EXPLORING MACHINE LEARNING ARCHITECTURES FOR BRAIN COMPUTER INTERFACE SYSTEMS

Master's Thesis

Tien Dat NGUYEN

FAN: nguy1025 - Student ID: 2216627

Supervisor: Dr. Trent Lewis

Submitted to the College of Science and Engineering in partial fulfilment of the requirements for the degree of Master of Science (Computer Science) at Flinders University - Adelaide Australia

August 2020

DECLARATION OF ORIGINALITY

I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Tien Dat Nguyen

07 August 2020

ABSTRACT

Objectives

This study explored the architecture of a machine learning application in a Brain Computer Interface (BCI) system and experimented various classification algorithms in a P300 Speller BCI.

Methods

This study used the P300 Speller BCI datasets from the BCI Competition. The datasets were pre-processed and segmented for analysing and fitting purpose. Wyrm Python toolbox and Scikit-learn Python package were used to help with the exploration. Various classifiers were also experimented: Linear Discriminant Analysis (LDA), Gradient Boosting, Ada Boost, K-neighbours, Decision Tree, Random Forest, Quadratic Discriminant Analysis (QDA), Linear Support Vector Classifier (SVC) and SVC. The predicted characters were compared with true labels downloaded from BCI Competition test results to calculate the percentage of correctly predicted characters.

Results

Main components of a machine learning architecture in a BCI system were explored: EEG signals pre-processing: artifacts removal, features selection, features extraction and predictive model training and results analysing.

For the P300 Speller datasets explored in this study, LDA overall produced better results, with the best score at 94.5% while other classifiers were in a much lower range: <30%, except for the case of Linear SVC with 56% for subject A and 90% for subject B. All those results were observed without Principal component analysis (PCA) and used 64 channels. However, the accuracy dropped when testing with 8 channels, single channel and when testing with fewer trials.

Conclusions

Even though this study only explored machine learning architecture in Speller BCI systems, the knowledge and methodologies can be used to explore P300 BCIs in general. Some classification algorithms were tested, and among those, LDA from Wyrm Python toolbox produced the highest prediction score. However, future work is needed to improve accuracy when using fewer channels or fewer trials.

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my supervisor, Dr Trent Lewis, for his invaluable support and guidance throughout the development of this thesis. His extensive knowledge not just on the research topic but also on various other academic topics and his technical skills are the big inspiration that keeps me going from the start till the end of the thesis. I am also grateful to my friends and colleagues who are always beside me and give me the encouragements and support that I need. Finally, my heartfelt thanks go to my lovely wife who always supports me in all and every matter in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BCI	Brain Computer Interface
EEG	Electroencaphalography
LDA	Linear Discriminant Analysis
SVC	Support Vector Classifier
SVM	Support Vector Machine
ERP	Event-related Potential
fMRI	Functional magnetic resonance imaging
PET	Positron Emission Tomography
fNIRS	Functional Near-Infrared Spectroscopy
BLDA	Bayesian linear discriminant analysis
PCA	Principal component analysis
ICA	Independent Component Analysis
AAN	Artificial Neural Network
CSP	Common Spatial Pattern
SWLDA	Stepwise Linear Discriminant Analysis
QDA	Quadratic Discriminant Analysis

1. INTRODUCTION

In recent years, Brain-computer interface (BCI) has steadily become an attractive research field thanks to the rapid growth of technology and science discoveries regarding the human brain, with more than 100 active research groups and numerous articles on the topic (Konrad & Shanks, 2010; Wolpaw, 2007). For people with severe motor disabilities, Amyotrophic Lateral Sclerosis (ALS), severe cerebral palsy or any other paralysis, they are incapable of any motor functions. In some extreme cases, an individual who is completely paralysed or in completely locked-in syndrome may not even be able to perform any physical movements of the body (Schalk et al., 2004). That leaves the brain as the only way to communicate with the world. BCI systems, in this case, act as a bridge between the patients and the environment around them. BCI systems record the brain activities as electrical signals, process them then produce the closest actions that the patients want as digital commands to control the external devices (Brouwer & Van Erp, 2010; Nicolas-Alonso & Gomez-Gil, 2012). Those commands can be as simple as move wheelchair forward, backward, stop or call some pre-defined phone number. A system as such would be incredibly helpful to people with disabilities and to people who take care of them (Nicolas-Alonso & Gomez-Gil, 2012).

Figure 1: P300 Visual Speller System (Selim et al., 2009)

In order to help the paralysed, the conventional BCI systems usually utilise some visual interface, a cursor or an on-screen keyboard. P300 Speller BCI is one famous example of such a BCI system (Selim et al., 2009) (see Figure 1). However, the idea of based on visual feedback can be improved further, as it has its limitations: force user to maintain visual attention to the screen, and it can be tiresome after some time. Thus, it leads to wonder, a P300 BCI system if done right, can reduce the physical efforted required to interact with the system and can still be comfortable after long time usage. A vibration-based stimulation P300 BCI, a vibrotactile BCI, can be the key to this question. A vibrotactile P300 BCI uses vibrotactile stimuli to provide feedback through the output of sensors (Chatterjee et al., 2007). Users only need to attend to the stimulation on the part of the body where the sensor is located (see Figure 2). This type of interaction eliminates the environmental impact and helps reduce the tediousness of long-time usage.

Figure 2: Vibrotactile stimulation hardware with 16 active EEG electrodes on the cap, EEG amplifier and vibrotactile stimulators (six coin motors) on the neck (Leeb et al., 2013)

On the high level, BCI systems work by taking brain signals collected by some electrophysiological monitoring method, process the data, use machine learning algorithms to best translate the processed signals to digital commands that the patients want to control the external device. To collect brain signals, there are several popular electrophysiological methods being used, such as electroencephalography (EEG), electrocorticography (ECoG), functional magnetic resonance imaging (fMRI), positron emission tomography (PET) or magnetoencephalography (MEG) (Hassanien, 2014), but EEG has gained its popularity over other methods and has been used in various research experiments and applications (Nicolas-Alonso & Gomez-Gil, 2012; Ramadan & Vasilakos, 2017).

Playing an essential role of any BCI systems is machine learning algorithms to process data and classify brain signal patterns and from that convert to computer-ready commands like move forward, backward or stop (Selim et al., 2009). Some well-known algorithms applied in BCI are

Bayesian linear discriminant analysis (BLDA), linear support vector machine (SVM), Fisher linear discriminant analysis (FLDA), generalised Anderson's task linear classifier (GAT), linear discriminant analysis (LDA). According to the performance test carried out by Selim, Wahed and Kadah in 2009 in BCI Competition III, BLDA and SVM achieved the highest accuracy overall for the tested subjects. BLDA achieved on average 75% at 5th sequence, 98% at 15th sequences while SVM scored 75% and 97% respectively (Selim et al., 2009).

Raw brain signals contain not only stimuli response signals but also other bio-signals and noise interference such as muscle signals, eye movements, outside sounds, changes in environment or electricity background noise (Selim et al., 2009). That makes the results of recording brain signal activities more troublesome, as these results are the input for machine learning algorithms. If the data sources are not good and fit for feature extraction purpose, classification methods will not be able to train the model correctly, hence not able to render expected prediction with accuracy and relevance. Therefore, various methods for averaging input data and generalisation are also needed to ensure the essential outcome of machine learning algorithms.

The original approach of the thesis was to explore machine learning architecture in a vibrotactile BCI system. Unfortunately, due to Covid-19 global pandemic situation, all workplaces, universities and schools were closed (Looi & Pring, 2020; Peters, 2020), the research team could not perform testing and data recording for vibrotactile BCI on the Flinders University's laboratory. The BCI Competition III dataset II (hereinafter referred to as P300 Speller datasets) are used instead for the purpose of exploring EEG signals and exploring machine learning algorithms. Even though the features in Speller BCI and vibrotactile BCI are different, they do share the same approach to explore and analyse EEG signals and build up a machine learning application's architecture. Thus, the results from exploring Speller BCI system can be used in future research on vibrotactile BCI system when the research team have vibrotactile datasets ready.

This study highlighted foundation understandings of EEG, BCI, machine learning. It then explored the architecture of a machine learning application in a BCI system as well as tested some popular machine learning classification algorithms on P300 Speller BCI datasets. Finally, the results and future research approaches were discussed.

2. LITERATURE REVIEW

2.1. EEG

2.1.1. Brainwaves

As a BCI system's essential purpose is to translate brain activities to digital commands, it first needs to understand and be able to process brain activities. Neural impulses are generated when neurons received stimuli. Stimuli can be under different forms, such as visual (sight), auditory (sound), tactile (touch) stimuli (Galun, 2012). When neurons received such stimuli, they trigger a series of processes known as depolarisation, repolarisation, and hyperpolarisation (Lerner, 2014). These processes constitute an action potential. An action potential is a temporary change in the voltage between the inside and outside of the neural cell from the resting potential (Lerner, 2014). This change in voltage is very minimal, measured in millivolts (mV), and is approximately 100mV when it changes from the resting membrane potential -70 mV to about +30 or +40 mV . It also handles the task of transmitting information among nerves, produces brief voltage fluctuations that help transmit electrical signals to all brain neurons (Burgess & Gruzelier, 1993). However, as individual neurons can trigger neural impulses to respond to stimuli, there must be a way for the brain to acknowledge and decide on what to do, different neurons must be in communication. Such connections are referred to as clusters or neural networks. When different neurons inside a cluster trigger responses to stimuli at approximately the same time, macroscopic oscillations, or brain waves are produced (da Silva, 1991).

Brain activities are generally characterised by a mix of brain waves. Depending on the physical activities of the human host, one type of brain wave can predominate other brain waves. There are five brain wave frequency bands: delta, theta, alpha, beta and gamma, as shown in Figure 3.

Delta waves

Delta waves range from 1Hz to 4Hz and the slowest brain waves. They are generated in deep meditation and dreamless sleep (Bhat, 2018).

Theta waves

Theta waves are in the range of 3Hz-8Hz. Theta waves are present during sleep and relaxation (Bhat, 2018).

Alpha waves

Alpha waves range from 8Hz to 12Hz. Alpha waves occur when the brain is in resting state (Bhat, 2018).

Beta waves

Beta waves are in the range of 12Hz-38Hz. Beta waves are observed in a normal, conscious state of the brain (Bhat, 2018).

Gamma waves

Gamma waves are in the range of 38Hz-70Hz and are the fastest brain waves and are generated during higher levels of consciousness, mental activities (Tatum IV, Do, 2014) or during muscle activity at 100-200 times the magnitude of brain waves (Whitham et al., 2007).

Image removed due to copyright restriction

Figure 3. Brain wave frequencies (Abhang, 2016)

2.1.2. Neuroimaging methods

As BCI systems convert brain signals to user commands, they need a system to record and keep track of all changes in brain activities. Modern non-invasive neuroimaging methods allow scientists and researchers to address this big task without invading the patients by injecting/implanting the electrodes directly onto the brain. It is especially important in the science of language acquisition and human brain development, as now researchers can study the brain structure and its changes when children learn to talk, read, write and grow up (Kovelman, 2012).

There are two main non-invasive neuroimaging methods: the method that measures changes of electrical signals in brain activities, electrophysiological, and the other that measures changes in blood flow of the brain, hemodynamic (Kovelman, 2012). Because electrical signals are fast, thus electrophysiological methods, such as electroencephalography (EEG) and magnetoencephalography (MEG), provide excellent temporal resolution (milliseconds). However, such methods are much weaker than hemodynamic methods in terms of spatial resolution due to the uncertainty of the location of ERP sources (Luck, 2005). Thus, such methods like ERP are better for research on the temporal aspect of neural activities, not for spatial, location aspect of brain activities.

On the other hand, hemodynamic measures such as fMRI, PET and fNIRS (functional Near-Infrared Spectroscopy) are limited by the blood-oxygen-level dependence (BOLD) response. Thus hemodynamic methods are poorer temporal resolution compared to ERP, range from 2 to 5 seconds (Friston, 2009). However, hemodynamic measures produce a much better spatial resolution compared to ERP, as blood flow changes are slow and sustained. That makes hemodynamic a well-suited method for researching on the location of neural activities (Kovelman, 2012).

Out of those conventional neuroimaging methods, EEG is the most commonly used method for BCIs (Nicolas-Alonso & Gomez-Gil, 2012; Ramadan & Vasilakos, 2017). EEG method is non-invasive, requires relatively cheap hardware, provides high portability and is safer to users. These are very important as one of the issues with applying BCI in real life is that existing systems are usually complicated, expensive and not at a portable level that is convenient to physically disabled users (Figure 1).

2.1.3. Brain signals classifications

The brain control signals can be classified into some main types: Spontaneous Signal, Visual Evoked Signal (VEP) and Hybrid Signal (Ramadan & Vasilakos, 2017). Spontaneous Signals are the brain signals that are generated without external stimulations. Motor and sensorimotor rhythms, Slow Cortical Potentials (SCP) and Non-motor cognitive tasks are some of the most recognised Spontaneous signals (Ramadan & Vasilakos, 2017).

On the other hand, VEP refers to the brain signals that get generated unconsciously when there are external stimulations to the human body. P300 and Steady-state Evoked Potentials (SSEP) are some well-known VEPs (Ramadan & Vasilakos, 2017). P300 is the brain signal elicited approximately 300ms after the subject attends to an infrequent stimulus (Birbaumer et al., 1999; Sutton et al., 1965). P300 and P300-based BCIs will be discussed more in depth in Section 2.2.4. SSEP refers to the steady-state responses that are elicited when attending to external stimuli, can be visual, auditory or somatosensory stimuli (Bera, 2015; Pritchard, 1981). A steady-state visual evoked potentials (SSVEP) BCI typically asks users to attend on one of the visually flickering items, e.g. boxes of letters on screen. SSVEP-based BCI has gained more and more attraction recently, thanks to its less demanding on user-training and high information transfer rate (ITR) (Chen et al., 2015; Gao et al., 2014; Wang et al., 2008). ITR was first described by Wolpaw et al. as a metric to determine the performance of BCI systems (Wolpaw et al., 2002). For comparison, the ITR of Farwell and Donchine's P300 Speller (Farwell & Donchin, 1988) was around 30 bits per minute (bpm) (Gao et al., 2014; Wolpaw et al., 2002), while some SSVEP BCIs can achieve much higher ITR at 105 bpm (Chen et al., 2014).

Different types of signals and their characteristics, advantages as well as disadvantages are described in Table 1 below.

Table 1: Control Signals Summary (Ramadan & Vasilakos, 2017)

2.1.4. EEG

As mentioned in section 2.1.2 above, electroencephalogram (EEG) is the most common method to record brain waves in BCI systems (Kovelman, 2012). EEG was first recorded on human by Hans Berger in 1924, even though the first findings of electrical activities of the brain on animal were by Richard Caton in 1875, Adolf Beck in 1890 (Coenen et al., 2014). EEG is a non-invasive method where electrodes are placed on an EEG cap to capture brain signals. Through the electrodes, the voltage changes caused by different activities within neurons of the brain are recorded under digital waveform as seen in Figure 4.

Since then there are many studies on EEG and its applications in various fields in medical and research (Abiri et al., 2019; Black et al., 2017; da Silva, 1991; Rutkowski, 2016).

Figure 4: EEG Artifacts Detection (Benito Núñez, 2011)

Electrodes are placed on an EEG cap (Figure 5) in different positions that expect signals from respective brain areas in those locations. Electrodes are typically small devices, usually metal, that conduct the pathway for potential electrocortical from the scalp to a recording machine. Electrodes can be wired or wireless, can be dry or wet - require some special kind of gel to help boost up the signal transmission (Chi & Cauwenberghs, 2010; Grummett et al., 2015; Lopez-Gordo et al., 2014; Mathewson et al., 2017). When the electrical waves reach the electrodes on the EEG cap, those electrical waves can push or pull electrons on the metal part of the electrodes. Each pair of electrodes are connected to a differential amplifier. As the voltages of original brain waves are insignificant, those amplifiers magnify the voltages so the output can be recorded. Typically, EEG signals of an adult range from 10μV to 100μV (Aurlien et al., 2004). An integrated voltmeter is used to measure the fluctuations of push-pull between any two electrodes over a period of time, the result of this process is the EEG (Tatum IV, Do, 2014) as seen in Figure 4.

Figure 5: EEG caps (Marini et al., 2019)

There are a few international system standards for how to position EEG electrodes to achieve best EEG result, such as 10/20, 10/10 or 10/5 systems (Jurcak et al., 2007). While 10/20 system has been in use since the early days for a relatively small number of electrodes, 21 (Klem et al., 1999), the newer extensions namely 10/10 and 10/5 systems were proposed to solve the problem of having many electrodes, up to 81 for 10/10 system and more than 300 electrodes for 10/5 system, and higher spatial resolution (Jurcak et al., 2007). Locations of electrodes in 10-20 and 10-10 systems can be seen in Figure 6. Electrodes are labelled by the area of the brain that it is positioned to read data from, Fp for pre-frontal, F for fontal, T for temporal, P for parietal, O for occipital, C for central. “z” is also used for reference or measurement. For numbers, even numbers are for electrodes positioned on the right side of the head while odd numbers are for electrodes positioned on the left side of the head (Jurcak et al., 2007). It’s worth noting that for extension systems like 10-10 or 10-5 systems, a new naming system called Modified Combinatorial Nomenclature (MCN) is used to name the positions of the electrodes with a few new letters like AF, FC, FT, CP, TP and PO for intermediate positions (Acharya et al., 2016).

As mentioned above, EEG has numerous advantages when compared with other methods such as excellent temporal resolution (really fast), more economical as costs for hardware are usually lower (Vespa Paul et al., 1999), completely non-invasive as the electrodes are placed on the cap to be worn on the head, not like ECoG electrodes are implanted on the cortical surface of the brain , requiring medical operations. However, EEG has its disadvantages and limitations, such as low spatial resolution, setup time for the recording takes time as many electrodes need to be cleaned, applied gels or pastes and placed onto the precise locations of head areas to maximise accuracy and minimise noises of the signals recorded. One more weakness of EEG method is that all the signals come through the electrodes, there can be other biological artifacts, or noises produced together with the stimuli signals, make it hard to differentiate which one is the genuine responses of the brain to stimuli. This is known as the signal-to-noise ratio (Schlögl et al., 2002), signal-to-noise high means there are too many noises. As such, more efforts for pre-processing and analysing raw data coming from EEG method are necessary to extract useful data for the BCI systems.

Image removed due to copyright restriction

Figure 6: EEG electrode 10-20 system (left) and EEG electrode 10-10 system (right) (Benito Núñez, 2011)

As can be observed in Figure 4, EEG method recording can produce unwanted EEG signals, known as EEG artifacts, such as muscular or eye blink artifacts (Burgess & Gruzelier, 1993). This is because the electrodes record all the fluctuations in electrical signals happening inside the brain, which may contain several undesirable signals from both physiological and extra-physiological sources (Burgess & Gruzelier, 1993). Those unwanted signals can be the movement of body parts (muscular artifacts), cardiac artifacts or environmental artifacts like power line electromagnetic or fluorescent lights (Thorp & Steinmetz, 2008). These noises need to be filtered out so the output EEG signals can be as clean as possible to be credible input of BCI systems.

2.1.5. Independent Component Analysis

There are several approaches to help with the above problem of EEG artifacts, such as Independent Component Analysis (ICA) (Delorme et al., 2001; Whitmer et al., 2010), Principal Component Analysis (PCA) (Abdi & Williams, 2010) and regression methods (Makeig et al., 1996). Among those approaches, ICA has proven to be more reliable and provide better quality than PCA and regression methods (Brown et al., 2001; McMenamin et al., 2010; Plöchl et al., 2012).

Independent component analysis (ICA) is a signal processing technique to be used as a common approach to reduce noises from EEG output by finding the hidden independent components. Using ICA can help separate the different components/sources produced in EEG output so that the neural and non-neural activities are distinguishable for further processing, typically ignore the non-neural activities and only keep neural ones (Delorme et al., 2001).

2.1.6. Principal Component Analysis

Principal component analysis (PCA) is a commonly used multivariate technique in EEG processing for dimensionality reduction. PCA accomplishes that by analysing the similarities between data points from given datasets to extract distinctive information, only keep the important information, describe them by inter-correlated quantitative dependent variables and represent them as principal components (Abdi & Williams, 2010). After PCA, the data will be represented by a lower dimension space, and this helps all further processing and classifications perform faster in both time and space (Abdi & Williams, 2010).

2.2. BCI

2.2.1. BCI Overview

Brain-computer interface (BCI) system is a combination of hardware and software to form a communication channel between the brain and the outside world without any human interference. The first research on BCI was carried out by researchers at the University of California and first published in 1973 (Vidal, 1973). It was also the first time the term “Brain-computer interface”, or BCI, was used to address the study subject, making its author, Professor Vidal of University of California widely accepted as the inventor of BCI. Since then, BCI has become one of the most attractive fields in research and science (Haider & Fazel-Rezai, 2017), mostly in laboratories, to help people with severe motor disabilities, Amyotrophic Lateral Sclerosis (ALS), cannot move any parts of the body. BCI systems support patients by allowing them to use only their brain to “control things”, or in more details, use their brain to think or react to some stimulus, then the BCI systems pick up the changes in their brain signal via some recording method, pre-process the brain signal data to extract useful features and use machine learning algorithms to predict the commands that patients expect. A general diagram of how a BCI system works is described in Figure 7.

2.2.2. BCI system structure

One BCI system typically consists of following major parts: signal acquisition and conditioning, feature extraction and classification, feedback loop and an external device (Edlinger et al., 2012).

Signal acquisition refers to the process of recording brain signals (Edlinger et al., 2012). One example of BCI system is described in Figure 2, a vibrotactile stimulation hardware with 16 active EEG electrodes on an EEG cap, an EEG amplifier and vibrotactile stimulator (six coin motors) on the neck. All these hardware and setup are for recording EEG signals through the electrodes on the cap, then those EEG signals will be amplified and digitised. The vibrotactile stimulators in this example are used for triggering the stimuli, which in turn trigger the responses from the brain.

