Decoding EEG Imagination and Perception Signals

Stephen Charles

MSc in Data Science
The University of Bath, Bath, UK
2022

Decoding EEG Imagination and Perception Signals

Submitted by: Stephen Charles

Abstract

Subjects have been shown to mistake perceptual and imaginative experience (Perky, 1910). The synergy between imagination and perception is a subject of debate (Cavedon-Taylor, 2021; Brogaard and Gatzia, 2017; Nanay, 2010), and there have been studies (Dijkstra, Bosch and van Gerven, 2017) which attempt to clarify what connection these two modalities hold. The overall goal of this project is to create an automated pre-processing pipeline to clean EEG data. Following this, we can attempt to decode (classify) what a person is thinking about based upon their neural activity. Through this context, a comparison of stochastic and deterministic machine learning model performance is drawn. For classification purposes the advantage in a probabilistic approach would be confidence in outcome. Confidence intervals allow us to see where a model underperforms and draw conclusions as to why. In real world *Brain Computer Interface* applications, this kind of transparency in analysis would be beneficial for *Electroencephologram*. If there is high certainty for two classes which are similar in semantics/content, it would be safe to proceed with either class.

An automated pre-processing pipeline is also proposed, which attempts to clean a raw data signal, removing stationary signals and artifacts. This primarily uses a combination of python packages, most notably

- `Python MNE` (Gramfort et al., 2013a; Li et al., 2022) - A tool for use in neuroscience analysis with a high level sklearn like interface.
- `EEGLab` (Delorme and Makeig, 2004) - An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis.
- `Autoreject` (Jas et al., 2016) - A library to automatically reject bad trials and repair bad sensors in magneto/electroencephalography (M/EEG) data.
- `Pyriemann` (Barachant and King, 2015) - A machine learning library based on scikit-learn API. It provides a high-level interface for classification and manipulation of multivariate signal through Riemannian Geometry of covariance matrices.
- `IC-Label` (Pion-Tonachini, Kreutz-Delgado and Makeig, 2019) - An automated procedure to classify IC using a trained neural network on thousands of IC's.

A comparison between both AudViz and *Bath* datasets (Section 4.1) for the entire pipeline was performed across both all modalities (audio, visual and orthographic) and individually, which was shown to perform well with the prior. After 100 independent trials with unique random seeds, each with 5-fold cross validation, performance greater than the baseline of 3 class classification (33.3%) was not achieved on the latter. The nature of classification for imagination and perception is a broader problem than the lateralized classes in the AudViz dataset, which involves tasks separated by the left and right audio and visual responses.

All of the code required to reproduce the experiments described in later chapters is located here.

Contents

List of Figures

List of Tables

Acknowledgements

I dedicate this work to Gina Charles, my mother, whom without her support in recent years I would not be where I am now. The sacrifices she made to provide me future success is a rare act of selflessness, and to say that I am grateful would be greatly understated.

I'd also like to thank Holly Wilson for our abstract debates on the mind, which formed much of my critical thought process in the review.

Chapter 1

Introduction

1.1 Motivation

The overall goal of this project is to decode (classify) what a person is thinking about based upon their neural activity, and secondarily to compare the suitability of deterministic and stochastic machine learning approaches in this setting. We propose to take *Electroencephalogram* (Chapter 2.1.2) neural data off participants when they perform a particular task. This will then be decoded using various machine learning approaches (Chapter 2.3) to train on *perceived stimuli* and extract a label to correctly classify the signal, focusing on decoding *imagined stimuli*.

One particular study by Perky (1910), now called *The Perky Effect*, had subjects report an imaginary experience that reflected a projected image which was higher than the visible threshold. When quizzed afterwards they stated that they did not see anything. This suggests that subjects mistake perceptual experience for imaginative experience. That they possibly didn't have a perceptual experience but their unconscious perception influenced their imaginative experience.

In light of such phenomenon, I aim to research the synergy of imagination and perception (Chapter 2.1.4) to determine how possible the decodability of a classification signal depends on an individuals capacity for vivid imagination.

The primary objective will be to create an automated pre-processing pipeline to clean the input signal from artifacts, and determine if classification is possible with the imagination and perception *Bath* dataset (Section 4.1).

Secondarily, I intend to draw comparisons between *deterministic* and *stochastic* (Chapters 2.3, 2.4) machine learning approaches, as the latter has not been widely adopted in *Neuroscience*, yet Bielza and Larrañaga (2014) suggests this may change going forward.

Brain Computer Interfaces (Figure 1.1) measure neural signals where a participant is performing a particular task. These signals can be used with different kinds of neuroimaging modalities such as electroencephalography or functional magnetic resonance imaging. Once the neural signals are obtained we can attempt to decode some information about the task that the person was performing when the brain activity was measured. If we were to make measurements while a person was moving their hand to the left or right, and try to decode their intention, we can translate this signal into an external action, for example,

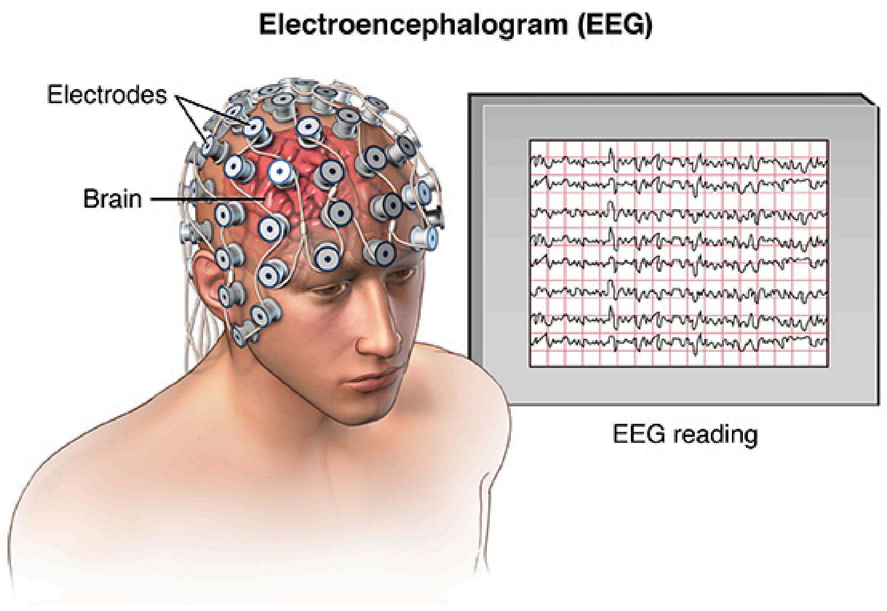


Figure 1.1: A commonly used *Brain Computer Interface*, *Electroencephalogram*.

moving a robot left or right. Imagination can also be captured, where the intent can be deciphered and decoded to then be translated into external actions. With this in mind, we can ask the question

"Does decodability depend on an individuals capacity for vivid imagination?"

As the capacity to imagine vividly varies from subject to subject, its unclear whether the ability to decode imagination is invariant to this phenomenon.

The different types of BCI inputs in our dataset include:

- **Imagination vs Perception**
  - Perceived stimuli: Directly interacting with sensory modalities.
  - Imagined stimuli: Recreate an abstract modality mentally.

  Imagined stimulus is more valuable as the BCI user can generate any picture or sound mentally, to try and drive BCI rather than relying on modality already present in their external environment.
- **Sensory Modalities**
  - Pictorial: imagining a picture.
  - Orthographic: imagining text.
  - Auditory: imagining sounds.

  For example, we can imagine an image, text and sound form of coffee. All are semantically coffee, but can be represented in different modalities.

The general areas we are interested in for these studies occur in the neocortex. The neocortex is one of the most recently evolved parts of the brain (Lee et al., 2019). The gyri and sulcus (brain ridges) create more surface area proportional to their quantity, and in general a higher number is linked to increased intelligence (Rushton and Ankney, 2009). We can divide the neocortex into different sensory modalities, of which these are processed in specific regions of the brain. Despite this, the architecture driving these are equivalent (Meijer et al., 2019).

If someone is born congenitally blind from birth, we might find that areas associated with visual processing can be repurposed into auditory processing (Burton, 2003). While processing for the visual and auditory cortices occurs uniformly across the entire brain, specific regions can be targeted which have a higher concentration of use for different tasks (Lewis, Beauchamp and DeYoe, 2000).

1.2 Deterministic vs. Probabilistic

Philosophically, it can be argued that crossing the road is both deterministic and probabilistic. We make the choice to cross safely based on past experiences, which are set variables, and given current events the same outcome would always be chosen. Such an assumption is deterministic. But when choosing to cross, there is a level of hesitation in the moment, only when our confidence in the priors reaches a threshold do we act and cross. Thus the same concept could be probabilistic.

In machine learning, a deterministic approach could for example, consist of a standard *Convolutional Neural Network*, later described in Chapter 2.3.2. Each input would produce the same output. A probabilistic model such as Bayesian CNN (Chapter 2.4.1) (Yamins et al., 2014) would have probability distributions for each neuron instead of linear weights, thus each input would produce a different probabilistic outcome.

For classification purposes the advantages in a probabilistic approach would be confidence in outcome, allowing us to see where a model underperforms and draw conclusions as to why. In real world BCI application, this kind of transparency in analysis would be beneficial for EEG. If there is high certainty for two classes which are similar in semantics/content, it would be safe to proceed with either class.

The secondary aim of this project is to analyse the performance of both approaches when applied to our dataset.

1.3 Data Collection

Our dataset aims to answer the following questions:

- Is there an overall variance depending on how vividly an individual imagines?
- Which neural signals (regions and timepoints) enable us to classify invariant of their vividness, such as regions related to semantics.

The data is to be collected at the University of Bath, involving 20+ participants neural data, collected from 128 EEG channels using ANT-Neuro hardware and a 10:20 montage[1]. Auditory and visual imagery vividness will be collected separately. Each electrode records a combination of inhibitory and excitatory neurons in the billions. Such a complex signal is considered macro level, due to the signal detection occurring on the surface of the brain. Neural activity of approximately 5-6cm^2 is captured, by comparison fMRI captures around 2mm^2. EEG is therefore quite low in spatial resolution, but very fast in temporal resolution, of which we can record 16000 samples a second.

Gel is placed on the electrodes to boost conductivity and an amplifier is used to boost the signal by 100,000 times. Such a process has a very high signal to noise ratio, due to the

[1]The 10:20 montage refers to the 10% or 20% inter-electrode distance (Fig 1.2)

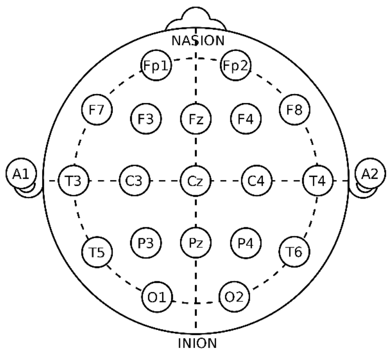


Figure 1.2: The 10:20 montage is a method to describe the locations of applied scalp electrodes in EEG (Reaves et al., 2021).

signal dispersing through the scalp and skull, transferring outwardly to the electrodes. The brain stem and hippocampus is difficult to measure with EEG as a result, thus processing of the raw data to remove noise will be required.

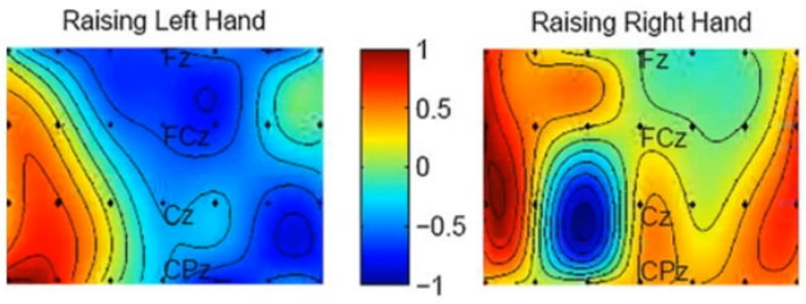


Figure 1.3: A common spatial pattern applied where someone is imagining raising their left or right hand (Mane, Biradar and Shastri, 2015).

Lateralisation will also have an impact on the kinds of data used; visual tasks have their processing roles reversed from left to right, where auditory tasks involving speech specifically occur mostly use the left (Figure 1.3).

1.4 Semantic Representation

Semantic decoding refers to the identification of semantic concepts (Rybář, Poli and Daly, 2021). Previous work has shown that multiple categories can be classified accurately, provided that they are semantically different. Semantic difference can be displayed using a technique called *Hyperalignment* (Haxby et al., 2020). We have selected *penguin, guitar and flower* for categories to be used in the study using this method (Figure 1.5). They were chosen because of their high semantic difference, in the hope that in the brain there is also a distinct representation to measure.

The study involves people performing both imagination and perception tasks. For example, a penguin is shown on the screen, and they would be asked to reconstruct the picture mentally. The three categories of penguin, guitar and flower will be represented in three sensory modalities (Macpherson, 2011), pictorial, orthographic and auditory.

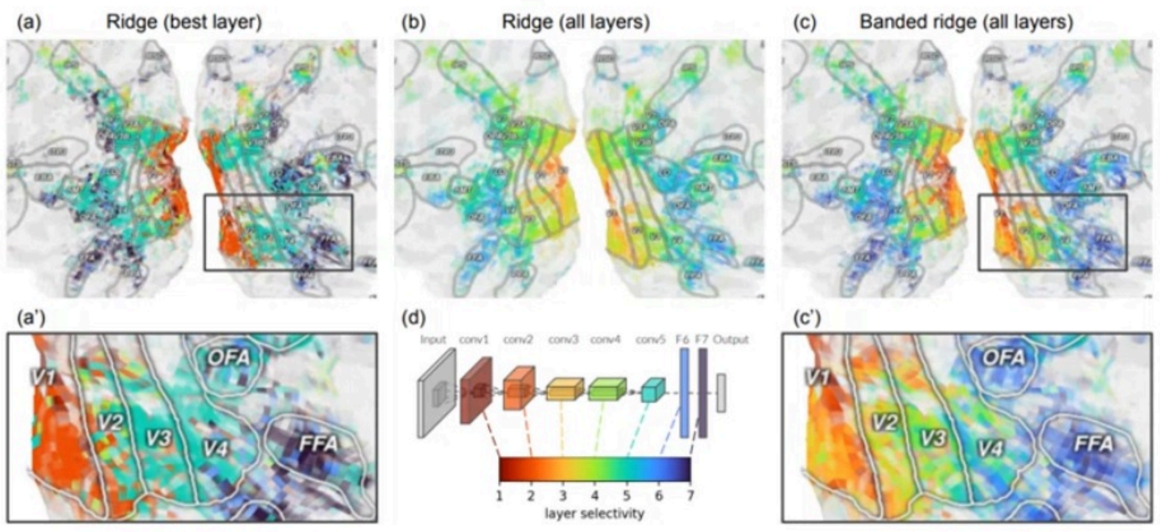


Figure 1.4: Model comparison in terms of layer mapping to the visual cortex (Hartmann, Schirrmeister and Ball, 2018; Dupre La Tour et al., 2021).

1.5 Preprocessing

We intend to experiment with both the raw and pre-processed data for classification. The proposed preprocessing steps are shown in Figure 1.6. In removing noise, the intention is that we classify based upon the intended signal. By classifying on the raw data, we establish a baseline by which we can judge improvements made by preprocessing.

The proposed steps are as follows:

- Import in the raw data and events.
- Downsample the data.
- Apply a bandpass filter.
  - There will be unwanted noise in the day from the environment and muscle movement. Use a high and low pass filter. High pass to remove DC components. Low filter to remove high frequency components.
- Re-reference the data.
  - Provides a baseline activity of physiological noise. We get this from the reference electrodes, usually positioned on the mastoid bone (behind ear).
- Inspect the electrodes and noisy channels.
  - Reject or interpolate. We only reject channels that are often noisy rather than ones which show periodic noise.
- Epoching.
  - Detecting experimental events based on triggers.
  - Reject bad epochs.
- Run ICA to reject noisy components.
  - This helps us get rid of components of the data that are heavily influenced by motor related artefacts.
- Detect experimental events.
  - We are primarily interested in activity associated with our semantic modalities, thus everything else can be considered as noise.

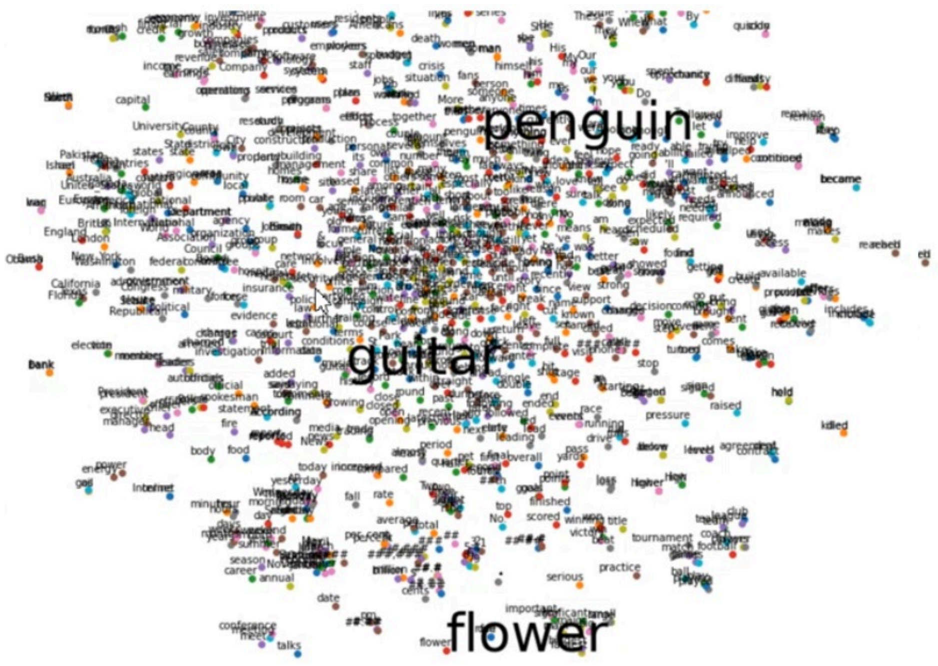


Figure 1.5: Three semantically different categories to be used in the imagination study displayed using Hyperalignment akin to Haxby et al. (2020).

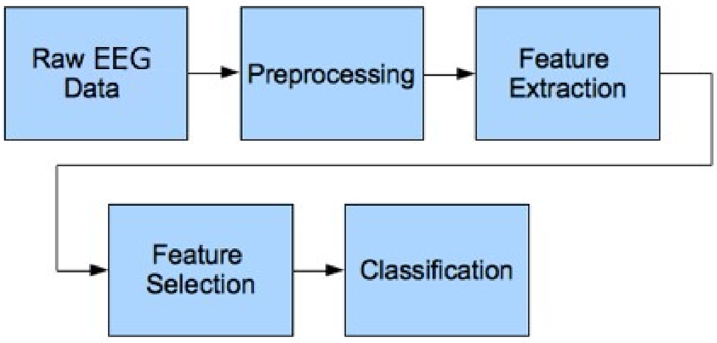


Figure 1.6: The proposed pre-processing pipeline.

1.6 Pipeline

The initial approach would be to combine all three sensory modalities and apply a machine learning decoder to obtain the categorical output. We can then follow up with further investigative study decoding individual modalities linked to the concept of the associated imagination, and be able to ignore semantics such as a particular instance of the object.

When we want to perform classification from independent EEG channels (electrodes) we can use raw data and deep learning algorithms to bypass any pre-processing steps as such models are capable of implicitly finding the associated patterns. Even so, pre-processing in this case may prove advantageous, and certainly for other algorithms such as *Support Vector Machines*, pre-processing input is necessary.

1.6.1 Preprocessing

Initial pre-processing would be to remove artefacts (such as eye movement blinks which cause spikes in amplitude), and bad channels (when the electrodes suffer from low impedance). Time window selection such as fourier/wavelet/spatial transforms would also allow us to extract specific features useful for classification (Mane, Biradar and Shastri, 2015). Image transforms such as in Figure 1.3 are useful translations as inputs for *Convolutional Neural Networks*.

Visual processing occurs much like a CNN in Figure 1.4, where we have various kinds of filters, orientations and colours (Hartmann, Schirrmeister and Ball, 2018; Dupre La Tour et al., 2021). Different layers deal with different types of information.

