1 2 9 0

UNIVERSIDADE Ð COIMBRA

João Pedro Amaral Nogueira

Transfer Learning Techniques for P300-based Brain-Computer Interfaces

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor Marco Simões and Davide Borra and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2022

This page is intentionally left blank.

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Đ
COIMBRA

Department of Informatics Engineering

João Pedro Amaral Nogueira

Transfer Learning Techniques for P300-based Brain-Computer Interfaces

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Prof. Marco Simões and Davide Borra and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2022

Acknowledgements

First and foremost, I would like to thank both my advisors, Professor Marco Simões and Davide Borra, for all the assistance, suggestions, availability, and shared knowledge that were crucial for the success of this work.

I would also like to thank my family, girlfriend, and closest friends for all the support given to me, which helped me to push through the hardest times and motivated me to overcome all the problems I had to face.

Lastly, I would like to thank everyone I crossed paths with during the master's degree, from professors to colleagues, as they all contributed positively in their own way.

This page is intentionally left blank.

Abstract

Brain-Computer Interfaces (BCIs) have a wide range of applications, particularly, in the medical field, where there is a huge potential for life improvement for patients with conditions like amputees or paralyzes making it a popular field of research.

Deep Learning approaches on electroencephalography (EEG) signal classification have shown improvements in accuracy, increasing BCIs' quality, but, due to high inter- and intra-subject variability of brain signals, in addition to the lack of large enough datasets available, these methods are hard to implement or very time-consuming to calibrate.

One way of overcoming these constraints is by using Transfer Learning techniques in EEG classification. These consist on transferring knowledge by using pre-trained weights from one related domain as the training starting point on the creation of a new model of another domain.

In this thesis, the main goal is to explore Transfer Learning techniques in P300 Event-related Potentials. This is done by replicating and continuing a previous existent study, using a state-of-the-art Convolutional Neural Network P300 classifier and experimenting with different Transfer Learning methods on different P300 datasets.

The results showed a positive outcome with an increase of 0.03 F-score when using spherical spline interpolation as a method to solve missing channels problems. The use of different Transfer Learning approaches showed no improvements on one dataset due to high signal disparity, and a 0.026 increase in F-score when using pre-trained weights as model initialization and freezing the first convolutional layer during training on a more similar dataset.

Keywords

Brain-Computer Interface, Electroencephalography Signal, P300, Event-related Potentials, Deep Learning, Convolutional Neural Networks, Transfer Learning

This page is intentionally left blank.

Resumo

As interfaces cérebro-computador têm uma ampla gama de aplicações, principalmente na área médica, onde há um enorme potencial de melhoria de vida para pacientes com condições como amputados ou paralisados, tornando-se assim, um campo popular para pesquisas.

As abordagens de Deep Learning na classificação de sinais de eletroencefalografia (EEG) mostraram melhorias na precisão, aumentando a qualidade dos BCIs, mas, devido à alta variabilidade inter e intra-sujeitos dos sinais cerebrais, além da falta de conjuntos de dados grandes o suficiente disponíveis, esses métodos são difíceis de implementar ou muito demorados para calibrar.

Uma maneira de superar essas restrições é usar técnicas de Transferência de Aprendizagem na classificação de EEG. Estes consistem em transferir conhecimento usando pesos pré-treinados de um domínio relacionado como ponto de partida do treinamento na criação de um novo modelo de outro domínio.

Nesta tese, o objetivo principal é explorar técnicas de Transferência de Aprendizagem em Potenciais Relacionados a Eventos do P300. Isso é feito replicando e continuando um estudo anteriormente feito, usando um classificador P300 de Rede Neuronal Convolucional de última geração e experimentando diferentes métodos de Transferência de Aprendizagem em diferentes conjuntos de dados P300.

Os resultados obtidos mostraram um aumento de 0,03 F-score ao usar a interpolação esférica como método para resolver problemas de canais inexistentes. O uso de diferentes abordagens de Transferência de Aprendizagem não mostrou melhorias em um dos conjunto de dados usados devido à alta disparidade de sinal e um aumento de 0,026 no F-score ao usar pesos pré-treinados como inicialização do modelo e congelar a primeira camada convolucional durante o treino num conjunto de dados mais semelhante.

Palavras-Chave

Interface Cérebro-Computador, Sinal de Eletroencefalografia, P300, Potencial Relacionado a Eventos, Deep Learning, Rede Neuronal Convolucional, Transferência de Aprendizagem

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

ASD	Autism Spectrum Disorder.
BCI	Brain-Computer Interface.
CNN	Convolutional Neural Network.
DNN	Deep Neural Network.
EEG	Electroencephalogram.
ERP	Event-Related Potential.
ErrP	Error-related potentials.

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Controlling machines with the power of the mind has been Human's long-time desire for decades. From fictional movies to conceivable medical solutions, the idea of using the human brain as an interface to communicate with computers has always been a trending topic, gathering a lot of attention and research.

Breakthroughs on medical science, with the discovery of brain signals in the 1920s, and the Electroencephalogram (EEG) in 1929, both by Hans Berger [1], created the possibility of using brain waves as a communication channel, as proposed by Kamiya [2], in 1968, later conceptualized by Vidal [3], in 1973, as a Brain-Computer Interface (BCI). These ideas lead to the development of the P300-Speller, In 1988, by Farwell and Donchin [4], the first functional implementation of its kind, which used the P300 brain wave, an event-related potential Event-Related Potential (ERP) component related to the process of decision-making, to input text or commands to a computer by selecting the letters from a flashing screen through thought.

Machine Learning has played a critical role on BCIs, as it made possible the interpretability of brain signals to be used as inputs for commands. This decoding started from more traditional classifiers using Linear Discriminant Analysis or Support Vector Machine on features previously extracted from the EEG signal, to more recent deep learning approaches such as weightless neural networks [5] or Deep Neural Network (DNN) like Convolutional Neural Network (CNN) [6] where the signal could be used with less pre-processing. However, although able to produce better results, deep learning implementations demand a high amount of data to be viable, which, due to the high cost of collecting and privacy issues, make it scarce in this domain. Additionally, the high intra- and inter-subject variabilities of brain signals require time-consuming training and calibrations to achieve high performance [7].

One way of overcoming these problems is the implementation of Transfer Learning, firstly described by Stevo Bozinovski and Ante Fulgosi, in 1976 [8], which subsists on the idea of using models trained on data from related domains to solve different tasks. This approach uses a pre-trained model created from a bigger dataset, as a starting training point for a new model to be fine-tuned and optimized to a specific subject and or brain signal.

The aim of this thesis is to explore Transfer Learning techniques to address the time-consuming calibration constraint on P300-based BCIs, and possibly to apply the same techniques to other brain signals.

1.1 Motivation and Objectives

BCIs have a wide range of applications, more specifically, in the medical field, where there is a huge potential for life improvement for a vast number of patients with different conditions from amputees to paralyzes. Unfortunately, the use of these types of devices is constraint by the lack of data to train robust models, in addition to brain signals having high variability between subjects which results in long times of preparation and calibration. The goal of this thesis is to take advantage of an existing larger P300 dataset and apply different transfer learning techniques to pre-trained models, in order to develop a one-fits-all solution capable of reducing calibration times without compromising accuracy, making the use of this technology in an everyday setting a possibility.

This shall be accomplished by:

- Identifying P300 or related datasets to train and evaluate the models;
- Implementing and validating the existing state-of-the-art solution;
- Developing solutions to overcome the differences between datasets;
- Implementing Transfer Learning techniques to train the models, aiming for smaller calibration times and improved accuracy;
- Analyzing and reporting experimental results using offline data;

1.2 Approach

In order to fulfill the set out goals, the work was divided in different steps, with each one being described on the following section.

1.2.1 Implementing and validating the state-of-the-art solution

The first step was to replicate the approach that won the 2019 IFMBE Scientific Challenge [9], organized during MEDICON 2019, which consisted in achieving the best possible object-detection accuracy based on the P300 signals, using the BCIAUT-P300: A Multi-Session and Multi-Subject Benchmark Dataset on Autism for P300-Based Brain-Computer-Interfaces [10]. This solution uses a CNN based on EEGNet [11] and will be the architecture used throughout this whole experiment. This step was also important to get familiarized with the BCIAUT-P300 dataset as it was the one used to create the main models to used for the Transfer Learning techniques.

1.2.2 Developing solutions to overcome dataset differences

Although the aim of Transfer Learning is to be able to use the same model with different datasets, to optimize it's implementation, the datasets underwent a pre-processing process in order to standardize features like input channel numbers, sample rate and epoch duration. This was be done by manipulating the raw EEG signal.

1.2.3 Creating baseline models

The third step was to create baseline models, using the architecture referenced on step one. On this step, different types of models were created. First, a baseline model was trained for each different dataset using the parameters optimized for the BCIAUT-P300. Then, several attempts were made by changing the hyper-parameters in order to optimize them for each dataset creating new baseline models. Finally, using the whole BCIAUT-P300 data, a transfer model was created from each optimized hyper-parameters.

1.2.4 Implementing Transfer Learning techniques

In this step, it was created new models for each relevant dataset, using Transfer Learning, in order to test and compare it's usage. This was achieved by using the weights trained from the transfer models as starting point of training new models, instead of randomly initialized. Besides that, other models were also obtained by freezing different layers of the transfer model, only letting the other layers' weights to be updated during training, a Transfer Learning technique called fine-tuning.

1.2.5 Comparing results and drawing conclusions

Lastly, with all the different models created, testing metrics were gathered and compared between the different transfer methods, the baseline and the literature, if available. The metrics chosen for comparison are F-score and balanced accuracy. The first one being used as it greatly emphasises the P300 recognition accuracy, and the second one as the datasets are highly unbalanced, with a great difference between targets and non-targets. To conclude, a deep analysis of the outcome results was be done, drawing conclusions about the whole work and finalizing with new ideas and suggestions for future works.

1.3 Document Structure

This document is divided into seven main chapters:

- The first one is the Introduction where a brief explanation of the thesis is given as well as the main acknowledgements, motivation and objectives.
- The second chapter focuses on explaining the basic concepts needed to understand the topic by giving the background on BCIs, EEG, P300 ERP, Deep Learning, and Transfer Learning.
- The third chapter addresses state of the art related to these thesis' topics.
- The fourth chapter describes all the experimental setup.
- The fifth chapter details all the results gathered from the experiment analysing and discussing them.
- Finally, the sixth chapter has the conclusion and final thoughts about any future work.

This page is intentionally left blank.

Chapter 2

Background

This chapter explains the main concepts needed to understand this thesis such as what is a Brain-Computer Interface, what is the P300 signal and how it is detected, and what is Transfer Learning and how it can be used.

2.1 Brain-Computer Interface

In Computer Science, an interface is an electronic circuit that controls the connection between two devices and provides trustworthy communication between them. From keyboards to touch screens or more recently, virtual reality devices, there is a wide variety of interface types. Specifically, Brain-Computer Interfaces, are a kind of interface that uses brain signals as inputs to communicate with external devices, originally conceptualized by Vidal [3], in 1973, Brain-Computer Interface (BCI)s have a large range of applications in various fields, but are mostly researched for medical solutions like repairing human cognitive or sensory-motor functions. From the technical point of view, a BCI system operates in 4 phases that form a cycle as shown on figure 2.1. There is the signal acquisition phase where brain signals are recorded, usually from the Electroencephalogram (EEG), followed by the pre-processing where the signal is treated in order to be usable for the next phase. The next phase is decoding/encoding where the processed signal is classified and translated into a command sent to the application. With the given command the application should give the user a feedback of the action that was taken finishing the cycle and starting once again.

Related to how the brain signals are collected, BCIs can be separated into three different categories: invasive BCIs, which are implanted directly into the grey matter of the brain during neurosurgery, and produce the highest quality signals of BCI devices but are prone to scar-tissue build-up. Partially invasive BCIs, that are still implanted inside the skull but, contrary to the latter, rest outside the brain, reducing the risks of scar tissue or rejection with the downfall of collecting lower quality signals. Lastly, non-invasive BCIs, which produce the lowest quality signals from the three, due to higher signal noise, yet are easier to implement, being non-invasive EEG-based BCIs the most common BCI used in a substantial majority of published work [12].

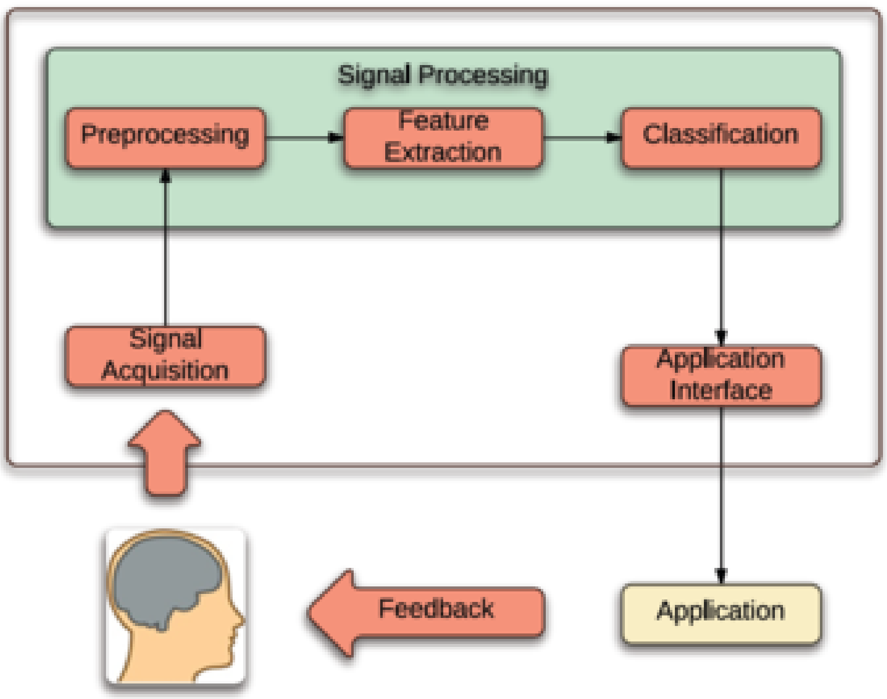

Figure 2.1: BCI operating loop.

2.2 Electroencephalography

The production of electrical currents by the human brain was discovered in the early 1920s by Hans Berger [1], who, by placing electrodes on the scalp managed to measure these signals. This concept was termed Electroencephalography and proved to be a key tool in neuroscience, especially for understanding or diagnosing neuro-pathologies [13].

EEG measures the voltage fluctuations produced by the brain activity, captured by multiple electrodes attached to the scalp on specific points seen in figure 2.2. This setup needs preparation in order to be able to collect adequate data, such as scrubbing the scalp to remove dead skin that can create impedance and applying a conductive gel, before placing the electrodes.

Unfortunately, being this a non-invasive approach, not only the signal is weaker as it has to travel through several layers of the head, but it also suffers from a lot of noise either from other bio-signals or outside electromagnetic interference. To be able to reliably use and interpret the signals collected, several trials must be conducted averaging the results so that the signal-to-noise ratio is increased. The resulting data collected from the EEG can then be interpreted by either analyzing the frequency oscillations over time or by focusing on event-related potentials which are potential fluctuations time-locked to an event, such as ’stimulus onset’ or ’button press’.

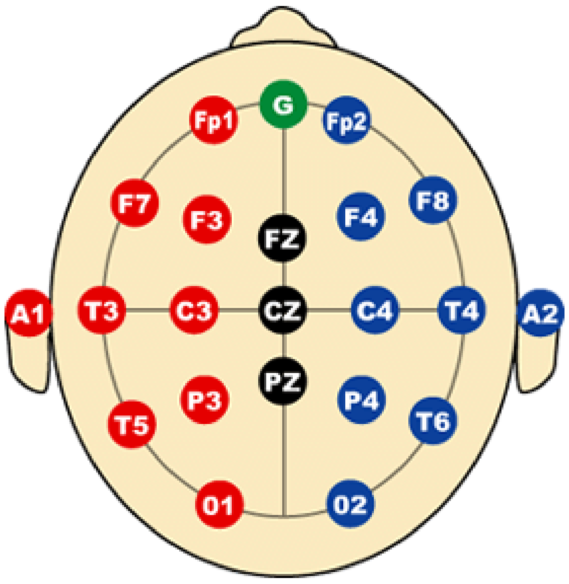

Figure 2.2: International 10-20 Electrode Placement System. [14]

2.3 P300

The data collected from an EEG can be interpreted in different ways, one of them being the analysis of event-related potentials which are the measured brain response to a specific sensory, cognitive, or motor event. These types of waveforms are a positive or negative voltage deflection of the brain signal and are usually named with a letter (N/P) referring to the polarity, followed by a number that indicates either the latency in milliseconds or the ordinal position of the component in the waveform [15].

Following the nomenclature rules previously mentioned, the P300 component is a positive peak that usually occurs 300 milliseconds after the stimulus, shown as the furthest Event-Related Potential (ERP) on the right in 2.3, but can peak between 250 milliseconds to 500 milliseconds and is best measured with electrodes covering the parietal lobe. This ERP is linked to attention and decision-making process and is usually elicited using the oddball paradigm [16], in which a low-probability stimulus of interest is mixed and presented alongside several high-probability stimuli of non-interest. For example, flashing several times a red light (frequent stimulus of non-interest) and once a green light (infrequent stimulus of interest) produces a positive peak on the EEG around 300 milliseconds after the green light is shown. ERPs, like the P300, are widely used in BCIs as the input signals to be translated to a command for the machine. One of the best well-known examples is the previously referenced P300-Speller which uses a grid of letters of the alphabet and flashes each row and column registering which one triggered the P300 signal and sending it to the computer creating a text.

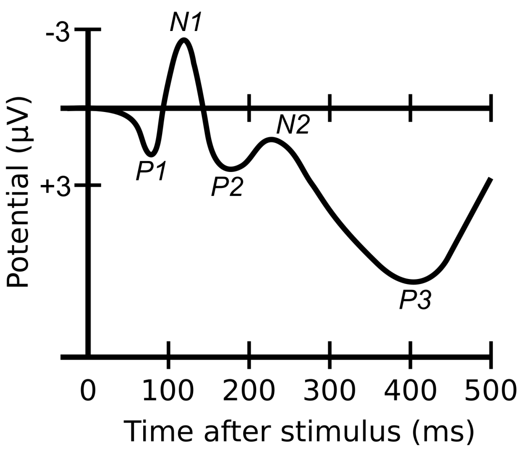

Figure 2.3: Event-related potential components. From left to right: P100, N100, P200, N200 and P300.

2.4 Deep Learning

The term Deep Learning was introduced to the machine learning community by Rina Dechter in 1986 [17] and it refers to the method based on the use of artificial neural networks which are a model that simulates the structure of neural cells [18]. This artificial neural network, also called Deep Neural Network (DNN), see image 2.4, is formed by several connected layers of artificial neurons, hence the adjective "deep", that learns via a learning process to map the input to the output (e.g., by performing binary classification). This input-output relationship is defined by learning the weights of the connections between the artificial neurons that define the network. The basic DNN architecture consists of an input layer, an output layer, and several hidden layers, and can directly learn the data after normalization to get the features and classifier jointly.

The advantage of this machine learning method is that it progressively extracts higher-level features from the raw input [19]. However, this method lacks in interpretability and requires large datasets (often >100K of training examples) to learn robust and useful features from the input data. Different types of DNNs are applied to EEG signal analysis, such as convolutional neural network , recurrent neural networ and autoencoder [20].

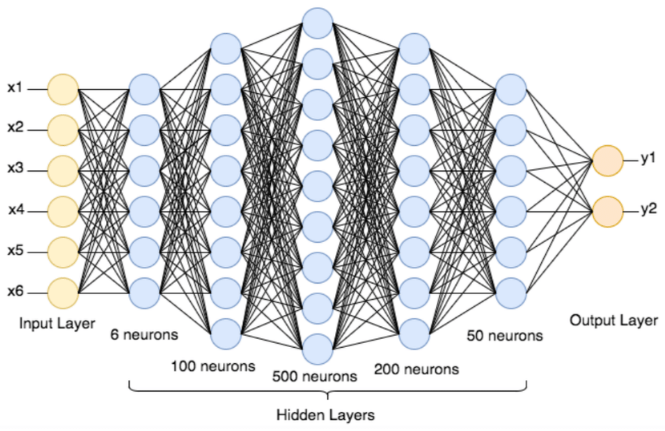

Figure 2.4: Deep Neural Network structure example.

2.5 Transfer Learning

There are several methods in Machine Learning, most of them inspired by how Humans learn, one of them is Transfer Learning. Firstly described by Stevo Bozinovski and Ante Fulgosi, in 1976 [8], Transfer Learning is based on the idea that knowledge from a domain can be used to learn more efficiently something from a similar domain, just like someone who has previous knowledge in playing the guitar, will most likely learn piano much more efficiently than someone with no knowledge in any instrument and this is done by transferring from one task to another. This method is especially useful when there is a lack of information for training, being able to train something with a dataset from a different domain, or to reduce the training time, by using pre-trained models from the same or different domain and fine-tuning them to the specific task.

On BCIs and brain signal classification, more specifically P300 classification, this technique tackles both main problems already state, offering a solution for the time-consuming subject-specific training and calibration by using a pre-trained model created from a different subject dataset [21].

Besides initializing the trainable weights of each layer using a pre-trained model, instead of randomly, each layer can then be frozen during trainig. This allows to keep the original weights on the initial part of the decoding, only fine-tuning the lower layers of the model in order to adapt to the new information.

