Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Analyzing Data Augmentation Methods for Convolutional Neural Network-based Brain-Computer Interfaces

Gabriel Oliveira Moreira Faria

JUIZ DE FORA
JULHO, 2023

Analyzing Data Augmentation Methods for Convolutional Neural Network-based Brain-Computer Interfaces

Gabriel Oliveira Moreira Faria

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Heder Soares Bernardino

Coorientador: Alex Borges Vieira

JUIZ DE FORA

JULHO, 2023

Analyzing Data Augmentation Methods for Convolutional Neural Network-based Brain-Computer Interfaces

Gabriel Oliveira Moreira Faria

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Heder Soares Bernardino
Prof. Doutor em Modelagem Computacional

Alex Borges Vieira
Prof. Doutor em Ciência da Computação

Leonardo Goliatt da Fonseca
Prof. Doutor em Modelagem Computacional

Carlos Cristiano Hasenclever Borges
Prof. Doutor em Engenharia Civil

JUIZ DE FORA

11 DE JULHO, 2023

Resumo

Interfaces cérebro-máquina (ICMs) são sistemas que usam sinais cerebrais para interagir com dispositivos externos, com uma ampla variedade de aplicações que vão do entretenimento à área da saúde. Nas ICMs, sinais cebrais de imaginação motora são um dos tipos de sinal mais utilizados, permitindo que indivíduos controlem dispositivos externos através da simulação mental de movimentos do corpo. Nesse contexto, redes neurais convolucionais (RNCs) podem ser utilizadas para classificar sinais de eletroencefalograma (EEG) de imaginação motora e traduzi-los em comandos específicos. No entanto, esses sinais são geralmente ruidosos e podem variar significativamente ao longo do tempo e entre pessoas, o que frequentemente torna necessária a coleta de uma grande quantidade de dados para calibrar esses modelos. Esse processo pode ser demorado e cansativo para os usuários das ICMs, um problema particularmente significativo em aplicações em saúde. Para resolver esse problema, este trabalho investiga a eficácia do aumento de dados para reduzir a necessidade de coleta dados e melhorar a acurácia de classificação dos sinais. Cinco métodos de aumento de dados previamente documentados na literatura são avaliados em dois conjuntos de dados de referência de imaginação motora utilizando uma RNC de poucos parâmetros. As avaliações abrangem uma variedade de configurações de eletrodos de EEG, tarefas de imaginação motora e tamanhos de amostras de treinamento. Os métodos são então comparados entre si em termos de acurácia e efeitos sobre a distribuição dos dados. Os resultados indicam que o aumento de dados pode aliviar a necessidade de dados originais, resultando em maior acurácia mesmo com menos amostras de treinamento. Os resultados também mostram que a combinação de diferentes métodos de aumento de dados pode levar a melhorias adicionais na acurácia.

Palavras-chave: interface cérebro-máquina, imaginação motora, rede neural convolucional, aumento de dados.

Abstract

Brain-computer interfaces (BCIs) are systems that use brain signals to interact with external devices, with a wide variety of applications spanning from entertainment to healthcare. Within BCIs, motor imagery brain signals are one of the most commonly used, enabling individuals to control external devices by mentally simulating body movements. In this context, convolutional neural networks (CNNs) can be used to classify motor imagery electroencephalogram (EEG) signals and translate them into specific commands. However, these signals are usually noisy and can vary significantly over time and among people, making it frequently necessary to collect a large amount of data to calibrate these models. This process can be time-consuming and fatiguing for BCI users, a particularly significant issue in healthcare settings. In order to address this problem, this work investigates the effectiveness of data augmentation in reducing the need for data and improving classification accuracy. Five data augmentation methods previously documented in the literature are evaluated across two motor imagery benchmark datasets utilizing a few-parameter CNN. The evaluations encompass a variety of configurations of EEG electrodes, motor imagery tasks, and training sample sizes. Each method is then compared against one another in terms of both accuracy and data distribution effects. The overall findings indicate that data augmentation can alleviate the need for original data, resulting in higher accuracy even with fewer training samples. The findings also indicate that combining different data augmentation methods can lead to further improvements in accuracy.

Keywords: brain-computer interface, motor imagery, convolutional neural network, data augmentation.

Acknowledgements

The author expresses gratitude to Professors Heder S. Bernardino and Alex B. Vieira for their invaluable assistance and guidance not only during the development of this work, but also throughout his academic journey. The author also acknowledges the helpful comments provided by Gabriel H. de Souza. Furthermore, the author would like to extend appreciation for the support received from CAPES, CNPq, FAPEMIG, FAPESP, UFJF, and OpenBCI.

Contents

List of Figures

List of Tables

List of Abbreviations

AP	Amplitude Perturbation
BCI	Brain-Computer Interface
CNN	Convolutional Neural Network
CSP	Common Spatial Pattern
EEG	Electroencephalography
EMD	Empirical Mode Decomposition
ERD	Event-Related Desynchronization
ERS	Event-Related Synchronization
GNN	Graph Neural Network
IMF	Intrinsic Mode Function
ISTFT	Inverse Short-Time Fourier Transform
LDA	Linear Discriminant Analysis
NA	Noise Addition
RNN	Recurrent Neural Network
SR	Segmentation and Recombination
STFT	Short-Time Fourier Transform
SW	Sliding Window

1 Introduction

Brain-computer interfaces (BCIs) have a wide range of applications to improve accessibility, health, and quality of life for people, including wheelchair control (CARLSON; MILLAN, 2013), prosthesis operation (STAFFA; GIORDANO; FICUCIELLO, 2020), and neurorehabilitation (MANE; CHOUHAN; GUAN, 2020). In these applications, the most common paradigm is motor imagery. Motor imagery is a cognitive process that occurs when a person imagines a body movement. This process generates brain activity patterns that BCIs can classify to emit commands to external devices and provide feedback to the user. For example, a stroke patient can use a motor imagery BCI to move a robotic hand by imagining the movement of one hand during a rehabilitation treatment to recover upper limb movement.

Motor imagery signals are most commonly captured using electroencephalography (EEG), mainly due to its non-invasiveness and affordability compared with the alternatives (PAWAR; DHAGE, 2020). However, motor imagery EEG signals are often distorted by other brain signals and nervous activities, including involuntary muscle movements and eye blinks, resulting in a low signal-to-noise ratio (SINGH et al., 2021).

In this context, many approaches have been proposed to preprocess, extract features from, and classify these signals (SINGH et al., 2021). Recently, convolutional neural networks (CNNs) have been the most prevalent architecture in the deep learning domain (CRAIK; HE; CONTRERAS-VIDAL, 2019), having been successfully applied to perform end-to-end classification on motor imagery signals (LASHGARI et al., 2021; LAWHERN et al., 2018; SCHIRRMEISTER et al., 2017).

Nevertheless, machine learning—especially deep learning—often requires a large amount of training data. Moreover, creating a universal motor imagery BCI (LOTTE, 2015) is particularly challenging, as each person's brain signals are different and tend to change over time. Therefore, it is necessary to collect significant amounts of data to calibrate a motor imagery BCI before a user can operate it. This issue is especially relevant in applications such as neurorehabilitation, where patients may experience fatigue

and demotivation due to the mental effort required for motor imagery and the possibility of receiving inaccurate feedback from the BCI. In this scenario, increasing classification performance and reducing calibration times remain active research challenges (SINGH et al., 2021).

1.1 Related Work

Some works in the literature have addressed these issues. Particularly, several works have investigated data augmentation, the generation of artificial data from real data, as a means to solve this problem. For instance, Lotte (2015) evaluates several techniques to reduce calibration time in a classical machine learning-based BCI using common spatial pattern (CSP) for feature extraction and linear discriminant analysis (LDA) for classification. The evaluated techniques include not only data augmentation, but also regularization, user-to-user transfer, and semi-supervised. The author shows that data augmentation methods based on segmentation and recombination of input data improve accuracy in motor imagery classification, especially in low-data regimes, achieving competitive results with other approaches.

Meanwhile, Dinarès-Ferran et al. (2018) propose a new data augmentation method for motor imagery EEG data based on empirical mode decomposition (HUANG et al., 1998) and evaluate its effectiveness with an LDA classifier and a CSP feature extractor. The authors conclude that the artificially generated samples can partially replace the original training samples and thus reduce calibration times, achieving similar error rates as when using as little as 50% of the original samples.

Moreover, Lashgari, Liang and Maoz (2020) and He et al. (2021) survey and review several data augmentation methods used throughout the deep learning EEG literature, reporting on techniques based on noise addition, sliding windows, mathematical transforms, and deep generative models.

Nevertheless, the existing work involving data augmentation for motor imagery BCIs either includes a limited number of experiments and data augmentation techniques, does not evaluate performance on neural network-based classifiers, or does not directly address the reduction of BCI calibration times.

1.2 Goal Statement

In this context, this work presents the contributions from (FARIA et al., 2022), which further studies the extent to which data augmentation can improve classification accuracy and reduce calibration times in CNN-based motor imagery BCIs. Particularly, we analyze the effectiveness of five simple data augmentation methods from the literature (LASHGARI; LIANG; MAOZ, 2020; HE et al., 2021) in addressing these issues: Sliding Window (SCHIRRMEISTER et al., 2017), Noise Addition (LASHGARI et al., 2021), Amplitude Perturbation (LI et al., 2019), Segmentation and Recombination (LOTTE, 2015), and Empirical Mode Decomposition (DINARÈS-FERRAN et al., 2018). Additionally, based on our experimental results, we also propose a combination of two of these data augmentation methods. For our evaluations, we perform experiments with a few-parameter CNN on two motor imagery datasets, including multiple test scenarios with cases ranging from minimal to moderate quantities of training data.

The remainder of this work is structured as follows. Chapter 2 gives an overview of the fundamental concepts that are pertinent to our research. In Chapter 3, we present and elaborate on the datasets, neural network architecture, and data augmentation methods that we analyze. Chapter 4 defines the experimental protocol, presents the results, and provides an analysis and discussion of the findings. Finally, Chapter 5 concludes the work.

2 Background

The present work builds on a broad range of concepts, ranging from machine learning and neural networks to the functioning of the human brain. This chapter aims to provide an overview of these concepts to ensure that the work is reasonably self-contained, including recommendations for further reading when appropriate. The chapter starts with a concise introduction to brain-computer interfaces, highlighting how signals are collected and providing insights into the motor imagery cognitive process. It then introduces the fundamentals of machine learning, which plays a crucial role in decoding brain signals. Particularly, the chapter covers how computers learn from data and generalize to new situations and presents the concept of data augmentation. Finally, it explores deep neural networks, delving into their structure, algorithms, and optimization methods.

2.1 Brain-Computer Interfaces

There are many ways to define brain-computer interfaces (BCIs). According to Wolpaw et al. (2002), "a direct brain-computer interface is a device that provides the brain with a new, non-muscular communication and control channel." To Levine et al. (1999), "a direct brain interface (DBI) accepts voluntary commands directly from the human brain without requiring physical movement and can be used to operate a computer or other technologies." Meanwhile, Donoghue (2002) states that "a major goal of a BMI (brain-machine interface) is to provide a command signal from the cortex. This command serves as a new functional output to control disabled body parts or physical devices, such as computers or robotic limbs."

In this work, we define brain-computer interfaces as systems that allow human interaction with external devices through a direct connection between the brain and the machine. By bypassing traditional routes such as muscles or nerves, BCIs enable individuals to control and interact with technology using their minds alone, fundamentally changing the way we interact with machines in addition to opening up new possibilities

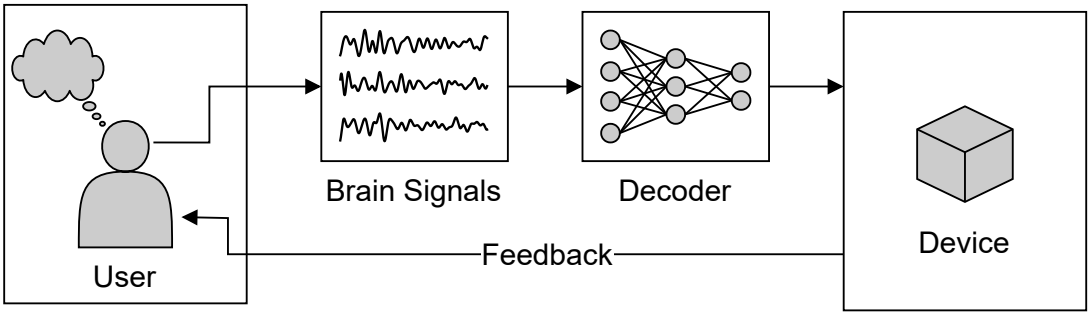


Figure 2.1: Brain-computer interface interaction loop.

for individuals with disabilities or neurodegenerative conditions.

