10.3389/fpsyg.2011.00236>.

HATAMI TANN GAVET, J. D. N. Classification of time-series imaaages using deep convolutional neural networks. In: . [S.l.]: The 10th International Conference on Machine Vision (ICMV 2017), ICMV Committees, Nov 2017, Vienne, Austria. ff10.1117/12.2309486ff. ffhal-01743695ff, 2017.

HJELM, S. I.; BROWALL, C. Brainball-using brain activity for cool competition. In: **Proceedings of NordiCHI**. [S.l.: s.n.], 2000. v. 7, n. 9, p. 19.

KEMP, B.; OLIVAN, J. European data format 'plus' (edf+), an edf alike standard format for the exchange of physiological data. **Clinical Neurophysiology**, v. 114, n. 9, p. 1755–1761, 2003. ISSN 1388-2457. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1388245703001238>.

KEMP, B. et al. A simple format for exchange of digitized polygraphic recordings. **Electroencephalography and Clinical Neurophysiology**, v. 82, n. 5, p. 391–393, 1992. ISSN 0013-4694. Available from Internet: <https://www.sciencedirect.com/science/article/pii/0013469492900097>.

LECUN Y., B. L. B. Y. . H. P. Gradient-based learning applied to document recognition. In: . [S.l.]: Proceedings of the IEEE, 86(11), 2278–2324. doi:10.1109/5.726791, 1998.

LEEB C. BRUNNER, G. R. M.-P. A. S. R.; PFURTSCHELLER, G. **BCI Competition 2008 – Graz data set B**. 2008.

LIM, S. H. et al. A novel method for tracking and analysis of eeg activation across brain lobes. **Biomedical Signal Processing and Control**, v. 40, p. 488–504, 2018. ISSN 1746-8094. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1746809417301301>.

LIN, C.-T. et al. Brain computer interface-based smart living environmental auto-adjustment control system in upnp home networking. **IEEE Systems Journal**, v. 8, n. 2, p. 363–370, 2014.

LITTLE, T. **The Oxford Handbook of Quantitative Methods, Vol. 2: Statistical Analysis**. Oxford University Press, 2013. (Oxford Library of Psychology). ISBN 9780199934904. Available from Internet: <https://books.google.com.br/books?id=bCFwAgAAQBAJ>.

LOTTE, F. et al. A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. IOP Publishing, v. 15, n. 3, p. 031005, apr 2018. Available from Internet: <https://doi.org/10.1088/1741-2552/aab2f2>.

LOTTE, F. et al. A review of classification algorithms for EEG-based brain–computer interfaces. **Journal of Neural Engineering**, IOP Publishing, v. 4, n. 2, p. R1–R13, jan. 2007. Available from Internet: <https://doi.org/10.1088/1741-2560/4/2/r01>.

LOUIS, E. S. et al. Electroencephalography (eeg): An introductory text and atlas of normal and abnormal findings in adults, children, and infants. In: . [S.l.: s.n.], 2016.

MAKEIG, S. Auditory event-related dynamics of the eeg spectrum and effects of exposure to tones. **Electroencephalography and Clinical Neurophysiology**, v. 86, n. 4, p. 283–293, 1993. ISSN 0013-4694. Available from Internet: <https://www.sciencedirect.com/science/article/pii/001346949390110H>.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MORRISON, J. B. F. **Brain Facts**. [S.l.]: [S.1.]: Society for Neuroscience, 2018.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: **ICML (3)**. JMLR.org, 2013. (JMLR Workshop and Conference Proceedings, v. 28), p. 1310–1318. Available from Internet: <http://dblp.uni-trier.de/db/conf/icml/icml2013.html#PascanuMB13>.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc., 2019. p. 8024–8035. Available from Internet: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PFURTSCHELLER, G. Event-related synchronization (ERS): an electrophysiological correlate of cortical areas at rest. **Electroencephalography and Clinical Neurophysiology**, Elsevier BV, v. 83, n. 1, p. 62–69, jul. 1992. Available from Internet: <https://doi.org/10.1016/0013-4694(92)90133-3>.

PFURTSCHELLER, G.; ARANIBAR, A. Event-related cortical desynchronization detected by power measurements of scalp EEG. **Electroencephalography and Clinical Neurophysiology**, Elsevier BV, v. 42, n. 6, p. 817–826, jun. 1977. Available from Internet: <https://doi.org/10.1016/0013-4694(77)90235-8>.

Prashant, P.; Joshi, A.; Gandhi, V. Brain computer interface: A review. In: **2015 5th Nirma University International Conference on Engineering (NUiCONE)**. [S.l.: s.n.], 2015. p. 1–6.

ROSSUM, G. V.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SANTANA, G. M. Segmentation of ultrasonic vocalizations in infant mice using deep learning. 2019.

SCHLöGL, A. Gdf - a general dataformat for biosignals version v2.51. In: . [S.l.]: axiv.org, 2013.

SILVA, V. F. et al. Ensemble learning based classification for BCI applications. In: . IEEE, 2017. Available from Internet: <https://doi.org/10.1109/enbeng.2017.7889483>.

SIMONYAN, K.; ZISSERMAN, A. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. 2014. Available from Internet: <https://arxiv.org/abs/1409.1556>.

Stack Overflow. **2020 Developer Survey**. 2020. <https://insights.stackoverflow.com/survey/2020>. Accessed: 2021-04-29. Available from Internet: <https://insights.stackoverflow.com/survey/2020>.

THE Ten Twenty Electrode System: International Federation of Societies for Electroencephalography and Clinical Neurophysiology. Taylor and Francis, 1961. 13-19 p. Available from Internet: <https://doi.org/10.1080/00029238.1961.11080571>.

VECCHIATO, G. et al. The study of brain activity during the observation of commercial advertsing by using high resolution eeg techniques. In: **2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society**. [S.l.: s.n.], 2009. p. 57–60.

WANG, J.; FENG, Z.; LU, N. Feature extraction by common spatial pattern in frequency domain for motor imagery tasks classification. In: . IEEE, 2017. Available from Internet: <https://doi.org/10.1109/ccdc.2017.7978220>.

WANG, Z.; OATES, T. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. In: . [S.l.]: 2015 AAAI Workshop, 2015.

YANG CHEN-YI YANG, Z.-X. C. N.-W. L. C.-L. Multivariate time series data transformation for convolutional neural network. In: . [S.l.]: Proceedings of the 2019 IEEE/SICE International Symposium on System Integration. Paris, France, January 14-16, 2019, 2019.