Image removed due to copyright restriction

Figure 7. Diagram of a BCI system (Wolpaw et al., 2002)

Signal processing consists of two parts: feature extraction and translation algorithm (Wolpaw et al., 2002). EEG signals output of the signal acquisition step are the input of this step where the signals are passed through to the BCI hardware, usually a computer, with algorithms configured to pre-process data, remove noises, extract features and train the classifier by the processed EEG data (Edlinger et al., 2012). All these steps are done in the machine learning component of the BCI system. The output of this step is a digital command to be sent to the external device to perform actual actions.

When the machine learning component produces a predicted digital command, it sends that command to the external device so that the device can perform the action that the user wants (Edlinger et al., 2012). This external device can be a wheelchair or a screen, and it is pre-programmed to respond to actions such as calling a phone number, moving the wheelchair forward or typing a character on a screen.

2.2.3. BCI software platforms

There exists many publicly available software platforms to support the development of BCI. Some commonly used are BCI2000, OpenViBE, TOBI, BCILAB, BCI++, xBCI, OpenBCI and BF++. Those platforms include algorithms and modules necessary for BCI systems like data acquisition, signal processing, a wide range of classification and regression methods. A summary of those BCI software platforms is available in Table 2.

Table 2. BCI Software platforms (Ramadan & Vasilakos, 2017)

Table removed due to copyright restriction

2.2.4. P300-based BCI

P300 Event-related Potentials (ERP)

P300 ERP was first mentioned by (Sutton et al., 1965) as the positive deflection of voltage in EEG recording that occurs approximately 300 milliseconds after a stimulus is triggered (Birbaumer et al., 1999; Sutton et al., 1965). There have been many research since then on P300 ERP to explore its characteristics and further utilise it on studying the human brain, especially to apply in BCIs (Haider & Fazel-Rezai, 2017; Hoffmann et al., 2008; Krusienski et al., 2008; Polikoff et al., 1995; Sellers & Donchin, 2006).

P300-based BCIs allow users to interact with BCI systems by responding to external stimuli, either visual, auditory or tactile. When the user focuses on the stimulus, the brain signal responses usually occurs roughly 300 milliseconds after the stimuli triggered, which can be easily detected on the EEG reading. It was observed that in some cases, the latency after the triggering of the stimuli could be in the range from 250 to 750 milliseconds (Polich, 2007).

Oddball paradigm

As mentioned above, the P300 ERP can be observed from EEG recording, but only when the subject focuses (actively engages) in the target. That can be focusing on character “A” on a matrix of characters on screen in a Speller system (see Figure 1 for Speller BCI), or focusing on the left-hand vibration pad instead of the right-hand vibration pad, depending on the target of the required task. How oddball paradigm works is that an infrequent target is presented on top of frequent standard stimuli in the background (Polich, 2007). This procedure is referred to as oddball paradigm (Kaufmann et al., 2013). When the subject responds to the target stimulus and not the standard stimuli, the P300 ERP can be observed by time-locking the target stimulus with EEG signals and measuring the amplitude in µV and the latency in milliseconds (Polich, 2007).

Visual BCIs

Visual BCIs are more popular and attract more researches compare to the other types. Visual BCIs usually work by showing the task on a screen, can be a matrix of characters or a list of images. P300 Speller BCI is the most used application of Visual BCI where a matrix of characters appears on a screen (see Figure 1 for an example of a Speller BCI). The characters then will flash by a group of columns and rows (Farwell & Donchin, 1988). User will be asked to only focus and respond on only one character at a time before the BCI system changes to a new one. Figure 8 shows an example of P300-evoked EEG signal recorded in P300 Speller system.

Figure 8: P300-evoked brain signal response measured in P300 Speller system (Yoo, 2017)

Such Visual BCIs usually achieve higher prediction accuracy than auditory or tactile BCIs (Chang et al., 2013). However, it brings up several problems such as it requires patients to look at the screen for a long time while scanning the rows and columns. That causes nauseous and makes patients feel isolated from the world as they cannot take their eyes off the screen (Brunner et al., 2010), especially for Locked-in Syndrome users who are in total paralysis except that they can control only vertical eye movement (Bauer et al., 1979).

Auditory BCIs

Auditory BCI systems use sound with different volume, pitch and direction as the stimuli (Caglayan & Arslan, 2012; N J Hill et al., 2005). User will be tasked to only respond to a specific sound volume, pitch, direction or combination of those configurations. Auditory BCIs can solve the gaze-dependent problem that Visual BCIs have, but at the current state, auditory BCIs have not reached the accuracy level as of Visual BCIs (Chang et al., 2013), even though they are getting more attention in the research field (Caglayan & Arslan, 2012; Chang et al., 2013; N J Hill et al., 2005).

Tactile BCIs

Tactile BCIs utilise the tactile sensory by having tactors placed on the skin of the body. The stimuli are through triggering a touch-like stimulus, e.g. a vibration pad vibrates on the left arm, neck, hands (Brouwer & Van Erp, 2010; Kaufmann et al., 2013). User is tasked to focus on a target location (e.g. left arm) and ignore the rest. A prominent sub-type of Tactile BCI is Vibrotactile BCI.

Vibrotactile BCIs

A vibrotactile BCI uses vibration stimuli to provide feedback through the output of sensors and is gaze-independent (Chatterjee et al., 2007), patients do not need to focus continuously on some screen but only need to attend to the vibration stimuli (Figure 2). An example of vibration stimuli setup is in Figure 17 where four vibration pads are attached on a user's hand with hardware to control the triggering of such vibration. Each pad is programmed as a representation of an action the user wants to perform, for example: to spell a character, call a phone number or to move wheelchair forward. All the vibration stimuli are recorded and time-locked together with EEG signals to generate final EEG datasets to be used in BCI system to detect the EEG signals' deflections on pair with the vibration stimuli. From that, the BCI system can determine which sensor pad the user most likely focused on, which translates to which action he wanted to perform.

2.3. Machine Learning

2.3.1. Machine Learning Overview

Ever since the invention of the computer and the rapid development of computational technologies, the question "Can machines think?" had been around for centuries. Alan Turing described characteristics of a thinking machine and the implications in building such machine in his paper in 1950 (Harnad, 2006). Even though Arthur Samuel first used the term Machine Learning (ML) in 1959 (Samuel, 1988), but a more formal and accepted definition of machine learning was from a computer scientist named Mitchell at Carnegie Mellon University (Mitchell & Learning, 1997). In Mitchell's definition, "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell & Learning, 1997). Machine learning is considered a subset of artificial intelligence which utilises various algorithms to build a model using training data, to achieve the outcome of making predictions while there is no explicit command programmed to perform the task (Bishop, 2006; Samuel, 1988).

An example of machine learning is to determine if an image is a dog or a cat. A selection of images of dog and cat will be collected for the training purpose. There are several types of training such as supervised learning where the training data contains both the image and its label (cat or dog); unsupervised learning where the training data only contains the image, not the label; or semi-supervised learning where the training data is incomplete, a part of the data does not have the output with it. Each training method works differently, but the common output is that the input data set is provided so that machine learning algorithms can create a mathematical model. This model can predict if any future input image will be a dog or a cat with some probability level in percentage. See Figure 9 for an example of a machine learning algorithm applied in a mobile application to predict if an image is a dog or a cat.

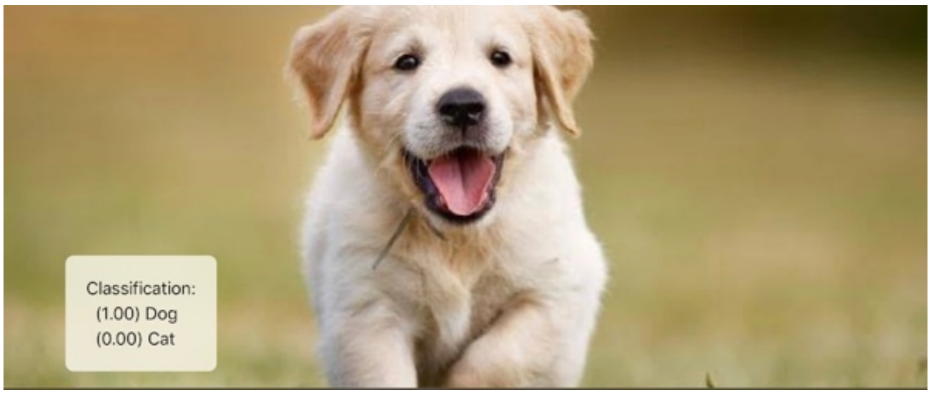

Figure 9: An example from an implementation of Machine Learning in an iOS application by the author to classify whether an object in an image is a dog or a cat. The machine learning algorithm predicted a Dog from the example image with probability 100%

Machine learning and data mining are two terms that usually confuse many adopters and researchers who show interest in the artificial intelligence area. They share some commonalities such as they use roughly the same methodology to produce the result, including pre-processing input data, using algorithms to classify data and create a model using the training data and finally yield the prediction or decision result. Machine learning uses some of data mining methods for unsupervised learning, and data mining also uses various machine learning methods. However, there are a few fundamental differences between the two in the goal and knowingness of the training data. Data mining works with unknown features (properties) from the training data and focuses on the discovery part . The knowledge and relationships between objects, properties in data mining data set are usually not yet defined, hence the need to discover any useful information that can infer how two or more properties of the data link to the others. On the other hand, machine learning works with known features to reach the goal of delivering a prediction or decision.

Machine learning and its applications have been growing dramatically in numerous fields, including both science and commercial usage. Some applications include computer vision to recognise text and object from image, video; speech recognition to transcribe speech (audio) to text, even translate in real-time; detect credit card fraud, behaviour forecasting, and much more. Almost all industries make use of machine learning at some level to perform predictive analysis tasks to assist in making faster, better decisions. Agriculture uses Precision agriculture concept to optimise decision support system (McBratney et al., 2005). Financial markets use predictive modelling system to predict prices or trading (Qiao & Beling, 2016). BCI systems use machine learning to translate brain signals to digital commands to control external devices to help patients with severe motor function disability, like a P300 Speller system (Figure 1). With the growth of technology and the era of Big Data, more and more massive datasets with both structured and unstructured data need machine learning to generate useful predictive analysis.

Some more examples of Machine Learning being used in nowadays daily life are Google Mail's spam detection. Complicated machine learning algorithms are used by Google engineers to classify if one coming email is considered a spam email or not and send it to the junk folder if the predicted outcome is yes. Google Mail even learns from user's behaviour and decisions, that if it detected an email as spam but user later un-spams it, the algorithms take this decision as new input for training the model. Another application of machine learning that people use every day is the intelligent assistant on most iPhones, Siri. The machine learning algorithms are well integrated into Siri service, users only need to talk to Siri, and it intelligently and correctly recognises the commands, even though the voice commands can be different each time. For example, "What time is it", "tell me time", "what is the current time" all can be understood the same way by Siri. All these are possible by using machine learning techniques. In the e-commerce websites, one thing surfers will usually see is a "Recommendation List" section or "Inspired by your browsing history" list as Amazon.com names it. In this section, the websites based on the user's web activities (how long they stay in one page, scroll how many times, the mouse is in which area of the website, visited which types of products) then use machine learning algorithms to analyse and predict the list of products with high probability that users will purchase.

2.3.2. Machine Learning Paradigms

There are typically three main types that are being used widely: supervised learning, unsupervised learning and reinforcement learning:

Supervised learning

Supervised learning means the input data set, or training data set, are labelled with the expected outcome. With both input X, output Y provided and the label mapping from X to Y before the learning, the task of a supervised learning model is to learn the "how" of the mapping in order to infer the testing data set (Alpaydin, 2014). Take the example in Section 2.3.1 about machine learning algorithm to predict an image is a dog or a cat. For supervised learning, the training set will include both the images of dogs and cats, together with the labels clearly stating each of the images is a dog or a cat. The outcome is included in the training data set. The classifier learns that to create the predictive model. Then this new model can be used to predict and tell if any new image is a dog or a cat (Figure 9). Supervised learning algorithm includes classification for use when an object needs to be categorised into a specific group of a set of groups, and regression for when estimating a relationship between a dependent variable and at least one independent variable (Alpaydin, 2014).

Unsupervised learning

On the contrary to supervised learning which has all the objects labelled, unsupervised learning does not have the outputs, just the objects. From the objects, unsupervised learning algorithms analyse the data, find similarities and categorise data into groups or clusters (Alpaydin, 2014). Base on such identified similarities and categorised clusters, new objects are analysed and predicted based on how many similarities they share with the predictive model. In unsupervised learning methods, data mining techniques are usually employed to analyse data to find hidden patterns or unknown relationships between properties of data.

One interesting example of unsupervised learning model is finding the most potential group of customers from an e-commerce website's data (Alpaydin, 2014). The website has information on customers, their transactions history and their buying behaviours. In this case, there are no clearly defined outputs such as finding the top age range or top cities of the customers. The task of the unsupervised learning model is to analyse all that information and to find the hidden relationships between all the demographic information and purchasing behaviours (Alpaydin, 2014). It can be customers who are in a specific age group, are working in a specific occupation, living in a specific part of the city, or any combination of those with any other information.

Reinforcement learning

Another machine learning model is reinforcement learning where the output is not a labelled class or clusters of data input but a sequence of actions. In this case, the outcome of a sequence of actions is more important than one single action alone (Alpaydin, 2014). One interesting application of reinforcement learning is Game playing, chess game, for example. The rules of chess game are simple to play, but the vast number of possibilities that each move can generate make it difficult for algorithms to play well. After every move, the algorithms will have to calculate all the possible moves after that and the likely outcomes towards winning or losing. That process repeats after each turn because the game state can change dramatically after that turn. In such case, the reinforcement learning model needs to calculate all possible moves ahead not just one single move and choose the best outcome towards winning.

2.3.3. Machine learning popular algorithms

As a result of learning from the training dataset, a mathematical model is created to make predictions. There are different ways the machine learning models can be created by how they interact with the input data and how the learning process is. Below are some commonly used machine learning algorithms.

Linear Regression

Linear regression models relationship between a dependent variable and independent variables by a linear approach (Stigler, 1981). The simplest case is simple linear regression when there is only one predictor (independent) variable and only one response (dependent) variable (Godfrey, 1985). When there are more than one predictor variables, the multiple linear regression can be used to analyse the relationship among those independent variables and predict the response approach (Breiman & Friedman, 1997).

Figure 10. Regression Algorithms (Brownlee, 2019)

Logistic Regression

Logistic regression uses a logistic function to model the relationship between a categorical response variable and predictor variables (Berkson, 1944; Cramer, 2002). An example is to predict if it will rain tomorrow. This is a binary logistic regression where the response variable only has 2 possible values: rain or not, pass or fail grade, a picture of cat or dog, etc. There are other types of logistic regression: multi-nominal logistic regression and ordinal logistic regression. For multi-nominal logistic regression, categorial response can be of 3 or more possible values and there is no ordering among the values (Cramer, 2002). Some examples include predicting what character from a handwriting or predicting what University a student will choose based on his/her age, gender, average grade. In those cases, there is no ordering among the response values. However, for ordinal logistic regression, categorical possible values are in order (Cramer, 2002). Predicting the rating a consumer will rate a restaurant (1-10) or predicting the final exam grade (High Distinction, Distinction, Credit, Pass, Fail) are some examples of ordinal logistic regression.

Decision Tree

Decision tree learning takes use of a decision tree data structure as a predictive model to reach the prediction (Breiman, 1984; Friedman et al., 1996). A decision tree is an organisation of tree-like items that includes all the decisions and their outcomes, including chance events, costs (Breiman, 1984). It represents a path from general to more specific conditions, like the way that humans make decisions. In a decision tree, all options, events and chances are nicely visualised, make it easy for the new prediction to walk through the tree step by step (Kretowski, 2019). Given its intelligibility and simplicity, decision trees are ranked one of the most popular machine learning and data mining algorithms (Holzinger, 2015; X. Wu et al., 2008).

Image removed due to copyright restriction

Figure 11. Decision Tree Algorithms (Brownlee, 2019)

Random Forest

Random Forest is an ensemble machine learning method which produces the output (either for a classification or regression problem) by constructing a multitude of many decision trees and aggregating those decision trees (Ho, 1998, 1995). In Random Forest, random sampling of subsets of data are used and processed in parallel, called Bagging, whereas in the other ensemble technique, Boosting, one decision tree learns from one other tree to determine the features and the prediction results (Breiman, 1996; Ho, 1995; Opitz & Maclin, 1999).

Bayesian Algorithms

A Bayesian Network is also referred to by Bayes Network, Belief network or probabilistic directed acyclic graphical model. A Bayesian Network bases on several inter-influential variables and predicts the result based on probabilities (Bell, 2020). For example, a Bayesian network can be used to predict the probabilities of some diseases based on some set of symptoms.

Image removed due to copyright restriction

Figure 12. Bayesian Algorithms (Brownlee, 2019)

SVM (Support Vector Machine)

SVM works by using the input dataset to teach the algorithm to classify unseen data point. Input dataset will be classified into classes by plotting as points in a n-dimension space (with n being the number of features of the dataset) and finding a hyperplane to separate the data points of different classes (Boser et al., 1992; Cortes & Vapnik, 1995; Drucker et al., 1997). SVM algorithms are widely used in various applications such as text categorisation, hand-writing recognising, image classification, stock market forecast (Anaissi et al., 2017; Fu et al., 2014; Hao et al., 2009; Z. Li et al., 2002; Qi et al., 2014; X. Wu et al., 2008; M.-D. Yang et al., 2010).

K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN) algorithm works by storing all the input data points and classifying the new unseen data directly by the closest class among its nearest k input data points (neighbours) (Altman, 1992; Dudani, 1976; Y. Yang, 1999), hence the name. Due to its nature of not pre-training data, KNN consumes large computational power when classifying new data and may have bias problem if the class distribution is heavily unbalanced (John Lu, 2010). However, normalisation the input data and assigning weights to each point based on its distance to the new point can help improve its accuracy and performance considerably (John Lu, 2010).

Image removed due to copyright restriction

Figure 13. K-Nearest neighbours algorithm (Brownlee, 2019)

Dimensionality Reduction Algorithms

Dimensionality Reduction algorithms' task is to reduce the number of features, or variables, before going through the fitting and training process (Alpaydin, 2014). The input data may have too many variables that are not correlated to the problem. Therefore, they are redundant and should be removed. By doing that, the more important features that correlate and contribute to the classification or regression problem are kept. Some popular dimensionality reduction algorithms include Independent component analysis (ICA) and Principal component analysis (PCA) which are discussed in sections 2.1.5 and 2.1.6 respectively.

Image removed due to copyright restriction

Figure 14. Dimensionality Reduction Algorithms (Brownlee, 2019)

Gradient Boosting

Like Random Forest, Gradient Boosting is an ensemble machine learning algorithm and it works by combining the results of other estimators/prediction models to produce a better one. However, while Random Forest builds model through independent trees, Gradient Boosting adds a weak model iteratively to a stage-wise additive model to continuously optimise/minimising loss function (Friedman, 2001; Mason et al., 2000). Gradient Boosting follows greedy approach, once a new weak model is added, all previous models stay untouched and are never changed again.

Artificial neural network (ANN)

Artificial neural network (AAN) models how the neurons in a brain connect and work with each other. Hecht-Nielsen defined AAN as "a computer system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs" (Hecht-Nielsen, 1989). The AAN network is essentially a network of inputs and outputs, much like the neurons in a brain connecting with each other, forming tens of billions of interconnected structures (Bell, 2020). Artificial neural networks are used for a variety of different tasks due to its capacity to process large data volume with high speed such as computer vision, speech recognition or medical diagnosis (Bell, 2020).

One popular approach takes use of AAN is Deep learning, which comprises of multiple hidden layers in an AAN (Lee et al., 2009).

Figure 15. Left: Artificial Neural Network Algorithms. Right: Deep Learning Algorithms (Brownlee, 2019)

2.4. Machine learning in BCI

In EEG-based BCIs, the training data set is the recorded EEG signals. These signals contain both EEG signals and different noises such as muscle signals, eye movements, outside sounds, changes in environment, electricity background noise (Selim et al., 2009; Vidal, 1973). Classifying these mixed signals is a challenging problem, not just because of the noises but also because the signals vary from subject to subject. It means some subjects produce EEG signals clearer than others. That causes issues for the machine learning algorithms as the training data is either too noisy or too biased, underfit or overfit. To deal with this problem, various techniques have been employed to process and classify raw EEG signals.

For the motor-imagery BCIs, one important part of processing data is to determine the spatiotemporal filters which later will be used as feature extractors (Müller et al., 2008). Common spatial pattern (CSP) is the most commonly used and very efficient algorithm for this exact purpose. To get to that point, CSP has attracted a lot of research effort on extending and improving the algorithm (Lemm et al., 2005; Ramoser et al., 2000; Tomioka et al., 2007). CSP algorithm works by finding and separating multivariate signals into subcomponents which maximise the variance of signals of one condition and minimise variances of another (Koles et al., 1990). Using CSP, Ramoser et al. were able to get an accuracy level of more than 90% on single-trial EEG measurements in a motor-imagery-based BCI (Ramoser et al., 2000).

However, (J. Li & Zhang, 2010) pointed out that CSP does not work well with EEG classification problems in motor-imagery BCIs. Li & Zhang listed a few shortcomings, most importantly that CSP is more a decomposition technique than a classification one, and CSP usually requires a large number of channels, which is not ideal. Li & Zhang also proposed a new tensor-based scheme to help with the discriminative feature extraction. In their proposal, a regularised tensor discriminant analysis (RTDA) algorithm is used to extract multi-way subspace from EEG data. The result was auspicious that the number of used channels can be reduced to only two channels with close to lossless in performance (J. Li & Zhang, 2010).

Even though various researches have been done to test the effectiveness of motor-imagery BCIs and have some positive results (Kübler et al., 2005; Lemm et al., 2005; J. Li & Zhang, 2010; Pfurtscheller & Neuper, 2001), motor-imagery BCIs didn’t work well with subjects who were completely locked-in according to research done by (N Jeremy Hill et al., 2006). There are more and more efforts spent on the P300 event-related potential control-signal to improve detection ability or develop new applications (Krusienski et al., 2008; Piccione et al., 2006; Polikoff et al., 1995; Sellers & Donchin, 2006). Hoffmann et al. in 2008 tested a P300 paradigm using six different images flashing and various number of electrodes (4, 8, 116, 32) (Hoffmann et al., 2008). The authors used Bayesian Linear Discriminant Analysis (BLDA) and Fisher’s Linear Discriminant Analysis (FLDA) as classifiers. Even though FLDA, due to the small sample size problem, showed poorer result compared to BLDA, but both of the classification methods still achieved relatively higher than other P300 BCI systems for disabled users. Hoffmann et al. also tested the predictions with various configurations for electrodes, with 4, 8, 16, and 32 electrodes. For FLDA, accuracy slightly decreased when there were more than 8 electrodes in use. On the opposite, BLDA algorithm saw a small increase when there were 16 or 32 electrodes.