The following sections provide a brief summary of key concepts in (GRAIMANN; ALLISON; PFURTSCHELLER, 2010), which offers an introduction to brain-computer interfaces.

2.1.1 Overview

The main idea behind BCIs is the ability to decode and interpret brains signals and translate them into meaningful commands. This process involves capturing neural activity from the brain, processing and analyzing the signals, and then generating an output that can be used to control external devices or systems. In order to establish this communication pathway, BCIs typically employ a combination of concepts from the fields of neuriphisiology, signal processing, and machine learning.

The main interaction loop of a BCI is depicted in Figure 2.1. The loop consists of four essential components: the user, the brain activity acquisition, the signal decoding, and the device to be controlled.

First, the user generates specific brain activity patterns through various mental tasks or cognitive processes. Such brain activities can be recorded and captured through different means.

Next, the acquired brain signals are passed through the signal decoding step. Associated with this step is signal preprocessing, which typically include filtering, artifact removal, and noise reduction to enhance the quality of the signals. Relevant features are then extracted from the preprocessed signals and fed into a classification or decoding algorithm. This algorithm learns from training data to identify and classify specific brain

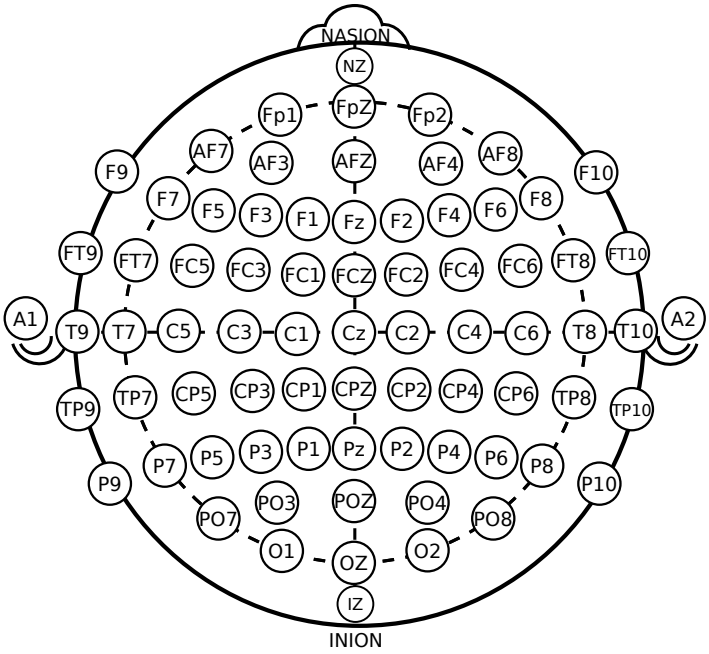


Figure 2.2: International 10-10 system for EEG electrode placement.
From (OXLEY, 2021).

patterns that correspond to different mental tasks or commands.

Finally, the output from the decoding algorithm is used to control the external device or computer system. This output can be in the form of commands, cursor movements, robotic actions, or any other action depending on the specific application of the BCI. The controlled device or system responds accordingly, establishing an interactive loop connecting the user's brain activity with the external environment.

Importantly, BCIs possess bidirectional capabilities, not only enabling brain control of external devices but also providing sensory feedback to the user. This feedback can be visual, auditory, or tactile, delivering information about the current state of the controlled device or system to the user.

2.1.2 Electroencephalography

Electroencephalography (EEG) is one of the primary modalities used in BCIs. It involves the measurement and recording of electrical activity generated by the brain using electrodes placed on the scalp. EEG offers several advantages, such as non-invasiveness, high temporal resolution, and relatively low cost (RAMADAN; VASILAKOS, 2017), while being able to capture neural signals associated with different cognitive and motor processes.

EEG brain signals are obtained by placing multiple electrodes on the scalp, following specific electrode arrangements. One of the most common of such arrangements is the 10-10 system, illustrated in Figure 2.2. By strategically positioning electrodes according to anatomical landmarks, this system can capture electrical potentials generated by the synchronized activity of large populations of neurons.

These EEG signals exhibit distinctive patterns of rhythmic oscillations, which are classified into different frequency bands. The main frequency bands include delta (0.5-4 Hz), theta (4-8 Hz), alpha (8-13 Hz), beta (13-30 Hz), and gamma (30-100 Hz). Each frequency band is associated with specific brain states and cognitive functions. As such, these frequency bands serve as windows into the dynamic activity of the brain, providing insights into different mental states, attentional processes, and cognitive tasks.

Two disadvantages of EEG are its sensitivity to noise from the environment and its relatively low spatial resolution. In this context, signal preprocessing techniques are commonly applied to EEG data to improve the quality and extract relevant information. These techniques involve filtering, artifact removal, and noise reduction to enhance the signal-to-noise ratio. Additionally, preprocessing steps also include segmenting the data into epochs, which correspond to specific task-related events or intervals.

2.1.3 Motor Imagery

Motor imagery is a key paradigm utilized in BCIs. It involves the mental simulation of motor actions without actual physical movement. By imagining specific motor tasks, individuals generate distinct patterns of brain activity that can be detected and decoded by BCIs. Exploiting this phenomenon, motor imagery-based BCIs provide a means to voluntarily control external devices using brain activity.

During motor imagery, the brain produces event-related desynchronization (ERD) or event-related synchronization (ERS) patterns. ERD refers to a decrease in the power of specific frequency bands, while ERS represents an increase. For instance, when an individual engages in motor imagery of movement in their left hand, it can lead to a noticeable ERD in the beta frequency band over the corresponding sensorimotor cortex region of the brain. Conversely, if the individual were to imagine movement in their

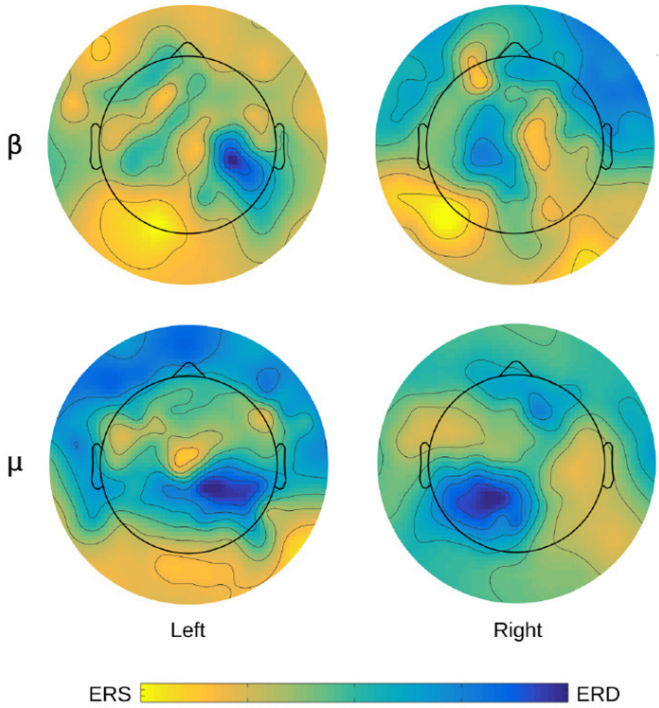


Figure 2.3: Maps of signal strength (μV^2) distribution on the scalp (μ = 8–12 Hz; β = 18–30 Hz) during imagined movement of the left or right hand.
Adapted from (WIERZGAŁA et al., 2018).

right hand, a similar analysis would reveal an ERD in the beta frequency band over the sensorimotor cortex on the opposite hemisphere. A visualization of this phenomenon through EEG signals is depicted in Figure 2.3.

Decoding such motor imagery-related brain signals requires the development of robust machine learning algorithms. These algorithms learn to classify different patterns of brain activity associated with specific motor imagery tasks. By combining EEG measurements with signal processing and machine learning algorithms, motor imagery-related brain signals can be translated into actionable commands for controlling robotic limbs, virtual avatars, and other devices.

2.2 Machine Learning

Machine learning is a field of study that focuses on developing computer algorithms that learn from data. Rather than being explicitly programmed, machine learning algorithms use computational methods to identify patterns and relationships within datasets, allow-

ing them to improve their accuracy and effectiveness over time[1]. For example, given an appropriate set of input-output pairs of data, a machine learning algorithm could learn how to detect objects in an image.

There are three main machine learning paradigms: supervised learning, unsupervised learning, and reinforcement learning (BISHOP, 2006). In supervised learning, the algorithm is trained on labeled data, with known inputs and outputs, to learn how to predict new outputs from new inputs. Meanwhile, unsupervised learning involves training the algorithm on unlabeled data, allowing it to discover patterns and relationships without explicit guidance. In turn, reinforcement learning is concerned with agents that learn by interacting with their environment, taking actions with the goal of maximizing rewards over time.

The most widely used paradigm in machine learning, and the one used throughout this work, is supervised learning. Therefore, the next section covers this approach exclusively. For an extensive discussion on machine learning and related topics, the reader is referred to (MURPHY, 2022).

2.2.1 Supervised Learning

In supervised learning, a model learns to predict outputs from inputs based on a labeled dataset. For example, using a dataset of images labeled as containing or not containing a cat, we can train a cat recognition model that uses pixel data as input. To do so, we optimize the parameters of the model to minimize the difference between the predicted outputs and the target outputs, which is indicated by a loss function.

More formally, the supervised learning problem consists in learning a mapping $f : \mathcal{X} \to \mathcal{Y}$ from inputs $\boldsymbol{x}$ to outputs $\boldsymbol{y}$, parameterized by $\boldsymbol{\theta}$. Given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$, the goal is to find the function parameters $\boldsymbol{\theta}^*$ that minimize the loss $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})$ between the predicted outputs $\hat{\boldsymbol{y}}$ and the target outputs $\boldsymbol{y}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i) \tag{2.1}$$

[1]This definition is inspired by a quote commonly attributed to Arthur Samuel that "Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed".

This process of fitting the parameters to the training data is called empirical risk minimization (VAPNIK, 1998), and the term being minimized is called the empirical risk.

However, the goal of supervised learning is not simply to fit the training data, but rather to generalize to unseen samples. As Mitchell (1980) argues, generalization necessarily involves induction, which means deriving a general rule from a limited set of examples. Therefore, inductive biases are essential for machine learning models to generalize, as they determine the most suitable hypothesis[2] from a set of equally plausible options that fit the training data. Such biases originate from factors such as model architecture, learning procedure, and distribution of initial parameters. Importantly, they often dictate how well a model can generalize in a given problem.

In order to estimate the generalization performance of the model, a separate test dataset is typically used. A model that performs well on the test set is said to generalize, indicating that it has learned to make accurate predictions on new data. However, if a model performs well on the training set but poorly on the test set, it is said to have overfitted, indicating that it has memorized the training data instead of learning to make more general predictions.

One technique frequently used to avoid overfitting is regularization, in which a constraint is imposed on the parameters of the model to prevent it from fitting the training data too closely. This is often accomplished by introducing an additional penalty term to the loss function.

To conclude this discussion, it is worth mentioning the two main types of supervised learning: classification, in which the outputs are discrete categories, and regression, in which they are numeric values. Since this work revolves around a classification problem, the following section exclusively covers this topic.

2.2.2 Classification

Classification is a particular supervised learning setting that consists in learning a class label predictor function. Given an input, a classifier predicts which class it belongs to. For example, Figure 2.4 shows the decision boundary of a classifier that determines what

[2]Here, a hypothesis refers to a set of model parameters, whereas the set of admissible hypotheses is known as the hypothesis space.