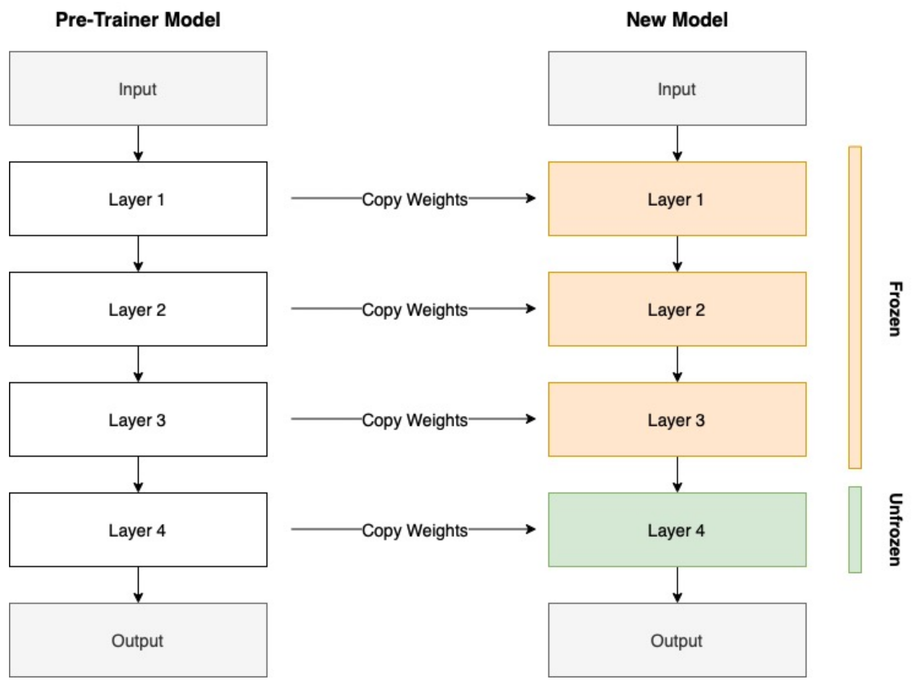

Figure 2.5: Example of Transfer Learning training method with three frozen layers and one unfrozen to fine-tune.

This page is intentionally left blank.

Chapter 3

State of the art

This chapter has a brief explanation of the current state of the art approaches on BCIs, P300 classification, and Transfer Learning and how they are related to this thesis. It is divided into three sub-chapters, each one related to each topic mentioned.

3.1 P300-based Brain-Computer Interface

Brain-computer interfaces enable users to send messages or commands directly through brain activity, without any movement, using brain signals measured with electroencephalography. There are several categories of BCIs, depending on the type of brain activity used for control, including steady-state visual evoked potential [22], event-related desynchronization [23], slow cortical potential based [24] and event-related potentials like P300 [4], which are the ones used on this thesis.

As mentioned previously, the first fully functional P300 BCI was the P300-Speller by Farwell and Donchin [4], it worked by flashing a 6 × 6 matrix of English letters and other content presented on a monitor using it as the stimulus for the P300 peak. The study reported results from four participants who were all healthy and was conducted by asking the subject to count silently each time one specific target cell of the matrix flashed while ignoring all other flashes, followed by a random row or column of the matrix flash. This approach made that only the row or column flashes that included the target letter elicited a P300. The system could, therefore, infer the target based on identifying which row and column produced the largest P300. Throughout the years several studies were able to improve the P300-Speller by proposing a region-based approach [25], contrary to the classical row-column, shown on figure 3.1, which suffered from more human errors due to non-target characters being adjacent to the target [26] or by completely changing the flashing letter paradigm to showing familiar faces as it was shown, by Kaufmann, Schulz, Grünzinger, & Kübler [27], to produce more distinct ERP target versus non-target differences, improving the P300 classification. The latter approach, sometimes reference as "face-speller" proven also to be more adequate for patients with neurodegenerative disease and an unacceptable accuracy rate for communication than using the classical P300-Speller, mainly due largely to early ERP components such as the N400, confirmed by a study done by Geronimo & Simmons [28], in 2017, where it was compared the face speller to a conventional matrix speller when used by amyotrophic lateral sclerosis patients. In 2020, Ratcliffe & Puthusserypady proposed a novel graphical user interface for P300 BCI using a region-based T9 system with familiar face presentation cues capable of eliciting strong P300 responses, achieving an average accuracy of 91.3 ± 4.8% and an Information Transfer

Rate of 2.2 ± 1.1 commands/minute (12.2 ± 6.0 bits/minute) on an experiment conducted using ten participants with normal or corrected to normal vision [29].

Stepping away from speller BCIs, recently, other types of BCIs using P300 ERPs elicited by visual queues have been developed, one especially important to this thesis, as it is where the main dataset was taken from and where the online experiments will be conducted, combines the use of Virtual Reality, which has been vastly used in neuro-rehabilitation, with BCIs to use improve Autism Spectrum Disorder patients social skills [30]. This BCI presented the user an immersive virtual environment of a bedroom, with an avatar that would give the join-attention social cue of pointing to an object with a head movement, making it the target, and asking the patient to look at it and count the number of times it would flash. The object flashing would be done in a randomized order, alongside other different objects inside the room, which would make the event of flashing the target object the rarest one, which elicits the P300 ERP.

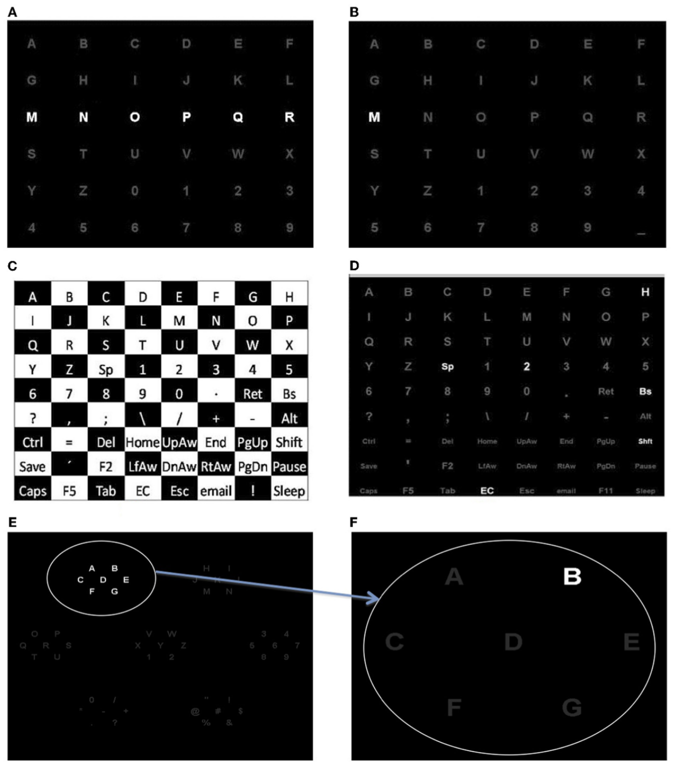

Figure 3.1: Different P300-Speller paradigms.(A) Row/column paradigm: row and columns are flashed. (B) Single character paradigm: each character is flashed. (C,D) Checkerboard paradigm. (E,F) Region based paradigm where a set of characters in level 1 (E) are expanded in level 2 for spelling character "B" (F).

There are also other ways of eliciting the P300 ERP besides using visual stimulus, these approaches were developed as a way of overcoming the inability showed by patients in a locked-in state to use them, as a result of lack or uncontrolled eye movements [31]. One solution is to use auditory stimulus, even though it is still not as efficient as the visual stimulus approach, the use of different tones on loudspeakers placed around the subject managed over 80% accuracy rates on healthy users [32] but showed to be unsuccessful on locked-in patients, however, follow-up studies came to prove an improved performance on the latter using more training sessions, making it a viable approach [33]. Another solution for users that are unable to either use visual or auditory BCIs are tactile BCIs, that use small discs, named tactors, placed over specific areas providing vibrotactile stimulation that can elicit ERP activity that appears somewhat similar in shape and distribution to ERP activity elicited in P300 BCIs based on other modalities. Kaufmann, Herweg, & Kübler, in 2014 [34], managed to create a tactile BCI able to control a virtual wheelchair through four tactors, each one representing a navigation direction, placed on both thighs, abdomen, and lower neck of 15 participants where 11 of them successfully steered through a building delivering navigation commands by focusing their attention on the desired tactile stimulus in an oddball-paradigm.

The state of the art of P300 BCIs has come a long way since the Farwell and Donchin's P300-Speller [4], and one great contributor to all these breakthroughs is the annual BCI Award [35], which every year recognizes outstanding and innovative research in the field of Brain-Computer Interfaces and chooses a winner from 12 nominated projects judged by an independent and international jury assembled with world-leading BCI experts who work with invasive and non-invasive brain-computer interfaces and neurotechnologies in both research and clinical environments.

3.2 P300 Classification

Signal classification, or decoding is always an important part of any BCI since it is where the brain signals recorded are translated into commands for the application. As it has been mentioned before, EEG signals are noise sensitive, non-linear and non-stationary, characteristics that depend on the subject and the environment [36], and also suffer from inter-subject variability, due to anatomical and physiological differences among subjects [7]. These specificities of the EEG signal make it a real challenge to develop an efficient classification method, not only in accuracy, but also in training time.

Traditional machine learning techniques such as Linear Discriminant Analysis [37] [38] or support vector machines [39] are still commonly used, even though they take prolonged calibration time and suffer from an under-sampling problem [40]. Recenlty, some improvements have been made on traditional methods, like the use of Riemannian geometry, by Korczowski et al. [41], where this method was tested in a classification of single-trials Event-Related Potential on two subjects playing a collaborative BCI game and proved to outperform significantly the mean performance of the two users and analogous classifiers based on the step-wise linear discriminant analysis [4], or the use of weightless neural networks [5] which also proved to outperform the state-of-the-art algorithms several times and showed that that smaller memories, regarding the weightless neural network hyper-parameters, achieve better results most of the times.

Deep learning revealed to be an interesting and feasible approach to EEG signal classification. While other methods require feature extraction and prior signal treatment, deep neural networks automatically learn the relevant features for a given task being able to process the raw data. With the adaptation of Convolutional Neural Networks from image recognition to EEG signal some relevant results were achieved, such as the work developed by Borra et al.[6], that used an adaptation of EEGNet [11] to outperform and win the 2019 IFMBE Scientific Challenge [9]. This work is especially relevant to this thesis because not only is the classification approach to be used, it already experiments with Transfer Learning with some success. However, deep learning methods also have their drawbacks, two of them being the lack of interpretability and the large amount of hyper-parameters to tune. One possibility already in development from the last method discussed is converting the Convolutional Neural Network (CNN) to an Interpretable Convolutional Neural Network and applying hyper-parameter tuning approaches (e.g., Bayesian optimization).

3.3 Transfer Learning

Machine Learning is widely and successfully used in many applications but traditional machine learning approaches only succeed when the training data and the testing data have the same input feature space and data distribution. Moreover, it is needed a large amount of training data to able to create a robust model. Unfortunately, when it comes to EEG data, the previous assumptions are not met. First, brain signals are complex, which leads to non-stationary, non-linear, and non-Gaussian EEG signals [42]. Second, the characteristics of EEG signals are highly subject to various individual differences such as age or mentality, and can even differ for different individuals for the same event [43]. Last, EEG signals are highly noisy and susceptible to be distorted by artificial interference [44]. Additionally, there is a lack of recorded data available as Dutta and Nandy [45] reported when faced with the difficulty in finding EEG data sets to estimate mental stats with deep learning models.

One way of overcoming these problems is by using Transfer Learning. Transfer Learning was first proposed at the 1995 NIPS- 95 seminar, on 'Learning to Learn' [46], as applying the knowledge learned in one domain into a different but related domain, but was later redefined, in 2005, by the Defense Advanced Research Projects Agency, as the ability of the system to recognize and apply the knowledge and skills learned in previous tasks to a new task [47]. In definition, considering the following concepts:

- *Domain*: A domain D consists of the feature space X of n dimensions and the probability distribution P(X) of X, where $X = x_1, x_2, \ldots, X_n$. In transfer learning, the domain containing known knowledge is called source domain, which is usually represented by D_S, and the domain containing unknown knowledge to be learned is called target domain, which is usually represented by D_T.
- *Task*: A task is a learning goal, which consists of the label space Y and the prediction function $f(.)$ (also written as P(Y|X) in probability theory, which means the probability of Y under condition X), where $Y = y_1, y_2, \ldots, Y_n$. According to the definition of task, the label space of the source domain and the target domain are represented as Y_S and Y_T.
- *Transfer learning*: When there is a difference between domains or tasks, the knowledge learned in the source domain can be transferred to the target domain through transfer learning. It is worth noting that the precondition of transfer learning is that the domains or tasks must be different but similar to some extent. In specific problems, the knowledge learned is usually labeled as Y. Labels and their corresponding features make up data in this domain, which is denoted as $D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

Herewith, the formal definition of Transfer Learning is: Given a source domain D_S and a target domain D_T, T_S and T_T are the tasks of D_S and D_T, where $D_S \neq D_T$ or $T_S \neq T_T$ .The goal of transfer learning is to apply the knowledge in D_S to help learn the knowledge in D_T , where $D_S = X_S, P(X_S)$, $D_T = \{X_T, P(X_T)\}$,$T_S = \{Y_S, P(Y_S|X_S)\}$ and $T_T = \{Y_T, P(Y_T|X_T)\}$. In this definition, the condition $D_S \neq D_T$ or $T_S \neq T_T$ is the problem that the transfer learning approaches mainly try to solve [48]. Regarding EEG signal analysis, there are many different TL methods that can be applied from marginal distribution adaptation, when the marginal distributions of the source domain and the target $(P(X_S) \neq P(X_T))$ are different, especially applied on cross-subject tasks, like the proposed method by Santana et al. which used a cross-subject classifier to predict the stimulus presented to a subject from the analysis of the brain activity [49], to conditional distribution adaptation, when the conditional distribution of the source domain and the target domain $P(Y_S|X_S) \neq P(Y_T|X_T))$ are different, like Khalaf and Akcakaya introduced, using Bhattacharyya distance to measure the similarity of the conditional distributions between new users and existing users [50]. As for this thesis, the most relevant methods are the ones for deep neural networks, as it is the approach used.

Concerning Transfer Learning methods in DNN, there are three different approaches, as seen in figure 3.2. Fine-tuning is the most commonly used method and consists of using the weights of a pre-trained model to train the new model with the new target data and can either be done by fine-tuning the whole network or freezing some layers and fine-tuning others. The first approach produced an increase in classification accuracy as shown by Wu et al. who proposed a parallel multiscale filter bank CNN and fine-tuned it with 10, 20, 50, and 100 samples from the target domain [51]. The second approach has been proven to improve DNN performance by Yosinki et al. by freezing some layers and fine-tuning certain layers via experiments [52], or Raghu et al. that also retrained the last layer of 10 pre-trained models to detect the variants of seizures and managed to outperform conventional feature and clustering-based methods [53].

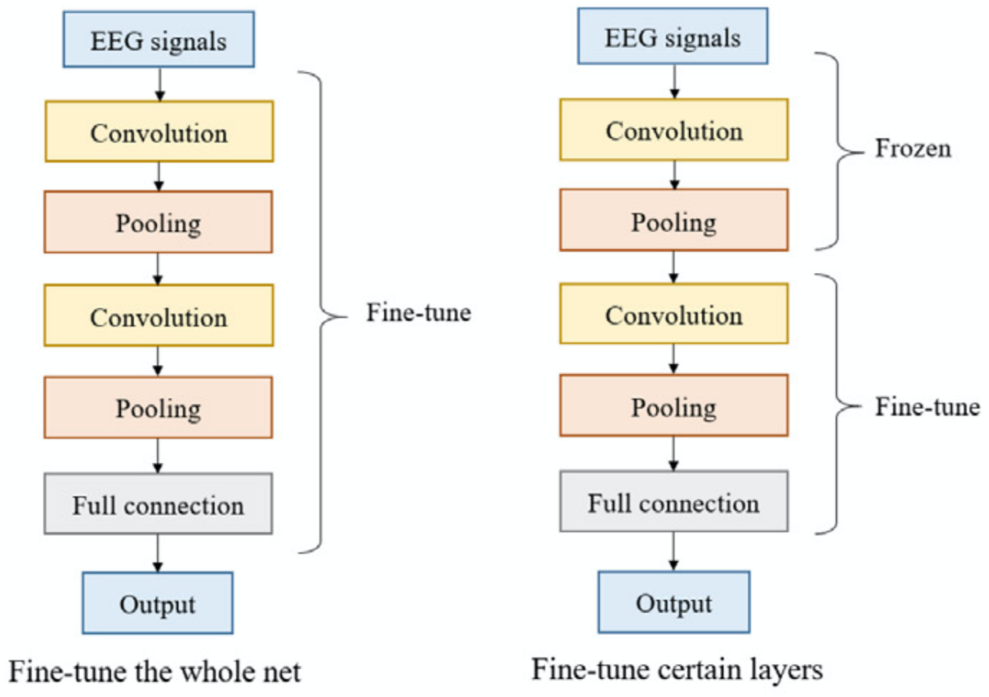

Figure 3.2: Transfer Learning - DNN fine-tuning.[48]

Although fine-tuning is easy to implement and understand, it has its flaws, as it becomes less effective when hen the distributions of source domain and target domain are different. In this case, researchers formulated the idea to adjust the cost function of the original network by adding a domain loss to measure the distribution of the source data and the target data, naming it deep neural network adaptation, as seen on figure 3.3. Zhang et al. proposed a cross-subject transfer learning model using deep neural network adaptation by narrowing the difference of the feature distribution between the source and target domain via deep domain confusion [54]. Another example of this method is done by Svanera et al. where a multivariate link between the output and the fully connected layer of a CNN model was established through reduced rank regression and ridge regularization to transfer and decode the features to brain fMRI data [55].

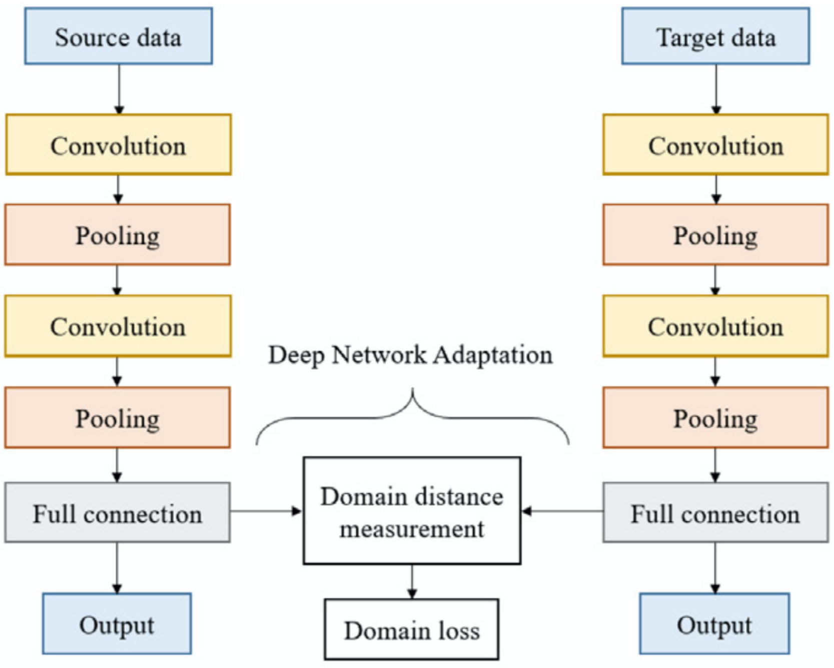

Figure 3.3: Transfer Learning - Deep Network Adaptation.[48]

Finally, there is the generative adversarial network method, first proposed in 2014 [56], which consists of having fake samples generated by a generative network from a given data and estimated by a discriminative network to distinguish its source, as shown on figure 3.4. This is applied in Transfer Learning by generating the samples from both domain and target sources, sending them to the discriminator, and feeding the result back to the generator, iterating these steps until the features cannot be distinguished. This was used by Ma et al. to reduce the influence of subject variability in BCI by extracting the common features between the subjects and the features specific to subjects without using any target domain data [57].

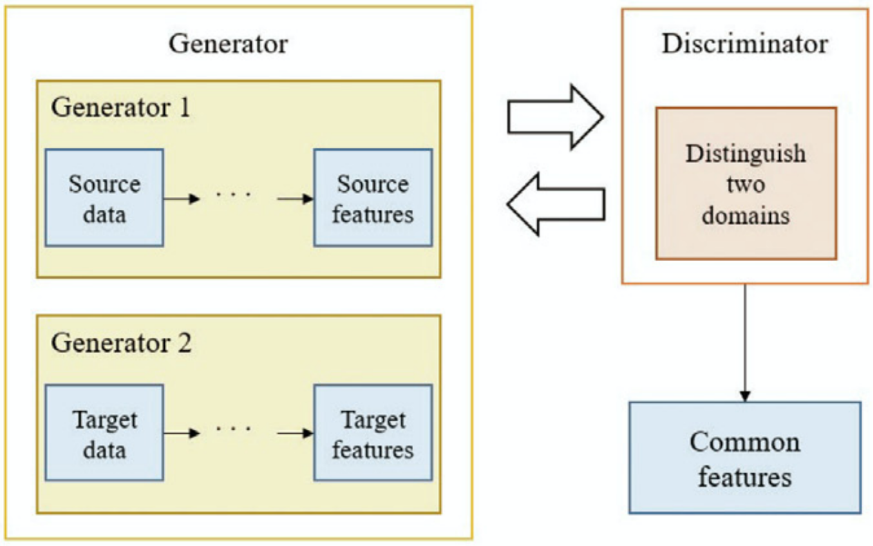

Figure 3.4: Transfer Learning - Generative adversarial network transfer.[48]

This page is intentionally left blank.