Borghini et al. even though did not test as low as 8 electrodes, but also proved that decreasing the number of electrodes from 64 down to 19 and the accuracy outcome still maintained of 99% (Borghini et al., 2014). Research done by Fukuzumi et al. also followed this approach and found that from the 32 electrodes, only 7 electrodes had remarkable reactions to evoked related potentials, CP5, CP6, T5, T6, PO3, PO4 and any one electrode in the remaining. It means that dataset from only 7 electrodes should be enough for the training and classification purposes (Fukuzumi et al., 2014).

To summarise from these results, it is proven that for P300-based BCIs, the number of electrodes can be as low as 8 and still can achieve good enough accuracy.

One of the advantages that P300-based BCIs have over other BCI systems is that it requires less time of the subjects in training. A patient with disabilities on a wheelchair, they communicate with the wheelchair by a BCI system. They will not want to wait for minutes or hours while the system processes data, trains the model, gives prediction then executes commands to move the wheelchair forward or backward. Applications like this require a BCI system to be fast in real-time without significant delay or long training period. Variety of researches have been pushing the theory and practical applications of single-trial extraction (Daly et al., 2010; Hoffmann et al., 2008; Lelievre et al., 2013; Müller et al., 2008). The results have been promising. Lelièvre et al. used linear support vector machine (SVM) classifier in a single trial P300 classification on a moving sound stimuli BCI; and achieved significantly higher accuracy in comparison with the classic stepwise linear discriminant analysis (SWLDA) classifier (Lelievre et al., 2013) (see Figure 16).

Figure 16: Single-trial BCI-Speller classification using Linear SVM classifier vs classic SWLDA classifier (Lelievre et al., 2013)

In the vibrotactile BCI field, Yao *et al.* conducted an interesting experiment to test the performance between imagined sensation and vibrotactile stimuli BCI systems where one group of subjects performed some task (focus attention on a particular body part) with assistant from actual vibrotactile stimulation. The other group performed the task totally by imagining it, there was no stimulation involved (Yao et al., 2019). The result indicated that the subjects who imagined, or thought, recorded a comparable accuracy with the group with actual stimulation, both reached 75%. The advantage of imaged-sensation-based BCIs is that it is free and independent from stimulation. Users only need to imagine or think about it. The result of this research can be the foundation for future research to improve the accuracy further and make more applications out of this new BCI paradigm.

Even though the study above suggests some interesting direction for future approach, but more studies need to be undertaken to prove its reliability before there can be real-life applications. Up to date, there are not many studies on vibrotactile stimulation BCI systems. One of the reasons may be that visual P300 type is easier to engage and because P300 visual Speller is the most popular application of P300 BCIs. However, for P300 Speller system, users must look at the screen for a long period. After some time, users can get fatigued or tired because of too much eye movement. That gives opportunities for future research to work more on vibrotactile P300 BCIs, especially in the machine learning perspective.

3. METHODOLOGIES

3.1. Original Vibrotactile BCI approach

As mentioned in Section 1, the original idea behind this study was to explore machine learning aspect of vibrotactile BCIs. The procedure of the research was described in Figure 18.

The recording of EEG was planned to take place in Flinders University Laboratory facility inside a Faraday cage (see Figure 17). Vibration tactors would be placed on the subject's arm. The monitoring/capturing hardware and the computer outside of the cage would capture the data coming from inside the cage. This setup would allow monitoring and recording EEG data for the experiment with as little as possible contamination to the data. After that, EEG signals recorded and time-locked with vibration triggers would be recorded in Matlab application (Matlab, The Mathworks, Natick, Massachussetts, USA). The output of this process would be ".mat" files containing the EEG recorded together with mapping labelled targets/nontargets to be used as training datasets for machine learning algorithms.

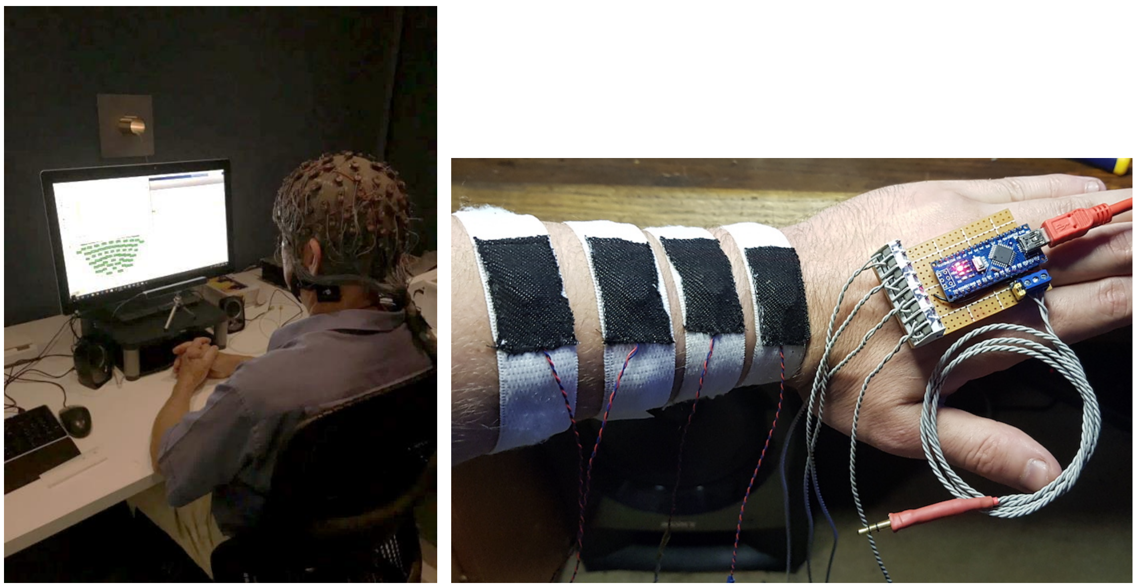

Figure 17. Left: Signal acquisition setup in Faraday cage located at Flinders University. Right: Placement of vibration tactors on subject's arm

The focus of this study was to use machine learning algorithms to explore EEG signals and build prediction models to predict the correct tactor that the subject paid attention to. As mentioned in Section 1 about the Covid-19 situation preventing the research team from carrying out such Signal acquisition, the P300 Speller datasets were used because they share the same data type ".mat" and the analysing and processing of the EEG data would be closely related since they both base on P300 oddball paradigm.

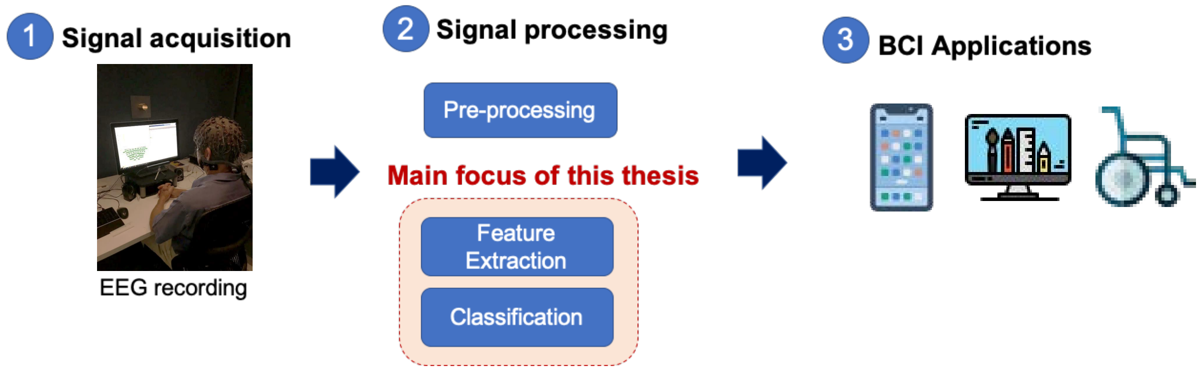

Figure 18. BCI research Steps. Icon made by Smashicons, Freepik from www.flaticon.com

3.2. About P300 Speller datasets

In the experiment carried out for the BCI Competition III dataset II (http://bbci.de/competition), BCI2000 system was used for signal acquisition of two subjects A and B when performing task on a Speller BCI. The task was to spell the given word one character at a time by focusing on each character out of 36 characters appeared in a matrix. For each character, each column and row of the matrix 6x6 was intensified (flashed) randomly at a rate of 5.7Hz. The 12 intensifications for each character (6 columns, 6 rows) were repeated 15 times. After that, the screen was blank (no intensification) for 2.5 seconds to let the subject know that character was done and prepare to focus on the next character.

The P300 Speller datasets contain training and testing datasets for each subject in Matlab ".mat" format. The training set is for the task of spelling 85 characters and the testing set is for the same task with 100 characters. The training dataset contains the correct label in the "TargetChar" variable. In contrast, the testing dataset doesn't. However, we downloaded the true labels results from the Competition website to calculate the accuracy of predictive models on the testing datasets.

The recorded data used 64-channel EEG cap following 10-20 system (see Figure 6), was bandpass filtered from 0.1-60Hz and then digitised at 240Hz. The details about P300 Speller datasets can be seen in Appendix A: P300 Speller BCI Competition Dataset.

3.3. Results analysing

The output of this study is a machine learning application architecture with classification results to predict the characters from the P300 Speller testing datasets. Since we have the true labels of both training and testing datasets, the accuracy will be calculated by comparing the number of correct characters predicted with the true labels in percentage. For example, if the predicted labels are "HEPVO" and the true labels are "HELLO", there are 3 correct characters out of total 5 characters, so the accuracy is 3/5 = 60%.

3.4. Software and tools

Python is amongst the most popular programming languages, and a dominant choice for machine learning (Shukla & Parmar, 2016; Unpingco, 2019). Scikit-learn is a free Python package providing a wide range of machine learning algorithms, data analysing and processing in neuroscience (Pedregosa et al., 2011; *Scikit-Learn*, n.d.; Unpingco, 2019).

Wyrm is an open-source Python toolbox that provides convenient methods for faster and easier BCI development (Venthur et al., 2015). This study used many supporting methods from this toolbox for loading EEG, data segmentation and pre-processing.

The wyrm Python toolbox (hereinafter referred to as Wyrm) provides its own implementation of LDA classifier. We will use this LDA classifier as exploration following the examples given in its official Github repository (https://github.com/bbci/wyrm) and then compare with some other classifiers from Scikit-learn.

4. EXPERIMENTS

4.1. Data pre-processing

The EEG data are loaded using Wyrm's loading method. After loaded, the data are converted to Wyrm's Data type which is a series of continuous EEG stream in {time, channel} format with position of the stimuli in the data stream marked. The markers are based on the "Flashing" data from the dataset. A 5th order Butterworth bandpass 30Hz low-pass filter is used to remove unwanted EEG artifacts (Zhu et al., 2010).

To process further, this continuous data stream needs to be segmented to epochs of 700ms. Here we picked 700ms because even though the ERP responses occur approximately 300ms after the onset of stimulus, this delay varies from 250ms to 750ms due to many factors (Polich, 2007). Figure 19 plots the P300 ERP in 2 channels FCz and Cz where the deflections are prominent after a delay of around 280ms-300ms after stimulus onset (time = 0).

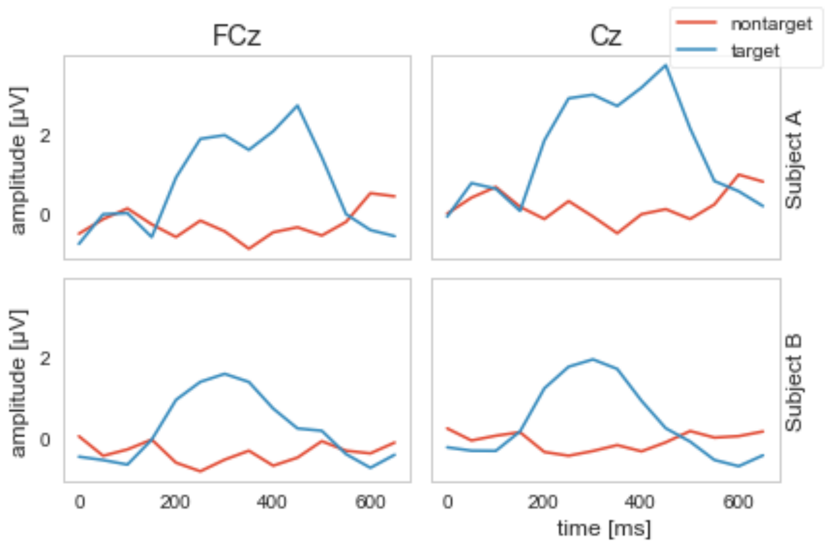

Figure 19. P300 ERP from the EEG training datasets

Next step is to reduce dimensionality using PCA. Using Scikit-learn support to create a PCA with configuration to retain 95% of the variance. Only training datasets are used for feature reduction.

The classifiers will be tested in several experiments: with PCA, without PCA, and for each with/without PCA, perform training on 64 channels, 8 channels (FCz, C3, Cz, C4, CP1, CPz, CP2, Pz) and single channel Cz.

Table 3. Dimensionality reduction with PCA for each subject.
For a/b (-x%): a is the number of features after reduction, b is before reduction, x is the percentage of features reduced.

Channels used	Subject A	Subject B
64 channels	132/896 (-85%)	87/896 (-90%)
8 channels	21/112 (-81%)	21/112 (-81%)
Cz channel	8/14 (-43%)	8/14 (-43%)

4.2. Classifications

For classification, the classifiers used for exploration are listed in Table 4. The reason we picked those classifiers is they are among the most commonly used in machine learning. The last one is Wyrm's LDA while the rest we use built-in support from Scikit-learn.

Table 4. Classifiers used

Classifiers used	Configurations
GradientBoostingClassifier	learning_rate=0.1
AdaBoostClassifier	n_estimators=100, random_state=0
KNeighborsClassifier	k = 3
DecisionTreeClassifier	Default scikit-learn configurations: *criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0*
RandomForestClassifier	n_estimators=100
LinearDiscriminantAnalysis	solver='svd'
QuadraticDiscriminantAnalysis	Default scikit-learn configurations: priors=None, reg_param=0.0, store_covariance=False, tol=0.0001
LinearSVC	max_iter=10000
SVC	C=10, gamma='auto'
Wyrm-LDA	Use shrinkage covariance estimation

Same subject training is the focus of this study; thus, the training dataset of each subject is used to train the classifiers of that subject. The predicted result of each classifier is used in combination with other 14 intensification outputs (each row or column is intensified 15 times) to choose the most probable row and column that subject paid attention to. From the predicted row and column, the predicted letter can be extracted from the letter matrix 6 rows x 6 columns. For each character, prediction result for each intensification (repetition) is calculated based on the features of all past intensifications of that character. For example, there are 15 intensifications (of randomly intensification 6 rows and 6 columns) for character "A", the prediction result for the 5th intensification is calculated using features of all intensifications from the 1st to the 5th intensification.

80% of data from training set are used for training the classifiers, then the classifiers are tested against the whole training dataset and testing dataset in several configurations: with PCA, without PCA, use 64 channels, 8 channels (FCz, C3, Cz, C4, CP1, CPz, CP2, Pz) and single channel (Cz). The charts comparing classifiers are provided below, details will be discussed in Section 5.

4.2.1. With PCA

Use all 64 channels

For training dataset of subject A, Decision Tree, SVC and Random Forest classifiers produced the highest accuracy, 100% only after 4 repetitions. First repetition (single trial), those 3 classifiers got more than 50%. On the other hand, LDA (Scikit-learn), LinearSVC and Gradient Boosting produced poor results, around 10-15%.

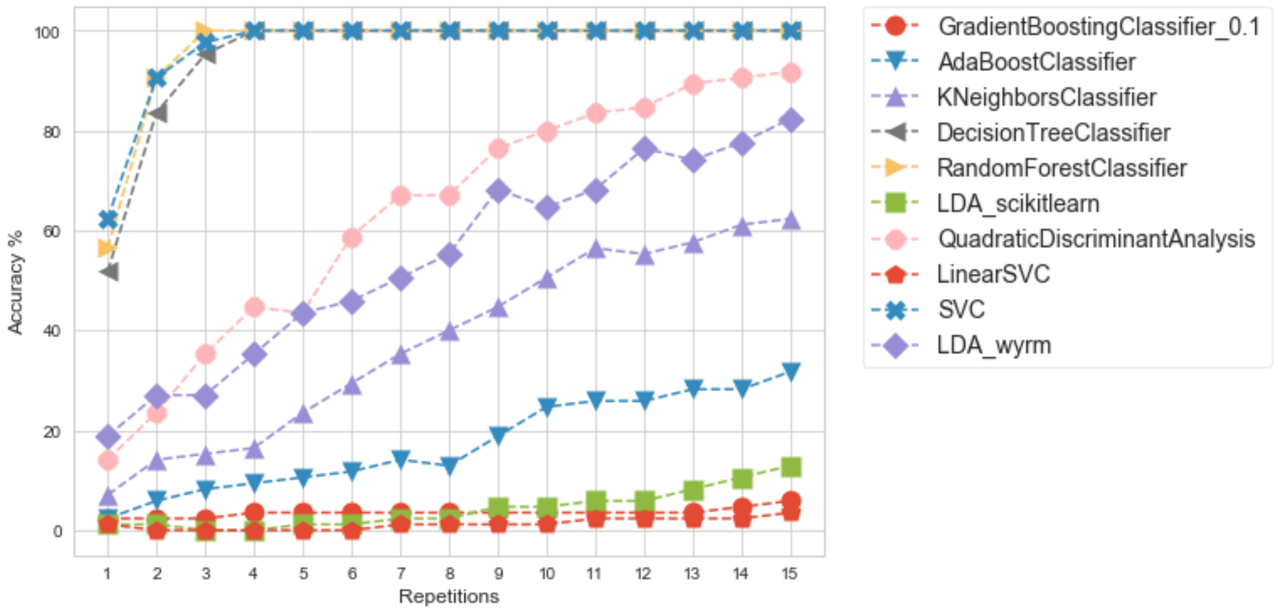

Figure 20. Classifiers Accuracy on Training dataset Subject A, 64 channels, with PCA

However, when testing against the Testing dataset, LDA (Wyrm) was the best classifier at 82% while the rest performed poorly, less than 20%, some are 5% or less (RandomForest, LinearSVC, KNeighbors).

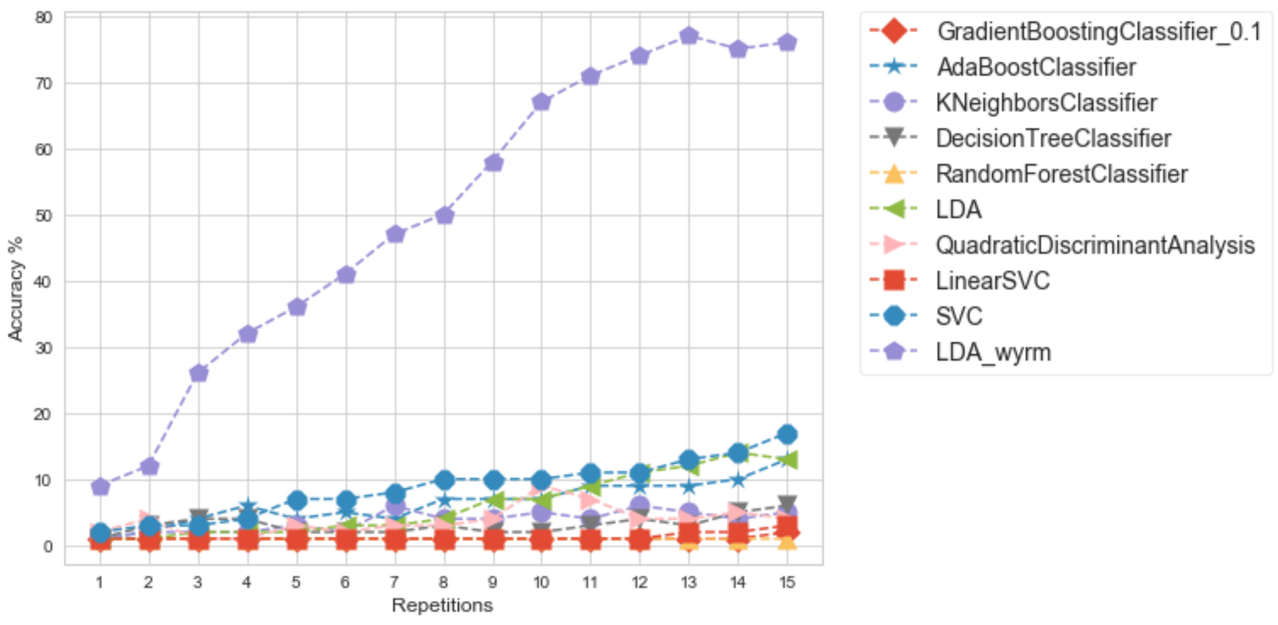

Figure 21. Classifiers Accuracy on Testing dataset Subject A, 64 channels, with PCA

Like subject A, Decision Tree, SVC and Random Forest, together with QDA are the best classifiers when training with subject B training dataset. For subject B testing dataset, LDA (Wyrm) achieved 72%, considerably higher than all other classifiers (below 20%).