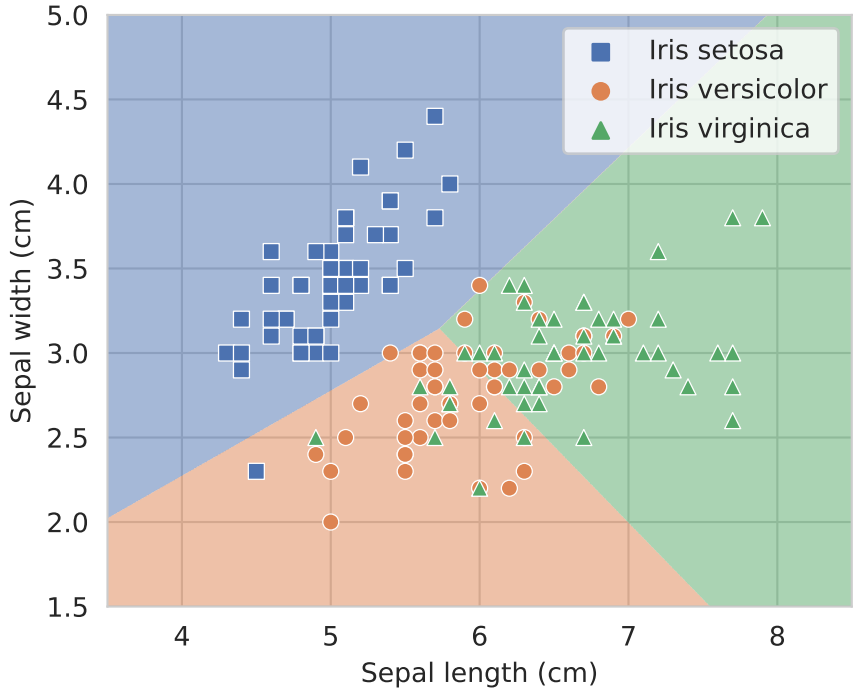


Figure 2.4: Decision boundary of a classifier on the Iris dataset. Adapted from (GÉRON, 2019).

flower species an individual belongs to based on sepal length and width measurements.

Typically, a classification model outputs the probability that an input sample belongs to each class, assigning it to the class with the highest probability. When the classification problem has only two classes (usually referred to as the positive and the negative class), the model is called a binary classifier and is commonly optimized using the binary cross-entropy loss function:

$$\ell(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y}) \tag{2.2}$$

In the equation above, $y \in \{0, 1\}$ indicates whether the sample belongs to the positive class, while $\hat{y}$ represents the predicted probability that it does. In the more general multiclass classification scenario, in which the problem involves K possible classes, the categorical cross-entropy loss function is used instead:

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{j=1}^{K} y^{(j)} \log \hat{y}^{(j)} \tag{2.3}$$

In this case, the output vector $\hat{\boldsymbol{y}}$ represents the predicted probabilities for each class,

whereas the vector $\boldsymbol{y}$ indicates the true class labels using one-hot encoding.

Specifically, one-hot encoding refers to a method of representing categorical variables using sparse binary vectors where each category is indicated by a single non-zero element. For example, in a color classification problem where the classes belong to the set $\{\text{red}, \text{green}, \text{blue}\}$, red could be represented as $[1, 0, 0]$, green as $[0, 1, 0]$, and blue as $[0, 0, 1]$.

2.2.3 Data Augmentation

When using supervised learning algorithms, we may sometimes need considerable amounts of data in order to solve a problem. However, in many instances, the availability of labeled data is limited, which can compromise the performance of trained models. To make up for this gap, we can use data augmentation.

Data augmentation is a strategy that involves generating artificial samples from available real samples. Essentially, we create new samples by applying some transformation to the original samples while keeping their labels intact, incorporating the generated data points into the training dataset. The resulting augmented dataset usually encompasses a broader range of variations and patterns, leading to more a robust model after training.

Particularly, data augmentation can be thought of as a way of injecting inductive bias into the model. For example, in an image classification task, if we want the model to be invariant to shifting in the input, we can provide randomly shifted images as training samples. Similarly, if we want the model to be robust to small amounts of noise, we can perturb the inputs of the model with noise while keeping target labels the same. Through this process, data augmentation can often improve generalization performance, especially when few data points are available.

2.3 Deep Neural Networks

Deep neural networks are a powerful type of machine learning model known for their remarkable ability to learn and fit complex data, having led to significant advances in

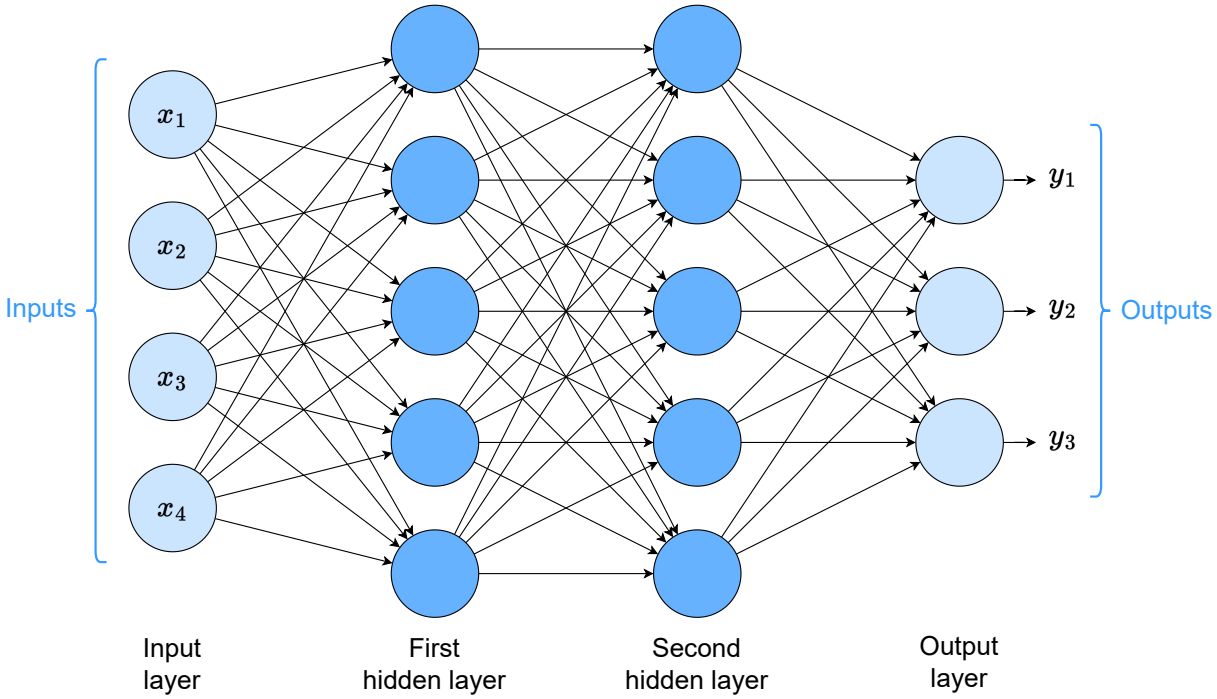


Figure 2.5: Illustration of a feedforward neural network.

computer vision, speech recognition, and natural language processing. Originally inspired by the structure and function of the biological brain, neural networks have the neuron as their basic processing unit. Organized as a series of interconnected neurons, these networks compute the output by applying a sequence of differentiable linear and nonlinear transformations to the input, with each neuron contributing to the overall computation.

Neural networks are highly flexible, allowing for a wide variety of model architectures, such as the convolutional neural network (CNN) for image data, the graph neural network (GNN) for graph data, and the recurrent neural network (RNN) for sequence data. In the following section, we begin exploring the most basic neural network architecture: the feedforward neural network.

2.3.1 Feedforward Neural Networks

Feedforward neural networks (FFNs) are neural networks in which information flows in a single direction, from input to output neurons. The simplest of such networks is the multilayer perceptron (MLP), shown in Figure 2.5. This network consists in a sequence of fully connected layers of neurons, where each layer feeds into the next. Starting at the input layer, the information is processed through one or more hidden layers before being

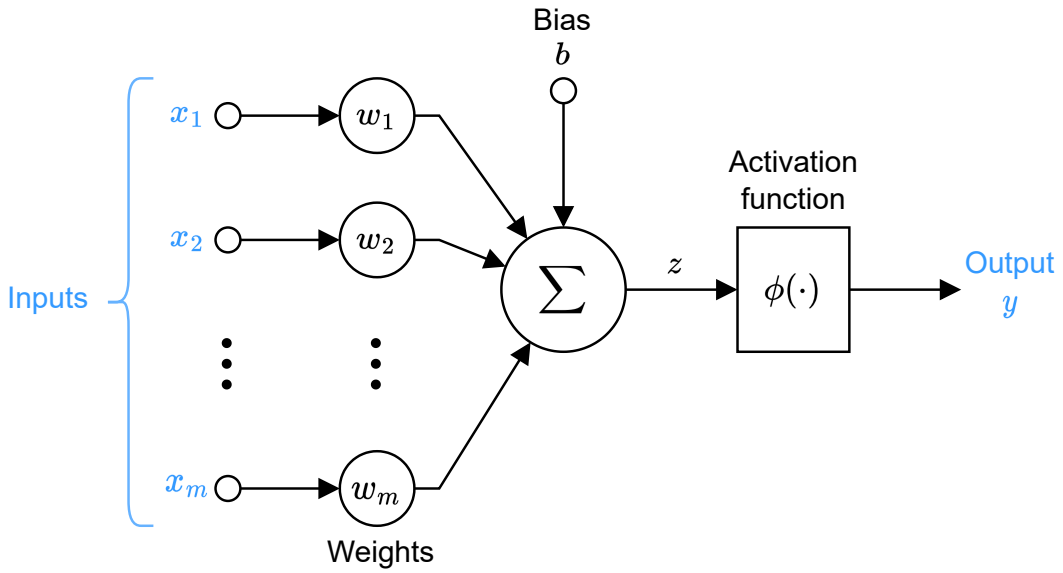


Figure 2.6: The artificial neuron model.
Adapted from (HAYKIN, 2009).

output at the final layer.

Each neuron in the network follows the model shown in Figure 2.6. In this model, a neuron takes one or more input signals, multiplies each by its corresponding weight, and then adds the weighted inputs along with a bias value. The resulting sum is then passed through a nonlinear activation function, which allows neural networks to model complex relationships between inputs and outputs. Mathematically, each neuron computes an affine transformation of its inputs followed by an activation function:

$$y = \phi\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{2.4}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the input vector, $\boldsymbol{w} \in \mathbb{R}^m$ is the weight vector, $b \in \mathbb{R}$ is the bias value, and $\phi(\cdot)$ is the activation function.

There are many possible activation functions. Some of the most widely used ones are shown in Figure 2.7. The sigmoid activation function (denoted σ), for example, can be understood as a smoothed version of the step function, in which the output is one when the input is positive and zero otherwise. The hyperbolic tangent (tanh) function, in turn, is similar to the sigmoid function but with output values ranging from -1 to 1. Finally, the ReLU activation function is linear on the positive domain and zero otherwise.

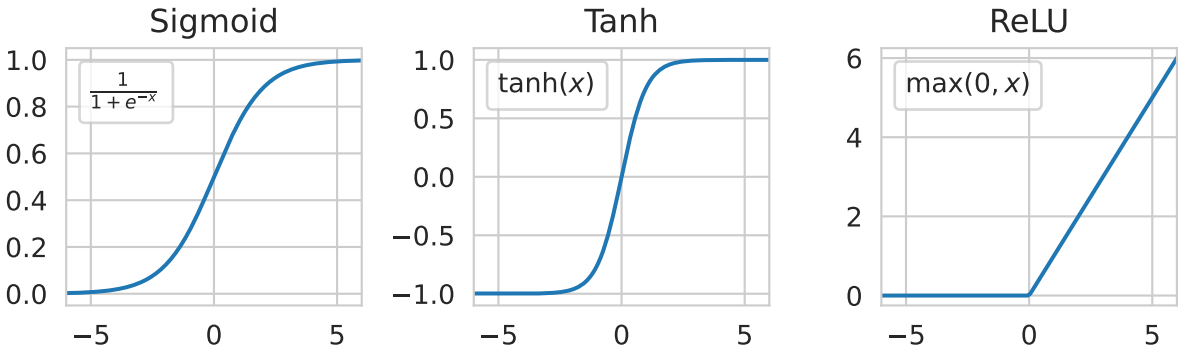


Figure 2.7: Three common activation functions.

Another notable activation function is the softmax, defined as:

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)} \tag{2.5}$$

This function normalizes a set of input values into a probability distribution, in which the output values add up to one. Thus, it is commonly used to compute class probabilities in multiclass classification problems[3].

In summary, each layer in an MLP computes a linear transformation of its inputs plus a bias vector, followed by the application of a nonlinear activation function. The process of computing the output of the network layer by layer is called forward pass and is described in Algorithm 1.

Algorithm 1: Neural network forward pass.