Chapter 4

Experimental Setup

This chapter describes the tools and environment used in this work. Here, all the main frameworks, toolkits and datasets are enumerated alongside the explanation of all the required steps in development such as data pre-processing and model creation.

4.1 Environment

4.1.1 Programming Language

Python is a high-level, interpreted, general-purpose programming language and was the programming language of choice to conduct this work. Python is vastly used on machine learning projects due to it's open-source libraries that facilitate the creation and training of ML models as well as data treatment like signal manipulation. This decision was also backed by the use of Python on the related work.

4.1.2 Speechbrain

In order to implement and train the architecture and model of the solution chosen for the P300 classification, the toolkit [58] was used. SpeechBrain is an open-source and all-in-one conversational AI toolkit based on PyTorch developed to easily create state-of-the-art speech technologies but, due to flexibility, customization and signal processing tools, can be used in other areas like EEG signal decoding and classification.

4.1.3 MNE

As mentioned before, the datasets needed to undergo some pre-processing in order to mitigate their differences. For this, the MNE [59] [60] library was used as it offers a wide range of EEG signal data treatment and manipulation.

4.2 Datasets

In order to train and validate the model several datasets were used. One to work as the base model for the training of the transfer model and two more for experimental purposes.

4.2.1 BCIAUT-P300

The BCIAUT-P300 [10] is a unique benchmark dataset released for the IFMBE 2019 scientific challenge [9] and was created while validating a P300-based BCI aimed to improve Autism Spectrum Disorder (ASD) patients' joint attention. The BCI paradigm was based on an immersive virtual environment, shown on 4.1, presenting a bedroom with several pieces of furniture, an avatar, and 8 objects of interest (1. books on a shelf, 2. a radio on top of a dresser, 3. a printer on a shelf, 4. a laptop on a table, 5. a ball on the ground, 6. a cork-board on the wall, 7. a wooden plane hanging from the ceiling, and 8. a picture on the wall). The avatar would give the social join-attention cue to the participant, pointing to one of the objects of interest making it the target, and then flashing the 8 objects randomly, eliciting the P300 response when the target object flashed.

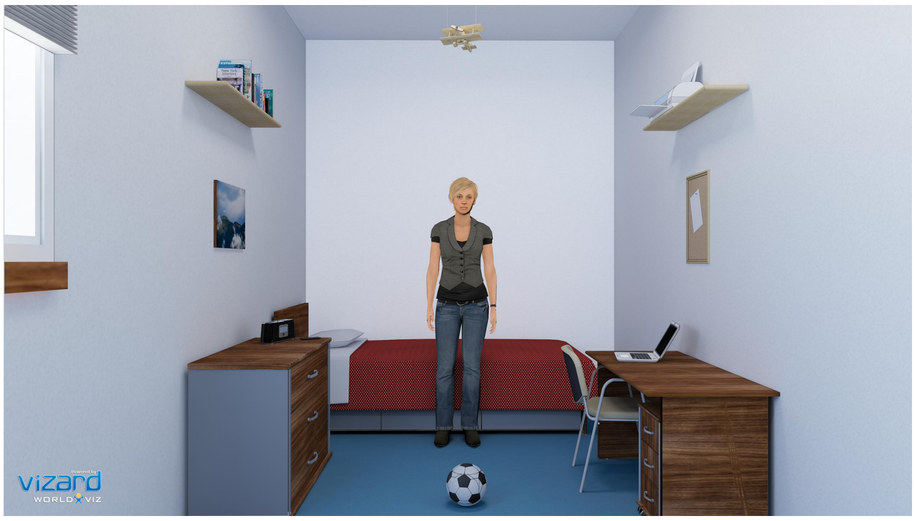

Figure 4.1: Snapshot of the virtual environment, showing the scenario, the virtual avatar and the objects for joint-attention targets.[9]

The data is composed of EEG signals from 7 identical training sessions (spread over 4 months) of 15 different high-functional ASD participants, recorded using the g.Nautilus system module (g.tec medical engineering GmbH, Austria) from 8 active electrodes positioned at C3, Cz, C4, CPz, P3, Pz, P4, POz locations with the reference electrode placed at the right ear and the ground electrode at AFz location, at a sampling rate of 250 Hz and pre-processed using a notch-filtered at 50 Hz and pass-band-filtered between 2 and 30 Hz. Each session was divided into two parts: calibration and online phase. Data from calibration and online phases were named in the dataset as train and test data, respectively. The first phase is composed of 20 blocks, each block containing 10 runs, resulting in 200 target P300 signals and 1400 non-target signals acquired at this phase. Regarding the online phase, 50 blocks were recorded for each participant using K runs per block. The value of K varied between subjects and sessions since it was an output of the calibration phase, ranging from 3 to 10, resulting in an average 2838 trials for the test data.

Each block consists of the user trying to identify one of the objects as the target and one run is composed by a single flash of each object once for 100 ms at different times and random order, with an inter-stimulus interval of 200 ms as it can be seen on the 4.2 schematic.

As of this writing, the BCIAUT-P300 is the largest P300 dataset publicly available, making it the perfect fit to use as the training dataset for the transfer model. The amount of training examples make it suitable for deep learning approaches as shown by the results of

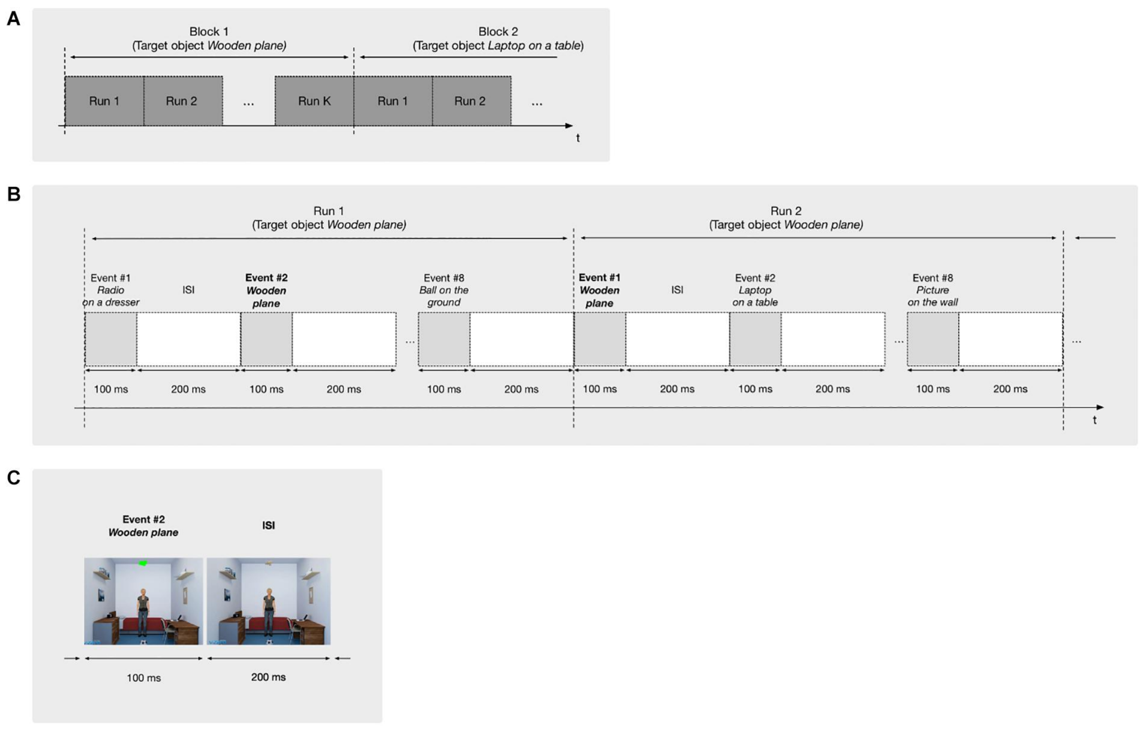

Figure 4.2: Structure of the paradigm with its subdivisions in blocks, runs and events. (A) Structure of the blocks: each block is used to identify a single target object and is composed by K runs. (B) Structure of the runs: each run is composed by 8 events, each consisting of the flashing of one of the objects. (C) Structure of an event: it consists of the flashing of the corresponding object by 100 ms, followed by an interval of 200 ms.[9]

the IFMBE 2019 scientific challenge [9] and will be used to create models whose trained weights will be used to try to improve the same solution on other datasets with less data.

4.2.2 BCI-DOUBLE-ERRP

BCI-Double-ErrP-Dataset [61] is an EEG dataset recorded while participants used a P300-based BCI speller 4.3. This speller uses a P300 post-detection based on Error-related potentials (ErrP) to detect and correct errors (i.e. when the detected symbol does not match the user's intention). After the P300 detection, an automatic correction is made when an ErrP is detected (this is called a "Primary ErrP"). The correction proposed by the system is also evaluated, eventually eliciting a "Secondary ErrP" if the correction is wrong. The overall approach is called "Double-ErrP detection and correction" [62].

The dataset contains EEG signals of 2 sessions from 8 participants, 7 able-bodied and one tetraplegic, recorded using a scalp EEG with 12 Ag/AgCl electrodes, except for one participant who used passive electrodes (g.USBamp bioamplifier, g.tec, Austria), using the right earlobe and AFz as a reference and ground electrodes respectively. The signals were sampled at a rate of 256 Hz and pre-processed using a notch filter at 50 Hz and a band-pass filter at [0.1 - 100] Hz.

Sessions are divided into three phases, with the first two being part of the training data and the last one the test data. The first session has two calibration phases, one for the P300-Speller calibration, where the participants focused attention on the 10 letters of the word *"INTERFACES"*, for each letter, all symbols flashing 9 times collecting 90 targets

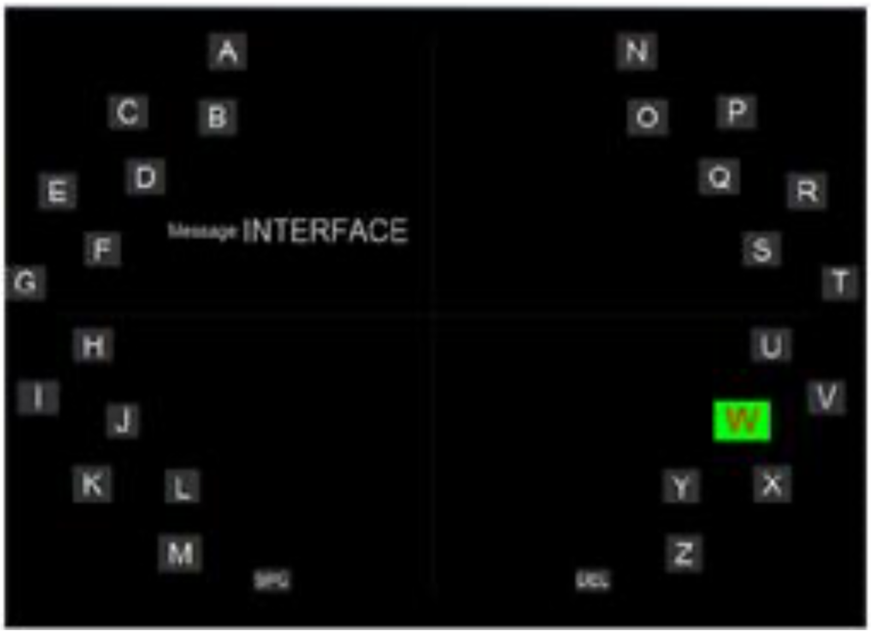

Figure 4.3: Lateral single character speller paradigm. Each symbol is highlighted once in each round. The highlight time of the stimuli is 75 ms with a 75 ms inter-stimuli interval and the inter-trial interval is 4s. [61]

and 2430 non-targets. The second calibration phase is related to the ErrP, where the participants had to write several times the Portuguese sentence *"ESTOU-A-ESCREVER-COM-UMA-INTERFACE-BCI"* (38 characters) without either interruption or correction. The second session was held on a different day from the first one and is the third phase where the participants spelled the same sentence of the ErrP calibration and both first and second ErrP were detected.

The data is stored individually for each sentence spelled by each participant in each session and contains 15 fields. The first one is the timestamp of each sample followed by the 12 signal samples recorded. The 14th field has information regarding the symbols and is labeled as shown in 4.4 with "90" the code of the target symbol when highlighted, "1:28" the codes of the non-targets when highlighted, and "100" the code indicating that the detected letter is shown to the user. The last field is a zero vector except for the timestamps when feedback occurs. The test data has an extra field like the previous one but regarding the second ErrP feedback. Although there are two different event-related potentials, only the P300 data is used on this work. This dataset is used to train and validate models using transfer learning tecniques with the previously created models from BCIAUT-P300.

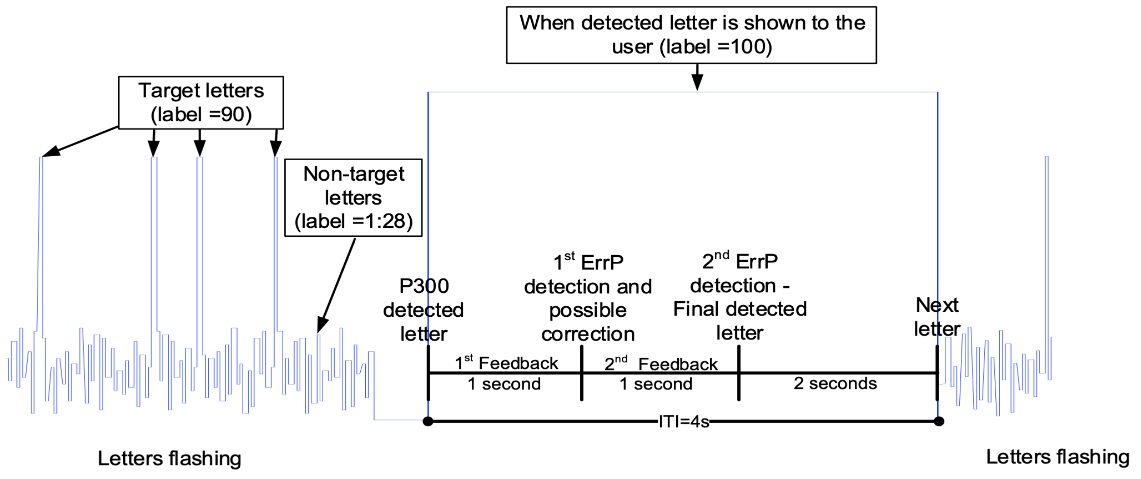
When detected letter is shown to the user (label =100)
Target letters (label =90)
Non-target letters (label =1:28)
P300 detected letter
1st ErrP detection and possible correction
2nd ErrP detection - Final detected letter
Next letter
1st Feedback 1 second
2nd Feedback 1 second
2 seconds
ITI=4s
Letters flashing
Letters flashing

Figure 4.4: Description of events saved on the 14th field. [61]

4.2.3 ERPCORE

The ERPCORE [63] is a freely available online resource consisting of optimized paradigms, experiment control scripts, example data from 40 neurotypical adults for 7 widely used event-related potentials components: N170, mismatch negativity, N2pc, N400, P300, lateralized readiness potential , and error-related negativity. Regarding the P300 data, it was elicited in an active visual oddball task adapted from Luck et al. [64] by randomly presenting one of 5 letters (A, B, C, D and E) where one was primarily set as the target and the other four, non-targets, thus creating and odd event with a probability of 20%. The visual stimuli were shown for 200ms followed by an interval of 1200 to 1400 ms shown in 4.5. Every letter was selected as target once and 8 runs were collected per block, thereby producing 200 events, with 40 targets and 160 non-target.

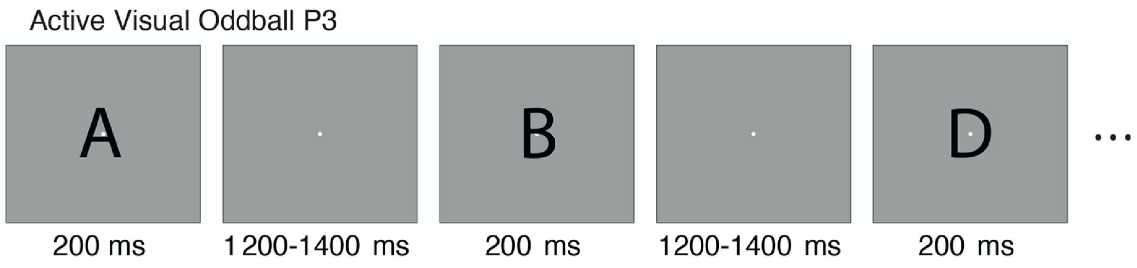

Figure 4.5: Active visual oddball P300 experiment [63] example.

The EEG signal was continuously recorded using a Biosemi ActiveTwo recording system with active electrodes (Biosemi B.V., Amsterdam, the Netherlands). It was used 30 scalp electrodes, mounted in an elastic cap and placed according to the International 10/20 System (FP1, F3, F7, FC3, C3, C5, P3, P7, P9, PO7, PO3, O1, Oz, Pz, CPz, FP2, Fz, F4, F8, FC4, FCz, Cz, C4, C6, P4, P8, P10, PO8, PO4, O2) which was low-pass filtered using a fifth order sinc filter with a half-power cutoff at 204.8 Hz and then digitized at 1024 Hz with 24 bits of resolution. There were also three more recorded signals the horizontal electrooculogram (HEOG) from electrodes placed lateral to the external canthus of each eye and the vertical electrooculogram (VEOG) from an electrode placed below the right eye, which were not used on this work. This dataset served the same purpose as the BCI-DOUBLE-ERRP.

4.3 Data Pre-Processing

As previously stated, even though the goal of transfer learning is to be able to use the same model to datasets with differences among them, reducing these will increase the chances of success. As seen on table 4.1, there are some which need to be handle in order to be able to use the model, like the number of channels 4.6 which are the model inputs, and others, like sample rate and frequency filter, which can easily be matched using signal processing functions. Besides these manipulations, the datasets also come as raw EEG signal which requires to be processed into epochs and correctly labled as target and non-target for training purposes.

	BCIAUT-P300	**BCI-DOUBLE-ERRP**	**ERPCORE**
Subjects	15	8	40
Sessions	7	2	1
Events (probability)	8 (p = 0.125)	28 (p = 0.036)	5 (p = 0.2)
Training Examples per Subject	11200	2520	200
Training Targets (Non-Targets)	1400 (98000)	90 (2430)	40 (160)
Training/Test Set Separation	Yes	Yes	No
Number of Channels	8	12	30
Channels	C3, Cz, C4, CPz, P3, Pz, P4, POz	C3, Cz, C4, CPz, P3, Pz, P4, POz, Fz, PO7, PO8 ,Oz	C3, Cz, C4, CPz, P3, Pz, P4, FP1, F3, F7, FC3, C5, P7, P9, PO7, PO3, O1, Oz, FP2, Fz, F4, F8, FC4, FCz, C6, P8, P10, PO8, PO4, O2
Sample Rate	250Hz	256Hz	1024Hz
Frequency Filter	2 - 30 Hz	0.1 - 100 Hz	None
Stimulus Duration	100ms	75ms	200ms
Inter-Stimuli Interval	200ms	75ms	1200-1400ms

Table 4.1: Comparison table between the three different datasets.

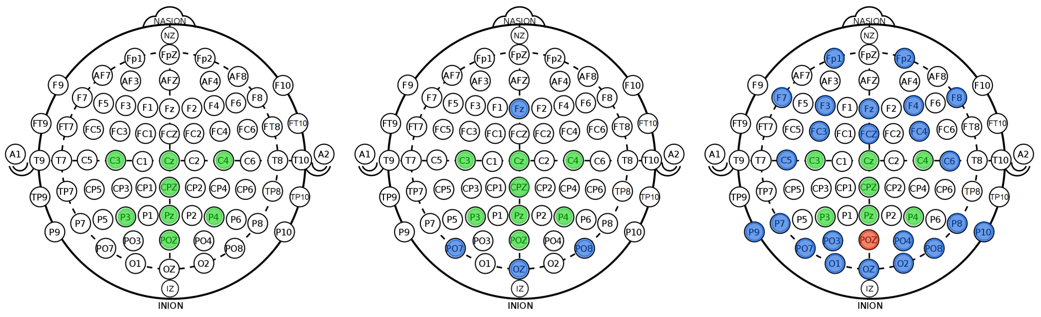

Figure 4.6: EEG cap electrodes placement [65] from left to right: BCIAUT-P300, BCI-DOUBLE-ERRP and ERPCORE. Green circles representing the matching channels with BCIAUT-P300, the blue circles the non-matching and the red circles the missing.

4.3.1 BCI-DOUBLE-ERRP

BCI-DOUBLE-ERRP pre-processing is done by firstly loading the raw EEG signal and applying a bandpass filter 2 to 30 Hz to match the BCIAUT-P300, which is possible as the the original signal is filtered from 0.1 to 100 Hz. Afterwards, the epochs are extracted and labeled, creating 1 second epochs with the stimulus at 0 seconds of each epoch. Finally, the sampling rate is downsampled to match from 256 Hz to 250 Hz using the band-limited sinc interpolation method for sampling rate conversion [66] and the EEG signal is standardized using the mean and standard deviation values. Regarding the channels, as this dataset has the same 8 channels used on BCIAUT-P300 (C3, Cz, C4, CPz, P3, Pz, P4, POz), all the other channels are disregarded.