1.6.2 Machine Learning Models

The secondary goal of this project is to assess whether probabilistic models (which introduce more model complexity) have advantages over the standard deterministic approaches performed previously. There are many to choose from to draw comparisons from, but I will focus on *Support Vector Machines*, *Random Forests* and *Neural Networks* as these were shown to be the most popular in the field (Reaves et al., 2021).

Support Vector Machines

Dabas et al. (2018) used *Support Vector Machines* to classify EEG based emotional signals. They reported notably that SVMs in this instance were more accurate than a *Naive Bayes* approach. Interesting to note here, that the Naive Bayes method is not *strictly* Bayesian and could be arguably deterministic in this use case.

Random Forests

Bellman et al. (2018) used *Random Forests* to perform classification predictions to measure an individuals capability of facial recognition, and states that

> "It was found that, outperforming previous works, unaware facial recognitions could be detected with fairly high accuracies using a method that combines multiple sensors from a BCI device and utilizing out-of-the-box classification methods."

Thus there is a possible argument for model simplicity of the deterministic approach.

Neural Networks

Previous work for mapping CNN layers to brain regions (Yamins et al., 2014) uses the *voxelwise encoding* (Wu, David and Gallant, 2006) framework. Initially, brain activity is recorded while a subject is exposed to visual stimuli. The same stimulus is then shown to a pretrained CNN. Lastly, a regression model is trained on each *voxel*[2] to predict brain activity to classify features.

It could be proposed that we could classify each imagined category using a deterministic convolutional neural network (Roy et al., 2019) initially, then adopt a more probabilistic

[2]A voxel is a 3D pixel.

model such as a Bayesian CNN (Yamins et al., 2014). In the context of neuroscience, an excellent review of their use in the field is given by Bielza and Larrañaga (2014), of which he states that

> "Despite their applicability in many fields, they have been little used in neuroscience, where they have focused on specific problems, like functional connectivity analysis from neuroimaging data."

Of which presents itself that the probabilistic approach in neuroscience is largely unexplored.

1.7 Resources Required

It is anticipated that the dataset, consisting of 128 independent EEG channels (features), and the semantic/modality labels, will be computationally expensive. This will be noted before the implementation stage to be considered when choosing machine learning models. Access to the HEX (Bath, 2022) cluster is possible should the need arise however, should this level of scale be required.

Tools for EEG analysis such as `MNE` (Gramfort et al., 2013a) and `EEGLab` (Delorme and Makeig, 2004) in python and matlab will be used to perform pre-processing steps for later input to the machine learning pipelines. Other python packages such as `Numpy, Pandas, Matplotlib, Seaborn, Tensorflow, Keras` and `Sklearn` will also likely prove useful in the design stage, described further in Chapter 3. `Bambi` (Capretto et al., 2020) will likely be a great exploration tool for the analysis of probabilistic methods.

Chapter 2

Literature and Technology Survey

This review is split up into two main sections. Neuroscience in Section 2.1 provides context towards the main project aim towards decodability of imagination and perception. The Machine Learning review in Sections 2.3 and 2.4 cover a brief history and explanation of each method commonly used, and propositions for use in this study.

2.1 Neuroscience

While this project is primarily based upon building a machine learning pipeline, comparing deterministic and probabilistic models applied to EEG data, background context is needed to draw conclusions from the classification and predictions computed. The overarching aim from the machine learning model is to determine if imagination and perception have any correlative forms when decoding.

We begin with a discussion on *Brain Computer Interfaces* in Section 2.1.1, and how this fits into our use case with the use of *Electroencephologram* expanded upon in Section 2.1.2. This section also covers the data collection, pre-processing and common machine learning approaches taken previously.

As we aim to determine the decodability for vivid imagination, underlying context into the current discourse in this area has been provided in Section 2.1.4. *Aphantasia*, for example, (Section 2.1.5) is a phenomenon where individuals are blind to the experience of visual imagery in the mind. Such conditions need special consideration in our study, and may explain discrepancies in future outcomes.

2.1.1 Brain Computer Interfaces

Technological advancement in neuroscience has provided solutions to many problems through the use of captured neurological signals using *Electroencephalography* (EEG) (Olejniczak, 2006; Mane, Biradar and Shastri, 2015) and functional Magnetic Resonance Imaging (fMRI) (Gore, 2003). Neurological data capture devices such as these when combined with a computer interface to control external devices are often called *brain computer interfaces*. BCI's allow the control of these devices through brain wave interpretation. They have assisted in the detection and treatment of neurological afflictions through neurofeedback signals specific to each person. Other uses of BCI's allow the use of external devices such as prostheses or autonomous systems.

BCI's can be used for communication interfaces; someone with paralysis could potentially move a mouse cursor through imagined motor movements. Studies such as these (Sun, Hsieh and Syu, 2020) also give insight and understanding of the human mind. EEG's are a cost effective and portable solution when compared with fMRI (Rybář, Poli and Daly, 2021). EEG's strengths lie in its high temporal resolution, with a weakness in spatial resolution.

The primary challenges of such systems are their inherent noise associated with the received signal, and the ability to acquire detailed information from the observed signals. In this vein, more collaboration between computer scientists and neuroscientists is required to keep up with advancements in both fields. EEG signals have inherent non-stationarity which complicates the ability for a model to genereralise; signals vary between subjects depending on their current mind state.

2.1.2 Electroencephologram

Electroencephologram (EEG) gives us a measure of the electrical activity in the brain, recording frequencies observed through the brain's current activity. EEG signals were notably discovered in 1875, through Richard Caton's work with animals (Hosseini, Hosseini and Ahi, 2020), the term itself was claimed by Dr. Hans Berger in 1924 upon successfully recording the first brain signals using this approach. Olejniczak (2006) states that EEG is

> "A graphic representation of the difference in voltage between two different cerebral locations plotted over time, mostly consisting of synaptic activity, though contaminated with noise from other sources and distorted by the signals measurement through the skull."

EEG is often used in neuroscience, of which the primary modes of study have been in motor, cognitive and sensory imaging. Its high temporal resolution and low cost have made it popular, in part due to its non-invasive techniques. The drawback is that the skull obscures the information transferred, which means that EEG has a low signal to noise ratio, compounded by other noise artifacts such as motion. The signals themselves also have inconsistency between individuals, influenced by emotional state and movement.

2.1.3 Data Collection and Pre-Processing

EEG data collection typically focuses on target frequencies known to be associated with a particular research problem. Data collection through electrode placement is classified into a 10:20 montage in our study, which adheres to a national standard (Oostenveld and Praamstra P, 2001). In the 10:20 system (Figure 1.2), electrodes are 10% and 20% of the skulls left to right, and front to back, distance apart respectively.

Wavelengths used in EEG vary between delta ($\delta < 4$Hz), theta ($\theta \approx$ 4-8Hz), alpha ($\alpha \approx$ 8-12Hz), beta ($\beta \approx$ 12-30Hz) and gamma ($\gamma \approx$ 30-45Hz). Delta waves are typically removed in the pre-processing steps, as they are assumed to be less useful.

Denoising

Noise removal in the initial stage uses an approach called *denoising*. Most commonly, the technique applies a bandpass filter which allows a specific frequency to be investigated. As mentioned before, low frequencies such as delta ($\delta < 4$Hz) are associated with muscle-movement and generally the first to be removed. Often, more dynamic filters are used, such as Volterra (Wu, Zhang and Xiaojun, 2019) or Butterworth (Molla et al., 2020) for more complex tasks. More complex still, are fully adaptive filters such as the one used in Torse and Desai (2016), which have use cases in epilepsy detection.

By selecting a frequency range, much of the noise is removed, yet artifacts can still remain. To combat this, a technique called *Independent Component Analysis* is used to automatically remove them (Torse and Desai, 2016). In contrast to *Principal Component Analysis*, which is used for dimensionality reduction, ICA aims to separate the information by transforming it to a maximally independent basis.

As EEG data is non-stationary, *time-frequency* analysis gives us temporal information, indicating when artifacts are likely to occur. The difficulty in artifact removal (Zotev et al., 2014) is to isolate movement noise such as heart rate and eye blink. EEG data most commonly employs *Wavelet Transforms* (Collazos-Huertas et al., 2020) to assist in the denoising process. This filter has the added benefit of dimensionality reduction, caused in part by the reconstruction process. Zotev et al. (2014) uses a variation of this called the *Flexible Analytic Wavelet Transform*.

Fourier Transforms are used to deconstruct a received signal into its component frequencies, however time information is lost through this method. Because of this, it is not often used with EEG. There are cases (Santoso, 2020) where time information can be preserved through the use of windowing, applying FT to subsets of the data.

Feature Extraction

At the feature extraction stage, regression models (Russel and Norvig, 2002; Bishop, 2006) are commonly used. Some examples include *Support Vector Machines* (Zhang, 2020), *Principal Component Analysis* (Miranda, Aranha and Ladeira, 2019), *Ensemble Learning* (Li et al., 2020) and *Extreme Gradient Boosting* (Russel and Norvig, 2002).

More recently, *Neural Networks* (Section 2.3.1) and *Convolutional Neural Networks* (Section 2.3.2) have become popular due to their ability to approximate underling functions in data irrespective to noise (Luo, Hu and Li, 2020).

Classification and Prediction

The accuracy of the classification stage is dependant on the quality and amount of features extracted during the pre-processing steps. *Transfer Learning*, where we use previously trained neural networks, would allow for more accurate predictions from smaller datasets, should the need arise in our study. Reaves et al. (2021) suggests that transfer learning will be an important procedure to adopt in EEG classification going forward. As there are inherent differences in signals between individuals, this process would allow for shortened learning times and to generalise to a larger distribution.

Similarly to the feature extraction stage, many of the same models are used in the classification and prediction step. The most popular methods used previously, summarised in Reaves et al. (2021), suggests *Support Vector Machines*, *k-Nearest Neighbour* and *Convolutional Neural Networks* dominate the field of EEG analysis.

2.1.4 Imagination and Perception

Cognitive penetration refers to the proposition that perceptual experiences are influenced by personal mental states such as our beliefs, expectations and emotions (Teng, 2021). Macpherson (2012) presents various arguments for its existence. She states that

> "It is theorised to occur when one's perceptual experience is altered by the states of one's cognitive system."

The theory is that there are two processes which together, if true, would amount to defining *cognitive penetration*. The first process suggests that cognitive states cause visual imagining to occur, and the content of such imaginations reflect ones beliefs or desires. The second process is such that visual imagination interacts with perceptual processing to yield one state with *phenomenal character*, as evidenced by the *Perky Effect* (Perky, 1910). Shoemaker (1994) describes that

> "The phenomenal character of an experience is what it is like subjectively to undergo the experience. If you are told to focus your attention upon the phenomenal character of your experience, you will find that in doing so you are aware of certain qualities."

This character would be determined by both the processing underlying the visual imagining and perception.

Carbon (2014) suggests that states that are influenced by cognitive states can interact with the *phenomenal character* of perpetual experiences. Such interactions could possibly account for cases where illusory perception is detected.

The relevance to our study is the interaction between imagination and perception. If we can find correlation between the two signals it would provide further evidence for phenomenon like *cognitive penetration*. We aim to determine if decodability depends on an individuals capacity for vivid imagination. The distinction between illusory perception and imagination is quite blurred, and with the pipeline we intend to define a boundary between the two through machine learning classification.

There are cases however, where people have been shown to be "blind" to imagination.

2.1.5 Aphantasia

People can completely lack the experience of visual imagery, a condition now known as *Aphantasia* (Keogh, 2018; Zeman, Dewar and Della Sala, 2015). Research is needed to determine if such a phenomenon is based on poor metacognition; visual images appear in their mind, but they are unable to observe them. Studies have suggested that aphantasics self-rated visual object imagery was below average, yet spatial imagery was above (Zeman et al., 2010). This suggests that aphantasia is a condition involving a lack of sensory and phenomenal imagery, and not a lack of metacognition or the inability to introspect.

My personal experience with this leads me to believe that I am one such aphantic. Conversations with my family suggest I am an isolated case, as when questioned they are immediately able to describe a specific object in detail. For example, when asked to visualise an apple, they immediately describe its dimensions, color, smell, location etc. Hypothetically, there is a possibility this is due to the inability to connect imagination with recall.

2.2 Python MNE

The following example for visualising the statistical significance thresholds in EEG data is taken from this MNE-Python tutorial. It shows the basics of using EEG data to perform exploratory and visual analysis using MNE-Python (Gramfort et al., 2013b). Before any machine learning is performed, EDA techniques such as these will be carried out to ensure the data we are feeding the models is optimal and what we expect.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind

import mne
from mne.channels import find_ch_adjacency, make_1020_channel_selections
from mne.stats import spatio_temporal_cluster_test

np.random.seed(0)

# Load the data
path = mne.datasets.kiloword.data_path() / 'kword_metadata-epo.fif'
epochs = mne.read_epochs(path)
# These data are quite smooth, so to speed up processing we'll (unsafely!) just
# decimate them
epochs.decimate(4, verbose='error')
name = "NumberOfLetters"

# Split up the data by the median length in letters via the attached metadata
median_value = str(epochs.metadata[name].median())
long_words = epochs[name + " > " + median_value]
short_words = epochs[name + " < " + median_value]
```

Listing 2.1: Visualising statistical significance thresholds on EEG data (Part 1).

If we wish to test a specific point and time and space, we can convert the data in a Pandas Dataframe structure. The `mne.Epochs` class has a method called `mne.Epochs.to_data_frame()`, returning the data in a DataFrame structure. This is shown in Listing 2.2.

Next a mass-univariate analysis at all sensors and time points can be explored. This requires a correction for multiple tests. Here, they use a cluster-based permutation to allow for the derivation of power from the spatio-temporal correlation structure underlying in the data (Listing 2.3).

The results of the mass univariate analysis can be easily visualised by plotting `mne.Evoked` objects as images using the method `mne.Evoked.plot_image()`. Shown in Listing 2.4, channels are grouped by regions of interest to illustrate the localising effects on the head.

```python
time_windows = ((.2, .25), (.35, .45))
elecs = ["Fz", "Cz", "Pz"]
index = ['condition', 'epoch', 'time']

# display the EEG data in Pandas format (first 5 rows)
print(epochs.to_data_frame(index=index)[elecs].head())

report = "{elec}, time: {tmin}-{tmax} s; t({df})={t_val:.3f}, p={p:.3f}"
print("\nTargeted statistical test results:")
for (tmin, tmax) in time_windows:
    long_df = long_words.copy().crop(tmin, tmax).to_data_frame(index=index)
    short_df = short_words.copy().crop(tmin, tmax).to_data_frame(index=index)
    for elec in elecs:
        # extract data
        A = long_df[elec].groupby("condition").mean()
        B = short_df[elec].groupby("condition").mean()

        # conduct t test
        t, p = ttest_ind(A, B)

        # display results
        format_dict = dict(elec=elec, tmin=tmin, tmax=tmax,
                           df=len(epochs.events) - 2, t_val=t, p=p)
        print(report.format(**format_dict))
```

Listing 2.2: Visualising statistical significance thresholds on EEG data (Part 2).

```python
# Calculate adjacency matrix between sensors from their locations
adjacency, _ = find_ch_adjacency(epochs.info, "eeg")

# Extract data: transpose because the cluster test requires channels to be last
# In this case, inference is done over items. In the same manner, we could
# also conduct the test over, e.g., subjects.
X = [long_words.get_data().transpose(0, 2, 1),
     short_words.get_data().transpose(0, 2, 1)]
tfce = dict(start=.4, step=.4)  # ideally start and step would be smaller

# Calculate statistical thresholds
t_obs, clusters, cluster_pv, h0 = spatio_temporal_cluster_test(
    X, tfce, adjacency=adjacency,
    n_permutations=100)  # a more standard number would be 1000+
significant_points = cluster_pv.reshape(t_obs.shape).T < .05
print(str(significant_points.sum()) + " points selected by TFCE ...")
```

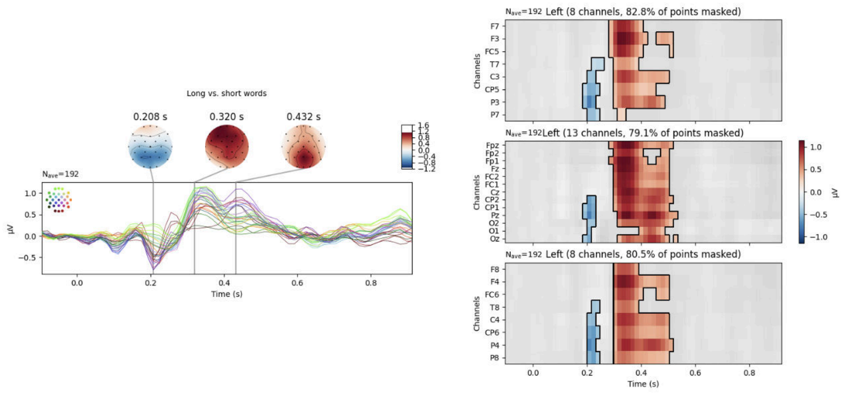
Listing 2.3: Visualising statistical significance thresholds on EEG data (Part 3).

```python
# We need an evoked object to plot the image to be masked
evoked = mne.combine_evoked([long_words.average(), short_words.average()],
                            weights=[1, -1])  # calculate difference wave
time_unit = dict(time_unit="s")
evoked.plot_joint(title="Long vs. short words", ts_args=time_unit,
                  topomap_args=time_unit)  # show difference wave