Input : $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$, vector of input values.
Output : $\boldsymbol{y} \in \mathbb{R}^{d_{\text{out}}}$, vector of output values.
Parameters: $\boldsymbol{W}_l \in \mathbb{R}^{d_{l-1} \times d_l}$, $\boldsymbol{b}_l \in \mathbb{R}^{d_l}$

begin
  $\boldsymbol{a}_0 \leftarrow \boldsymbol{x}$
  for $l \leftarrow 1$ **to** L **do**
    $\boldsymbol{z}_l = \boldsymbol{W}_l \boldsymbol{a}_{l-1} + \boldsymbol{b}_l$ `// compute the pre-activation` $\boldsymbol{z}_l$
    $\boldsymbol{a}_l = \phi_l(\boldsymbol{z}_l)$ `// apply the nonlinear activation function` ϕ_l
  end for
  return $\boldsymbol{y} = \boldsymbol{a}_L$
end

[3]In practice, the softmax activation function is not used in the model's output layer. Instead, the model usually outputs logits while the softmax computation is incorporated into the loss function.

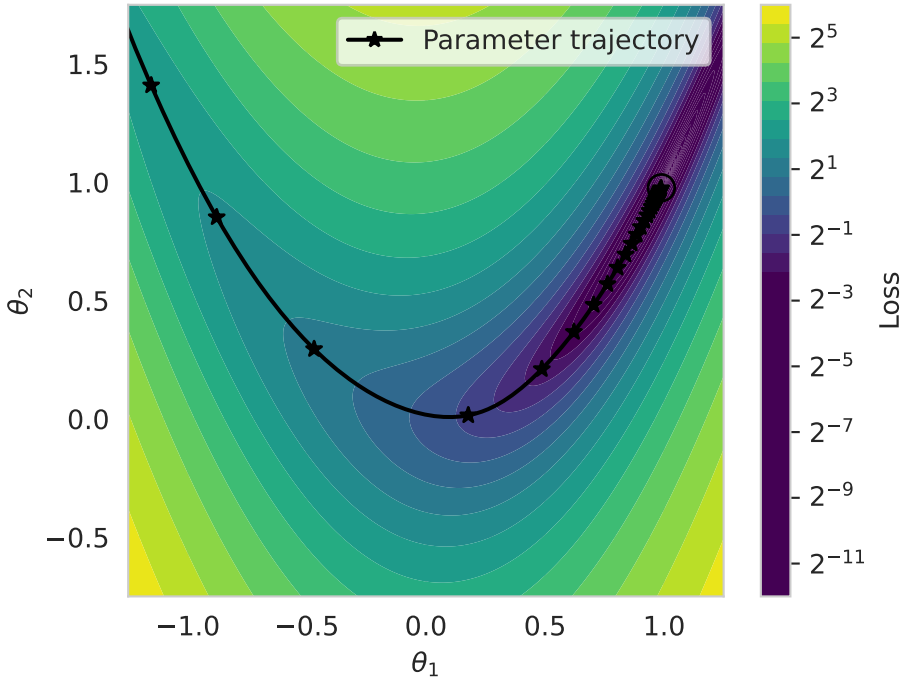


Figure 2.8: Illustration of gradient descent optimization over a loss landscape.

2.3.2 Gradient Descent

Gradient descent is a general gradient-based algorithm that can efficiently optimize differentiable cost functions. It is used in many fields of science and engineering for the optimization of functions of continuous variables, especially in the field of machine learning. Neural networks, being differentiable models, use gradient descent to learn their weight and bias parameters and minimize the loss function.

The optimization process begins by randomly initializing the parameters of the model. Then, the gradient descent algorithm iteratively updates the parameters according to the gradients of the loss function. At each iteration, the algorithm takes a small step to decrease the value of the loss function, where the gradients indicate the direction of the parameter updates. After many iterations, the algorithm usually converges to a local minimum. This process is illustrated in Figure 2.8.

In practice, neural networks typically use the stochastic gradient descent algorithm, which randomly samples a portion of the training dataset at each optimization step. To compute the loss function, the algorithm samples a batch $\mathcal{B} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{m}$ of

instances from the training set and calculates the average loss over the batch:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{y}_i) \tag{2.6}$$

This approach can be more efficient than using the entire dataset while still providing good gradient estimates. Based on this, the stochastic gradient descent algorithm minimizes the loss function by iteratively applying the following update rule:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(t)}) \tag{2.7}$$

where λ is a hyperparameter called the step size or learning rate, which controls the magnitude of the parameter updates at each step. Notably, choosing an appropriate learning rate is crucial for ensuring the algorithm's convergence and overall performance.

One limitation of vanilla stochastic gradient descent is its poor performance in narrow valleys, where it may exhibit zig-zag patterns, and nearly flat surfaces such as saddle points, which it may take a long time to traverse. Several modifications of gradient descent have been proposed to address these shortcomings and improve performance. These modifications incorporate concepts such as momentum and have led to the development of advanced optimizers such as Adam (KINGMA; BA, 2014).

To conclude, it is worth noting that the treatment of gradient descent in this section is very succint. For a more in-depth exploration of gradient descent and other optimization techniques, the reader is referred to (KOCHENDERFER; WHEELER, 2019).

2.3.3 Backpropagation

In the previous section, we discussed how neural networks use gradient descent for optimization. However, to apply gradient descent, it is necessary to compute the gradients of the loss function with respect to the network parameters. For this purpose, neural networks use the backpropagation algorithm.

The backpropagation algorithm, at its core, leverages the chain rule to efficiently compute gradients. It consists of two passes through the network: a forward pass and a backward pass. In the forward pass, the input is propagated through the network, layer

by layer, to compute the final output and the loss value. In the backward pass, gradients are computed recursively, starting from the output layer and moving back to the input layer. For instance, we have that:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_3} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}_4} \frac{\partial \boldsymbol{a}_4}{\partial \boldsymbol{\theta}_3} \tag{2.8}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_2} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}_4} \frac{\partial \boldsymbol{a}_4}{\partial \boldsymbol{a}_3} \frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{\theta}_2} \tag{2.9}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_1} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}_4} \frac{\partial \boldsymbol{a}_4}{\partial \boldsymbol{a}_3} \frac{\partial \boldsymbol{a}_3}{\partial \boldsymbol{a}_2} \frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{\theta}_1} \tag{2.10}$$

More generally, in an L-layer MLP, the gradients with respect to the parameters of the layer l can be computed as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_l} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}_L} \left(\prod_{i=l+1}^{L} \frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{a}_{i-1}} \right) \frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{\theta}_l} \tag{2.11}$$

In practice, these gradients are automatically computed by modern deep learning libraries using a technique called reverse mode differentiation. This technique can analytically compute gradients for any differentiable computational graph, which neural networks are a special case of.

3 Materials and Methods

In this work, we analyze the effectiveness of data augmentation methods on motor imagery EEG classification. This chapter presents the essential components involved in our analysis. It starts by describing the datasets employed in the experiments, including their characteristics and data collection protocols. Next, it delves into the neural network architecture of the CNN-based classifier employed in the evaluation of the data augmentation methods. Lastly, the chapter concludes by providing a detailed account of each data augmentation method explored in our analyses.

3.1 Datasets

We use two motor imagery EEG datasets from BCI Competition IV (TANGERMANN et al., 2012), which are commonly used in the literature as a benchmark for motor imagery classification algorithms. The datasets are described in the following sections.

3.1.1 BCIC-IV 2a

The dataset 2a from BCI Competition IV consists of data from four motor imagery tasks: left hand, right hand, tongue, and both feet imagination. The data comprise EEG recordings from 22 electrodes placed according to the 10-20 system. The EEG signals were sampled at 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz. The dataset includes data from 9 healthy subjects, who participated in two sessions on different days, each consisting of 288 trials (72 for each task).

Each trial is approximately 8 seconds long and adheres to the protocol shown in Figure 3.1, which is defined as follows: (i) 0–2 s: a fixation cross appears on the screen and a beep is played; (ii) 2–3.25 s: a cue appears on the screen, indicating the motor imagery task the participant should perform. (iii) 3–6 s: the participant imagines the movement; and (iv) 6 s: the fixation cross disappears and a short break is provided before the next trial.

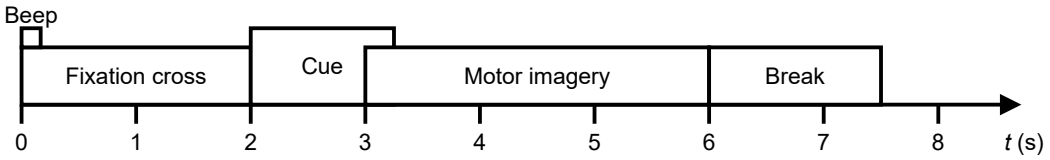


Figure 3.1: Structure of a trial in the BCIC-IV 2a dataset.
Adapted from (TANGERMANN et al., 2012).

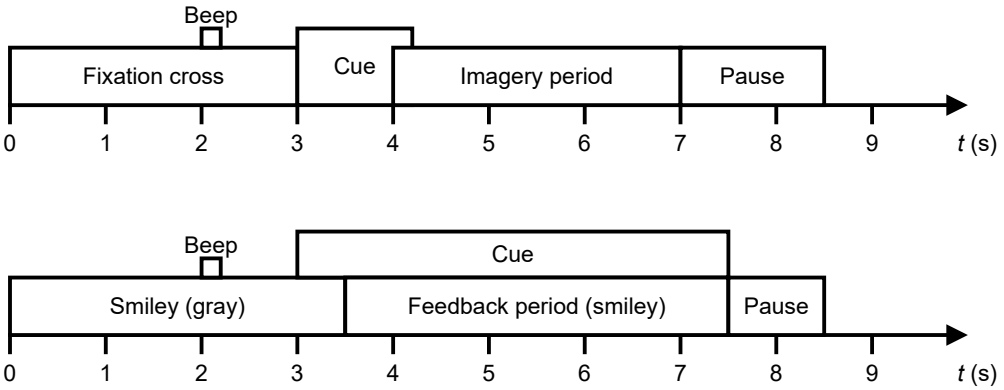


Figure 3.2: Structure of a trial in the BCIC-IV 2b dataset. Without feedback (upper) and with feedback (lower).
Adapted from (TANGERMANN et al., 2012).

3.1.2 BCIC-IV 2b

The dataset 2b from BCI Competition IV consists of data from two motor imagery tasks: left hand and right hand imagination. The data comprise EEG recordings from 3 electrodes: C3, Cz, and C4. Notably, these electrodes encompass the brain region where motor imagery activity is most prominent (GRAIMANN; ALLISON; PFURTSCHELLER, 2010). The EEG signals were sampled at 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz. The dataset includes data from 9 healthy subjects, where each subject participated in five recording sessions with at least 120 trials each (60 for each task). The first two sessions, recorded on different days, had no feedback, whereas the last three provided feedback to the user.

The structure of the trials from this dataset is shown in Figure 3.2. For the sessions without feedback, each trial is approximately 9 seconds long and adheres to the following protocol: (i) 0–3 s: a fixation cross appears on the screen and a beep is played; (ii) 3–4.25 s: a cue appears on the screen, indicating the target motor imagery task. (iii) 4–7 s: the subject imagines the movement; and (iv) 7 s: the fixation cross disappears and a short break is provided before the next run.

Table 3.1: Summary of the EEGNet architecture.

Block	Layer	# Filters	Kernel Size	Output Size
Block 1	Input			$(1, E, T)$
	Conv2d	F_1	$(1, K)$	(F_1, E, T)
	BatchNorm2d			(F_1, E, T)
	DepthConv2d[1]	$D \times F_1$	$(E, 1)$	$(D \times F_1, 1, T)$
	BatchNorm2d			$(D \times F_1, 1, T)$
	ELU			$(D \times F_1, 1, T)$
	AvgPool2d		$(1, 4)$	$(D \times F_1, 1, T//4)$
	Dropout			$(D \times F_1, 1, T//4)$
Block 2	SepConv2d	F_2	$(1, 16)$	$(F_2, 1, T//4)$
	BatchNorm2d			$(F_2, 1, T//4)$
	ELU			$(F_2, 1, T//4)$
	AvgPool2d		$(1, 8)$	$(F_2, 1, T//32)$
	Dropout			$(F_2, 1, T//32)$
Classifier	Dense[2]	$N \times (F_2 \times T//32)$		N
	Softmax			N

[1] The weights of the layer are constrained such that $\|w\|^2 < 1$.
[2] The weights of the layer are constrained such that $\|w\|^2 < 0.25$.

For the sessions with feedback, each trial is also approximately 9 seconds long and obeys the following protocol: (i) 0–3.5 s: a gray smiley face appears on the screen and a beep is played; (ii) 3–7.5 s: a cue appears on the screen, indicating the target motor imagery task. (iii) 3.5–7.5 s: the subject imagines the movement and receives feedback from the BCI through the smiley face; and (iv) 7.5 s: the smiley face disappears and a short break is provided before the next run.