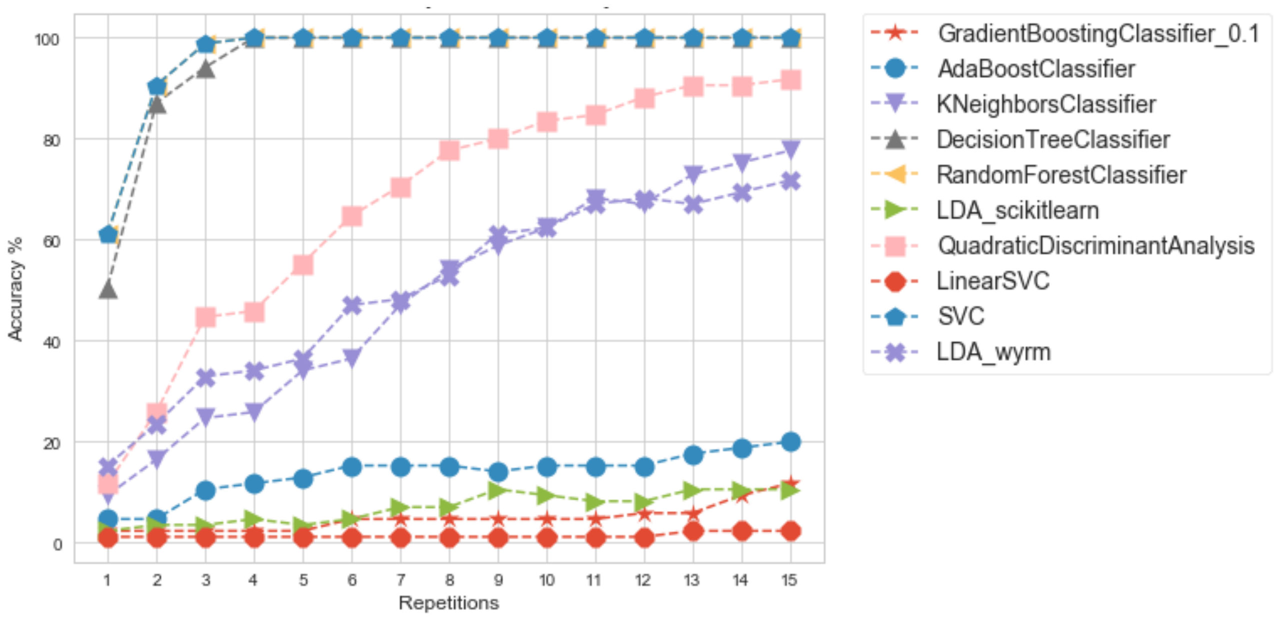

Figure 22. Classifiers Accuracy on Training dataset Subject B, 64 channels, with PCA

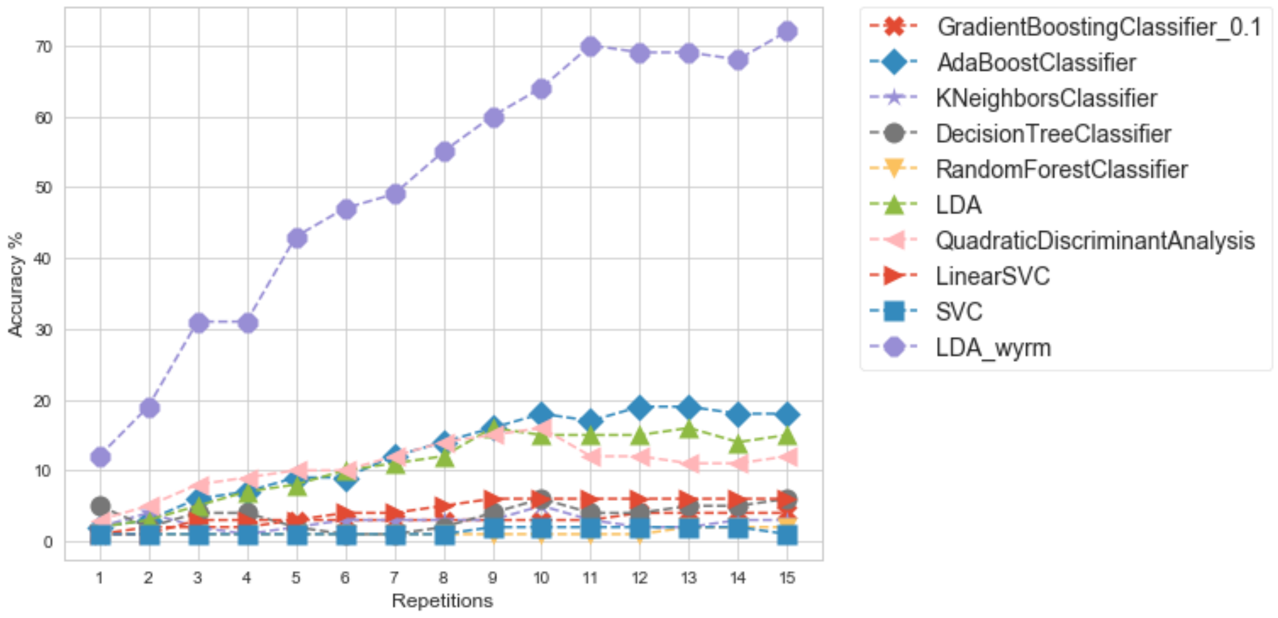

Figure 23. Classifiers Accuracy on Testing dataset Subject B, 64 channels, with PCA

Use 8 channels

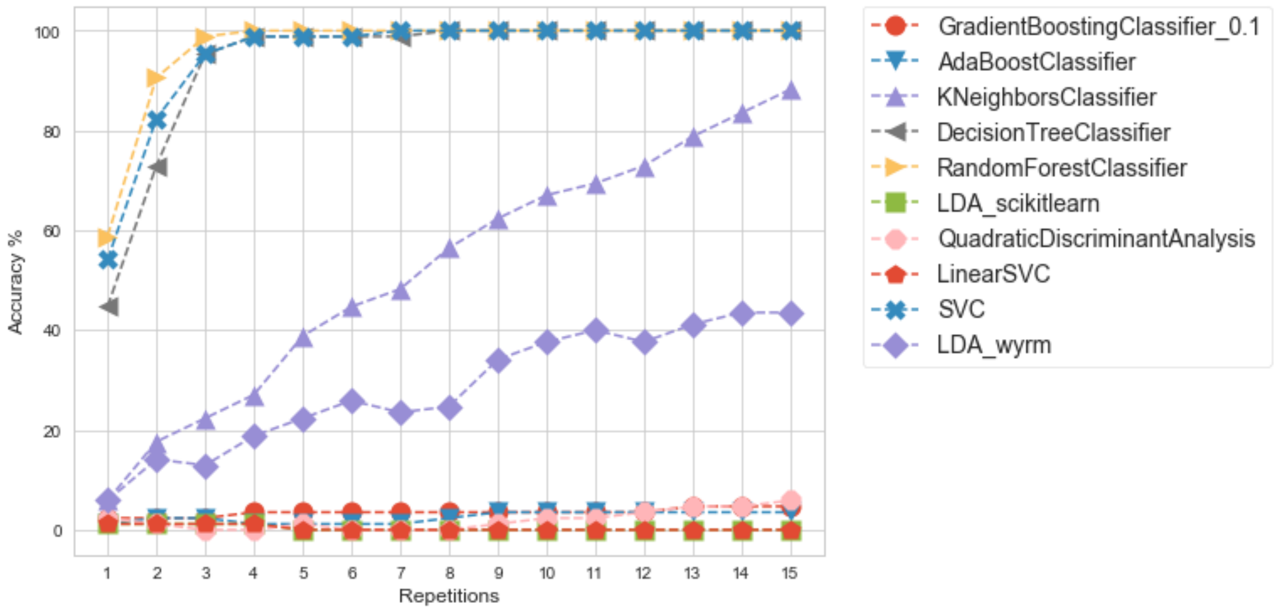

Figure 24. Classifiers Accuracy on Training dataset Subject A, 8 channels, with PCA

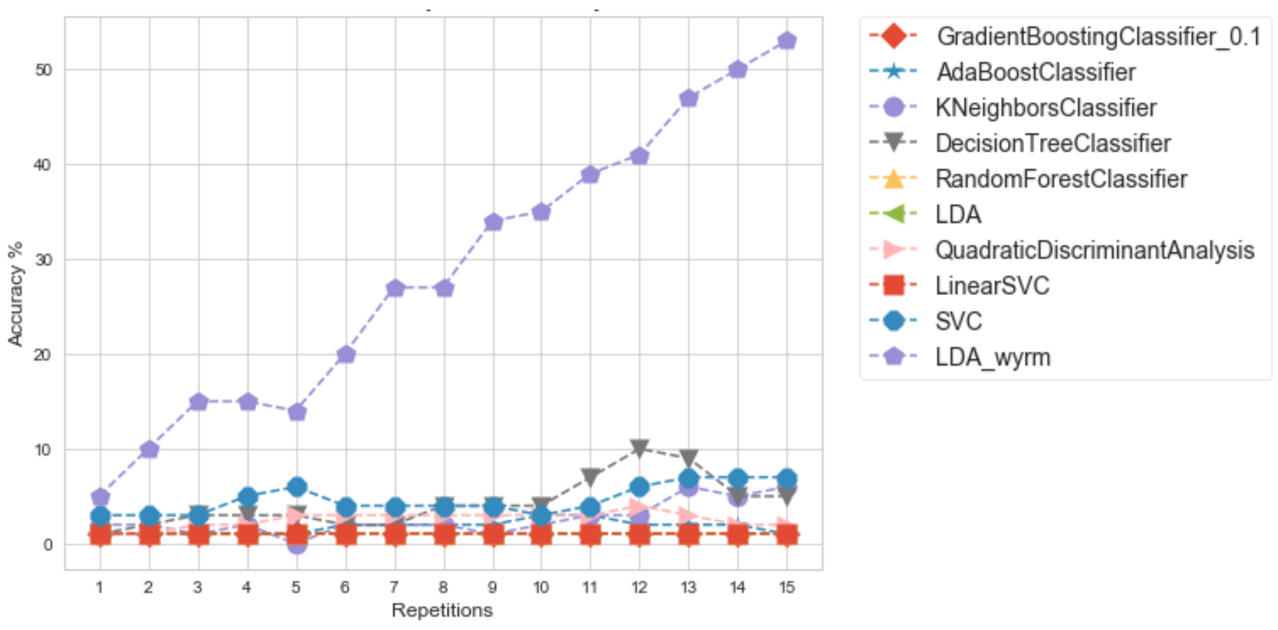

Figure 25. Classifiers Accuracy on Testing dataset Subject A, 8 channels, with PCA

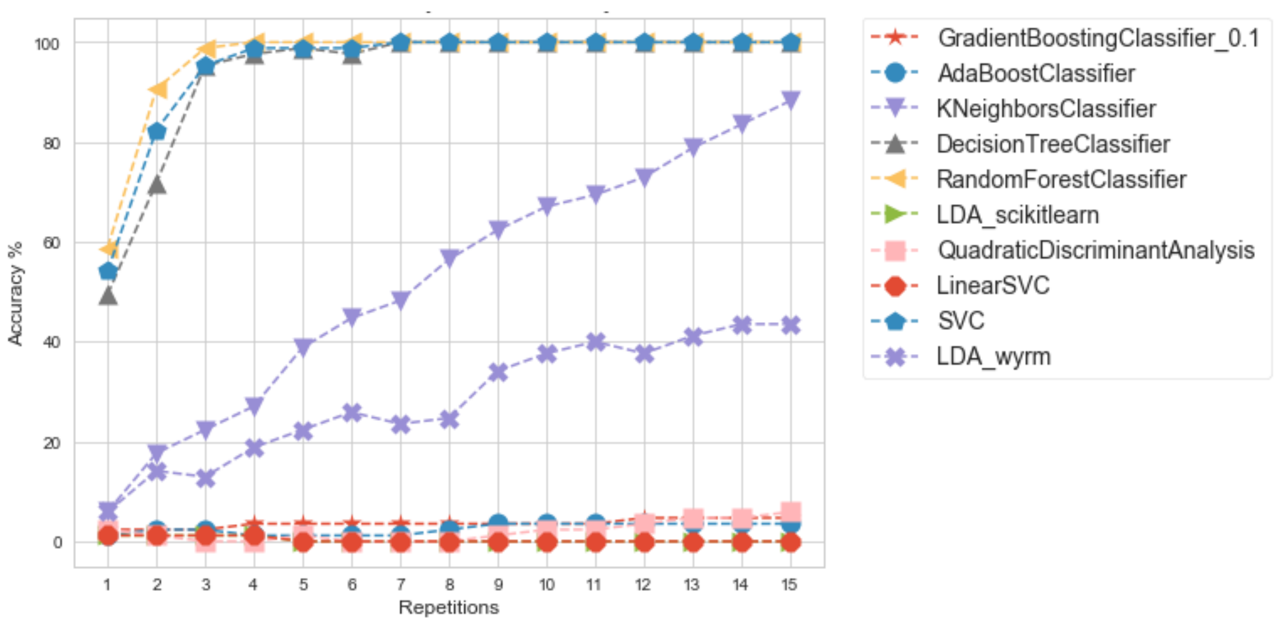

Figure 26. Classifiers Accuracy on Training dataset Subject B, 8 channels, with PCA

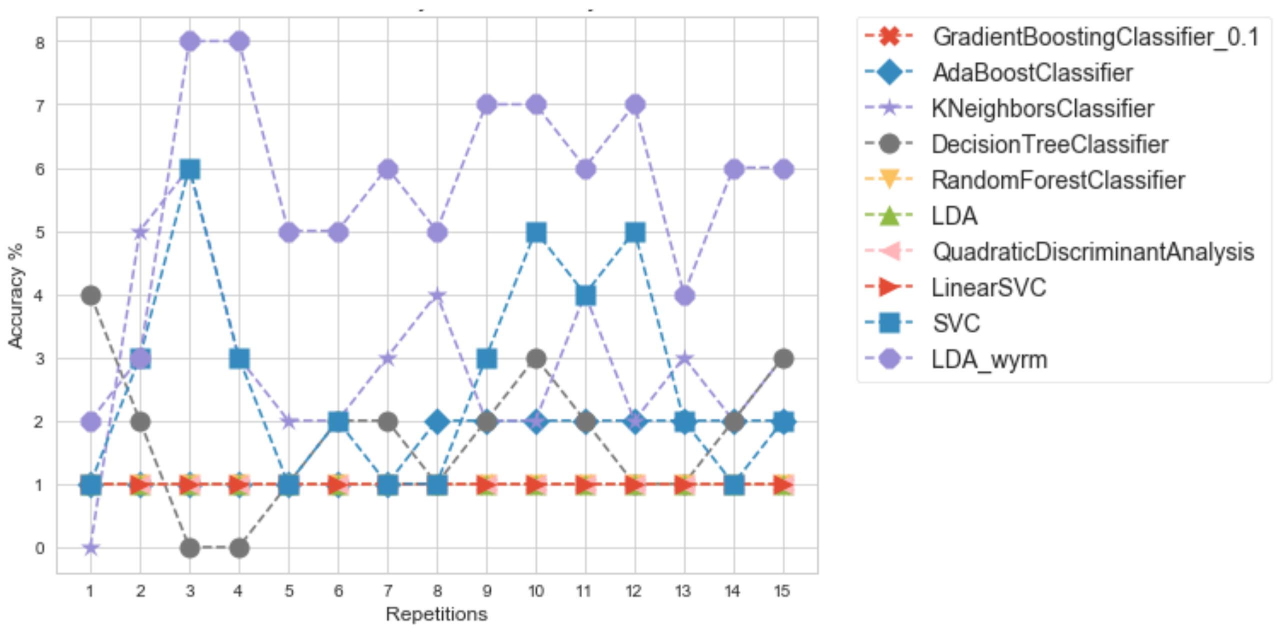

Figure 27. Classifiers Accuracy on Testing dataset Subject B, 8 channels, with PCA

Use single channel (Cz)

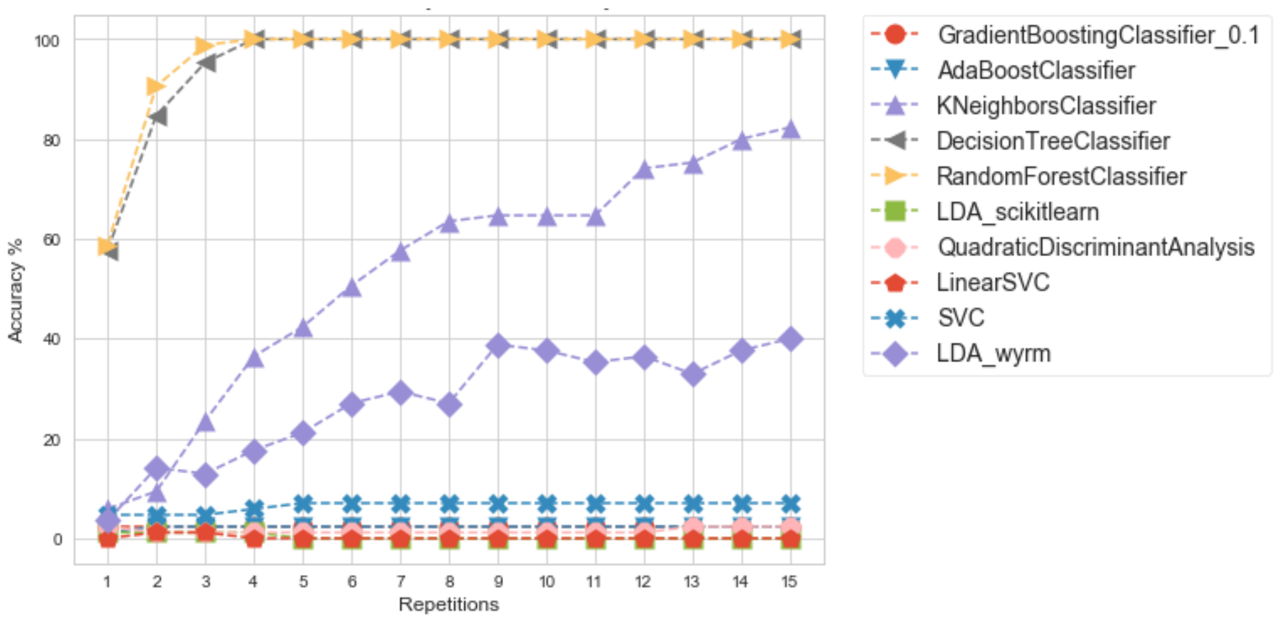

Figure 28. Classifiers Accuracy on Training dataset Subject A, single channel Cz, with PCA

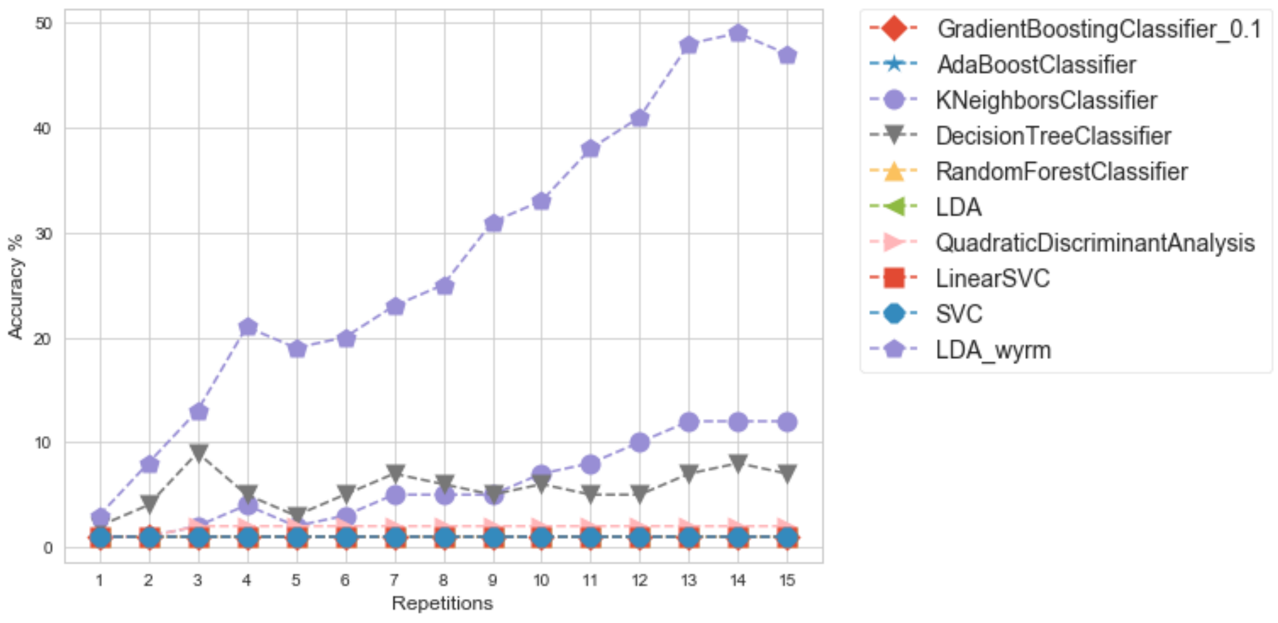

Figure 29. Classifiers Accuracy on Testing dataset Subject A, single channel Cz, with PCA

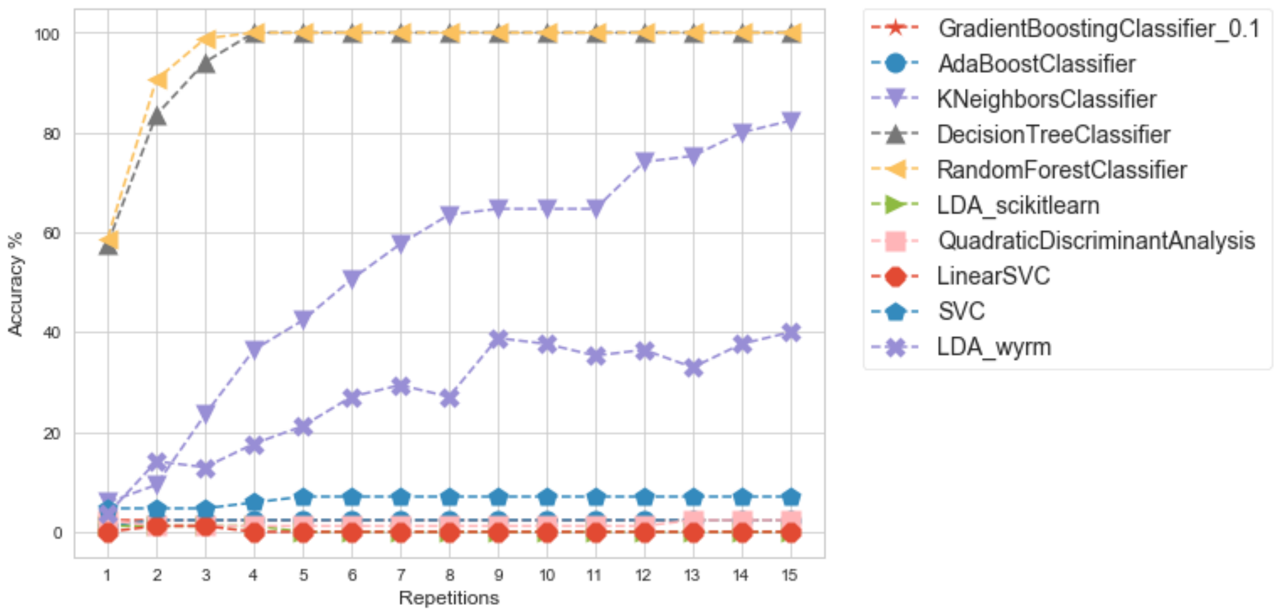

Figure 30. Classifiers Accuracy on Training dataset Subject B, single channel Cz, with PCA

