UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

A comparative analysis of machine learning algorithms for EEG-BCI applications

Permalink

https://escholarship.org/uc/item/9fw659zq

Author

Ibrahim, Mina

Publication Date

2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

A comparative analysis of machine learning algorithms for EEG-BCI applications

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Biomedical Engineering

by

Mina Ibrahim

Thesis Committee:
Professor Zoran Nenadic, Chair
Professor Frithjof Kruggel
Associate Professor Beth Lopour

2023

© 2023 Mina Ibrahim

DEDICATION

This work is dedicated to my parents whose unconditional love and support throughout the highs and the lows were essential in helping me achieve this accomplishment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This material is based upon work partially supported by the National Science Foundation under Grant No. 1646275.

I would like to thank and acknowledge,

Dr. Nenadic for his valuable advice and and discussions that helped shape and guide this research,

My labmates who helped by proposing new ideas, correcting my errors, participating in experiments and providing snippets of code to help conduct the research,

Dr. Lopour and Dr. Kruggel for taking the time to review and assess this work.

ABSTRACT OF THE THESIS

A comparative analysis of machine learning algorithms for EEG-BCI applications

By

Mina Ibrahim

Master of Science in Biomedical Engineering

University of California, Irvine, 2023

Professor Zoran Nenadic, Chair

Brain-computer interfaces (BCIs) are a means of controlling devices in our environment using brain signals. They are promising tools for persons with neurological disabilities including speech and muscular impediments that externally connect the brain and affected muscle groups using computational devices. There are several brain imaging modalities categorized as invasive or non-invasive with electroencephalography (EEG) being the most popular due to its practicality and affordability. A BCI pipeline consists of data acquisition, preprocessing, feature extraction, classification followed by an action by a device. There are multiple machine learning (ML) algorithms that can be used for classification however there are no guidelines for selecting an algorithm for a certain control signal or BCI application.

In this thesis, we compare five common ML algorithms, namely Logistic regression (LR), neural networks (NNs), support vector machines (SVMs), k-nearest neighbor (KNN) and a Bayesian classifier for two common control signals used in EEG-BCIs, the P300 potential and sensorimotor rhythms. For P300, the Bayesian decoder and a fully connected NN performed the best for cross-validations performed on each session's data. SVM performed the best for leave-one-subject-out cross-validations where the training data didn't include any of the test subject's data. SVM and LR performed the best for sensorimotor data for cross-validations on each session. A procedure mimicking real-time decoding performed on the

sensorimotor data didn't differentiate between the algorithms in terms of accuracy however a recurrent neural network based on long short term memory units had the lowest lags defined as the temporal offset between the predictions and true labels. These results provide a good foundation for the selection of ML algorithms for BCI applications. Future works can build on this by incorporating more ML algorithms and testing on additional BCI control signals.

Chapter 1

Introduction

Brain-computer interfaces (BCIs) are a means of using brain activity to control computers or external devices [2, 3]. BCIs have been used for medical or non-medical purposes [4]. The medical applications include communication, motor function and neurorehabilitation. For example, individuals with Amyotropic Lateral Sclerosis (ALS) or locked-in-syndrome (LIS) can communicate by using BCIs that decode intended letters or characters [5]. People with central nervous system diseases such as spinal chord injury could bypass the injury by using a BCI to decode motor intentions and in turn control a motor assistive device such as a robotic arm/exoskeleton [6]. Through repetition, BCIs can be used to strengthen neuronal pathways to help recover affected brain functions such as movement or speech in individuals with brain diseases such as stroke [7]. Non-medical applications of BCIs can be used for entertainment and feedback on alertness or drowsiness. Entertainment applications include using BCIs to control computer games [8] or avatars [9]. BCIs can also be used to monitor individuals' alertness or drowsiness during work [10] or while driving [11] for safety reasons and accident prevention.

BCIs can be classified as invasive or non-invasive depending on the method or modality

used to collect the neuronal activity. The main invasive modalities are electrocorticography (ECoG) and multielectrode arrays (MEA) [12]. The main advantage of invasive modalities is their close or direct proximity to the brain allowing them to acquire high fidelity neuronal signals. ECoG grids contain electrodes that can detect electrical signals from the brain and can either be placed on the dura or in direct contact to the brain surface. One of the first ECoG based BCIs allowed users to control a two-dimensional cursor [13]. MEA are made up of needles that penetrate the brain's surface allowing them to collect intracortical signals. One of the first uses of MEA in humans allowed the control of a two-dimensional cursor, a prosthetic hand and a robotic arm [14]. Invasive modalities tend to have high signal-to-noise ratios (SNRs) and high spatiotemporal resolutions when compared to non-invasive modalities however they require surgery. This may lead to side effects or complications including foreign body rejection, scar tissue, inflammation and gliosis [15, 16].

There are numerous non-invasive modalities that utilize metabolic, magnetic or electrical properties of the brain for their recordings [3]. They include functional magnetic resonance imaging (fMRI), magnetoencephalography (MEG), near-infrared spectroscopy (NIRS) and electroencephalography (EEG). fMRI measures changes in cerebral blood volume, flow and oxygenation levels in response to neuronal activity using electromagnetic fields [17]. Their main advantage is their high spatial resolution enabling localizing brain activity sources [18]. However their slow temporal resolution of 1-2 seconds in addition to the delayed metabolic response to neuronal activity of 3-6 seconds [19] has limited their use for real-time BCIs. Subjects were able to alternate between activating two brain regions while getting feedback of their activity on a graph using a fMRI-based BCI measuring blood oxygenation levels [20]. MEG measures magnetic fields produced by by neuronal electrical activity [21]. One of their main advantages is that the magnetic signals are less distorted by the skull than electrical signals [22]. The first MEG-based BCI used motor imagery (MI) to spell words [23]. NIRS measures cortical metabolic activity using infrared light. A light source emits rays through the skull that can be used to measure changes in oxy and

deoxy-hemoglobin concentrations. This is possible as both hemoglobin species have different absorption spectra and therefore their concentrations can be measured by monitoring light attenuation at different wavelengths using a detector. NIRS is a popular imaging choice for BCIs due to its portability and affordability however it has a low spatial resolution [24] and similar to fMRI measures a delayed metabolic response to neuronal activity [25]. One of the first applications of NIRS-based BCIs was using motor imagery to select targets on a screen [25]. EEG measures the potential difference at the scalp arising from synaptic electrical activity of millions of cortical neurons [26]. It has good temporal resolution on the order of milliseconds but a low spatial resolution due to a low number of electrodes as well as the dispersion effect of intermediate layers including skin, skull and meninges [27]. In 1988, Farwell and Donchin presented an EEG-based BCI allowing individuals to control an onscreen speller [5]. Owing to its high temporal resolution, comparative affordability and portability, EEG is the most common imaging tool used for BCIs.

A BCI consists of multiple steps or stages between the user's intentions and the control of the external device. This includes signal acquisition, preprocessing, feature extraction/selection, classification and a control interface [28]. This is typically followed by feedback to the user forming a loop and allowing the user to modify their behavior in order to better control the BCI. Each stage has complications and challenges that need to be addressed for the whole system to operate. One of the main hurdles is finding a suitable machine learning (ML) algorithm to classify the signals. There are numerous ML algorithms that have been used and others that are being explored or have yet to be used for BCIs. Some of the well known ones include linear discriminant analysis (LDA), support vector machines (SVM), k-nearest neighbor (KNN), Bayesian classifiers and neural networks (NN). They all have advantages and disadvantages but it can often be difficult to know which one to use for a particular BCI control signal or application without trying out multiple algorithms which can be very time consuming. Most research articles only address one or a couple of algorithms and are usually targeted towards one specific application. Some studies have compared multiple ML

algorithms for singular applications including gait decoding [29], motor imagery (MI) [30, 31] and active brain state [32]. Other studies have assessed the performance of a single type of ML algorithm on multiple applications. For example, a study tested NNs on motor, speech imagery and error processing tasks [33]. A lot of these studies compare different subsets of ML algorithms for different BCI tasks. This makes it difficult to find trends and directly compare ML algorithms across different tasks and applications. Therefore, my objective in this thesis is to directly compare several ML algorithms for two very common EEG control signal types used for BCIs, P300 and sensorimotor rhythms, to offer a guide in the selection process of a ML algorithm for a particular BCI application. Although this thesis will not test all known ML algorithms and control signals used for BCIs, it can be used as a good starting point for future works to build up on. Additionally, the knowledge may transfer over to different control signals or applications and offer insights to similar or adjacent applications.

Chapter 2

Background

The following sections detail previous research and background with regards to EEG-based BCIs.

2.1 Control signals

Controlling a BCI requires a mental strategy or action also known as a control signal. Control signals can be evoked or non-evoked meaning they can be elicited by an external stimulus or via a conscious change in mental state respectively [34]. Two of the most common evoked potentials are visual evoked potentials (VEPs) and P300 potentials. VEPs occur when a visual stimulus causes an observable effect on an individual's brain signals [35]. The stimulus can range from checkered patterns or images to flickering lights. The most famous VEP is the steady state visual evoked potential (SSVEP) which occurs in response to a light source flickering at a constant frequency above 6 Hz [36]. SSVEPs result in neurons firing at the same rate of the light source which can be observed as a sinusoidal signal on EEG. There are different SSVEP stimuli patterns or strategies however the most common one consists of

a graphical user interface with buttons/symbols/characters flickering at different frequencies where the user can focus on the button they wish to select which can be detected and controlled by a BCI. SSVEPs can be used for communication to control remotes, navigation panels or spelling. They can also be used for controlling robotic orthoses or wheelchairs for persons with motor disabilities. The advantages of using SSVEPs as control signals is that they offer high information transfer rates (ITRs, 30-60 bits/min) and can be used with relatively little to no training [36]. However they require users to stare at flashing lights for long periods of time which can cause fatigue [37].

P300 potentials occur due to an unexpected stimulus resulting in an increase in measured potential around 300 ms following the stimulus [5]. They are usually presented using the oddball paradigm where the unexpected stimulus is presented infrequently among frequent occurrences of non-target stimuli [38]. One of the well known applications using P300 potentials is the P300-speller [5] where letters are presented onscreen in a grid with sets of characters, typically the rows and columns, being flashed in a random or ordered fashion. The user focuses on the letter they wish to select and a P300 potential results when the letter is flashed. There are studies that average the potentials across multiple flashes to increase their chances of detecting a P300 potential while others have attempted to decode potentials from single flashes. Applications using P300 potentials have moderate ITRs (20-25 bits/min) [3] and can also be used with little to no user training. Similar to SSVEPs, they also suffer from prolonged gazing at screens which can cause user fatigue.

Sensorimotor rhythms are non-evoked signals that arise near the motor cortex during overt or imagined movement. They involve oscillations in the mu and beta frequency bands and are accompanied with increases in measured potential known as event related synchronization (ERS) or decreases in potential, known as event related desynchronization (ERD) [39]. Studies have shown that imagined movement or motor imagery (MI) activate the same brain regions during actual movement [40] making it possible for individuals with motor disabil-

ities to use sensorimotor rhythms for BCI control. They have been used by individuals with paraplegia or tetraplegia to control robotic arms or exoskeletons to assist with motor functions. Sensorimotor rhythms do not require external stimuli making them useful for real-world applications. However they typically result in low ITRs (3-35 bits/min) [3, 34] with limited degrees of freedom and require lengthy training periods of weeks or months for user control.

2.2 Applications

The applications of BCIs are numerous and include communication, locomotion, neurorehabilitation, environmental control and entertainment which can be used by individuals with disabilities or healthy users. There are non-BCI based methods to aid with communication or locomotion including eye-tracking devices [41] or devices that use residual muscle control [42] which can have higher ITRs than BCIs. However they become increasingly difficult to use as the severity of ALS or level of LIS increases. In those cases where the subject's motor control is severely affected, BCIs may be the only alternative to restore those functions without assistance from others. One of the main hurdles faced by individuals with LIS is their loss of communication abilities. There are a multitude of BCIs that have been used for communication restoration purposes. A popular application is spelling where users can select letters on a screen. One of the first EEG-based BCIs developed by Farwell and Donchin used P300 potentials to spell [5]. Subjects were presented with a 6-by-6 grid containing letters and characters. Rows and columns were flashed in a sequential order while the subject focused on the intended character. When a row or column containing the character is flashed a P300 potential is elicited. By combining the row and column that elicited P300s, the character can be decoded. Other BCI spellers have been developed using slow cortical potentials (SCPs) [43] and SSVEPs [44]. Other communication oriented applications include moving cursors

[45] and web browsing [46].

There have been numerous studies that have used BCIs to control motor-assistive devices for users experiencing loss of motor control due to ALS, spinal cord injury (SCI) or stroke. A popular method has been to use functional electrical stimulation (FES) where electrodes can stimulate certain muscles to aid in movement. The stimulators can be controlled by a BCI to bypass the injury and restore movement in the affected areas. Our group was able to assist an individual with paraplegia in over-ground walking using a BCI-FES system in 2015 [47]. Another approach has been to use BCIs to control robotic arms, orthoses or exoskeletons to aid with locomotion. While invasive modalities such as MEAs have shown more accurate control with higher degrees of freedom [6], there have been an increasing number of studies investigating the use of EEG for robotic-assistive devices [48].

Studies that have used BCIs to aid with movement have also reported that the brain can adapt to the feedback via changes in neural connections [49]. This is promising for individuals with stroke as new neuronal pathways can be formed to regain motor control or other brain functions. This has opened up the field of using BCIs for neurorehabilitation. Studies have shown that using BCI-controlled robotic orthoses coupled with physical therapy can lead to motor recovery [50].

Individuals suffering with motor disabilities can also utilize BCIs to control basic environmental functions such as TVs, lights and thermostats. For instance a study used residual motor abilities in individuals with motor disabilities to control a TV, lights, a telephone as well as other applications [51]. Historically BCIs have mainly been used for medical or therapeutic reasons by impaired users however there has been growing interest in non-medical applications for healthy individuals. Although most healthy people can communicate with the outside world using motor functions at faster rates than with BCIs, BCIs may be able to augment a user's experience which is of particular interest to the entertainment industry. People could use motor imagery (MI) or other control signals to control computer games [52]

or humanoid robots [53].

2.3 Signal processing pipeline

The pipeline for BCIs usually consists of signal acquisition, preprocessing, feature extraction, classification and an application interface [28]. Some of these steps may be omitted or combined with other steps depending on the application or algorithms being used. For instance, it is common to omit preprocessing and combine feature extraction, selection and classification when using deep neural networks. Following signal acquisition, the data may be preprocessed to reduce noise and remove artifacts. Artifacts in the EEG signal can be biological or non-biological. Biological artifacts are organic sources of electrical signals measured by EEG that don't originate from neuronal processes in the brain. They include eye movements or electrooculography (EOG), muscle activity or electromyography (EMG) and heart rhythms or electrocardiography (ECG) [54]. Non-biological artifacts arise from electrical sources in the environment including power line noise, electronic devices and poor electrode impedances. Artifacts and noise can be minimized by detrending the data, using various filters such as bandpass filters or using independent component analysis (ICA) to omit certain artifacts. Detrending removes the baseline drift in short segments of data. Filtering the data can omit noise that resides at certain frequencies such as 60 Hz power line noise or low frequency drift below 1 Hz. ICA is a method that separates independent sources contributing to a signal [55]. It can be used to extract sources of noise in the signal and reconstruct the denoised signal.

Feature extraction methods typically reduce the dimensions of the data while amplifying differences between classes. They include band power (BP) features [56], power spectrum density (PSD) [57] and common spatial pattern (CSP) [58]. BP features are calculated by summing the powers of frequencies in specific bands. They can be useful for control signals

whose activities are known to lie in specific frequency bands such as alpha waves for eyes open and closed tasks or mu and beta waves for sensorimotor rhythms. PSD is the power at frequencies up to half the sampling frequency, also known as the Nyquist frequency. PSDs are useful for control signals where the associated features are more evident in the frequency domain than in the time domain. CSP transforms the data into a subspace that maximizes the variance between classes.

2.4 Machine learning algorithms

Classification algorithms evaluate the extracted features and compute decision boundaries in order to classify data instances into one of the class labels. They include linear discriminant analysis (LDA), support vector machines (SVMs), k-nearest neighbor (KNN), Bayesian classifiers and neural networks (NNs). LDA is a linear classifier that creates a hyperplane in the feature space to separate classes [59]. It produces the hyperplane that maximizes the distance between the class means and the inter-class variance. It is computationally efficient, typically used for binary BCI classifiers [3], and extendable to multi-class using multiple hyperplanes. It doesn't perform well when presented with outliers or noisy data. It is commonly used in MI [60] and P300 [61] tasks.

SVM, like LDA, operates by creating hyperplanes to separate data into different classes. It computes the hyperplanes by maximizing the margin in between the different classes [62]. The feature vectors that lie on the edges of the margin are known as support vectors. This method creates a linear decision boundary and is known as linear SVM. There is also a non-linear version of SVM for more complex problems where a non-linear decision boundary is preferred. It is achieved by casting the data into a higher-dimensional space by deriving new features from the original ones. A linear decision boundary is then used in the higher-dimensional space to separate the classes. SVM is effective for high-dimensional datasets

with small sample sizes which is characteristic of many BCI datasets. However like LDA, it is not well suited for outliers and noisy data. SVM is a powerful algorithm popularly used for BCI applications including MI [63] and P300 spellers [64]. A study also demonstrated that SVM outperformed LDA in five different EEG mental tasks [65].

KNN classifies data by looking at the classes of its nearest neighbors in the feature space [59]. It can produce non-linear decision boundaries and is very efficient with low-dimensional data. It doesn't perform well however with large or high-dimensional datasets. KNN has been shown perform equally or better than SVM for some BCI tasks including MI [66] and emotion or engagement level detection [67].

Bayesian classifiers use the Bayes rule to classify data points which works by assigning probabilities of feature vectors belonging to each class [68]. The data point is then assigned to the class with the highest calculated probability. They can produce non-linear decision boundaries however are used less commonly for BCIs. There are different kinds of Bayesian classifiers including Bayes quadratic and Hidden Markov models. Bayesian classifiers have been used on sensorimotor rhythms and cursor movement applications in the BCI field [69, 70, 71].

Finally neural networks were initially designed to mimic how the brain processes information. They consist of layers of neurons that compute the weighted sum of neurons from the previous layer followed by a non-linear activation function [72]. They can learn complex patterns from raw data reducing the need for hand engineered features. However they require a lot of computing power as well as large training datasets. The simplest NN is a fully connected neural network (FCNN) which consists of one or more hidden layers in between the input and output layers where each neuron is connected to all neurons from the previous layer. There are more complex architectures including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Neural networks have been receiving a lot of attention recently within the BCI field and have outperformed other common ML algorithms. They've been

commonly used for MI tasks in the literature outperforming popular algorithms including SVM [73, 74] and filter bank common spatial patterns (FBCSP) [73, 75, 76]. Additionally, studies have designed NN architectures that can generalize to multiple control signals and paradigms [77, 78] with competitive results.

Chapter 3

Description of decoding algorithms

In this section, I will describe the concepts and mathematical calculations behind the decoding algorithms used in the following sections. All algorithms presented are trained using supervised learning where the labels of the data are known. Additionally, all algorithms will be described in the context of binary classification since there are two classes for both datasets in the analysis.

3.1 CPCA+AIDA+Bayesian (CAB)

The first model is one that was established in our lab and has been successfully used in multiple online BCI experiments. The model consists of three methods, classwise principal component analysis (CPCA), approximate information discriminant analysis (AIDA) and a Bayesian classifier. The initial two are used for feature extraction and dimensionality reduction while the final one is used to classify the input. Principal component analysis (PCA) rotates the data to produce orthogonal projections/components along the axes of maximum variability in the data. This allows you to select only a few of the top components

that explain the majority of the variability in the data and discard the rest. This helps to get rid of noise in the data and reduces the dimensions for more efficient processing. CPCA [79] extends this by producing multiple subspaces each with its own components where each subspace is trained using data from one class and offset by the difference in class means to account for subspaces whose components are almost parallel. This is beneficial for multiclass problems as the method utilizes class-wise statistics to extract more useful components for each class. New data is typically best represented in the subspace the data belongs to however this isn't always the case during classification. For a c-class problem, c feature extraction matrices are produced by the method. Test data can then be projected to each subspace and class specific probabilities can be calculated to determine the class the data belongs to. AIDA [80] is another supervised feature extraction method that helps to reduce the dimensionality of the data while maintaining or enhancing class separability. The overall feature matrices are then the product of the feature matrices produced by CPCA and AIDA. Let i be the number of subspaces, n be the dimension of each data sample and m_i be the size of the feature vector selected by the user, then the feature matrix $F_i \in \mathbb{R}^{n \times m_i}$ is calculated as $F_i = F_i^{CPCA} F_i^{AIDA}$ where F_i^{CPCA} and F_i^{AIDA} are the CPCA and AIDA feature matrices respectively. For a given test data vector $\bar{x}$, the feature vector $\bar{x}^i \in \mathbb{R}^{m_i}$ is then calculated as $\bar{x}^i = F_i^T \bar{x}$. The Bayes rule is then applied to determine the probability of the feature vector belonging to each class w_j for $1 \leq j \leq c$ where c is the number of classes in each subspace

$$P(w_j^i \mid \bar{x^i}) = \frac{f(\bar{x^i} \mid w_j^i) P(w_j)}{f(\bar{x^i})} \tag{3.1}$$

where $P(w_j^i \mid \bar{x^i})$ are the posterior probabilities, $P(w_j)$ are the class prior probabilities, $f(\bar{x^i} \mid w_j^i)$ are the class-conditional probability density functions (PDFs) estimated as Gaussian densities in this case and $f(\bar{x^i})$ are the marginal distributions. The class with the highest probability of the data belonging to is then determined using the maximum a posteriori rule

as

$$J(i) = \arg\max_{1 \leq j \leq c} P(w_j^i \mid \bar{x^i}) \tag{3.2}$$

The predicted class is determined by finding the subspace that best discriminates the data, calculated as

$$I = \arg\max_{1 \leq i \leq c} P(w_{J(i)}^i \mid \bar{x^i}) \tag{3.3}$$

making $J(I)$ the predicted class.