# Create ROIs by checking channel labels
selections = make_1020_channel_selections(evoked.info, midline="12z")

# Visualize the results
fig, axes = plt.subplots(nrows=3, figsize=(8, 8))
axes = {sel: ax for sel, ax in zip(selections, axes.ravel())}
evoked.plot_image(axes=axes, group_by=selections, colorbar=False, show=False,
                  mask=significant_points, show_names="all", titles=None,
                  **time_unit)
plt.colorbar(axes["Left"].images[-1], ax=list(axes.values()), shrink=.3,
             label="µV")

plt.show()
```



Listing 2.4: Visualising statistical significance thresholds on EEG data (Part 4).

2.3 Deterministic Machine Learning

Deterministic machine learning is a model, which when given a specific input, will have the same output value. *Probabilistic* (Section 2.4) machine learning is the converse to this, in that when a model receives a specific input, the output is generated by a probability distribution and thus will be different upon each iteration of the same input.

Reaves et al. (2021) discovers in his review that the most commonly applied machine learning algorithms in neuroscience for EEG data are *Neural Networks, Random Forests* and *Support Vector Machines.* A brief overview and description of each of these is covered here, and I plan to use these for the primary decodability objectives. They will also be a useful baseline to compare against the probabilistic models, if time permits, for the secondary objective.

2.3.1 Artificial Neural Networks

Since Rumelhart, Hinton and Williams (1986) developed back-propagation, *Neural Networks* became a popular modelling choice, and by 1995 they began to have competition from other algorithms. It wasn't until Santana et al. (2012) released his paper on *Deep Learning* that domination began, and to date they have been used as classifiers, value predictors, auto encoders and reinforcement learning agents. Understanding how the brain works is still a widely debated topic in neuroscience (Lisman, 2015), and analogies are often drawn between the two.

Deep learning algorithms make use of different transformations and functions to approximate a complex pattern in a dataset. A simple *Artificial Neural Network* (ANN) is the foundational concept of this approach, the history of which is well reviewed in Schmidhuber (2015).

ANN's consist of a linear series of non-linear transformations, alternatively defined as *feed-forward neural networks.* They were conceived to be loosely based on the process of biological neurons; lots of simple signals combined allow for complex evaluation. Figure 2.1 shows arrows representing linear transformations between nodes (equivalent to a neuron biologically), which combine to transform four values from the *input layer* $\boldsymbol{x_D}$ to two in the *output layer* $\boldsymbol{y_K}$.

A generalised linear model is expressed in terms of a weighted sum of fixed basis functions $\phi_m(\boldsymbol{x})$. A single layer of a neural network is represented as

$$y(\boldsymbol{x}) = g\left\{\sum_{m=1}^{M} w_m \phi_m(\boldsymbol{x})\right\} \tag{2.1}$$

where g represents a link function. A neural network with two layers of weights $\boldsymbol{w}$ and $\boldsymbol{V}$ is defined as

$$y(\boldsymbol{x}) = g\left\{\sum_{m=1}^{M} w_m \phi\left(\sum_{d=1}^{D} v_{md} x_d\right)\right\} \tag{2.2}$$

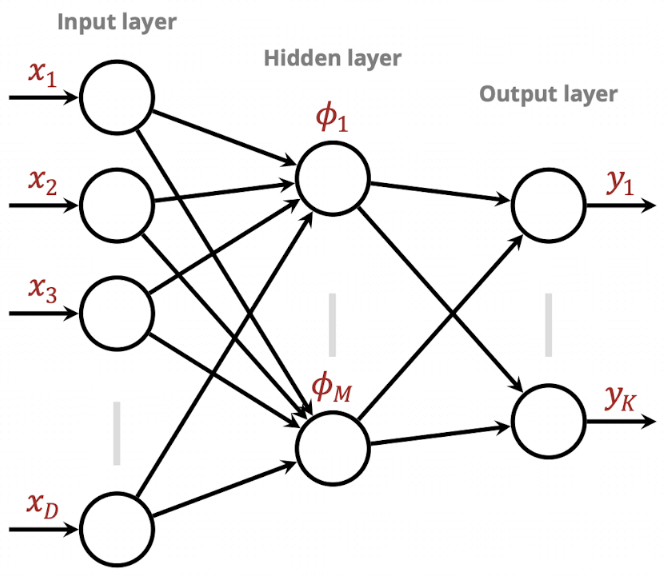


Figure 2.1: A fully connected artificial neural network.

which in contrast to Equation 2.1 has adaptive basis functions ϕ_m, along with the weight vector $\boldsymbol{w} = (w_1, \ldots, w_M)$ and parameters $\boldsymbol{V}$. The parameters represent the weights between the input and hidden layer, thus $\boldsymbol{V}$ is a $M \times D$ matrix. The architecture can be extended far beyond this representation to more layers, which then becomes a deep neural network.

A constant input or basis function is then added, referred to as the bias, generally defined as an extra hidden unit indexed by zero $x_0 = 1$ and $\phi_0(\boldsymbol{x}) = 1$.

The basis function in a linear model is called an activation function in a neural network. This introduces non-linearity and allows the model to fit more complex functions, becoming a universal function approximator (Hornick, 1989). Hodgkin and Huxley (1952) describes how neurons in the brain use action potential firing in such a manner, akin to activation functions.

The most commonly used activation function is the *rectified linear unit* (ReLU)

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \ \forall \ \phi \in [0, \infty] \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

,

yet recently swish activation (Ramachandran, Zoph and Le, 2017) has been a contender to replace it.

Depending on the intent of the output layer (regression or classification), further normalisation is applied to constrain. For multi-class classification, the most commonly used normalisation is *softmax* (Equation 2.4), of which all the elements from the output sum to 1.

$$\sigma(\mathbf{y})_i = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}} \qquad \text{for } i = 1, \ldots, K \text{ and } \mathbf{y} = (y_1, \ldots, y_K) \in \mathbb{R}^K. \tag{2.4}$$

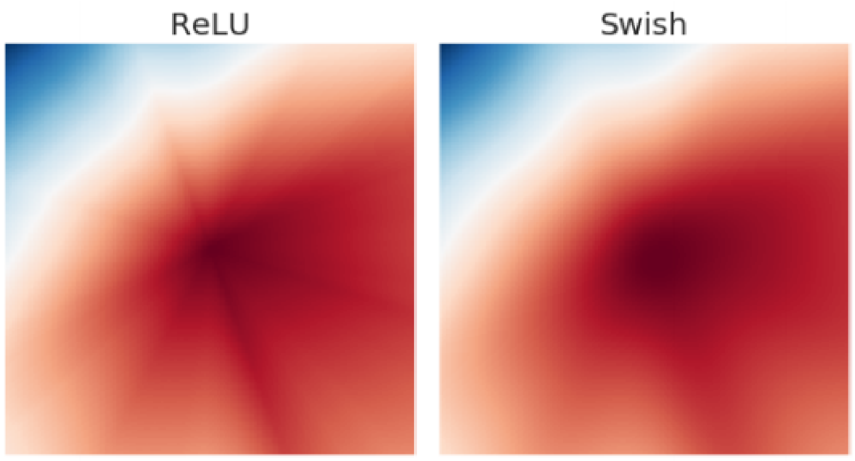


Figure 2.2: An illustration of the smoothing benefits of the swish activation versus relu (Ramachandran, Zoph and Le, 2017).

Training

Before any training is performed on a dataset, we split the data into training, validation and test sets. The model is trained on the training set, and validated on the validation set at various intervals to monitor the performance of the model. Finally, the test set is used to perform a final evaluation to test the model on unseen data.

Supervised learning is the process where features (data) and labels (target variable) are supplied to a machine learning model. In the case of a neural network, upon seeing the training data, it seeks to minimise the loss from the errors between the training set and the validation set.

Loss

A loss function is generally a test metric between the true value of the label y, compared with the predicted value $\hat{y}$ of the label. Commonly used loss functions for regression models for example use the mean squared error

$$\mathbb{L}(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y - \hat{y}^2). \tag{2.5}$$

For classification tasks, categorical cross entropy is used

$$\begin{aligned}\mathbb{L}(y, i) &= -\log(\sigma(\boldsymbol{y})_i) \\ &= -\log\left(\frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}\right).\end{aligned} \tag{2.6}$$

Optimisation

If we consider each of the weights and biases of the model as a dimension in an n-dimensional space we can use numerical methods to find a combination of these which minimises the loss and allows the model to converge to a point $\boldsymbol{\theta} \in \mathbb{R}^n$ which meets the initial intent.

Gradient Descent is the foundational approach used for this, and is of the form

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \epsilon \nabla_{\boldsymbol{\theta}_k} \mathbb{L}(f(\boldsymbol{x}, \boldsymbol{\theta}_k), \boldsymbol{y}), \tag{2.7}$$

where the loss function $\mathbb{L}$ is to be minimised, $\boldsymbol{x}$ represents the features and $\boldsymbol{y}$ represents the labels, $\boldsymbol{\theta}_k$ are the weights and biases at epoch k, and ϵ is the *learning rate*. The learning rate ϵ is a *hyper-parameter*, which means something that the user has to optimise and is not automatically learned or optimised by the model. The gradient of the function space $\nabla_{\boldsymbol{\theta}} F(\boldsymbol{\theta})$ is calculated via back propagation of the errors algorithm introduced by Rumelhart, Hinton and Williams (1986).

Stochastic Gradient Descent, the theory of which was founded by Robbins and Monro (1952) descends along the function in the parameter space taking note of the negative of its gradient. An additional parameter of *momentum* allows further adjustment to the learning rate ϵ.

AdaGrad (Duchi, 2011) scales the learning rates of each parameter dimension individually ($\boldsymbol{\zeta} \in \mathbb{R}^n$)

$$\zeta_i = \frac{\epsilon}{\delta + \sqrt{r_i}}, \tag{2.8}$$

where δ is a stabilising constant included for numerical stability, and r_i is the cumulative square of the gradients for the parameter θ_i. This accelerates the learning rate ϵ for shallow gradients, and decelerates for steep gradients.

Adam (Kingma and Ba, 2015) adopts the policy of exponential decay rates. A decaying gradient history is used to estimate the first and second order momentum terms which is then used to update the parameters (weights and biases).

Regularisation

Over-fitting is when a model minimises the training loss to such extent that it effectively memorises the training data, leading to a poorly performing model on unseen data. *Under-fitting* is the converse of this. We aim to have a model which is *generalized* to have similar performance between the training and test sets.

Commonly used methods of combat this issue in neural networks are batch normalisation, dropout, early stopping and data augmentation.

Batch Normalisation (Ioffe and Szegedy, 2015) is a transformation usually applied after the activation function, though there appears to be no consensus on whether it should be applied before or after. The transformation is of the form

$$\boldsymbol{x}_i = \gamma \hat{\boldsymbol{x}}_i + \beta \tag{2.9}$$

where γ and β are learnable parameters which normalise the batched input $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ to $\boldsymbol{x}_i$. This improves the generalisation capabilities of the model. Future inputs are normalised in the same manner, and creates a situation where we are more likely to produce feature maps that the model is able to classify correctly.

Dropout (Srivastava and Hinton, 2014) layers are usually added at the end of a *Convolutional Neural Network* through each block of fully connected layers. This transformation randomly selects the outputs of its current iteration or epoch, and sets them to 0. The probability with which this transformation is applied can be set, and by training with less information we reduce the co-dependence of model outputs on individual features.

Early Stopping is where we take subset of the training set and withhold it for validation, and throughout the training process the model performance can be tracked and validated each epoch. We can then choose to stop the training early based upon whether the validation step sees improvement.

Data Augmentation (Shorten and Khoshgoftaar, 2019) is a process by which the input is perturbed and transformed such that each epoch trains on slightly different data. This further reduces the models ability to memorise the training set, reducing over-fitting.

Transfer Learning

A model previously trained can be reused as a starting point for a new model, even if it was trained for a different task. This transfer of knowledge from a pre-trained model can speed up training times for the new model, and give better generalised accuracy.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks are widely adopted in computer vision tasks, such as images or videos. The inspiration for CNNs was perhaps conceived by Wiesel (1968), where they researched the striate cortex of monkeys. Goodfellow, Bengio and Courville (2016) state in Chapter 9.10 that

> “Their great discovery was that neurons in the early visual system responded most strongly to very specific pattern of light, such as precisely oriented bars, but responded hardly at all to other patterns.”

LeCun and Boser (1989) extended this to computer vision, which aimed at classifying hand written zip codes, now known as the MNIST dataset.

CNNs are chosen over ANNs for image tasks because of reduced computational complexity. Convolution generally precedes Batch Normalisation and ReLU, often followed by a Pooling function. The operation of convolution is performed via

$$\begin{aligned} s(t) &= \int p(a)k(t-a)da \\ &= (p * k)(t) \end{aligned} \tag{2.10}$$

where $*$ denotes convolution, input p, output s, and k as the kernel.

Equation 2.10 in discrete form becomes

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{2.11}$$

The feature map is generated by scanning the input map across the kernel.

Hartmann, Schirrmeister and Ball (2018) suggests that CNNs might help to better understand the compositional structure of EEG time series data. A joint model is also shown by Dupre La Tour et al. (2021) to increase prediction accuracy, which leads to finer mappings from CNN layers to the visual cortex.

2.3.3 Random Forests

Random Forests (Ho, 1995) are an ensemble learning algorithm, which constructs many *Decision Trees* at training time. The classification version selects the output class most commonly returned by each tree, and regression takes the mean prediction of all individual trees.

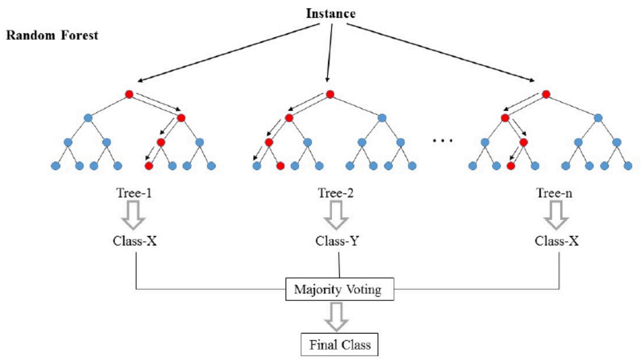


Figure 2.3: A random forest where bootstrapped samples are voted into a final class.

The technique for random forests uses *bootstrap aggregating* (bagging). For a dataset $X = x_0, \ldots, x_n$ with output $Y = y_0, \ldots, y_n$ bagging B times we select a random sample with replacement of the training set and fit decision trees to the sample. The procedure is then:

- For $b = 1, \ldots, B$:
  - Sample, with replacement, n training examples for X_b, Y_b.
  - Train a tree f_b on X_b, Y_b

Predictions can then be made on the test set by averaging the predictions from all individual regression trees.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x') \tag{2.12}$$

Bootstrapping in this way leads to better model performance as it decreases the variance of the model.

2.3.4 Support Vector Machines

Support Vector Machines (Boser, Guyon and Vapnik, 1992) construct a maximised hyperplane which can be used to classify data into separate classes. For linear SVMs, if we are given a dataset of n points consisting of $(\boldsymbol{x}_0, y_0), \ldots, (\boldsymbol{x}_n, y_n)$ where the $y \in (-1, 1)$ we can write the hyperplane as a set of points $\boldsymbol{x}$

$$\boldsymbol{w}^T \boldsymbol{x} - b = 0, \tag{2.13}$$

where $\boldsymbol{w}$ is the normal vector to the hyperplane.

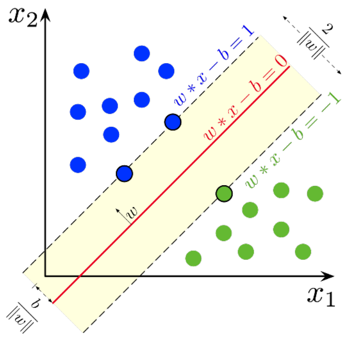


Figure 2.4: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

2.4 Probabilistic Machine Learning

As stated previously, we define probabilistic (or stochastic) machine learning to be the converse of deterministic models, in that when a model receives a specific input, the output is generated by a probability distribution and thus will be different upon each iteration of input and output. Outside of Neural Networks, there are many stochastic approaches available such as Gaussian Processes and various sampling methods available in packages such as `bambi` (Capretto et al., 2020).

Bayesian approaches such as these in the field of neuroscience are thus far quite sporadic (Bielza and Larrañaga, 2014). Song, Kolar and Xing (2009) measured the interactions between regions of the brain and recorded the responses to visual stimuli, who imagined moving a body part based on visual cues. His approach was to use a time-varying dynamic bayesian network.

2.4.1 Bayesian Neural Networks

Research into *Bayesian Neural Networks* dates back to the early 90s (Yamins et al., 2014). This variant of neural networks can be treated probabilistically by maximising likelihood rather than minimising the error.

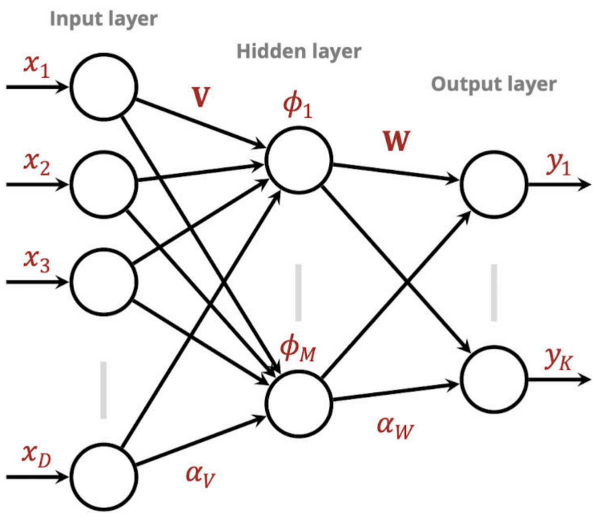


Figure 2.5: A Bayesian neural network with two layers of weights. The input has $\mathcal{D}$ dimensions, M hidden units and K outputs. The hyperparameters α_v and α_w correspond to the input hidden weights $\boldsymbol{V}$ and output weights $\boldsymbol{w}$.

Likelihood

Typically there is a corresponding pairing of error function and likelihood between deterministic and probabilistic models:

- Regression: Squared error $\iff$ Gaussian likelihood with linear link function (activation).
- Classification: Cross-entropy $\iff$ Bernoulli likelihood with sigmoid activation.

In general to switch between the two approaches the error $\equiv$ negative-log-likelihood. To specify the priors of the parameters and compute the posterior distribution we follow

$$p(\boldsymbol{w}, \boldsymbol{V}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w}, \boldsymbol{V})p(\boldsymbol{w}, \boldsymbol{V})}{p(\mathcal{D})} \tag{2.14}$$

Weight Priors

The priors over each layer should be independent

$$p(\boldsymbol{w}, \boldsymbol{V}) = p(\boldsymbol{w})p(\boldsymbol{V}). \tag{2.15}$$

Usually, the priors will be Gaussian with hyperparameterisation in terms of alpha α.

$$\text{Output layer: } p(\boldsymbol{w}|\alpha_w) = \left(\frac{\alpha_w}{2\pi}\right)^{\frac{M}{2}} \prod_{m=1}^{M} \exp\left\{-\frac{\alpha_w}{2}\boldsymbol{w}_m^2\right\} \tag{2.16}$$

$$\text{Hidden layer: } p(\boldsymbol{V}|\alpha_v) = \left(\frac{\alpha_v}{2\pi}\right)^{\frac{MD}{2}} \prod_{d=1}^{D}\prod_{m=1}^{M} \exp\left\{-\frac{\alpha_v}{2}v_{md}^2\right\} \tag{2.17}$$

Automatic Relevance Determination

Automatic Relevance Determination (MacKay, 1994) is exclusive to the Bayesian framework, with no non-Bayesian equivalent. I chose to highlight this approach as it allows us to determine which inputs are informative for the purposes of making predictions. It is implemented by making a specific choice of prior: one which assigns an individual hyperparameter to the group of weights emerging from each input x_d.

In contrast to Equation 2.17, the ARD prior has D separate hyperparameters, one associated with each input

$$p(\boldsymbol{V}|\alpha_{v1}, \ldots, \alpha_{vD}) = \prod_{d=1}^{D}\left(\frac{\alpha_{vd}}{2\pi}\right)^{\frac{M}{2}} \prod_{m=1}^{M} \exp\left\{-\frac{\alpha_{vd}}{2}v_{md}^2\right\}. \tag{2.18}$$

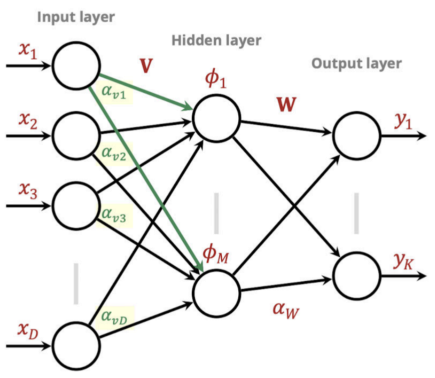


Figure 2.6: A Bayesian neural network with two layers with an ARD prior. There are D individual hyperparameters $\alpha_{v1}, \ldots, \alpha_{vD}$ each corresponding to an input x_d. The output layer is the same as before.

2.5 Classification Metrics

In a binary classification model, there are two classes to classify. Predictions made from a model with this metric will either be true or false, along with positive and negative. These are generally denoted as *TP, FP, TN, FN*. For clarification, below is the detailed definition of each predication.

- True Positive: correctly classifies the positive class.
- True Negative: correctly classifies the negative class.
- False Positive: incorrectly classifies the positive class.
- False Negative: incorrectly classifies the negative class.

Confusion Matrix

A confusion matrix in the best case scenario lists all values within the trace, containing all the *TP* and *TN*. This idea can be extended to a multi-class problem, but can be more difficult to interpret.

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{2.19}$$

Metrics

Accuracy is a metric which illustrates how close or far a set of measurements are to their true value. *Precision* tells you how dispersed those measurements are. *Recall* tells you the dispersion of the measurements to their false value. The following metrics in multi-class scenarios are calculated in the same way as one vs rest. The F_1 score is the harmonic mean of the precision and recall.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.20}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.21}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.22}$$

$$\text{F}_{Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.23}$$

2.6 Further Reading/Considerations