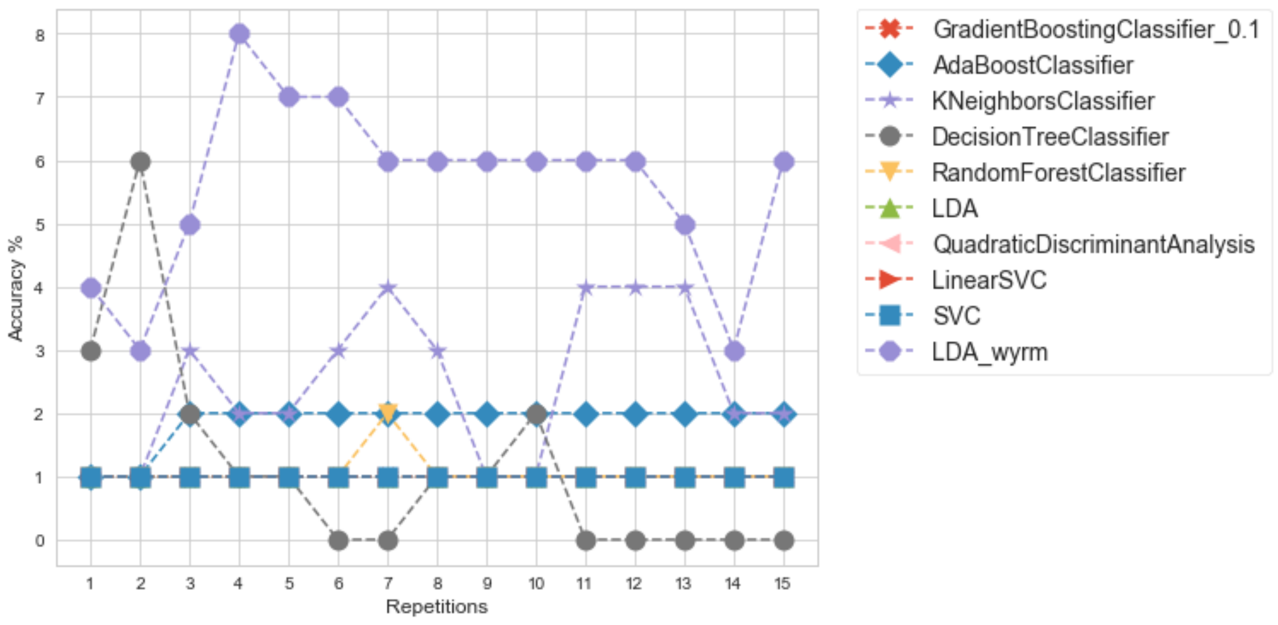

Figure 31. Classifiers Accuracy on Testing dataset Subject B, single channel Cz, with PCA

4.2.2. Without PCA

Use all 64 channels

When using all 64 channels and without PCA, the number of features is large, the space and time needed to train and test models were significantly longer than when having lower number of features by PCA.

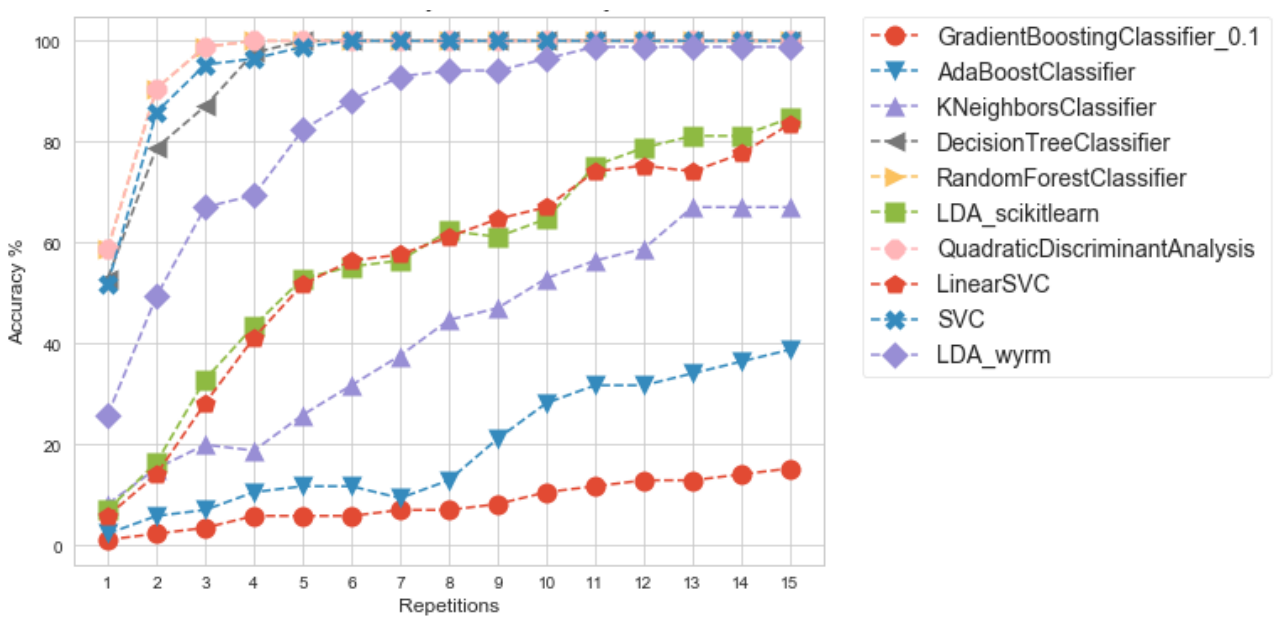

Figure 32. Classifiers Accuracy on Training dataset Subject A, 64 channels, without PCA

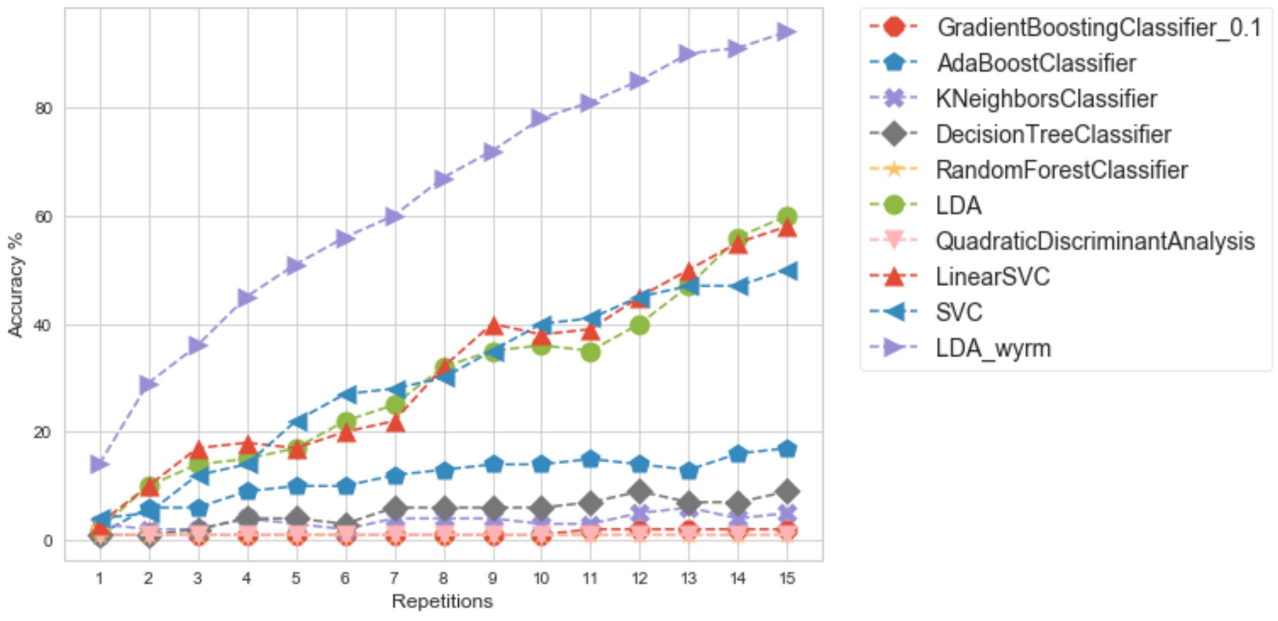

Figure 33. Classifiers Accuracy on Testing dataset Subject A, 64 channels, without PCA

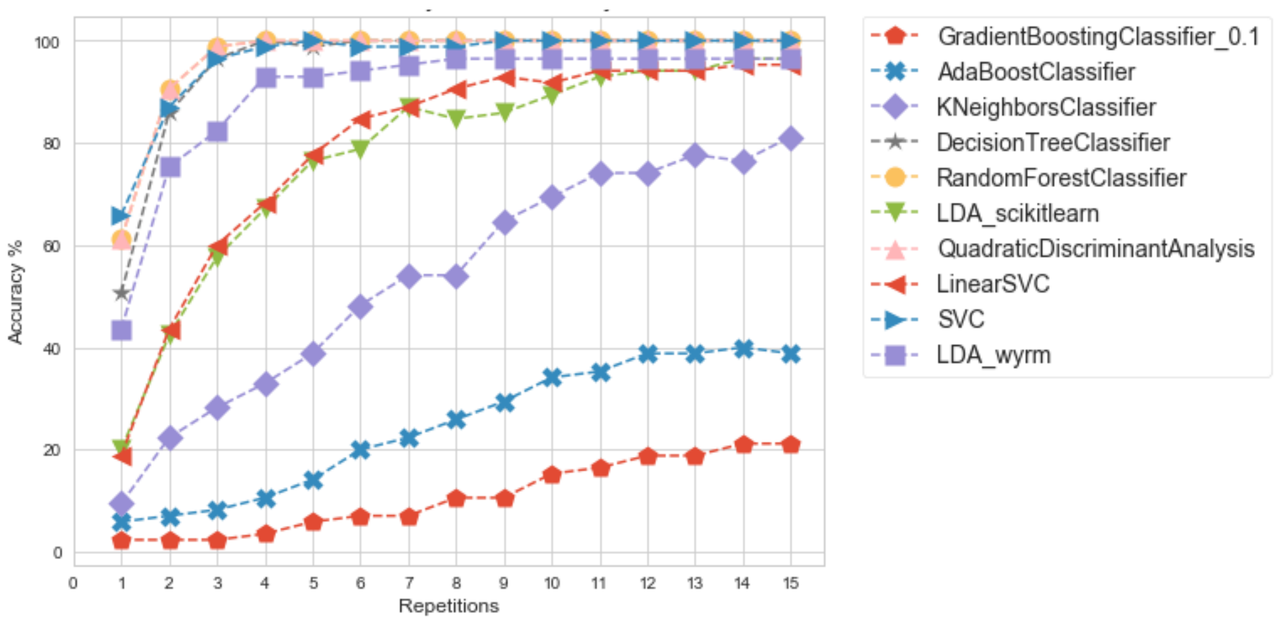

Figure 34. Classifiers Accuracy on Training dataset Subject B, 64 channels, without PCA

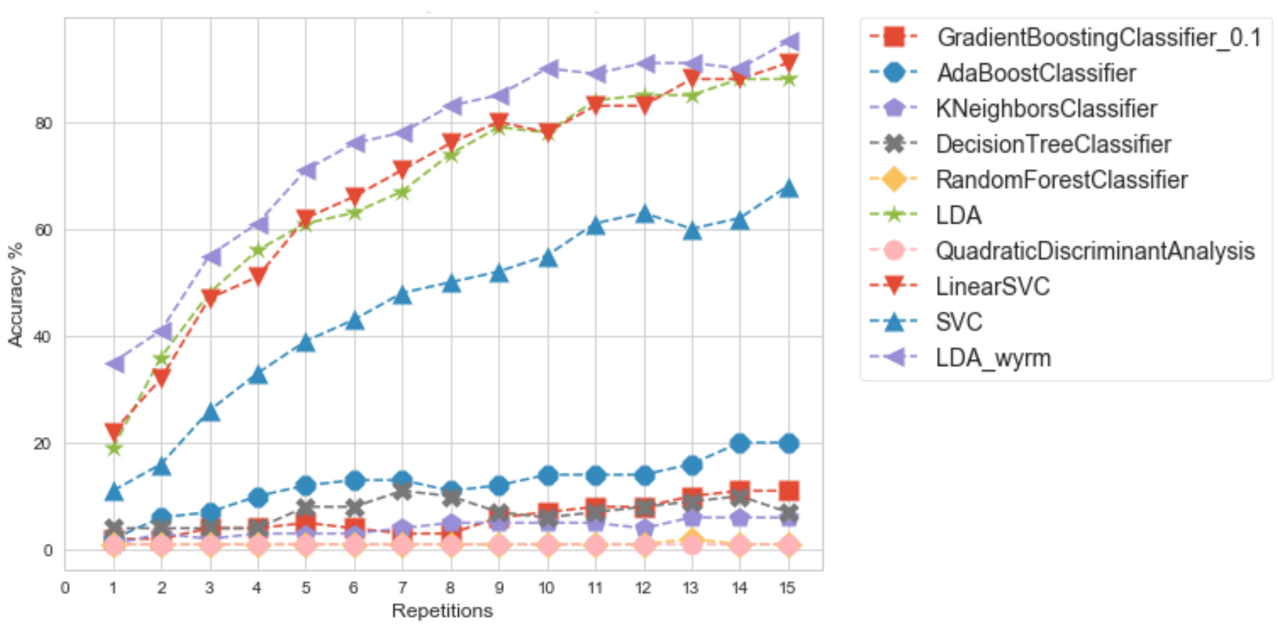

Figure 35. Classifiers Accuracy on Testing dataset Subject B, 64 channels, without PCA

Use 8 channels

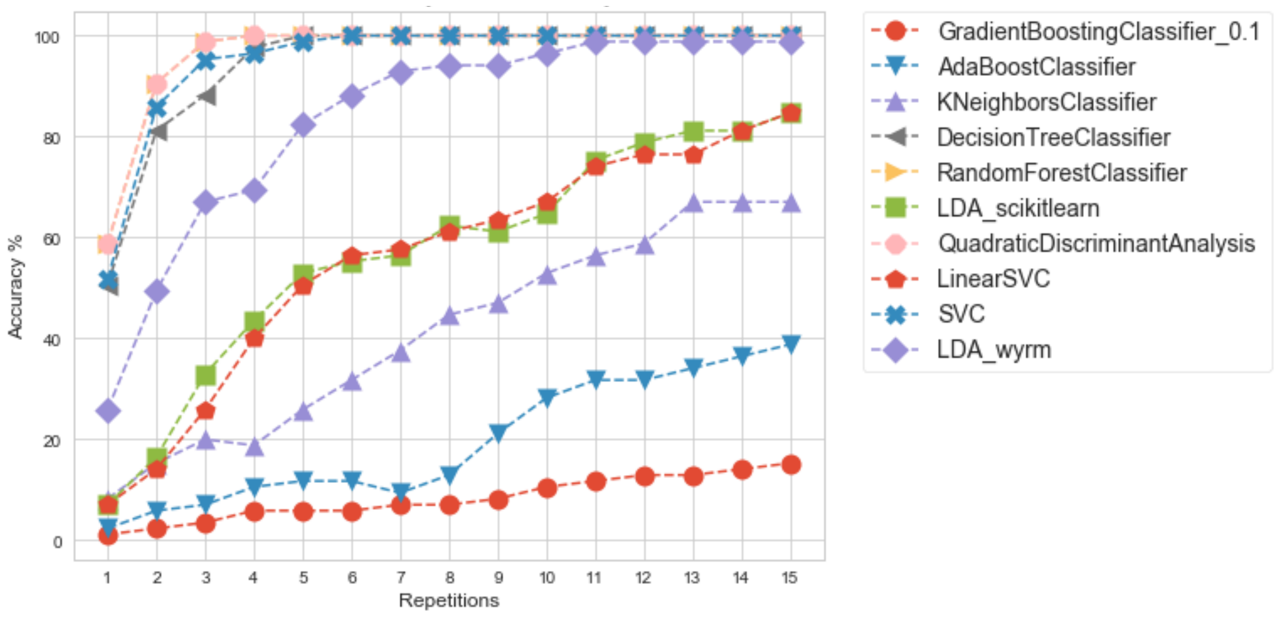

Figure 36. Classifiers Accuracy on Training dataset Subject A, 8 channels, without PCA

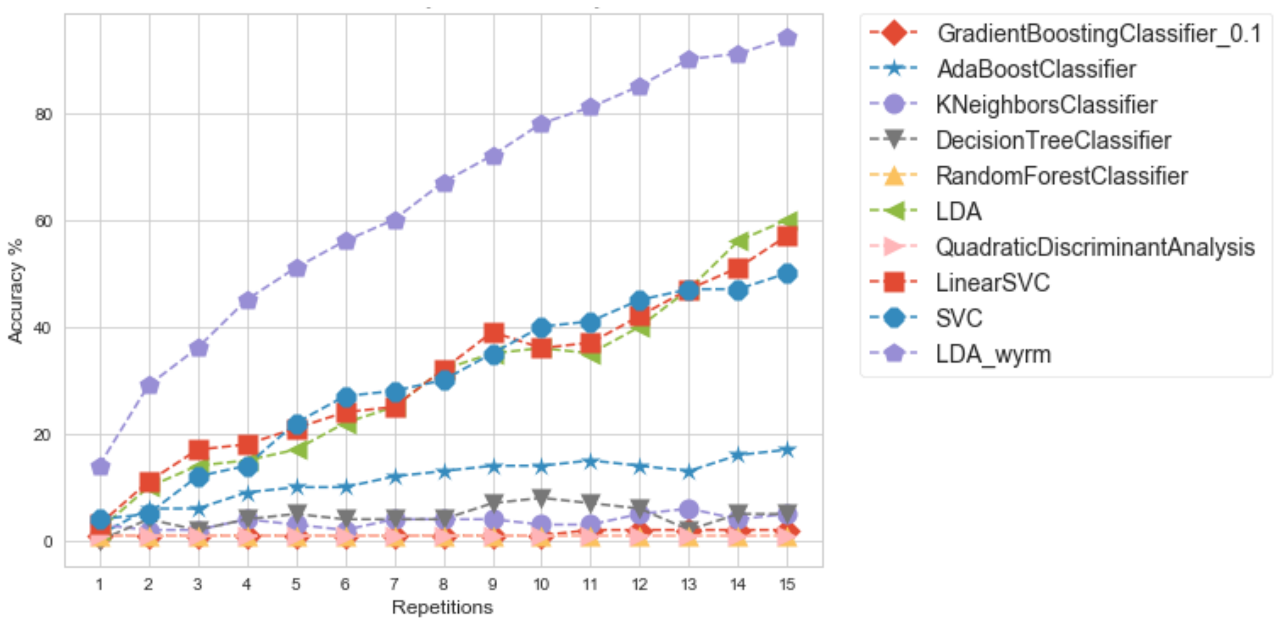

Figure 37. Classifiers Accuracy on Testing dataset Subject A, 8 channels, without PCA

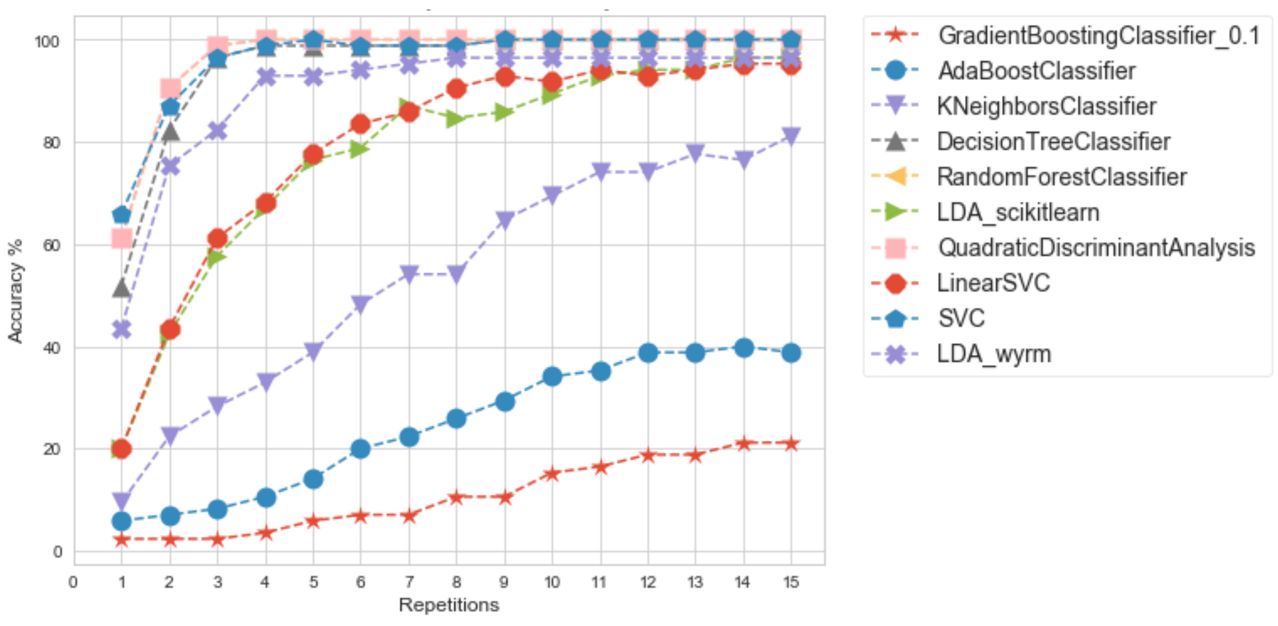

Figure 38. Classifiers Accuracy on Training dataset Subject B, 8 channels, without PCA