4.3.2 ERPCORE

As for the ERPCORE, the MNE library [59] is mostly used for the pre-processing. Just like the previous dataset, a bandpass filter is applied between 2 and 30 Hz and the sample rate downsampled from 1024 Hz to 250 Hz. Following the pre-processing done on the literature, the EEG signal is re-referenced to the average of P9 and P10 channels and 1 second epochs are extracted with stimulus at 200ms (-0.2 to 0.8 stimulus onset) having the time shifted to account for the LCD monitor delay. However, focusing on the channel selection for the model input, as seen on 4.6, the POz channel is missing. To handle this situation three different approaches were tested. First, the missing channel was swapped for no data, an array of zeros which the model would eventually learn to ignore throughout the training. Second, the closest channel, Oz, was used. And third, new data was created through interpolation using the spherical spline method [67], which projects the sensor locations onto a unit sphere and interpolates the signal at the missing sensor locations based on the signals at the surrounding locations.

4.4 Models and Architecture

4.4.1 Decoding Solution Architecture

As mentioned before, the decoding solution used for the P300 classification is based on the winner of the 2019 IFMBE Scientific Challenge [9]. It is a deep learning approach using the architecture seen on table 4.7 with a detailed description regarding the main hyper-parameters, output activation shapes, and a number of trainable hyper-parameters introduced on table 4.2.

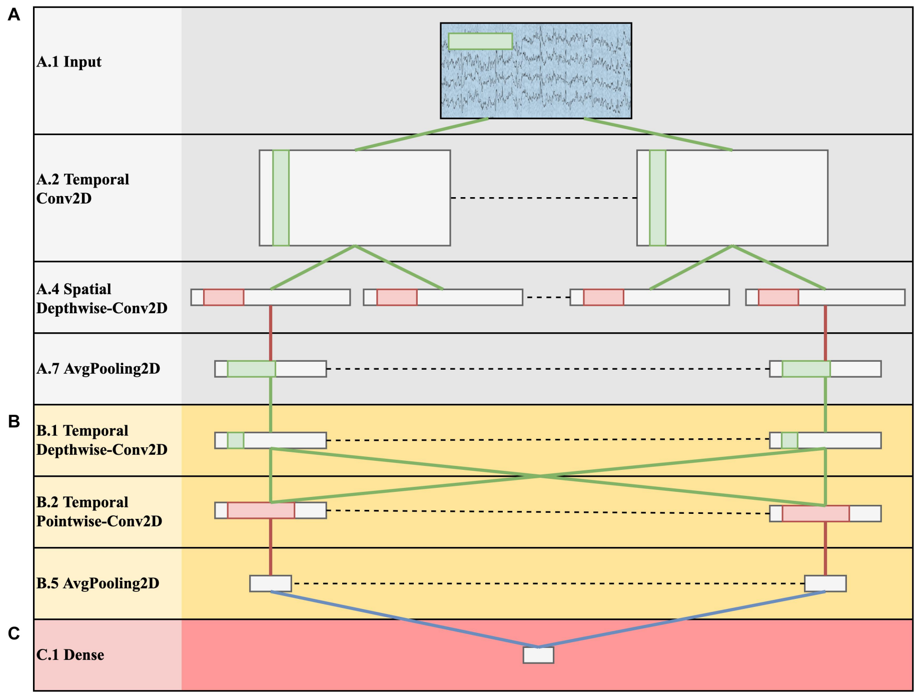
A
A.1 Input
A.2 Temporal Conv2D
A.4 Spatial Depthwise-Conv2D
A.7 AvgPooling2D
B
B.1 Temporal Depthwise-Conv2D
B.2 Temporal Pointwise-Conv2D
B.5 AvgPooling2D
C
C.1 Dense

Figure 4.7: Architecture schematization of the winning solution based on EEGNet. The represented shapes correspond to the output of each layer. Green lines represent convolutional connections, red lines pooling connections, and blue lines dense connections. The CNN is composed by a temporal and spatial feature extractor (A), a summary feature extractor (B) and a classification module (C). [6]

Subnet.	Layer ID	Layer	Hyper-parameters	# pars	Output shape	Activation
A	A. 1	Input		0	(1,8,140)	
	A. 2	Temporal Conv2D	K = 8, F = (1,65), P = (0,32)	520	(8,8,140)	Linear
	A. 3	BatchNorm2D		16	(8,8,140)	
	A. 4	Spacial Depthwise-Conv2D*	D = 2, K = 16, F = (8,1), P = (0,0)	128	(16,1,140)	Linear
	A. 5	BatchNorm2D		32	(16,1,140)	
	A. 6	Activation		0	(16,1,140)	Exponential Linear Units (ELU)
	A. 7	AvgPooling2D	F = (1,4)	0	(16,1,35)	
	A. 8	Dropout	p = 0.25	0	(16,1,35)	
B	B. 1	Temporal Depthwise-Conv2D	D = 1, K = 16, F = (1,17), P = (0,8)	272	(16,1,35)	Linear
	B. 2	Temporal Pointwise-Conv2D	K = 16, F = (1,1), P = (0,0)	256	(16,1,35)	Linear
	B. 3	BatchNorm2D		32	(16,1,35)	
	B. 4	Activation		0	(16,1,35)	ELU
	B. 5	AvgPooling2D		0	(16,1,4)	
	B. 6	Dropout	p = 0.25	0	(16,1,4)	
C	C. 1	Dense	N = 2	130	(2)	Linear
	C. 2	Activation		0	(2)	Softmax

Table 4.2: CNN hyper-parameters configuration. K and F are the numbers and the size of kernels respectively. P is padding size, D the depth multiplier, N the number of neurons in the dense layer and finally p the dropout rate. *Unitary kernel max-norm constraint.[6]

It is a CNN created from an adaptation of EEGNet [11] trained to discriminate between P300 and non-P300 classes. The full network is composed of 3 main subnetworks, performing different operations on the input. The first 3 layers after the input work as a temporal and spatial feature extractor that learns meaningful temporal and spatial filters, which is followed by other three layers that act as a summary feature extractor that learns to extract temporal summaries for each feature map of the subnetwork A individually. Finally, a classification module that finalizes the classification task based on the output of the subnetwork B. The input is a 2-D representation composed by the EEG channels along one dimension (spatial dimension) and time steps along the other dimension (temporal dimension).

From a transfer learning usability point of view, it is important to be aware of which layers have trainable weights and bias, which will be the values used to initialize, or even use as definitive values for new models. Therefore, concerning the selected model, there are eight different layers with trainable weights and bias as seen in table 4.3.

Layer ID	Layer	Trainable Values
A. 2	Temporal Conv2D	Weight
A. 3	BatchNorm2D	Weigh; Bias
A. 4	Spacial Depthwise-Conv2D	Weight
A. 5	BatchNorm2D	Weight; Bias
B. 1	Temporal Depthwise-Conv2D	Weight
B. 2	Temporal Pointwise-Conv2D	Weight
B. 3	BatchNorm2D	Weight; Bias
C. 1	Dense	Weight; Bias

Table 4.3: Layers with each trainable values, IDs matching table 4.2

4.4.2 Models and Hyper-Parameters

In order to fully test and compare the potential of the transfer learning, several different models had to be created, all using the same, already described architecture, but from different datasets and with different hyper-parameters.

Regarding hyper-parameters, there will be three different kind, each one with the best combination found for each dataset. The first one uses the hyper-parameters shown on table 4.2, as it is the optimal for the BCIAUT-P300, as seen on the previous work and replicated and validated here. The second, uses the hyper-parameters shown on table 4.4 which was the best combination found for the BCI-DOUBLE-ERRP dataset. The third model uses the hyper-parameters on table 4.5, as they were the ones which gave the best results when training the ERPCORE dataset. The training methodology and the datasets used for each model will be addressed on a later training and testing topic.

Subnet	Layer ID	Layer	Hyper-Parameters
A	A. 2	Temporal Conv2D	number kernels = 8, kernel size = (34, 1), padding size = (17, 0)
	A. 4	Spacial Depthwise-Conv2D	depth multiplier = 8, number kernels = 64, kernel size = (1, 8), padding size = (0, 0)
	A. 7	AvgPooling2D	kernel size = (4, 1)
	A. 8	Dropout	dropout rate = 0.55
B	B. 1	Temporal Depthwise-Conv2D	depth multiplier = 1, number kernels = 64, kernel size = (17, 1), padding size = (8, 0)
	B. 2	Temporal Pointwise-Conv2D	number kernels = 64, kernel size = (1, 1) ,padding size = (0, 0)
	B. 6	Dropout	dropout rate = 0.55
C	C. 1	Dense	number neurons = 2

Table 4.4: Model hyper-parameters optimized for the BCI-DOUBLE-ERRP dataset.

Subnet	Layer ID	Layer	Hyper-Parameters
A	A. 2	Temporal Conv2D	number kernels = 4, kernel size = (40, 1), padding size = (20, 0)
	A. 4	Spacial Depthwise-Conv2D	depth multiplier = 4, number kernels = 16, kernel size = (1, 8), padding size = (0, 0)
	A. 7	AvgPooling2D	kernel size = (4, 1)
	A. 8	Dropout	dropout rate = 0.4
B	B. 1	Temporal Depthwise-Conv2D	depth multiplier = 1, number kernels = 16, kernel size = (13,1), padding size = (6, 0)
	B. 2	Temporal Pointwise-Conv2D	number kernels = 16, kernel size = (1, 1) ,padding size = (0, 0)
	B. 6	Dropout	dropout rate = 0.4
C	C. 1	Dense	number neurons = 2

Table 4.5: Model hyper-parameters optimized for the ERPCORE dataset.

4.5 Training and Testing

4.5.1 Training Parameters and Data

To be able to provide comparable results, all models were trained with the following training parameters:

- Maximum of 250 epochs.
- Adam optimizer with 0.001 learning rate.
- F-score as the test metric.
- 20% of the training dataset used as validation.
- Batch size of 64.

Also, the random seed was set and fixed for every run. As for the data selected from the datasets, from BCIAUT-P300 all data from the 7 sessions of the 15 possible subjects was used. From BCI-DOUBLE-ERRP it was used the data from 2 sessions of 7 subjects from the possible 8, ignoring subject 2 (S2) for using passive electrodes. On the ERPCORE dataset, 34 subjects were chosen from the possible 40, due to high artifact presence on the recorded signal on subjects 6, 9, 19, 30, 35 and 40, according to the article [63].

For both BCI-DOUBLE-ERRP and ERPCORE dataset, all models created are trained using a cross-session subject specific approach. This indicates that one model is trained for each subject using only it's data from every session available. Contrary to this, as the BCIAUT-P300 is used to create the models to transfer the knowledge, all the data on the dataset is used for training each model.

4.5.2 Models to Transfer

To later apply transfer learning there is the need to previously train models so their trained weights can then be used for this goal. The models used to transfer the knowledge will all come from BCIAUT-P300 dataset. Although this dataset has the training and test sets separated, to create these models, all sets, the full 466000 examples, will be used in training. Having no testing set means there will be no testing metrics to evaluate the model, this is not an issue has only the trained weights are important from these models. Three different types of these models, seen on table 4.6, each one created with different hyper-parameters, one using the hyper-parameters optimized for BCIAUT-P300 , displayed on table 4.2, another one using the hyper-paramteres optimized for the BCI-DOUBLE-ERRP, as seen on table 4.4, and the last one using the the hyper-parameters optimized for ERPCORE, shown on table 4.5 . This was necessary as the change on the hyper-parameters changed the number of trainable parameters and to be able to apply the transfer learning these need to match.

Mode ID	Dataset	Hyper-parameters
BCIAUT_T	BCIAUT-P300*	BCIAUT-P300 optimized 4.2
DERRP_T	BCIAUT-P300*	BCI-DOUBLE-ERRP optimized 4.4
ERP_T	BCIAUT-P300*	ERPCORE optimized 4.5

Table 4.6: Models trained to be used for transfer knowledge. *BCIAUT-P300 dataset used as a whole to train, no training and testing separation.

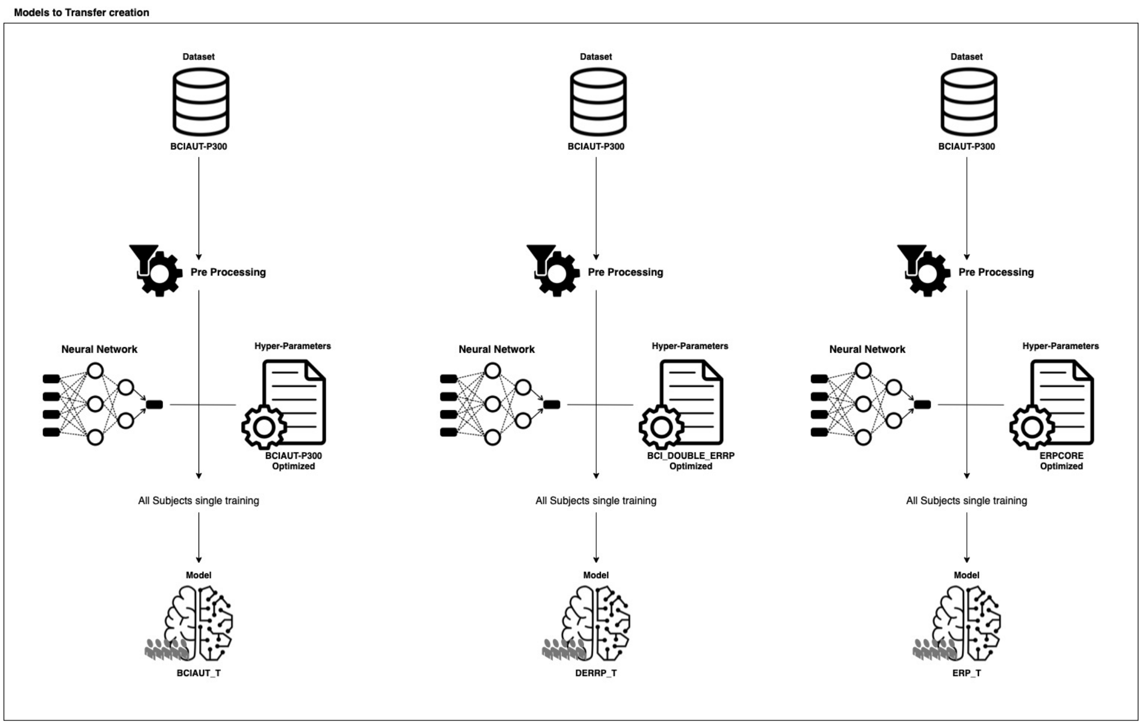

Figure 4.8: Creation of models to use as base for transfer knowledge. From left to right: BCIAUT_T, the model used to initialize models trained with the BCIAUT-P300 optimized hyper-parameters. DERRP_T, the model used to initialize models trained with the BCI-DOUBLE-ERRP optimized hyper-parameters. ERP_T, the model used to initialize models trained with the ERPCORE optimized hyper-parameters

4.5.3 Baseline Models

All models were subject-specific trained using a cross-session approach where all the data from all sessions from the specific subject was used to train and test the model. For each of the relevant datasets, BCI-DOUBLE-ERRP and ERPCORE, two baseline models were created, presented on table 4.7. One using the optimized hyper-parameters for BCIAUT-P300 and the other with the optmized hyper-parameters for the target dataset. These models provide the baseline metrics for comparison with models created using transfer learning.

The strategy used to create these models can be seen on the detailed scheme presented on Figures 4.9 for the BCI-DOUBLE-ERPP, and 4.10 for ERPCORE.

Mode ID	Dataset	Hyper-parameters
DERRP_baseline	BCI-DOUBLE-ERRP	BCIAUT-P300 optimized 4.2
DERRP_baseline_optimized	BCI-DOUBLE-ERRP	BCI-DOUBLE-ERRP optimized 4.4
ERP_baseline	ERPCORE	BCIAUT-P300 optimized 4.2
ERP_baseline_optimized	ERPCORE	RPCORE optimized 4.5

Table 4.7: Baseline models training parameters.

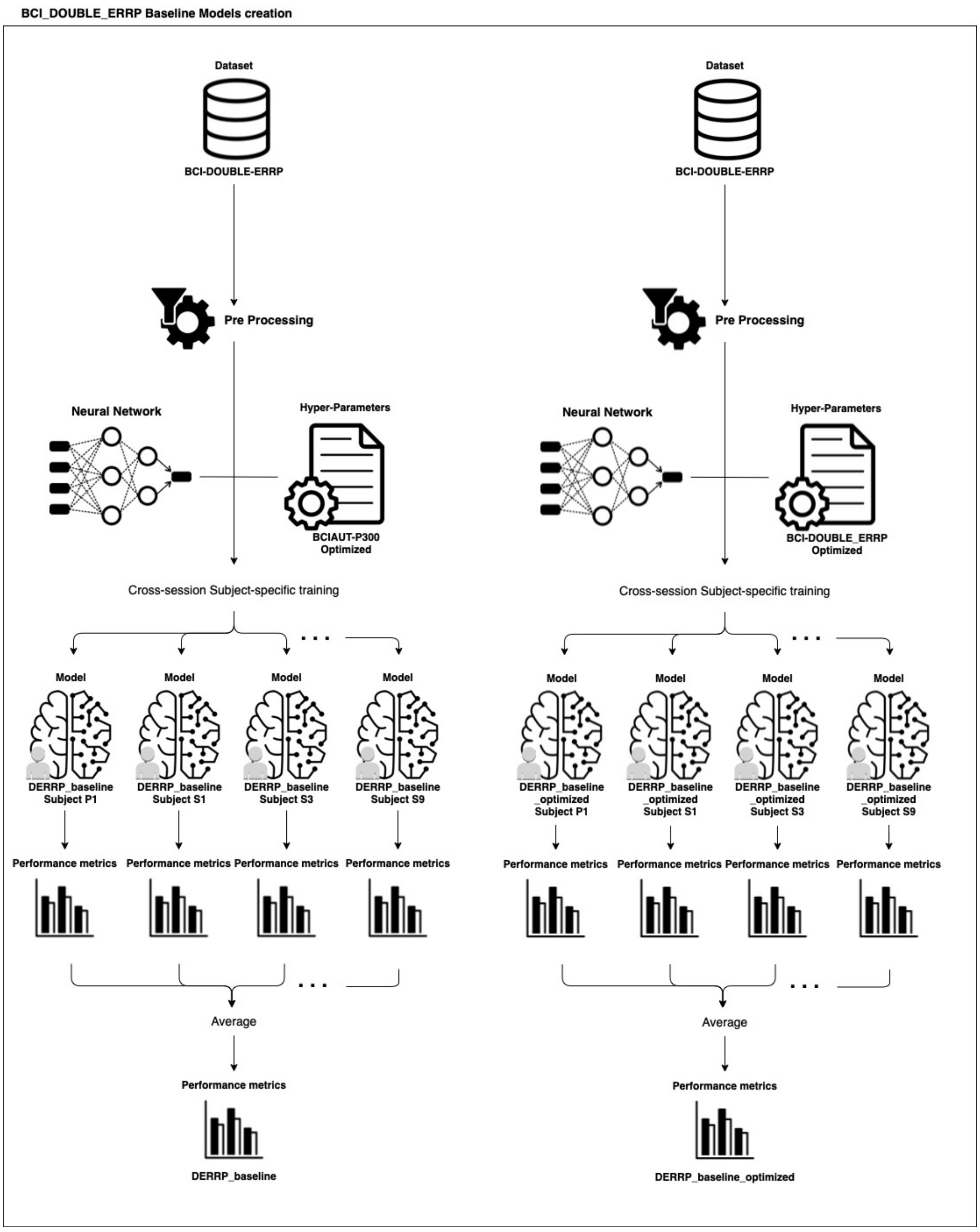
BCI_DOUBLE_ERRP Baseline Models creation
Dataset
BCI-DOUBLE-ERRP
Pre Processing
Neural Network
Hyper-Parameters
BCIAUT-P300 Optimized
BCI-DOUBLE_ERRP Optimized
Cross-session Subject-specific training
Model
DERRP_baseline Subject P1
DERRP_baseline Subject S1
DERRP_baseline Subject S3
DERRP_baseline Subject S9
DERRP_baseline _optimized Subject P1
DERRP_baseline _optimized Subject S1
DERRP_baseline _optimized Subject S3
DERRP_baseline _optimized Subject S9
Performance metrics
Average
DERRP_baseline
DERRP_baseline_optimized

Figure 4.9: Creation of BCI-DOUBLE-ERRP Baseline models. On the left, the baseline model created using the BCIAUT-P300 optimized hyper-parameters, on the right, the one trained with the dataset optimized hyper-parameters.