Bishop (2006) is an excellent resource for both supervised and unsupervised machine learning. For further insight into Bayesian approaches, Mackay (2003) provides his entire taught course notes in textbook form. Section 41 in particular provides the necessary background for all the above proposed inference methods. A *Variational Auto Encoder* (Kingma and Welling, 2013) may prove to be a useful stochastic approach, originally conceived for unlabelled data, but has seen success in supervised learning as well. Bowles (2020) gives a fantastic review of Neural Networks if more detail is needed in this area, specifically *attention* gating layers.

A review on EEG analysis and the pipeline methods used can be found in Reaves et al. (2021). He found that the most commonly used algorithms used were *Neural Networks, Support Vector Machines* and *Random Forests*. Most notably, he states that *Transfer Learning* is predicted to be an important analysis method for future research.

Cavedon-Taylor (2021) gives insight on the existence of cognitive penetration and the conceptual link between imagination and perception. Dijkstra, Bosch and van Gerven (2017) suggests that "The more the neural response during imagery is similar to the neural response during perception, the more vivid or perception-like the imagery experience is" (Figure 2.7). Dreams have also been hypothesised to be imaginative states; Siclari et al. (2017) used EEG recordings to report on user dream states upon awakening, shown in Figure 2.8.

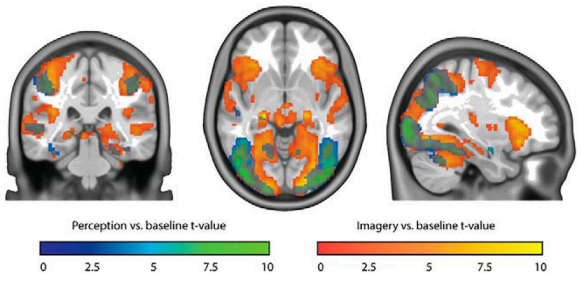


Figure 2.7: Perception and imagery versus baseline. Blue-green represents t values for perception versus baseline. Red-yellow represents t values for imagery versus baseline (Dijkstra, Bosch and van Gerven, 2017).

Figure 2.8: Sagittal MRI scans of the human brain can be used to detect changes in brain activity to offer clues as to what a person is dreaming (Siclari et al., 2017).

Chapter 3

Requirements

Project Plan

To meet this projects objectives an outline is proposed in Appendix A.1. The highest priority lies in the construction of the pipeline to process and decode the EEG data at hand. I expect this stage, which requires experimentation, to take up the majority of the time. Most notably the pre-processing stage to remove artifacts is very much trial and error. Due to the Bath dataset being an untested set, I will also aim to establish a baseline on well benchmarked data during the design phase.

Data Collection

Our dataset aims to answer the following questions:

- Is there an overall variance depending on how vividly an individual imagines?
- Which neural signals (regions and timepoints) enable us to classify invariant of their vividness, such as regions related to semantics.

The dataset is to be collected at the University of Bath, involving 20+ participants neural data, collected from 128 EEG channels using ANT-Neuro hardware and a 10:20 montage[1]. Auditory and visual imagery vividness will be collected separately.

Python Packages

There have been many python packages released recently which have made prototyping and analysis in machine learning accessible. Some of the ones I intend to use are:

- `Pandas` (Wes McKinney, 2010) provides fast, flexible, and expressive data structures designed to make working with tabular data more intuitive.
- `Numpy` (Harris et al., 2020) is a core scientific package in python, useful for efficient matrix and vector operations.
- `Matplotlib` (Hunter, 2007) is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

[1]The 10:20 montage refers to the 10% or 20% inter-electrode distance (Fig 1.2)

- `Seaborn` (Waskom, 2021) is a Python data visualization library based on `matplotlib`. It provides a high-level interface for drawing attractive and informative statistical graphics.
- `Sklearn` (Pedregosa et al., 2011) provides an accessible way to prototype machine learning algorithms and a high level API to quickly provide analysis.
- `Pytorch` (Paszke et al., 2019) is primarily made for deep learning tasks, capable of accelerating tensor operations on cuda cores.
- `Pytorch Lightning` (Falcon et al., 2019) is an extension to `Pytorch` which allows for a more streamlined approach to prototyping neural networks.
- `Tensorflow` (Abadi et al., 2015) is a deep learning library similar to `Pytorch`.
- `Bambi` (Capretto et al., 2020) is a high-level Bayesian model-building interface written in Python. It works with the probabilistic programming frameworks PyMC3 and is designed to make it extremely easy to fit Bayesian mixed-effects models common in biology, social sciences and other disciplines.
- `Python MNE` (Gramfort et al., 2013a; Li et al., 2022) is a tool for use in neuroscience analysis with a high level sklearn like interface.
- `EEGLab` (Delorme and Makeig, 2004) is an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis.
- `Autoreject` (Jas et al., 2016) is is a library to automatically reject bad trials and repair bad sensors in magneto/electroencephalography (M/EEG) data.
- `Pyriemann` (Barachant and King, 2015) is a machine learning library based on scikit-learn API. It provides a high-level interface for classification and manipulation of multivariate signal through Riemannian Geometry of covariance matrices.
- `IC-Label` (Pion-Tonachini, Kreutz-Delgado and Makeig, 2019) is an automated procedure to classify IC using a trained neural network on thousands of IC's.

Computing

In the design phase, I will mostly be able to prototype and implement the models on my home computing devices. In the testing stage, the Hex Cluster (Bath, 2022) will prove to be a useful tool, in allowing access to more powerful machines to train machine learning models on GPUs.

Reproduction

All of the code required to perform the experiments described in later chapters are located here.

Chapter 4

Design

4.1 Datasets

Imagination and Perception

The imagination and perception dataset was created at the University of Bath, and collected by Holly Wilson, Jinha Yoon, Becky Dakin and Aneekha Bal. They used a 128 channel gel-based ANT-Neuro headset, which recorded at 1024 Hz. The amount of trials undertaken by participants varied as a function of their fatigue. Many participants requested to finish before completing the full experiment. Additionally, though all participants were requested to complete three sessions, the majority of participants completed 1-2 sessions. A baseline recording was taken during the 10 second countdown to the first trial, after the participant presses a space bar to begin the experiment. Breaks occurred roughly every 7 minutes. The three semantic categories used are *Penguin*, *Guitar* and *Flower*. The three sensory modalities include pictorial (visual), orthographic (visual) and speech (audio). The two tasks include perception and imagination, of which there are roughly 40 trials for each of the 18 conditions.

It is suggested that imagination and perception tasks are manifested in the α band (Xie, Kaiser and Cichy, 2020), thus a safe band to investigate for decoding in the cleaning stage is $7 - 35$Hz.

AudViz

For testing purposes I plan to assess model performance using the sample dataset *AudViz* from mne. I chose this dataset as the events consist of auditory and visual stimuli, which possibly contain similar eeg signals related to our imagination and perception dataset. Performing analysis with this dataset, and creating plots to compare through various stages will allow a basis of comparison to understand failures and successes with our dataset. Figure 4.2 shows the extracted epochs, illustrating what we possibly expect to see at this stage.

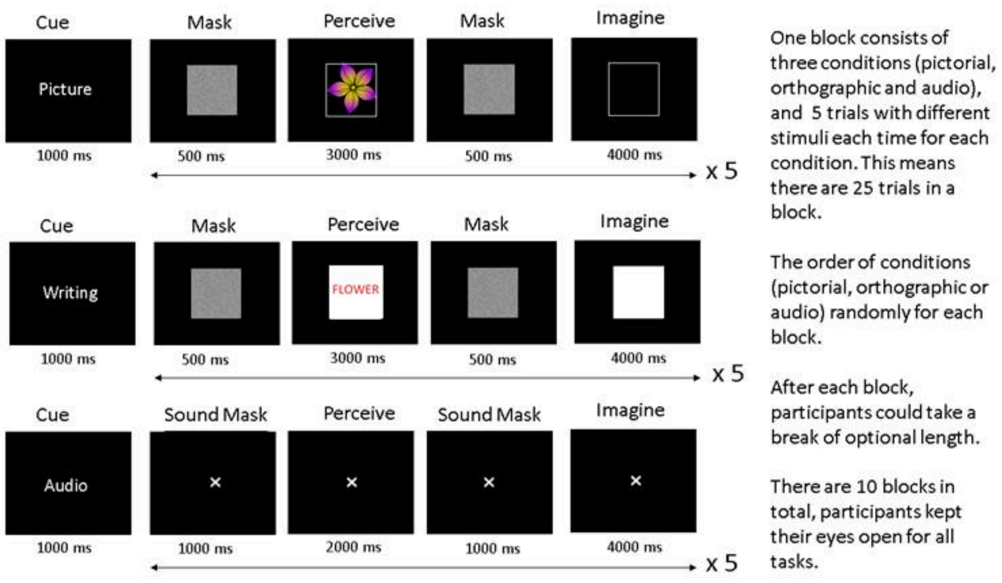

Cue
Mask
Perceive
Mask
Imagine
Picture
1000 ms
500 ms
3000 ms
500 ms
4000 ms
x 5
Cue
Mask
Perceive
Mask
Imagine
Writing
FLOWER
1000 ms
500 ms
3000 ms
500 ms
4000 ms
x 5
Cue
Sound Mask
Perceive
Sound Mask
Imagine
Audio
1000 ms
1000 ms
2000 ms
1000 ms
4000 ms
x 5
One block consists of three conditions (pictorial, orthographic and audio), and 5 trials with different stimuli each time for each condition. This means there are 25 trials in a block.
The order of conditions (pictorial, orthographic or audio) randomly for each block.
After each block, participants could take a break of optional length.
There are 10 blocks in total, participants kept their eyes open for all tasks.

Figure 4.1: The timings of the different event modalities for the imagination and perception dataset.

4.2 Exploratory Data Analysis

Plotting the data in its raw form is the first step of any analysis, to ensure it is what we expect before proceeding. This also helps us to visually inspect for artifacts. A *Power Spectral Density* plot is shown in Figure 4.3 which can be interpreted as a histogram for frequency information. We can see that harmonic power line noise occurs roughly every 50Hz, which will need to be filtered out. The process for this this is further discussed in Section 4.3.

4.2.1 Montage

To visualise the layout of the electrodes, we can apply the montage to the raw data. The ANT-Neuro montage in our dataset has compatibility issues with `Python MNE`, yet we have found a temporary workaround which will likely be improved later on. Currently the rough montage mapping is demonstrated in Figure 4.4. The montage is in a 10:20 format and the reference is CPz.

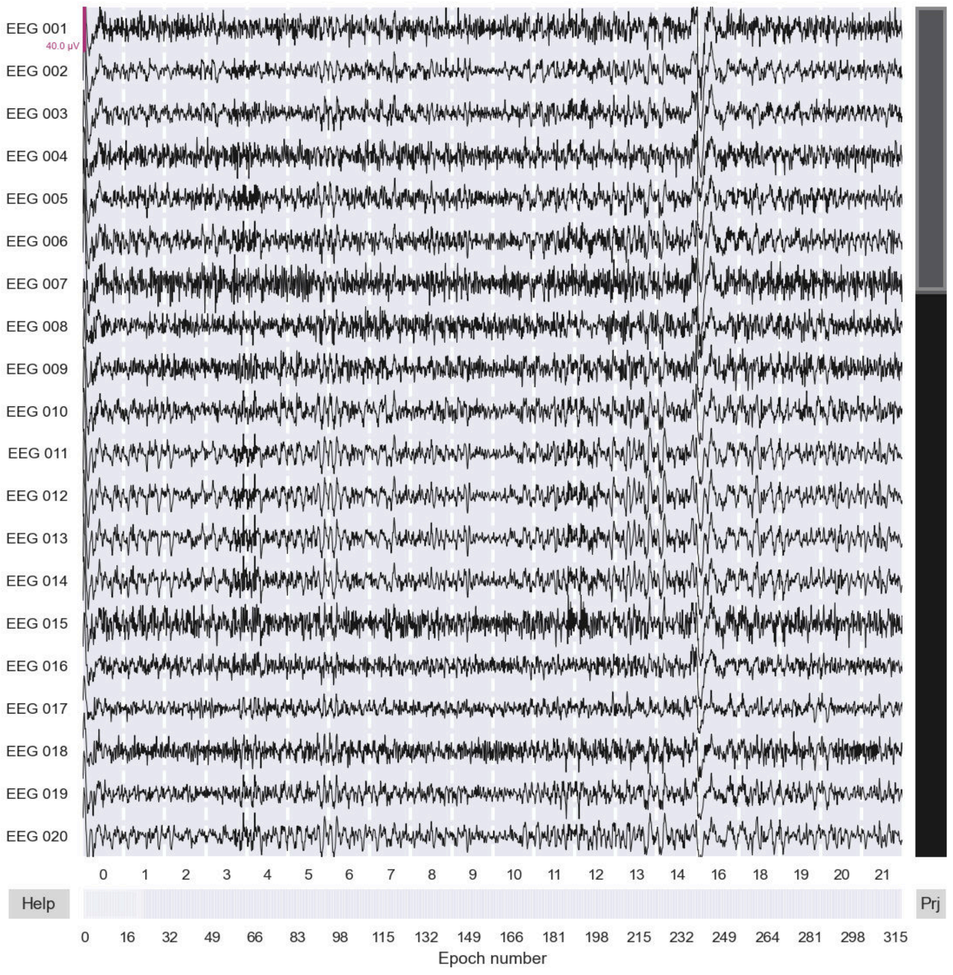


Figure 4.2: The extracted epochs from the AudViz dataset.

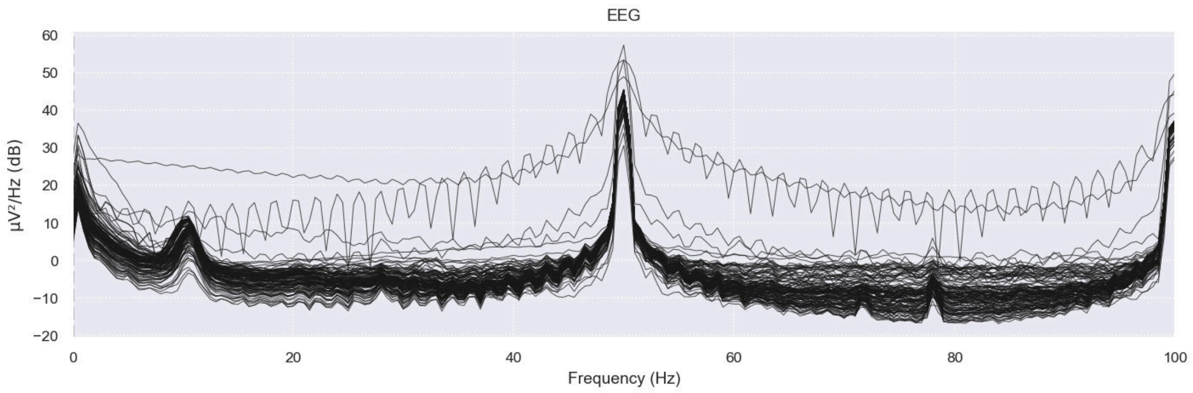


Figure 4.3: The power spectral density is similar to a histogram for frequency information. The spike $\approx 50 - 60$Hz is likely due to AC powerline artifacts. Negative values on the dB scale suggest that the power is less than the reference power. Higher values indicate a more powerful signal.

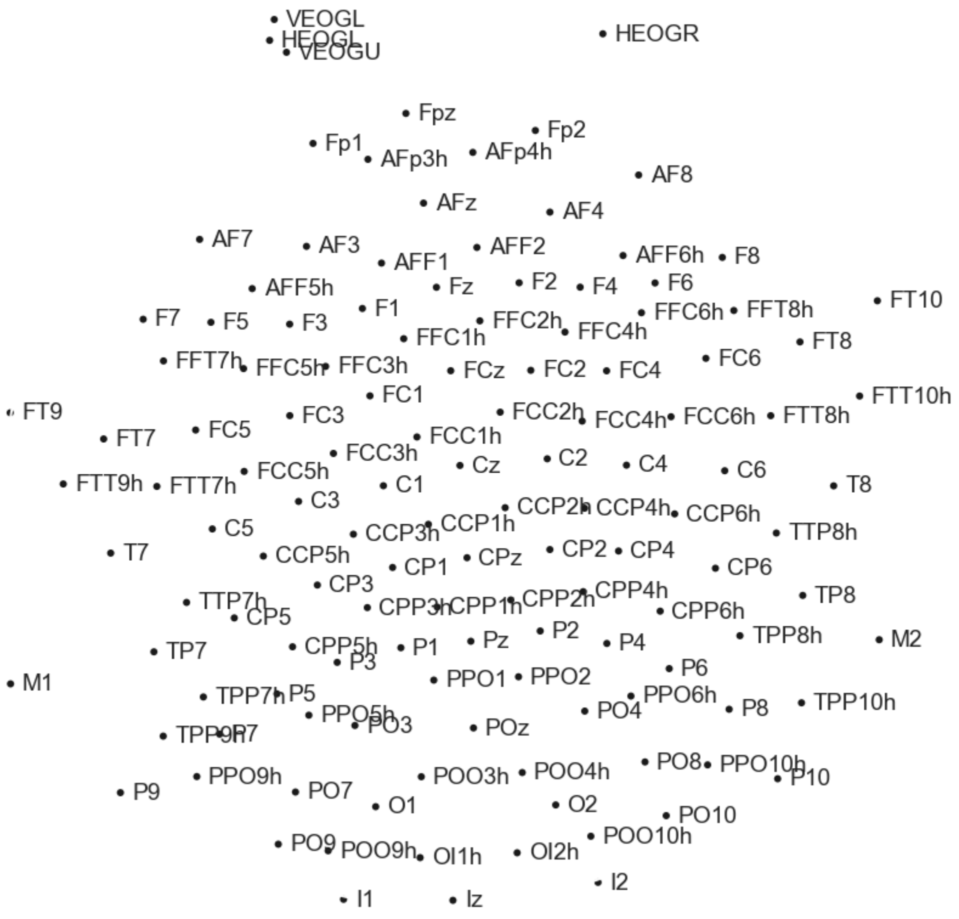


Figure 4.4: The approximate ANT-Neuro, 10:20 montage mapping.

4.3 Pre-Processing

The most important stage in the pipeline is the preprocessing stage. We wish to remove as many unwanted artifacts as possible to leave us with a clean signal to make the best classification possible. Often, the first step is to downsample the data to make the following computations faster, via `raw.resample(128)`, which downsamples the data to 128Hz.

4.3.1 Artifact Detection, Removal and Repair

Artifacts are unwanted persistent signals which are associated with noise, which detract from the signal of interest. This interference can be caused by:

- **Environment**
  - Persistent oscillations centered around the AC power line, which have a frequency range of $\approx 50 - 60$Hz.
  - Inconsistent signal jumps due to environmental vibration, like footsteps or doors closing.
  - Electromagnetic field noise from cell phones, earths magnetic field.
- **Instrumentation**
  - Electromagnetic interference from other electronic equipment such as unshielded headphones.
  - Sensor malfunctions like poor scalp contact which cause random high-amplitude fluctuations, or a constant zero signal.
- **Biological**
  - Periodic signals from electrical activity of the heart.
  - Short and large step-like deflections caused by eye movements and blinking.
  - Bursts of high frequency fluctuations across multiple channels from muscular activity.

Options for dealing with artifacts in EEG data include:

- Ignoring the artifacts entirely.
- Excluding the corrupted parts.
- Repairing the artifacts by suppressing the corrupted parts while maintaining the signal of interest.

Artifact repair methods are quite varied, some of which include digital filtering, independent component analysis and signal-space separation/projection. Generally this kind of reconstruction occurs before epoching, though it can also be performed after this step. Makoto suggests that there are benefits to channel repair after epoching, due to the discarding of irrelevant data in this step.

If channel signals appear completely flat, it could be that the electrode was broken, or the impedance strength was insufficient to record a signal. Missing data can be filled in with interpolation. Variance in signal noise between channels is also problematic, and these can either be removed or have *Independent Component Analysis* performed to repair them.

Our large cap has a broken channel (CCP1h), thus this will always be omitted in analysis. The artifacts driven by eye muscles are visualised in Figure 4.5. We select a few channels close to the eyes (FP1 and FP2) to then estimate eog activity. With the knowledge of their structure, we can then choose to remove these later in the ICA analysis.

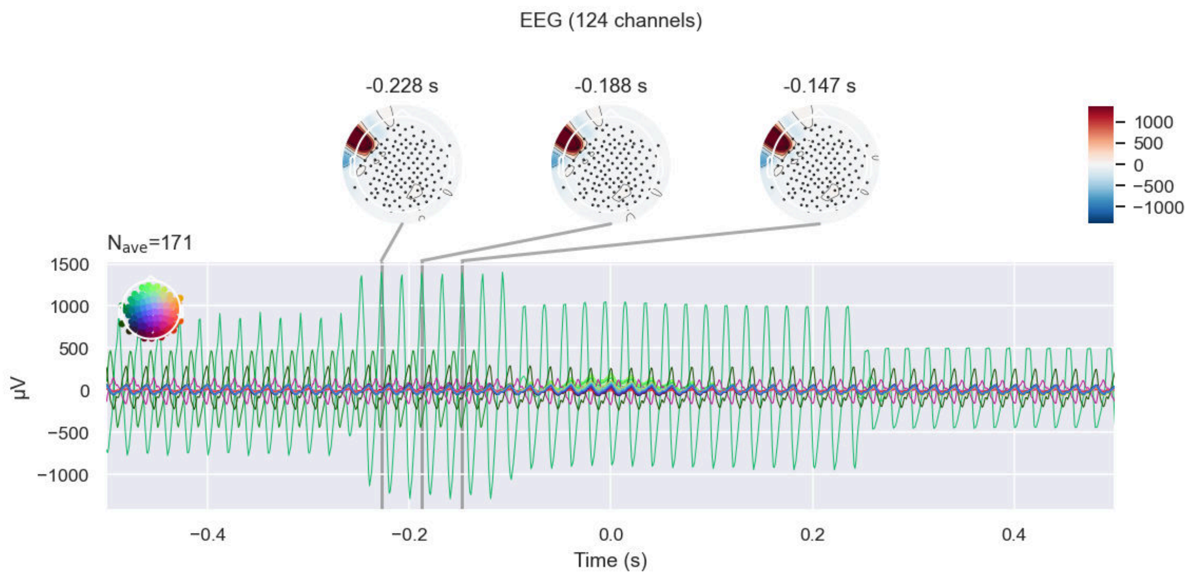


Figure 4.5: The visual artifacts driven by eye muscles.

To remove the power-line noise ($\approx 50-60$Hz) we can apply *Notch Filtering* to the raw object `raw.notch_filter([50,100,150], picks='eeg', phase='zero-double')`, which removes the low-frequency drifts. Such noise is possible to be present at harmonic frequencies, thus we also remove $50, 100, 150$Hz. Movements of the head, wires and scalp perspiration are all contributing factors to this type of artifact. High frequency noise is related to face and neck muscles, as well as electromagnetic interference. After the removal, for participant 8, session 2, the *Power Spectrum Density* is shown in Figure 4.6, complete with montage color mapping.

Low-pass and high-pass filtering allow noise above and below a specified frequency to remain in the data. The command in `MNE` for this is `raw.filter(1, 100, method='iir')`. for example.

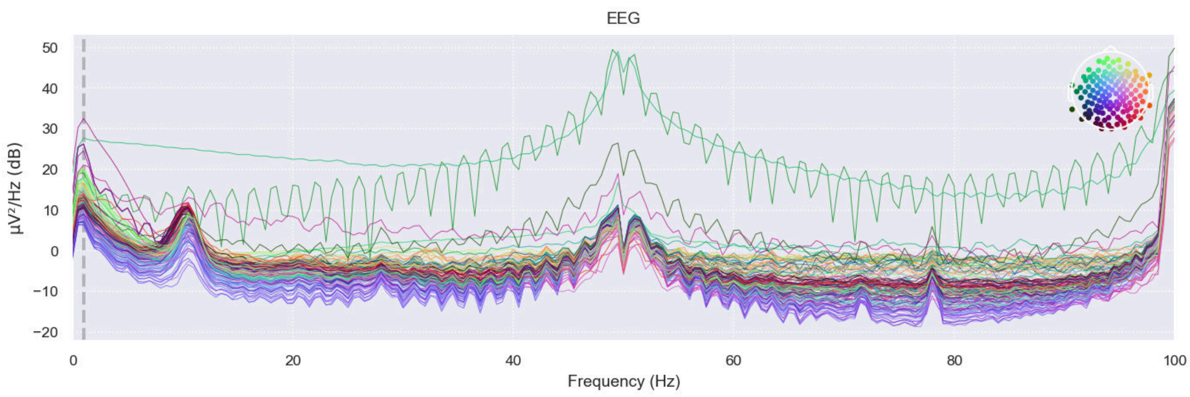


Figure 4.6: The Power Spectrum Density with the notch filter applied.

Another step used is *Independent Component Analysis*, which unmixes separate channels of data. Such a procedure can be performed on filtered, raw or epoched data. ICA is quite sensitive to low-frequency drifts, thus a high pass filter should be applied before fitting. On epoched data, the baseline should not be corrected before using ICA. The *Picard* (Ablin, Cardoso and Gramfort, 2018) version is deemed to be faster and more robust for this process.

To deal with noisy channels we can either remove them from analysis, or interpolate them. In general, channels that should be rejected are noisy throughout the temporal range. If a channel is only noisy periodically, this generally effects all channels simultaneously, and brief periods such as these can be rejected at the *Epoching* step.

Epoching is the process by which the data is epoched based on the trials different stages. This is done to achieve discrete time periods at which point event X was happening and thus the cognitive process Y should reflect this in the data. We wish to narrow the analysis to these time periods in each trial.

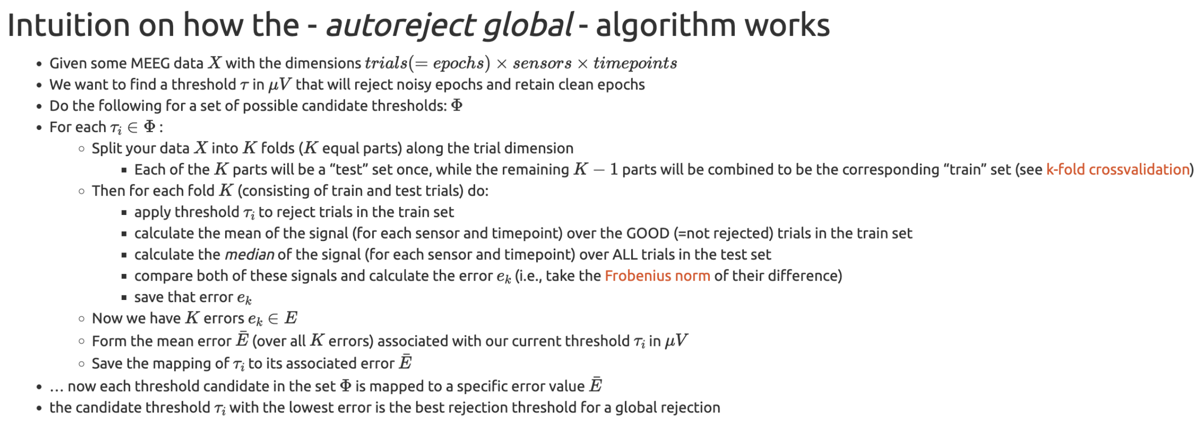
Accurate timings between events and recording software is very important here, and in our study, participant 15, session 1 had synchronisation issues which made the data unusable in analysis.

4.3.2 Automated Pre-Processing

To remove line noise, a notch filter is generally applied in cyclic frequencies of 50Hz, but we can also use a *Spectrum Fit* to automatically detect periodic components. This is achieved via `raw.notch_filter(freqs=None, method='spectrum_fit')`. We can then select a target range for our study, for example the α and β range by filtering with `raw.filter(7, 35)`.

After the data has been *Epoched*, we can apply the python implementation of *Ransac*, `Ransac()`, similar to the one in the PREP Pipeline, and `AutoReject()` (Jas et al., 2016), shown in Figure 4.7. *Ransac* rejects bad channels, and *Autoreject* rejects bad epochs.

The final step is *Independent Component Analysis* followed by *IC-Label* (Pion-Tonachini, Kreutz-Delgado and Makeig, 2019), using the mne-python implementation. `IC-Label` is an automated procedure to classify independent components using a trained neural network on thousands of crowd sourced IC's.



Intuition on how the - *autoreject global* - algorithm works

- Given some MEEG data X with the dimensions $trials(= epochs) \times sensors \times timepoints$
- We want to find a threshold τ in μV that will reject noisy epochs and retain clean epochs
- Do the following for a set of possible candidate thresholds: Φ
- For each $\tau_i \in \Phi$:
    - Split your data X into K folds (K equal parts) along the trial dimension
        - Each of the K parts will be a "test" set once, while the remaining $K-1$ parts will be combined to be the corresponding "train" set (see k-fold crossvalidation)
    - Then for each fold K (consisting of train and test trials) do:
        - apply threshold τ_i to reject trials in the train set
        - calculate the mean of the signal (for each sensor and timepoint) over the GOOD (=not rejected) trials in the train set
        - calculate the *median* of the signal (for each sensor and timepoint) over ALL trials in the test set
        - compare both of these signals and calculate the error e_k (i.e., take the Frobenius norm of their difference)
        - save that error e_k
    - Now we have K errors $e_k \in E$
    - Form the mean error $\bar{E}$ (over all K errors) associated with our current threshold τ_i in μV
    - Save the mapping of τ_i to its associated error $\bar{E}$
- ... now each threshold candidate in the set Φ is mapped to a specific error value $\bar{E}$
- the candidate threshold τ_i with the lowest error is the best rejection threshold for a global rejection

Figure 4.7: The automated algorithm used to automatically reject bad epochs (Jas et al., 2016).

The steps suggested follow the procedure demonstrated in Appendix C.

4.3.3 Feature Extraction

This stage typically involves supervised techniques which aim to extract patterns based upon the training data, using the features x and labels y. It is important to ensure that only the training set is fit, such as a `StandardScaler()`, by which we then use the mean and standard deviation to transform both the training and test sets with the same information. If we fit and transform the entire dataset this would result in data leakage, as information about the test set would contaminate the training set.

Examples of feature extraction methods for EEG pipelines include:

- *xDAWN* (Rivet* et al., 2009): A supervised spatial filtering method which maximises the signal to noise ratio of event-related potentials, and is generally effective in classification studies.
- *Riemannian Geometry* (Barachant et al., 2012): A technique to map covariance matrices to tangent space which converts euclidean vectors while conserving the inner structure of the manifold.

4.4 Machine Learning

4.4.1 Deterministic Classifiers

Neural Network

I based my Neural Network, shown in Appendix C, on *EEGNet* (Lawhern et al., 2016), which is a compact convolutional net designed to be used with EEG signals. A modification to this approach is suggested by Huang et al. (2020), called *S-EEGNet*, which adds bilinear interpolation to improve classification accuracy further. Another variant of the *EEGNet* architecture is *TSGL-EEGNet* (Deng et al., 2021) which aims to improve upon *EEGNet* by using *Temporary Constrained Sparse Group Lasso* (TCSGL) to enhance its performance.

Other Models

From my review, I found that the most popular non network models used were *Random Forests, Support Vector Machines* and *Logistic Regression*. As I adapted my neural network to use a wrapper for use with the `Sklearn` pipeline (Appendix C), I can test these approaches in a very similar fashion. The pipeline for these follow an *xDAWN* denoising and a projection into *Riemann* tangent space.