3.2 Logistic Regression

Logistic regression can be thought of as an extension of linear regression. Linear regression looks at the linear relationship between features to predict a continuous outcome. This can be expressed with the equation $\hat{y} = \sum_{i=1}^{n} w_i x_i + b$ where n is the number of features. The goal of logistic regression is to classify the inputs and therefore a discrete outcome is required instead of a continuous one. This can be accomplished by predicting the probability of the input belonging to each class. So instead of predicting a continuous variable y, we predict the log odds expressed as $\ln \frac{p}{1-p}$ for binary classification which ends up equating to $p = \frac{1}{1+e^{-\hat{y}}}$, commonly known as the sigmoid function. This results in a value between 0 and 1 where one class is predicted above 0.5 and the other below 0.5. The error of the outcome can be measured using a loss function which for logistic regression is the cross entropy loss function:

$$L(p, y) = -\left(y \ln(p) + (1 - y) \ln(1 - p)\right). \tag{3.4}$$

For correct predictions p is close to y and the loss is small however when p is different from y eg when $y = 1$ and $p \approx 0$ the loss is very large. The log odds and cross-entropy functions can be extended for multi-class problems. The weights w and bias b can then be optimized to minimize the loss using gradient descent.

3.3 Neural networks

Neural networks were designed to mimic how the brain processes information via neuronal connections. The main and simplest components of neural networks are neurons. Neurons can have multiple inputs and one output. Each input is weighted to determine the importance of the input to the outcome of the neuron. The weighted inputs are summed up and typically applied to a nonlinear activation function. Often a term is added to the sum known as a bias resulting in the output also known as the activation of the neuron. The function can be expressed as

$$a = g\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.5}$$

where a is the activation of the neuron, g is the activation function, n is the number of input neurons, w_i are the weights connected to the input neurons, x_i are the activations of the input neurons and b is the bias term. There are multiple activation functions with the most common ones being the rectified linear unit (ReLU), the sigmoid function and the hyperbolic tangent (Tanh). The simplest form of neural networks are fully connected neural networks (FCNN) which are comprised of multiple layers of neurons between the input and output of the model also known as hidden layers where each neuron in a layer is connected to every neuron from the previous layer. The first layer consists of the input features and the final layer is the output of the model which is used to predict the class labels. For binary classification tasks the final layer is a single node with a sigmoid activation producing a value

between 0 and 1. This value can be thought of as the probability of the input belonging to the class with label 1. Similar to LR one class is predicted for values under 0.5 while the other class is predicted for values greater than 0.5.

The learnable parameters of the model are the weights and biases of each neuron which are trained using the backpropagation algorithm. To understand backpropagation, we first need to talk about gradient descent which is the method used to reduce the error associated with the predicted value. The error of the model can be determined using a loss function which measures how far off the predicted value is from the class label. The loss can be calculated for each sample or over multiple samples in which case it is termed the cost function. A common loss function used for classification problems is the cross-entropy loss function (Eq 3.4), also used in logistic regression. The closer the class labels and predicted values are the lower the cost making this function a good proxy to the performance of the model. As this is a function of the output of the model it also follows that it is a function of the weights and biases of the model. Therefore by taking partial derivatives of the weights and biases with respect to the cost we can compute the effect of small changes in the weights on the cost and slightly increment or decrement them to lower the cost. This is known as gradient descent as we are moving in a direction opposite of the gradient to reduce the cost. The equation to change the weights in order to reduce the cost using gradient descent can be represented as

$$W_i = W_i - \alpha \frac{\partial J}{\partial W_i} \tag{3.6}$$

where W_i is the ith weight, α is known as the learning rate and $\frac{\partial J}{\partial W_i}$ is the partial derivative of the cost J with respect to weight W_i. The minus sign before the partial derivative indicates that we are moving in the direction opposite of the derivative therefore lowering the cost and the learning rate affects the step size or the magnitude of change to the weight. To better understand this we can imagine a graph where the cost function is a hyperplane and the axes are the cost, weights and biases of the model. We can use a simple example of a 3D

graph where the x and y axes are weights, say W_1 and W_2, and the z-axis is the cost. The surface of the cost function would have minimas and maximas in relation to the weights. Therefore gradient descent can be thought of as a ball rolling on the function surface till it reaches a trough. Since the surface contains multiple minimas which only increase with the number of dimensions as the number of weights increase, gradient descent typically leads to a local minima and not the global minima. However for multidimensional complex problems, there are typically multiple minimas that result in adequate performance so reaching one of those minimas is sufficient. The backpropagation algorithm starts by computing the partial derivative of the cost function with respect to the predicted value. Then using the chain rule the partial derivative of the cost with respect to the weights in the previous layer can be computed. And by continuing to use the chain rule the partial derivatives can be computed back to the weights of the first layer. This process of moving back through the layers of the network is where backpropagation gets its name from. By using batches of training samples this process can be repeated several times to lower the cost and optimize the parameters of the model.

3.3.1 Convolutional Neural Networks

One of the most prevalent neural network architectures are CNNs. They have shown great success in multiple fields but are most known for their image processing capabilities. CNNs consist of convolutional layers each composed of multiple filters that, as the name suggests, convolve over the input array and are multiplied elementwise with the portion of the image they are over. This can be expressed for a 2D input and filter as

$$S(x, y) = \sum_{i=1}^{k} \sum_{j=1}^{l} I(x + i, y + j)F(i, j) \tag{3.7}$$

where $I(x, y)$ is the value at position (x, y) of the input array, F is a 2D filter with the size $k \times l$, and $S(x, y)$ is the the sum or convolution output. Similar to feedforward networks, a bias term is often added to the sum and an activation function is applied to the output. In this sense, the components of the filters can be seen as the weights in the conventional FCNN model. One could wonder, why don't we just flatten the image into a tall vector and process it using a FCNN? There are multiple benefits that make CNNs superior to FCNNs when it comes to processing multidimensional arrays. Firstly they preserve the spatial aspect of the data enabling the network to capture spatial features in the array. Next, the filter elements or weights are reused throughout the array allowing the network to generalize features located in different locations of the array. Reusing the same weights also greatly reduces the number of parameters the network needs to learn, speeding up the computation during training.

Filters can have as many dimensions as the input arrays. The output of the convolution can either be the same size as the input or smaller. To obtain an output with the same size, the input array is padded with zeros in order to make it larger so that it shrinks to its original size following convolution. Typically a pooling layer is applied after each convolution layer. Pooling layers combine values in order to reduce the dimensionality of the data. The two most common types of pooling are average pooling and max pooling where average pooling averages all the values over a portion of the image and max pooling replaces the portion with its maximum value. The benefits of having a pooling layer is to reduce the dimensionality of the data therefore reducing the number of parameters of the model and speeding up the computation. Additionally they help to further extract features from the data especially in the case of max pooling as it reduces the noise of low valued and insignificant activations. CNNs usually consist of multiple pairs of convolution and pooling layers. The first layers typically extract low level features such as lines, edges or simple shapes in images whereas the deeper layers extract more abstract and high level features of the image such as eyes, ears or other facial features for facial recognition tasks. The output of the last convolution layer is then usually flattened and fed to a standard feedforward network for classification

purposes.

3.3.2 Recurrent Neural Networks

The second most common type of neural networks are RNNs. The main advantage of RNNs is that they are able to store memory of previously encountered inputs and use that information in the decision of future outputs. They do this by storing a hidden state which is updated after each input is fed to the network. This makes RNNs great at processing temporal sequences, they are especially known for language processing. The hidden state is computed using the following equation:

$$h_t = f\left(w_x x_t + w_h h_{t-1} + b_h\right) \tag{3.8}$$

where f is the activation function, w_x is the input weight matrix, x_t is the input at time t, w_h is the hidden state weight matrix, h_{t-1} is the hidden state at the previous time step and b_h is the hidden state bias vector. This expression is similar to the computation of activations in a feedforward neural network with the addition of $w_h h_{t-1}$ which is the weighted hidden state from the previous time step. The output is then computed from the hidden state as follows:

$$y_t = f\left(w_y h_t + b_y\right). \tag{3.9}$$

There are many types of RNNs including one-to-one, one-to-many, many-to-one and many-to-many. For our binary classification task, a many-to-one RNN is used that produces a single output, corresponding to the predicted class label, for multiple inputs. Backpropagation works in the same way as in a feedforward model with an additional temporal component. The weights are updated by applying the chain rule back through previous timesteps which is known as backpropagation through time. One of the major issues in training RNNs is the

vanishing gradient problem. As the number of timesteps in the model increases the further back you need to backpropagate to update the weights. This means you need to multiply multiple gradients together which are often less than 1 as in the case of sigmoid or Tanh activation functions which reduces the update value of the weights down to 0 hence you can reach a point where the weights are updated very slowly. There are a couple of architectural solutions to this problem with the most common 2 being gated recurrent units (GRUs) and long-short term memory (LSTM) units. These are more advanced recurrent units that include additional elements known as gates that control what information to retain from the input and hidden states and what to discard. GRUs contain a reset and update gate which control how much information to retain from the hidden state and how much of the new information from the input to incorporate respectively. Similarly LSTMs contain three gates which are the input, output and forget gates which perform a similar function to GRU gates.

3.4 SVM

The objective of the SVM algorithm is to find a hyperplane or multiple hyperplanes in the case of multiclass problems that separate the data belonging to each class while maximizing the margin between the classes. This ends up being a line in 2-D, a plane in 3-D or a multidimensional hyperplane in higher dimensions. The equation of the hyperplane is $w^T x + b = 0$ where the positive class is predicted if $w^T x + b \geq 0$ and the negative class if $w^T x + b < 0$. The closest points perpendicular to the decision boundary for each class become support vectors. For mathematical convenience, the hyperplane equation is scaled so that $w^T x + b = 1$ for support vectors in the positive class and $w^T x + b = -1$ in the negative class. Therefore the width of the margin between support vectors in each class is $\frac{2}{||w||}$ where $||w||$ is the euclidean norm of w. The objective then becomes to maximize $\frac{2}{||w||}$ which is also the same

as minimizing $||w||$ within the constraints of keeping the training data out of the margin and in their respective classes which can be expressed as $y_i \left(w^T x_i + b\right) \geq 1$ where y is -1 for the negative class and +1 for the positive class. If all the training data is linearly separable then the margin is known as a hard margin. Most of the time however this is not the case and the margin becomes a soft margin as some data points lie on the wrong side of the decision boundary. The optimization constraint is then modified to $y_i \left(w^T x_i + b\right) \geq 1 - \zeta_i$ where ζ_i is a non-negative tunable parameter known as a slack variable allowing the constraint to be met when data points fall on the wrong side of their margin line. ζ_i can be an arbitrarily large number to ensure all points meet the constraint requirement however this increases the tolerance of misclassified points which is not desired. To avoid this a regularization term is added to penalize the model for high ζ values. The optimization equation then becomes

$$minimize\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\zeta_i \tag{3.10}$$

where m is the number of samples and C is the regularization term. If C is very large then the model will be sensitive to misclassifications and overfit to the training data. On the other end if C is very small the model will have a larger margin however it will likely underfit to the data. This optimization problem is solved using Lagrangian multipliers which we won't cover the specific maths of here. However the final solution of the Lagrangian problem contains a summation over all $x_i \cdot x_j$ which are dot products between each pair of sample points. This term can be rewritten as $k(x_i, x_j) = x_i \cdot x_j$ called a kernel function which is useful as we extend the method to non-linearly separable data. For most problems, the classes require non-linear hyperplanes to optimally separate the data. To achieve this, the solution is computed in a higher dimension where the data are linearly separable. This is done by using a kernel function that computes the feature dot products in another space without having to compute the transformation of the data into the other space. Popular kernels are linear: $k(x_i, x_j) = x_i \cdot x_j$, polynomial: $k(x_i, x_j) = (x_i \cdot x_j + c)^d$ and RBF (Radial Basis

Function): $k(x_i, x_j) = exp\left(-\gamma||x_i - x_j||^2\right)$. RBF is the most popular kernel as it results in a complex boundary by transforming the data into an infinitely dimensional space. The constant γ is a regularization term, small values of γ reduce the complexity of the boundary resembling a linear boundary whereas large values of γ increase the complexity however may lead to overfitting.

3.5 KNN

KNN is a simple algorithm however it can powerful and is often used for low dimensional data. There is one primary selectable variable k corresponding to the number of neighbors used to classify the data. The algorithm selects the k closest neighbors using euclidean distance as the distance measure and predicts the class with the highest occurrence among the neighbors. One modification to this method is weighing each neighbor proportionally to the inverse of its distance giving higher priority to closer neighbors. Low values of k result in noisy predictions while high values of k increase the prediction confidence however blur the class boundaries. KNN usually performs well with low dimensional data however at high dimensions all points start becoming equidistant to all other points rendering KNN ineffective.

Chapter 4

A comparative analysis on P300 signals

4.1 Methods

4.1.1 Dataset Description

This dataset consists of data collected from 7 able bodied individuals. Each subject has a variable number of sessions ranging from 1-12 with a mean of 4.875. During each session the subject was instructed to focus on a single character on an onscreen speller for 30s. The speller contains 42 letters/characters arranged in a 6x7 grid (Figure 4.1). During this time characters were flashed in a semi-random fashion in groups of 6. Characters were randomly organized into the groups with preference to more common characters appearing in the earlier groups. More detail on this grouping algorithm can be found in [1]. All 7 groups, each containing 6 characters, were flashed sequentially to cover all 42 characters in the grid. The ratio of occurrence of P300 potentials (oddballs) to absence of P300 potentials (non-

oddballs) is therefore 1:6 since only one group out of the seven contained the target character. Following one traversal of the grid, the groups were re-randomized and the characters were flashed again. This procedure was repeated for the 30s duration. There was a brief break after and then the subject was instructed to focus on a different character. This procedure was repeated for 10 different characters and each session lasted for $\sim$ 6.5 mins. Trials of 400ms from the start of each flash were segmented. The time between flashes or the inter-trial interval was 400ms and the duration of each flash was 250ms. A 19-channel EEG cap was used to collect data where the channels were arranged according to the 10-20 International standard. Data was only collected from a subset of 8 channels (C3, Cz, C4, P3, Pz, P4, O1, O2) covering the central, parietal and occipital regions of the brain. The data was bandpassed between 1-35 Hz and digitized at a sampling rate of 200 Hz using Biopac amplifiers and the MP150 acquisition system. Therefore each 400ms trial resulted in a data matrix of 8x80 corresponding to the number of channels by time points. More details of the experimental procedure and data collection can be found in [1]. The objective of the classifier is to classify each trial as oddball or non-oddball by detecting a P300 potential in the oddball trials.

4.1.2 Pre-processing

As mentioned above, data was segmented into 400ms trials from the stimulus onset. As the P300 potential is a low frequency change in amplitude and since each trial contains one instance of a P300 potential, it makes more sense to keep the data in the time domain rather than switching to the frequency domain. The data was z-score normalized to avoid a bias towards certain features/data points in the dataset.

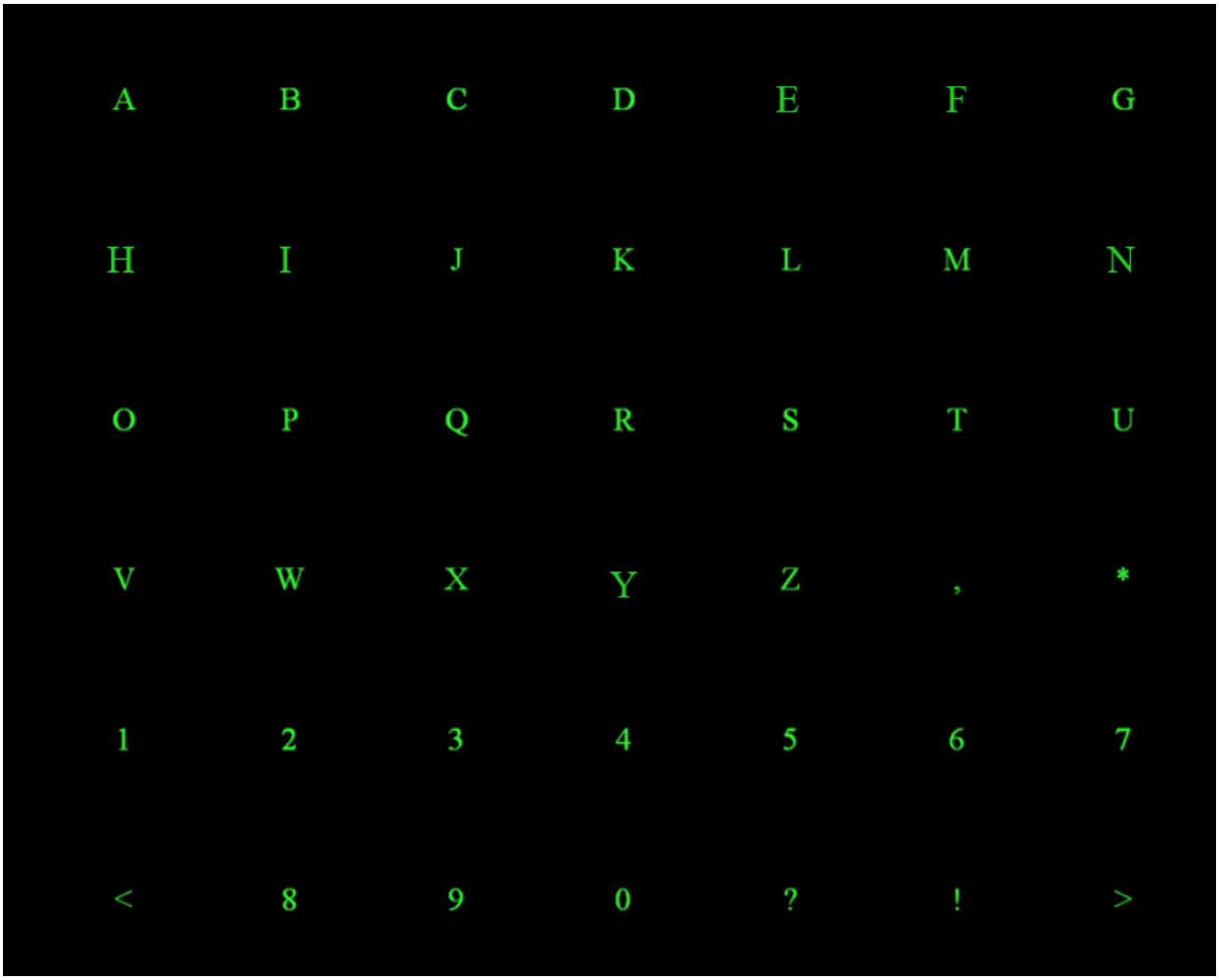

Figure 4.1: Layout of the P300-speller grid. Adapted from [1].

4.1.3 Model parameters

A subset of the dataset (4 out of 39 sessions) was used as a validation set to optimize the parameters/hyperparameters used by the different algorithms.

CAB

The components retained during the CPCA procedure are ones whose corresponding eigenvalue is greater than the mean of all eigenvalues. The size of the feature vector resulting from the AIDA procedure is chosen to be 1. A linear PDF was used for the classifier and

the priors were empirical representing the probability of observing each class in the training data.

LR

The LogisticRegression class in the scikit learn library was used to train the LR algorithm. A coordinate descent algorithm was used as the optimizer for the LR model which is similar to gradient descent.

NN

The Keras library was used to train the NN models. A FCNN and a CNN were developed and tested on the data. The hyperparameters that were tuned included the learning rate, weight decay, batch size, number of epochs, number of fully connected layers, number of hidden units in each layer, batch normalization, dropout rate, kernel regularization, number of convolutional layers, number of filters in each convolutional layer, and filter shape for each convolutional layer.