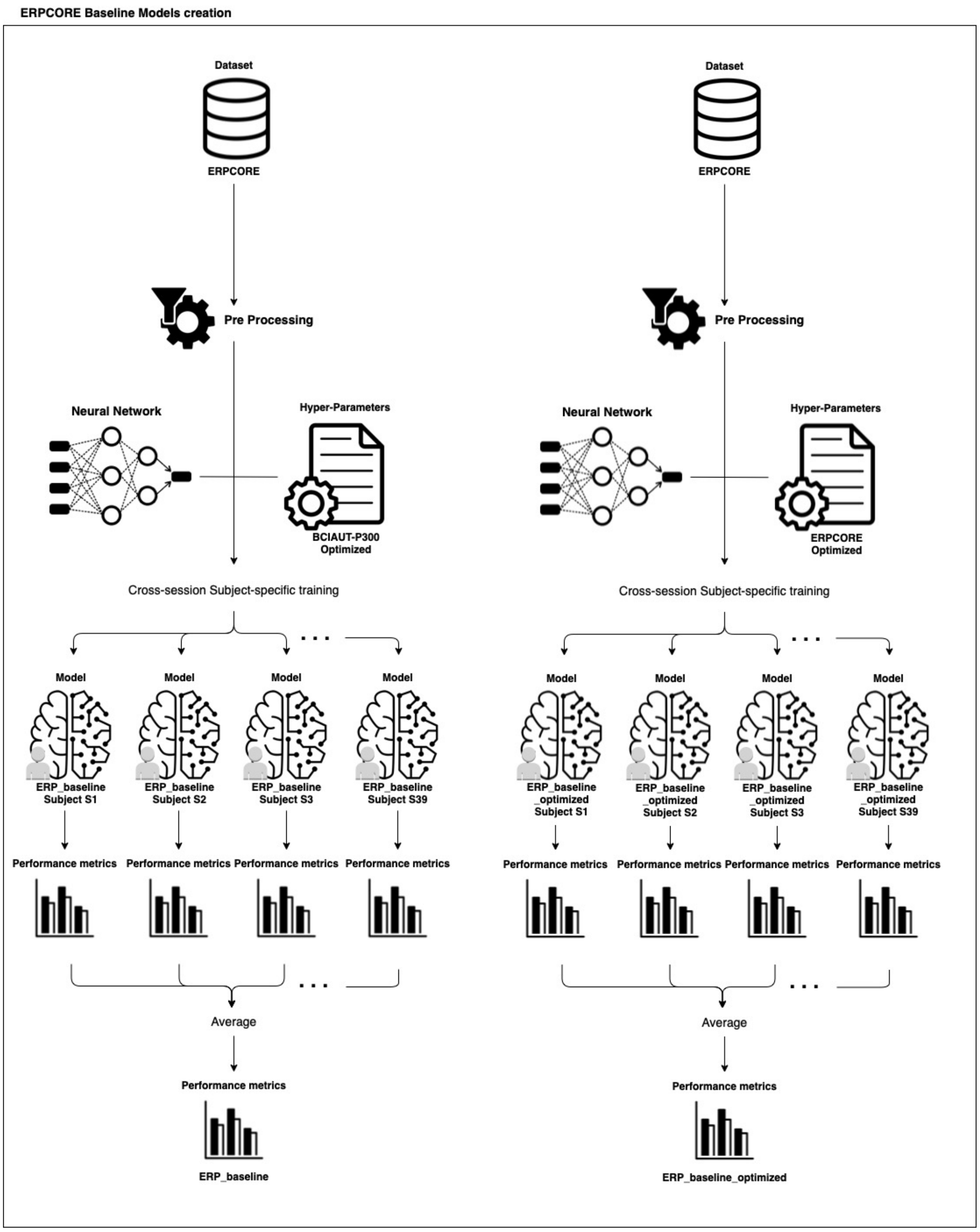

Figure 4.10: Creation of ERPCORE Baseline models. On the left, the baseline model created using the BCIAUT-P300 optimized hyper-parameters, on the right, the one trained with the ERPCORE optimized hyper-parameters.

4.5.4 Transfer Learning

Similarly to the baseline models training, all models were subject-specific trained using a cross-session approach where all the data from all sessions from the specific subject was used to train and test the model. To implement transfer learning, at the start of the training of a new model, instead of initializing each trainable weight randomly, the corresponding weights previously trained on the models *BCIAUT_ T*, *DERRP_ T* and *ERP_ T* are used as initialization. Subsequently, the layers are chosen to be frozen, meaning the trainable parameters will not be changed throughout the training, or unfrozen. In this work, the layers will be frozen top to bottom, to compare which combination has better results.

In order to ease referencing all these different new models throughout the rest of this work, a nomenclature was created. The models trained with transfer learning will start with the name of dataset to which they belong to (DERRP for BCI-DOUBLE-ERRP and ERP for ERPCORE), followed by the word *optimized* when using the hyper-paramenters optimized for that dataset, and ending with a number (from 0 to 7), which refers to the number of frozen layers, presented on table 4.8. For example, a model for the ERPCORE dataset, using BCIAUT-P300 hyper-paramaters with the whole subnet A frozen, is named *ERP_ 4*.

Model ID	Layers Frozen
x_x_0	None
x_x_1	A. 2 Temporal Conv2D
x_x_2	A. 2 Temporal Conv2D A. 3 BatchNorm2D
x_x_3	A. 2 Temporal Conv2D A. 3 BatchNorm2D A. 4 Spacial Depthwise-Conv2D
x_x_4	A. 2 Temporal Conv2D A. 3 BatchNorm2D A. 4 Spacial Depthwise-Conv2D A. 5 BatchNorm2D
x_x_5	A. 2 Temporal Conv2D A. 3 BatchNorm2D A. 4 Spacial Depthwise-Conv2D A. 5 BatchNorm2D B. 1 Temporal Depthwise-Conv2D
x_x_6	A. 2 Temporal Conv2D A. 3 BatchNorm2D A. 4 Spacial Depthwise-Conv2D A. 5 BatchNorm2D B. 1 Temporal Depthwise-Conv2D B. 2 Temporal Pointwise-Conv2D
x_x_7	A. 2 Temporal Conv2D A. 3 BatchNorm2D A. 4 Spacial Depthwise-Conv2D A. 5 BatchNorm2D B. 1 Temporal Depthwise-Conv2D B. 2 Temporal Pointwise-Conv2D B. 3 BatchNorm2D

Table 4.8: Model nomenclature used for number of frozen layers on the models trained with transfer learning.

The whole creation process for the transfer models can be seen step by step on Figures 4.11 and 4.12, for the BCI-DOUBLE-ERRP and ERPCORE datasets, respectively.

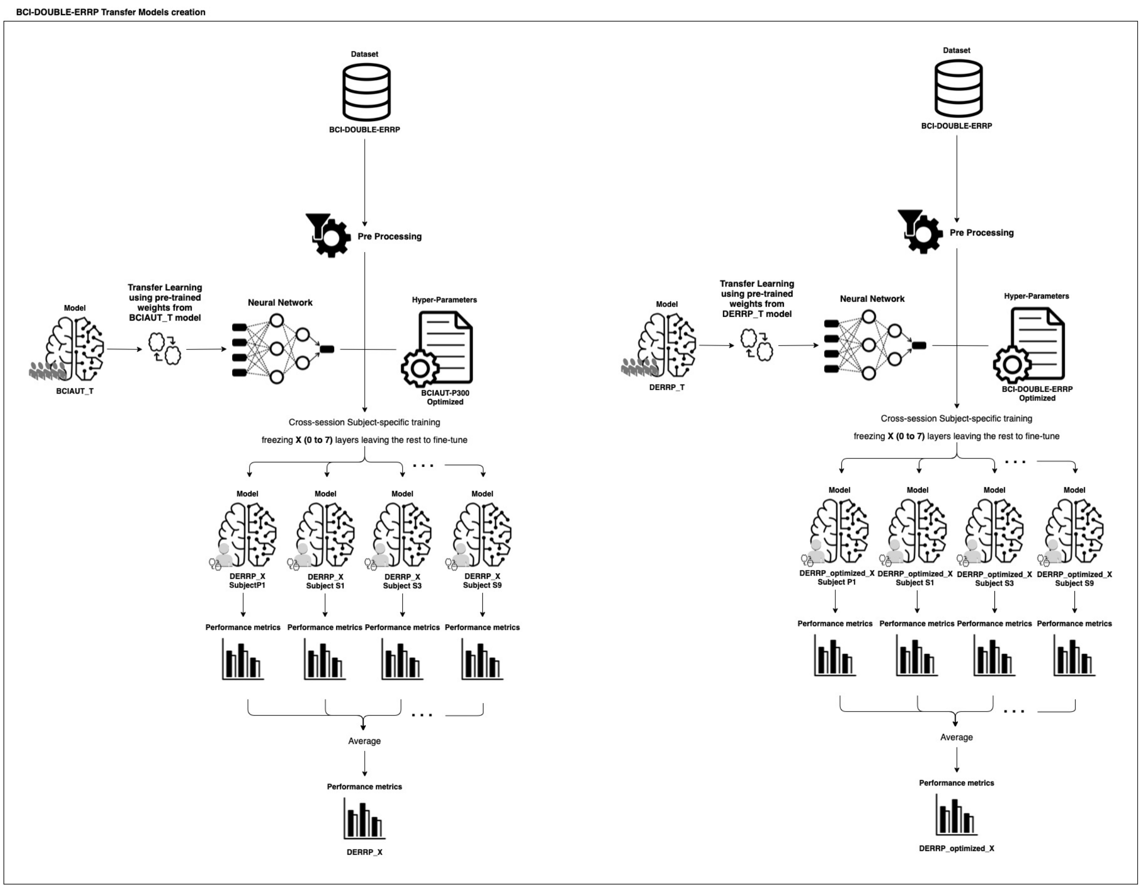
BCI-DOUBLE-ERRP Transfer Models creation
Dataset
BCI-DOUBLE-ERRP
Pre Processing
Model
BCIAUT_T
Transfer Learning using pre-trained weights from BCIAUT_T model
Neural Network
Hyper-Parameters
BCIAUT-P300 Optimized
Cross-session Subject-specific training
freezing X (0 to 7) layers leaving the rest to fine-tune
Model
DERRP_X SubjectP1
DERRP_X Subject S1
DERRP_X Subject S3
DERRP_X Subject S9
Performance metrics
Average
Performance metrics
DERRP_X
Dataset
BCI-DOUBLE-ERRP
Pre Processing
Model
DERRP_T
Transfer Learning using pre-trained weights from DERRP_T model
Neural Network
Hyper-Parameters
BCI-DOUBLE-ERRP Optimized
Cross-session Subject-specific training
freezing X (0 to 7) layers leaving the rest to fine-tune
Model
DERRP_optimized_X Subject P1
DERRP_optimized_X Subject S1
DERRP_optimized_X Subject S3
DERRP_optimized_X Subject S9
Performance metrics
Average
Performance metrics
DERRP_optimized_X

Figure 4.11: Creation of BCI-DOUBLE-ERRP Transfer Learning models. On the left, the models created using the BCIAUT_T pre-trained model, on the right, the DERRP_T pre-trained model.

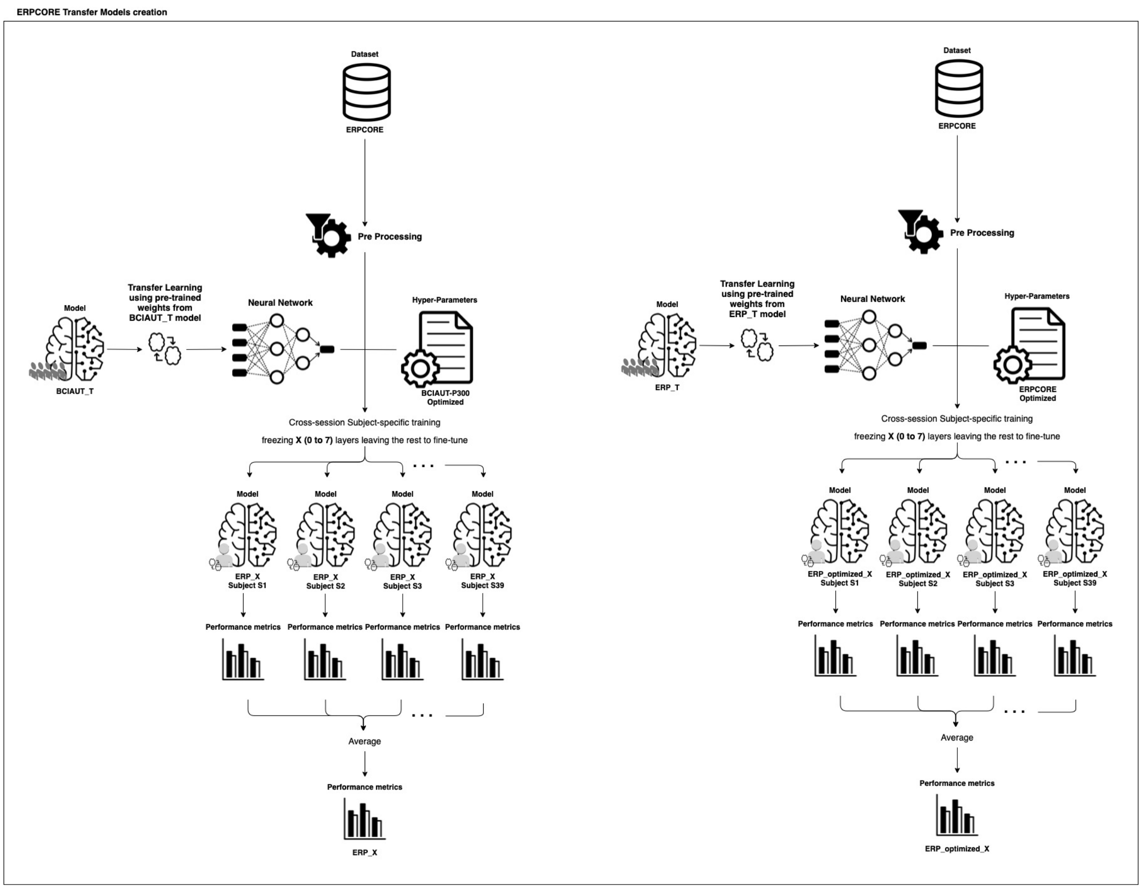
ERPCORE Transfer Models creation
Dataset
ERPCORE
Pre Processing
Model
BCIAUT_T
Transfer Learning using pre-trained weights from BCIAUT_T model
Neural Network
Hyper-Parameters
BCIAUT-P300 Optimized
Cross-session Subject-specific training
freezing X (0 to 7) layers leaving the rest to fine-tune
Model
ERP_X Subject S1
ERP_X Subject S2
ERP_X Subject S3
ERP_X Subject S39
Performance metrics
Average
Performance metrics
ERP_X
Dataset
ERPCORE
Pre Processing
Model
ERP_T
Transfer Learning using pre-trained weights from ERP_T model
Neural Network
Hyper-Parameters
ERPCORE Optimized
Cross-session Subject-specific training
freezing X (0 to 7) layers leaving the rest to fine-tune
Model
ERP_optimized_X Subject S1
ERP_optimized_X Subject S2
ERP_optimized_X Subject S3
ERP_optimized_X Subject S39
Performance metrics
Average
Performance metrics
ERP_optimized_X

Figure 4.12: Creation of ERPCORE Transfer Learning models. On the left, the models created using the BCIAUT_T pre-trained model, on the right, the ERP_T pre-trained model.

4.5.5 Testing

To be able to make an analyses and comparison between the models created each of one needed to be tested in order to extract the testing metrics. In this work the metrics chosen to compare the accuracy were the F-score and balanced accuracy. The F-score (F_1) is a test metric calculated from the precision and recall (also known as sensitivity) of the test by the following formula:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

With precision being the ration between the true positives and the sum of the true and false positives, and recall the ration between true positives and the sum of the true positives and false negatives.

Balanced accuracy is calculated using sensitivity and specificity (also called true negative rate) of the test by the formula below:

$$BalancedAccuracy = \frac{\text{Sensitivity} + \text{Specificity}}{2}$$

With specificity being the ration between true negatives and the sum of true negatives and false positives.

Regarding the data used to test the models, for BCI-DOUBLE-ERRP is straightforward as the dataset has already a separation for training set and test set. All the metrics for each subject-specific models are gathered and the mean and standard deviation is calculated to obtain the final metric for the model. However, ERPCORE is a single session dataset, without any type of separation. In this situation, a k-fold cross validation is used to test the models. This procedure can be described by the following steps:

- Shuffle the dataset randomly.
- Split the dataset into k groups
- For each unique group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
  - Fit a model on the training set and evaluate it on the test set
  - Retain the evaluation score and discard the model
- Summarize the skill of the model using the sample of model evaluation scores

It was selected 10 as the number of folds, and to get the final metrics all the folds metrics for each subject-specific models are gathered and the mean and standard deviation is calculated.

Lastly, to compute the performance of each different approach, since the models created are subject-specific, meaning one model is trained for each subject, all the testing metrics from each of those models for a particular approach were averaged and compared.

This page is intentionally left blank.

Chapter 5

Results and Analysis

In this chapter the results obtained from the experiment are shown. All the models have their performance evaluated and compared through the gathered metrics.

5.1 Results

5.1.1 BCI-DOUBLE-ERRP

Baseline Models

On image 5.1 it is shown the comparison between the performance of the two baseline models for the BCI-DOUBLE-ERRP dataset.

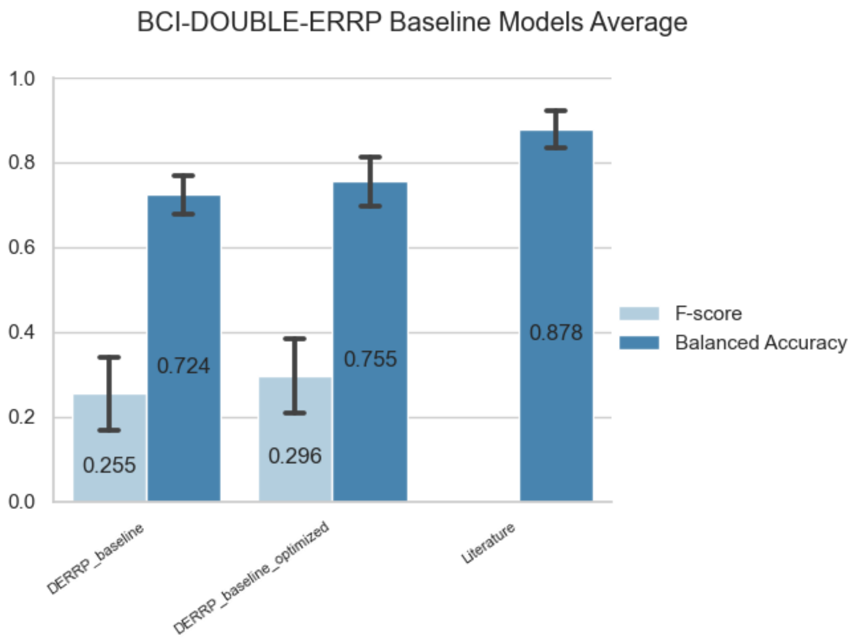

Figure 5.1: BCI-DOUBLE-ERRP baseline models' performance. The values shown are the average of the subject-specific models' metrics.

Table 5.1 shows a more in depth comparison between the baseline models created from our method with the literature solution.

Subject	DERRP_Baseline	DERRP_Baseline_optimized	Literature[61]
P1	0.668	0.685	0.797
S1	0.669	0.742	0.832
S3	0.763	0.840	0.92
S4	0,.720	0.727	0.805
S5	0.714	0.687	0.892
S6	0.730	0.772	0.919
S9	0.804	0.831	0.897
Mean	0.724	0.755	0.866
Standard Deviation	0.049	0.0623	0,053

Table 5.1: BCI-DOUBLE-ERRP subject-specific models' performance.

Transfer Learning

Images 5.2 and 5.3 show the performance metrics for every combination of frozen layers on the transfer models using the BCIAUT-P300 optimized hyper-parameters and the BCI-DOUBLE-ERRP optimized hyper-parameteres, respectively. Subject-specific metrics can be found in Appendix A, in tables A.1 and A.2 for the BCIAUT-P300 optimized models and tables A.3 and A.4 for the BCI-DOUBLE-ERRP optimized models.

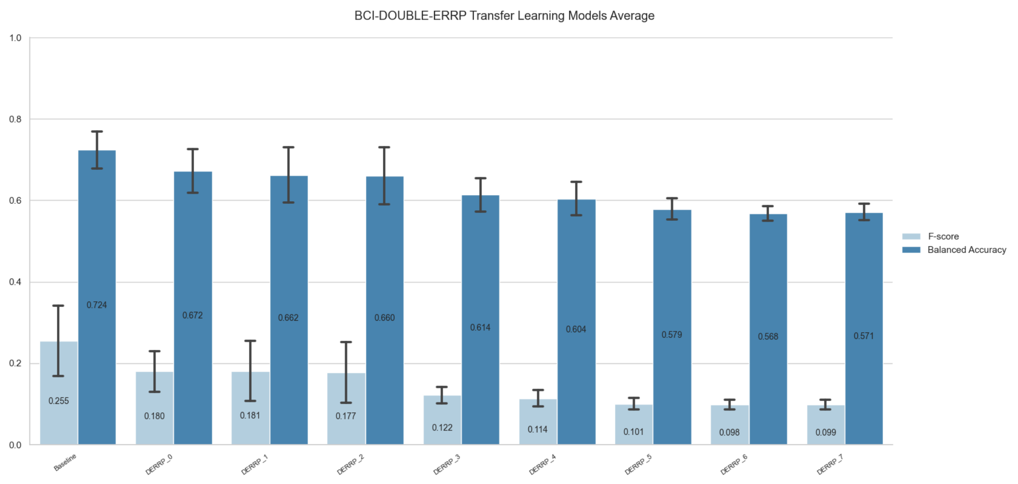

Figure 5.2: BCI-DOUBLE-ERRP Transfer Learning performance. Values shown are computed by averaging each transfer learning configuration models' subject-specific metrics.