```
pipe = Pipeline([
    ('xDawnCov', XdawnCovariances(nfilter=2)),
    ('TangentSpace', TangentSpace()),
])
```

Proof of Concept

As a proof of concept I used the sample dataset *AudViz* from mne using the following code.

```
from mne.datasets import sample
import mne

# Read epochs
```

Name	Contents
LA	Response to left-ear auditory stimulus
RA	Response to right-ear auditory stimulus
LV	Response to left visual field stimulus
RV	Response to right visual field stimulus

Table 4.1: The event labels for the *AudViz Dataset*.

```
data_path = sample.data_path()

# Set parameters and read data
raw_fname = data_path + '/MEG/sample/sample_audvis_filt-0-40_raw.fif'
event_fname = data_path + '/MEG/sample/sample_audvis_filt-0-40_raw-eve.fif'
tmin, tmax = -0., 1
event_id = dict(aud_l=1, aud_r=2, vis_l=3, vis_r=4)

# Setup for reading the raw data
raw = mne.io.Raw(raw_fname, preload=True, verbose=False)
# Replace baselining with high-pass
raw.filter(2, None, method='iir')
events = mne.read_events(event_fname)

# Set bad channels
raw.info['bads'] = ['MEG 2443']
picks = mne.pick_types(
    raw.info, meg=False, eeg=True, stim=False, eog=False, exclude='bads'
)
# Read epochs
epochs = mne.Epochs(
    raw, events, event_id, tmin, tmax, proj=False,
    picks=picks, baseline=None, preload=True, verbose=False
)
```

The `mne` documentation for *AudViz* states:

> "In this experiment, checkerboard patterns were presented to the subject into the left and right visual field, interspersed by tones to the left or right ear."

Event labels for this sample are shown in Table 4.1 and the results generated will act as a baseline of comparison for the future analysis with our dataset, and to ensure the models are functioning as required. The models used were *Random Forest, Support Vector Machine, Logistic Regression, Gaussian Process, Linear Discriminant Analysis, EEGNet* and *TSGL-EEGNet*. The non network classifier confusion matrices are shown in Figure 4.8, and the network versions are displayed in Figure 4.9.

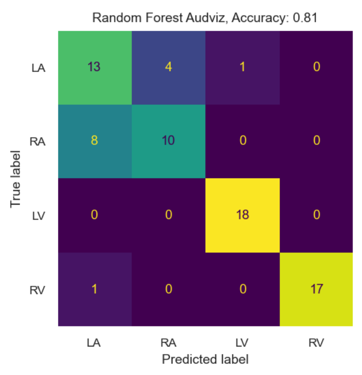
To assess model performance, 100 independent trials were taken to assess the accuracy, all of which underwent 3 cross-fold validation. The results are displayed in Table 4.2. The best results, on average, were observed to be using *Logistic Regression*. Performance for the standard models could be improved further by adjusting the hyperparameters, as these were just left at the default settings in `sklearn`. The neural network used the following hyperparameters