3.2 Convolutional Neural Network Architecture

Convolutional neural networks (CNNs) are a special type of neural network based on the convolution operation. Each CNN layer consists of a set of learned filters which are slid across the input in order to perform local feature extraction. By doing so, CNNs can detect meaningful patterns at different positions of the input.

Owing to their structure, CNNs are particularly well-suited for processing signal data such as EEG, being able to detect various spatial and frequency patterns at different moments in time. This motivates the use of a CNN architecture in this work. In the following section, we discuss the specific convolutional architecture selected for our

analysis.

3.2.1 EEGNet

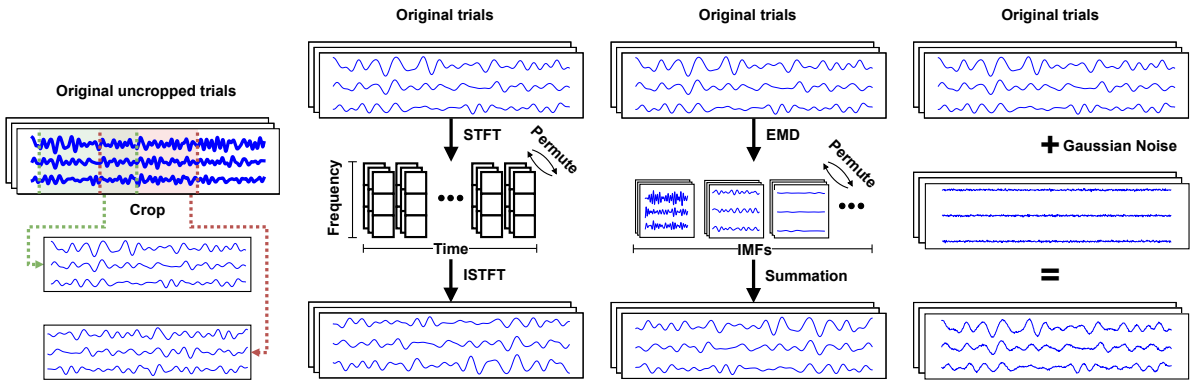
Since our analyses include scenarios with very few training data and considering the limitations associated with deep learning computational resources for both training and inference (WANG et al., 2020), we use EEGNet. EEGNet is a few-parameter CNN for EEG-based BCIs (LAWHERN et al., 2018) which can achieve good classification performance with few training data. EEGNet takes raw motor imagery EEG signals as input in order to classify imagined movement. Its architecture uses separable and depthwise convolutions to extract features from the input signals while reducing the number of parameters and includes several regularization and normalization mechanisms. The architecture of the network is summarized in Table 3.1. The model consists of three blocks, where the first two are dedicated to feature extraction and the last one is dedicated to classification.

In Block 1, two convolutional steps are performed sequentially. The first step involves applying 2D convolutional filters to capture frequency information. These filters generate multiple feature maps which we can interpret as representing different bandpass frequencies. The second step utilizes a depthwise convolution, which learns spatial filters for each temporal filter in the previous step. This allows for efficient extraction of frequency-specific spatial filters, enhancing the processing of EEG data.

In Block 2, a separable convolution is employed, which consists of a depthwise convolution followed by pointwise convolutions. The depthwise convolution captures temporal information from individual feature maps, summarizing each map independently. This step is useful for EEG signals as different feature maps may represent data at various time scales. The pointwise convolutions then merge the outputs, combining the summarized feature maps. The main advantages of separable convolutions are reducing the number of parameters and explicitly decoupling relationships within and across feature maps, resulting in more efficient and effective processing of EEG data.

In both blocks, the exponential linear unit (ELU) nonlinearity[4] is used. Each

[4]The ELU activation function is defined as $\phi(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$.



(a) Sliding Window (b) Segmentation and Recombination (c) Empirical Mode Decomposition (d) Noise Addition

Figure 3.3: Visual scheme of the analyzed data augmentation methods. Amplitude Perturbation was omitted for brevity.

nonlinearity is preceded by a batch normalization layer in order to improve stability. Additionally, the network includes weight normalization on both the first block and the classifier block. Lastly, each of the two first blocks include a dropout layer for regularization and an average pooling layer for dimensionality reduction.

It should be noted that EEGNet can be configured according to several hyperparameters. The exact configurations used in this work are detailed in Section 4.1.4.

3.3 Data Augmentation Methods

We evaluate five simple, popular EEG data augmentation methods previously reported in the literature: Sliding Window (SCHIRRMEISTER et al., 2017), Noise Addition (LASHGARI et al., 2021), Amplitude Perturbation (LI et al., 2019), Segmentation and Recombination (LOTTE, 2015), and Empirical Mode Decomposition (DINARÈS-FERRAN et al., 2018). These augmentation methods rely on simple transformations and require no training, unlike approaches that involve deep generative modeling (LASHGARI; LIANG; MAOZ, 2020; HE et al., 2021). Consequently, we exclusively consider these techniques for our analysis. Each data augmentation method is visually represented in Figure 3.3 and further described in detail over the following sections.

3.3.1 Sliding Window

Sliding Window generates new training samples by shifting an input window across trials by a random number of timesteps, essentially translating and cropping the EEG signals. For instance, if the classification model takes 2 seconds of motor imagery signals as input and each trial contains 3 seconds of motor imagery, we can create multiple samples by cropping 2-second windows at various positions. This approach serves two purposes: mitigating overfitting to specific temporal locations of features and maximizing data utilization when the input window size is smaller than the motor imagery period in each trial.

The generation of an artificial trial $\tilde{Z}^{(i)} \in \mathbb{R}^{E \times T}$, where E is the number of electrodes and T is the number of timesteps in the input window, can be expressed as

$$\tilde{Z}^{(i)} = Z^{(i)}_{1...E,t...t+T} \tag{3.1}$$

where $Z^{(i)} \in \mathbb{R}^{E \times T}$ is the ith trial of the original dataset and $t \in \{t_{\min}, \ldots, t_{\max}\}$ is the randomly selected initial timestep of the shifted input window.

3.3.2 Segmentation and Recombination

Segmentation and Recombination works by splitting original trials into several segments and then mixing segments from different trials of the same class. This process is done in the time-frequency domain to avoid introducing noise between concatenated segments from different trials (LOTTE, 2015). The method operates in three steps:

1. First, each trial is transformed into its time-frequency representation through the application of a short-time Fourier transform (STFT), as in $TF^{(i)} = \mathrm{STFT}(Z^{(i)})$. We define $Z^{(i)} \in \mathbb{R}^{E \times T}$ as the ith trial of the original dataset and $TF^{(i)} \in \mathbb{C}^{E \times M \times N}$ as its time-frequency representation, where E is the number of electrodes, T is the number of timesteps in the trial, M is the number of frequency bins in the Fourier spectrum, and N is the number of time windows of the time-frequency representation.

2. Then, windows from different trials of the same class are randomly permuted to

create a new time-frequency representation:

$$\tilde{TF}^{(i)} = [TF_1^{(P_1)}, TF_2^{(P_2)}, \ldots, TF_n^{(P_n)}] \tag{3.2}$$

where $\tilde{TF}^{(i)} \in \mathbb{C}^{E \times M \times N}$ is the artificial time-frequency representation and $TF_n^{(P_i)} \in \mathbb{C}^{E \times M}$ is the nth time window of the time-frequency representation of the original trial P_i.

3. Finally, an inverse short-time Fourier transform (ISTFT) is applied to the concatenated windows, generating an artificial trial $\tilde{Z}^{(i)} \in \mathbb{R}^{E \times T}$, as in $\tilde{Z}^{(i)} = \text{ISTFT}(\tilde{TF}^{(i)})$.

3.3.3 Empirical Mode Decomposition

Empirical mode decomposition (EMD) is a signal processing and analysis technique that can decompose nonlinear and non-stationary signals into a finite set of intrinsic mode functions (IMF) (HUANG et al., 1998). More precisely, the method decomposes an arbitrary signal $X \in \mathbb{R}^L$, where L is the length of the signal, into a set of IMFs $C = \{c_i \in \mathbb{R}^L \mid i \in 1 \ldots n\}$ and a final residue $r_n \in \mathbb{R}^L$ such that the signal can be recovered through $X = \sum_{i=1}^{n} c_i + r_n$.

Based on this relation, the use of EMD as a data augmentation method was proposed in (DINARÈS-FERRAN et al., 2018). The method works by randomly permuting and summing IMFs from different trials of the same class. First, the IMFs of each channel of each trial of the original dataset are calculated. Then, a new trial $\tilde{Z}^{(i)} \in \mathbb{R}^{E \times T}$ is generated:

$$\tilde{Z}^{(i)} = \sum_{j=1}^{N} C_j^{(P_j)} + R_n^{(i)} \tag{3.3}$$

where $C_j^{(P_j)} \in \mathbb{R}^{E \times T}$ denotes the jth IMFs (for every electrode) of the trial P_j, $R_n^{(i)} \in \mathbb{R}^{E \times T}$ denotes the residues (for every electrode) of the ith trial of the original dataset, and N is the maximum number of IMFs of a signal in the dataset. Importantly, the signals do not all yield the same number of IMFs, as it varies according to their characteristics. Thus, we set the jth IMF c_j of a signal to zero when it is undefined.

3.3.4 Noise Addition

Noise Addition produces new training samples by perturbing the raw signals of the original trials with Gaussian noise. The idea behind this method is that the classifier should be invariant to small deviations in the signal. The generation of an artificial trial $\tilde{Z}^{(i)} \in \mathbb{R}^{E \times T}$ can be expressed as

$$\tilde{Z}^{(i)} = Z^{(i)} + P \tag{3.4}$$

where $Z^{(i)} \in \mathbb{R}^{E \times T}$ is the ith trial of the original dataset and $P \in \mathbb{R}^{E \times T}$ is the perturbation drawn from a normal distribution $P \sim \mathcal{N}(\mu, \sigma^2)$.

3.3.5 Amplitude Perturbation

Amplitude Perturbation is similar to Noise Addition but instead adds Gaussian noise to the amplitudes of the signals in the time-frequency domain. The method works as follows:

1. First, each trial is transformed into its time-frequency domain $TF^{(i)} \in \mathbb{C}^{E \times M \times N}$ using an STFT, as in $TF^{(i)} = \mathrm{STFT}(Z^{(i)})$.

2. Further, we extract the amplitude and the phase of the complex numbers in $TF^{(i)}$ into $A^{(i)} \in \mathbb{R}^{E \times M \times N}$ and $\Phi^{(i)} \in \mathbb{R}^{E \times M \times N}$, respectively.

3. Then, we perturb the amplitudes of the signal with Gaussian noise:

$$\tilde{A}^{(i)} = A^{(i)} + P \tag{3.5}$$

   where $\tilde{A}^{(i)} \in \mathbb{R}^{E \times M \times N}$ is the perturbed amplitude matrix, and $P \in \mathbb{R}^{E \times M \times N}$ is the perturbation drawn from a normal distribution $P \sim \mathcal{N}(\mu, \sigma^2)$.

4. Next, the perturbed complex time-frequency representation $\tilde{TF}^{(i)} \in \mathbb{C}^{E \times M \times N}$ is recovered by applying $\tilde{a}(\cos\phi + j \sin\phi)$ to each element of the matrix, where $\tilde{a}$ is the corresponding amplitude in $\tilde{A}^{(i)}$, ϕ is the corresponding phase in $\Phi^{(i)}$, and j is the imaginary unit.

5. Finally, a new artificial trial $\tilde{Z}^{(i)} \in \mathbb{R}^{E \times T}$ is generated using an ISTFT, as in $\tilde{Z}^{(i)} = \mathrm{ISTFT}(\tilde{TF}^{(i)})$.

4 Experiments and Results

This chapter details the experiments, the results, and the analyses involved in our investigation of the effects of data augmentation in motor imagery classification. It begins by presenting the overall experimental setup, including the model configuration, the data augmentation settings, and the evaluated test scenarios. The chapter then presents the results of our experiments, reporting the classification accuracy for each of the test scenarios. Additional results are reported in the supplementary material[5]. Moving forward, the chapter delves into the analysis of the results, which includes an examination the achieved accuracies, statistical tests, and an inspection of data distribution effects. Lastly, the chapter concludes by discussing the main findings, as well as acknowledging potential limitations of the study and highlighting opportunities for future research.