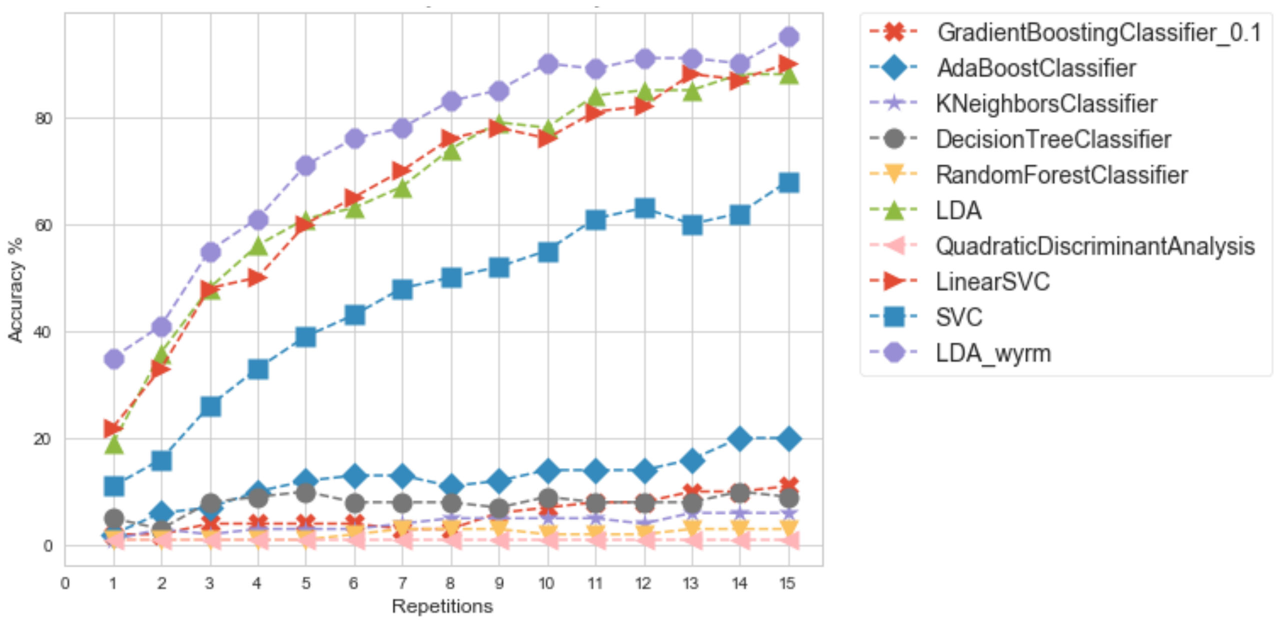

Figure 39. Classifiers Accuracy on Testing dataset Subject B, 8 channels, without PCA

Use single channel Cz

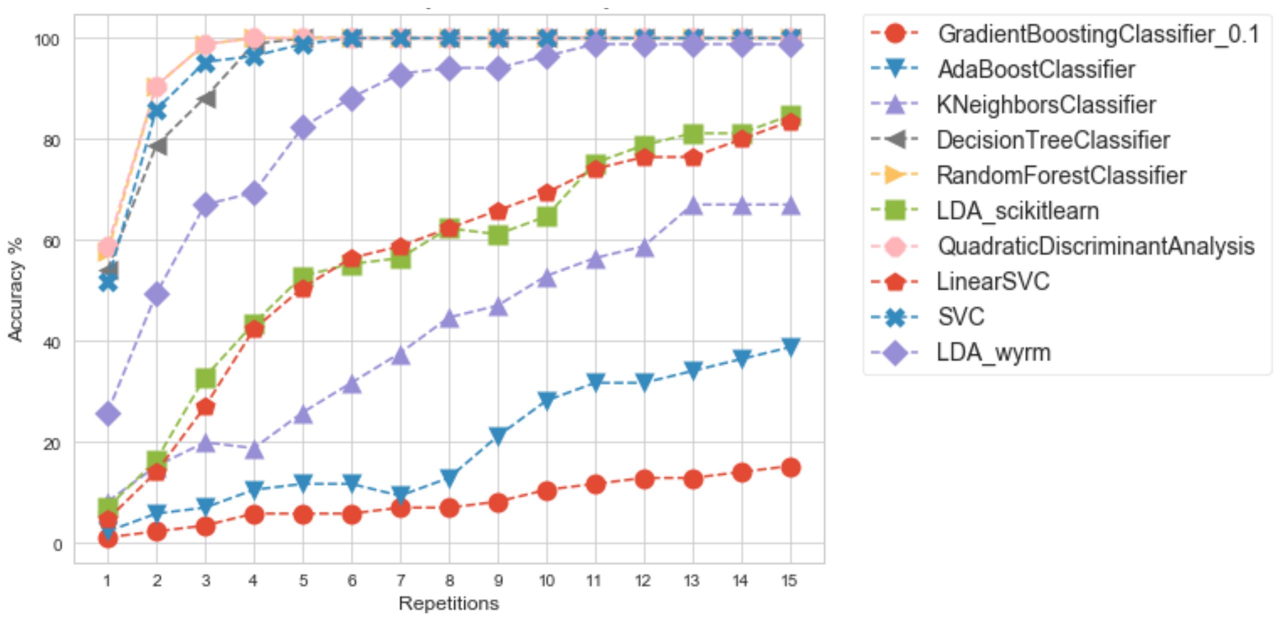

Figure 40. Classifiers Accuracy on Training dataset Subject A, channel Cz, without PCA

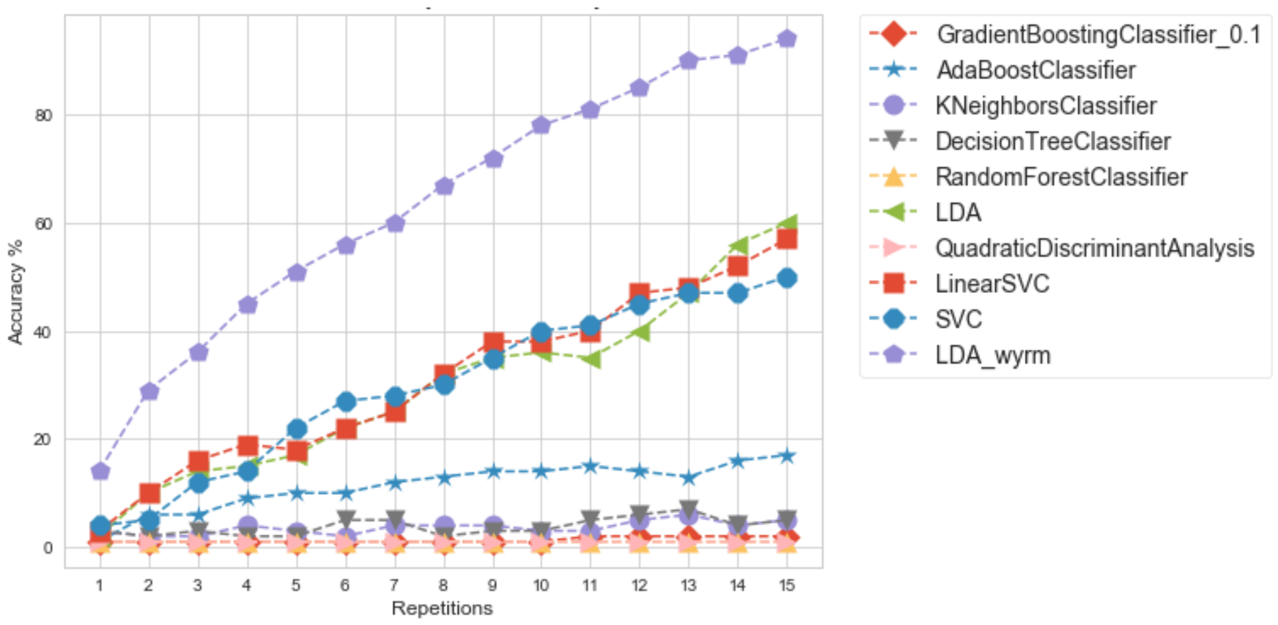

Figure 41. Classifiers Accuracy on Testing dataset Subject A, channel Cz, without PCA

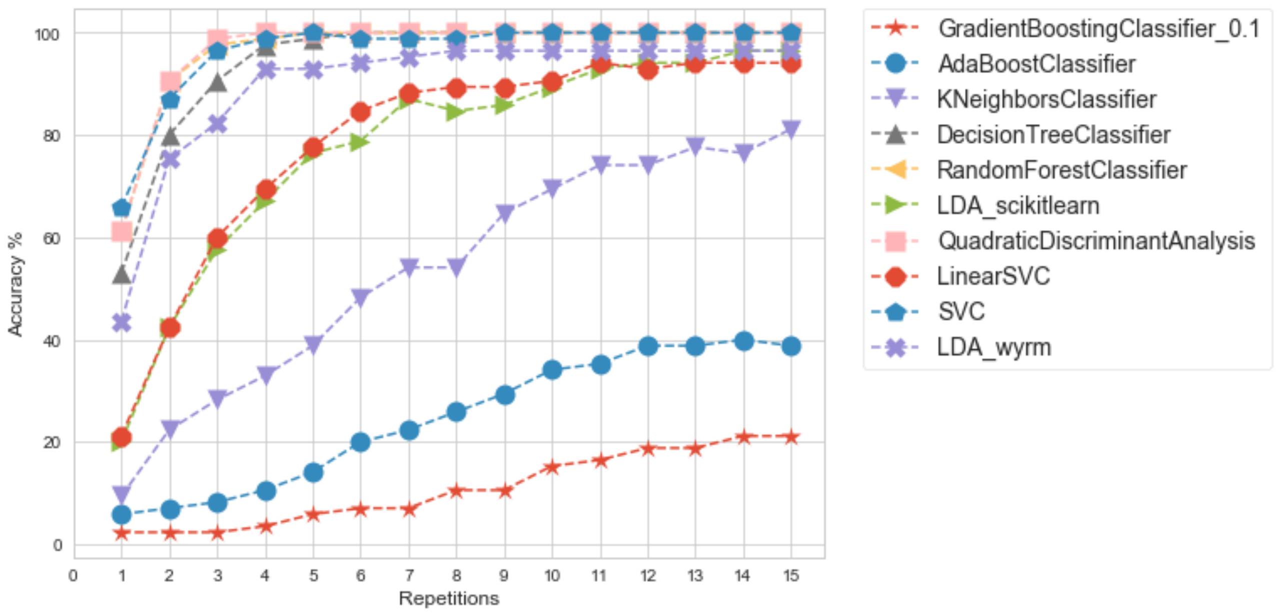

Figure 42. Classifiers Accuracy on Training dataset Subject B, channel Cz, without PCA

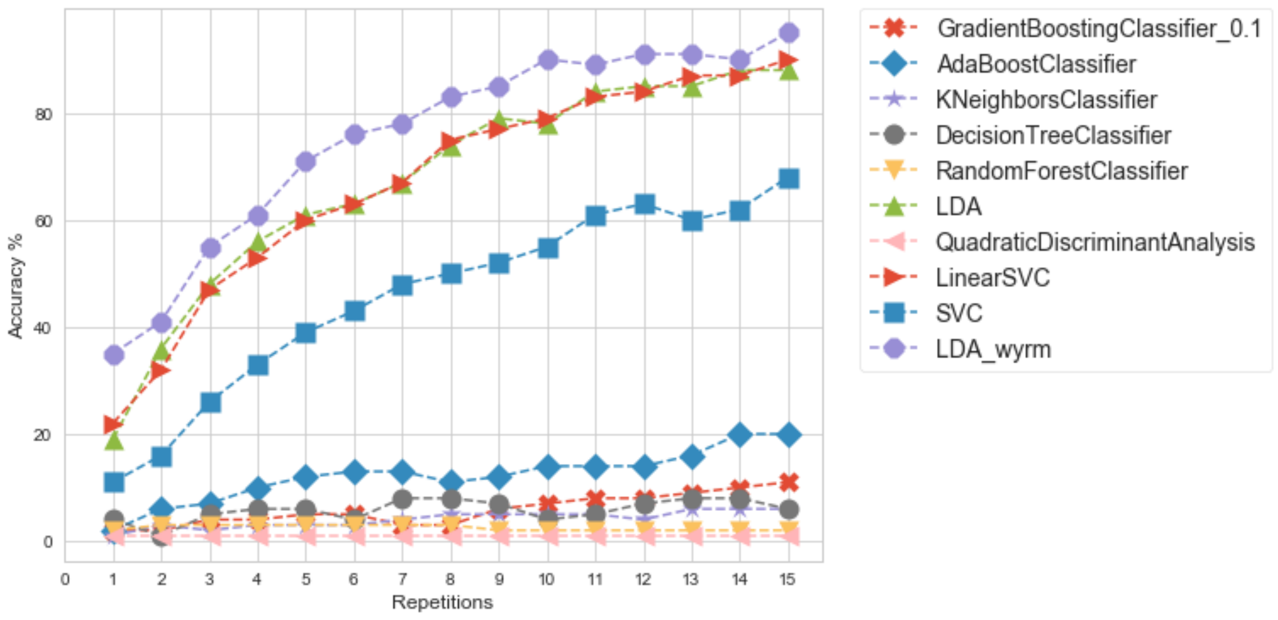

Figure 43. Classifiers Accuracy on Testing dataset Subject B, channel Cz, without PCA

5. DISCUSSION

It is evident that LDA (Wyrm) classifier produced the best results in all cases. The highest accuracy achieved is 94.5% on average of both subjects when not doing PCA after 15 repetitions. When doing PCA, the highest accuracy is 76% and 72% for subject A and B respectively after 15 repetitions. Applying PCA to reduce dimensionality causes the prediction performance to be slightly lower than without applying PCA, however still in the acceptable range. Applying PCA helped reduce the feature dimensionality (see Table 3) which improved the time and memory space needed to classify data (Abdi & Williams, 2010). This study performed offline BCI therefore the need for saving time and memory may not be critical. However, for online BCI systems where shorter calibration time is needed (Zhu et al., 2011), features dimensionality reduction has proven to be useful (Lenhardt et al., 2008; D. Wu, 2017).

The BBCI competition on the same P300 Speller datasets with accuracy after 15 repetitions are 96.5%, 90.5% and 90% respectively[1]. Comparing to those results, LDA (Wyrm), at 94.5%, performed very close to the 1st winner. Interestingly, while our Linear SVM reached an average of 73% without PCA and only 4.5% with CPA, the 1st winner's model used Linear SVM could achieve 96.5%[2]. Alain Rakotomamonjy, author of the 1st winner model, performed a channel selection to pick the best channels susceptible to P300 ERP out of 64 channels. The result highlights the importance of channel selection procedure to eliminate spurious channels, reduce number of features while increase recognition performance, which will be clearer when we consider the accuracy results of 8 channels and 1 channel.

When performing the tests with only 8 channels or 1 channel (Cz), LDA (Wyrm) classifier also reached significant scores compared to other classifiers. For single channel (Cz), it scored 76% and 75% for subject A and B respectively. For 8 channels, it scored 60% for subject A but only 3% for subject B, indicating that some of 8 channels did not perform well for subject B and should be

[1] http://www.bbci.de/competition/iii/results/index.html#albany

[2] http://www.bbci.de/competition/iii/results/albany/AlainRakotomamonjy_desc.txt

removed from classification. Channel selection approaches mentioned by (Colwell et al., 2014; Khairullah et al., 2020) can be used in future research to help improve this situation.

However, some classifiers like Random Forest, AdaBoost or Gradient Boosting did not perform well. While they may perform relatively well against training dataset, they got notably low accuracy scores when predicting the testing dataset. When performing on 8 channels and 1 channel (Cz), the performance dropped significantly for both with PCA and without PCA. This could be due to the unbalanced weighted features problem among channels, suggesting that the 8 selected channels may not be the set of channels that gives the best performance. (Colwell et al., 2014; Khairullah et al., 2020) have conducted studies into this channel selection problem and their approaches can be used in this study to select the most performant channels. However, the focus of this study is to explore machine learning aspect of BCI system and try out different classification methods, thus future research is needed to investigate the issue and improve the accuracy for single trial cases and fewer than (or maximum) 8 channels.

The gap between prediction results against Training and Testing datasets can be seen across almost all classifiers. Many of classifiers performed well against training set but did very poorly against the testing dataset. One possibility is that classifiers are overfitted, meaning the models have too many parameters that correspond too closely to the training dataset, therefore they cannot predict future data reliably (Dietterich, 1995; Yeom et al., 2018). Analysing confusion matrix and loss matrix can also help to analyse the misclassifications (Kuncheva, 2014).

6. CONCLUSION

This study explored BCI systems and its related aspects, including brain waves, EEG, P300 oddball and more. Machine learning architecture in a BCI system was explored, and common machine learning algorithms were experimented on P300 Speller datasets. The results show that the LDA classifier from Wyrm Python toolbox achieved the highest performance, at 94.5% after 15 repetitions.

Accuracy scores when applied PCA were lower than when not applied PCA. Without PCA, LDA (Wyrm) scored 94% and 95% for subject A and B respectively, but when applying PCA, the scores were 76% and 72%. This is consistent with the results from study conducted by (Selim et al., 2009) where Selim et al. tested Bayesian linear discriminant analysis (BLDA) with and without PCA.

However, due to the inaccuracy when experimented with fewer trials and fewer channels, there is a possibility that the data were contaminated by muscular movement. Future work is needed to explore that aspect to remove artifacts and apply a suitable channel selection approach to improve the accuracy of the models.

7. REFERENCES

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(4), 433–459. https://doi.org/10.1002/wics.101

Abhang, P. A. (2016). *Introduction to EEG- and speech-based emotion recognition* (B. Gawali & S. Mehrotra (Eds.)). Amsterdam, Netherlands : Academic Press.

Abiri, R., Borhani, S., Sellers, E. W., Jiang, Y., & Zhao, X. (2019). A comprehensive review of EEG-based brain–computer interface paradigms. *Journal of Neural Engineering*, *16*(1), 011001. https://doi.org/10.1088/1741-2552/aaf12e

Acharya, J. N., Hani, A. J., Cheek, J., Thirumala, P., & Tsuchida, T. N. (2016). American Clinical Neurophysiology Society Guideline 2: Guidelines for Standard Electrode Position Nomenclature. *Neurodiagnostic Journal*, *56*(4), 245–252. https://doi.org/10.1080/21646821.2016.1245558

Alpaydin, E. (2014). *Introduction to machine learning* (Third edit). Cambridge, Massachusetts : MIT Press.

Altman, N. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175.

Anaissi, A., Khoa, N. L. D., Rakotoarivelo, T., Alamdari, M. M., & Wang, Y. (2017). Self-advised incremental one-class support vector machines: An application in structural health monitoring. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 10634, pp. 484–496). https://doi.org/10.1007/978-3-319-70087-8_51

Aurlien, H., Gjerde, I. O., Aarseth, J. H., Eldøen, G., Karlsen, B., Skeidsvoll, H., & Gilhus, N. E. (2004). EEG background activity described by a large computerized database. *Clinical Neurophysiology*, *115*(3), 665–673.

Bauer, G., Gerstenbrand, F., & Rumpl, E. (1979). Varieties of the locked-in syndrome. *Journal of Neurology*, *221*(2), 77–91. https://doi.org/10.1007/BF00313105

BCI Competition III: Results. (n.d.). Retrieved August 7, 2020, from http://www.bbci.de/competition/iii/results/index.html#albany

Bell, J. (2020). *Machine learning: hands-on for developers and technical professionals*. John Wiley & Sons.

Benito Núñez, I. M. (2011). *EEG artifact detection*.

Bera, T. K. (2015). Noninvasive electromagnetic methods for brain monitoring: a technical review. In *Brain-Computer Interfaces* (pp. 51–95). Springer.

Berkson, J. (1944). Application of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association*, *39*(227), 357–365. https://doi.org/10.1080/01621459.1944.10500699

Bhat, D. S. (2018). *Assessing Performance of Detectors of High Frequency Oscillations in EEG Signals*. University of Texas at El Paso.

Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., & Flor, H. (1999). A spelling device for the paralysed. *Nature*, *398*(6725), 297–298.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Black, C., Voigts, J., Agrawal, U., Ladow, M., Santoyo, J., Moore, C., & Jones, S. (2017). Open Ephys electroencephalography (Open Ephys+ EEG): a modular, low-cost, open-source solution to human neural recording. *Journal of Neural Engineering*, *14*(3), 35002.

Borghini, G., Astolfi, L., Vecchiato, G., Mattia, D., & Babiloni, F. (2014). Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness. *Neuroscience & Biobehavioral Reviews*, *44*, 58–75.

Boser, B., Guyon, I., & Vapnik, V. (1992). *A training algorithm for optimal margin classifiers* (pp. 144–152). https://doi.org/10.1145/130385.130401

Breiman, L. (1984). Classification and regression trees. In J. H. (Jerome H. . Friedman, R. A. Olshen, & C. J. Stone (Eds.), *Regression trees*. Belmont, Calif. : Wadsworth International Group.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *59*(1), 3–54.

Brouwer, A.-M., & Van Erp, J. B. F. (2010). A tactile P300 brain-computer interface. *Frontiers in Neuroscience*, *4*, 19.

Brown, G. D., Yamada, S., & Sejnowski, T. J. (2001). Independent component analysis at the neural cocktail party. *Trends in Neurosciences*, *24*(1), 54–63.

Brownlee, J. (2019, August 12). *A Tour of Machine Learning Algorithms*. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Brunner, P., Joshi, S., Briskin, S., Wolpaw, J. R., Bischof, H., & Schalk, G. (2010). Does the p300 speller depend on eye gaze? *Journal of Neural Engineering*, *7*(5), 56013. https://doi.org/10.1088/1741-2560/7/5/056013

Burgess, A., & Gruzelier, J. (1993). Individual reliability of amplitude distribution in topographical mapping of EEG. *Electroencephalography and Clinical Neurophysiology*, *86*(4), 219–223.