```
pipe = Pipeline([
    ('xDawn', Xdawn(nfilter=4)),
    ('scale', Scaler(scalings='mean')),
    ('reshape', Reshape())
    ])
param_grid = {
    'clf__epochs':[250],
    'clf__batch_size':[16],
    'clf__kern_length':[32],
    'clf__F1':[8],
    'clf__F2':[16],
    'clf__D':[2],
    }
```

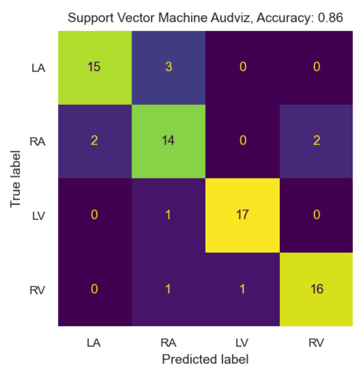
which includes the *xDAWN* feature extraction pipeline. There were trials where the neural network had perfect accuracy, which suggests a possible overfitting to a single participant. Regularization methods such as data augmentation or increasing dropout could help here. The overall difficulty in classification for this particular dataset is fairly easy for a machine learning model to decode due to the lateralization of the neural response for the visual and auditory tasks.

Model	Mean	Min	Max
Random Forest	0.833	0.708	0.931
Support Vector Machine	0.891	0.806	0.972
Logistic Regression	0.897	0.819	0.958
Gaussian Process	0.876	0.792	0.958
Stochastic Gradient Descent	0.870	0.778	0.972
Linear Discriminant Analysis	0.800	0.667	0.889
EEGNet	0.960	0.889	1.000
TSGL-EEGNet	0.969	0.847	1.000

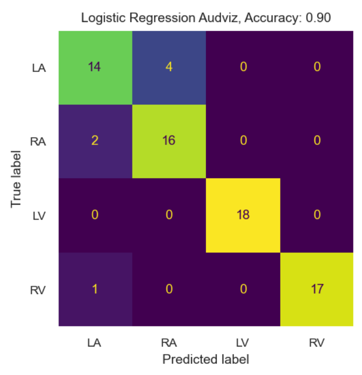
Table 4.2: The model prototype accuracy tested on the *AudViz Dataset*, with 100 trials and 3 fold cross validation at the training stage, evaluated on a hold out test set. The feature extraction involves the *xDAWN* and *Riemann Geometry* pipeline. Without this stage, average accuracy was $\approx 63\%$. The mean, minimum and maximum accuracy is shown for each model, suggesting that the best performing standard model on average for this particular eeg dataset is *Logistic Regression*. The neural networks show improved average performance, however the computation time for 100 trials takes $\approx$ 2 hours to perform, compared to $\approx$ 15 minutes for the others.



(a) *Random Forest*

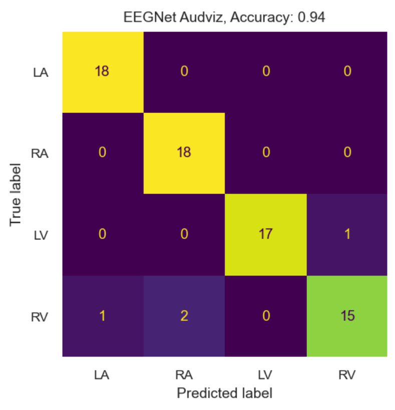


(b) *Support Vector Machine*

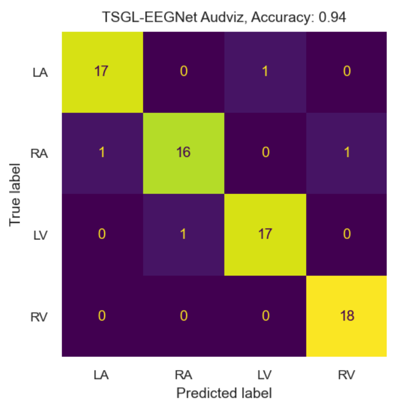


(c) *Logistic Regression*

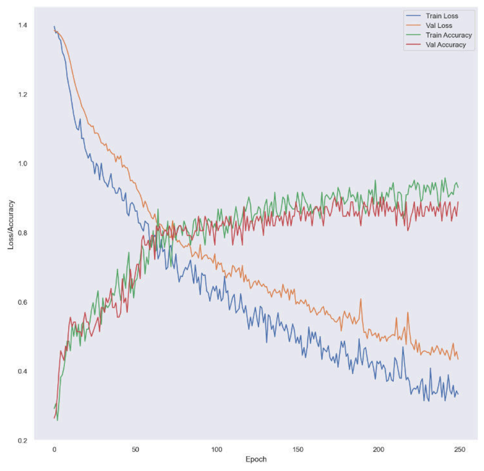
Figure 4.8: The confusion matrices for *Random Forest*, *Support Vector Machine*, and *Logistic Regression*, applied to the `mne` sample dataset *AudViz*, pre-processed with the *xDAWN* and *Riemann Geometry* pipeline.



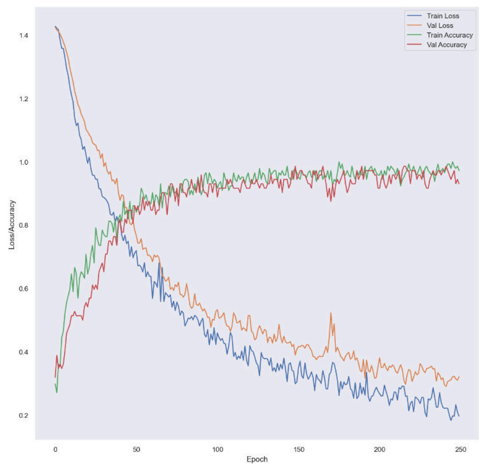
(a) *EEGNet*



(b) *TSGL-EEGNet*



(c) History for *EEGNet*

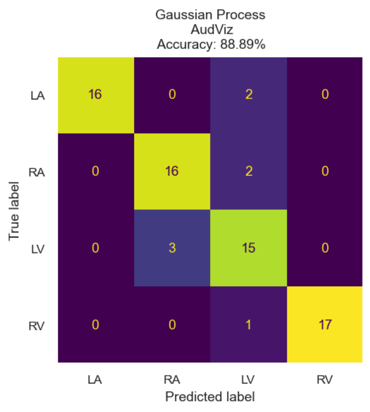


(d) History for *TSGL-EEGNet*

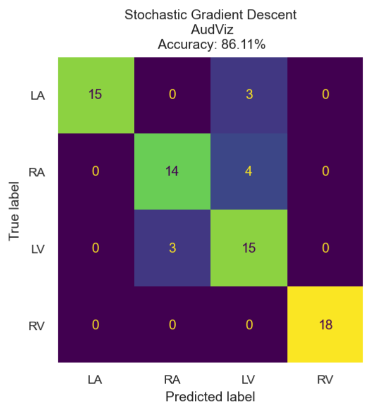
Figure 4.9: The confusion matrices and history for *EEGNet* and *TSGL-EEGNet*, applied to the `mne` sample dataset *AudViz*. Subplot 5.4e shows an improved convergence, thus this does indicate the modifications perfom as expected.

4.4.2 Stochastic Classifiers

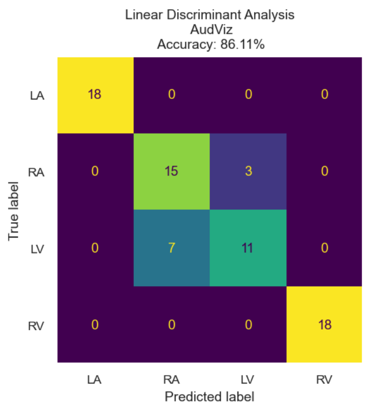
Implementations of *Gaussian Process*, *Stochastic Gradient Descent* and *Linear Discriminant Analysis* from `sklearn` offer a stochastic approach of classification.



(a) *Gaussian Process Analysis*



(b) *Stochastic Gradient Descent*



(c) *Linear Discriminant Analysis*

Figure 4.10: The confusion matrices for *Gaussian Process*, *Stochastic Gradient Descent* and *Linear Discriminant Analysis*, applied to the `mne` sample dataset *AudViz*, with the feature extraction of the *xDAWN* and *Riemann Geometry* pipeline.

Chapter 5

Implementation and Testing

The pipeline consists of three core steps:

- Pre-Processing (Section 5.1)
- Feature Extraction (Section 5.2)
- Decoding (Section 5.3)

The pre-processing stage aims to clean the raw data such that we can preserve the signal of interest relating to the event related potentials. Feature extraction aims to boost the cleaned data through methods like *Common Spatial Patterns* or *xDAWN*, alongside transformations like projecting into *Riemann* space. Decoding takes this transformed data and seeks to minimise a loss function, calculated by comparing the predictions versus the true labels. Decoding in this way is generally referred to as *Machine Learning*.

The preprocessing stage was performed in both `eeglab` and `mne`. The interactivity using the graphical user interface in `eeglab` allows for quick inspection of each cleaning step. The same can be achieved in `mne`, and steps for both are outlined in Section 5.1.

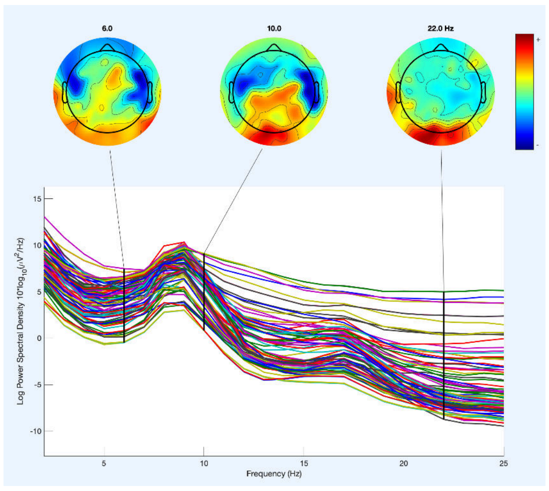
The code for the feature extraction and decoding stage is shown in Appendix C.

5.1 Pre-Processing

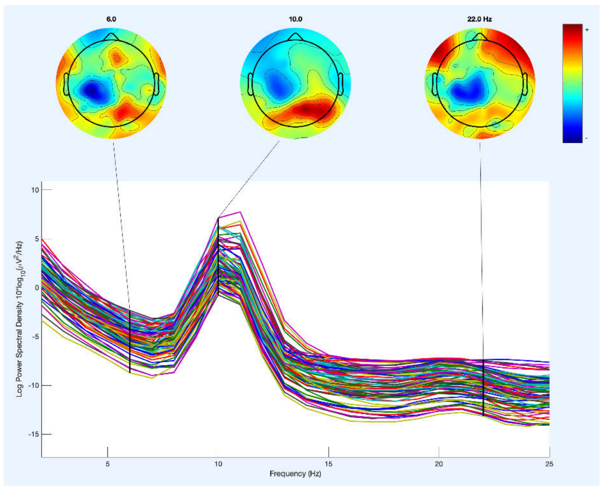
5.1.1 Using `eeglab`

Makoto's preprocessing pipeline provides a generalised insight into best practices for cleaning EEG data. Before feature extraction, I performed these steps in `eeglab`:

- Downsample to 128Hz.
- Filter to a band of $1 - 40$Hz.
- Run *Automatic Signal Rejection.*
- Re-reference to the average.
- Inspect the spectra plots shown in Figure 5.1 and drop channels to follow the cleaned distribution shown in Figure 5.1b.
- Inspect the *Event Related Potential* plots.
- ICA/ICA-Label (Figure 5.3), reject components like channel noise and eye blinks.
- Epoching, reject bad epochs on inspection.



(a) Noisy channel spectra.



(b) Cleaned channel spectra.

Figure 5.1: Channel spectra and topographical maps. Each coloured line represents a single channels spectral activity.

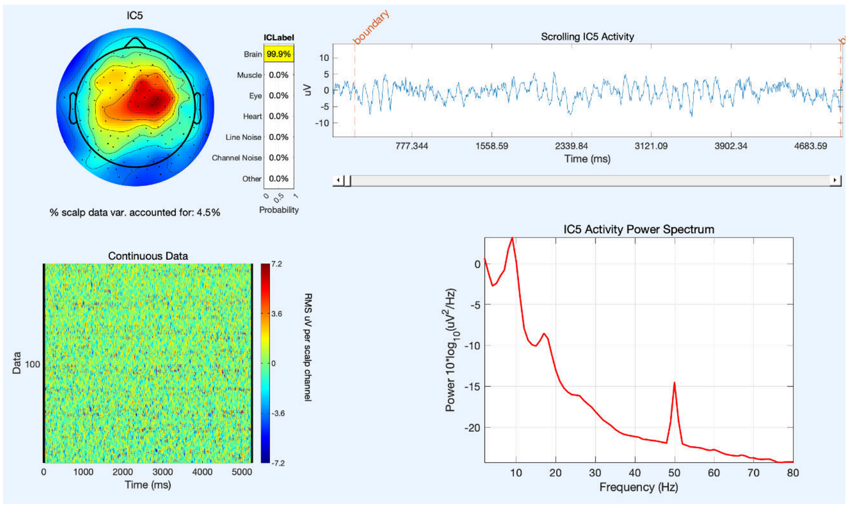


Figure 5.2: ICA component 5, suggesting a high probability of brain activity from ICA-Label.

5.1.2 Using `mne`

The goal for the `mne` pipeline was to achieve a level of automation in pre-processing. The following steps were taken during the artifact removal process.

- Bandpass filter in the α and β range using `raw.filter(7, 35)`.
- Epoching, reject bad epochs and channels using *Ransac* and *Autoreject* and decimate (downsample) to 128Hz.
- Re-reference to the average.
- ICA/ICA-Label (Figure 5.3), reject classified components like channel noise and eye blinks. Selecting components labelled as brain or other give the best results.

The notch filter can be used to clean line noise from the signal if target frequencies under investigation fall within the 50Hz range. It should be noted that the suggested lowpass filter to be applied before decimation is a third of the sample rate to avoid aliasing.

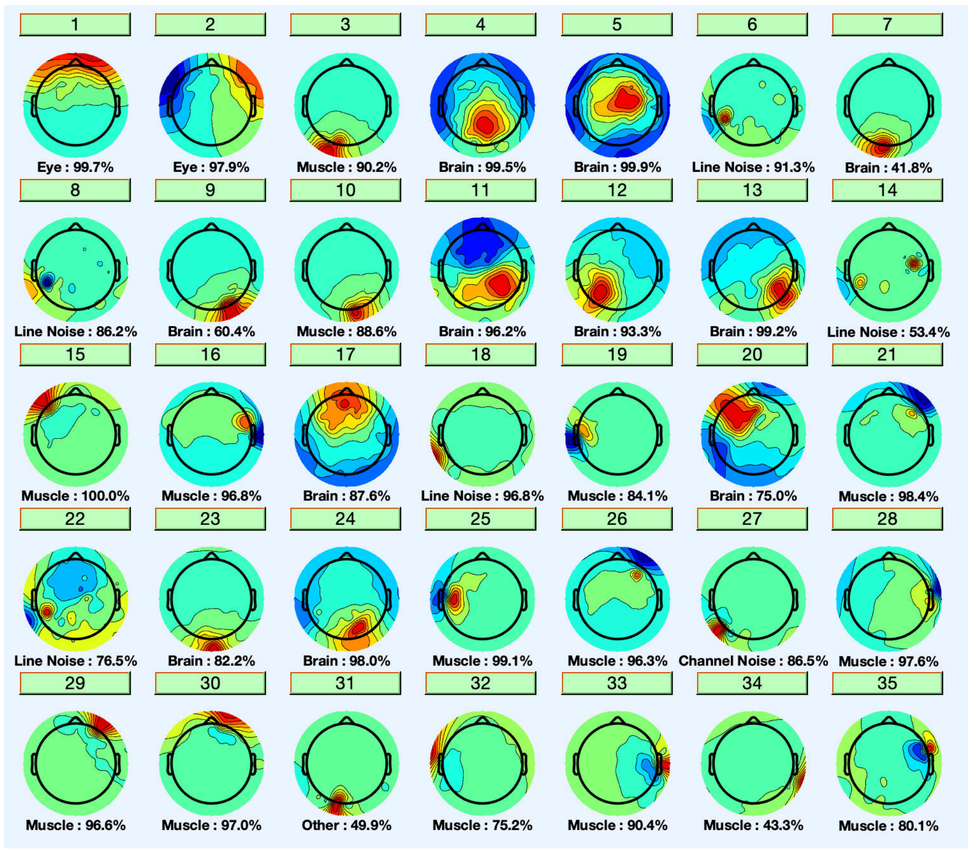


Figure 5.3: The probability distributions suggesting possible signal sources using ICA-Label. We aim to have cleaned data prior to fitting ICA such that we maximise the brain signal components discovered.

```
# Notch Filter
raw.notch_filter(freqs=None, method='spectrum_fit').
```

This method in particular is an automated solution which finds the most significant spikes channel by channel and applies a notched window to each frequency artifact found.

Re-referencing the data to the average with

```
# Re-Reference
raw.set_eeg_reference(ref_channels='average')
```

is best used when the electrode montage covers the whole head. Alternatively, our reference channel `['Cpz']` could be used for this purpose. Upon testing, both gave similar results after *ICA* decomposition.

In `MNE`, epoching is performed via

```
# Epoching
events, event_ids = mne.events_from_annotations(
    raw, verbose = False
)
epochs = mne.Epochs(
    raw=raw,
    events=events,
    event_id=event_ids,
    preload=True,
    tmin=0,
    tmax=4,
    baseline=None,
    event_repeated='merge',
    decim=8
)
```

where the events are the objects perceived or imagined through each modality in our trial. Slicing around these events occurs in the range $t \in [0, 4]$.

Autoreject (Jas et al., 2016) is an automated rejection algorithm designed for use after epoching to infer bad trials and attempt to repair them. A good explanation of the algorithm can be found here.

```
# Autoreject
ar = AutoReject(n_jobs=6)
epochs = ar.fit_transform(epochs)
```

Ransac from the PREP pipeline (Bigdely-Shamlo et al., 2015) is another automated rejection algorithm infer bad channels and attempt to repair/remove them. A good explanation of the algorithm can be found here.

```
# Ransac
rs = Ransac(n_jobs=6)
epochs = rs.fit_transform(epochs)
```

I used the extended version of *Picard* in `mne` after extracting the epochs, followed by the `IC-Label` plugin.

```
# ICA
ica_epoch_filt = epochs.copy().filter(l_freq=1., h_freq=None)
ica = mne.preprocessing.ICA(
   n_components=122, method='picard', fit_params=dict(extended=True)
)
ica.fit(ica_epoch_filt)
# ICA Labels
ic_labels = label_components(raw, ica, method='iclabel')
labels = ic_labels["labels"]
exclude_idx = [idx for idx, label in enumerate(labels)
                if label not in ["brain", "other"]]
print(f"Excluding these ICA components: {exclude_idx}")
# Apply ICA
ica.apply(epochs, exclude=exclude_idx)
```

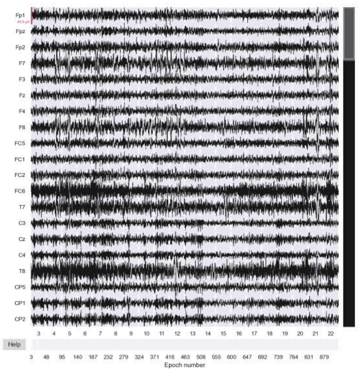
To interactively inspect the epochs object its possible

```
# Blocks code execution until popup plot is closed
%matplotlib qt
imagine_orthographic_epochs.plot(block=True)
```

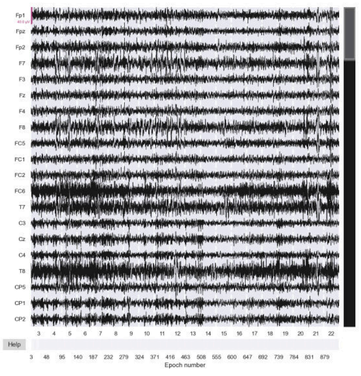
to halt the further execution of code until the plot is closed. Figure 5.4c shows an example of this process, where we can mark channels and epochs for rejection. After selection, and upon closing the plot the parts marked for rejection are applied to the object and displayed in the output

```
Dropped 7 epochs: 4, 20, 22, 25, 32, 40, 44
The following epochs were marked as bad and are dropped:
[22, 136, 142, 168, 194, 250, 286]
```

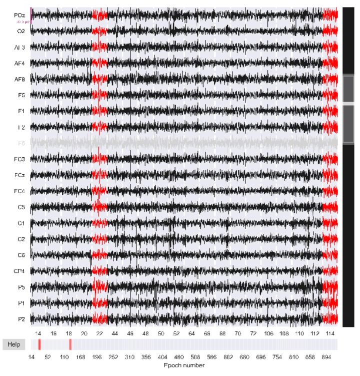
```
Channels marked as bad:
['O1', 'O2', 'P8', 'I1', 'Iz', 'I2', 'POO10h', 'OI1h', 'OI2h'].
```



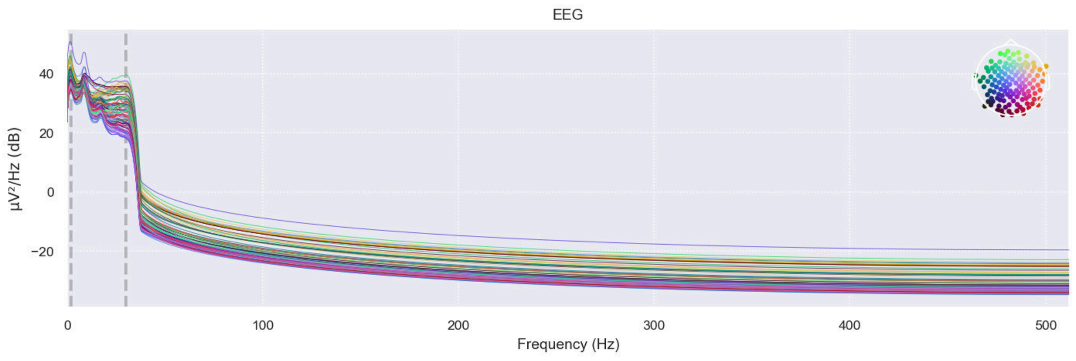
(a) Epochs sampled at 1080Hz.



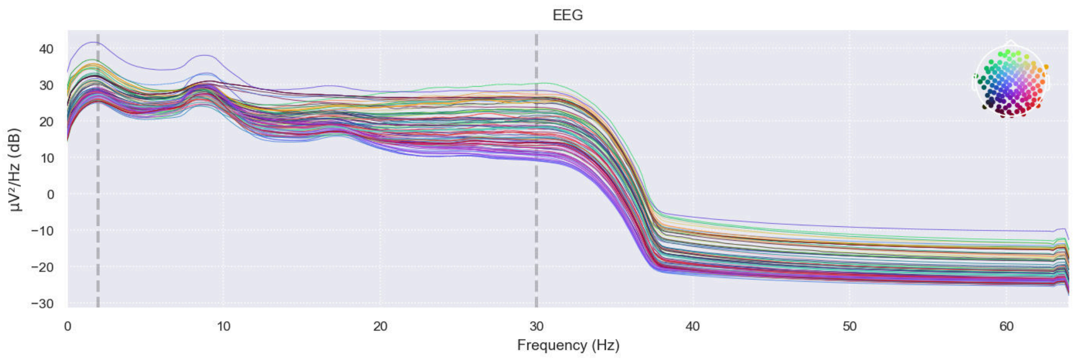
(b) Epochs sampled at 128Hz.



(c) Epoch rejection at 128Hz.



(d) The *Power Spectral Density* for 1080Hz.



(e) The *Power Spectral Density* for 128Hz.

Figure 5.4: Our data was recorded at a sample rate of 1080Hz, and Figure 5.4b shows that we can reduce our samples by 5x and preserve similar spatial patterns. Such downsampling is beneficial for the feature extraction and decoding stage, as there is less data to compute. Figure 5.4c illustrates the interactive epoch and channel rejection process, to manually inspect and clean the data by eye.

5.2 Feature Extraction

The first implementation I considered for feature extraction was the *xDAWN* and *Riemannian Geometry* which projects the Euclidian vectors into tangent space. For the `sklearn` pipeline I used the implemented classes from the `pyriemann` (Barachant and King, 2015) package

```
from pyriemann.estimation import XdawnCovariances
from pyriemann.tangentspace import TangentSpace

pipe = Pipeline([
        ('xDawnCov', XdawnCovariances()),
        ('TangentSpace', TangentSpace()),
        ('Scale', StandardScaler()),
    ]),
```

which ensures the training data only has access to the labels to avoid data leakage. The neural network only implements the *xDAWN* feature extraction with the following hyperparameters

```
from pyriemann.spatialfilters import Xdawn

pipe = Pipeline([
    ('xDawn', Xdawn(nfilter=4)),
    ('scale', Scaler(scalings='mean')),
    ('reshape', Reshape())
    ])
param_grid = {
    'clf__epochs':[500],
    'clf__batch_size':[16],
    'clf__kern_length':[32],
    'clf__F1':[8],
    'clf__F2':[16],
    'clf__D':[2],
    }.
```

5.3 Decoding

The decoding stage allows the use of defining multiple different machine learning models for experimentation. More can be added in the dictionaries defined at the beginning of the pipeline.

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
# Choose model
model = 'Logistic Regression'
standard = {
    'Support Vector Machine':LinearSVC(),
    'Logistic Regression':LogisticRegression(solver='liblinear'),
    'Random Forest':RandomForestClassifier(),
    'Gaussian Process':GaussianProcessClassifier(),
    'Stochastic Gradient Descent':SGDClassifier(),
    'Multi Layer Perceptron':MLPClassifier(),
    'Linear Discriminant Analysis':LinearDiscriminantAnalysis(),
}
nets = {
    'EEGNet':KerasClassifier(eeg_model),
```

```
        'TSGL-EEGNet':KerasClassifier(tsgl_eeg_model)
    }
```

The results of the entire pipeline are shown in Table 5.1, and it appears that the pipeline struggles to classify based upon the observed performance.

Model	Mean	Min	Max
Random Forest	0.332	0.225	0.559
Support Vector Machine	0.330	0.299	0.456
Logistic Regression	0.331	0.228	0.518
Gaussian Process	0.334	0.187	0.523
Stochastic Gradient Descent	0.338	0.234	0.567
Linear Discriminant Analysis	0.334	0.312	0.476
EEGNet	0.330	0.278	0.498
TSGL-EEGNet	0.334	0.298	0.472

Table 5.1: The model accuracy tested on the cleaned *Bath Dataset*, with 100 trials and 3 fold cross validation at the training stage, evaluated on a hold out test set. The feature extraction involves the *xDAWN* and *Riemann Geometry* pipeline. Without using the preprocessing pipeline, the raw data average accuracy was $\approx 33\%$. The mean, minimum and maximum accuracy is shown for each model, which appear to have no statistical difference.

Chapter 6

Results

The set of results listed in Section 6.1 offer a broad analysis of the effects the automated pipeline has on decoding the raw bath dataset versus the cleaned signal.

6.1 Ablation Study

The raw data was processed in its entirety, including all epochs across all modalities, the results for which are in Table 6.1. Tables 6.2 and 6.3 show the results for the same pipeline processed across individual modalities (audio, visual and orthographic) for both imagination and perception tasks. It is expected that the latter will show performance improvement due to the specific tasks associated with each modality occur in different areas of the brain. If we can consistently show decoding accuracy greater than baseline (33%), further investigation can be carried out to tweak the pipeline. Cleaning EEG data is both a scientific and explorative journey, in which trial and error of the order taken for specific steps has to be taken.