4.1 Experimental Setup

4.1.1 Overview

We evaluated the performance of the five data augmentation methods described in Chapter 3 on the BCIC-IV 2a and 2b datasets and compared them against a baseline (no data augmentation). We tested the effects of data augmentation on classification accuracy in two scenarios: within-session (using data from a single recording session at a time) and cross-session (using data from multiple sessions). In each scenario, we explored four settings to analyze the behavior of data augmentation with respect to input and output dimensionality: BCIC-IV 2a dataset with (i) 22 electrodes and 4 classes; (ii) 3 electrodes and 4 classes; (iii) 22 electrodes and 2 classes; and BCIC-IV 2b dataset with (iv) 3 electrodes and 2 classes. For setting (ii), we considered the C3, Cz, and C4 electrodes—the same used in the BCIC-IV 2b dataset. For the same reason, we adopted left and right hand motor imagery tasks in setting (iii). In addition, after preliminary experiments, we evaluated the combination of the two best-performing data augmentation methods to ver-

[5]https://github.com/gbrlfaria/bci-data-augmentation.

ify whether it could further improve classification performance. Specifically, we combine Sliding Window and Segmentation and Recombination into SW+SR.

We used blockwise stratified 4-fold cross-validation, as done in (LAWHERN et al., 2018). At each iteration, we divided the dataset into two training folds, one validation fold, and one test fold. Next, the model was trained using the two training folds and the best weights were selected according to the lowest loss on the validation fold. Finally, the accuracy of the selected model was evaluated on the test fold. During training, we used a batch size of 32, whereas the number of training epochs varied across scenarios.

We adopted the on-the-fly data augmentation strategy, which performs the following procedure before every epoch: (i) a portion of the original training samples is selected for augmentation, where each sample has a 50% chance of being selected; (ii) a data augmentation method is applied to the selected samples, generating new artificial samples of the same classes as the original ones; and (iii) the initially selected samples are replaced with the newly generated ones. The modified training dataset is then fed into the neural network. We chose this strategy as it can generate a large number of artificial samples while keeping a balance with the original data.

All experiments were performed on an Nvidia Tesla T4 GPU. The neural network was implemented[5] using Pytorch and the data were acquired and preprocessed using the MOABB (JAYARAM; BARACHANT, 2018), MNE (GRAMFORT et al., 2013), and Braindecode (SCHIRRMEISTER et al., 2017) libraries.

4.1.2 Test Scenarios

In the within-session scenario, we trained and tested the classifiers by applying cross-validation to each session of each subject separately. The main goal was to evaluate how much data augmentation can shorten the calibration time in a scenario where a subject is using a BCI for the first time. Thus, we simulated multiple situations using different quantities of original data to train the classifier.

For each cross-validation iteration on the BCIC-IV 2a dataset, we limited the amount of data in the training folds to 6, 12, 18, 24, 30, and 36 samples per class and the amount of data in the validation fold to 3, 6, 9, 12, 15, and 18 samples per class,

Table 4.1: Summary of the EEGNet configurations used.

Dataset	# Timesteps	# Electrodes	# Classes	# Parameters
BCIC-IV 2a	256 (2 s)	22	4	1,716
	256 (2 s)	3	4	1,412
	256 (2 s)	22	2	1,458
BCIC-IV 2b	256 (2 s)	3	2	1,154

respectively.

For the BCIC-IV 2b dataset, we limited the training data of each cross-validation iteration to 6, 12, 18, 24, and 30 samples per class and the validation data to 3, 6, 9, 12, and 15 samples per class, respectively. In this scenario, we set the number of epochs to 1000, as there were fewer batches per epoch.

In the cross-session scenario, we trained and tested the classifier by applying cross-validation to data from all sessions of each subject. Unlike in the previous scenario, we were primarily interested in evaluating how further data augmentation can improve classification performance when given larger quantities of data from multiple sessions. Here, we set the number of training epochs to 500, following the same experimental design adopted by (LAWHERN et al., 2018).

4.1.3 Data Preprocessing

For both datasets, we adopt the same procedure as in (LAWHERN et al., 2018). First, we resample the signal at 128 Hz and bandpass-filter it between 4 Hz and 40 Hz. Next, the resulting data are standardized using an electrode-wise exponential moving average. Finally, we crop the trial into a two-second window, from 0.5 s to 2.5 s after the cue onset.

4.1.4 Model Hyperparameters

We set the architecture hyperparameters of EEGNet to $F_1 = 8$, $F_2 = 16$, and $D = 2$, with a dropout probability of $p = 0.5$, which are the same used by the original configuration of EEGNet-8,2 (LAWHERN et al., 2018). We made this choice since our datasets and preprocessing are identical or similar to those used in (LAWHERN et al., 2018). Compared with the original architecture, we only change the number of input electrodes E and the

number of output classes N across our experiments, when necessary. Table 4.1 shows the number of parameters of each EEGNet configuration used in our experiments. As defined in (LAWHERN et al., 2018), we train the CNN with the categorical cross-entropy loss function, using the Adam optimizer with its default parameters (KINGMA; BA, 2017) and learning rate $\eta = 0.001$.

4.1.5 Data Augmentation Parameters

We now detail the parameters used for each data augmentation method.

Sliding Window

We set $t_{\min}$ and $t_{\max}$ so that the leftmost window begins at the cue onset while the rightmost window ends 4 s after the cue onset, which coincides with the end of the motor imagery period in most of the trials.

Segmentation and Recombination

We adopt the same setup used in (LOTTE, 2015). That is, we use a Hamming window of 250 ms with a 50% overlap for STFT and re-divide the time-frequency representation into eight segments before the recombination in Equation (3.2).

Empirical Mode Decomposition

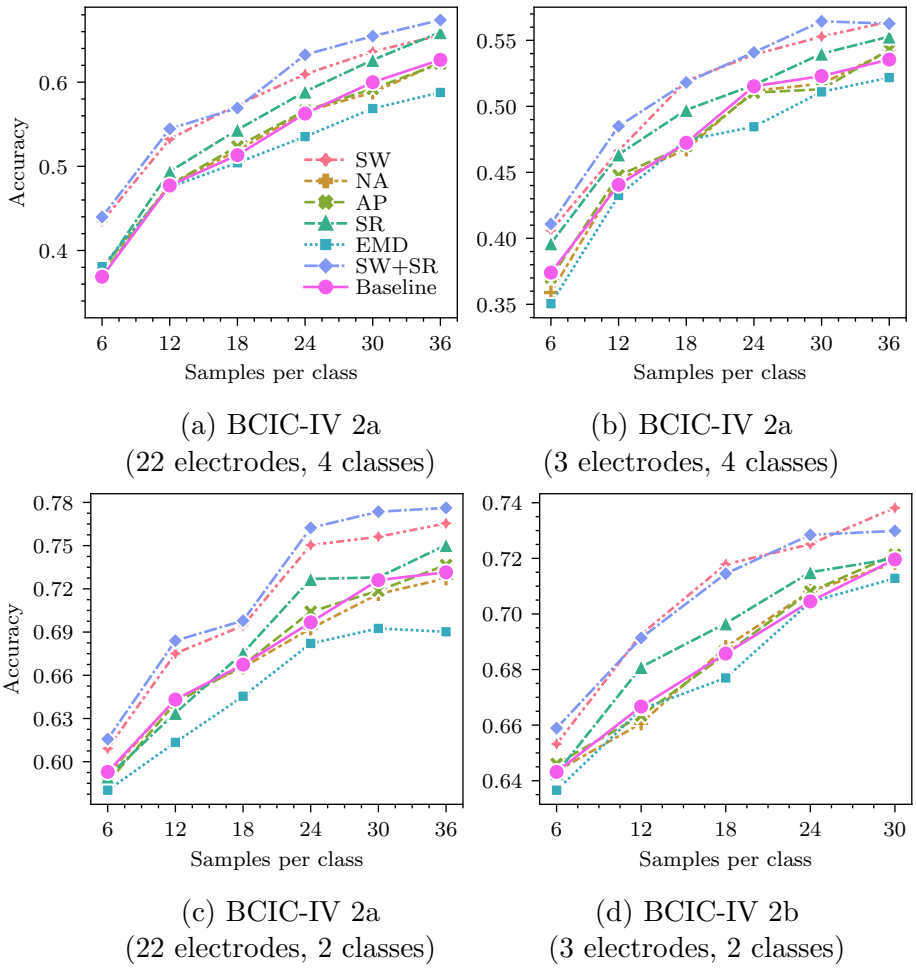
No trial exhibited more than 8 IMFs. There are no other parameters in this method.

Noise Addition

We use $\mu = 0$ and $\sigma = 0.1$, as those parameters led to the best results in (LASHGARI et al., 2021) when using the same datasets adopted in the present work.

Amplitude Perturbation

As in (LI et al., 2019), we used the Hanning window for STFT. However, as the other parameters of STFT were not specified in (LI et al., 2019), we applied a 250 ms window with a 50% overlap. Also, we did not use the noise parameters presented in (LI et al.,



(a) BCIC-IV 2a (22 electrodes, 4 classes)

(b) BCIC-IV 2a (3 electrodes, 4 classes)

(c) BCIC-IV 2a (22 electrodes, 2 classes)

(d) BCIC-IV 2b (3 electrodes, 2 classes)

Figure 4.1: Mean test accuracy for within-session motor imagery classification with respect to the number of original training samples per class, averaged over subjects, sessions, and folds. SW is Sliding Window, NA is Noise Addition, AP is Amplitude Perturbation, SR is Segmentation and Recombination, EMD is Empirical Mode Decomposition, and Baseline is the reference experiment without data augmentation.

2019) as the standard deviation of the noise was smaller than 1% of the standard deviation of the amplitudes observed in our data, leading to unnoticeable effects. Instead, similarly to Noise Addition, we use $\mu = 0$ and $\sigma = 0.015$, which is about 10% of the standard deviation of the amplitudes.

4.2 Results

4.2.1 Classification Accuracy

Figure 4.1 shows the mean classification accuracy with respect to the number of original training samples in the within-session scenario. The proposed combination SW+SR per-

Table 4.2: Mean test accuracies (± standard deviation) for cross-session motor imagery classification.

Dataset	BCIC-IV 2a			BCIC-IV 2b
Method	(22 electrodes, 4 classes)	(3 electrodes, 4 classes)	(22 electrodes, 2 classes)	(3 electrodes, 2 classes)
Baseline	0.696 ± 0.146	0.566 ± 0.113	0.798 ± 0.153	0.762 ± 0.142
SW	**0.708 ± 0.148**	**0.574 ± 0.117**	0.797 ± 0.153	**0.764 ± 0.142**
NA	0.688 ± 0.150	0.562 ± 0.116	0.791 ± 0.160	0.761 ± 0.144
AP	0.690 ± 0.149	0.565 ± 0.118	0.794 ± 0.153	0.760 ± 0.145
SR	0.705 ± 0.152	0.573 ± 0.119	**0.808 ± 0.153**	0.749 ± 0.152
EMD	0.640 ± 0.159	0.548 ± 0.130	0.749 ± 0.163	0.740 ± 0.153
SW+SR	0.706 ± 0.150	0.570 ± 0.115	0.801 ± 0.168	0.749 ± 0.149

formed better than every other method in all cases except for the BCIC-IV 2b dataset, in which Sliding Window took the lead. Nonetheless, SW+SR and Sliding Window both outperformed the baseline. Meanwhile, Segmentation and Recombination improved the mean accuracy in 18 out of 23 cases, whereas Noise Addition and Amplitude Perturbation did not lead to noticeable improvements and decreased accuracy in some cases. Finally, EMD performed poorly and often reduced accuracy.

Regarding the amount of original data, the Sliding Window and SW+SR methods achieved higher classification accuracy with 33% fewer data in almost one-third of the experiments and at least 25% fewer data in almost half of them. Particularly, SW+SR increased the mean accuracy from 69.7% to 76.2% on the BCIC-IV 2a dataset (2 classes and 22 electrodes), surpassing the top baseline result while using 24 fewer training trials and 12 fewer validation trials. Similar results can be observed for Sliding Window. We highlight that the classifier reached a mean accuracy of 82.64% (the baseline reached 54.51%) with 24 training samples per class on subject 7, and that it reached a mean accuracy of 78.82% (the baseline reached 63.54%) with 6 training samples per class on subject 3, both using SW+SR on the BCIC-IV 2a dataset with 22 electrodes and two classes.