The FCNN model consists of 1 hidden layer followed by the output layer. The architecture is outlined in Table 4.1. The input matrix of shape (8,80) is flattened to a vector of shape (640,1). The vector is connected to a fully connected layer also known as a Dense layer with 64 nodes. The outputs of the Dense layer are passed on to a batch normalization layer. Batch normalization is a two step process that initially normalizes the data to their z-score value. The normalized values are then linearly transformed as follows:

$$Z = \gamma * Z_{norm}^{(i)} + \beta \tag{4.1}$$

where $Z_{norm}^{(i)}$ are the normalized features and γ and β are two learnable parameters that

scale and shift the features. A ReLU activation is then applied to the output vector and a dropout layer is used to reduce over fitting. The activations are then passed to a Dense layer with 2 nodes with a softmax activation that predicts the likelihood of the sample belonging to each class.

Table 4.1: Fully connected neural network architecture for P300 dataset.

Layer	Layer type	# nodes	# parameters	Output dimension	Activation function
1	Input			(8,80)	
	Flatten			(640,1)	
2	Dense	64	41,024	(64,1)	
	BatchNorm		256	(64,1)	
	Activation			(64,1)	ReLU
	Dropout			(64,1)	
Classifier	Dense	2	130	(2,1)	
	Activation			(2,1)	Softmax

The CNN consists of 3 convolutional layers followed by 1 fully connected layer and the output layer (Table 4.2). The (8,80) matrix input is fed to a convolutional layer with 32 filters of size (8,1) spanning the entire channel or spatial dimension to produce a (1,80,32) matrix. This layer therefore computes spatial relations and can help in identifying important channels for the classification task. The output is then fed to a batch normalization layer and activated with the ReLU function. The matrix is then passed to a Maxpool layer with a (1,2) filter that replaces every 2 consecutive data points in the time dimension with their maximum value. This reduces the dimension of the data while keeping the most important activations with the highest values. There are another two blocks of convolution consisting of a convolutional, batch norm, activation and max pool layers operating in the time dimension using convolution filter sizes of (1,10). The final output from the last convolutional block with size (1,3,128) is then flattened and passed to a fully connected block consisting of a 64 node Dense, batch normalization, activation and dropout layers. The final layer is a 2 node Dense layer with a softmax activation to classify the data.

Table 4.2: Convolutional neural network architecture for P300 dataset.

Layer	Layer type	# nodes	# filters	Filter size	# parameters	Output dimension	Activation function
1	Input					(8,80)	
2	Conv2D		32	(8,1)	288	(1,80,32)	
	BatchNorm				128	(1,80,32)	
	Activation					(1,80,32)	ReLU
	Maxpool			(1,2)		(1,40,32)	
3	Conv2D		64	(1,10)	20544	(1,31,64)	
	BatchNorm				256	(1,31,64)	
	Activation					(1,31,64)	ReLU
	Maxpool			(1,2)		(1,15,64)	
4	Conv2D		128	(1,10)	82048	(1,6,128)	
	BatchNorm				512	(1,6,128)	
	Activation					(1,6,128)	ReLU
	Maxpool			(1,2)		(1,3,128)	
5	Flatten					(384,1)	
	Dense	64			24640	(64,1)	
	BatchNorm				256	(64,1)	
	Activation					(64,1)	ReLU
	Dropout					(64,1)	
Classifier	Dense	2			130	(2,1)	
	Activation					(2,1)	Softmax

SVM

The SVC class in the scikit learn library was used to train the SVM algorithm. After comparing the results on the validation set using different kernels, the RBF kernel was found to perform the best and is used for the SVM model. The value of the regularization parameter C was selected as 100 and the RBF kernel coefficient gamma was selected as 0.0001.

KNN

The KNeighborsClassifier class in the scikit learn library was used to train the SVM algorithm. The value of k indicating the number of neighbors was selected to be 8 and neighbors were weighted based on their distance meaning closer neighbors had higher influence on the decision than further neighbors.

4.1.4 Within subject cross-validation

The number of trials per session ranged from 593 to 740. Trials from each session were randomly shuffled and split evenly into 10 folds. A 10-fold stratified cross-validation was performed on data from each session where 1 fold was used as the validation set while the remaining 9 folds were combined to form the training set. This was repeated till each fold had been held out for validation. This was repeated 10 times to test on alternate shuffles of the data.

4.1.5 Leave-one-subject out

Next, all runs for each subject were combined and a leave one out cross validation across subjects was performed where one subject's data would be held for validation while the data from the remaining 7 subjects were used for training. This was repeated till all subjects were used as the validation set.

4.1.6 Performance metrics

To determine the effectiveness of each classifier and to be able to compare them to each other, we need to calculate performance metrics for the different classifiers across the datasets. The chosen metrics were accuracy, precision, recall and F1 score defined using the following equations:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.5}$$

where TP, FP, TN and FN stand for true positives, false positives, true negatives and false negatives respectively.

4.1.7 Statistical analysis

To test if differences in performance metrics are significant between the different algorithms an appropriate statistical test needs to be used. While common methods include the parametric paired Student t-test [81] or the non-parametric Wilcoxon signed-rank test [82], those methods test for difference between two sets of data. To test for differences between multiple classifiers a multiple hypotheses statistical test needs to be used. It is generally not recommended to run single hypothesis tests such as the t-test between all set pairs of data as it is expected to randomly reject a few null hypothesis by chance when running multiple tests. ANOVA is a common multiple hypotheses test that looks for differences between the sets of data and rejects the null hypothesis if the variability between datasets is sufficiently large. ANOVA is however a parametric test and makes assumptions including that the data

is normally distributed, data samples are independent of each other and that the datasets have similar variance. A non-parametric equivalent to ANOVA is the Friedman test which ranks each dataset for every sample of data and computes the average rank for each dataset. The null hypothesis states that the average ranks are equal between datasets and is rejected when they are not. If the null hypothesis is rejected, the post-hoc Nemenyi test can be used to compare all pairs of datasets to compute which are significantly different from each other.

4.1.8 Visualization techniques

The weights or coefficients learned by each algorithm can be visualized as a matrix with the same shape as the input to identify the most important regions used to discriminate the classes. This cannot be performed for KNN however since it is a non-parametric algorithm meaning it doesn't learn parameters based on the training data and therefore has no coefficients to visualize. Apart from NNs, the algorithms are linear and therefore each input feature has a corresponding coefficient which can be displayed in the matrix input format of channels x time. Although a non-linear RBF kernel is used for our SVM model, it is substituted with a linear kernel for the visualization to obtain coefficients in the original dimension space. Since NNs are non-linear and have multiple layers of weights, we can't correspond each input feature with a coefficient. A method called activation maximization was used instead to identify important features. This method works by iteratively modifying the input to the model with the aim of maximizing the likelihood of the input belonging to a certain class. This can give us an idea of which input features are deemed more important in deciding which class the input belongs to. However this method is not a direct visualization of the model coefficients in contrast to the other algorithms and results can vary depending on the parameters used to modify the input as well as the number of iterations used for the optimization.

4.2 Results

4.2.1 Within subject cross-validation

The 10x10 fold cross-validations performed on each session are averaged for each subject and the mean accuracies are reported in Table 4.3. The Friedman test is significant with a p-value of 1.49e-20 indicating that there are differences in accuracy between the different classifiers. The Nemenyi test indicates that KNN is outperformed by all other algorithms. The CAB algorithm outperformed the CNN & SVM algorithms and FCNN outperformed SVM. The full cross-validation results for each session are given in Appendix A.

Table 4.3: Mean session 10-fold cross validation accuracies for each subject using different algorithms. Values in bold indicate the highest mean accuracies for each subject.

Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	94.34 ± 3.57	**95.91 ± 1.30**	95.73 ± 1.06	95.51 ± 1.13	94.99 ± 1.67	88.17 ± 0.68
2	2	88.68 ± 0.95	89.08 ± 0.60	**90.93 ± 0.23**	89.98 ± 0.10	90.10 ± 0.30	86.57 ± 0.61
3	3	**94.47 ± 1.66**	93.42 ± 2.44	93.82 ± 1.93	93.68 ± 1.44	93.27 ± 2.12	87.49 ± 0.66
4	1	**92.74 ± 0.00**	90.98 ± 0.00	91.76 ± 0.00	91.92 ± 0.00	90.66 ± 0.00	88.11 ± 0.00
5	6	**94.17 ± 0.72**	92.98 ± 1.61	93.72 ± 0.88	93.61 ± 0.77	93.22 ± 0.98	87.00 ± 0.75
6	12	**95.53 ± 1.54**	94.82 ± 2.37	95.30 ± 1.46	94.67 ± 1.60	94.78 ± 1.74	87.61 ± 1.34
7	5	90.35 ± 1.16	89.97 ± 0.84	**90.82 ± 0.66**	90.02 ± 0.56	90.13 ± 1.18	85.22 ± 0.58
8	2	88.69 ± 0.29	88.84 ± 0.22	**89.42 ± 0.32**	87.66 ± 0.30	89.39 ± 0.50	85.21 ± 0.18

LR has higher recall scores than all other algorithms and KNN has lower scores than all other algorithms according to the Friedman and Nemenyi tests.

Table 4.4: Mean session 10-fold cross validation recall scores for each subject using different algorithms. Values in bold indicate the highest mean recall values for each subject.

Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	68.26 ± 26.79	**85.68 ± 5.16**	77.48 ± 7.93	77.74 ± 8.87	74.88 ± 13.03	22.25 ± 7.87
2	2	35.39 ± 11.39	**63.42 ± 2.76**	51.86 ± 2.43	53.94 ± 0.63	57.88 ± 3.07	18.01 ± 8.10
3	3	73.26 ± 7.43	**76.98 ± 8.44**	68.45 ± 10.02	72.30 ± 6.33	72.41 ± 8.83	18.49 ± 5.56
4	1	62.92 ± 0.00	**68.76 ± 0.00**	58.99 ± 0.00	63.71 ± 0.00	61.12 ± 0.00	27.87 ± 0.00
5	6	69.91 ± 5.74	**75.37 ± 5.64**	67.08 ± 5.04	70.30 ± 5.15	70.12 ± 4.49	14.72 ± 4.63
6	12	78.38 ± 7.33	**84.27 ± 6.79**	77.05 ± 7.15	77.06 ± 6.82	77.96 ± 6.16	19.43 ± 10.30
7	5	51.68 ± 6.15	**62.30 ± 2.94**	51.31 ± 4.31	54.38 ± 3.27	54.06 ± 6.94	2.74 ± 1.68
8	2	43.81 ± 1.24	**59.34 ± 0.28**	42.14 ± 0.33	39.23 ± 0.09	49.28 ± 1.39	4.48 ± 1.23

CAB and FCNN have higher precision scores than all other algorithms. CNN and SVM have

higher precision scores than LR.

Table 4.5: Mean session 10-fold cross validation precision scores for each subject using different algorithms. Values in bold indicate the highest mean precision values for each subject.

Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	91.32 ± 4.41	86.07 ± 5.62	**91.60 ± 3.39**	89.98 ± 3.35	88.83 ± 3.86	85.40 ± 6.62
2	2	71.95 ± 0.52	61.84 ± 1.46	**77.91 ± 0.88**	69.78 ± 1.11	68.53 ± 0.29	61.95 ± 0.84
3	3	**86.59 ± 6.51**	77.73 ± 9.04	85.73 ± 6.07	82.13 ± 5.02	79.17 ± 7.26	80.43 ± 3.06
4	1	**82.35 ± 0.00**	68.54 ± 0.00	77.94 ± 0.00	76.00 ± 0.00	70.13 ± 0.00	71.43 ± 0.00
5	6	**86.87 ± 1.11**	75.49 ± 6.12	85.65 ± 2.87	82.31 ± 2.03	80.11 ± 4.05	73.04 ± 10.96
6	12	**89.19 ± 4.69**	81.09 ± 8.74	88.70 ± 4.26	84.53 ± 5.25	84.67 ± 6.56	78.87 ± 13.03
7	5	73.77 ± 4.82	66.59 ± 3.79	**77.76 ± 2.75**	70.17 ± 1.88	70.74 ± 4.18	41.57 ± 25.77
8	2	66.18 ± 0.96	61.58 ± 0.80	**73.09 ± 2.77**	60.82 ± 2.25	68.23 ± 2.44	35.06 ± 7.79

All algorithms have higher F1 scores than KNN and LR has higher F1 scores than SVM.

Table 4.6: Mean session 10-fold cross validation F1 scores for each subject using different algorithms. Values in bold indicate the highest mean F1 values for each subject.

Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	73.28 ± 26.57	**85.76 ± 4.47**	83.64 ± 4.96	83.02 ± 5.52	80.41 ± 8.87	34.35 ± 9.07
2	2	46.40 ± 10.47	62.62 ± 2.10	62.22 ± 1.47	60.83 ± 0.02	**62.72 ± 1.93**	26.96 ± 9.91
3	3	**79.31 ± 6.89**	77.35 ± 8.69	76.00 ± 8.54	76.88 ± 5.75	75.62 ± 8.13	29.72 ± 7.54
4	1	**71.34 ± 0.00**	68.65 ± 0.00	67.15 ± 0.00	69.31 ± 0.00	65.32 ± 0.00	40.10 ± 0.00
5	6	**77.34 ± 3.71**	75.42 ± 5.84	75.19 ± 4.20	75.77 ± 3.70	74.76 ± 4.13	24.15 ± 6.65
6	12	**83.38 ± 6.08**	82.60 ± 7.66	82.40 ± 5.87	80.60 ± 6.10	81.16 ± 6.24	29.96 ± 12.79
7	5	60.71 ± 5.75	**64.37 ± 3.29**	61.76 ± 3.69	61.26 ± 2.76	61.16 ± 5.75	5.05 ± 3.23
8	2	52.72 ± 1.20	**60.44 ± 0.53**	53.44 ± 0.47	47.68 ± 0.62	57.23 ± 1.79	7.93 ± 2.14

Figure 4.2 displays the features of interest for the different algorithms trained on data from a single session. There are a lot of common features of interests shared between the algorithms. There's a prominent region of interest around 200-300ms for CAB, LR and SVM at O1 which likely corresponds to the N200 potential [83]. This region is absent from the FCNN and CNN algorithms which may potentially be explained by the algorithms not identifying these regions as discriminative, or deeming other features more relevant and having higher discriminative power or the activation maximization method didn't pick up on all the regions of interest for those algorithms. All of the algorithms identified a region of interest around 200ms at Pz with varying degrees which likely corresponds with the P200 potential [83]. All algorithms additionally identified P4 at 300ms as a region of interest which likely corresponds to the P300 potential. The feature maps for the NN algorithms are noticeably noisier than

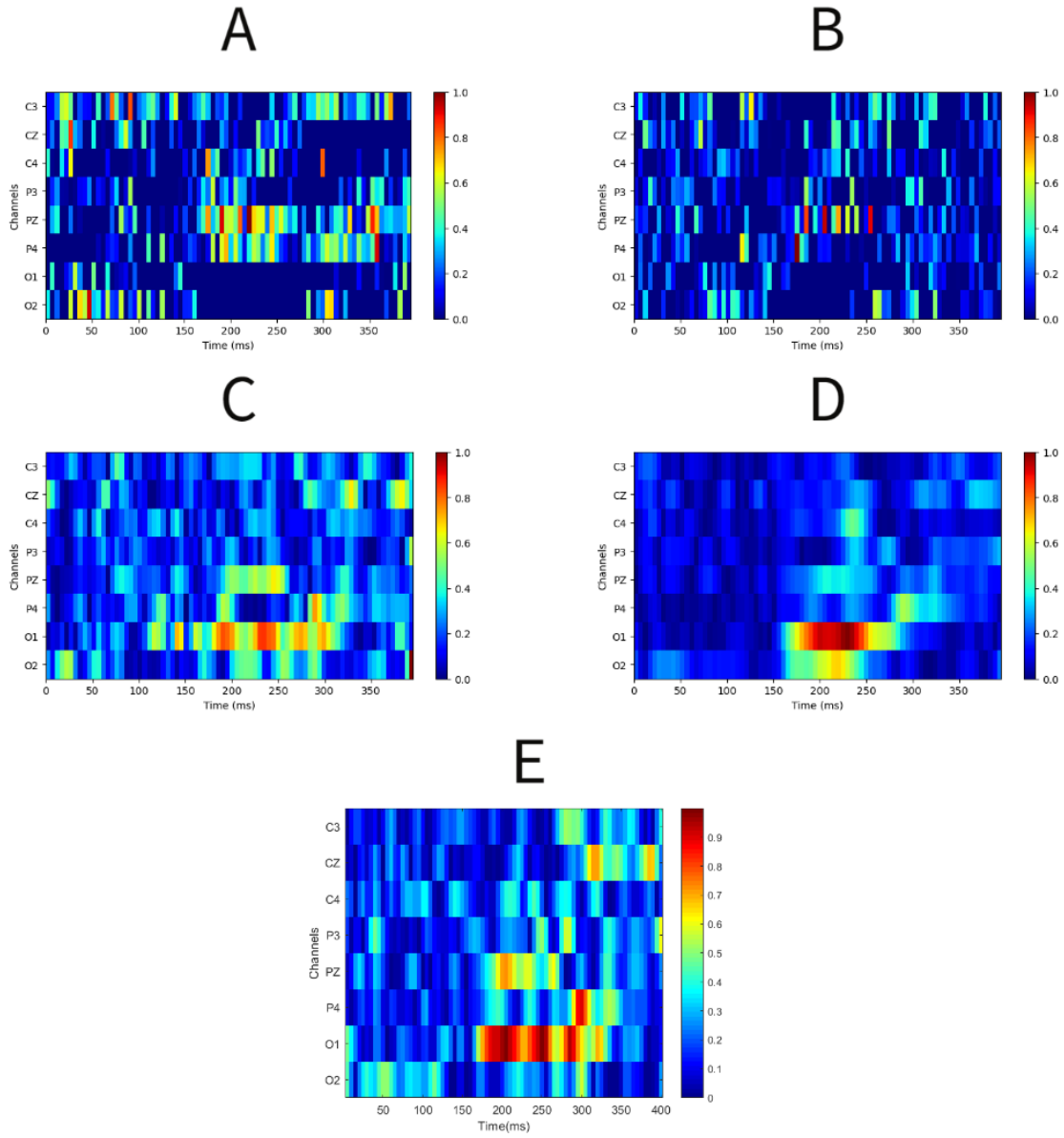

Figure 4.2: Coefficients or weights for each algorithm after training. (A) FCNN activation maximization image. (B) CNN activation maximization image. The other three images show the coefficients for (C) SVM, (D) LR and (E) CAB. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent features of interest.

the other feature maps and highlight regions that aren't known to be associated with the P300 potential and are absent from the majority of other feature maps, including some channels such as C3, Cz and O2 below 100ms. This could be explained by non-optimal parameter selection for the activation maximization algorithm or it could be an indication

of overfitting which is also evident by the fact that training accuracies were generally greater than validation/test accuracies.

4.2.2 Leave-one-subject out

For inter-subject testing, all sessions for each subject are combined and an 8-fold cross validation is performed where each fold consists of all the data for one subject. It is important to note that the chance level for this dataset is not 50% since the ratio of non-oddballs to oddballs is 6:1. Therefore the chance level is 6/7 or 85.71%. Due to the smaller sample size of 8 for each algorithm in the inter-subject cross-validation scheme, the power of the statistical tests is reduced and the only significance was found between SVM and KNN with SVM having the higher accuracy scores. However there are still a couple of observations to be made including that most of the KNN scores are very close to chance level and with the exception of subject 1, CAB evaluated all the data as non-oddballs making the accuracies equal to chance level.

Table 4.7: Leave-one-subject out accuracies, where the data from all subjects except for the test subject were combined for training of the algorithms. FCNN and CNN were trained 10 times with random initialization and their mean scores and standard deviation are reported. Values in bold represent the highest scores for each test subject.

Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	91.63	88.08	90.71 ± 0.50	89.84 ± 0.99	**91.79**	85.54
2	2	85.55	85.68	85.61 ± 0.14	**85.72 ± 0.19**	85.34	85.34
3	3	85.41	90.27	90.30 ± 0.56	89.60 ± 0.68	**91.32**	86.91
4	1	85.65	89.03	88.37 ± 0.85	84.44 ± 1.33	**89.52**	86.61
5	6	85.67	**88.31**	87.90 ± 0.56	87.86 ± 1.10	88.04	85.82
6	12	85.57	85.09	**90.47 ± 0.37**	90.23 ± 1.25	89.31	86.96
7	5	85.44	89.36	89.39 ± 0.27	88.19 ± 0.66	**89.42**	84.70
8	2	85.62	**85.75**	85.71 ± 0.16	85.16 ± 0.69	**85.75**	85.27

The majority of CAB recall scores are 0 as there are no true positives with the exception of subject 1 as the test dataset. All algorithms with the exception of KNN have statistically

higher recall scores than CAB. LR additionally has statistically higher scores than KNN.

Table 4.8: Leave-one-subject out recall scores, where the data from all subjects except for the test subject were combined for training of the algorithms. Values in bold represent the highest scores for each test subject.