Raw Data

The figures for five subjects from the bath dataset (Appendix B) were processed using the cleaning pipeline discussed previously, and also outlined in Appendix C. It is known that at the recording stage, subject 10 had a strong level of participation. It is the assumption then, that we should see stronger signals which correlate to the intended event related potentials to which we are investigating.

At a glance, Figure B.1 shows similarities across the different subjects raw data, which at this stage is more indicative of the artifacts present in the shared environment they were recorded in. Subject 10 appears to have a rogue channel which is an order of magnitude greater than the others. Strong event related potentials seem to occur in all subjects around the 0.3 and 3 second mark, the first event is likely explained by an evoked response. Subject 8 and 14 look the cleanest, yet it was noted during the recording that event timings in subject 8 were corrupted, which compounds timing errors with future processing.

The PSD plots in Figure B.2 illustrate the effects of the artifacts distorting the channels. Here we can see that subject 14 and 17 also have large outlier channels, perhaps explained by a faulty sensor with low scalp contact.

Ransac/Autoreject

Figure B.4 shows the rejection logs after *Random Sample Consensus* for each subject, illustrating which epochs and channels were either interpolated or marked as bad and dropped. Very few bad epochs and channels are detected in participant 14, in agreement with previous analysis suggesting that this trial had less artifacts present.

After *Ransac* and *Autoreject* (Figure B.5), we observe that the signals are much more constrained and share similar event timings between participants. Participant 10 seems to have the least signal to noise ratio in comparison to the other trials.

The PSD plots in Figure B.6 confirm that the variance between channels in participant 14 are well constrained. In comparison to the PSD before ransac (Figure B.2) we can see that the automated cleaning process has had a positive effect in cleaning rogue channels.

The topography maps shown in Figure B.7 gives an idea of how our chosen band of investigation (7-35Hz) will impact the following ICA decomposition. Participant 14 appears to have a smoother transition in its signal to noise, and exhibits much more constraint as previously shown. The others have significantly more variance in this regard.

A quick comparison of the first few channels in Figure B.8 to B.3 shows the impact of the cleaning process across the board. Again, participant 14 can be seen to have consistent signals across the channels. Visual inspection suggests we can see possible patterns but still the signal to noise is quite poor.

ICA

The components classified automatically using IC-Label as brain activity are shown in Figure B.9. We can see that the majority of these do indeed look promising, though it can be argued that a few seem quite spurious. As this is still an open source project which can only improve given more time and training data, I see this as an important step in the right direction for future work and investigation.

Upon applying the chosen ICA components we observe in Figure B.11 the effect of ICA decomposition. Marked in red are the signals before, and black after cleaning. Strong spikes remain in all but participant 17. The spikes occur across all subjects, of which the first is likely an evoked response in input stimuli.

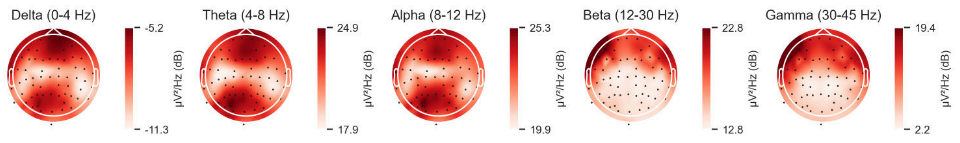
The signal to noise issues present before ICA in Figure B.5 show improvement in Figure B.12 from before, indicating that many of the artifacts showing long term trends have been removed, leaving us primarily with non-stationary data.

Finally, the channel plots in Figure B.13 show the impact of the entire pipeline in comparison to Figure B.3, before any pre-processing. It can be seen that the differences are quite drastic, and much of the artifacts appear to have been removed. Its possible this approach is too aggressive, but the nature of this dataset in particular is a challenge for decoders to see any separation between the classes.

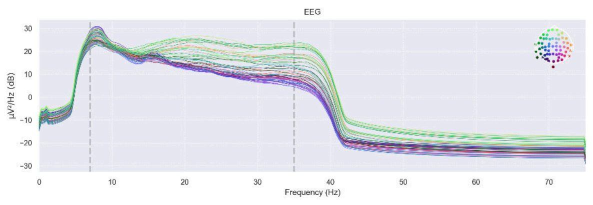
xDawn Features

Following the cleaning pipeline, the signal to noise features are boosted and then projected into tangent space. The effect of this is shown in Figure 6.2. Figure 7.1a explains how the classifiers were able to easily separate the classes for the AudViz dataset in the proof of concept stage. The Bath dataset when using the same pipeline struggles to achieve the same seperation (Figure 7.1c), likely explaining why classification accuracy cannot exceed the baseline in the decoding stage.

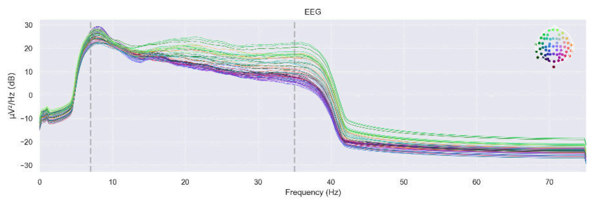
Comparing Bath to AudViz



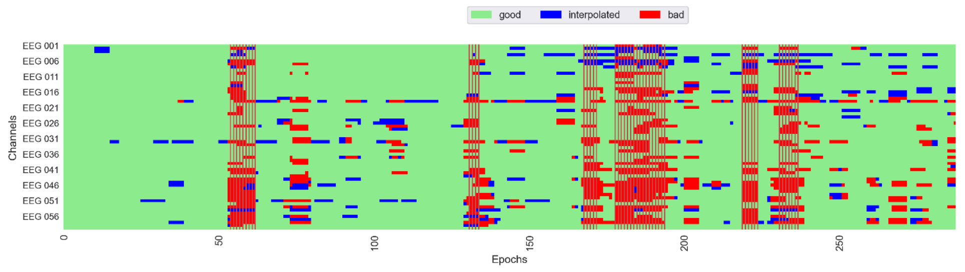
(a) Topographical Map



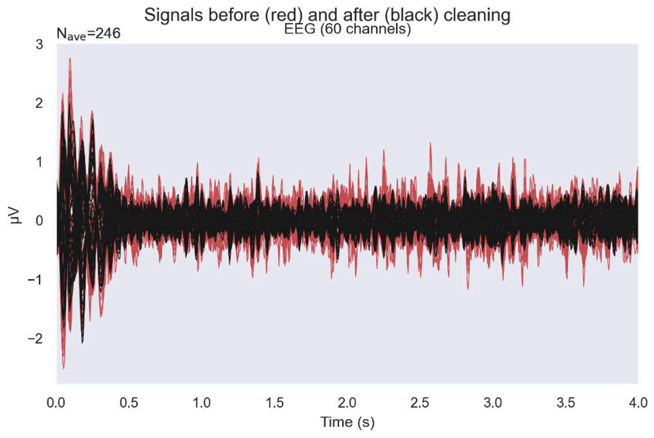
(b) Before Ransac/Autoreject



(c) After Ransac/Autoreject

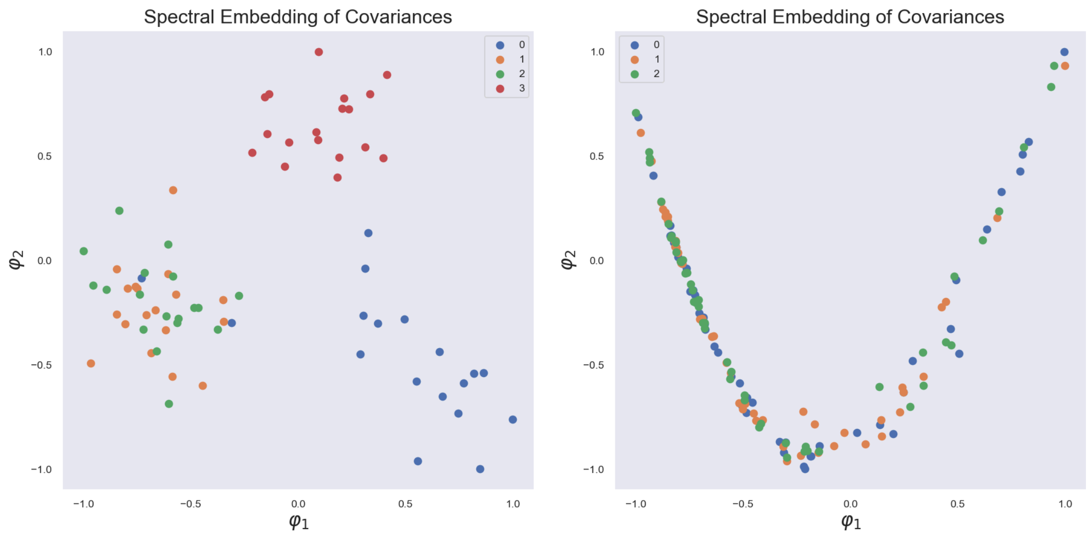


(d) Rejection Table



(e) ICA proposal

Figure 6.1: Comparing to the plots in Appendix B.1 the signals in the Audviz dataset appear to be cleaner with a stronger correlation between less channels, and less outliers. A similar amount of epochs and events are rejected at the Ransac/Autoreject stage, yet the before and after psd plots show that a similar distribution is maintained. The ICA proposal plot suggests many of the artifacts present were carefully handled during the collection stage.



(a) AudViz **(b)** Bath

Figure 6.2: The xDawn covariances show the difficulties facing the baseline classification accuracy for the Bath dataset.

Decoding

Table 6.1 illustrates a comparison between the use of the proposed pipeline on the Bath dataset (Participant 14 Session 2), in its raw and cleaned form.

Data (All Modalities)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.331	0.276	0.435	0.332	0.225	0.559
Support Vector Machine	0.328	0.278	0.429	0.330	0.299	0.456
Logistic Regression	0.329	0.278	0.498	0.331	0.228	0.518
Gaussian Process	0.334	0.192	0.519	0.334	0.187	0.523
Stochastic Gradient Descent	0.337	0.269	0.495	0.338	0.234	0.567
Linear Discriminant Analysis	0.333	0.226	0.504	0.334	0.312	0.476
EEGNet	0.331	0.282	0.470	0.330	0.278	0.498
TSGL-EEGNet	0.339	0.306	0.488	0.334	0.298	0.472

Table 6.1: The model accuracy tested on the *Bath Dataset* including all modalities (both Imagination and Perception), with 100 trials and 3 fold cross validation at the training stage, evaluated on a hold out test set. The feature extraction involves the *xDAWN* and *Riemann Geometry* pipeline. Both raw and cleaned data average accuracy was 33%±1%. The mean, minimum and maximum accuracy is shown for each model. There appears to be no significance to the results, as greater than baseline accuracy was not achieved.

The difficulties in class separation following the feature extraction observed in Figure 6.2 translated to poor performance (Table 6.1) in the decoding stage as predicted. Table 6.2 illustrates the same issues, when inferring across the individual modalities and tasks.

Data (Audio Perception)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.333	0.282	0.454	0.331	0.245	0.543
Support Vector Machine	0.322	0.277	0.435	0.333	0.288	0.464
Logistic Regression	0.328	0.292	0.489	0.332	0.234	0.508
Gaussian Process	0.335	0.255	0.501	0.331	0.199	0.561
Stochastic Gradient Descent	0.331	0.278	0.423	0.338	0.243	0.487
Linear Discriminant Analysis	0.331	0.232	0.501	0.331	0.266	0.480
EEGNet	0.330	0.290	0.412	0.335	0.292	0.489
TSGL-EEGNet	0.340	0.307	0.475	0.329	0.278	0.488

Data (Visual Perception)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.330	0.283	0.432	0.331	0.224	0.555
Support Vector Machine	0.331	0.286	0.499	0.330	0.278	0.431
Logistic Regression	0.328	0.271	0.509	0.329	0.229	0.519
Gaussian Process	0.335	0.272	0.504	0.333	0.239	0.492
Stochastic Gradient Descent	0.336	0.271	0.499	0.331	0.261	0.549
Linear Discriminant Analysis	0.329	0.291	0.488	0.331	0.299	0.466
EEGNet	0.339	0.267	0.490	0.335	0.295	0.485
TSGL-EEGNet	0.335	0.301	0.485	0.336	0.299	0.475

Data (Orthographic Perception)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.336	0.282	0.477	0.331	0.226	0.561
Support Vector Machine	0.329	0.277	0.431	0.333	0.298	0.495
Logistic Regression	0.339	0.243	0.530	0.330	0.225	0.515
Gaussian Process	0.331	0.199	0.520	0.328	0.187	0.525
Stochastic Gradient Descent	0.340	0.271	0.458	0.331	0.266	0.514
Linear Discriminant Analysis	0.331	0.257	0.544	0.331	0.300	0.544
EEGNet	0.333	0.219	0.481	0.335	0.293	0.458
TSGL-EEGNet	0.334	0.307	0.549	0.333	0.286	0.499

Table 6.2: The model accuracy tested on the *Bath Dataset* across individual modalities (Perception), with 100 trials and 3 fold cross validation at the training stage, evaluated on a hold out test set. The feature extraction involves the *xDAWN* and *Riemann Geometry* pipeline. Both raw and cleaned data average accuracy was 33%±1%. The mean, minimum and maximum accuracy is shown for each model. There appears to be no significance to the results, as greater than baseline accuracy was not achieved.

Data (Audio Imagination)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.328	0.291	0.499	0.333	0.266	0.555
Support Vector Machine	0.323	0.245	0.555	0.331	0.289	0.476
Logistic Regression	0.329	0.244	0.476	0.331	0.244	0.516
Gaussian Process	0.331	0.204	0.566	0.333	0.277	0.531
Stochastic Gradient Descent	0.337	0.292	0.478	0.339	0.255	0.499
Linear Discriminant Analysis	0.335	0.256	0.523	0.330	0.275	0.490
EEGNet	0.331	0.286	0.453	0.335	0.293	0.555
TSGL-EEGNet	0.339	0.204	0.563	0.336	0.292	0.491

Data (Visual Imagination)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.329	0.245	0.466	0.335	0.264	0.445
Support Vector Machine	0.333	0.290	0.489	0.331	0.298	0.444
Logistic Regression	0.337	0.260	0.590	0.340	0.236	0.520
Gaussian Process	0.332	0.284	0.510	0.330	0.263	0.433
Stochastic Gradient Descent	0.332	0.260	0.491	0.336	0.270	0.539
Linear Discriminant Analysis	0.333	0.290	0.480	0.332	0.204	0.499
EEGNet	0.331	0.292	0.473	0.336	0.290	0.480
TSGL-EEGNet	0.329	0.300	0.455	0.331	0.240	0.479

Data (Orthographic Imagination)	Raw			Clean		
Model	Mean	Min	Max	Mean	Min	Max
Random Forest	0.329	0.240	0.553	0.337	0.252	0.520
Support Vector Machine	0.325	0.279	0.435	0.335	0.259	0.494
Logistic Regression	0.336	0.247	0.521	0.334	0.253	0.501
Gaussian Process	0.335	0.190	0.494	0.333	0.245	0.566
Stochastic Gradient Descent	0.339	0.257	0.464	0.331	0.270	0.520
Linear Discriminant Analysis	0.333	0.255	0.520	0.338	0.228	0.555
EEGNet	0.341	0.266	0.493	0.338	0.290	0.451
TSGL-EEGNet	0.331	0.290	0.501	0.332	0.295	0.500

Table 6.3: The model accuracy tested on the *Bath Dataset* across individual modalities (Imagination), with 100 trials and 3 fold cross validation at the training stage, evaluated on a hold out test set. The feature extraction involves the *xDAWN* and *Riemann Geometry* pipeline. Both raw and cleaned data average accuracy was 33%±1%. The mean, minimum and maximum accuracy is shown for each model. There appears to be no significance to the results, as greater than baseline accuracy was not achieved.

Chapter 7

Conclusion

Main Objectives

The overall goal of this project was to create an automated pre-processing pipeline to clean EEG data, which we can then use to decode (classify) what a person is thinking about based upon their neural activity. Through this context, a comparison of stochastic and deterministic machine learning model performance was drawn. Makoto has shared many valuable insights into the experimentation stages of pre-processing, and this pipeline was designed to implement all of the best practices suggested. Appendix C.1 contains the code outline for this stage, in which the steps are

- Apply the custom montage.
- Filter the channels between 7-35Hz.
- Epoch the raw data around the events, and decimate (resample) to 128Hz.
- Perform *Ransac* and *Autoreject* to remove artifacts from both entire channels and individual events.
- Run *Independent Component Analysis* using the extended *Picard* algorithm.
- Choose which ICA's to keep using the neural network IC-Label and apply the chosen ICA's.

With this cleaned signal we then perform feature extraction using a projection into tangent space and *xDawn* spatial filtering. This can then be fed into any machine learning algorithm.

The pipeline showed great success with the AudViz dataset, yet struggled with the *Bath* dataset. Despite this, a comparison of machine learning models was able to be drawn on the prior dataset, and the neural networks show great promise in both their training accuracy, and future ability to use transfer learning across participants.

Pre-Processing

A comparison between both AudViz and *Bath* datasets for the entire pipeline was drawn, and performs well with the prior. After inspection across 5 participants (Appendix B.1) in the *Bath* dataset for the pre-processing pipeline, participant 14 session 2 was chosen for the ablation study as this signal showed the most promise. The study was broken down into three investigations, the results of which are shown in Tables 6.1, 6.2 and 6.3. The first was processed with both imagination and perception across all the modalities of audio, visual and orthographic. The second and third subclass imagination and perception individually, along with each modality separately.

Figure 6.1 offers a comparison to the Bath ablation processed in the same pipeline with the AudViz dataset. The topographical map perhaps sheds some insight into the the differences between the two datasets. In AudViz, there appears to be localised clusters for each frequency band, yet in Bath the map appears almost to look like a gaussian kernel, with a radial pattern. This might suggest that the tasks are too general, and the experiment could show improvement at the data collection phase by introducing more lateralised tasks. Repeating the experiment with just the left and right eye for the perception task for example, could allow us to localise the target signals and in comparison to recording both simultaneously.

The distribution of the psd in Figure 6.1c has a more complex pattern, especially toward the 35Hz range, where it seems to diverge. At the 50Hz range, there appears to be no powerline artifacts either. On inspection alone, it is difficult to explain the difference in accuracy between the Bath and AudViz datasets. Perhaps the topographical map is the vital component in assessing future performance of classification, and a radial pattern indicates a lack of features.

Decoding

After 100 independent trials with unique random seeds, each with 5-fold cross validation and stratified to ensure even class distributions between the test and training set, performance greater than the baseline of 3 class classification (33.3%) was not achieved on the latter. The outcome was the same when processing the raw and clean data across all modalities simultaneously (Table 6.1) and separately (Tables 6.2 and 6.3). The nature of classification for imagination and perception is a broader problem than the lateralized classes in the AudViz dataset, which involves tasks separated by the left and right audio and visual responses, which likely explains the difficulty in class separation in the Bath dataset.

When testing models which performed $> 50\%$ in classification accuracy on a further hold out set, performance was not consistent, and either achieved high accuracy due to chance, or possibly overfit to their training sets.

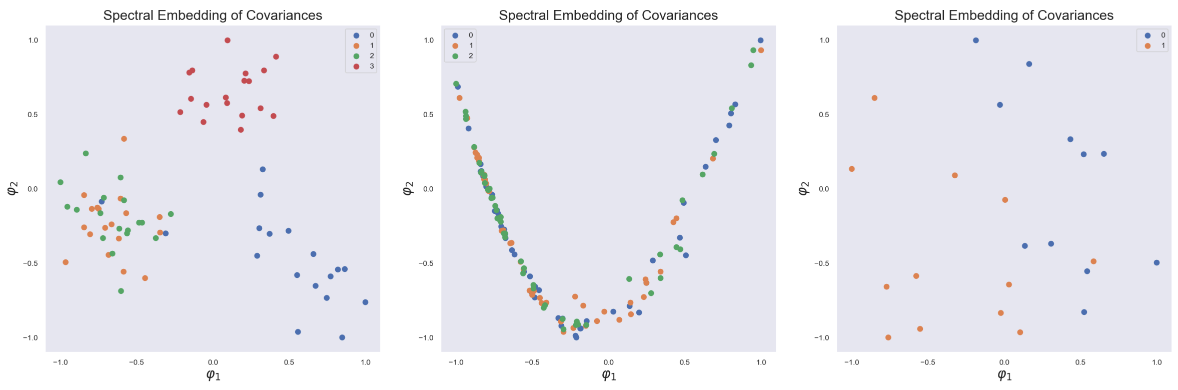
Performing the same analysis on participant 17 session 1 yielded the same outcome as participant 14 session 2, showing the difficulties in classification are present across subjects. Studies have shown (Özbeyaz and Arıca, 2018) that a subjects familiarity with given tasks have improved signal to noise ratio, and as such, classification with respect to familiarity has a degree of lateralization. This could be a valid consideration when collecting data for future participants in the *Bath* dataset, as secondary or third sessions could be investigated to determine if a subjects experience with the task allows for increased resolution in the event potentials.