Caglayan, O., & Arslan, R. B. (2012). *P300 based auditory visual brain computer interface* (pp. 1–4). https://doi.org/10.1109/SIU.2012.6204826

Chang, M., Nishikawa, N., Struzik, Z., Mori, K., Makino, S., Mandic, D., & Rutkowski, T. (2013). Comparison of P300 Responses in Auditory, Visual and Audiovisual Spatial Speller BCI Paradigms. *ArXiv.Org*. https://doi.org/10.3217/978-3-85125-260-6-156

Chatterjee, A., Aggarwal, V., Ramos, A., Acharya, S., & Thakor, N. V. (2007). A brain-computer interface with vibrotactile biofeedback for haptic information. *Journal of Neuroengineering and Rehabilitation*, *4*(1), 40.

Chen, X., Chen, Z., Gao, S., & Gao, X. (2014). A high-ITR SSVEP-based BCI speller. *Brain-Computer Interfaces*, *1*(3–4), 181–191. https://doi.org/10.1080/2326263X.2014.944469

Chen, X., Wang, Y., Nakanishi, M., Gao, X., Jung, T.-P., & Gao, S. (2015). High-speed spelling with a noninvasive brain-computer interface.(PNAS PLUS: NEUROSCIENCE). *Proceedings of the National Academy of Sciences of the United States*, *112*(44), E6058. https://doi.org/10.1073/pnas.1508080112

Chi, Y. M., & Cauwenberghs, G. (2010). Wireless non-contact EEG/ECG electrodes for body sensor networks. *2010 International Conference on Body Sensor Networks*, 297–301.

Coenen, A., Fine, E., & Zayachkivska, O. (2014). Adolf Beck: a forgotten pioneer in electroencephalography. *Journal of the History of the Neurosciences*, *23*(3), 276–286.

Colwell, K. A., Ryan, D. B., Throckmorton, C. S., Sellers, E. W., & Collins, L. M. (2014). Channel selection methods for the P300 Speller. *Journal of Neuroscience Methods*, *232*, 6–15. https://doi.org/10.1016/j.jneumeth.2014.04.009

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1023/A:1022627411411

Cramer, J. (2002). *The Origins of Logistic Regression* (Vols. 02-119/4). Tinbergen Institute.

da Silva, F. L. (1991). Neural mechanisms underlying brain waves: from neural membranes to networks. *Electroencephalography and Clinical Neurophysiology*, *79*(2), 81–93.

Daly, I., Williams, N., Nasuto, S. J., Warwick, K., & Saddy, D. (2010). Single trial BCI operation via Wackermann parameters. *2010 IEEE International Workshop on Machine Learning for Signal Processing*, 409–414.

Delorme, A., Makeig, S., & Sejnowski, T. (2001). Automatic artifact rejection for EEG data using high-order statistics and independent component analysis. *Proceedings of the Third International ICA Conference*, 9–12.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys (CSUR)*, *27*(3), 326–327.

Drucker, H., Surges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In *Advances in Neural Information Processing Systems* (pp. 155–161).

Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(4), 325–327. https://doi.org/10.1109/TSMC.1976.5408784

Edlinger, G., Rizzo, C., & Guger, C. (2012). *Brain Computer Interface* (K.-P. Hoffmann & R. S. Pozos (Eds.)). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74658-4_52

Farwell, L. A., & Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, *70*(6), 510–523. https://doi.org/10.1016/0013-4694(88)90149-6

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Friedman, J. H., Kohavi, R., & Yun, Y. (1996). Lazy decision trees. *AAAI/IAAI, Vol. 1*, 717–724.

Friston, K. J. (2009). Modalities, modes, and models in functional neuroimaging. *Science*, *326*(5951), 399–403.

Fu, Y.-W., Chen, H.-L., Chen, S.-J., Shen, L. M., & Li, Q. Q. (2014). A new evolutionary support vector machine with application to parkinson's disease diagnosis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8795, pp. 42–49).

Fukuzumi, S., Yamaguchi, H., Tanaka, K., Yamazaki, T., Yamanoi, T., & Kamijo, K. (2014). *A New Computational Method for Single-Trial-EEG-Based BCI BT - Human Interface and the Management of Information. Information and Knowledge Design and Evaluation* (S. Yamamoto (Ed.); pp. 148–156). Springer International Publishing.

Galun, R. (2012). *Sensory physiology and behavior* (Vol. 15). Springer Science & Business Media.

Gao, S., Wang, Y., Gao, X., & Hong, B. (2014). Visual and Auditory Brain–Computer Interfaces. *IEEE Transactions on Biomedical Engineering*, *61*(5), 1436–1447. https://doi.org/10.1109/TBME.2014.2300164

Godfrey, K. (1985). Simple linear regression in medical research. *New England Journal of Medicine*, *313*(26), 1629–1636.

Grummett, T. S., Leibbrandt, R. E., Lewis, T. W., DeLosAngeles, D., Powers, D. M. W., Willoughby, J. O., Pope, K. J., & Fitzgibbon, S. P. (2015). Measurement of neural signals from inexpensive, wireless and dry EEG systems. *Physiological Measurement*, *36*(7), 1469–1484. https://doi.org/10.1088/0967-3334/36/7/1469

Haider, A., & Fazel-Rezai, R. (2017). Application of P300 event-related potential in brain-computer interface. *Event-Related Potentials and Evoked Potentials. InTech*, 19–38.

Hao, P., Ying, D., & Longyuan, T. (2009). *Application for Web Text Categorization Based on Support Vector Machine* (Vol. 2, pp. 42–45). https://doi.org/10.1109/IFCSTA.2009.132

Harnad, S. (2006). The annotation game: On Turing (1950) on computing, machinery, and intelligence. In *The Turing test sourcebook: philosophical and methodological issues in the quest for the thinking computer*. Kluwer.

Hassanien, A. E. (2014). *Brain-Computer Interfaces: Current Trends and Applications* (J. Kacprzyk, L. C. Jain, A. E. Hassanien, & A. T. Azar (Eds.); 2015th ed., Vol. 74). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-10978-7

Hecht-Nielsen, R. (1989). Neural network primer: part i. *AI Expert*, 4–51.

Hill, N J, Lal, T. N., Bierig, K., Birbaumer, N., & Schölkopf, B. (2005). An auditory paradigm for brain-computer interfaces. In *Advances in Neural Information Processing Systems*.

Hill, N Jeremy, Lal, T. N., Schroder, M., Hinterberger, T., Wilhelm, B., Nijboer, F., Mochty, U., Widman, G., Elger, C., & Scholkopf, B. (2006). Classifying EEG and ECoG signals without subject training for fast BCI implementation: comparison of nonparalyzed and completely paralyzed subjects. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *14*(2), 183–186.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844. https://doi.org/10.1109/34.709601

Ho, T. K. (1995). *Random decision forests*. *1*, 278–282 vol.1. https://doi.org/10.1109/ICDAR.1995.598994

Hoffmann, U., Vesin, J.-M., Ebrahimi, T., & Diserens, K. (2008). An efficient P300-based brain–computer interface for disabled subjects. *Journal of Neuroscience Methods*, *167*(1), 115–125.

Holzinger, A. (2015). Data Mining with Decision Trees: Theory and Applications. *Online Information Review*, *39*(3), 437–438. https://doi.org/10.1108/OIR-04-2015-0121

John Lu, Z. Q. (2010). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. In *Journal of the Royal Statistical Society: Series A (Statistics in Society)* (Vol. 173, Issue 3, pp. 693–694). https://doi.org/10.1111/j.1467-985X.2010.00646_6.x

Jurcak, V., Tsuzuki, D., & Dan, I. (2007). 10/20, 10/10, and 10/5 systems revisited: their validity as relative head-surface-based positioning systems. *Neuroimage*, *34*(4), 1600–1611.

Kaufmann, T., Holz, E., & Kübler, A. (2013). Comparison of tactile, auditory, and visual modality for brain-computer interface use: a case study with a patient in the locked-in state . In *Frontiers in Neuroscience* (Vol. 7, p. 129). https://www.frontiersin.org/article/10.3389/fnins.2013.00129

Khairullah, E., Arican, M., & Polat, K. (2020). Brain-computer interface speller system design from electroencephalogram signals with channel selection algorithms. *Medical Hypotheses*, *141*, 109690. https://doi.org/https://doi.org/10.1016/j.mehy.2020.109690

Klem, G. H., Lüders, H. O., Jasper, H. H., & Elger, C. (1999). The ten-twenty electrode system of the International Federation. *Electroencephalogr Clin Neurophysiol*, *52*(3), 3–6.

Koles, Z. J., Lazar, M. S., & Zhou, S. Z. (1990). Spatial patterns underlying population differences in the background EEG. *Brain Topography*, *2*(4), 275–284.

Konrad, P., & Shanks, T. (2010). Implantable brain computer interface: Challenges to neurotechnology translation. *Neurobiology of Disease*, *38*(3), 369–375. https://doi.org/10.1016/j.nbd.2009.12.007

Kovelman, I. (2012). *Neuroimaging Methods* (pp. 43–59). Oxford, UK: Wiley-Blackwell.

https://doi.org/10.1002/9781444344035.ch4

Kretowski, M. (2019). *Evolutionary Decision Trees in Large-Scale Data Mining* (Vol. 59). Springer.

Krusienski, D. J., Sellers, E. W., McFarland, D. J., Vaughan, T. M., & Wolpaw, J. R. (2008). Toward enhanced P300 speller performance. *Journal of Neuroscience Methods*, *167*(1), 15–21.

Kübler, A., Nijboer, F., Mellinger, J., Vaughan, T. M., Pawelzik, H., Schalk, G., McFarland, D. J., Birbaumer, N., & Wolpaw, J. R. (2005). Patients with ALS can use sensorimotor rhythms to operate a brain-computer interface. *Neurology*, *64*(10), 1775–1777.

Kuncheva, L. I. (Ludmila I. (2014). *Combining pattern classifiers : methods and algorithms* (2nd ed.). Hoboken, New Jersey : Wiley.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning*, 609–616.

Leeb, R., Gwak, K., & Kim, D.-S. (2013). Freeing the visual channel by exploiting vibrotactile BCI feedback. *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 3093–3096.

Lelievre, Y., Washizawa, Y., & Rutkowski, T. M. (2013). Single trial BCI classification accuracy improvement for the novel virtual sound movement-based spatial auditory paradigm. *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 1–6.

Lemm, S., Blankertz, B., Curio, G., & Muller, K.-R. (2005). Spatio-spectral filters for improving the classification of single trial EEG. *IEEE Transactions on Biomedical Engineering*, *52*(9), 1541–1548.

Lenhardt, A., Kaper, M., & Ritter, H. J. (2008). An Adaptive P300-Based Online Brain–Computer Interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *16*(2), 121–130. https://doi.org/10.1109/TNSRE.2007.912816

Lerner, K. L. (2014). *Action Potential* (pp. 34–36).

Li, J., & Zhang, L. (2010). Regularized tensor discriminant analysis for single trial EEG classification in BCI. *Pattern Recognition Letters*, *31*(7), 619–628.

Li, Z., Tang, S., & Wang, H. (2002). *Pairwise coupling support vector machine and its application on handwritten digital recognition* (Vol. 2, pp. 1194–1198 vol.2). https://doi.org/10.1109/ICCCAS.2002.1178997

Looi, J. C. L., & Pring, W. (2020). Private metropolitan telepsychiatry in Australia during Covid-19: current practice and future developments. *Australasian Psychiatry*, 1039856220930675. https://doi.org/10.1177/1039856220930675

Lopez-Gordo, M. A., Sanchez-Morillo, D., & Valle, F. P. (2014). Dry EEG electrodes. *Sensors*, *14*(7), 12847–12870.

Luck, S. J. (Steven J. (2005). *An introduction to the event-related potential technique*. Cambridge, Mass. : MIT Press.

Makeig, S., Bell, A. J., Jung, T.-P., & Sejnowski, T. J. (1996). Independent component analysis of electroencephalographic data. *Advances in Neural Information Processing Systems*, 145–151.

Marini, F., Lee, C., Wagner, J., Makeig, S., & Gola, M. (2019). A comparative evaluation of signal quality between a research-grade and a wireless dry-electrode mobile EEG system. *Journal of Neural Engineering*, *16*(5), 54001.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems* (pp. 512–518).

Mathewson, K. E., Harrison, T. J. L., & Kizuk, S. A. D. (2017). High and dry? Comparing active dry EEG electrodes to active and passive wet electrodes. *Psychophysiology*, *54*(1), 74–82.

McBratney, A., Whelan, B., Ancev, T., & Bouma, J. (2005). Future directions of precision agriculture. *Precision Agriculture*, *6*(1), 7–23.

McMenamin, B. W., Shackman, A. J., Maxwell, J. S., Bachhuber, D. R. W., Koppenhaver, A. M., Greischar, L. L., & Davidson, R. J. (2010). Validation of ICA-based myogenic artifact correction for scalp and source-localized EEG. *Neuroimage*, *49*(3), 2416–2432.

Mitchell, T. M., & Learning, M. (1997). Mcgraw-hill science. *Engineering/Math*, *1*, 27.

Müller, K.-R., Tangermann, M., Dornhege, G., Krauledat, M., Curio, G., & Blankertz, B. (2008). Machine learning for real-time single-trial EEG-analysis: from brain–computer interfacing to mental state monitoring. *Journal of Neuroscience Methods*, *167*(1), 82–90.

Nicolas-Alonso, L. F., & Gomez-Gil, J. (2012). Brain computer interfaces, a review. *Sensors*, *12*(2), 1211–1279.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, *12*, 2825–2830.

Peters, R. (2020). Reflections on COVID-19 in Sydney, Australia. *City & Society*, *32*(1), n/a-n/a. https://doi.org/10.1111/ciso.12267

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

Piccione, F., Giorgi, F., Tonin, P., Priftis, K., Giove, S., Silvoni, S., Palmas, G., & Beverina, F. (2006). P300-based brain computer interface: reliability and performance in healthy and paralysed participants. *Clinical Neurophysiology*, *117*(3), 531–537.

Plöchl, M., Ossandón, J. P., & König, P. (2012). Combining EEG and eye tracking: identification, characterization, and correction of eye movement artifacts in electroencephalographic data. *Frontiers in Human Neuroscience*, *6*(2012), 278. https://doi.org/10.3389/fnhum.2012.00278

Polich, J. (2007). Updating P300: an integrative theory of P3a and P3b. *Clinical Neurophysiology : Official Journal of the International Federation of Clinical Neurophysiology*, *118*(10), 2128–2148. https://doi.org/10.1016/j.clinph.2007.04.019

Polikoff, J. B., Bunnell, H. T., & Borkowski Jr, W. J. (1995). Toward a P300-based computer interface. *RESNA'95 Annual Conference and RESNAPRESS and Arlington Va*, 178–180.

Pritchard, W. S. (1981). Psychophysiology of P300. *Psychological Bulletin*, *89*(3), 506.

Qi, Z., Chang, Z., Song, H., & Zhang, X. (2014). Application of support vector machine in the decision-making of maneuvering. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8631, Issue 2, pp. 352–358). https://doi.org/10.1007/978-3-319-11194-0_27

Qiao, Q., & Beling, P. (2016). Decision analytics and machine learning in economic and financial systems. *Formerly The Environmentalist*, *36*(2), 109–113. https://doi.org/10.1007/s10669-016-9601-x

Ramadan, R. A., & Vasilakos, A. V. (2017). Brain computer interface: control signals review. *Neurocomputing*, *223*, 26–44. https://doi.org/10.1016/j.neucom.2016.10.024

Ramoser, H., Muller-Gerking, J., & Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, *8*(4), 441–446. https://doi.org/10.1109/86.895946

Rutkowski, T. M. (2016). Robotic and virtual reality BCIs using spatial tactile and auditory oddball paradigms.(Report). *Frontiers in Neurorobotics*, *10*. https://doi.org/10.3389/fnbot.2016.00020

Samuel, A. L. (1988). Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress. In *Computer Games I* (pp. 366–400). Springer New York. https://doi.org/10.1007/978-1-4613-8716-9_15

Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., & Wolpaw, J. R. (2004). BCI2000: A general-purpose brain-computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering*, *51*(6), 1034–1043. https://doi.org/10.1109/TBME.2004.827072

Schlögl, A., Slater, M., & Pfurtscheller, G. (2002). Presence research and EEG. *Proceedings of the 5th International Workshop on Presence*, *1*, 9–11.

Scikit-learn. (n.d.). Retrieved June 22, 2020, from https://scikit-learn.org/stable/

Selim, A. E., Wahed, M. A., & Kadah, Y. M. (2009). *Machine learning methodologies in P300 speller Brain-Computer Interface systems* (pp. 1–9).

Sellers, E. W., & Donchin, E. (2006). A P300-based brain–computer interface: Initial tests by ALS patients. *Clinical Neurophysiology*, *117*(3), 538–548. https://doi.org/10.1016/j.clinph.2005.06.027

Shukla, X. U., & Parmar, D. J. (2016). Python - A comprehensive yet free programming language for statisticians. *Journal of Statistics and Management Systems*, *19*(2), 277–284. https://doi.org/10.1080/09720510.2015.1103446

Stigler, S. M. (1981). Gauss and the Invention of Least Squares. *The Annals of Statistics*, *9*(3), 465–474. https://doi.org/10.1214/aos/1176345451

Sutton, S., Braren, M., Zubin, J., & John, E. R. (1965). Evoked-potential correlates of stimulus uncertainty. *Science (New York, N.Y.)*, *150*(3700), 1187–1188. https://doi.org/10.1126/science.150.3700.1187

Tatum IV, Do, W. O. (2014). *Handbook of EEG Interpretation, Second Edition.* (2nd ed..). New York: Springer Publishing Company.

Thorp, C. K., & Steinmetz, P. N. (2008). Interference and noise in human intracranial microwire recordings. *IEEE Transactions on Biomedical Engineering*, *56*(1), 30–36.

Tomioka, R., Aihara, K., & Müller, K.-R. (2007). Logistic regression for single trial EEG classification. *Advances in Neural Information Processing Systems*, 1377–1384.

Unpingco, J. (2019). *Python for Probability, Statistics, and Machine Learning* (2nd ed. 20). Cham : Springer International Publishing : Imprint: Springer.

Venthur, B., Dähne, S., Höhne, J., Heller, H., & Blankertz, B. (2015). Wyrm: A Brain-Computer Interface Toolbox in Python. *Neuroinformatics*, *13*(4), 471–486. https://doi.org/10.1007/s12021-015-9271-8

Vespa Paul, M., Nenov Val, R., & Nuwer Marc, R. (1999). Continuous EEG Monitoring in the Intensive Care Unit: Early Findings and Clinical Efficacy. *Journal of Clinical Neurophysiology*, *16*(1), 1–13. https://doi.org/10.1097/00004691-199901000-00001

Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, *2*, 157.

Wang, Y., Gao, X., Hong, B., Jia, C., & Gao, S. (2008). Brain-computer interfaces based on visual evoked potentials: Feasibility of practical system designs. *IEEE Engineering in Medicine and Biology Magazine*, *27*(5), 64–71. https://doi.org/10.1109/MEMB.2008.923958

Whitham, E., Pope, K., Fitzgibbon, S., Lewis, T., Clark, C. R., Loveless, S., Broberg, M., Wallace, A., DeLosAngeles, D., Lillie, P., Hardy, A., Fronsko, R., Pulbrook, A., & Willoughby, J. (2007). Scalp electrical recording during paralysis: Quantitative evidence that EEG frequencies above 20Hz are contaminated by EMG. *Clinical Neurophysiology : Official Journal of the International Federation of Clinical Neurophysiology*, *118*, 1877–1888. https://doi.org/10.1016/j.clinph.2007.04.027

Whitmer, D., Worrell, G., Stead, M., Lee, I. K., & Makeig, S. (2010). Utility of independent component analysis for interpretation of intracranial EEG. *Frontiers in Human Neuroscience*, *4*, 184. https://doi.org/10.3389/fnhum.2010.00184

Wolpaw, J. R. (2007). Brain-computer interfaces as new brain output pathways. *Journal of Physiology*, *579*(3), 613–619. https://doi.org/10.1113/jphysiol.2006.125948

Wolpaw, J. R., Birbaumer, N., Mcfarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain–computer interfaces for communication and control. *Clinical Neurophysiology*, *113*(6), 767–791. https://doi.org/10.1016/S1388-2457(02)00057-3

Wu, D. (2017). Online and Offline Domain Adaptation for Reducing BCI Calibration Effort. *IEEE Transactions on Human-Machine Systems*, *47*(4), 550–563. https://doi.org/10.1109/THMS.2016.2608931

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D., & Steinberg, D. (2008). Top 10 algorithms in data mining. *An International Journal*, *14*(1), 1–37. https://doi.org/10.1007/s10115-007-0114-2

Yang, M.-D., Huang, K.-S., Lin, J.-Y., & Liu, P. (2010). Application of support vector machines to airborne hyper-spectral image classification. In *Lecture Notes in Electrical Engineering* (Vol. 67, pp. 439–444). https://doi.org/10.1007/978-3-642-12990-2_50

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, *1*(1), 69–90. https://doi.org/10.1023/A:1009982220290

Yao, L., Sheng, X., Mrachacz-Kersting, N., Zhu, X., Farina, D., & Jiang, N. (2019). Sensory Stimulation Training for BCI System Based on Somatosensory Attentional Orientation. *IEEE Transactions on Biomedical Engineering*, *66*(3), 640–646. https://doi.org/10.1109/TBME.2018.2852755

Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018). Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 268–282. https://doi.org/10.1109/CSF.2018.00027

Yoo, Y. (2017). Sequential detection of P300 waves for high-throughput brain-computer interfaces. *International Journal of Fuzzy Logic and Intelligent Systems*, *17*(2), 68–75.

Zhu, D., Garcia Molina, G., Mihajlovic, V., Aarts, R. M., & Stephnidis, C. (2011). *Online BCI implementation of high-frequency phase modulated visual stimuli* (S. P. Systems, A. array signal processing, & B. D. Lab (Eds.)).

Zhu, D., Molina, G. G., Mihajlovic, V., & Aarts, R. M. (2010). *Phase synchrony analysis for SSVEP-based BCIs* (Vol. 2, pp. V2-329-V2-333). https://doi.org/10.1109/ICCET.2010.5485465

8. APPENDICES

Appendix A: P300 Speller BCI Competition Dataset

.   .

This page is intentionally left blank.

This page is intentionally left blank.

Appendix B: Python code used in this study

Some methods are based on Wyrm Python toolbox with some modifications to adapt to the study.