Test Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	62.74	**72.24**	62.98	55.08	63.24	7.48
2	2	0.00	**5.69**	2.75	3.27	1.42	2.37
3	3	0.00	54.83	56.85	57.54	**58.57**	15.26
4	1	0.00	56.18	49.78	26.52	**57.30**	11.24
5	6	0.00	**20.14**	16.70	18.30	16.87	2.24
6	12	0.00	**81.72**	73.81	66.19	81.64	17.16
7	5	0.00	34.27	34.57	30.86	**35.58**	9.74
8	2	0.00	4.76	3.76	**6.48**	2.86	3.33

Again the precision scores for CAB with the exception of subject 1 are 0 as there are no true positives. FCNN and SVM scores are statistically higher than CAB scores.

Table 4.9: Leave-one-subject out precision scores, where the data from all subjects except for the test subject were combined for training of the algorithms. Values in bold represent the highest scores for each test subject.

Test Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	75.11	56.83	69.71	68.29	**75.84**	49.17
2	2	0.00	54.55	54.55	**63.64**	33.33	38.46
3	3	0.00	71.84	70.82	66.79	**76.42**	75.38
4	1	0.00	63.29	61.97	43.64	65.38	**71.43**
5	6	0.00	92.13	93.27	86.18	**98.00**	65.00
6	12	0.00	49.02	64.96	66.11	59.44	**69.58**
7	5	0.00	82.43	**82.59**	72.05	81.20	39.69
8	2	0.00	55.56	53.33	41.18	**60.00**	36.84

With the exception of subject 1, CAB's F1 scores are 0 due to the recall and precision scores being 0. LR, FCNN and SVM have statistically higher F1 scores than CAB.

Table 4.10: Leave-one-subject out F1 scores, where the data from all subjects except for the test subject were combined for training of the algorithms. Values in bold represent the highest scores for each test subject.

Test Subject	N_sessions	CAB	LR	FCNN	CNN	SVM	KNN
1	8	68.37	63.61	66.17	60.98	**68.97**	12.98
2	2	0.00	**10.31**	5.24	6.22	2.72	4.46
3	3	0.00	62.19	63.07	61.82	**66.32**	25.38
4	1	0.00	59.52	55.21	32.99	**61.08**	19.42
5	6	0.00	**33.05**	28.33	30.19	28.78	4.33
6	12	0.00	61.28	**69.10**	66.15	68.79	27.53
7	5	0.00	48.41	48.74	43.21	**49.48**	15.64
8	2	0.00	8.77	7.02	**11.20**	5.46	6.11

4.3 Discussion

From looking at the accuracy scores for the 10x10 session cross-validations we can deduce that CAB and FCNN were the two best performing algorithms from the statistical tests and given that they have the highest average scores for 7 out of the 8 subjects. KNN didn't perform very well which can be expected since it's performance usually deteriorates as the dimensions of the input increases since data points become more equidistant to each other in high dimensional space. LR had higher recall scores than all other algorithms meaning it had a better chance of classifying an oddball correctly. This result tells us that LR weighs correct oddball classification more than the other algorithms. Since the ratio of oddballs to non-oddballs in each session's data is 1:6, most algorithms are biased towards classifying non-oddballs correctly since they appear in higher proportion. One way to remedy this behavior is to modify the class weights to be inversely proportional to each other so in this case giving oddballs more weight so that misclassifying one oddball trial comes at the same cost as misclassifying 6 non-oddball trials. However this was found to reduce the overall accuracy as it led to more false positives which is undesirable for this application since selecting an incorrect letter comes at the cost of having to erase and reselect the letter whereas false

negatives correspond to not selecting a letter at all which just slows down the ITR and is therefore less costly. CAB and FCNN also had higher precision scores than the other algorithms meaning they had lower false positives with respect to all positive predictions. This helps explain why they also have the highest average accuracies as they predicted a higher proportion of non-oddballs correctly which are more prevalent than oddballs in comparison to the other algorithms. F1 score looks at the balance between precision and recall and is high when both are high or low when one of them is low. CAB and LR had the highest F1 scores with LR on average having the highest F1 scores which follows from it having the highest recall scores and giving more importance to the oddball class than the other algorithms.

Previous studies have shown that Bayesian linear discriminant analysis (BLDA) outperforms other algorithms including SVM [84, 85] and NNs [85] for single subject P300 tasks. Other studies found that weren't any significant differences in accuracy between different ML algorithms including BLDA and SVM [86]. Although we found that CAB and FCNN statistically outperformed the other algorithms in terms of accuracy, the majority of the algorithms excluding KNN were within 1-2% of each other on average for each subject. Therefore our recommendation would be to use the CAB method as it has lower computational complexity and runs faster than FCNN especially during training time.

While NNs have been popularized in recent times and seem to be outperforming most other machine learning algorithms in a wide variety of tasks, they do not seem to particularly stand out from the other algorithms for this dataset. There are a couple of possible explanations for this including the size of the dataset, the requirement for hyperparameter tuning for NNs or the complexity of the task. Firstly, NNs typically require large datasets with lots of samples to perform optimally. However for each session we approximately have 750 samples of data which is a relatively small sample size for a data feature matrix of size 8x80 or 640 dimensional vector. Next, there are multiple hyperparameters that need to be tuned to

maximize the performance of NNs. We tested about 50 different hyperparameter sets for the 10x10 cross validation model where each hyperparameter set was tested on 4 different sessions and repeated 3 times for every session shuffling the data for each repetition. However there are lots of more hyperparameter variations that could be tested and different NN types including RNNs, RBMs, Transformers, etc. which may increase the performance. Finally the task may not be too complex and therefore the majority of algorithms are able to achieve relatively equal levels of performance and the error may be due to mislabeled data, for instance a false positive caused by a character flashing near the gaze of the subject.

A slight surprising fact is that the FCNN algorithm outperformed the CNN algorithm as CNNs are usually thought to be more powerful than simple fully connected networks. An explanation of this outcome might be that CNNs typically work well with image like inputs. While our input may appear to be like a 2D image, stacking the channels on top of each other is not spatially accurate to where they are located over the subject's head. A more promising input format for CNNs may be spectrograms of frequency by time or 3D images of EEG values overlaid on a 2D spatial layout of the channels for each time sample, e.g. [87], or frequency band, e.g. [88].

For the inter-subject testing, all algorithms have chance level accuracies for subjects 2 and 8, likely due to a combination of those subjects having different feature patterns to the other 6 subjects as well as having low cross-validation scores for their sessions. Out of the remaining 6 subjects, SVM has the highest accuracies for 4 subjects. KNN had chance level or close to chance level accuracies for all subjects again showing KNN's disadvantage when it comes to high dimensional data. CAB had chance level accuracies for all subjects except for subject 1. Although the algorithm excelled during cross-validations of each session it seems unable to find common feature patterns across sessions and subjects. However it is to be noted that z-score normalization was performed on the combined training data for leave-one-subject out which may not be optimal for performance due to alterations between sessions including

contact impedance values between the scalp and electrodes for instance. A more fitting normalization scheme may be to z-score each session's data separately during training and normalize each session in testing using a small subset of it's data.

Studies that performed cross-subject P300 testing with large datasets found that NNs outperformed other algorithms. For instance a study showed that a CNN outperformed other ML algorithms including a Bayesian classifier, LDA, KNN and SVM on an auditory P300 task with a dataset of 22 subjects [89]. FCNN was within about 1% of SVM for accuracy for all subjects however we recommend the latter for cross-subject P300 classification models since SVM is less computationally complex and requires less hyperparameter tuning in comparison with NNs.

We can see from Figure 4.2 that a lot of the algorithms learn to prioritize similar features to discriminate between oddball and non-oddball trials. For instance, all algorithms have high coefficient values for Pz around 200-250 ms post stimulus. There are some discrepancies however between some of the algorithms including SVM, LR and CAB (Figure 4.2(C),(D),(E)) picking up on an important region at O1 between 200-300 ms which isn't observed in the NN feature matrices (Figure 4.2(A),(B)). This is perhaps due to the NNs not identifying this region as being discriminative or more likely that the activation maximization algorithm wasn't optimal in identifying all the regions of interest. Most algorithms additionally highlighted the region of O2 between 0-50 ms which is odd given that visual processing usually takes around 100-150 ms in humans [90]. This is likely explained by residual P300 potential following an oddball trial which may perhaps be useful in classifying a non-oddball trial since the sets of characters flashed were unique during each traversal of the grid. LR seems to be the least affected by this phenomenon since it has the lowest coefficient values in this region in comparison to the other algorithms. Additionally, there are other regions highlighted for the NN algorithms under 100 ms post stimulus mainly in C3 and Cz which are not known regions in the brain that partake in visual processing so these are probably artifacts from

overfitting or the activation maximization algorithm. Overfitting can perhaps be remedied by increasing the regularization of the model including a higher kernel regularization term or dropout rate or by having a larger training dataset.

Chapter 5

A comparative analysis on sensorimotor rhythms

5.1 Methods

5.1.1 Dataset Description

This dataset consisted of a foot dorsiflexion task performed by 5 able-bodied individuals. Sessions were comprised of alterations between dorsiflexion of the right or left foot and idling. The subject was prompted by onscreen commands to switch between the two conditions and each dorsiflexion or idling period lasted for $\sim$ 6s. The session lasted for 20 mins producing 100 epochs of movement and idling each. Data was collected from an EEG cap with 64 electrodes arranged according to the 10-20 international standard. The signals were amplified, bandpassed between 0.01 and 50 Hz and digitized at a sampling rate of 256 Hz using two 32-channel NeXus amplifiers. Two electrogoniometers were attached to the anterior surface of the subject's ankles to measure foot dorsiflexion. The goniometer data was used to seg-

ment and label movement and idling epochs. More details on the experimental procedure and data collection can be found in [91].

5.1.2 Pre-processing

Several channels do not contain relevant motor control signals and therefore add noise to the dataset. These channels were excluded and data from the central electrodes closest to the leg motor region was kept. Following the removal of unwanted channels, a subset of 15 channels remained (Cz, C1, C2, CPz, CP1, CP2, FCz, FC1, FC2, Fz, F1, F2, Pz, P1, P2). Data was bandpassed between 1-35 Hz to remove low frequency drift and high frequencies that do not contain relevant motor rhythm features.

Three feature extraction methods were trained to extract relevant features and reduce the dimension of the data. These were ICA, Information Discriminant Analysis (IDA) [92] and CSP. ICA aims to decompose the signal into independent non-Gaussian sources. IDA is a discriminant feature extraction method that reduces the dimensionality of the data while maximizing class separability. CSP is a method used to separate a signal into additive components by maximizing the variance between two conditions or classes. Using each method, the top 5 components were selected to replace the channel dimension. The top components were determined by segmenting the components into trials of dorsiflexion or idling. The PSD of the trials was then computed and the signal-to-noise ratio (SNR) between dorsiflexion and idling trials was calculated as

$$SNR = 2\,\frac{(\mu_D - \mu_I)^2}{\sigma_D^2 + \sigma_I^2}$$

where $\mu_D, \mu_I, \sigma_D, \sigma_I$ are the means and standard deviations of the dorsiflexion and idling trials respectively. Finally the maximum SNR of each component was computed, and the 5 components with the highest maximum SNR scores were selected. A more detailed expla-

nation of this method is presented in Appendix B.

For the cross-validation analysis, trials of 4s were segmented, detrended and labeled using the goniometer data. 0.5s from each side of class transition boundaries were excluded to avoid regions of early anticipation and reaction times of the subjects. The signal power of each trial was then computed in 2 Hz bins from 6-30 Hz and converted to log scale. Cross-validations were performed on the 4 differently processed data based on feature extraction method (None, ICA, IDA, CSP).

For the pseudo-online analysis, overlapping window trials of 750 ms with an overlap of 500 ms were segmented and detrended. The signal powers of each trial were also computed in 2 Hz bins from 6-30 Hz and converted to log scale to produce the features to be used by the ML algorithms.

5.1.3 Model parameters

A subset of the dataset (3 out of 11 sessions) was used as a validation set to optimize the parameters/hyperparameters used by the different algorithms.

CAB

The components retained during the CPCA procedure are ones whose corresponding eigenvalue is greater than the mean of all eigenvalues. The size of the feature vector resulting from the AIDA procedure is chosen to be 1. A linear PDF was used for the classifier and the priors were empirical representing the probability of observing each class in the training data.

LR

The optimizer algorithm with the best performance was the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm which is a quasi-Newton method that performs a similar function to the gradient descent algorithm.

NN

A fully connected neural network (FCNN) was developed and used for cross-validation testing on the data. A LSTM model was also developed for testing the data in a pseudo online fashion where data samples were decoded in consecutive temporal order. The hyperparameters that were tuned included the learning rate, weight decay, batch size, number of epochs, number of fully connected layers, number of hidden units in each layer, batch normalization, dropout rate, kernel regularization, and number of LSTM units and layers for the LSTM model.

The FCNN model consists of 1 hidden layer followed by the output layer. The architecture is outlined in Table 5.1. The input matrix of shape (15,12) is flattened to a vector of shape (180,1). The vector is connected to a fully connected layer also known as a Dense layer with 128 nodes. The outputs of the Dense layer are passed on to a batch normalization layer. A ReLU activation is then applied to the output vector and a dropout layer is used to reduce over fitting. The activations are then passed to a Dense layer with 2 nodes with a softmax activation that predicts the likelihood of the sample belonging to each class. The same model was also used for the data preprocessed using the feature extraction methods with the only exception of the input shape being (5,12) instead of (15,12).

The LSTM model consists of 1 LSTM layer followed by two Dense layers. The architecture is outlined in Table 5.2. The input matrix of shape (3,5,12), corresponding to the number of time-steps, CSP components and frequency bins respectively, is flattened to a matrix of

Table 5.1: Fully connected neural network architecture for Movement dataset.

Layer	Layer type	# nodes	# parameters	Output dimension	Activation function
1	Input			(15,12)	
	Flatten			(180,1)	
2	Dense	128	23,168	(128,1)	
	BatchNorm		512	(128,1)	
	Activation			(128,1)	ReLU
	Dropout			(128,1)	
Classifier	Dense	2	258	(2,1)	
	Activation			(2,1)	Softmax

shape (3,60). The matrix is passed on to an LSTM layer with 34 units. The outputs are batch normalized and activated using a Tanh function. The activations are then passed to two Dense layers with 16 and 8 nodes. The outputs of each Dense layer are passed onto batch normalization, ReLU activation and dropout layers. The final activations are then passed to a 2 node Dense layer with a softmax activation that classifies the data.

Table 5.2: LSTM neural network architecture for Movement dataset.

Layer	Layer type	# nodes	# parameters	Output dimension	Activation function
1	Input			(3,5,12)	
	Flatten			(3,60)	
2	LSTM	34	12,920	(34)	
	BatchNorm		136	(34)	
	Activation			(34)	Tanh
3	Dense	16	560	(16)	
	BatchNorm		64	(16)	
	Activation			(16)	ReLU
	Dropout			(16)	
4	Dense	8	136	(8)	
	BatchNorm		32	(8)	
	Activation			(8)	ReLU
	Dropout			(8)	
Classifier	Dense	2	18	(2)	
	Activation			(2)	Softmax

SVM

After comparing the results on the validation set using different kernels, the RBF kernel was found to perform the best and is used for the SVM model. The value of the regularization parameter C was selected as 10 and the RBF kernel coefficient gamma was selected as 0.001.

KNN

The value of k indicating the number of neighbors was selected to be 9 and all neighbors were weighted equally.

5.1.4 Pseudo-online analysis

A pseudo-online procedure was developed and tested to simulate real-time decoding using the algorithms. Two schemes were developed, the first being a window probability averaging method used for the CAB, LR, FCNN, SVM and KNN algorithms and the second utilized the recurrent nature of a RNN model using LSTM units.

Window probability averaging

Each of the algorithms used output probabilities of each data sample belonging to each class. These probabilities can be averaged for multiple consecutive windows of data in order to increase the chance of predicting the correct class. This averaging method is outlined in this section and is summarized in Figure 5.1. Each session's data is split into train and test sets with an 80/20 % split respectively without shuffling in order to maintain the temporal order of the data. 10% of the training data is used to calibrate thresholds for the class probabilities. A model is trained using the training data and initially tested on the calibration data. The

dorsiflexion probabilities of each 3 consecutive windows are averaged and a class prediction is made using two thresholds. If the last prediction made by the model, also known as the state, is idle, then the state will remain in idle unless the averaged probabilities are greater than the upper threshold at which point the state will transition to dorsiflexion. Similarly, if the current state is dorsiflexion, the state will remain in dorsiflexion unless the averaged probabilities are less than the lower threshold at which point the state will transition to idle. The next window is then combined with the last 2 windows from the previous cluster and the process is repeated till the last data window is reached.

Finally, the predicted states and the true states, determined by the label of the final window in each 3 window cluster, are compared using different temporal shifts to determine the highest accuracy accounting for temporal misalignment between the true and predicted states. This misalignment between the true and predicted states arises due to the fact that we are combining 3 windows during prediction that may contain outdated information in the earlier windows. Therefore the predicted state is expected to lag the true state which is most evident at class transitions. The accuracies are determined for different combinations of lower and upper thresholds to determine the threshold combination with the highest accuracy for the calibration data. The lower threshold is incremented by 0.05 from 0 to 0.95 while the upper threshold is incremented for each value of the lower threshold by 0.05 from the lower threshold to 0.95. Then the model is evaluated on the test data with the same process as the calibration data using the optimal thresholds. The lag between the predicted and true states that produces the highest accuracy is recorded.

LSTM

An LSTM model is also trained and evaluated on the dataset. The same 80/20 % split for training and test data is used for each session. Again each 3 consecutive windows are combined to produce a sample. For each sample, the LSTM computes and updates the

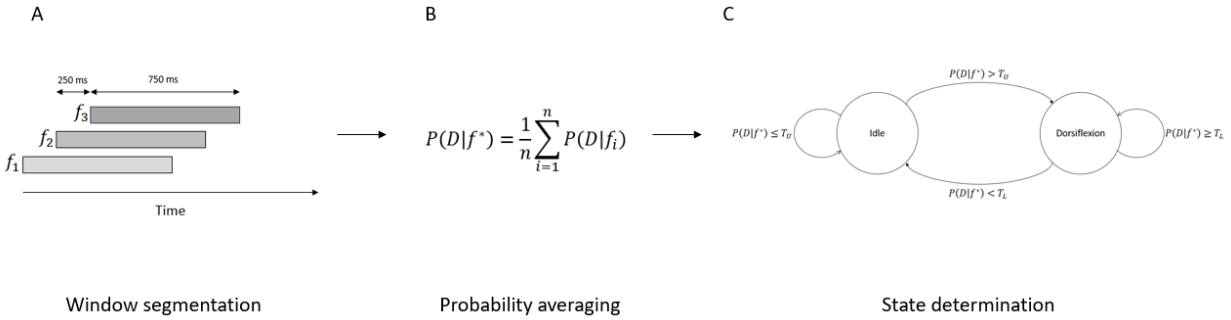

Figure 5.1: Description of the window probability averaging method. (A) shows the window segmentation procedure displaying three overlapping feature vectors f_1, f_2 & f_3. (B) is the probability averaging equation that averages the dorsiflexion probabilities over n windows where n is 3 in this case. (C) shows how the current state is determined where $P(D|f^*)$ is the averaged dorsiflexion probability and T_L & T_U are the lower and upper thresholds respectively.

hidden state of the model for each of the 3 windows and produces one output at the end of the sample. This method is used during training and testing using the respective splits of the data. Finally, the lag between the predicted and true states for the test data that produces the highest accuracy is determined.

5.2 Results

5.2.1 Within subject cross-validation

The results for 10x10 cross-validations for each session using Channel by Frequency data can be found in Table 5.3. The Friedman test between the 5 ML algorithms is significant with a p-value of 0.0082 however the only significant result from the Nemenyi test is that SVM has higher accuracies than KNN (p-value = 0.0032). LR and SVM appear to be the best performing algorithms with each having the highest accuracy for 4 sessions.

Table 5.3: Accuracies for 10x10 cross-validation for each session. Bold values represent the highest accuracy for that session.

Subject	Session number	CAB	LR	NN	SVM	KNN
1	1	**94.43 ± 0.93**	94.18 ± 0.83	93.93 ± 1.17	94.38 ± 0.67	89.05 ± 1.44
	2	93.58 ± 0.76	**96.17 ± 0.45**	93.93 ± 1.06	94.93 ± 0.58	89.80 ± 0.84
2	1	94.72 ± 0.69	**97.13 ± 0.47**	95.95 ± 0.84	96.92 ± 0.40	94.26 ± 0.50
	2	96.08 ± 0.49	**96.36 ± 0.61**	94.83 ± 0.78	96.29 ± 0.55	92.94 ± 0.38
3	1	86.81 ± 0.90	85.71 ± 1.02	86.67 ± 0.77	**86.86 ± 1.15**	82.00 ± 0.87
	2	87.83 ± 0.97	91.06 ± 0.82	90.00 ± 0.90	**91.36 ± 0.86**	74.34 ± 1.33
	3	94.46 ± 0.96	**95.25 ± 0.86**	92.77 ± 1.40	**95.25 ± 1.07**	87.52 ± 0.79
4	1	**93.18 ± 0.41**	91.39 ± 0.92	92.79 ± 0.84	92.89 ± 0.63	92.89 ± 0.67
	2	79.65 ± 1.03	76.80 ± 1.38	80.85 ± 1.25	80.15 ± 0.92	**81.00 ± 1.24**
5	1	97.31 ± 0.71	97.56 ± 0.47	**97.71 ± 0.64**	96.67 ± 0.59	91.14 ± 0.73
	2	94.93 ± 0.61	94.78 ± 0.64	94.98 ± 0.68	**95.77 ± 0.60**	92.04 ± 0.44

Table 5.4: Accuracies for 10x10 cross-validation for each session using ICA preprocessed data. Bold values represent the highest accuracy for that session.