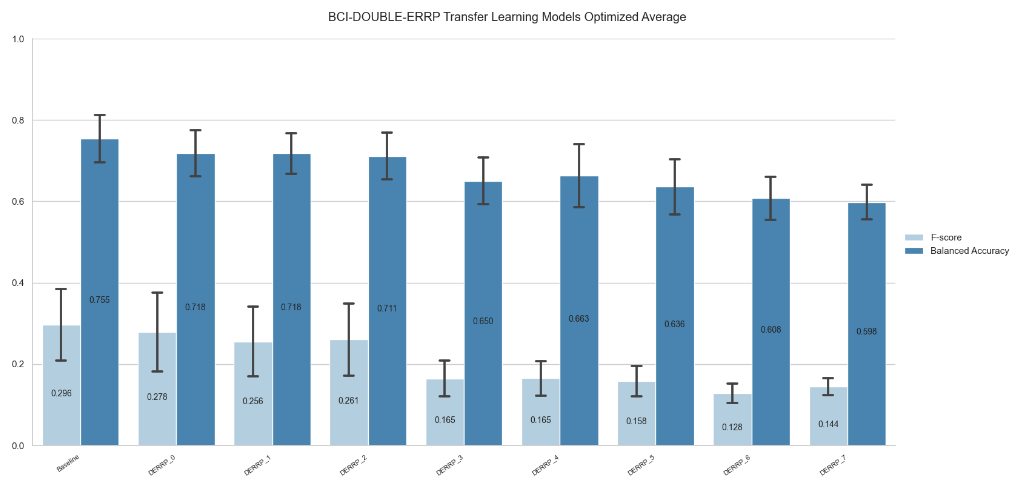

Figure 5.3: BCI-DOUBLE-ERRP Transfer Learning performance for the dataset's optimized hyper-parameters. Values shown are computed by averaging each transfer learning configuration models' subject-specific metrics.

5.1.2 ERPCORE

Baseline Models

Regarding the ERPCORE dataset, there was a first test concerning which would be the best approach to solve the missing channel issue, with the results of the three methods sampled on figure 5.4.

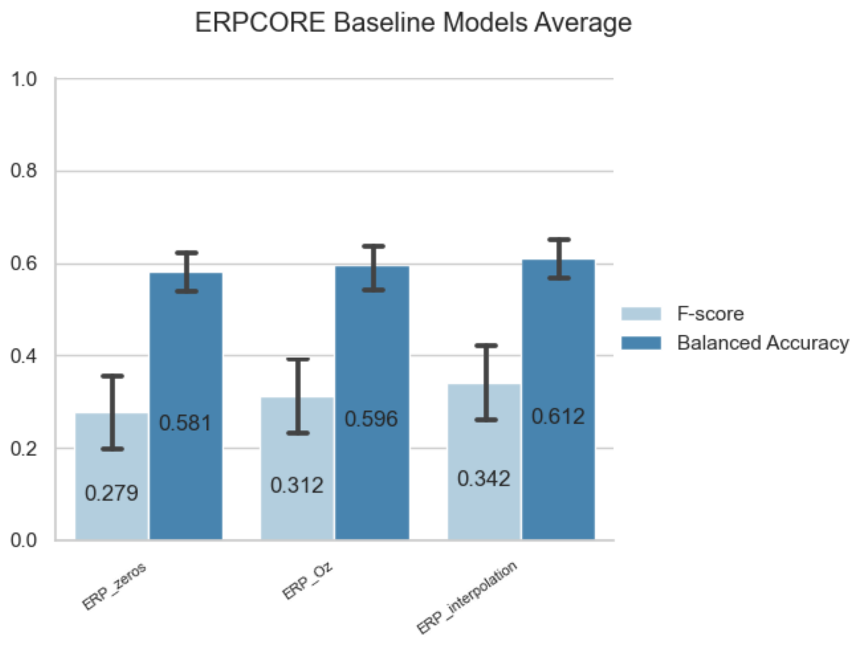

Figure 5.4: Comparison between the three different approaches tested to replace the missing channel. From left to right: Using an array of zeros, using the Oz channel and using spherical spline interpolation.

Using the method with the best outcome from the previous results, the baseline models were created using both the hyper-parameters optimized for the BCIAUT-P300 dataset and optimized for the ERPCORE, with the results displayed on figure 5.5. The values presented are the mean of the mean of every fold from each subject-specific model.

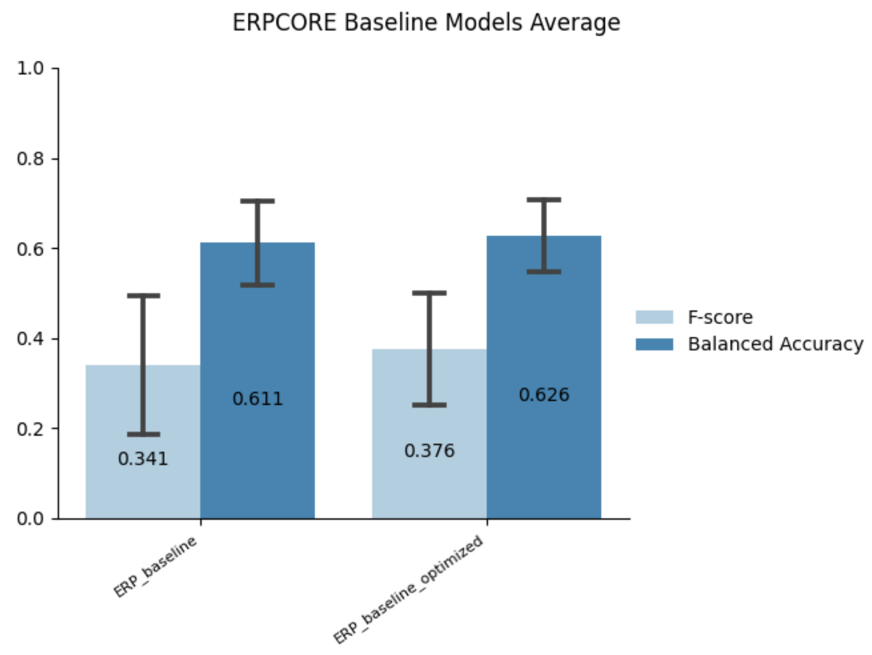

Figure 5.5: Performance metrics for both baseline models for the ERPCORE dataset. Values obtained by averaging the subject-specifics' models' metrics.

Transfer Learning

To test the performance of transfer learning on the ERPCORE dataset, the same 8 combinations of frozen layers were tested both with the optimized hyper-parameters and with the original ones, with the results showing in figures 5.7 and 5.6, respectively. Subject-specific metrics can be found in Appendix B, in tables B.1 and B.2 for the BCIAUT-P300 optimized models and tables B.3 and B.4 for the BCI-DOUBLE-ERRP optimized models.

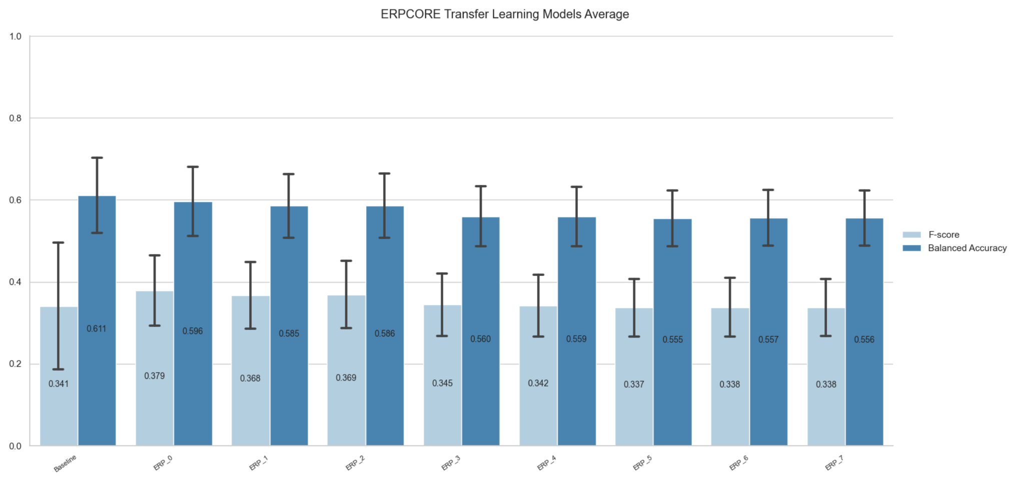

Figure 5.6: ERPCORE Transfer Learning performance for the BCIAUT-P300 optimized hyper-parameters. Values shown are computed by averaging each transfer learning configuration models' subject-specific metrics.

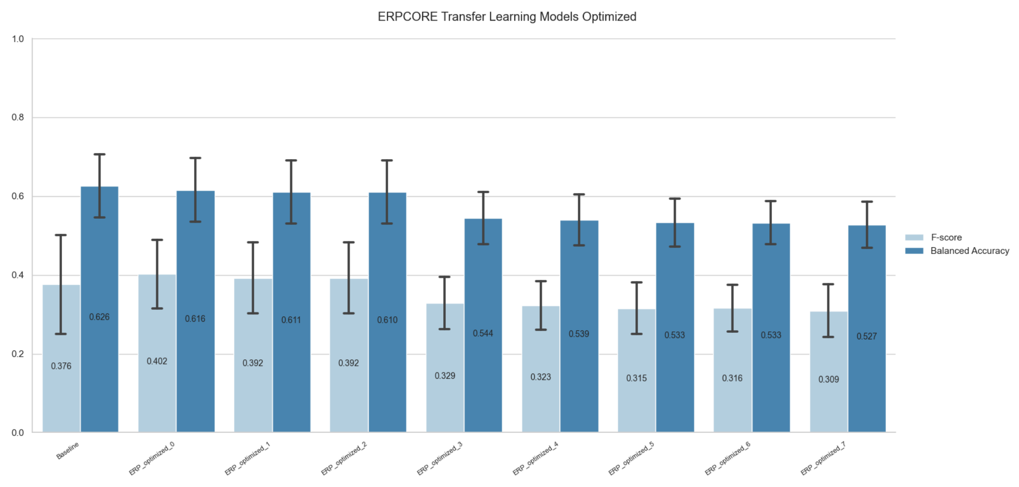

Figure 5.7: Transfer Learning performance for the dataset's optimized hyper-parameters. Values shown are computed by averaging each transfer learning configuration models' subject-specific metrics.

5.2 Analysis and Discussion

Starting with the results collected from the models trained and tested with the BCI-DOUBLE-ERRP dataset, unfortunately, the outcome was not as bright as it was expected. Focusing on the baseline models (figure 5.1), not only the model trained from the BCIAUT-P300 hyper-parameters proved to be far less effective than the one from the literature, scoring an average of less than 14% balanced accuracy (seen on table 5.1), but the architecture showed to be hard to optimize, with the best effort to only reach an average balanced accuracy of 75.5%, still not being able to reach the 86.6% achieved on the dataset paper. One explanation to these results is the fact that the dataset is very small, and extremely unbalanced, with only one target for every 27 non-targets, which are known factors to hinder the implementation of deep learning approaches, like the one used on this work [68], contrary to the more classical approach taken on the dataset's paper: a Bayes classifier with previous feature extraction.

Moving on to the transfer learning implementation on BCI-DOUBLE-ERRP (figures 5.2 and 5.3), the performance deteriorated even more, with the best results reached by freezing none, one or two layers, but always being bellow both the baseline performance and the baseline optimized performance. This suggests that the network wasn't able to fine-tune the weights with the knowledge previously learnt from the BCIAUT-P300 dataset which can not just be also explained by the lack of target examples to train but by the differences between the two datasets. By analysing the average signal on the Pz electrode on both datasets, figure 5.8, it is clear the extreme differences between them. These differences might be explained by the inter-stimuli time, where the BCIAUT-P300 has a 200 ms and the BCI-DOUBLE-ERRP only 75 ms, which can be the cause of so many responses on the BCI-DOUBLE-ERRP.

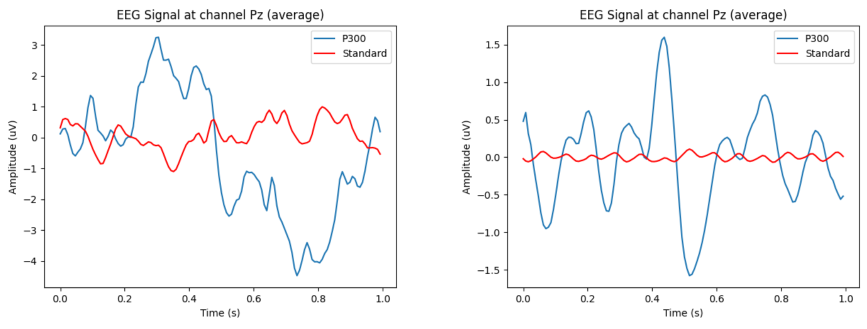

Figure 5.8: Average of the signal on the Pz electrode for the BCIAUT-P300 dataset on the left, and BCI-DOUBLE-ERP dataset on the right.

Focusing on the ERPCORE dataset, the first thing to analyse is the outcome of the three different approaches to solve the missing channel issue, represented on figure 5.4. Although only for 1.6% balanced accuracy and 0.03 F-score, the spherical spline interpolation method is the best solution for this problem. The proximity between the performance of not having any information, by using the zeros array, or by using a different channel, shows the adaptability of the newtwork to be able to ignore the zeros array. The reason behind the success of the interpolation is that, despite the fact that the signal is artificially created, it uses information from the surrounding electrodes which brings more information for the model to use. Looking to figure 5.9, the interpolated POz channel has a better P300 wave signal than the Oz.

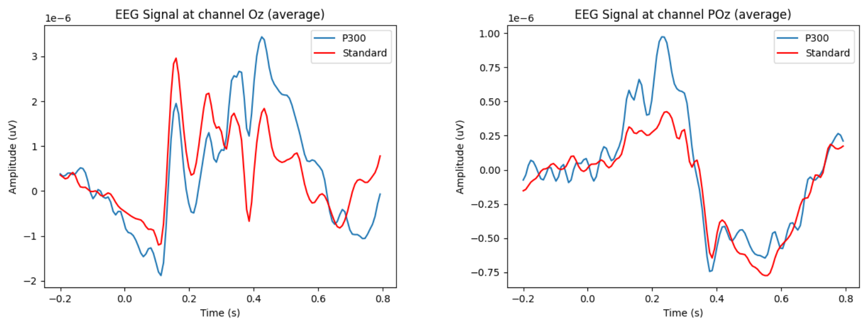

Figure 5.9: Comparison between the average of the signal on the Oz electrode (left) and the artificially created POz with interpolation (right) on the ERPCORE dataset .

As for the baseline models (figure 5.5), the model showed a better response than to what has been seen on the previous dataset, achieving a better performance, whilst still being under 0.05 F-score. This is probably due to this dataset being more balanced, 1 target for every 4 non-targets. Besides that, it was also possible to attain better results by adapting the hyper-parameters to optimize the model to ERPCORE.

Regarding the transfer learning, it also produced some interesting results, being able to achieve a better performance in some models, when compared to the baseline. The models seem to improve the most when initialized with the pre-trained weights and leaving all layers, or the first Temporal Conv2D and BatchNorm2D frozen to be fine-tuned [69], with a clear decline in performance for models which have the rest of the layers frozen. The positive response was mostly seen on the baseline model trained with the original hyper-parameters having the best performance with no layers frozen, improving more than 0.03 in F-score but worsening <1% in balanced accuracy. As for the optimized model, much like the baseline, the improvements were only seen on the F-score, up to 0.026, on the same transfer approaches (no layers frozen, having the first Temporal Conv2D and having both the first Temporal Conv2D and BatchNorm2D frozen) , meaning it improved on correctly classifying targets, but probably started to classify more false positives. The close similarities between the two datasets when it comes to the signal wave (image 5.10) might be the reason for this positive outcome.

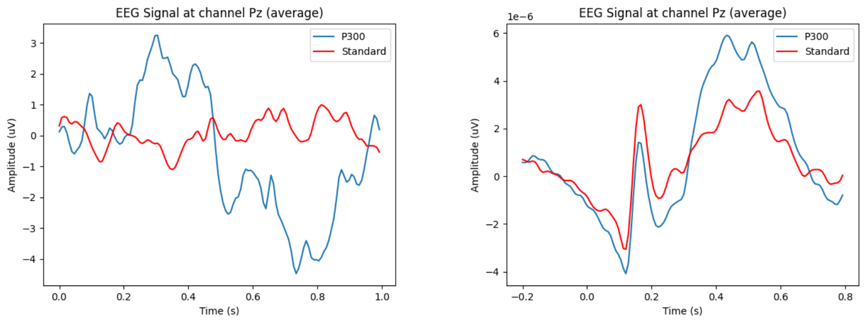

Figure 5.10: Average of the signal on the Pz electrode for the BCIAUT-P300 dataset on the left, and ERPCORE dataset on the right.

This page is intentionally left blank.

Chapter 6

Conclusion and Future Work

Recently, the use of deep neural networks and deep learning, opposed to more classical machine learning methods, has been increasing in every field and the research area of BCIs is no exception. With the hopes of improving the lives of many with numerous life improving medical applications, the decoding and classification of brain signals, like EEG, using deep learning has gathered much attention. Unfortunately, the lack of available data in addition the high inter- and intra-subject variability of brain signals is a hurdle for the successful implementation of these methods.

The main goal of this work was to explore transfer learning techniques, by taking advantage of a large enough dataset, BCIAUT-P300, and an existing deep learning solution for P300 classification, a compact CNN based on EEGNet, and apply them to other P300 datasets, creating robust models capable of performing well. An experiment was set, using the BCIAUT-P300 as the dataset to train the models to be used to transfer knowledge and creating different baseline models for the BCI-DOUBLE-ERRP and ERPCORE datasets, finishing by training and testing different with different combinations of frozen layers.

However, not every result was positive, the models for the BCI-DOUBLE-ERRP showed lack of performance, with a F-score below 0.3 and a balanced accuracy of 75% but always 10% under the dataset paper's, with the models using transfer learning showing any improvements. This was mainly due to the substantial differences between the BCIAUT-P300 and the BCI-DOUBLE-ERRP signals and the latter lack of balance and target examples. Fortunately, ERPCORE showed better results, with models achieving 0.4 F-score and up to 62% balanced accuracy with improvements when using transfer learning and freezing the first convolutional layer. In addition, spherical spline interpolation showed promising results to handle missing electrode problems.

In conclusion, although the results were not extraordinary, it was proven that transfer learning can be useful and applied to EEG decoding with positive outcomes, leaving much room for improvement and subsequent research.

For future work, it would be interesting to try different architectures or a deeper search for more optimal hyper-parameters in order to improve the model's performance. Also, the implementation of autoencoders in order to fully take advantage of the information on the unused channels. And Lastly, testing with different datasets, including different event-related potentials.

This page is intentionally left blank.

References

[1] L F Haas. Hans berger (1873-1941), richard caton (1842-1926), and electroencephalography. *J. Neurol. Neurosurg. Psychiatry*, 74(1):9, January 2003.

[2] J Stoyva and J Kamiya. Electrophysiological studies of dreaming as the prototype of a new strategy in the study of consciousness. *Psychol. Rev.*, 75(3):192–205, May 1968.

[3] J J Vidal. Toward direct brain-computer communication. *Annu. Rev. Biophys. Bioeng.*, 2(1):157–180, 1973.

[4] L A Farwell and E Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalogr. Clin. Neurophysiol.*, 70(6):510–523, December 1988.

[5] Marco Simões, Carlos Amaral, Felipe França, Paulo Carvalho, and Miguel Castelo-Branco. Applying weightless neural networks to a p300-based brain-computer interface. In *IFMBE Proceedings*, IFMBE proceedings, pages 113–117. Springer Singapore, Singapore, 2019.

[6] Davide Borra, Silvia Fantozzi, and Elisa Magosso. Convolutional neural network for a P300 brain-computer interface to improve social attention in autistic spectrum disorder. In *IFMBE Proceedings*, IFMBE proceedings, pages 1837–1843. Springer International Publishing, Cham, 2020.

[7] Simanto Saha and Mathias Baumert. Intra- and inter-subject variability in EEG-based sensorimotor brain computer interface: A review. *Front. Comput. Neurosci.*, 13:87, 2019.

[8] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44, 09 2020.

[9] Marco Simões, Davide Borra, Eduardo Santamaría-Vázquez, GBT-UPM , Mayra Bittencourt-Villalpando, Dominik Krzemiński, Aleksandar Miladinović, Neural_Engineering_Group , Thomas Schmid, Haifeng Zhao, Carlos Amaral, Bruno Direito, Jorge Henriques, Paulo Carvalho, and Miguel Castelo-Branco. Bciaut-p300: A multi-session and multi-subject benchmark dataset on autism for p300-based brain-computer-interfaces. *Frontiers in Neuroscience*, 14, 2020.

[10] Carlos Amaral, Marco Simões, and Miguel Castelo-Branco. BCIAUT_P300, 2020.

[11] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 15(5):056013, jul 2018.

[12] Mamunur Rashid, Norizam Sulaiman, Anwar P. P. Abdul Majeed, Rabiu Muazu Musa, Ahmad Fakhri Ab. Nasir, Bifta Sama Bari, and Sabira Khatun. Current status, challenges, and possible solutions of eeg-based brain-computer interface: A comprehensive review. *Frontiers in Neurorobotics*, 14, 2020.

[13] S Prabhakar, P Syal, and T Srivastava. P300 in newly diagnosed non-dementing parkinson's disease : effect of dopaminergic drugs. *Neurol India*, 48, 2000.