Table 4.2 shows the mean test accuracies and the standard deviations for each augmentation method in the cross-session scenario. Sliding Window, the best method overall, failed to increase classification accuracy on the BCIC-IV 2a dataset with 22 electrodes and 2 classes. Similarly, Segmentation and Recombination and SW+SR failed on the BCIC-IV 2b dataset. Moreover, SW+SR did not obtain the highest accuracy in any case in this scenario. Meanwhile, Noise Addition, Amplitude Perturbation, and EMD

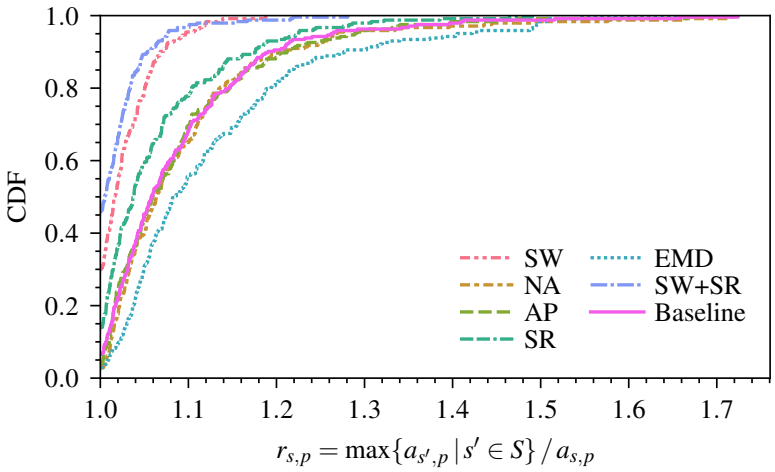


Figure 4.2: Performance profiles of the mean test accuracy $a_{s,p}$ (over sessions and folds) over every test-problem p.

reduced accuracy in all cases, where EMD was the worst-performing method.

4.3 Analysis

4.3.1 Performance Profiles

We further inspect the results using performance profiles (DOLAN; MORÉ, 2002), which provide a single overview of the performance of the methods across all test scenarios. Here, we interpret each tuple (dataset, number of electrodes, number of classes, subject, number of training samples) as a test-problem $p \in \mathcal{P}$ and each classifier (corresponding to each augmentation method, in addition to the baseline) as a solver $s \in \mathcal{S}$. Then, we define the performance $a_{s,p}$ as the mean accuracy of the solver s on the test-problem p. Essentially, each performance profile is the cumulative distribution function (CDF) of the ratio $r_{s,p} = \frac{\max\{a_{s',p} \mid s' \in \mathcal{S}\}}{a_{s,p}}$ for every solver s and every test-problem p, where $r_{s,p}$ is the ratio between the best accuracy for p and the accuracy of the solver s on p. This gives us: (i) the approach that obtained the best result the most times (largest $P(r_{s,p} \leq 1)$), (ii) the most reliable approach (smallest τ such that $P(r_{s,p} \leq \tau) = 1$), and (iii) the approach with the best overall performance (largest area under the curve).

The performance profiles are shown in Figure 4.2. The best-performing method overall is SW+SR, followed by Sliding Window. SW+SR is also most frequently the best, reaching top accuracy in 45.7% of the test-problems. Additionally, the baseline had the

Table 4.3: Results (p-values) of the Dunn's test.

	Baseline	SW	NA	AP	SR	EMD
SW	$\mathbf{10^{-20}}$					
NA	10^{0}	$\mathbf{10^{-24}}$				
AP	10^{0}	$\mathbf{10^{-20}}$	10^{0}			
SR	$\mathbf{10^{-3}}$	$\mathbf{10^{-7}}$	$\mathbf{10^{-5}}$	$\mathbf{10^{-3}}$		
EMD	$\mathbf{10^{-3}}$	$\mathbf{10^{-39}}$	10^{-2}	$\mathbf{10^{-3}}$	$\mathbf{10^{-13}}$	
SW+SR	$\mathbf{10^{-32}}$	10^{-1}	$\mathbf{10^{-37}}$	$\mathbf{10^{-33}}$	$\mathbf{10^{-16}}$	$\mathbf{10^{-56}}$

* Statistically significant values ($p < 0.05$) are boldfaced.

best accuracy in 6.6% of the cases. Moreover, the most reliable method is Sliding Window (CDF(1.19) = 1), followed by SW+SR (CDF(1.28) = 1).

4.3.2 Statistical Significance

We analyzed the statistical significance of the results shown in the performance profiles. First, we applied the Kruskal-Wallis one-way analysis of variance to the $r_{s,p}$ values and rejected the null hypothesis at a significance level of $\alpha = 0.05$ ($p = 1.39 \times 10^{-86}$). Then, we used Dunn's test to calculate pairwise p-values between different augmentation approaches. We adjusted the obtained p-values using the Bonferroni correction method. The results are summarized in Table 4.3. We can see that the performance profile curves of all augmentation methods differ significantly from the baseline, except for Noise Addition and Amplitude Perturbation. However, the difference between the Sliding Window and SW+SR curves, despite visible, is not statistically significant. Still, the results indicate that Sliding Window and SW+SR are significantly better than the other methods.

4.3.3 Data Distribution

We now analyze how the distribution of the data changes as we apply data augmentation. To that end, we first visualize the covariance matrices of the EEG signals with t-SNE (MAATEN; HINTON, 2008) using the Riemannian distance, similarly to (ZANINI et al., 2018). Essentially, this visualization technique projects the data into a low-dimensional representation such that similar samples are close together while dissimilar samples are far apart.

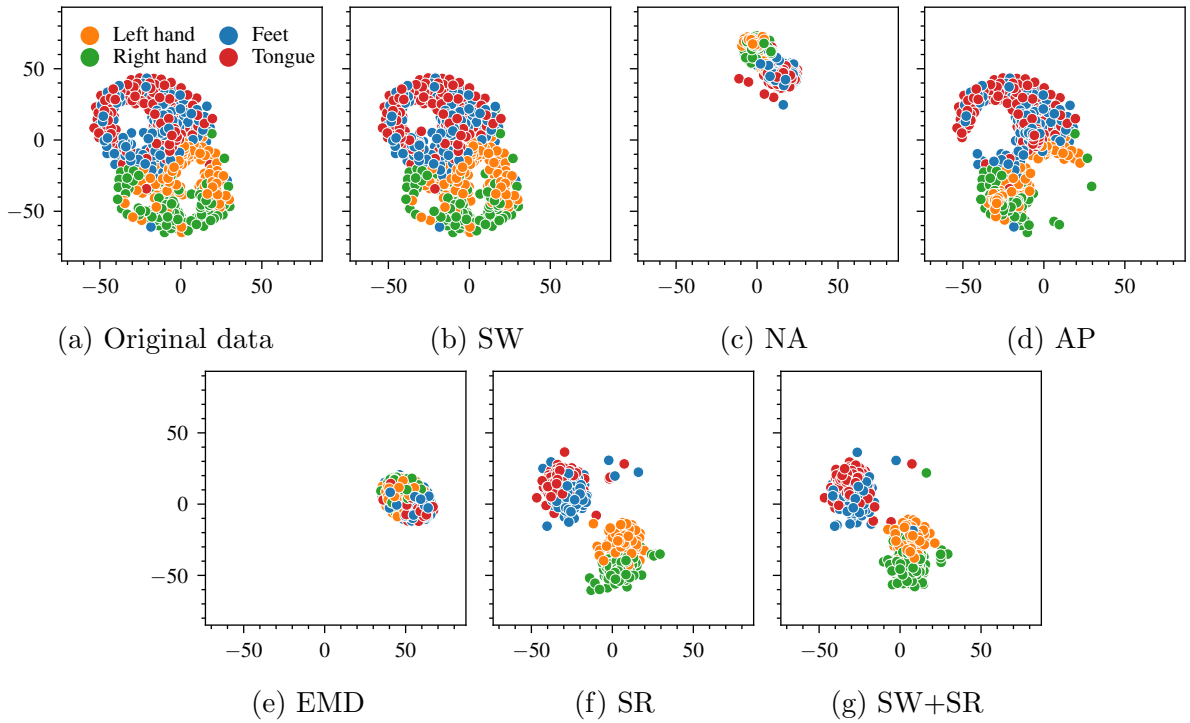
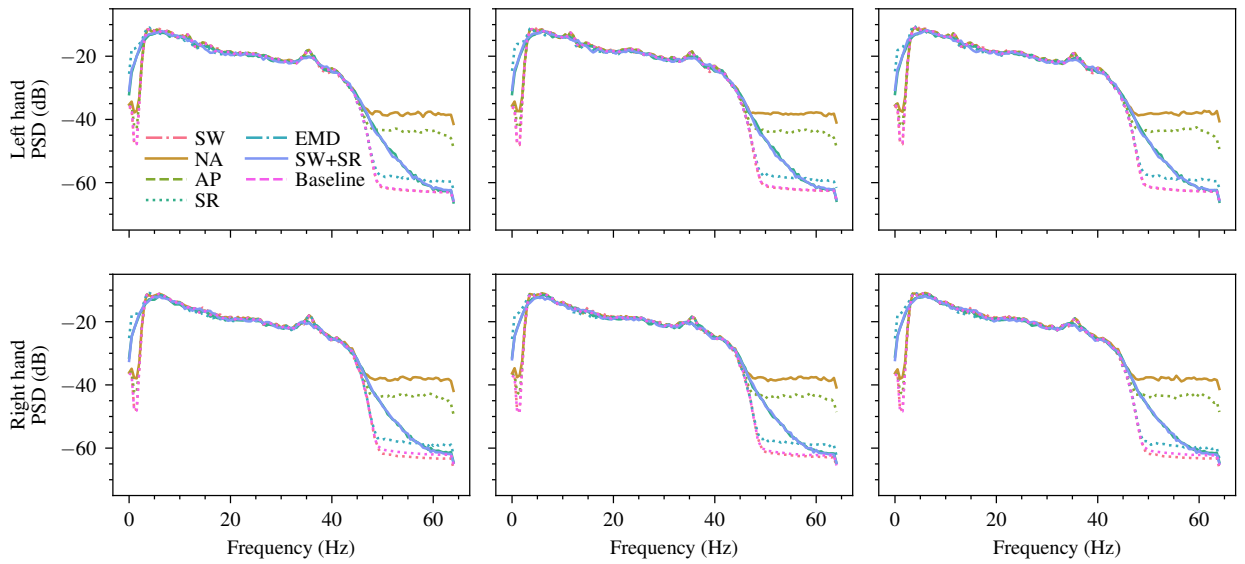


Figure 4.3: Two-dimensional t-SNE projections of the covariance matrices of the signals from subject 1 of dataset BCIC-IV 2a.

Figure 4.3 shows the t-SNE projections of the data generated by each data augmentation method for subject 1 of dataset BCIC-IV 2a, in addition to the baseline. We can see that the Sliding Window data distribution is similar to the original one. Also, the data generated by Noise Addition and EMD data are visibly different from the baseline, as both are collapsed and centered away from the original distribution. Meanwhile, the Amplitude Perturbation data are projected close to the original data, but have a visibly different clustering. Segmentation and Recombination and SW+SR data points are both clustered within the region occupied by the original data.

Similar patterns can be observed for other subjects in dataset BCIC-IV 2a and non-preprocessed signals. However, when using only the C3, Cz, and C4 electrodes, which is the case for dataset BCIC-IV 2b, the pattern observed for SW+SR and Segmentation and Recombination is not visible. Similarly, the centroid shift observed for Noise Addition is not visible, although EMD is still visibly different from the baseline. These additional visualizations are available in the supplementary material.

Additionally, in order to visualize the frequency distribution of the artificial signals, we calculate the the mean power spectral density (PSD) of the EEG signals from



(a) C3 (b) Cz (c) C4

Figure 4.4: Mean power spectral density of the C3, Cz, and C4 electrodes for left and right-hand motor imagery signals from subject 1 of the BCIC-IV 2a dataset.

subject 1 of the BCIC-IV 2a dataset using Welch's method, as shown in Figure 4.4. We can see that Sliding Window has the closest distribution to the baseline. All other methods deviate from the baseline over high frequencies ($>$ 40 Hz) and low frequencies ($<$ 2 Hz). Noise Addition and Amplitude Perturbation have the highest deviations over high frequencies and the lowest deviations over low frequencies, apart from Sliding Window. Meanwhile, the remaining methods follow the baseline more closely over high frequencies, but have a relatively higher deviation over low frequencies. Interestingly, EMD has a frequency distribution similar to the baseline over high frequencies. Again, similar patterns are observed for other subjects and datasets.