Subject	Session number	CAB	LR	NN	SVM	KNN
1	1	96.92 ± 0.46	95.87 ± 0.80	96.87 ± 0.63	**97.56 ± 0.68**	95.67 ± 0.55
	2	97.56 ± 0.28	**97.61 ± 0.30**	97.41 ± 0.58	97.46 ± 0.35	95.37 ± 0.32
2	1	92.05 ± 1.19	92.72 ± 0.85	93.08 ± 0.77	**94.00 ± 0.46**	92.77 ± 0.74
	2	**97.48 ± 0.59**	96.57 ± 1.01	96.15 ± 0.72	96.78 ± 0.64	97.34 ± 0.42
3	1	88.24 ± 0.87	84.43 ± 0.71	88.10 ± 1.66	**89.24 ± 0.71**	88.71 ± 0.48
	2	94.65 ± 0.86	94.80 ± 0.60	96.46 ± 0.68	**96.92 ± 0.15**	96.01 ± 0.66
	3	95.45 ± 0.69	96.63 ± 0.66	94.65 ± 1.61	**97.03 ± 0.63**	90.20 ± 0.69
4	1	95.22 ± 0.48	90.10 ± 0.93	94.23 ± 0.95	95.12 ± 0.20	**95.57 ± 0.57**
	2	**81.85 ± 0.85**	76.20 ± 1.38	78.70 ± 2.52	80.70 ± 1.78	80.65 ± 0.55
5	1	97.56 ± 1.03	**98.01 ± 0.74**	97.31 ± 0.24	97.51 ± 0.63	96.32 ± 0.24
	2	**96.17 ± 0.47**	91.69 ± 0.83	93.83 ± 0.92	93.38 ± 0.83	93.08 ± 0.27

According to the Friedman test, there is no significant difference between accuracies for each algorithm when the data is preprocessed using ICA, IDA or CSP and the top 5 components selected to replace the channel dimension as outlined in the methods section. When comparing the raw, ICA, IDA and CSP preprocessed data for each algorithm, the accuracies using CSP preprocessed data are significantly higher than accuracies using raw data except for LR according to the Nemenyi test.

Figure 5.2 shows the coefficients or weights learned by the algorithms on data from a single

Table 5.5: Accuracies for 10x10 cross-validation for each session using IDA preprocessed data. Bold values represent the highest accuracy for that session.

Subject	Session number	CAB	LR	NN	SVM	KNN
1	1	97.51 ± 0.33	97.36 ± 0.67	**98.06 ± 0.61**	97.46 ± 0.57	96.57 ± 0.41
	2	96.42 ± 0.56	97.16 ± 0.23	97.61 ± 0.58	**98.46 ± 0.15**	95.67 ± 0.32
2	1	94.62 ± 0.69	**97.85 ± 0.50**	97.28 ± 0.83	97.08 ± 0.61	95.85 ± 0.48
	2	94.90 ± 0.66	95.80 ± 0.63	95.17 ± 0.73	**96.29 ± 0.70**	95.17 ± 0.49
3	1	**90.81 ± 0.60**	78.48 ± 1.41	85.00 ± 1.42	86.57 ± 0.73	86.52 ± 1.07
	2	92.47 ± 1.27	95.35 ± 0.74	**97.12 ± 0.78**	96.97 ± 0.32	96.41 ± 0.57
	3	**95.74 ± 0.67**	95.54 ± 0.80	93.66 ± 2.04	94.46 ± 1.67	93.17 ± 1.12
4	1	93.88 ± 0.47	91.89 ± 0.71	93.48 ± 0.65	**95.12 ± 0.66**	94.38 ± 0.23
	2	**87.75 ± 1.09**	80.75 ± 1.68	81.20 ± 1.75	83.45 ± 0.61	81.50 ± 0.74
5	1	95.77 ± 0.42	96.02 ± 0.44	**96.62 ± 0.86**	95.12 ± 0.43	93.78 ± 0.56
	2	93.08 ± 0.86	92.69 ± 0.55	91.99 ± 0.85	**93.68 ± 0.39**	90.75 ± 0.55

Table 5.6: Accuracies for 10x10 cross-validation for each session using CSP preprocessed data. Bold values represent the highest accuracy for that session.

Subject	Session number	CAB	LR	NN	SVM	KNN
1	1	**97.46 ± 0.44**	95.52 ± 0.50	97.21 ± 0.55	97.41 ± 0.37	97.06 ± 0.47
	2	94.53 ± 0.52	93.43 ± 0.70	94.93 ± 0.76	**95.22 ± 0.24**	94.88 ± 0.59
2	1	96.56 ± 0.42	97.85 ± 0.38	97.74 ± 0.41	**98.26 ± 0.34**	95.85 ± 0.28
	2	96.01 ± 1.10	94.76 ± 0.72	95.38 ± 1.05	94.20 ± 0.45	**97.27 ± 0.21**
3	1	89.71 ± 0.68	85.71 ± 0.74	89.62 ± 1.02	**91.86 ± 0.58**	89.76 ± 0.38
	2	95.00 ± 0.84	96.46 ± 0.32	97.58 ± 0.38	**98.13 ± 0.32**	97.63 ± 0.39
	3	95.94 ± 0.73	**96.63 ± 0.79**	95.74 ± 1.00	96.53 ± 1.11	93.17 ± 1.43
4	1	94.88 ± 0.74	91.79 ± 0.75	95.27 ± 0.68	**96.52 ± 0.22**	95.82 ± 0.40
	2	84.85 ± 1.08	77.80 ± 0.71	82.65 ± 1.55	84.85 ± 0.98	**85.20 ± 1.08**
5	1	97.11 ± 0.46	97.76 ± 0.68	97.36 ± 0.55	**98.01 ± 0.44**	95.82 ± 0.33
	2	**96.77 ± 0.54**	96.62 ± 0.66	95.22 ± 0.78	96.42 ± 0.49	94.23 ± 0.51

session in the 16-18 Hz band displayed on topoplots. There are no corresponding values for KNN again as it is a non-parametric algorithm. Activation maximization was used to obtain the values for the FCNN algorithm. The coefficients are high at Cz for all algorithms. P1 also has high values for SVM, LR and CAB algorithms and P2 and C2 have high values for the FCNN algorithm. The coefficients of the remaining frequency bands for each algorithm can be found in Appendix C.

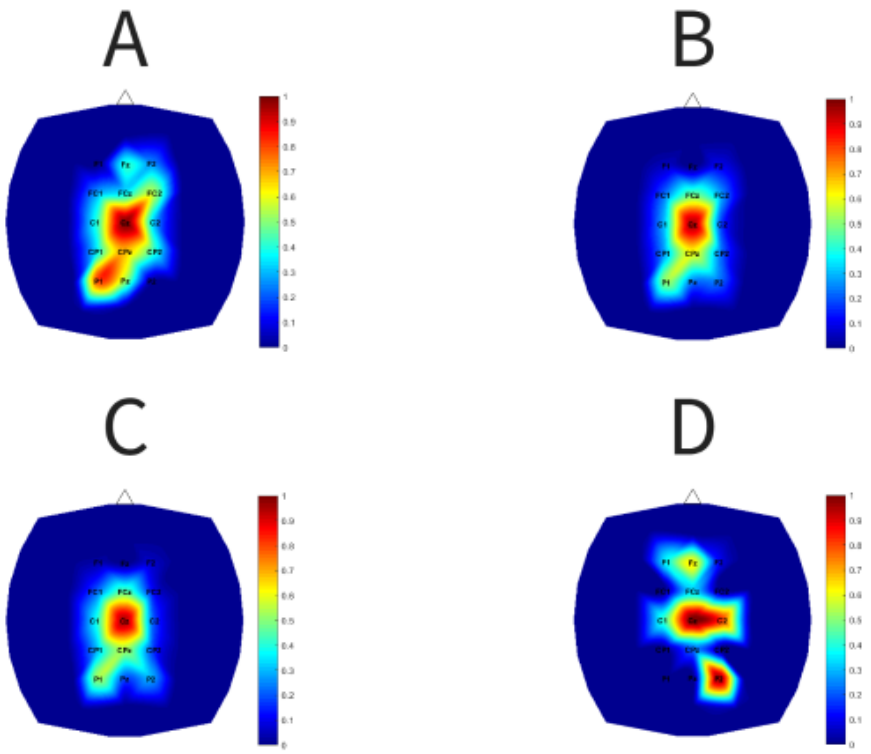

Figure 5.2: Topoplots of feature coefficients in the 16-18 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

5.2.2 Pseudo Online analysis

The CSP preprocessed data is used for pseudo online analysis as that is the best performing data type for the 10x10 session cross-validations.

The Friedman test did not report any significant differences between the accuracies using the different algorithms. However the lag using the LSTM method is significantly lower than the lag using the other algorithms except for FCNN according to the Nemenyi post hoc test although the lag is lower for LSTM compared with FCNN for every session.

Table 5.7: Accuracies and lags for pseudo online analysis. All algorithms are tested in the same manner averaging the probabilities of 3 windows of data post decoding except for LSTM which returns a decision for every 3 windows of data. Bold values indicate the highest accuracies and smallest lag values for each session.

Subject	Session number	CAB		LR		FCNN		SVM		KNN		LSTM	
		Accuracy (%)	Lag (s)	Accuracy (%)	Lag (s)	Accuracy (%)	Lag (s)	Accuracy (%)	Lag (s)	Accuracy (%)	Lag (s)	Accuracy (%)	Lag (s)
1	1	80.74	1.75	**84.19**	1.0	80.51	0.75	83.47	1.0	79.69	0.75	76.36	**0.25**
	2	82.46	1.00	77.41	1.25	82.58	0.75	**83.59**	1.0	76.02	0.75	79.32	**0.5**
2	1	**84.21**	1.00	83.68	1.25	81.13	1.0	83.28	1.0	80.82	1.0	83.62	**0.5**
	2	82.58	0.50	92.21	0.5	**92.93**	0.5	92.73	0.5	89.34	0.5	88.74	**0.25**
3	1	83.86	1.00	**84.09**	1.0	70.45	1.0	74.43	1.0	73.86	1.0	83.11	**0.5**
	2	69.40	1.25	**93.03**	0.75	91.29	0.75	92.32	0.75	84.32	0.75	92.84	**0.25**
	3	72.86	1.75	82.76	1.0	80.93	1.0	**84.79**	1.0	61.84	1.75	84.24	**0.5**
4	1	71.81	-0.50	82.65	-0.5	84.37	-0.5	**85.32**	-0.75	82.25	-0.75	84.05	**-1.0**
	2	65.84	-2.25	58.44	-1.5	63.07	-1.5	57.86	-1.25	76.76	**-2.5**	**82.33**	**-2.5**
5	1	80.21	1.25	**80.39**	1.5	77.48	**0.75**	78.69	1.0	74.67	1.25	79.53	**0.75**
	2	78.15	1.00	75.87	1.25	**86.97**	1.0	83.49	1.0	79.67	1.25	79.02	**0.5**

5.3 Discussion

All algorithms have relatively similar accuracies for each session's cross-validation except for KNN. SVM and LR seem to be the top 2 algorithms having the highest average cross-validation accuracies for 4 sessions each. Although KNN has the lowest average cross-validation accuracies across sessions, it performs better than it did for the P300 dataset in comparison to the other algorithms. This is likely due to the smaller feature vector for this dataset of 180 dimensions (15x12 matrix) compared with the 640 dimensional vector for the P300 dataset. This aligns with the guideline that KNN's performance deteriorates as the data feature dimensions increase. The 3 preprocessing techniques ICA, IDA and CSP improve the cross-validation accuracies for all algorithms. CSP provides the greatest boost in performance when comparing the 3 preprocessing methods. This is most evident for the KNN algorithm which fares pretty well compared to the other algorithms for the CSP pre-processed data. This is again likely due to the reduced data feature dimensions of 5x12 = 60 which is favorable for KNN. Although there was no statistical difference between the algorithms using the CSP preprocessed data, SVM had the highest accuracies for 6 out of the 11 sessions.

Previous studies comparing different ML algorithms for lower limb sensorimotor rhythm

classification have reported different results. Some have reported LDA outperforming other algorithms including SVM [93], KNN [93, 94], Naïve Bayes [93, 94] and LR [95] while others have reported KNN outperforming LDA [96, 97] and SVM [96, 97]. Our recommendation however would be to use SVM for sensorimotor rhythm classification since it was the best performing algorithm for over half of the sessions using the CSP preprocessed data.

For the pseudo-online analysis, there was no significant difference in accuracies between the different algorithms. In terms of lag however, LSTM had the smallest values in comparison to the other algorithms. While the probabilities of each data window were simply averaged for the other algorithms, LSTM has weights that are adapted to determine the degree of influence given to past vs current inputs. Giving more weight to the current input can allow the model to detect changes in state quicker than if it relied more on the previous inputs. However this comes with the cost of reducing the accuracy as previous inputs provide more context and insight on the current state. LSTMs optimize the weights associated with current and previous inputs to maximize the accuracy and minimize the lag. Two sessions had negative lags using all algorithms, both corresponding to subject 4. The most likely explanation for this is that the subject was anticipating the cues to switch states since each move/idle epoch was roughly of equal length of 6s. LR and SVM have the highest accuracies for 4 and 3 sessions respectively making them the top two performing algorithms in terms of accuracy.

All algorithms prioritized similar features as can be seen in Figure 5.2. They all had high coefficient values for Cz in the 16-18 Hz band which coincides with the leg region on the motor cortex and beta wave oscillations which are known to be involved with movement execution. There are a few differences between the algorithms in terms of features of interest including P1 @ 16-18 Hz for all algorithms except for FCNN and P2 and Fz for FCNN which are regions that may in fact be discriminative between idling and foot dorsiflexion since the parietal and frontal lobes are known to be involved with voluntary movement [98].

Chapter 6

Conclusion and Future Work

All tested algorithms have their own strengths and may be applicable for different BCI applications, however there are some recommendations that can be drawn from our findings. In general it is best to start with algorithms with low computational complexities as they use the least resources and can be trained the quickest. CAB and LR both performed very well for both types of datasets tested and have lower computational complexities than the other algorithms with the exception of KNN. KNN in general is not recommended for most BCI applications since BCI data is usually high dimensional. However it may perform competitively with low-dimensional preprocessed BCI data as observed with the CSP preprocessed movement data. For P300, CAB was one of the best performing algorithms for within-session cross-validations and is therefore recommended for that datatype. LR was one of the best performing algorithms for the sensorimotor data and is recommended for that datatype. SVM was the best performing algorithm for leave-one-subject out cross validations for the P300 data and is therefore recommended for inter-subject models trained and tested on different subjects. There was no significant difference in accuracy between the algorithms for the movement pseudo-online testing. LSTM however had the lowest lag and therefore seems to be most promising algorithm for real-time movement decoding. However, true on-

line experiments are required to validate this. On average, SVM was the best performing algorithm across all datasets examined in this work and is therefore our recommendation for other BCI tasks and datasets not investigated here. These recommendations are only general guidelines however and may not apply for all types of applications and further testing and comparison between algorithms may be required for specific projects. Also as further advancements are realized for different ML algorithms, new guidelines and recommendations may be proposed. In particular, there is a lot of attention currently being given to research on NNs and deep learning algorithms with new architectures, optimization algorithms and regularization techniques being proposed.

There are a number of directions that can be taken to build on this work. More data can be collected for both datasets presented here, which may improve the algorithms performances, particularly NNs since they typically require large training datasets. Furthermore, although a lot of the major algorithms have been tested in this thesis there are other algorithms including LDA, Gaussian mixture models (GMM), Riemannian geometry-based classifiers [99] and other NN architectures including RBMs and Transformers that can be explored. Additionally, the algorithms can be compared using other control signal types including SSVEP, SCP & Error-related potentials to see if similar trends are observed or if new ones arise. Transfer learning is also an important area that is receiving attention within the BCI field defined as the ability to apply knowledge from one domain or task to another. This includes training an algorithm on a set of subjects and using it on other subjects performing the same task or using models trained for a certain task for other tasks, for instance a model trained on sensorimotor data used on P300 data. This procedure is common with NNs, where a pretrained network is fine-tuned by training it on data from a different distribution than what is was originally trained on. This can be a BCI NN model being trained on data from a different BCI task or perhaps even a large non-BCI NN model trained on a large dataset from another field being fine-tuned for a BCI task. Finally, these models should also be tested online to determine if the offline results and trends hold for real-world applications.

Bibliography

[1] Po T Wang, Christine E King, An H Do, and Zoran Nenadic. Pushing the communication speed limit of a noninvasive bci speller. *arXiv preprint arXiv:1212.0469*, 2012.

[2] Andrea Kübler, Boris Kotchoubey, Jochen Kaiser, Jonathan R Wolpaw, and Niels Birbaumer. Brain–computer communication: Unlocking the locked in. *Psychological bulletin*, 127(3):358, 2001.

[3] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. Brain computer interfaces, a review. *sensors*, 12(2):1211–1279, 2012.

[4] Kaido Värbu, Naveed Muhammad, and Yar Muhammad. Past, present, and future of eeg-based bci applications. *Sensors*, 22(9):3331, 2022.

[5] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.

[6] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick Van Der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[7] Stefano Silvoni, Ander Ramos-Murguialday, Marianna Cavinato, Chiara Volpato, Giulia Cisotto, Andrea Turolla, Francesco Piccione, and Niels Birbaumer. Brain-computer interface in stroke: a review of progress. *Clinical EEG and neuroscience*, 42(4):245–252, 2011.

[8] Javier Asensio-Cubero, John Q Gan, and Ramaswamy Palaniappan. Multiresolution analysis over graphs for a motor imagery based online bci game. *Computers in biology and medicine*, 68:21–26, 2016.

[9] Woosang Cho, A Heilinger, R Xu, M Zehetner, S Schobesberger, N Murovec, R Ortner, and C Guger. Hemiparetic stroke rehabilitation using avatar and electrical stimulation based on non-invasive brain computer interface. *International Journal of Physical Medicine & Rehabilitation*, 5(04):10–4172, 2017.

[10] Sameer Raju Dhole, Amith Kashyap, Animesh Narayan Dangwal, and Rajasekar Mohan. A novel helmet design and implementation for drowsiness and fall detection of

workers on-site using eeg and random-forest classifier. *Procedia Computer Science*, 151:947–952, 2019.

[11] Jerone Dunbar, Juan E Gilbert, and Ben Lewis. Exploring differences between self-report and electrophysiological indices of drowsy driving: a usability examination of a personal brain-computer interface device. *Journal of Safety Research*, 74:27–34, 2020.

[12] Marc W Slutzky and Robert D Flint. Physiological properties of brain-machine interface input signals. *Journal of neurophysiology*, 118(2):1329–1343, 2017.

[13] Gerwin Schalk, Kai J Miller, Nicholas R Anderson, J Adam Wilson, Matthew D Smyth, Jeffrey G Ojemann, Daniel W Moran, Jonathan R Wolpaw, and Eric C Leuthardt. Two-dimensional movement control using electrocorticographic signals in humans. *Journal of neural engineering*, 5(1):75, 2008.

[14] Leigh R Hochberg, Mijail D Serruya, Gerhard M Friehs, Jon A Mukand, Maryam Saleh, Abraham H Caplan, Almut Branner, David Chen, Richard D Penn, and John P Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, 2006.

[15] Vadim S Polikov, Patrick A Tresco, and William M Reichert. Response of brain tissue to chronically implanted neural electrodes. *Journal of neuroscience methods*, 148(1):1–18, 2005.

[16] George C McConnell, Howard D Rees, Allan I Levey, Claire-Anne Gutekunst, Robert E Gross, and Ravi V Bellamkonda. Implanted neural electrodes cause chronic, local inflammation that is correlated with local neurodegeneration. *Journal of neural engineering*, 6(5):056003, 2009.

[17] JW Belliveau, DN Kennedy, RC McKinstry, BR Buchbinder, RMt Weisskoff, MS Cohen, JM Vevea, TJ Brady, and BR Rosen. Functional mapping of the human visual cortex by magnetic resonance imaging. *Science*, 254(5032):716–719, 1991.

[18] R Christopher DeCharms, Kalina Christoff, Gary H Glover, John M Pauly, Susan Whitfield, and John DE Gabrieli. Learned regulation of spatially localized brain activation using real-time fmri. *Neuroimage*, 21(1):436–443, 2004.