[14] Aruna Tyagi, Sunil Semwal, and gautam shah. A review of eeg sensors used for data acquisition. 08 2012.

[15] John Polich. Updating p300: an integrative theory of p3a and p3b. *Clin. Neurophysiol.*, 118(10):2128–2148, October 2007.

[16] T W Picton. The P300 wave of the human event-related potential. *J. Clin. Neurophysiol.*, 9(4):456–479, October 1992.

[17] Rina Dechter. Learning while searching in constraint-satisfaction-problems. In *AAAI*, 1986.

[18] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[19] Li Deng. Deep learning: Methods and applications. *Found. Trends® Signal Process.*, 7(3-4):197–387, 2014.

[20] Petr Nejedly, Jan Cimbalnik, Petr Klimes, Filip Plesinger, Josef Halamek, Vaclav Kremen, Ivo Viscor, Benjamin H Brinkmann, Martin Pail, Milan Brazdil, et al. Intracerebral eeg artifact identification using convolutional neural networks. *Neuroinformatics*, 17(2):225–234, 2019.

[21] Yang Liu, Chenguang Yang, and Zhijun Li. The application of transfer learning in p300 detection. In *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pages 412–417, 2018.

[22] C S Herrmann. Human EEG responses to 1-100 hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena. *Exp. Brain Res.*, 137(3-4):346–353, April 2001.

[23] G. Pfurtscheller and C. Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, July 2001.

[24] N Birbaumer, A Kübler, N Ghanayim, T Hinterberger, J Perelmouter, J Kaiser, I Iversen, B Kotchoubey, N Neumann, and H Flor. The thought translation device (TTD) for completely paralyzed patients. *IEEE Trans. Rehabil. Eng.*, 8(2):190–193, June 2000.

[25] Reza Fazel-Rezai and Kamyar Abhari. A comparison between a matrix-based and a region-based p300 speller paradigms for brain-computer interface. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1147–1150, 2008.

[26] Reza Fazel-Rezai. Human error in P300 speller paradigm for brain-computer interface. *Annu Int Conf IEEE Eng Med Biol Soc*, 2007:2516–2519, 2007.

[27] T Kaufmann, S M Schulz, C Grünzinger, and A Kübler. Flashing characters with famous faces improves ERP-based brain-computer interface performance. *J. Neural Eng.*, 8(5):056016, October 2011.

[28] Andrew Geronimo and Zachary Simmons. The p300 'face' speller is resistant to cognitive decline in als. *Brain-Computer Interfaces*, 4:1–11, 06 2017.

[29] Liam Ratcliffe and Sadasivan Puthusserypady. Importance of graphical user interface in the design of P300 based Brain-Computer interface systems. *Comput. Biol. Med.*, 117(103599):103599, February 2020.

[30] Carlos P. Amaral, Marco A. Simões, Susana Mouga, João Andrade, and Miguel Castelo-Branco. A novel brain computer interface for classification of social joint attention in autism and comparison of 3 experimental setups: A feasibility study. *Journal of Neuroscience Methods*, 290:105–115, 2017.

[31] Tobias Kaufmann, Elisa M Holz, and Andrea Kübler. Comparison of tactile, auditory, and visual modality for brain-computer interface use: a case study with a patient in the locked-in state. *Front. Neurosci.*, 7:129, July 2013.

[32] Martijn Schreuder, Thomas Rost, and Michael Tangermann. Listen, you are writing! speeding up online spelling with a dynamic auditory BCI. *Front. Neurosci.*, 5:112, October 2011.

[33] S Halder, I Käthner, and A Kübler. Training leads to increased auditory brain-computer interface performance of end-users with motor impairments. *Clin. Neurophysiol.*, 127(2):1288–1296, February 2016.

[34] Tobias Kaufmann, Andreas Herweg, and Andrea Kübler. Toward brain-computer interface based wheelchair control utilizing tactually-evoked event-related potentials. *J. Neuroeng. Rehabil.*, 11(1):7, January 2014.

[35] Bci award 2021. https://www.bci-award.com/.

[36] Florian Yger, Maxime Berar, and Fabien Lotte. Riemannian approaches in brain-computer interfaces: A review. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1753–1762, 2017.

[37] Violaine Guy, Marie-Helene Soriani, Mariane Bruno, Theodore Papadopoulo, Claude Desnuelle, and Maureen Clerc. Brain computer interface with the p300 speller: usability for disabled people with amyotrophic lateral sclerosis. *Annals of physical and rehabilitation medicine*, 61(1):5–11, 2018.

[38] Eda Akman Aydin, Omer Faruk Bay, and Inan Guler. P300-based asynchronous brain computer interface for environmental control system. *IEEE journal of biomedical and health informatics*, 22(3):653–663, 2017.

[39] Daniela De Venuto, Valerio F Annese, and Giovanni Mezzina. Real-time p300-based bci in mechatronic control by using a multi-dimensional approach. *IET Software*, 12(5):418–424, 2018.

[40] Zehong Cao. A review of artificial intelligence for eeg-based brain- computer interfaces and applications. *Brain Science Advances*, 6(3):162–170, 2020.

[41] L. Korczowski, M. Congedo, and C. Jutten. Single-trial classification of multi-user p300-based brain-computer interface using riemannian geometry. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1769–1772, 2015.

[42] D Puthankattil Subha, Paul K Joseph, Rajendra Acharya U, and Choo Min Lim. EEG signal analysis: a survey. *J. Med. Syst.*, 34(2):195–212, April 2010.

[43] Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy, and Florian Yger. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005, 2018.

[44] Hyohyeong Kang, Yunjun Nam, and Seungjin Choi. Composite common spatial pattern for subject-to-subject transfer. *IEEE Signal Processing Letters*, 16:683–686, 2009.

[45] Sumanto Dutta and Anup Nandy. Data augmentation for ambulatory EEG based cognitive state taxonomy system with RNN-LSTM. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 468–473. Springer International Publishing, Cham, 2019.

[46] Sebastian Thrun and Lorien Pratt. *Learning to Learn: Introduction and Overview*, pages 3–17. Springer US, Boston, MA, 1998.

[47] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[48] Zitong Wan, Rui Yang, Mengjie Huang, Nianyin Zeng, and Xiaohui Liu. A review on transfer learning in eeg signal analysis. *Neurocomputing*, 421:1–14, 2021.

[49] Roberto Santana, Luis Marti, and Mengjie Zhang. GP-based methods for domain adaptation: using brain decoding across subjects as a test-case. *Genet. Program. Evolvable Mach.*, 20(3):385–411, September 2019.

[50] Aya Khalaf and Murat Akcakaya. A probabilistic approach for calibration time reduction in hybrid EEG-fTCD brain-computer interfaces. *Biomed. Eng. Online*, 19(1):23, April 2020.

[51] Hao Wu, Yi Niu, Fu Li, Yuchen Li, Boxun Fu, Guangming Shi, and Minghao Dong. A parallel multiscale filter bank convolutional neural networks for motor imagery eeg classification. *Frontiers in Neuroscience*, 13, 2019.

[52] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.

[53] S. Raghu, Natarajan Sriraam, Yasin Temel, Shyam Vasudeva Rao, and Pieter L. Kubben. Eeg based multi-class seizure type classification using convolutional neural network and transfer learning. *Neural Networks*, 124:202–212, 2020.

[54] Jianhua Zhang, Yongcun Wang, and Sunan Li. Cross-subject mental workload classification using kernel spectral regression and transfer learning techniques. *Cogn. Technol. Work*, 19(4):587–605, November 2017.

[55] Michele Svanera, Mattia Savardi, Sergio Benini, Alberto Signoroni, Gal Raz, Talma Hendler, Lars Muckli, Rainer Goebel, and Giancarlo Valente. Transfer learning of deep neural network representations for fmri decoding. *Journal of neuroscience methods*, 328:108319, 2019.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[57] Bo-Qun Ma, He Li, Wei-Long Zheng, and Bao-Liang Lu. Reducing the subject variability of EEG signals with adversarial domain generalization. In *Neural Information Processing*, Lecture notes in computer science, pages 30–42. Springer International Publishing, Cham, 2019.

[58] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. SpeechBrain: A general-purpose speech toolkit, 2021. arXiv:2106.04624.

[59] Eric Larson, Alexandre Gramfort, Denis A Engemann, Jaakko Leppakangas, Christian Brodbeck, Mainak Jas, Teon Brooks, Jona Sassenhagen, Martin Luessi, Jean-Remi King, Daniel McCloy, Roman Goj, Guillaume Favelier, Richard Höchenberger, Clemens Brunner, Marijn van Vliet, Mark Wronkiewicz, Chris Holdgraf, Joan Massich, Yousra Bekhti, Alex Rockhill, Stefan Appelhoff, Alan Leggitt, Andrew Dykstra, Rob Luke, Romain Trachel, Lorenzo De Santis, Asish Panda, Mikołaj Magnuski, Martin Billinger, Britta Westner, Dan G Wakeman, Daniel Strohmeier, Hari Bharadwaj, Tal Linzen, Alexandre Barachant, Mathieu Scheltienne, Emily Ruzich, Christopher J Bailey, Adam Li, Clément Moutard, Luke Bloy, Fede Raimondo, Jussi Nurminen, Jair Montoya, Marmaduke Woodman, Ingoo Lee, Martin Schulz, Nick Foti, Cathy Nangini, José C García Alanis, Ross Maddox, Roan LaPlante, Ashley Drew, Christoph Dinh, Olaf Hauk, Guillaume Dumas, Thomas Hartmann, Johann Benerradi, Paul Pasler, Stefan Repplinger, Alexander Rudiuk, Ana Radanovic, Brad Buran, Mathurin Massias, Matti Hämäläinen, Praveen Sripad, Valerii Chirkov, Christopher Mullins, Félix Raimundo, Phillip Alday, Ram Pari, Simon Kornblith, Yaroslav Halchenko, Yu-Han Luo, Eduard Ort, Johannes Kasper, Keith Doelling, Mads Jensen, Tanay Gahlot, Adonay Nunes, Dirk Gütlin, kjs, Alejandro Weinstein, Camilo Lamus, Catalina María Galván, Cristóbal Moënne-Loccoz, Jon Houck, Natalie Klein, Antti Rantala, Burkhard Maess, Christian O'Reilly, Erica Peterson, Erkka Heinila, Henrich Kolkhorst, Hubert Banville, Jeff Hanna, Maggie Clarke, Matteo Anelli, Michiru Kaneda, Pierre-Antoine Bannier, Saket Choudhary, Scott Huberty, Simon Kern, Carina Forster, Cora Kim, Felix Klotzsche, Fu-Te Wong, Ivana Kojcic, Jesper Duemose Nielsen, Kaisu Lankinen, Kambiz Tabavi, Kostiantyn Maksymenko, Louis Thibault, Nathalie Gayraud, Nick Ward, dependabot

bot

, Antoine Gauthier, Basile Pinsard, Dominik Welke, Emily Stephen, Erik Hornberger, Evan Hathaway, Evgenii Kalenkovich, Fahimeh Mamashli, Giorgio Marinato, Hafeza Anevar, Jack Zhang, Jan Sosulski, Jeff Stout, Larry Eisenman, Lorenz Esch, Nicolas Barascud, Nicolas Legrand, Rotem Falach, Samuel Deslauriers-Gauthier, Silvia Cotroneo, Steve Matindi, Steven Bierer, Victor Férat, Victoria Peterson, Alessandro Tonin, Alexander Kovrig, Annalisa Pascarella, Apoorva Karekal, Christina Zhao, Dominik Krzemiński, Dominik Welke, Dominique Makowski, Ezequiel Mikulan, Jean-Baptiste Schiratti, Jen Evans, Jordan Drew, Joshua Teves, Kyle Mathewson, Laura Gwilliams, Lenny Varghese, Lx37, Marian Dovgialo, Matt Boggess, Matthias Eberlein, Mohamed Sherif, Nataliia Kozhemiako, Naveen Srinivasan, Niklas Wilming, Nikolai Chapochnikov, Oleh Kozynets, Pierre Ablin, Quentin Bertrand, Reza Shoorangiz, Rodrigo Hübner, Sara Sommariva, Sheraz Khan, Sophie Herbst, Sumalyo Datta, Thomas Jochmann, Tod Flak, Tom Dupré la Tour, Tristan Stenner, Tziona NessAiver, akshay0724, sviter, Abram Hindle, Achilleas Koutsou, Adeline Fecker, Adina Wagner, Alex Ciok, Aniket Pradhan, Anna Padee, Anne-Sophie Dubarry, Anton Nikolas

Waniek, Archit Singhal, Ariel Rokem, Austin Hurst, Ben Beasley, Bruno Nicenboim, Carlos de la Torre, Christian Clauss, Christian Mista, Chun-Hui Li, Claire Braboszcz, Darin Erat Sleiter, David Haslacher, David Sabbagh, Demetres Kostas, Desislava Petkova, Dmitrii Altukhov, Eberhard Eich, Elizabeth DuPre, Ellen Lau, Emanuele Olivetti, Enrico Varano, Etienne de Montalivet, Evgeny Goldstein, Federico Zamberlan, Frederik D Weber, Gansheng Tan, Geoff Brookshire, Hamid Maymandi, Hermann Sonntag, Hongjiang Ye, Ilias Machairas, Jakub Kaczmarzyk, Jan Zerfowski, Jasper J F van den Bosch, Jeroen Van Der Donckt, Johan van der Meer, Johannes Niediek, John Veillette, Josh Koen, Joshua J Bear, Judy D Zhu, Juergen Dammers, Julia Guiomar Niso Galán, Katarina Slama, Katrin Leinweber, Laetitia Grabot, Lau Møller Andersen, Leonardo S Barbosa, Liberty Hamilton, Lorenzo Alfine, Lukas Gemein, Lukas Hecker, Lukáš Hejtmánek, Manfred Kitzbichler, Manoj Kumar, Manorama Kadwani, Manu Sutela, Marcin Koculak, Martin van Harmelen, MartinBaBer, Matt Courtemanche, Matt Tucker, Matteo Visconti di Oleggio Castello, Matthias Dold, Matti Toivonen, Maureen Shader, Michael Krause, Milan Rybář, Mingjian He, Mohammad Daneshzand, Nicolas Gensollen, Nicole Proulx, Nikolas Chalas, Padma Sundaram, Paul Roujansky, Pedro Silva, Peter J Molfese, Quanliang Li, Rahul Nadkarni, Ramiro Gatti, Ramonapariciog Apariciogarcia, Reza Nasri, Richard Koehler, Riessarius Stargardsky, Robert Oostenveld, Robert Seymour, Robin Tibor Schirrmeister, Ryan Law, Sagun Pai, Sam Perry, Sebastian Major, Sebastien Treguer, Sebastián Castaño, Senwen Deng, Sergey Antopolskiy, Simeon Wong, Simon-Shlomo Poil, Sondre Foslien, Sourav Singh, Stanislas Chambon, Steven Bethard, Steven M Gutstein, Svea Marie Meyer, T Wang, Theodore Papadopoulo, Thomas Donoghue, Thomas Radman, Timon Merk, Timothy Gates, Tommy Clausner, Xiaokai Xia, Zhi Zhang, and buildqa. Mne-python, August 2022. If you use this software, please cite both the software itself, and the paper listed in the preferred-citation field.

[60] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013.

[61] Aniana Cruz, Gabriel Pires, and Urbano J Nunes. Error-related potentials (primary and secondary ErrP) and P300 event related potentials – BCI-Double-ErrP-Dataset, 2020.

[62] Aniana Cruz, Gabriel Pires, and Urbano J. Nunes. Double errp detection for automatic error correction in an erp-based bci speller. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(1):26–36, 2018.

[63] Emily S. Kappenman, Jaclyn L. Farrens, Wendy Zhang, Andrew X. Stewart, and Steven J. Luck. Erp core: An open resource for human event-related potential research. *NeuroImage*, 225:117465, 2021.

[64] Steven J. Luck, Emily S. Kappenman, Rebecca L. Fuller, Benjamin Robinson, Ann Summerfelt, and James M. Gold. Impaired response selection in schizophrenia: Evidence from the p3 wave and the lateralized readiness potential. *Psychophysiology*, 46(4):776–786, July 2009.

[65] Brylie Christopher Oxley, CC0, via Wikimedia Commons. International 10-20 system for eeg electrode placement, showing modified combinatorial nomenclature, 2020. [Online; accessed August 20, 2022].

[66] Julius O. Smith. *Digital Audio Resampling Home Page.* January 28, 2002.

[67] F. Perrin, J. Pernier, O. Bertrand, and J.F. Echallier. Spherical splines for scalp potential and current density mapping. *Electroencephalography and Clinical Neurophysiology*, 72(2):184–187, February 1989.

[68] Jayme Garcia Arnal Barbedo. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153:46–53, October 2018.

[69] Deepak Soekhoe, Peter van der Putten, and Aske Plaat. On the impact of data set size in transfer learning using deep neural networks. In *Lecture Notes in Computer Science*, pages 50–60. Springer International Publishing, 2016.

This page is intentionally left blank.

Appendices

This page is intentionally left blank.

Appendix A

BCI-DOUBLE-ERRP Subject-Specific Metrics

Table A.1: F-score metrics from Subject-specific models using BCIAUT-P300 optimized hyper-parameters on the BCI-DOUBLE-ERRP dataset.

	Model ID								
Subject	**DERRP_baseline**	**DERRP_0**	**DERRP_1**	**DERRP_2**	**DERRP_3**	**DERRP_4**	**DERRP_5**	**DERRP_6**	**DERRP_7**
P1	0.167526335	0.128519138	0.100781009	0.101382065	0.097626159	0.095223892	0.083747128	0.082172456	0.084102866
S1	0.199519231	0.119066081	0.114235205	0.11424245	0.094084207	0.089394209	0.088943753	0.084329144	0.083973144
S3	0.311671924	0.208298853	0.224927229	0.224927229	0.143970767	0.149006622	0.118109299	0.114089347	0.116400857
S4	0.244589309	0.178930651	0.168147642	0.167408727	0.124284546	0.12421262	0.100006427	0.102220167	0.100098582
S5	0.210808996	0.152993615	0.149832693	0.149795564	0.126496218	0.112491277	0.102671337	0.088460193	0.094208647
S6	0.209995551	0.197450853	0.17542317	0.149497487	0.114276914	0.095575819	0.086608442	0.105528543	0.098271156
S9	0.439279732	0.276022305	0.334525939	0.334227948	0.153070654	0.132276745	0.124515091	0.108763032	0.11522101
Mean	0.254770154	0.180183071	0.181124698	0.177354496	0.121972781	0.114025883	0.100657354	0.097937555	0.098896609
Std Dev	0.093083065	0.053881221	0.079120179	0.079855924	0.022009664	0.022224273	0.015804946	0.012764422	0.013154608

Table A.2: Balanced accuracy metrics from Subject-specific models using BCIAUT-P300 optimized hyper-parameters on the BCI-DOUBLE-ERRP dataset.

	Model ID								
Subject	**DERRP_baseline**	**DERRP_0**	**DERRP_1**	**DERRP_2**	**DERRP_3**	**DERRP_4**	**DERRP_5**	**DERRP_6**	**DERRP_7**
P1	0.667754569	0.661049866	0.593567676	0.592117139	0.593720788	0.590521549	0.556022951	0.55288818	0.557384844
S1	0.668611572	0.648307429	0.637878712	0.63797931	0.593355465	0.577863287	0.583463273	0.567426186	0.564349546
S3	0.763608445	0.757885876	0.758373206	0.758373206	0.641650422	0.663831295	0.599577648	0.595516569	0.596638904
S4	0.719530388	0.641456401	0.647013038	0.649023805	0.587003524	0.590808847	0.585814361	0.578971265	0.575738903
S5	0.713887208	0.677808026	0.66735565	0.667322041	0.653811252	0.622344895	0.588819991	0.558605566	0.572561672
S6	0.730174911	0.584591363	0.57210006	0.560749478	0.550029934	0.534715143	0.527967898	0.539965698	0.535224827
S9	0.803907918	0.733082707	0.757425972	0.757112689	0.676378446	0.648240973	0.609660726	0.583089669	0.595446951
Mean	0.723925002	0.672025953	0.661959188	0.660382524	0.613707119	0.60404657	0.578760978	0.568066162	0.571049378
Std Dev	0.048897626	0.058303279	0.073022839	0.075525809	0.044531588	0.044182881	0.027863143	0.019183637	0.021561346

Table A.3: F-score metrics from Subject-specific models using BCI-DOUBLE-ERRP optimized hyper-parameters on the BCI-DOUBLE-ERRP dataset.

	Model ID								
Subject	**DERRP_baseline _optimized**	**DERRP _optimized_0**	**DERRP _optimized_1**	**DERRP _optimized_2**	**DERRP _optimized_3**	**DERRP _optimized_4**	**DERRP _optimized_5**	**DERRP _optimized_6**	**DERRP _optimized_7**
P1	0.229585847	0.233732626	0.198434388	0.18345436	0.085173963	0.113577817	0.107610339	0.115307654	0.139992471
S1	0.231974739	0.155323507	0.1853576	0.17227021	0.150899353	0.13996401	0.126466443	0.114634371	0.124006597
S3	0.347912321	0.305408118	0.270890872	0.281735976	0.1685372	0.190971752	0.172471317	0.125726117	0.140568916
S4	0.276151949	0.307885211	0.270408017	0.296085639	0.195468416	0.184115227	0.162237918	0.101946864	0.133680121
S5	0.248635485	0.214853519	0.16865211	0.200028373	0.177723854	0.171115833	0.153367805	0.126205151	0.135374068
S6	0.248042031	0.248884298	0.25099842	0.24091857	0.139284325	0.112693232	0.149924371	0.132340157	0.143867898
S9	0.49134752	0.483139037	0.443965932	0.449490167	0.236597757	0.241742799	0.234917497	0.1820898	0.193143674
Mean	0.296235699	0.278460902	0.25552962	0.260569042	0.164812124	0.164882953	0.158142241	0.128321445	0.144376249
Std Dev	0.09513371	0.104538026	0.092841971	0.095912574	0.047365678	0.046500409	0.040339589	0.025725194	0.022444757

Table A.4: Balanced accuracy metrics from Subject-specific models using BCI-DOUBLE-ERRP optimized hyper-parameters on the BCI-DOUBLE-ERRP dataset.