4.4 Discussion

The results indicate that data augmentation can increase accuracy and reduce the amount of data needed for calibration, with the top-performing methods achieving greater accuracy with fewer training data compared with the baseline in the within-session scenario. Specifically, the performance profiles analysis suggests that SW+SR and Sliding Window are the best methods overall, where the latter is the most reliable.

According to the data distribution analyisis, Sliding Window is the data augmen-

tation method that generates the data that most closely resembles the original data. As expected, this is because this method generates samples by simply sliding a window over the original signals without modifying them. Because of this, Sliding Window not only is reliable, but also makes better use of the available training data. The downside is that this method only yields valid motor imagery signals at all times when the motor imagery period of each trial is larger than the input window.

On the other hand, the Segmentation and Recombination method yielded relatively modest results and generated samples that moderately deviate from the original data. Yet, this method still outperforms the baseline and can be applied regardless of the input window size. When combined with Sliding Window, it further enhances performance, leading to the best overall results on average.

Meanwhile, Noise Addition and Amplitude Perturbation performed similarly to the baseline and, in fact, reduced accuracy in several cases. Notably, both methods introduce high-frequency noise which is not present in the original filtered signals. At the same time, EMD performed poorly in all experiments. Although the reasons for this result are unclear, it is worth noting that the data generated by EMD were visibly different from the original data according to the data distribution analysis.

In the cross-session scenario, however, data augmentation was shown to be less effective. This could be attributed to diminishing returns resulting from the larger amount of original training data. Additionally, the inclusion of data from multiple sessions might have adversely impacted combination-based methods due to signal variations over time, which could explain the results observed on the BCIC-IV 2b dataset.

Overall, the results demonstrate that data augmentation can reduce the need for training data and increase accuracy in few-parameter CNN-based motor imagery BCIs, mainly when few original data points are available. This is relevant for motor imagery BCI applications in general, but especially for those such as neurorehabilitation, in which decreasing calibration times and increasing feedback accuracy directly lead to better treatment quality for patients.

However, there are limitations to the present study and further investigation is necessary in to understand and explain the impact of data augmentation on motor

imagery EEG classification. Future research should delve deeper into the qualitative differences between artificial and real motor imagery EEG signals. Furthermore, it is worth exploring alternative data augmentation methods, including variations of the ones already analyzed in this work, as well as investigating different datasets and neural network architectures. One notable candidate for further exploration are modern sequence processing architectures such as deep state-space models (GU et al., 2021; GU; GOEL; RÉ, 2022; SMITH; WARRINGTON; LINDERMAN, 2023). Another worthwhile avenue is to assess how data augmentation compares to and complements other techniques for reducing calibration times in neural network-based BCIs, such as transfer learning and unsupervised learning.

5 Conclusion

In this work, we have addressed the problem of lengthy calibration times in motor imagery-based brain-computer interfaces, a key challenge in many different applications. To this end, we investigated the extent to which data augmentation techniques can reduce calibration times while increasing decoding accuracy. Specifically, we analyzed five data augmentation methods previously reported in the literature in multiple scenarios, utilizing a few-parameter CNN on two distinct motor imagery datasets. In our analyses, we showed that data augmentation can significantly improve accuracy and reduce the amount of data needed for training, although the improvements were modest in the cross-session scenario, in which more data were available. Moreover, we combined the two best-performing data augmentation methods, Sliding Window and Segmentation and Recombination, into the SW+SR method. The results obtained showed that SW+SR yielded the best results overall, whereas Sliding Window was the most reliable method—and that both methods led to significantly larger classification accuracies in most experiments compared to other approaches. Future work may qualitatively investigate the impact of data augmentation on motor imagery EEG classification as well as explore alternative augmentation methods, datasets, and neural network architectures

Bibliography

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006.

CARLSON, T.; MILLAN, J. del R. Brain-controlled wheelchairs: A robotic architecture. *IEEE Robot. Autom. Mag.*, v. 20, n. 1, p. 65–73, 2013.

CRAIK, A.; HE, Y.; CONTRERAS-VIDAL, J. L. Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of Neural Engineering*, IOP Publishing, v. 16, n. 3, p. 031001, apr 2019.

DINARÈS-FERRAN, J.; ORTNER, R.; GUGER, C.; SOLÈ-CASALS, J. A new method to generate artificial frames using the empirical mode decomposition for an eeg-based motor imagery bci. *Frontiers in Neuroscience*, v. 12, 2018.

DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Math. Program.*, v. 91, p. 201–213, 2002.

DONOGHUE, J. P. Connecting cortex to machines: recent advances in brain interfaces. *Nature Neuroscience*, v. 5, n. 11, p. 1085–1088, 2002. ISSN 1546-1726.

FARIA, G.; SOUZA, G. H. de; BERNARDINO, H.; MOTTA, L.; VIEIRA, A. Analyzing data augmentation methods for convolutional neural network-based brain-computer interfaces. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2022. p. 1–8.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. ed. [S.l.]: O'Reilly Media, Inc., 2019.

GRAIMANN, B.; ALLISON, B.; PFURTSCHELLER, G. Brain–computer interfaces: A gentle introduction. In: _____. *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. ISBN 978-3-642-02091-9.

GRAMFORT, A.; LUESSI, M.; LARSON, E.; ENGEMANN, D. A.; STROHMEIER, D.; BRODBECK, C.; GOJ, R.; JAS, M.; BROOKS, T.; PARKKONEN, L.; HÄMÄLÄINEN, M. S. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, v. 7, n. 267, p. 1–13, 2013.

GU, A.; GOEL, K.; RÉ, C. Efficiently modeling long sequences with structured state spaces. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2022.

GU, A.; JOHNSON, I.; GOEL, K.; SAAB, K. K.; DAO, T.; RUDRA, A.; RÉ, C. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In: BEYGELZIMER, A.; DAUPHIN, Y.; LIANG, P.; VAUGHAN, J. W. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2021.

HAYKIN, S. S. *Neural networks and learning machines*. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HE, C.; LIU, J.; ZHU, Y.; DU, W. Data augmentation for deep neural networks model in eeg classification task: A review. *Frontiers in Human Neuroscience*, v. 15, 2021. ISSN 1662-5161. Available at: ⟨https://www.frontiersin.org/articles/10.3389/fnhum.2021.765525⟩.

HUANG, N. E.; SHEN, Z.; LONG, S. R.; WU, M. C.; SHIH, H. H.; ZHENG, Q.; YEN, N.-C.; TUNG, C. C.; LIU, H. H. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proc. of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, v. 454, n. 1971, p. 903–995, 1998.

JAYARAM, V.; BARACHANT, A. MOABB: trustworthy algorithm benchmarking for BCIs. *J. Neural Eng.*, IOP Publishing, v. 15, n. 6, p. 066011, sep 2018.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. arXiv, 2014. Available at: ⟨https://arxiv.org/abs/1412.6980⟩.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Available at: ⟨https://arxiv.org/abs/1412.6980⟩.

KOCHENDERFER, M. J.; WHEELER, T. A. *Algorithms for Optimization*. Cambridge, Massachusetts: The MIT Press, 2019.

LASHGARI, E.; LIANG, D.; MAOZ, U. Data augmentation for deep-learning-based electroencephalography. *Journal of Neuroscience Methods*, v. 346, p. 108885, 2020. ISSN 0165-0270.

LASHGARI, E.; OTT, J.; CONNELLY, A.; BALDI, P.; MAOZ, U. An end-to-end CNN with attentional mechanism applied to raw EEG in a BCI classification task. *J. Neural Eng.*, IOP Publishing, v. 18, n. 4, p. 0460e3, aug 2021.

LAWHERN, V. J.; SOLON, A. J.; WAYTOWICH, N. R.; GORDON, S. M.; HUNG, C. P.; LANCE, B. J. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, IOP Publishing, v. 15, n. 5, p. 056013, jul 2018.

LEVINE, S. P.; HUGGINS, J. E.; BEMENT, S. L.; KUSHWAHA, R. K.; SCHUH, L. A.; PASSARO, E. A.; ROHDE, M. M.; ROSS, D. A. Identification of electrocorticogram patterns as the basis for a direct brain interface. *Journal of Clinical Neurophysiology*, v. 16, n. 5, p. 439–447, 1999. ISSN 0736-0258.

LI, Y.; ZHANG, X.-R.; ZHANG, B.; LEI, M.-Y.; CUI, W.-G.; GUO, Y.-Z. A channel-projection mixed-scale convolutional neural network for motor imagery eeg decoding. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 27, n. 6, p. 1170–1180, 2019.

LOTTE, F. Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain–computer interfaces. *Proceedings of the IEEE*, v. 103, n. 6, p. 871–890, 2015.

MAATEN, L. van der; HINTON, G. E. Visualizing data using t-sne. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008.

MANE, R.; CHOUHAN, T.; GUAN, C. BCI for stroke rehabilitation: motor and beyond. *J. Neural Eng.*, IOP Publishing, v. 17, n. 4, p. 041001, aug 2020.

MITCHELL, T. M. *The Need for Biases in Learning Generalizations*. New Brunswick, NJ, 1980.

MURPHY, K. P. *Probabilistic Machine Learning: An introduction*. [S.l.]: MIT Press, 2022.

OXLEY, B. C. *International 10-20 system for EEG-MCN*. 2021. Retrieved from ⟨https://commons.wikimedia.org/wiki/File:International_10-20_system_for_EEG-MCN.svg⟩. Licensed under Creative Commons CC0 1.0 Universal Public Domain Dedication.

PAWAR, D.; DHAGE, S. Feature extraction methods for electroencephalography based brain-computer interface: A review. *IAENG International Journal of Computer Science*, v. 47, n. 3, 2020.

RAMADAN, R. A.; VASILAKOS, A. V. Brain computer interface: control signals review. *Neurocomputing*, v. 223, p. 26–44, 2017. ISSN 0925-2312.

SCHIRRMEISTER, R. T.; SPRINGENBERG, J. T.; FIEDERER, L. D. J.; GLASSTETTER, M.; EGGENSPERGER, K.; TANGERMANN, M.; HUTTER, F.; BURGARD, W.; BALL, T. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, v. 38, n. 11, p. 5391–5420, 2017.

SINGH, A.; HUSSAIN, A.; LAL, S.; GUESGEN, H. W. A comprehensive review on critical issues and possible solutions of motor imagery based electroencephalography brain-computer interface. *Sensors (Basel, Switzerland)*, v. 21, 2021.

SMITH, J. T.; WARRINGTON, A.; LINDERMAN, S. Simplified state space layers for sequence modeling. In: *The Eleventh International Conference on Learning Representations*. [S.l.: s.n.], 2023.

STAFFA, M.; GIORDANO, M.; FICUCIELLO, F. A wisard network approach for a bci-based robotic prosthetic control. *International Journal of Social Robotics*, v. 12, p. 749–764, 2020.

TANGERMANN, M.; MÜLLER, K.-R.; AERTSEN, A.; BIRBAUMER, N.; BRAUN, C.; BRUNNER, C.; LEEB, R.; MEHRING, C.; MILLER, K. J.; MÜLLER-PUTZ, G. R.; NOLTE, G.; PFURTSCHELLER, G.; PREISSL, H.; SCHALK, G.; SCHLÖGL, A.; VIDAURRE, C.; WALDERT, S.; BLANKERTZ, B. Review of the bci competition iv. *Frontiers in Neuroscience*, v. 6, 2012.

VAPNIK, V. N. *Statistical Learning Theory*. New York: Wiley, 1998.

WANG, X.; HERSCHE, M.; TöMEKCE, B.; KAYA, B.; MAGNO, M.; BENINI, L. An accurate eegnet-based motor-imagery brain-computer interface for low-power edge computing. 2020.

WIERZGAŁA, P.; ZAPAŁA, D.; WOJCIK, G. M.; MASIAK, J. Most popular signal processing methods in motor-imagery bci: A review and meta-analysis. *Frontiers in Neuroinformatics*, Frontiers, v. 12, p. 78, 2018. ISSN 1662-5196.

WOLPAW, J. R.; BIRBAUMER, N.; MCFARLAND, D. J.; PFURTSCHELLER, G.; VAUGHAN, T. M. Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, v. 113, n. 6, p. 767–791, 2002. ISSN 1388-2457. Available at: ⟨https://www.sciencedirect.com/science/article/pii/S1388245702000573⟩.

ZANINI, P.; CONGEDO, M.; JUTTEN, C.; SAID, S.; BERTHOUMIEU, Y. Transfer learning: A riemannian geometry framework with applications to brain–computer interfaces. *IEEE Transactions on Biomedical Engineering*, v. 65, p. 1107–1116, 2018.