[19] Ranganatha Sitaram, Andrea Caria, and Niels Birbaumer. Hemodynamic brain–computer interfaces for communication and rehabilitation. *Neural networks*, 22(9):1320–1328, 2009.

[20] Nikolaus Weiskopf, Klaus Mathiak, Simon W Bock, Frank Scharnowski, Ralf Veit, Wolfgang Grodd, Rainer Goebel, and Niels Birbaumer. Principles of a brain-computer interface (bci) based on real-time functional magnetic resonance imaging (fmri). *IEEE transactions on biomedical engineering*, 51(6):966–970, 2004.

[21] Stephan Waldert, Tobias Pistohl, Christoph Braun, Tonio Ball, Ad Aertsen, and Carsten Mehring. A review on directional information in neural signals for brain-machine interfaces. *Journal of Physiology-Paris*, 103(3-5):244–254, 2009.

[22] Riitta Salmelin, M Hámáaláinen, M Kajola, and R Hari. Functional segregation of movement-related rhythmic activity in the human brain. *Neuroimage*, 2(4):237–243, 1995.

[23] Thomas Navin Lal, Michael Schröder, N Jeremy Hill, Hubert Preissl, Thilo Hinterberger, Jürgen Mellinger, Martin Bogdan, Wolfgang Rosenstiel, Thomas Hofmann, Niels Birbaumer, et al. A brain computer interface with online feedback based on magnetoencephalography. In *Proceedings of the 22nd international conference on Machine learning*, pages 465–472, 2005.

[24] Gentaro Taga, Fumitaka Homae, and Hama Watanabe. Effects of source-detector distance of near infrared spectroscopy on the measurement of the cortical hemodynamic response in infants. *Neuroimage*, 38(3):452–460, 2007.

[25] Shirley M Coyle, Tomás E Ward, and Charles M Markham. Brain–computer interface using a simplified functional near-infrared spectroscopy system. *Journal of neural engineering*, 4(3):219, 2007.

[26] Sylvain Baillet, John C Mosher, and Richard M Leahy. Electromagnetic brain mapping. *IEEE Signal processing magazine*, 18(6):14–30, 2001.

[27] Nicolas Chauveau, Xavier Franceries, Bernard Doyon, Bernard Rigaud, Jean Pierre Morucci, and Pierre Celsis. Effects of skull thickness, anisotropy, and inhomogeneity on forward eeg/erp computations using a spherical three-dimensional resistor mesh model. *Human brain mapping*, 21(2):86–97, 2004.

[28] Swati Aggarwal and Nupur Chugh. Signal processing techniques for motor imagery brain computer interface: A review. *Array*, 1:100003, 2019.

[29] Sho Nakagome, Trieu Phat Luu, Yongtian He, Akshay Sujatha Ravindran, and Jose L Contreras-Vidal. An empirical comparison of neural networks and machine learning algorithms for eeg gait decoding. *Scientific reports*, 10(1):4372, 2020.

[30] Miznan Behri, Abdulhamit Subasi, and Saeed Mian Qaisar. Comparison of machine learning methods for two class motor imagery tasks using eeg in brain-computer interface. In *2018 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–5. IEEE, 2018.

[31] Rahul Sharma, Minju Kim, and Akansha Gupta. Motor imagery classification in brain-machine interface with machine learning algorithms: Classical approach to multi-layer perceptron model. *Biomedical Signal Processing and Control*, 71:103101, 2022.

[32] Alexander Chan, Christopher E Early, Sishir Subedi, Yuezhe Li, and Hong Lin. Systematic analysis of machine learning algorithms on eeg data for brain state intelligence. In *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 793–799. IEEE, 2015.

[33] Felix A Heilmeyer, Robin T Schirrmeister, Lukas DJ Fiederer, Martin Volker, Joos Behncke, and Tonio Ball. A large-scale evaluation framework for eeg deep learning architectures. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1039–1045. IEEE, 2018.

[34] Rabie A Ramadan and Athanasios V Vasilakos. Brain computer interface: control signals review. *Neurocomputing*, 223:26–44, 2017.

[35] J Vernon Odom, Michael Bach, Colin Barber, Mitchell Brigell, Michael F Marmor, Alma Patrizia Tormene, and Graham E Holder. Visual evoked potentials standard (2004). *Documenta ophthalmologica*, 108:115–123, 2004.

[36] Guangyu Bin, Xiaorong Gao, Yijun Wang, Bo Hong, and Shangkai Gao. Vep-based brain-computer interfaces: time, frequency, and code modulations [research frontier]. *IEEE Computational Intelligence Magazine*, 4(4):22–26, 2009.

[37] Sungchul Mun, Min-Chul Park, Sangin Park, and Mincheol Whang. Ssvep and erp measurement of cognitive fatigue caused by stereoscopic 3d. *Neuroscience letters*, 525(2):89–94, 2012.

[38] John Polich. Updating p300: an integrative theory of p3a and p3b. *Clinical neurophysiology*, 118(10):2128–2148, 2007.

[39] Gert Pfurtscheller and FH Lopes Da Silva. Event-related eeg/meg synchronization and desynchronization: basic principles. *Clinical neurophysiology*, 110(11):1842–1857, 1999.

[40] Marc Jeannerod. Mental imagery in the motor context. *Neuropsychologia*, 33(11):1419–1432, 1995.

[41] Byung Hyung Kim, Minho Kim, and Sungho Jo. Quadcopter flight control using a low-cost hybrid interface with eeg-based classification and eye tracking. *Computers in biology and medicine*, 51:82–92, 2014.

[42] Laura A Miller, Kathy A Stubblefield, Robert D Lipschutz, Blair A Lock, and Todd A Kuiken. Improved myoelectric prosthesis control using targeted reinnervation surgery: a case series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(1):46–50, 2008.

[43] Niels Birbaumer, Nimr Ghanayim, Thilo Hinterberger, Iver Iversen, Boris Kotchoubey, Andrea Kübler, Juri Perelmouter, Edward Taub, and Herta Flor. A spelling device for the paralysed. *Nature*, 398(6725):297–298, 1999.

[44] Xiaogang Chen, Yijun Wang, Masaki Nakanishi, Xiaorong Gao, Tzyy-Ping Jung, and Shangkai Gao. High-speed spelling with a noninvasive brain–computer interface. *Proceedings of the national academy of sciences*, 112(44):E6058–E6067, 2015.

[45] Jonathan R Wolpaw and Dennis J McFarland. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proceedings of the national academy of sciences*, 101(51):17849–17854, 2004.

[46] Ahmed A Karim, Thilo Hinterberger, Jürgen Richter, Jürgen Mellinger, Nicola Neumann, Herta Flor, Andrea Kübler, and Niels Birbaumer. Neural internet: Web surfing with brain potentials for the completely paralyzed. *Neurorehabilitation and Neural Repair*, 20(4):508–515, 2006.

[47] Christine E King, Po T Wang, Colin M McCrimmon, Cathy CY Chou, An H Do, and Zoran Nenadic. The feasibility of a brain-computer interface functional electrical stimulation system for the restoration of overground walking after paraplegia. *Journal of neuroengineering and rehabilitation*, 12(1):1–11, 2015.

[48] Bradley J Edelman, Jianjun Meng, Daniel Suma, Claire Zurn, E Nagarajan, BS Baxter, Christopher C Cline, and BJSR He. Noninvasive neuroimaging enhances continuous neural tracking for robotic device control. *Science robotics*, 4(31):eaaw6844, 2019.

[49] Mikhail A Lebedev and Miguel AL Nicolelis. Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation. *Physiological reviews*, 97(2):767–837, 2017.

[50] Ander Ramos-Murguialday, Doris Broetz, Massimiliano Rea, Leonhard Läer, Özge Yilmaz, Fabricio L Brasil, Giulia Liberati, Marco R Curado, Eliana Garcia-Cossio, Alexandros Vyziotis, et al. Brain–machine interface in chronic stroke rehabilitation: a controlled study. *Annals of neurology*, 74(1):100–108, 2013.

[51] Febo Cincotti, Donatella Mattia, Fabio Aloise, Simona Bufalari, Gerwin Schalk, Giuseppe Oriolo, Andrea Cherubini, Maria Grazia Marciani, and Fabio Babiloni. Non-invasive brain–computer interface system: towards its application as assistive technology. *Brain research bulletin*, 75(6):796–803, 2008.

[52] Jingjing Chen, Dan Zhang, Andreas K Engel, Qin Gong, and Alexander Maye. Application of a single-flicker online ssvep bci for spatial navigation. *PloS one*, 12(5):e0178385, 2017.

[53] Christian J Bell, Pradeep Shenoy, Rawichote Chalodhorn, and Rajesh PN Rao. Control of a humanoid robot by a noninvasive brain–computer interface in humans. *Journal of neural engineering*, 5(2):214, 2008.

[54] Mehrdad Fatourechi, Ali Bashashati, Rabab K Ward, and Gary E Birch. Emg and eog artifacts in brain computer interface systems: A survey. *Clinical neurophysiology*, 118(3):480–494, 2007.

[55] Tzyy-Ping Jung, Scott Makeig, Colin Humphries, Te-Won Lee, Martin J Mckeown, Vicente Iragui, and Terrence J Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37(2):163–178, 2000.

[56] Ramaswamy Palaniappan. Brain computer interface design using band powers extracted during mental tasks. In *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, pages 321–324. IEEE, 2005.

[57] Chungsong Kim, Jinwei Sun, Dan Liu, Qisong Wang, and Sunggyun Paek. An effective feature extraction method by power spectral density of eeg signal for 2-class motor imagery-based bci. *Medical & biological engineering & computing*, 56:1645–1658, 2018.

[58] Herbert Ramoser, Johannes Muller-Gerking, and Gert Pfurtscheller. Optimal spatial filtering of single trial eeg during imagined hand movement. *IEEE transactions on rehabilitation engineering*, 8(4):441–446, 2000.

[59] Richard O Duda, Peter E Hart, et al. *Pattern classification*. John Wiley & Sons, 2 edition, 2000.

[60] Reinhold Scherer, GR Muller, Christa Neuper, Bernhard Graimann, and Gert Pfurtscheller. An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate. *IEEE Transactions on Biomedical Engineering*, 51(6):979–984, 2004.

[61] Vladimir Bostanov. Bci competition 2003-data sets ib and iib: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram. *IEEE Transactions on Biomedical engineering*, 51(6):1057–1061, 2004.

[62] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[63] Alois Schlögl, Felix Lee, Horst Bischof, and Gert Pfurtscheller. Characterization of four-class motor imagery eeg data for the bci-competition 2005. *Journal of neural engineering*, 2(4):L14, 2005.

[64] Matthias Kaper, Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, and Helge Ritter. Bci competition 2003-data set iib: support vector machines for the p300 speller paradigm. *IEEE Transactions on biomedical Engineering*, 51(6):1073–1076, 2004.

[65] Deon Garrett, David A Peterson, Charles W Anderson, and Michael H Thaut. Comparison of linear, nonlinear, and feature selection methods for eeg signal classification. *IEEE Transactions on neural systems and rehabilitation engineering*, 11(2):141–144, 2003.

[66] Nurul E'zzati Md Isa, Amiza Amir, Mohd Zaizu Ilyas, and Mohammad Shahrazel Razalli. The performance analysis of k-nearest neighbors (k-nn) algorithm for motor imagery classification based on eeg signal. In *MATEC web of conferences*, volume 140, page 01024. EDP Sciences, 2017.

[67] Jing Fan, Joshua W Wade, Dayi Bian, Alexandra P Key, Zachary E Warren, Lorraine C Mion, and Nilanjan Sarkar. A step towards eeg-based brain computer interface for autism intervention. In *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 3767–3770. IEEE, 2015.

[68] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, 2 edition, 1990.

[69] Stephen J Roberts and William D Penny. Real-time brain-computer interfacing: a preliminary study using bayesian learning. *Medical and Biological Engineering and computing*, 38:56–61, 2000.

[70] Steven Lemm, Christin Schafer, and Gabriel Curio. Bci competition 2003-data set iii: probabilistic modeling of sensorimotor/spl mu/rhythms for classification of imaginary hand movements. *IEEE Transactions on Biomedical Engineering*, 51(6):1077–1080, 2004.

[71] Georg E Fabiani, Dennis J McFarland, Jonathan R Wolpaw, and Gert Pfurtscheller. Conversion of eeg activity into cursor movement by a brain-computer interface (bci). *IEEE transactions on neural systems and rehabilitation engineering*, 12(3):331–338, 2004.

[72] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[73] Yousef Rezaei Tabar and Ugur Halici. A novel deep learning approach for classification of eeg motor imagery signals. *Journal of neural engineering*, 14(1):016003, 2016.

[74] Arunabha M Roy. Adaptive transfer learning-based multiscale feature fused deep convolutional neural network for eeg mi multiclassification in brain–computer interface. *Engineering Applications of Artificial Intelligence*, 116:105347, 2022.

[75] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017.

[76] Kaishuo Zhang, Neethu Robinson, Seong-Whan Lee, and Cuntai Guan. Adaptive transfer learning for eeg motor imagery classification with deep convolutional neural network. *Neural Networks*, 136:1–10, 2021.

[77] 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, 2018.

[78] Wonjun Ko, Eunjin Jeon, Seungwoo Jeong, and Heung-Il Suk. Multi-scale neural network for eeg representation learning in bci. *IEEE Computational Intelligence Magazine*, 16(2):31–45, 2021.

[79] Koel Das and Zoran Nenadic. An efficient discriminant-based solution for small sample size problem. *Pattern Recognition*, 42(5):857–866, 2009.

[80] Koel Das and Zoran Nenadic. Approximate information discriminant analysis: A computationally simple heteroscedastic feature extraction technique. *Pattern Recognition*, 41(5):1548–1557, 2008.

[81] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[82] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[83] Bo Hong, Fei Guo, Tao Liu, Xiaorong Gao, and Shangkai Gao. N200-speller using motion-onset visual response. *Clinical neurophysiology*, 120(9):1658–1666, 2009.

[84] Dean J Krusienski, Eric W Sellers, François Cabestaing, Sabri Bayoudh, Dennis J McFarland, Theresa M Vaughan, and Jonathan R Wolpaw. A comparison of classification techniques for the p300 speller. *Journal of neural engineering*, 3(4):299, 2006.

[85] Nikolay V Manyakov, Nikolay Chumerin, Adrien Combaz, and Marc M Van Hulle. Comparison of classification methods for p300 brain-computer interface on disabled subjects. *Computational intelligence and neuroscience*, 2011:1–12, 2011.

[86] RJ Oweis, N Hamdi, A Ghazali, and K Lwissy. A comparison study on machine learning algorithms utilized in p300-based bci. *J Health Med Informat*, 4(126):2, 2013.

[87] Xinqiao Zhao, Hongmiao Zhang, Guilin Zhu, Fengxiang You, Shaolong Kuang, and Lining Sun. A multi-branch 3d convolutional neural network for eeg-based motor imagery classification. *IEEE transactions on neural systems and rehabilitation engineering*, 27(10):2164–2177, 2019.

[88] Kris van Noord, Wenjin Wang, and Hailong Jiao. Insights of 3d input cnn in eeg-based emotion recognition. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 212–215. IEEE, 2021.

[89] Akinari Onishi. Convolutional neural network transfer learning applied to the affective auditory p300-based bci. *Journal of Robotics and Mechatronics*, 32(4):731–737, 2020.

[90] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520–522, 1996.

[91] An H Do, Po T Wang, Christine E King, Ahmad Abiri, and Zoran Nenadic. Brain-computer interface controlled functional electrical stimulation system for ankle movement. *Journal of neuroengineering and rehabilitation*, 8:1–14, 2011.

[92] Zoran Nenadic. Information discriminant analysis: Feature extraction with an information-theoretic objective. *IEEE transactions on pattern analysis and machine intelligence*, 29(8):1394–1407, 2007.

[93] Anjum Naeem Malik, Javaid Iqbal, and Mohsin I Tiwana. Eeg signals classification and determination of optimal feature-classifier combination for predicting the movement intent of lower limb. In *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, pages 45–49. IEEE, 2016.

[94] Maged S Al-Quraishi, Irraivan Elamvazuthi, Tong Boon Tang, Muhammad Al-Qurishi, S Parasuraman, and Alberto Borboni. Multimodal fusion approach based on eeg and emg signals for lower limb movement recognition. *IEEE Sensors Journal*, 21(24):27640–27650, 2021.

[95] Madiha Tariq, Pavel M Trivailo, and Milan Simic. Classification of left and right foot kinaesthetic motor imagery using common spatial pattern. *Biomedical Physics & Engineering Express*, 6(1):015008, 2019.

[96] Rohit Bose, Anwesha Khasnobish, Sanniv Bhaduri, and DN Tibarewala. Performance analysis of left and right lower limb movement classification from eeg. In *2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 174–179. IEEE, 2016.

[97] Madiha Tariq, Pavel M Trivailo, and Milan Simic. Mu-beta event-related (de) synchronization and eeg classification of left-right foot dorsiflexion kinaesthetic motor imagery for bci. *Plos one*, 15(3):e0230184, 2020.

[98] Francine Malouin, Carol L Richards, Philip L Jackson, Francine Dumas, and Julien Doyon. Brain activations during motor imagery of locomotor-related tasks: A pet study. *Human brain mapping*, 19(1):47–62, 2003.

[99] Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Riemannian geometry applied to bci classification. In *International conference on latent variable analysis and signal separation*, pages 629–636. Springer, 2010.

Appendix A

Cross-validation results for each P300 session

This section contains all the cross-validation results for each P300 session using the 6 ML algorithms tested.

Table A.1: Average 10x10 cross-validations for each P300 session using CAB. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
1	1	509/84	95.55 ± 0.45	79.05 ± 2.11	98.27 ± 0.22
	2	523/87	96.13 ± 0.31	80.11 ± 1.09	98.80 ± 0.30
	3	522/88	95.75 ± 0.28	77.84 ± 2.02	98.77 ± 0.29
	4	623/107	97.42 ± 0.17	88.13 ± 1.08	99.02 ± 0.14
	5	627/103	92.23 ± 0.40	51.84 ± 2.06	98.87 ± 0.28
	6	625/105	95.03 ± 0.19	80.48 ± 0.50	97.47 ± 0.20
	7	623/107	85.71 ± 0.39	2.90 ± 3.07	99.94 ± 0.16

	8	630/108	96.88 ± 0.23	85.74 ± 1.70	98.79 ± 0.17
2	1	625/105	87.73 ± 0.53	24.00 ± 2.57	98.43 ± 0.33
	2	624/106	89.63 ± 0.30	46.79 ± 1.19	96.91 ± 0.39
3	1	625/105	92.59 ± 0.36	63.24 ± 2.21	97.52 ± 0.26
	2	631/109	94.19 ± 0.49	75.50 ± 1.44	97.42 ± 0.39
	3	623/107	96.63 ± 0.13	81.03 ± 0.89	99.31 ± 0.20
4	1	531/89	92.74 ± 0.34	62.92 ± 2.05	97.74 ± 0.23
5	1	529/86	93.06 ± 0.23	58.84 ± 2.46	98.62 ± 0.20
	2	530/88	94.74 ± 0.23	73.86 ± 1.20	98.21 ± 0.22
	3	533/88	94.20 ± 0.27	70.11 ± 2.28	98.18 ± 0.29
	4	623/107	93.92 ± 0.30	69.44 ± 1.87	98.12 ± 0.26
	5	632/108	93.81 ± 0.44	69.63 ± 1.50	97.94 ± 0.37
	6	626/104	95.32 ± 0.21	77.60 ± 1.20	98.26 ± 0.19
6	1	625/105	95.25 ± 0.37	78.76 ± 1.35	98.02 ± 0.30
	2	521/89	95.62 ± 0.47	81.12 ± 2.17	98.10 ± 0.37
	3	622/108	96.70 ± 0.19	84.72 ± 0.49	98.78 ± 0.22
	4	627/103	94.27 ± 0.48	69.32 ± 2.79	98.37 ± 0.31
	5	626/105	92.98 ± 0.81	68.29 ± 5.33	97.12 ± 0.36
	6	627/104	96.94 ± 0.33	84.71 ± 1.89	98.96 ± 0.14
	7	626/104	97.81 ± 0.28	89.62 ± 1.18	99.17 ± 0.21
	8	624/106	96.04 ± 0.25	78.96 ± 1.34	98.94 ± 0.15
	9	632/108	92.46 ± 0.43	63.89 ± 3.60	97.34 ± 0.37
	10	631/108	95.43 ± 0.39	77.69 ± 2.16	98.46 ± 0.26
	11	634/106	96.89 ± 0.25	84.34 ± 1.01	98.99 ± 0.29
	12	633/107	96.03 ± 0.37	79.16 ± 1.82	98.88 ± 0.23
7	1	623/104	89.02 ± 0.42	44.13 ± 2.37	96.52 ± 0.30
	2	622/108	91.99 ± 0.47	59.17 ± 2.33	97.68 ± 0.33