	Model ID								
Subject	**DERRP_baseline _optimized**	**DERRP _optimized_0**	**DERRP _optimized_1**	**DERRP _optimized_2**	**DERRP _optimized_3**	**DERRP _optimized_4**	**DERRP _optimized_5**	**DERRP _optimized_6**	**DERRP _optimized_7**
P1	0.684706317	0.639508042	0.669278893	0.656426719	0.568088592	0.560884557	0.566047824	0.612116455	0.566334979
S1	0.741926127	0.655993729	0.637358297	0.630539672	0.638880377	0.598700717	0.590875304	0.590298533	0.606951533
S3	0.839916654	0.809175161	0.797017016	0.812922029	0.66800751	0.756020692	0.760096569	0.715402356	0.682680604
S4	0.726673416	0.695388421	0.707063204	0.714462654	0.683003361	0.698315946	0.629887917	0.543079619	0.560411922
S5	0.686696445	0.708775506	0.715516313	0.683982508	0.711266757	0.714748262	0.643372572	0.633164259	0.625002488
S6	0.772134935	0.74529094	0.757179097	0.719706938	0.569527037	0.569050684	0.558753672	0.553165975	0.552321026
S9	0.830550204	0.774613901	0.743026274	0.760155093	0.714200185	0.743437058	0.700854855	0.606878633	0.593720264
Mean	0.754657728	0.718392243	0.718062728	0.711170802	0.650424831	0.663022559	0.635698388	0.607729404	0.598203259
Std Dev	0.062970041	0.061730465	0.053863392	0.062013402	0.061368464	0.084099643	0.073704055	0.057330859	0.045643915

Appendix B

ERPCORE Subject-Specific Metrics

Table B.1: F-score metrics from Subject-specific models using BCIAUT-P300 optimized hyper-parameters on the ERPCORE dataset.

	Model ID							
Subject	**ERP_baseline**	**ERP_0**	**ERP_1**	**ErP_2**	**ERP_3**	**ERP_4**	**ERP_6**	**ERP_7**
1	0.315845429	0.362806234	0.338053879	0.353609434	0.311282708	0.340143499	0.325299177	0.324087056
2	0.685238095	0.473363679	0.480898872	0.480898872	0.45316383	0.446758519	0.442931578	0.439206088
3	0.332140523	0.34517108	0.334377429	0.332963288	0.330258799	0.348281022	0.348639242	0.348639242
4	0.634347188	0.608946755	0.609389417	0.614843962	0.594201681	0.584364851	0.596422466	0.584884005
5	0.363614164	0.263627451	0.243071895	0.243071895	0.192451405	0.239076479	0.245887446	0.316552892
7	0.492967033	0.504279054	0.525584416	0.52998002	0.452456367	0.466640271	0.448572865	0.445276162
8	0.192777778	0.295346728	0.258813539	0.246194492	0.276922244	0.214649207	0.22518648	0.228727384
11	0.154487179	0.362503052	0.344395849	0.344395849	0.403779915	0.391155566	0.389726995	0.395779031
12	0.322820513	0.385240184	0.371485491	0.367080729	0.369311741	0.36392327	0.367575415	0.364709918
13	0.227435897	0.304070867	0.312142763	0.312142763	0.320479067	0.326735617	0.346150939	0.356608701
14	0.13	0.290996197	0.316020617	0.335465061	0.322407732	0.333140133	0.338742129	0.338742129
15	0.415598291	0.374925776	0.361465201	0.374897651	0.363698462	0.380466139	0.381647764	0.385999491
16	0.52515873	0.444444444	0.424796054	0.424796054	0.361610106	0.321411693	0.384653092	0.366876751
17	0.236245421	0.328353839	0.312670882	0.300170882	0.305178627	0.299500721	0.30203543	0.304241313
18	0.597394827	0.428697691	0.396955267	0.424632035	0.374797637	0.359375314	0.36493087	0.378640771
19	0.119404762	0.250027101	0.308413466	0.308413466	0.28463364	0.307336721	0.330345379	0.330345379
20	0.185617716	0.332236842	0.332023564	0.332023564	0.363253968	0.348616252	0.354722695	0.35120329
21	0.298809524	0.320658508	0.27262654	0.274412254	0.29260181	0.281914197	0.305438728	0.2996115
22	0.473174603	0.481901987	0.448106909	0.448106909	0.388771278	0.385711706	0.363478237	0.363478237
23	0.103174603	0.453578089	0.389635854	0.389635854	0.273233728	0.303543906	0.301321684	0.301321684
24	0.282847153	0.405255446	0.373739581	0.373739581	0.300425261	0.316916858	0.327076306	0.304786096
25	0.441334221	0.435953817	0.416845426	0.406012092	0.36872722	0.385145801	0.396540681	0.369407814
26	0.257582418	0.319491064	0.362773827	0.362773827	0.349410002	0.246188779	0.2714082	0.258807189
27	0.322214452	0.301442756	0.293685808	0.291479925	0.347773109	0.326700541	0.301746881	0.305318309
28	0.350966811	0.2788586	0.272040418	0.265373752	0.335936578	0.301602441	0.22482311	0.217838983
29	0.281986934	0.337670458	0.369957543	0.369957543	0.344593903	0.386090739	0.357350591	0.367946661
31	0.196172439	0.261428571	0.299496223	0.299496223	0.280034709	0.259327639	0.274479154	0.274479154
32	0.285714286	0.299558775	0.320460095	0.320460095	0.238403786	0.266333079	0.244536542	0.241758764
33	0.204285714	0.44964924	0.387263341	0.390633305	0.338351648	0.369503829	0.327778888	0.335937396
34	0.491212121	0.433070789	0.368144049	0.364041485	0.290887413	0.356592819	0.286116629	0.282944245
36	0.454267677	0.353329799	0.30512623	0.318313043	0.358500388	0.339297924	0.329957265	0.301535687
37	0.487453248	0.431773227	0.414038706	0.414038706	0.27584391	0.250703854	0.277527162	0.263836686
38	0.555079365	0.558356088	0.553657454	0.545302475	0.52973471	0.502237762	0.413304473	0.413304473
39	0.164285714	0.409177733	0.381099929	0.381099929	0.320371581	0.286241459	0.288514856	0.323038665
Mean	0.340636907	0.379005645	0.367625192	0.368836971	0.344514381	0.342224371	0.337790275	0.33781974
Std Dev	0.156629665	0.087102895	0.082572706	0.083421181	0.077373357	0.076615083	0.07271798	0.07033671

Table B.2: Balanced accuracy metrics from Subject-specific models using BCIAUT-P300 optimized hyper-parameters on the ERPCORE dataset.

	Model ID								
Subject	**ERP_baseline**	**ERP_0**	**ERP_1**	**ErP_2**	**ERP_3**	**ERP_4**	**ERP_5**	**ERP_6**	**ERP_7**
1	0.578125	0.565625	0.540625	0.553125	0.496875	0.51875	0.53125	0.528125	0.53125
2	0.790625	0.703125	0.70625	0.70625	0.6875	0.675	0.653125	0.671875	0.665625
3	0.578125	0.559375	0.546875	0.54375	0.534375	0.55625	0.55625	0.55625	0.55625
4	0.81875	0.79375	0.790625	0.79375	0.775	0.765625	0.765625	0.778125	0.765625
5	0.61875	0.528125	0.50625	0.50625	0.4875	0.50625	0.528125	0.515625	0.578125
7	0.7125	0.725	0.74375	0.746875	0.675	0.69375	0.6875	0.68125	0.678125
8	0.53125	0.515625	0.490625	0.478125	0.50625	0.475	0.484375	0.4875	0.496875
11	0.5375	0.575	0.559375	0.559375	0.621875	0.603125	0.59375	0.603125	0.609375
12	0.621875	0.603125	0.5875	0.58125	0.58125	0.571875	0.578125	0.578125	0.571875
13	0.5125	0.515625	0.53125	0.53125	0.5375	0.55	0.56875	0.571875	0.56875
14	0.521875	0.515625	0.5375	0.559375	0.53125	0.534375	0.559375	0.540625	0.540625
15	0.68125	0.603125	0.5875	0.603125	0.578125	0.59375	0.6125	0.5875	0.596875
16	0.725	0.671875	0.646875	0.646875	0.578125	0.5375	0.578125	0.6	0.584375
17	0.578125	0.53125	0.525	0.509375	0.496875	0.4875	0.4875	0.484375	0.4875
18	0.765625	0.634375	0.61875	0.6375	0.6	0.6	0.5875	0.6	0.615625
19	0.48125	0.434375	0.5	0.5	0.471875	0.478125	0.509375	0.515625	0.515625
20	0.515625	0.53125	0.5375	0.5375	0.578125	0.5625	0.5875	0.575	0.56875
21	0.565625	0.553125	0.496875	0.496875	0.5125	0.515625	0.521875	0.54375	0.53125
22	0.675	0.7125	0.675	0.675	0.60625	0.60625	0.575	0.578125	0.578125
23	0.471875	0.675	0.609375	0.609375	0.459375	0.4875	0.465625	0.478125	0.478125
24	0.559375	0.63125	0.596875	0.596875	0.503125	0.521875	0.50625	0.534375	0.509375
25	0.665625	0.659375	0.6375	0.625	0.6	0.6125	0.61875	0.61875	0.59375
26	0.571875	0.54375	0.584375	0.584375	0.571875	0.4875	0.490625	0.521875	0.509375
27	0.58125	0.4875	0.503125	0.5	0.565625	0.575	0.54375	0.534375	0.55
28	0.584375	0.48125	0.478125	0.478125	0.528125	0.5	0.453125	0.434375	0.425
29	0.534375	0.559375	0.596875	0.596875	0.565625	0.6	0.584375	0.56875	0.58125
31	0.5	0.490625	0.503125	0.503125	0.48125	0.471875	0.48125	0.48125	0.48125
32	0.6	0.5625	0.578125	0.578125	0.49375	0.51875	0.4875	0.4875	0.484375
33	0.565625	0.66875	0.603125	0.603125	0.55625	0.5875	0.5625	0.553125	0.559375
34	0.678125	0.634375	0.578125	0.571875	0.5375	0.5875	0.534375	0.5125	0.51875
36	0.675	0.5625	0.515625	0.528125	0.584375	0.56875	0.56875	0.559375	0.534375
37	0.7125	0.659375	0.6375	0.6375	0.478125	0.45625	0.48125	0.48125	0.465625
38	0.74375	0.75625	0.75625	0.746875	0.734375	0.7125	0.65625	0.65625	0.65625
39	0.5125	0.63125	0.59375	0.59375	0.51875	0.490625	0.475	0.503125	0.525
Mean	0.610753676	0.596323529	0.585294118	0.585845588	0.559834559	0.559099265	0.555147059	0.556525735	0.55625
Std Dev	0.093291782	0.085762893	0.079066409	0.079323143	0.073759198	0.073301343	0.068942921	0.06942407	0.068318203

Table B.3: F-score metrics from Subject-specific models using ERPCORE optimized hyper-parameters on the ERPCORE dataset.

	Model ID								
Subject	**ERP_baseline _optimized**	**ERP_optimized _0**	**ERP_optimized _1**	**ErP_optimized _2**	**ERP_optimized _3**	**ERP_optimized _4**	**ERP_optimized _5**	**ERP_optimized _6**	**ERP_optimized _7**
1	0.315845429	0.273116124	0.29432103	0.29432103	0.317993673	0.321349925	0.31484127	0.31484127	0.308296703
2	0.616379731	0.601490045	0.628782329	0.635448995	0.397529692	0.357668492	0.373641881	0.369862915	0.384004329
3	0.357301978	0.362571089	0.339469468	0.339469468	0.325587158	0.325587158	0.30986249	0.31315126	0.320587158
4	0.65869663	0.496780997	0.467051053	0.467051053	0.499529915	0.463735431	0.443562271	0.366949717	0.372029082
5	0.211111111	0.25290404	0.2596337	0.244249084	0.338943834	0.253762512	0.257707848	0.284949495	0.298377897
7	0.530318897	0.597365967	0.588974359	0.588974359	0.414369487	0.432810458	0.452708125	0.467552448	0.443862412
8	0.333434343	0.284891909	0.316515151	0.316515151	0.264028136	0.269482681	0.270062657	0.291729323	0.251237067
11	0.304935065	0.435532213	0.420472276	0.420472276	0.363349983	0.321404151	0.37241533	0.342768865	0.319418327
12	0.516926407	0.477088467	0.473015873	0.473015873	0.353696581	0.348892206	0.37333665	0.360522876	0.388464052
13	0.223819844	0.317411298	0.351136135	0.353040897	0.330882353	0.327167183	0.29872635	0.315768957	0.318710133
14	0.455108225	0.337345987	0.331896992	0.328866689	0.303823954	0.322339882	0.279689722	0.309341492	0.286285936
15	0.172967033	0.406081769	0.438500389	0.438500389	0.380318897	0.380878533	0.423670186	0.393850561	0.37307134
16	0.466666667	0.52719697	0.572911255	0.572911255	0.436298847	0.416903769	0.391482684	0.391684149	0.373410364
17	0.375565317	0.38340803	0.400537518	0.400537518	0.340373483	0.340231148	0.345079365	0.340576441	0.326037822
18	0.457184628	0.39207642	0.358461539	0.358461539	0.307137307	0.278152403	0.286283716	0.200899101	0.253861694
19	0.220966811	0.35426859	0.347703506	0.347703506	0.34459224	0.349197504	0.337137943	0.337137943	0.345307878
20	0.45767507	0.405794255	0.391211338	0.391211338	0.36488427	0.378067227	0.338588659	0.350854037	0.34581202
21	0.393924964	0.431930847	0.318145743	0.319537684	0.187878788	0.220656566	0.17013431	0.23528083	0.129404762
22	0.45974692	0.400552781	0.393584719	0.393584719	0.435248611	0.427582418	0.395369192	0.381770598	0.394313026
23	0.186666667	0.435353535	0.40705267	0.410941558	0.300295525	0.313144995	0.316910279	0.30432256	0.296831473
24	0.193333333	0.387642602	0.391542317	0.391542317	0.373823529	0.373055556	0.329790095	0.331456762	0.323156566
25	0.361428571	0.445836386	0.442559075	0.442559075	0.409494721	0.3939819	0.374586247	0.398200133	0.39706377
26	0.289935065	0.399553195	0.39531857	0.401985237	0.326590909	0.306235676	0.297015729	0.299971989	0.315511204
27	0.358192427	0.346615933	0.369090909	0.369090909	0.219758308	0.224521352	0.238227091	0.238139769	0.204301613
28	0.335396825	0.248191253	0.281690417	0.281690417	0.273589744	0.279462759	0.252055722	0.27746337	0.308192086
29	0.321434676	0.382552973	0.344784076	0.344784076	0.33698718	0.361998557	0.308621379	0.335961538	0.324825175
31	0.262979798	0.301208223	0.320536229	0.320536229	0.226540616	0.22267507	0.229621849	0.229621849	0.198486318
32	0.498333333	0.350714286	0.241391941	0.256947497	0.24475913	0.266450217	0.217002801	0.207002801	0.221613191
33	0.329285714	0.394021256	0.383388576	0.383388576	0.246998311	0.246998311	0.263845174	0.255187166	0.270625762
34	0.562142857	0.469126984	0.516443001	0.500887446	0.293388719	0.283150624	0.294272382	0.296470184	0.306764302
36	0.371766567	0.375746606	0.285145296	0.286963478	0.313883029	0.325933233	0.333226382	0.338015759	0.32984349
37	0.46979243	0.4645671	0.47472028	0.47472028	0.277589439	0.245515873	0.249460784	0.248816739	0.23426713
38	0.485238095	0.56540404	0.43494228	0.43494228	0.308406593	0.274340659	0.253720169	0.264995005	0.202614053
39	0.237230898	0.372091503	0.358935574	0.358935574	0.320798225	0.316523199	0.332677045	0.342947546	0.341778078
Mean	0.376227421	0.402248049	0.392348988	0.392464346	0.328805035	0.322642871	0.315450993	0.315825454	0.309069594
Std Dev	0.126638956	0.088486599	0.091194462	0.091076094	0.067142629	0.063032639	0.065821373	0.059796632	0.067709727

Table B.4: Balanced accuracy metrics from Subject-specific models using ERPCORE optimized hyper-parameters on the ERPCORE dataset.

	Model ID								
Subject	**ERP_baseline _optimized**	**ERP_optimized _0**	**ERP_optimized _1**	**ErP_optimized _2**	**ERP_optimized _3**	**ERP_optimized _4**	**ERP_optimized _5**	**ERP_optimized _6**	**ERP_optimized _7**
1	0.5875	0.490625	0.50625	0.50625	0.51875	0.509375	0.50625	0.50625	0.496875
2	0.771875	0.7875	0.80625	0.809375	0.603125	0.565625	0.565625	0.571875	0.59375
3	0.5875	0.56875	0.55	0.55	0.525	0.525	0.509375	0.515625	0.521875
4	0.828125	0.7125	0.690625	0.690625	0.703125	0.675	0.6625	0.5875	0.59375
5	0.540625	0.446875	0.459375	0.446875	0.54375	0.475	0.478125	0.515625	0.53125
7	0.75625	0.784375	0.78125	0.78125	0.625	0.64375	0.665625	0.66875	0.64375
8	0.578125	0.5125	0.553125	0.553125	0.49375	0.496875	0.490625	0.509375	0.4625
11	0.5875	0.65	0.634375	0.634375	0.584375	0.540625	0.584375	0.559375	0.546875
12	0.703125	0.703125	0.703125	0.703125	0.584375	0.584375	0.603125	0.59375	0.615625
13	0.49375	0.5125	0.5625	0.565625	0.54375	0.5375	0.490625	0.515625	0.51875
14	0.68125	0.56875	0.56875	0.565625	0.540625	0.55625	0.515625	0.528125	0.509375
15	0.553125	0.6125	0.640625	0.640625	0.6	0.609375	0.6375	0.609375	0.590625
16	0.6875	0.7125	0.746875	0.746875	0.65	0.634375	0.621875	0.6125	0.596875
17	0.615625	0.59375	0.6125	0.6125	0.53125	0.5375	0.5375	0.53125	0.5125
18	0.690625	0.603125	0.575	0.575	0.525	0.48125	0.525	0.4625	0.49375
19	0.490625	0.559375	0.559375	0.559375	0.5375	0.54375	0.528125	0.528125	0.54375
20	0.66875	0.628125	0.615625	0.615625	0.578125	0.59375	0.553125	0.559375	0.55
21	0.6375	0.64375	0.55625	0.55625	0.440625	0.465625	0.45	0.5	0.425
22	0.66875	0.61875	0.61875	0.61875	0.64375	0.646875	0.61875	0.603125	0.61875
23	0.528125	0.64375	0.634375	0.63125	0.496875	0.509375	0.51875	0.503125	0.49375
24	0.553125	0.615625	0.6	0.6	0.584375	0.58125	0.54375	0.546875	0.534375
25	0.63125	0.671875	0.665625	0.665625	0.634375	0.615625	0.596875	0.621875	0.615625
26	0.55625	0.621875	0.61875	0.621875	0.575	0.525	0.503125	0.50625	0.525
27	0.60625	0.584375	0.609375	0.609375	0.403125	0.40625	0.440625	0.44375	0.4375
28	0.596875	0.471875	0.49375	0.49375	0.515625	0.5125	0.490625	0.509375	0.53125
29	0.56875	0.596875	0.5625	0.5625	0.546875	0.58125	0.534375	0.559375	0.546875
31	0.5875	0.5125	0.5375	0.5375	0.409375	0.40625	0.41875	0.41875	0.3875
32	0.69375	0.615625	0.546875	0.559375	0.534375	0.54375	0.490625	0.4875	0.490625
33	0.6	0.609375	0.609375	0.609375	0.4625	0.4625	0.48125	0.46875	0.48125
34	0.721875	0.678125	0.7125	0.7	0.5	0.49375	0.478125	0.48125	0.49375
36	0.6	0.5875	0.496875	0.5	0.515625	0.534375	0.546875	0.546875	0.5375
37	0.690625	0.69375	0.70625	0.70625	0.4625	0.44375	0.45625	0.453125	0.434375
38	0.70625	0.728125	0.653125	0.653125	0.575	0.571875	0.540625	0.540625	0.50625
39	0.521875	0.590625	0.571875	0.571875	0.5125	0.5125	0.53125	0.540625	0.5375
Mean	0.626194853	0.615625	0.610569853	0.610386029	0.544117647	0.538878676	0.5328125	0.532536765	0.527022059
Std Dev	0.081180822	0.081816209	0.081475556	0.081576125	0.067137445	0.065637134	0.062044982	0.0552902	0.059496399