The full source code is publicly available at https://gitlab.com/nguy1025/speller-bci

Define constants

```
ELOC = 'data/BCI_Comp_III_Wads_2004/eloc64.txt'

TRAIN_A = 'data/BCI_Comp_III_Wads_2004/Subject_A_Train.mat'
TRAIN_B = 'data/BCI_Comp_III_Wads_2004/Subject_B_Train.mat'

TEST_A = 'data/BCI_Comp_III_Wads_2004/Subject_A_Test.mat'
TEST_B = 'data/BCI_Comp_III_Wads_2004/Subject_B_Test.mat'

TRUE_LABELS_A =
'WQXPLZCOMRKO97YFZDEZ1DPI9NNVGRQDJCUVRMEUOOOJD2UFYPOO6J7LDGYEGOA5VHNEHBTXOO1TDOILUEE5B
FAEEXAW_K4R3MRU'
TRUE_LABELS_B =
'MERMIROOMUHJPXJOHUVLEORZP3GLOO7AUFDKEFTWEOOALZOP9ROCGZET1Y19EWX65QUYU7NAK_4YCJDVDNGQX
ODBEV2B5EFDIDNR'

MATRIX = ['abcdef',
          'ghijkl',
          'mnopqr',
          'stuvwx',
          'yz1234',
          '56789_']

# Features we care about
MARKER_DEF_TRAIN = {'target': ['target'], 'nontarget': ['nontarget']}
MARKER_DEF_TEST = {'flashing': ['flashing']}

N_INTENSIFY = 15  # 15 intensifications of row/column
N_STIMULI = 12  # 12 stimuli of each character

SEG_IVAL = [0, 700]  # The interval in milliseconds to cut around the markers
```

Load data

```
def load_mat_data(filename):
    STIMULUS_CODE = {
        # cols from left to right
        1: "agmsy5",
        2: "bhntz6",
        3: "ciou17",
        4: "djpv28",
        5: "ekqw39",
        6: "flrx4_",
        # rows from top to bottom
        7: "abcdef",
        8: "ghijkl",
        9: "mnopqr",
        10: "stuvwx",
        11: "yz1234",
        12: "56789_"
    }

    # load the matlab data
    data_mat = spio.loadmat(filename)
    # load the channel names (the same for all datasets
    eloc_file = ELOC
    with open(eloc_file) as fh:
        data = fh.read()
    channels = []
    for line in data.splitlines():
        if line:
            chan = line.split()[-1]
            chan = chan.replace('.', '')
            channels.append(chan)
    # fix the channel names, some letters have the wrong capitalization
    for i, s in enumerate(channels):
        s2 = s.upper()
        s2 = s2.replace('Z', 'z')
        s2 = s2.replace('FP', 'Fp')
        channels[i] = s2
    # The signal is recorded with 64 channels, bandpass filtered
    # 0.1-60Hz and digitized at 240Hz. The format is Character Epoch x
    # Samples x Channels
    data = data_mat['Signal']
    n_channels = len(channels)
    data = data.astype('double')

    # For each sample: 1 if a row/colum was flashed, 0 otherwise
    flashing = data_mat['Flashing'].reshape(-1)
    #flashing = np.flatnonzero((np.diff(a) == 1)) + 1
    tmp = []
    for i, _ in enumerate(flashing):
        if i == 0:
            tmp.append(flashing[i])
            continue
        if flashing[i] == flashing[i-1] == 1:
            tmp.append(0)
            continue
        tmp.append(flashing[i])
    flashing = np.array(tmp)
    # For each sample: 0 when no row/colum was intensified,
    # 1..6 for intensified columns, 7..12 for intensified rows
    stimulus_code = data_mat['StimulusCode'].reshape(-1)
    stimulus_code = stimulus_code[flashing == 1]
    # 0 if no row/col was intensified or the intensified did not contain
    # the target character, 1 otherwise
    stimulus_type = data_mat.get('StimulusType', np.array([])).reshape(-1)
    # The target characters
    target_chars = data_mat.get('TargetChar', np.array([])).reshape(-1)
    fs = 240  # Frequency samples
    data = data.reshape(-1, n_channels)
    timeaxis = np.linspace(
        0, data.shape[0] / fs * 1000, data.shape[0], endpoint=False)
    dat = Data(data=data, axes=[timeaxis, channels], names=[
              'time', 'channel'], units=['ms', '#'])
    dat.fs = fs
    # preparing the markers
    target_mask = np.logical_and((flashing == 1), (stimulus_type == 1)) if len(
        stimulus_type) > 0 else []
    nontarget_mask = np.logical_and(
        (flashing == 1), (stimulus_type == 0)) if len(stimulus_type) > 0 else []
    flashing = (flashing == 1)
    flashing = [[i, 'flashing'] for i in timeaxis[flashing]]
    targets = [[i, 'target'] for i in timeaxis[target_mask]]
    nontargets = [[i, 'nontarget'] for i in timeaxis[nontarget_mask]]
    dat.stimulus_code = stimulus_code[:]
    stimulus_code = zip([t for t, _ in flashing], [
                        STIMULUS_CODE[i] for i in stimulus_code])
    markers = flashing[:]
    markers.extend(targets)
    markers.extend(nontargets)
    markers.extend(stimulus_code)
    dat.markers = sorted(markers[:], key=lambda x: x[0])
    dat.target_chars = target_chars
    return dat
```

Define functions

```
def standardise_data(feature_vector, scaler, is_training):

    X = feature_vector.data
    y = feature_vector.axes[0]

    # Fit on training set only
    if is_training:
        scaler.fit(X)

    X = scaler.transform(X)
    # update value of pre-processed X
    feature_vector.data = X

    return feature_vector

def pca_apply(pca, feature_vector, is_training):

    X = feature_vector.data
    y = feature_vector.axes[0]

    print('pca_clean transform data isTraining {} shape BEFORE {}'.format(
        is_training, X.shape))

    if is_training:
        pca.fit(X)

    X = pca.transform(X)
    # update value of pre-processed X
    feature_vector.data = X
    print('pca_clean transform data isTraining {} shape AFTER {}'.format(
        is_training, X.shape))

    return feature_vector

def ica_apply(ica, feature_vector, is_training):

    X = feature_vector.data
    y = feature_vector.axes[0]

    print('ICA transform data isTraining {} shape BEFORE {}'.format(
        is_training, X.shape))

    if is_training:
        ica.fit(X)

    X = ica.transform(X)
    # update value of pre-processed X
    feature_vector.data = X
    print('ICA transform data isTraining {} shape AFTER {}'.format(
        is_training, (X.shape)))

    return feature_vector
```

```
def preprocessing_simple(dat, ica, pca, scaler, is_training, MRK_DEF, *args,
**kwargs):
    fs_n = dat.fs / 2
    b, a = proc.signal.butter(
        5, [30 / fs_n], btype='low')  # filter low-pass 30Hz
    dat = proc.filtfilt(dat, b, a)

    # Subsample the data to 20 Hz.
    dat = proc.subsample(dat, 20)

    # Convert a continuous data object to an epoched one.
    epo = proc.segment_dat(dat, MRK_DEF, SEG_IVAL)

    feature_vector = proc.create_feature_vectors(epo)

    if scaler is not None:
        feature_vector = standardise_data(feature_vector, scaler, is_training)

    if pca is not None:
        feature_vector = pca_apply(pca, feature_vector, is_training)

    if ica is not None:
        feature_vector = ica_apply(ica, feature_vector, is_training)

    return feature_vector, epo

# ## Method to process prediction
def process_prediction(fv, y_pred, nrepetition, true_label):
    label_len = len(true_label)

    # unscramble the order of stimuli
    unscramble_idx = fv.stimulus_code.reshape(
        label_len, N_INTENSIFY, N_STIMULI).argsort()
    static_idx = np.indices(unscramble_idx.shape)
    y_pred = y_pred.reshape(label_len, N_INTENSIFY, N_STIMULI)
    y_pred = y_pred[static_idx[0], static_idx[1], unscramble_idx]

    y_pred = y_pred[:, :nrepetition, :]

    # destil the result of the nrepetition runs
    y_pred = y_pred.sum(axis=1)

    y_pred = y_pred.argsort()

    cols = y_pred[y_pred <= 5].reshape(label_len, -1)
    rows = y_pred[y_pred > 5].reshape(label_len, -1)
    text = ''
    for i in range(label_len):
        row = rows[i][-1]-6
        col = cols[i][-1]
        letter = MATRIX[row][col]
        text += letter
    print
    print('---Constructed labels: %s' % text.upper())
    print('---True labels       : %s' % true_label)
    a = np.array(list(text.upper()))
    b = np.array(list(true_label))
    accuracy = np.count_nonzero(a == b) / len(a)
    print('---Accuracy: %.1f%%' % (accuracy * 100))
    return accuracy
```

Prepare and pre-process data

```
# # pre-process data
training_set = [TRAIN_A, TRAIN_B]
testing_set = [TEST_A, TEST_B]
labels = [TRUE_LABELS_A, TRUE_LABELS_B]

dat_training = [load_mat_data(training_set[0]), load_mat_data(training_set[1])]
dat_test = [load_mat_data(testing_set[0]), load_mat_data(testing_set[1])]

# # select channels we want
# CHANNELS = ['Cz']
CHANNELS = ["FCz", "C3", "Cz", "C4", "CP1", "CPz", "CP2", "Pz"]
dat_training = [proc.select_channels(
    dat_training[0], CHANNELS), proc.select_channels(dat_training[0], CHANNELS)]
dat_test = [proc.select_channels(
    dat_test[0], CHANNELS), proc.select_channels(dat_test[0], CHANNELS)]

scaler = [StandardScaler(), StandardScaler()]
# scaler = [None, None]

# choose the minimum number of principal components such that 95% of the variance is
retained.
pca = [PCA(0.95), PCA(0.95)]
# pca = [None, None] #choose the minimum number of principal components such that 95%
of the variance is retained.
# pca = [PCA(n_components=10), PCA(n_components=10)]

# ica = [FastICA(n_components=100,max_iter=1000, tol=0.005),
FastICA(n_components=100,max_iter=1000, tol=0.005)]
ica = [None, None]

fv_epo_train = [preprocessing_simple(dat_training[0], ica[0], pca[0], scaler[0], True,
MARKER_DEF_TRAIN), preprocessing_simple(
    dat_training[1], ica[1], pca[1], scaler[1], True, MARKER_DEF_TRAIN)]
fv_epo_test = [preprocessing_simple(dat_test[0], ica[0], pca[0], scaler[0], False,
MARKER_DEF_TEST), preprocessing_simple(
    dat_test[1], ica[1], pca[1], scaler[1], False, MARKER_DEF_TEST)]

fv_train = [x[0] for x in fv_epo_train]
epo_train = [x[1] for x in fv_epo_train]

fv_test = [x[0] for x in fv_epo_test]
epo_test = [x[1] for x in fv_epo_test]
```

Prepare classifiers

```
lda_covs = [None, None]  # classifiers

for subject in range(2):
    # train the lda with Covarience https://scikit-
learn.org/stable/modules/covariance.html
    cfy = proc.lda_train(fv_train[subject])

    print('Training done for subject {}'.format(subject))

    # add to list of classifiers
    lda_covs[subject] = cfy
```

```
# ## List of classifiers

# Define list of classifiers
classifiers = [
    [
        GradientBoostingClassifier(learning_rate=0.1),
        AdaBoostClassifier(n_estimators=100, random_state=0),
        KNeighborsClassifier(3),
        DecisionTreeClassifier(),
        RandomForestClassifier(n_estimators=100),
        LinearDiscriminantAnalysis(),
        QuadraticDiscriminantAnalysis(),
        LinearSVC(max_iter=10000),
        SVC(C=10, gamma='auto'),
    ],
    [
        GradientBoostingClassifier(learning_rate=0.1),
        AdaBoostClassifier(n_estimators=100, random_state=0),
        KNeighborsClassifier(3),
        DecisionTreeClassifier(),
        RandomForestClassifier(n_estimators=100),
        LinearDiscriminantAnalysis(),
        QuadraticDiscriminantAnalysis(),
        LinearSVC(max_iter=10000),
        SVC(C=10, gamma='auto')
    ]
]
```

Train classifiers

```
# ### Train data

# split data
log_cols = ['Accuracy', 'Classifier']

log_score_trainings = []

for subject in range(2):
    print("Start Subject: {}".format(subject))
    fv = fv_train[subject]
    X = fv.data
    y = fv.axes[0]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42)

    log_score_training = pd.DataFrame(columns=log_cols)

    for clf in classifiers[subject]:
        name = clf.__class__.__name__

        if name == "GradientBoostingClassifier":
            name = "GradientBoostingClassifier" + "_" + str(clf.learning_rate)

        if name == "LinearDiscriminantAnalysis":
            name = "LinearDiscriminantAnalysis" + "_" + \
                clf.solver + "_Shrinkage:" + str(clf.shrinkage)

        print("== Start Classifier: {}".format(name))
        clf.fit(X_train, y_train)

        y_pred = clf.predict(X_test)
        score = accuracy_score(y_test, y_pred)

        cm = confusion_matrix(y_test, y_pred)
        print(cm)

        TN = cm[1][1]
        FN = cm[0][1]
        TP = cm[0][0]
        FP = cm[1][0]

        print('TN: {}, FN: {}, TP: {}, FP: {}'.format(TN, FN, TP, FP))

        # Sensitivity, hit rate, recall, or true positive rate
        TPR = TP/(TP+FN)
        # Specificity or true negative rate
        TNR = TN/(TN+FP)
        # Precision or positive predictive value
        PPV = TP/(TP+FP)
        # Negative predictive value
        NPV = TN/(TN+FN)
        # Fall out or false positive rate
        FPR = FP/(FP+TN)
        # False negative rate
        FNR = FN/(TP+FN)
        # False discovery rate
        FDR = FP/(TP+FP)

        # Overall accuracy
        ACC = (TP+TN)/(TP+FP+FN+TN)
        print('\tAccuracy     : {:.2f}'.format(ACC))
        print('\tPrecision    : {:.2f}'.format(PPV))
        print('\tRecall       : {:.2f}'.format(TPR))
        F1 = 2*(PPV*TPR)/(PPV+TPR)
        print('\tF1           : {:.2f}'.format(F1))
        BM = TPR + TNR - 1
        print('\tInformedness : {:.2f}'.format(BM))

        print("== End Classifier: {}".format(name))
        print("== -------------")

        log_entry = pd.DataFrame([[name, score*100]], columns=log_cols)
        log_score_training = log_score_training.append(log_entry, sort=False)

    log_score_trainings.append(log_score_training)
```

Test classifiers on Training datasets & plot results

```
log_trainings = []

cols_repetitions = ["Repetition", "Accuracy", "ClfName"]

for subject in range(2):
    print("==== Start Subject: {}".format(subject))
    log_training_repeats = {}
    labels_train = dat_training[subject].target_chars[0]

    for clf in classifiers[subject]:
        name = clf.__class__.__name__

        if name == "GradientBoostingClassifier":
            name = "GradientBoostingClassifier" + "_" + str(clf.learning_rate)
        if name == "LinearDiscriminantAnalysis":
            name = "LDA_scikitlearn"

        print("==== Start classifier: {}".format(name))
        y_pred = clf.predict(fv_train[subject].data)

        log_training_repeat = pd.DataFrame(columns=cols_repetitions)
        for repeat_idx in range(1, N_INTENSIFY + 1):
            print("-- Repetition: {}".format(repeat_idx))
            name_repeat = name + "-" + str(repeat_idx)
            score = process_prediction(
                fv_train[subject], y_pred, repeat_idx, labels_train)
            log_entry = pd.DataFrame(
                [[repeat_idx, score*100, name]], columns=cols_repetitions)
            log_training_repeat = log_training_repeat.append(
                log_entry, sort=False)

        log_training_repeats[name] = log_training_repeat

        print("==== End classifier: {}".format(name))

    # append LDA cov
    name = 'LDA_wyrm'
    # add LDA cov estimator
    print("==== Start classifier: {}".format(name))
    y_pred = proc.lda_apply(fv_train[subject], lda_covs[subject])
    log_training_repeat = pd.DataFrame(columns=cols_repetitions)
    for repeat_idx in range(1, N_INTENSIFY + 1):
        print("-- Repetition: {}".format(repeat_idx))
        name_repeat = name + "-" + str(repeat_idx)
        score = process_prediction(
            fv_train[subject], y_pred, repeat_idx, labels_train)
        log_entry = pd.DataFrame(
            [[repeat_idx, score*100, name]], columns=cols_repetitions)
        log_training_repeat = log_training_repeat.append(log_entry, sort=False)

    log_training_repeats[name] = log_training_repeat

    log_trainings.append(log_training_repeats)

# ### Plot results of each repetitions for subject A

sns.set_style("whitegrid")
plt.figure(figsize=(8, 6))

marker = itertools.cycle(
    ('o', 'v', '^', '<', '>', 's', '8', 'p', 'X', 'D', "*"))

ax = plt.gca()
ax.set_prop_cycle(None)  # if matplotlib <1.5 use set_color_cycle

log_training_repeats = log_trainings[0]
for clf_name in log_training_repeats:
    log_training_repeat = log_training_repeats.get(clf_name)

    color = next(ax._get_lines.prop_cycler)['color']
    plt.plot('Repetition', 'Accuracy', data=log_training_repeat, label=clf_name,
             linestyle='dashed', markeredgecolor='none',  marker=next(marker),
color=color, markersize=12)

# Put the legend out of the figure
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
           borderaxespad=0., prop={'size': 14})
plt.xticks(ticks=list(range(1, 16)))
plt.xlabel('Repetitions')
plt.ylabel('Accuracy %')
plt.title('Classifier Accuracy TRAINING subject A')
plt.show()

# ### Plot results of each repetitions for subject B

sns.set_style("whitegrid")
plt.figure(figsize=(8, 6))

ax = plt.gca()
ax.set_prop_cycle(None)  # if matplotlib <1.5 use set_color_cycle

log_training_repeats = log_trainings[1]
for clf_name in log_training_repeats:
    log_training_repeat = log_training_repeats.get(clf_name)
    color = next(ax._get_lines.prop_cycler)['color']
    plt.plot('Repetition', 'Accuracy', data=log_training_repeat, label=clf_name,
             linestyle='dashed', markeredgecolor='none',  marker=next(marker),
color=color, markersize=12)

# Put the legend out of the figure
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
           borderaxespad=0., prop={'size': 14})
plt.xticks(ticks=list(range(1, 16)))
plt.xlabel('Repetitions')
plt.ylabel('Accuracy %')
plt.title('Classifier Accuracy TRAINING subject B')
plt.show()
```

Test classifiers on Testing datasets & plot results

```
log_testings = []

for subject in range(2):
    print("==== Start Subject: {}".format(subject))
    log_repeats = {}

    for clf in classifiers[subject]:
        name = clf.__class__.__name__

        if name == "GradientBoostingClassifier":
            name = "GradientBoostingClassifier" + "_" + str(clf.learning_rate)

        if name == "LinearDiscriminantAnalysis":
            name = "LDA"

        print("==== Start classifier: {}".format(name))
        y_pred = clf.predict(fv_test[subject].data)

        log_repeat = pd.DataFrame(columns=cols_repetitions)
        for repeat_idx in range(1, N_INTENSIFY + 1):
            print("-- Repetition: {}".format(repeat_idx))
            name_repeat = name + "-" + str(repeat_idx)
            score = process_prediction(
                fv_test[subject], y_pred, repeat_idx, labels[subject])
            log_entry = pd.DataFrame(
                [[repeat_idx, score*100, name]], columns=cols_repetitions)
            log_repeat = log_repeat.append(log_entry, sort=False)

        log_repeats[name] = log_repeat
        print("==== End classifier: {}".format(name))

    # append LDA cov
    name = 'LDA_wyrm'
    # add LDA cov estimator
    print("==== Start classifier: {}".format(name))
    y_pred = proc.lda_apply(fv_test[subject], lda_covs[subject])
    log_repeat = pd.DataFrame(columns=cols_repetitions)
    for repeat_idx in range(1, N_INTENSIFY + 1):
        print("-- Repetition: {}".format(repeat_idx))
        name_repeat = name + "-" + str(repeat_idx)
        score = process_prediction(
            fv_test[subject], y_pred, repeat_idx, labels[subject])
        log_entry = pd.DataFrame(
            [[repeat_idx, score*100, name]], columns=cols_repetitions)
        log_repeat = log_repeat.append(log_entry, sort=False)

    log_repeats[name] = log_repeat

    print("==== End classifier: {}".format(name))
    log_testings.append(log_repeats)

sns.set_style("whitegrid")
plt.figure(figsize=(8, 6))

# multiple line plot
ax = plt.gca()
ax.set_prop_cycle(None)  # if matplotlib <1.5 use set_color_cycle

log_repeats = log_testings[0]
for clf_name in log_repeats:
    log_repeat = log_repeats.get(clf_name)
    color = next(ax._get_lines.prop_cycler)['color']
    plt.plot('Repetition', 'Accuracy', data=log_repeat, label=clf_name,
linestyle='dashed',
             markeredgecolor='none',  marker=next(marker), color=color, markersize=12)

# Put the legend out of the figure
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
           borderaxespad=0., prop={'size': 14})
plt.xticks(ticks=list(range(1, 16)))
plt.xlabel('Repetitions')
plt.ylabel('Accuracy %')
plt.title('Classifier Accuracy TESTING subject A')
plt.show()


sns.set_style("whitegrid")
plt.figure(figsize=(8, 6))

# multiple line plot
ax = plt.gca()
ax.set_prop_cycle(None)  # if matplotlib <1.5 use set_color_cycle

log_repeats = log_testings[1]
for clf_name in log_repeats:
    log_repeat = log_repeats.get(clf_name)
    color = next(ax._get_lines.prop_cycler)['color']
    plt.plot('Repetition', 'Accuracy', data=log_repeat, label=clf_name,
linestyle='dashed',
             markeredgecolor='none',  marker=next(marker), color=color, markersize=12)

# Put the legend out of the figure
plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
           borderaxespad=0., prop={'size': 14})
plt.xticks(ticks=list(range(1, 16)))
plt.xlabel('Repetitions')
plt.ylabel('Accuracy %')
plt.title('Classifier Accuracy TESTING subject B')
plt.show()
```