	3	626/104	91.45 ± 0.41	58.75 ± 2.05	96.88 ± 0.27
	4	631/109	89.47 ± 0.24	47.43 ± 2.16	96.74 ± 0.26
	5	631/109	89.80 ± 0.47	48.90 ± 1.30	96.86 ± 0.50
8	1	625/105	88.40 ± 0.59	42.57 ± 3.08	96.10 ± 0.39
	2	625/105	88.99 ± 0.49	45.05 ± 2.25	96.37 ± 0.52

Table A.2: Average 10x10 cross-validations for each P300 session using LR. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
1	1	509/84	94.03 ± 0.45	82.50 ± 1.31	95.93 ± 0.43
	2	523/87	95.36 ± 0.44	85.29 ± 1.13	97.04 ± 0.58
	3	522/88	95.38 ± 0.30	84.66 ± 1.70	97.18 ± 0.41
	4	623/107	98.19 ± 0.27	93.83 ± 0.62	98.94 ± 0.25
	5	627/103	96.04 ± 0.19	79.71 ± 1.10	98.72 ± 0.07
	6	625/105	94.95 ± 0.48	86.00 ± 1.21	96.45 ± 0.54
	7	623/107	95.68 ± 0.25	79.72 ± 1.73	98.43 ± 0.10
	8	630/108	97.68 ± 0.20	93.70 ± 1.08	98.37 ± 0.14
2	1	625/105	89.67 ± 0.68	66.19 ± 3.60	93.62 ± 0.41
	2	624/106	88.48 ± 0.51	60.66 ± 1.94	93.21 ± 0.49
3	1	625/105	90.25 ± 0.34	65.52 ± 1.99	94.40 ± 0.39
	2	631/109	93.82 ± 0.38	79.82 ± 0.71	96.24 ± 0.40
	3	623/107	96.19 ± 0.35	85.61 ± 1.73	98.01 ± 0.25
4	1	531/89	90.98 ± 0.45	68.76 ± 1.31	94.71 ± 0.46
5	1	529/86	89.72 ± 0.73	63.60 ± 2.99	93.97 ± 0.58
	2	530/88	93.50 ± 0.40	77.84 ± 2.34	96.09 ± 0.28

	3	533/88	92.96 ± 0.57	75.45 ± 2.79	95.85 ± 0.39
	4	623/107	93.12 ± 0.30	77.20 ± 1.40	95.86 ± 0.33
	5	632/108	93.50 ± 0.52	76.30 ± 2.35	96.44 ± 0.41
	6	626/104	95.07 ± 0.53	81.83 ± 2.45	97.27 ± 0.32
6	1	625/105	94.59 ± 0.45	83.62 ± 1.64	96.43 ± 0.39
	2	521/89	95.41 ± 0.34	85.62 ± 1.80	97.08 ± 0.32
	3	622/108	97.03 ± 0.39	89.44 ± 1.32	98.34 ± 0.28
	4	627/103	93.49 ± 0.48	82.04 ± 0.99	95.37 ± 0.49
	5	626/105	91.71 ± 0.58	78.76 ± 1.81	93.88 ± 0.61
	6	627/104	95.73 ± 0.32	87.88 ± 1.50	97.03 ± 0.24
	7	626/104	97.58 ± 0.36	94.04 ± 1.20	98.16 ± 0.32
	8	624/106	97.11 ± 0.33	89.72 ± 1.60	98.37 ± 0.28
	9	632/108	89.03 ± 0.62	66.11 ± 2.04	92.94 ± 0.58
	10	631/108	94.21 ± 0.44	82.78 ± 0.85	96.16 ± 0.41
	11	634/106	96.43 ± 0.21	88.40 ± 1.34	97.78 ± 0.27
	12	633/107	95.58 ± 0.33	82.80 ± 1.68	97.74 ± 0.17
7	1	623/104	88.72 ± 0.54	58.17 ± 1.89	93.82 ± 0.39
	2	622/108	91.12 ± 0.39	67.13 ± 2.00	95.29 ± 0.33
	3	626/104	90.21 ± 0.58	61.15 ± 1.93	95.03 ± 0.53
	4	631/109	90.43 ± 0.60	63.39 ± 2.48	95.10 ± 0.40
	5	631/109	89.35 ± 0.56	61.65 ± 2.16	94.14 ± 0.56
8	1	625/105	88.62 ± 0.54	59.05 ± 2.45	93.58 ± 0.37
	2	625/105	89.05 ± 0.61	59.62 ± 2.14	94.00 ± 0.70

Table A.3: Average 10x10 cross-validations for each P300 session using FCNN. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
1	1	509/84	95.33 ± 0.36	76.55 ± 1.92	98.43 ± 0.28
	2	523/87	95.93 ± 0.68	79.20 ± 3.23	98.72 ± 0.51
	3	522/88	94.98 ± 0.40	76.14 ± 2.74	98.16 ± 0.29
	4	623/107	97.05 ± 0.25	85.51 ± 1.27	99.04 ± 0.19
	5	627/103	95.96 ± 0.47	76.70 ± 2.17	99.12 ± 0.25
	6	625/105	95.55 ± 0.33	80.19 ± 1.64	98.13 ± 0.34
	7	623/107	93.73 ± 0.37	58.97 ± 2.72	99.70 ± 0.17
	8	630/108	97.29 ± 0.50	86.57 ± 2.28	99.13 ± 0.24
2	1	625/105	91.16 ± 0.43	54.29 ± 2.29	97.36 ± 0.39
	2	624/106	90.70 ± 0.34	49.43 ± 2.28	97.71 ± 0.30
3	1	625/105	91.32 ± 0.35	55.71 ± 2.42	97.30 ± 0.47
	2	631/109	94.12 ± 0.40	69.45 ± 1.64	98.38 ± 0.30
	3	623/107	96.01 ± 0.49	80.19 ± 2.16	98.73 ± 0.38
4	1	531/89	91.76 ± 0.38	58.99 ± 2.15	97.25 ± 0.37
5	1	529/86	92.08 ± 0.45	57.33 ± 1.73	97.73 ± 0.46
	2	530/88	94.08 ± 0.36	70.57 ± 1.79	97.98 ± 0.40
	3	533/88	93.49 ± 0.54	65.68 ± 1.89	98.09 ± 0.47
	4	623/107	93.71 ± 0.32	68.13 ± 1.41	98.11 ± 0.29
	5	632/108	93.91 ± 0.42	67.22 ± 1.99	98.47 ± 0.35
	6	626/104	95.05 ± 0.44	73.56 ± 1.62	98.63 ± 0.34
	1	625/105	94.08 ± 0.39	72.67 ± 1.95	97.68 ± 0.33
	2	521/89	95.34 ± 0.45	77.19 ± 1.88	98.45 ± 0.47

	3	622/108	96.77 ± 0.40	84.26 ± 1.49	98.94 ± 0.32
	4	627/103	94.53 ± 0.53	72.62 ± 1.73	98.13 ± 0.46
	5	626/105	94.10 ± 0.30	72.76 ± 1.60	97.68 ± 0.44
	6	627/104	96.57 ± 0.39	83.94 ± 2.40	98.66 ± 0.18
	7	626/104	97.49 ± 0.39	87.88 ± 1.37	99.09 ± 0.34
	8	624/106	96.47 ± 0.43	81.04 ± 2.51	99.09 ± 0.26
	9	632/108	92.03 ± 0.78	60.00 ± 3.20	97.50 ± 0.67
	10	631/108	94.70 ± 0.37	75.46 ± 1.86	97.99 ± 0.27
	11	634/106	96.31 ± 0.34	81.98 ± 1.07	98.71 ± 0.40
	12	633/107	95.20 ± 0.35	74.77 ± 2.96	98.66 ± 0.31
7	1	623/104	90.00 ± 0.64	47.12 ± 1.43	97.16 ± 0.57
	2	622/108	91.79 ± 0.63	58.43 ± 3.45	97.59 ± 0.35
	3	626/104	91.30 ± 0.62	54.13 ± 3.28	97.48 ± 0.32
	4	631/109	90.73 ± 0.60	48.62 ± 2.78	98.00 ± 0.33
	5	631/109	90.27 ± 0.46	48.26 ± 2.18	97.53 ± 0.24
8	1	625/105	89.11 ± 0.70	42.48 ± 2.30	96.94 ± 0.56
	2	625/105	89.74 ± 0.69	41.81 ± 2.57	97.79 ± 0.48

Table A.4: Average 10x10 cross-validations for each P300 session using CNN. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
	1	509/84	95.43 ± 0.50	79.52 ± 1.90	98.06 ± 0.57
	2	523/87	95.74 ± 0.54	79.66 ± 3.45	98.41 ± 0.48
	3	522/88	94.79 ± 0.59	75.23 ± 3.44	98.08 ± 0.60
	4	623/107	97.19 ± 0.47	85.98 ± 2.09	99.12 ± 0.34
1					

	5	627/103	96.33 ± 0.33	81.94 ± 2.00	98.69 ± 0.36
	6	625/105	95.33 ± 0.26	80.57 ± 1.91	97.81 ± 0.37
	7	623/107	93.12 ± 0.94	55.61 ± 6.78	99.57 ± 0.18
	8	630/108	96.14 ± 0.37	83.43 ± 1.83	98.32 ± 0.33
2	1	625/105	89.88 ± 0.51	54.57 ± 4.09	95.81 ± 0.57
	2	624/106	90.08 ± 0.53	53.30 ± 1.90	96.33 ± 0.64
3	1	625/105	91.88 ± 0.60	63.71 ± 3.39	96.61 ± 0.74
	2	631/109	93.76 ± 0.50	74.40 ± 3.00	97.10 ± 0.48
	3	623/107	95.40 ± 0.30	78.79 ± 1.32	98.25 ± 0.22
4	1	531/89	91.92 ± 0.51	63.71 ± 2.41	96.65 ± 0.66
5	1	529/86	92.33 ± 0.44	61.98 ± 2.90	97.26 ± 0.43
	2	530/88	93.69 ± 0.72	70.00 ± 3.88	97.62 ± 0.62
	3	533/88	93.12 ± 0.60	66.25 ± 2.28	97.56 ± 0.60
	4	623/107	94.59 ± 0.53	75.05 ± 3.07	97.95 ± 0.48
	5	632/108	93.49 ± 0.49	71.11 ± 3.26	97.31 ± 0.39
	6	626/104	94.47 ± 0.62	77.40 ± 3.83	97.30 ± 0.71
6	1	625/105	93.33 ± 0.61	72.76 ± 3.05	96.78 ± 0.75
	2	521/89	94.05 ± 0.60	74.72 ± 4.64	97.35 ± 0.55
	3	622/108	95.99 ± 0.57	82.87 ± 2.53	98.26 ± 0.58
	4	627/103	94.18 ± 0.30	73.59 ± 3.00	97.56 ± 0.47
	5	626/105	94.13 ± 0.55	77.05 ± 3.11	97.00 ± 0.71
	6	627/104	96.47 ± 0.58	84.52 ± 2.45	98.45 ± 0.36
	7	626/104	96.88 ± 0.32	85.67 ± 2.21	98.74 ± 0.45
	8	624/106	95.53 ± 0.57	79.91 ± 2.19	98.19 ± 0.57
	9	632/108	90.62 ± 0.56	58.80 ± 2.28	96.06 ± 0.51
	10	631/108	94.26 ± 0.51	76.02 ± 1.68	97.39 ± 0.49
	11	634/106	95.53 ± 0.30	81.04 ± 2.29	97.95 ± 0.49

	12	633/107	95.07 ± 0.48	77.76 ± 2.83	97.99 ± 0.41
7	1	623/104	89.59 ± 0.80	50.10 ± 5.87	96.18 ± 0.36
	2	622/108	90.67 ± 0.83	58.43 ± 2.63	96.27 ± 0.93
	3	626/104	90.73 ± 0.68	57.98 ± 3.60	96.17 ± 0.48
	4	631/109	89.54 ± 0.76	52.57 ± 4.02	95.93 ± 0.72
	5	631/109	89.57 ± 0.69	52.84 ± 3.16	95.91 ± 0.44
8	1	625/105	87.36 ± 0.51	39.33 ± 2.83	95.42 ± 0.58
	2	625/105	87.96 ± 0.68	39.14 ± 5.12	96.16 ± 0.74

Table A.5: Average 10x10 cross-validations for each P300 session using SVM. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
1	1	509/84	95.11 ± 0.25	77.14 ± 1.90	98.07 ± 0.31
	2	523/87	95.89 ± 0.38	79.89 ± 1.87	98.55 ± 0.24
	3	522/88	94.89 ± 0.37	79.55 ± 2.10	97.47 ± 0.17
	4	623/107	97.23 ± 0.37	87.48 ± 2.06	98.91 ± 0.32
	5	627/103	94.99 ± 0.25	69.13 ± 0.73	99.23 ± 0.25
	6	625/105	94.56 ± 0.44	78.95 ± 1.73	97.18 ± 0.52
	7	623/107	91.12 ± 0.29	42.99 ± 1.72	99.39 ± 0.21
	8	630/108	96.11 ± 0.24	83.89 ± 1.81	98.21 ± 0.28
2	1	625/105	90.40 ± 0.63	60.95 ± 3.90	95.34 ± 0.23
	2	624/106	89.79 ± 0.58	54.81 ± 1.95	95.74 ± 0.56
3	1	625/105	90.52 ± 0.47	60.95 ± 2.17	95.49 ± 0.40
	2	631/109	93.59 ± 0.39	73.85 ± 1.03	97.00 ± 0.40
	3	623/107	95.68 ± 0.31	82.43 ± 1.71	97.96 ± 0.22

4	1	531/89	90.66 ± 0.62	61.12 ± 2.31	95.61 ± 0.57
5	1	529/86	91.35 ± 0.32	61.05 ± 1.89	96.28 ± 0.24
	2	530/88	93.16 ± 0.35	72.73 ± 1.83	96.55 ± 0.37
	3	533/88	93.54 ± 0.40	70.68 ± 1.82	97.32 ± 0.27
	4	623/107	92.92 ± 0.55	69.63 ± 1.46	96.92 ± 0.49
	5	632/108	93.93 ± 0.57	70.93 ± 2.24	97.86 ± 0.35
	6	626/104	94.45 ± 0.47	75.67 ± 1.43	97.57 ± 0.49
6	1	625/105	93.89 ± 0.51	74.86 ± 2.14	97.09 ± 0.31
	2	521/89	94.49 ± 0.56	77.53 ± 2.66	97.39 ± 0.38
	3	622/108	95.77 ± 0.32	80.56 ± 1.76	98.41 ± 0.21
	4	627/103	93.38 ± 0.38	73.01 ± 1.78	96.73 ± 0.38
	5	626/105	93.91 ± 0.42	77.14 ± 1.90	96.73 ± 0.32
	6	627/104	95.96 ± 0.22	82.12 ± 1.78	98.26 ± 0.32
	7	626/104	97.77 ± 0.16	90.19 ± 1.60	99.03 ± 0.28
	8	624/106	96.11 ± 0.28	81.60 ± 1.48	98.57 ± 0.23
	9	632/108	90.70 ± 0.46	63.80 ± 2.21	95.30 ± 0.21
	10	631/108	93.91 ± 0.44	74.91 ± 1.63	97.16 ± 0.40
	11	634/106	96.15 ± 0.36	82.74 ± 1.94	98.39 ± 0.24
	12	633/107	95.27 ± 0.51	77.01 ± 2.75	98.36 ± 0.25
7	1	623/104	89.74 ± 0.46	50.29 ± 1.93	96.32 ± 0.32
	2	622/108	92.32 ± 0.40	65.37 ± 1.67	96.99 ± 0.39
	3	626/104	90.34 ± 0.42	58.75 ± 1.63	95.59 ± 0.46
	4	631/109	89.26 ± 0.53	46.97 ± 1.78	96.56 ± 0.39
	5	631/109	89.01 ± 0.51	48.90 ± 2.43	95.94 ± 0.44
8	1	625/105	88.89 ± 0.71	47.90 ± 2.41	95.78 ± 0.60
	2	625/105	89.89 ± 0.32	50.67 ± 1.58	96.48 ± 0.37

Table A.6: Average 10x10 cross-validations for each P300 session using KNN. Non-odd/odd are the number of non-oddball/oddball trials for each session. TP, P, TN & N stand for true positives, positives, true negatives & negatives respectively where positives are oddball trials and negatives are non-oddball trials. TP/P & TN/N are the true positive and negative rates respectively.

Subject	Dataset	Non-odd/odd	Accuracy	TP/P	TN/N
1	1	509/84	87.45 ± 0.43	12.62 ± 2.33	99.80 ± 0.15
	2	523/87	87.93 ± 0.30	17.47 ± 1.69	99.66 ± 0.14
	3	522/88	87.15 ± 0.31	13.41 ± 1.82	99.58 ± 0.17
	4	623/107	87.81 ± 0.28	18.88 ± 1.61	99.65 ± 0.19
	5	627/103	88.63 ± 0.18	24.85 ± 1.58	99.11 ± 0.22
	6	625/105	88.16 ± 0.34	25.24 ± 2.14	98.74 ± 0.09
	7	623/107	89.04 ± 0.45	37.94 ± 2.01	97.82 ± 0.30
	8	630/108	89.16 ± 0.37	27.59 ± 2.22	99.71 ± 0.14
2	1	625/105	87.18 ± 0.55	26.10 ± 2.93	97.44 ± 0.31
	2	624/106	85.96 ± 0.30	9.91 ± 1.42	98.88 ± 0.23
3	1	625/105	86.92 ± 0.13	11.90 ± 1.36	99.52 ± 0.21
	2	631/109	87.14 ± 0.28	18.07 ± 1.48	99.06 ± 0.30
	3	623/107	88.41 ± 0.35	25.51 ± 2.47	99.21 ± 0.13
4	1	531/89	88.11 ± 0.30	27.87 ± 1.57	98.21 ± 0.26
5	1	529/86	87.53 ± 0.16	13.02 ± 1.25	99.64 ± 0.13
	2	530/88	87.18 ± 0.26	13.18 ± 1.70	99.47 ± 0.08
	3	533/88	87.78 ± 0.30	17.84 ± 2.10	99.32 ± 0.23
	4	623/107	87.58 ± 0.25	21.96 ± 1.04	98.84 ± 0.25
	5	632/108	85.84 ± 0.16	6.94 ± 0.62	99.32 ± 0.16
	6	626/104	86.10 ± 0.38	15.38 ± 2.28	97.84 ± 0.32
	1	625/105	89.38 ± 0.38	42.76 ± 2.27	97.22 ± 0.26
	2	521/89	88.05 ± 0.23	22.47 ± 1.74	99.25 ± 0.13

	3	622/108	88.21 ± 0.13	22.13 ± 1.05	99.68 ± 0.14
	4	627/103	89.36 ± 0.36	26.31 ± 2.27	99.71 ± 0.14
	5	626/105	86.33 ± 0.10	8.38 ± 0.93	99.41 ± 0.14
	6	627/104	87.43 ± 0.14	11.92 ± 0.77	99.95 ± 0.10
	7	626/104	90.10 ± 0.34	32.50 ± 1.41	99.66 ± 0.18
	8	624/106	86.70 ± 0.41	12.08 ± 2.18	99.38 ± 0.18
	9	632/108	86.36 ± 0.20	21.76 ± 1.95	97.41 ± 0.28
	10	631/108	86.74 ± 0.31	14.17 ± 1.25	99.16 ± 0.20
	11	634/106	86.80 ± 0.38	13.02 ± 1.18	99.13 ± 0.32
	12	633/107	85.84 ± 0.19	5.70 ± 0.78	99.38 ± 0.15
7	1	623/104	85.57 ± 0.10	2.88 ± 0.43	99.37 ± 0.09
	2	622/108	85.68 ± 0.21	5.46 ± 1.68	99.61 ± 0.11
	3	626/104	85.81 ± 0.24	2.50 ± 0.98	99.65 ± 0.16
	4	631/109	84.66 ± 0.30	2.66 ± 0.64	98.83 ± 0.35
	5	631/109	84.39 ± 0.11	0.18 ± 0.37	98.94 ± 0.17
8	1	625/105	85.38 ± 0.35	5.71 ± 1.13	98.77 ± 0.30
	2	625/105	85.03 ± 0.29	3.24 ± 0.76	98.77 ± 0.32

Appendix B

Top components selection method

This section explains the process of selecting the top 5 components using each of the three preprocessing methods (ICA, IDA, CSP) used for the Dorsiflexion task. Figure B.1 displays the process using ICA as an example. The first step is to compute all 15 components from the raw EEG signals. The processed signal is then segmented into trials, one trial per idle/dorsiflexion segment. The PSDs are then computed for each trial. The SNR is then computed as

$$SNR = 2\,\frac{(\mu_D - \mu_I)^2}{\sigma_D^2 + \sigma_I^2}$$

where $\mu_D, \mu_I, \sigma_D, \sigma_I$ are the means and standard deviations of the PSD trials for dorsiflexion and idling respectively. The maximum SNR value across all frequencies for each component is then computed. The 5 components with the greatest maximum SNR values are selected and further processed. This same process is applied for selecting the top 5 CSP components. IDA allows for the selection of the size of the feature space therefore this is chosen to be 5 and the selection method is not applied.

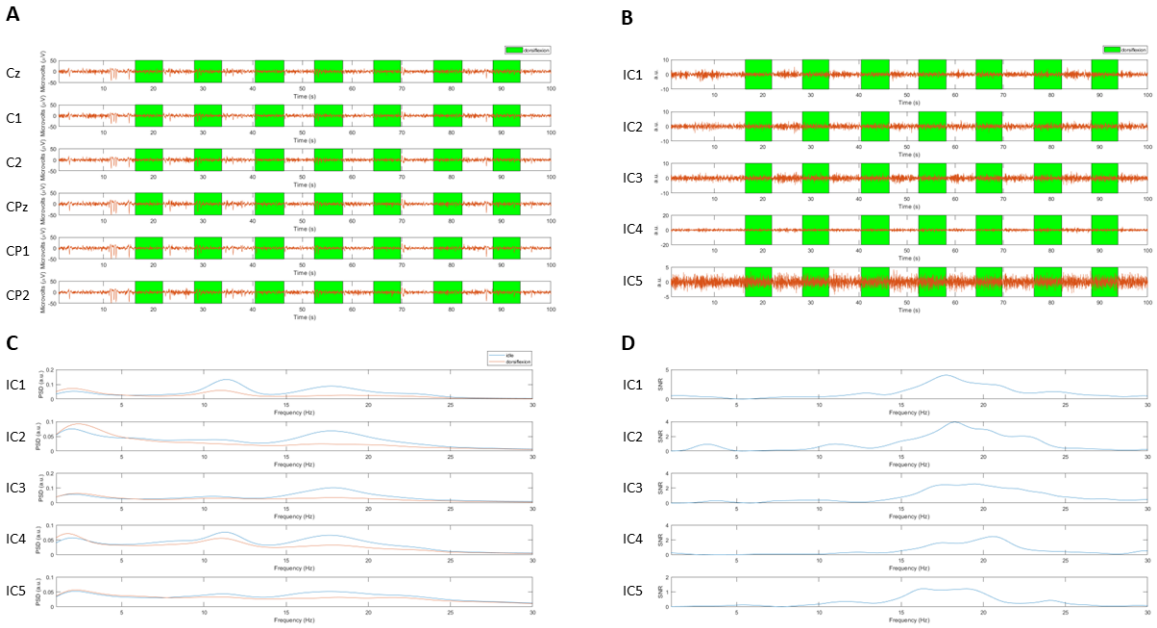

Figure B.1: Description of the top components selection method that was used to select the top 5 components for ICA, IDA & CSP during preprocessing of the dorsiflexion task. This example portrays the application of the method to ICA. (A) The raw EEG signals from 6 of the channels, periods of dorsiflexion are shaded in green while periods of idling are unshaded. (B) 5 of the 15 resulting ICA components with the same shading pattern as (A). One trial was segmented from each Idle and each Dorsiflexion region and the PSDs of the trials were computed. (C) Average PSDs for Idle and Dorsiflexion trials for the 5 ICA components shown in (B). (D) SNRs for the 5 ICA components calculated as $SNR = 2\ \frac{(\mu_D - \mu_I)^2}{\sigma_D^2 + \sigma_I^2}$ where $\mu_D, \mu_I, \sigma_D, \sigma_I$ are the means and standard deviations of the dorsiflexion and idling trials respectively. The maximum SNR for each ICA component was calculated and the 5 components with the highest maximum SNRs were selected to compute the data features.

Appendix C

All feature topoplots for Dorsiflexion task

This section contains all the feature coefficients learned by the algorithms for the Dorsiflexion task in Chapter 5 displayed on topoplots.

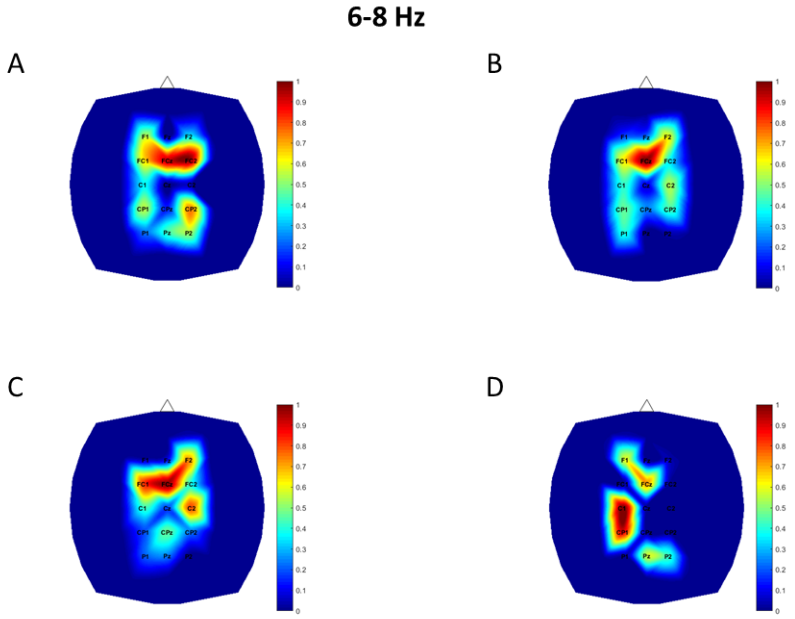

Figure C.1: Topoplots of feature coefficients in the 6-8 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

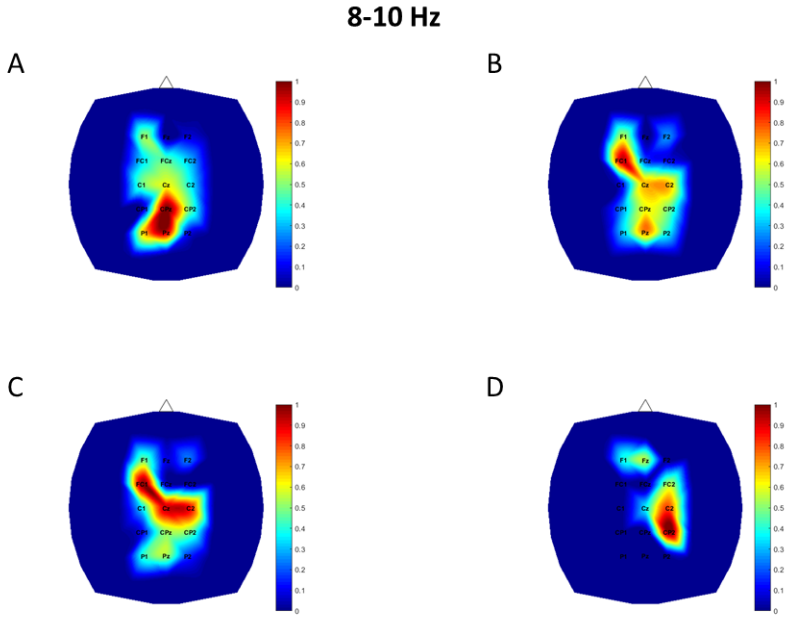

Figure C.2: Topoplots of feature coefficients in the 8-10 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

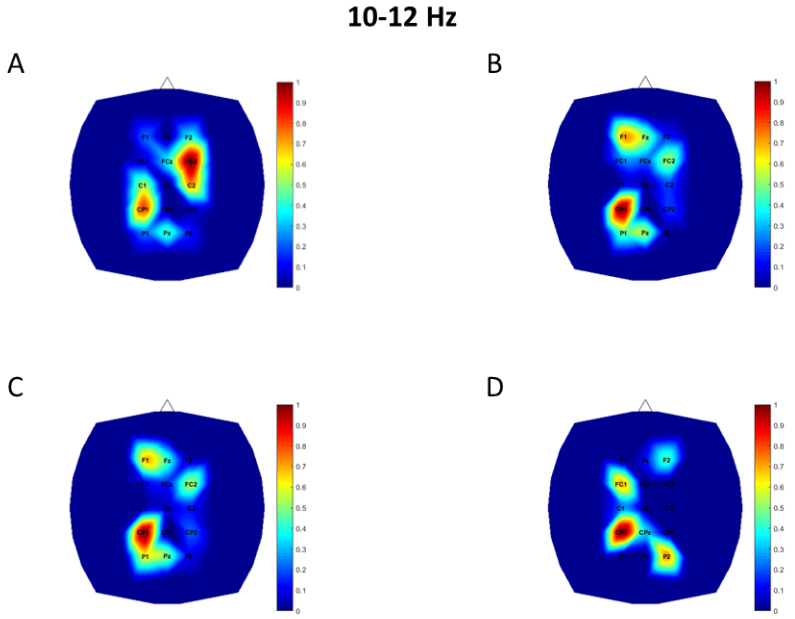

Figure C.3: Topoplots of feature coefficients in the 10-12 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

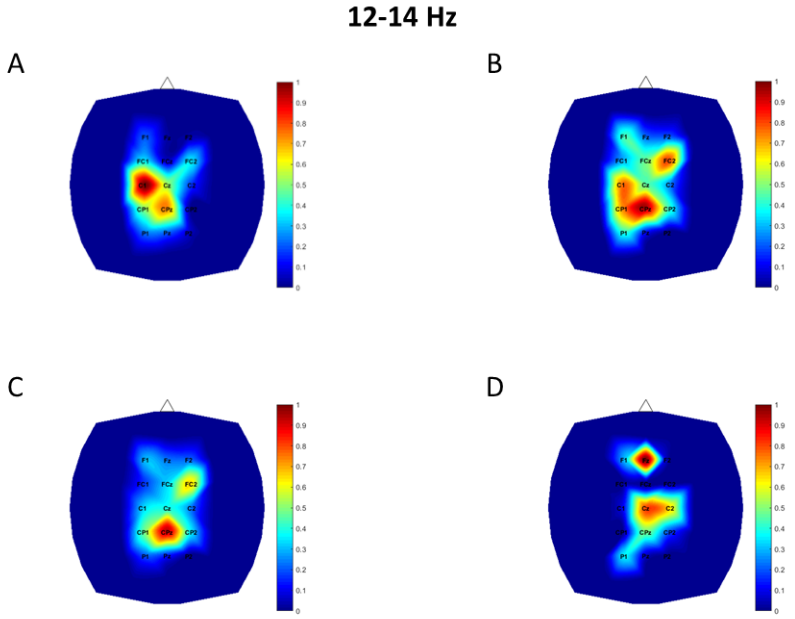

Figure C.4: Topoplots of feature coefficients in the 12-14 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

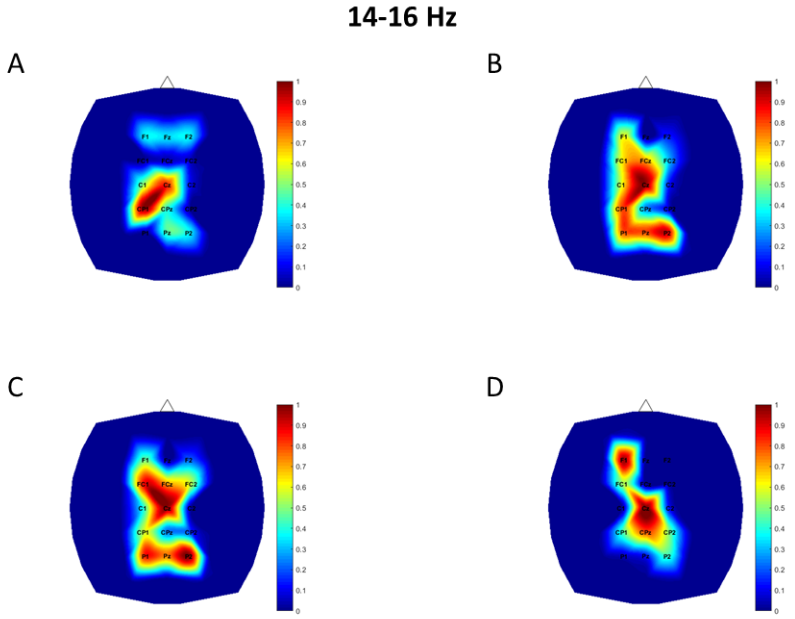

Figure C.5: Topoplots of feature coefficients in the 14-16 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

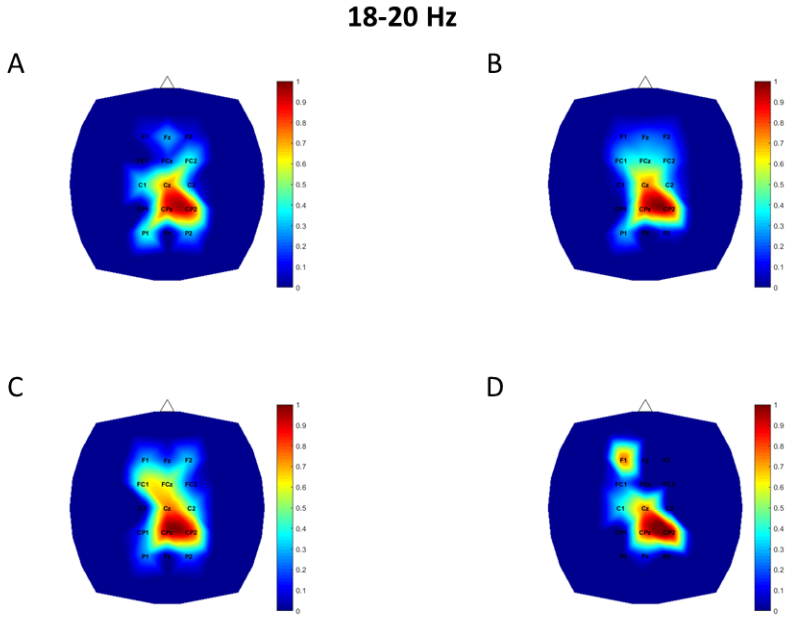

Figure C.6: Topoplots of feature coefficients in the 18-20 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

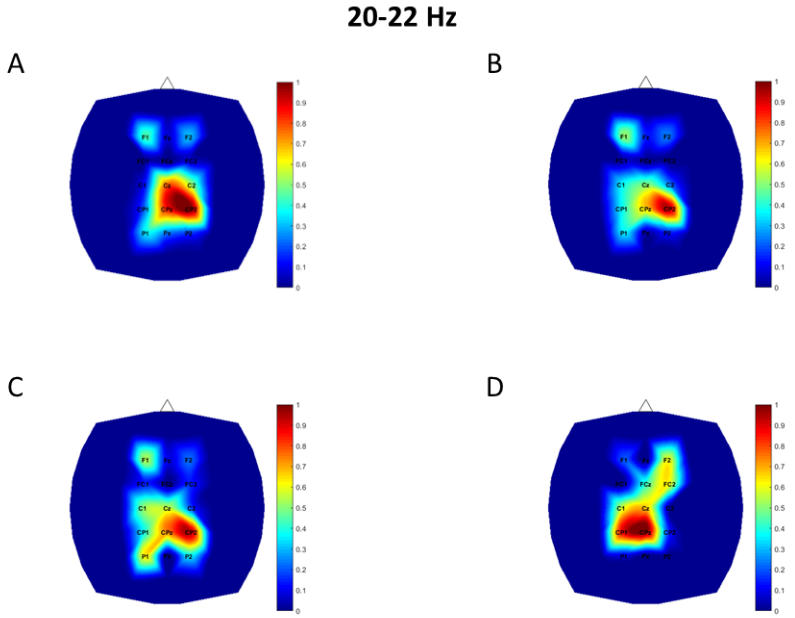

Figure C.7: Topoplots of feature coefficients in the 20-22 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

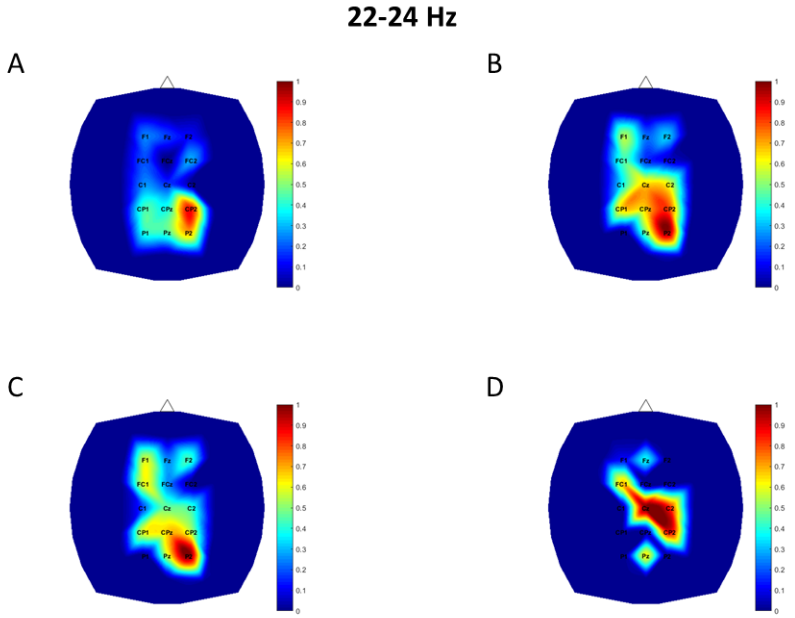

Figure C.8: Topoplots of feature coefficients in the 22-24 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

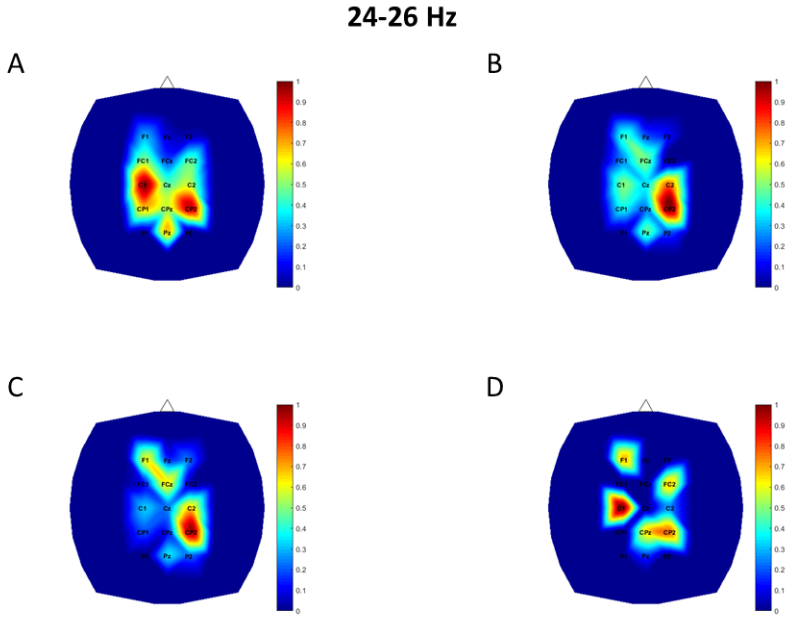

Figure C.9: Topoplots of feature coefficients in the 24-26 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.

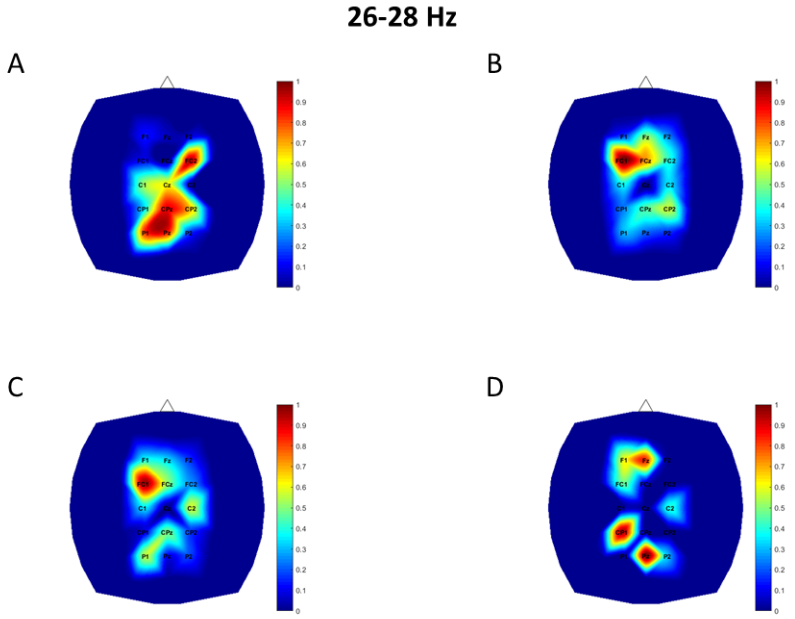

Figure C.10: Topoplots of feature coefficients in the 26-28 Hz band for each algorithm after training. (A) through (C) represent linear coefficients for the (A) CAB, (B) LR and (C) SVM algorithms. (D) is the output obtained using the activation maximization technique used for the FCNN algorithm. All coefficients have been scaled to be between 0 and 1, where the warmer colors on the images represent channels of interest.