Cross fold validation was a vital step in reducing the possibility of making assumptions about model accuracy. If the average of 100 runs achieves a higher than baseline accuracy we can be more confident about the stochastic nature during training that deterministic models achieve. Before I implemented this, I struggled to explain the discrepancies between accuracy in different training runs.

Feature Extraction

Figure 6.2 illustrated a possible explanation for the inability to achieve acceptable accuracy in the Bath dataset. The clusters of classes overlap heavily, thus for future experiments it is recommended that these feature plots are investigated before further machine learning classification is attempted. It's possible that through exploring the hyperparameter space at this stage would allow discovery of a set of conditions which does indeed create class separation for the next step of decoding.

Figure 7.1 shows another dataset from python-mne, for motor imagery. This is similar to AudViz and further shows the relative ease that lateralised tasks can be separated by task for classification. Figure 7.3 analyses the minimum distance to the mean for the covariances in both the bath and motor imagery datasets illustrates how few features are extracted at this stage.



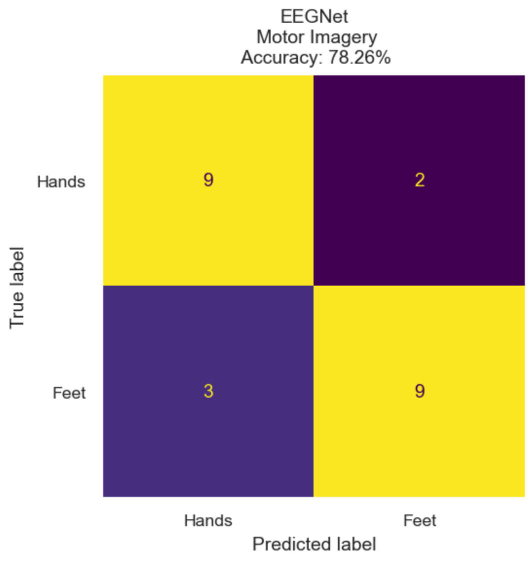
(a) AudViz **(b)** Bath **(c)** Motor Imagery

Figure 7.1: The xDawn Covariances show the difficulties facing the baseline classification accuracy for the Bath dataset. The motor imagery dataset also shows good class separation.

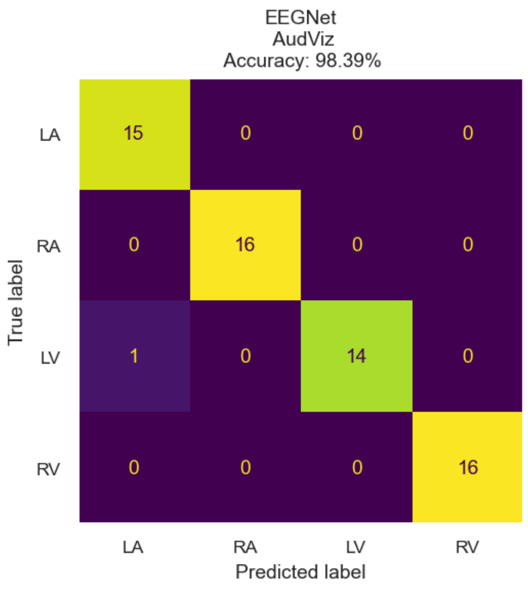
Further Work

During the testing phase with the AudViz dataset, the EEGNet family of neural networks had the best performance, and further developments by (Zhang et al., 2022) could be used in future pipelines. An improvement to my approach would implement the focal loss they describe, which allows the network to be unbiased during training on datasets with imbalanced classes, preventing possible overfitting to the majority class.

Further work could involve using the KARA One dataset in the proposed decoding pipeline to confirm its performance akin to the observed success using AudViz. KARA One (Zhao and Rudzicz, 2015) combines three modalities (EEG, face tracking, and audio) during imagined and vocalized phonemic and single-word prompts. They have shown success in accurately classifying imagined phonological categories solely from EEG data.

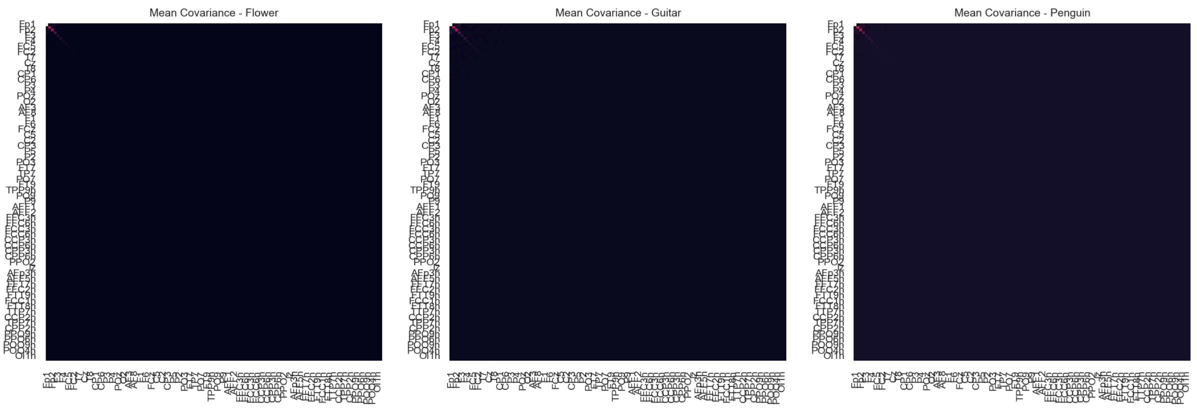


(a) Motor Imagery

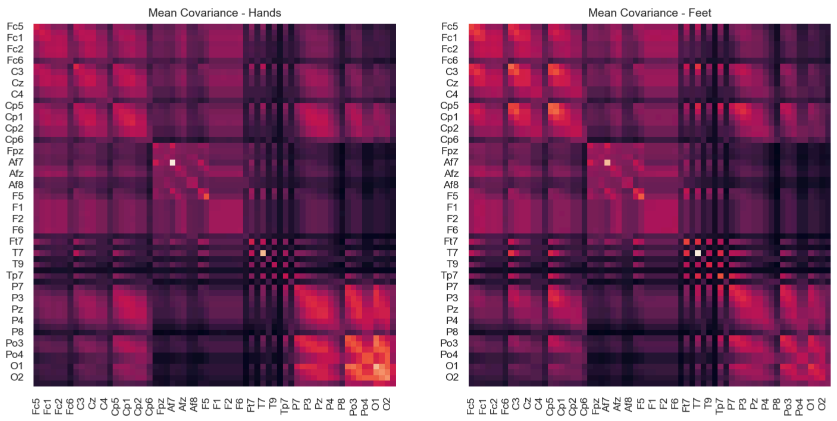


(b) Audviz

Figure 7.2: The confusion matrices for the Motor Imagery and AudViz datasets, processed with the full pipeline.



(a) Bath



(b) Motor Imagery

Figure 7.3: The minimum distance to mean covariances for both bath and motor imagery datasets illustrate the difficulties facing a classifier. The signal appears very flat and featureless.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems [Online]. Software available from tensorflow.org. Available from: https://www.tensorflow.org/.

Ablin, P., Cardoso, J.F. and Gramfort, A., 2018. Faster independent component analysis by preconditioning with hessian approximations. *IEEE transactions on signal processing* [Online], 66(15), aug, pp.4040–4049. Available from: https://doi.org/10.1109/tsp.2018.2844203.

Barachant, A., Bonnet, S., Congedo, M. and Jutten, C., 2012. Multiclass brain–computer interface classification by riemannian geometry. *Ieee transactions on biomedical engineering* [Online], 59(4), pp.920–928. Available from: https://doi.org/10.1109/TBME.2011.2172210.

Barachant, A. and King, J.R., 2015. pyriemann v0.2.2 [Online]. Available from: https://doi.org/10.5281/zenodo.18982.

Bath, U., 2022. Balena HPC cluster - The technical information for the University's High Performance Computing (HPC) systems. [Online]. Available from: https://www.bath.ac.uk/corporate-information/balena-hpc-cluster/.

Bellman, C., Martin, M., MacDonald, S., Alomari, R. and Liscano, R., 2018. Have We Met Before? Using Consumer-Grade Brain-Computer Interfaces to Detect Unaware Facial Recognition. *Computers in entertainment* [Online], 16(2), pp.1–17. Available from: https://doi.org/https://doi.org/10.1145/3180661.

Bielza, C. and Larrañaga, P., 2014. Bayesian networks in neuroscience: a survey. *Frontiers in computational neuroscience* [Online], 8. Available from: https://doi.org/10.3389/fncom.2014.00131.

Bigdely-Shamlo, N., Mullen, T., Kothe, C., Su, K.M. and Robbins, K.A., 2015. The prep pipeline: standardized preprocessing for large-scale eeg analysis. *Frontiers in neuroinformatics* [Online], 9. Available from: https://doi.org/10.3389/fninf.2015.00016.

Bishop, C., 2006. *Pattern Recognition and Machine Learning* [Online]. Springer. Available from: http://users.isr.ist.utl.pt/$\sim$wurmd/Livros/school/Bishop-PatternRecognitionAndMachineLearning-Springer2006.pdf.

Boser, B., Guyon, I. and Vapnik, V., 1992. A training algorithm for optimal margin classifiers. *Colt* [Online], p.144. Available from: https://doi.org/10.1.1.21.3818.

Bowles, M., 2020. Paying Attention to Radio Astronomy Data. *Thesis* [Online]. Available from: https://www.research.manchester.ac.uk/portal/files/188966450/FULL_TEXT.PDF.

Brogaard, B. and Gatzia, D.E., 2017. Unconscious imagination and the mental imagery debate. *Frontiers in psychology* [Online], 8. Available from: https://doi.org/10.3389/fpsyg.2017.00799.

Burton, H., 2003. Visual Cortex Activity in Early and Late Blind People. *J neurosci* [Online], 23(10), pp.4005–4011. Available from: https://doi.org/10.1523/JNEUROSCI.23-10-04005.2003.

Capretto, T., Piho, C., Kumar, R., Westfall, J., Yarkoni, T. and Osvaldo, A., 2020. Bambi: A simple interface for fitting Bayesian linear models in Python [Online]. Available from: `https://github.com/bambinos/bambi`.

Carbon, C., 2014. Understanding human perception by human-made illusions. *Front. hum. neurosci* [Online]. Available from: `https://doi.org/https://doi.org/10.3389/fnhum.2014.00566`.

Cavedon-Taylor, D., 2021. Untying the knot: imagination, perception and their neural substrates. *Synthese* [Online], 199, 12. Available from: `https://doi.org/10.1007/s11229-021-03110-x`.

Collazos-Huertas, D.F., Álvarez-Meza, A.M., Acosta-Medina, C.D., Castaño-Duque, G.A. and Castellanos-Dominguez, G., 2020. CNN-based framework using spatial dropping for enhanced interpretation of neural activity in motor imagery classification. *Brain informatics* [Online], 7(1). Available from: `https://doi.org/10.1186/s40708-020-00110-4`.

Dabas, H., Sethi, C., Dua, C., Dalawat, M. and Sethia, D., 2018. Emotion Classification Using EEG Signals. *Proceedings of the 2018 2nd international conference on computer science and artificial intelligence* [Online], pp.380–384. Available from: `https://doi.org/https://doi.org/10.1145/3297156.3297177`.

Delorme, A. and Makeig, S., 2004. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of neuroscience methods*, 134(1), pp.9–21. `j.jneumeth.2003.10.009.`

Deng, X., Zhang, B., Yu, N., Liu, K. and Sun, K., 2021. Advanced tsgl-eegnet for motor imagery eeg-based brain-computer interfaces. *Ieee access* [Online], 9, pp.25118–25130. Available from: `https://doi.org/10.1109/ACCESS.2021.3056088`.

Dijkstra, N., Bosch, S.E. and Gerven, M.A. van, 2017. Vividness of visual imagery depends on the neural overlap with perception in visual areas. *Journal of neuroscience* [Online], 37(5), pp.1367–1373. `https://www.jneurosci.org/content/37/5/1367.full.pdf`, Available from: `https://doi.org/10.1523/JNEUROSCI.3022-16.2016`.

Duchi, J., 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. 12, pp.2121–2159.

Dupre La Tour, T., Lu, M., Eickenberg, M. and Gallant, J.L., 2021. A finer mapping of convolutional neural network layers to the visual cortex. *Nips*, (Svrhm), pp.1–11.

Falcon et al., W., 2019. Pytorch lightning. *Github. note: https://github.com/pytorchlightning/pytorch-lightning*, 3.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. Available from: `http://www.deeplearningbook.org`.

Gore, J., 2003. Principles and Practice of Functional MRI of the Human Brain. *J clini neurophysiol* [Online], 112, pp.4–9. Available from: `https://doi.org/https://doi.org/10.1172/JCI19010`.

Gramfort, A., Luessi, M., Larson, E. and Denis, A., 2013a. MEG and EEG Data Analysis with MNE-Python. *Frontiers in neuroscience* [Online], 7(267), pp.1–13. Available from: `https://doi.org/https://doi.org/10.3389/fnins.2013.00267`.

Gramfort, A., Luessi, M., Larson, E., Engemann, D.A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L. and Hämäläinen, M.S., 2013b. MEG and EEG data analysis with MNE-Python. *Frontiers in neuroscience* [Online], 7(267), pp.1–13. Available from: `https://doi.org/10.3389/fnins.2013.00267`.

Harris, C.R., Millman, K.J., Walt, S.J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M.H. van, Brett, M., Haldane, A., Río, J.F. del, Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T.E., 2020. Array programming with NumPy. *Nature* [Online], 585(7825), September, pp.357–362. Available from: `https://doi.org/10.1038/s41586-020-2649-2`.

Hartmann, K.G., Schirrmeister, R.T. and Ball, T., 2018. Hierarchical internal representation of spectral features in deep convolutional networks trained for EEG decoding. *2018 6th international conference* *on brain-computer interface, bci 2018* [Online], 2018-Janua, pp.1–6. 1711.07792, Available from: https://doi.org/10.1109/IWW-BCI.2018.8311493.

Haxby, J.V., Guntupalli, J.S., Nastase, S.A. and Feilong, M., 2020. Hyperalignment: Modeling shared information encoded in idiosyncratic cortical topographies. *elife* [Online], 9, pp.1–26. Available from: https://doi.org/10.7554/eLife.56601.

Ho, T.K., 1995. Random decision forests. *Proceedings of 3rd international conference on document analysis and recognition* [Online]. vol. 1, pp.278–282 vol.1. Available from: https://doi.org/10.1109/ICDAR.1995.598994.

Hodgkin, A.L. and Huxley, A.F., 1952. A Quantitative Description of Membrane Current and its Application to Conduction and Excititation in Nerve. *J. physiol.*, (117), pp.500–544 A.

Hornick, K., 1989. Multilayer Feedforward Networks are Universal Approximators. *Practical neural network recipies in c++* [Online], pp.77–116. Available from: https://doi.org/10.1016/b978-0-08-051433-8.50011-2.

Hosseini, M., Hosseini, A. and Ahi, K., 2020. A review on Machine Learning for EEG signal processing in bioengineering. *Ieee reviews in biomedical engineering* [Online], 99(1). Available from: https://doi.org/https://doi.org/10.1109/rbme.2020.2969915.

Huang, W., Xue, Y., Hu, L. and Liuli, H., 2020. S-eegnet: Electroencephalogram signal classification based on a separable convolution neural network with bilinear interpolation. *Ieee access* [Online], 8, pp.131636–131646. Available from: https://doi.org/10.1109/ACCESS.2020.3009665.

Hunter, J.D., 2007. Matplotlib: A 2d graphics environment. *Computing in science & engineering* [Online], 9(3), pp.90–95. Available from: https://doi.org/10.1109/MCSE.2007.55.

Ioffe, S. and Szegedy, C., 2015. Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167v3.

Jas, M., Engemann, D., Raimondo, F., Bekhti, Y. and Gramfort, A., 2016. Automated rejection and repair of bad trials in meg/eeg. *In 6th international workshop on pattern recognition in neuroimaging* [Online], 159(4), pp.417–429. Available from: https://doi.org/10.1109/TBME.2011.2172210.

Keogh, R., 2018. The blind mind: No sensory visual imagery in aphantasia. *Cortex* [Online], 105(0010-9452), pp.53–60. Available from: https://doi.org/https://doi.org/10.1016/j.cortex.2017.10.012.

Kingma, D.P. and Ba, J.L., 2015. Adam - A method for stochastic optimisation. pp.1–15. arXiv:1412.6980v9.

Kingma, D.P. and Welling, M., 2013. Auto-encoding variational bayes [Online]. Available from: https://doi.org/10.48550/ARXIV.1312.6114.

Lawhern, V., Solon, A., Waytowich, N., Gordon, S., Hung, C. and Lance, B., 2016. Eegnet: A compact convolutional network for eeg-based brain-computer interfaces. *Journal of neural engineering* [Online], 15, 11. Available from: https://doi.org/10.1088/1741-2552/aace8c.

LeCun, Y. and Boser, B., 1989. Backpropogation applied to handwritten zip code recognition. *At & t bell laboratories* [Online]. Available from: http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf.

Lee, S.H., Lee, M., Jeong, J.H. and Lee, S.W., 2019. Towards an EEG-based intuitive BCI communication system using imagined speech and visual imagery. *Conference proceedings - ieee international conference on systems, man and cybernetics* [Online], 2019-Octob(July 2020), pp.4409–4414. Available from: https://doi.org/10.1109/SMC.2019.8914645.

Lewis, J.W., Beauchamp, M.S. and DeYoe, E.A., 2000. A Comparison of Visual and Auditory Motion Processing in Human Cerebral Cortex. *Cerebral cortex* [Online], 10(9), 09, pp.873–888. https://academic.oup.com/cercor/article-pdf/10/9/873/9751086/100873.pdf, Available from: https://doi.org/10.1093/cercor/10.9.873.

Li, A., Feitelberg, J., Scheltienne, M. and Saini, A., 2022. mne-icalabel. https://github.com/mne-tools/mne-icalabel.

Li, F., He, F., Wang, F., Zhang, D., Xia, Y. and Li, X., 2020. A novel simplified convolutional neural network classification algorithm of motor imagery EEG signals based on deep learning. *Applied sciences (switzerland)* [Online], 10(5). Available from: https://doi.org/10.3390/app10051605.

Lisman, J., 2015. The challenge of understanding the brain: where we stand in 2015. *Frontiers in neuroscience* [Online], 86, pp.864–882. Available from: https://doi.org/https://dx.doi.org/10.1016%2Fj.neuron.2015.03.032.

Luo, Z., Hu, Z. and Li, Z., 2020. Estimation of Motor Imagination Based on Consumer-Grade EEG Device. *Ml for cyber security*.

MacKay, D., 1994. Bayesian non-linear modelling for the energy prediction competition. *Ashrae transcations* [Online], 4, pp.448–472. Available from: https://bayes.wustl.edu/MacKay/pred.pdf.

Mackay, D., 2003. *Information theory, inference, and learning algorithms*, vol. 2. Available from: https://www.inference.org.uk/itprnn/book.pdf.

Macpherson, F., 2011. Indivuating the senses. *The senses: Classic and contemporary philosophical perspectives. oxford university press* [Online]. Available from: https://philpapers.org/rec/MACTSC-6.

Macpherson, F., 2012. Cognitive Penetration of Colour Experience: Rethinking the Issue in Light of an Indirect Mechanism. *Philosophy and phenomenological research* [Online], 84(1), pp.24–62. Available from: https://doi.org/https://philpapers.org/go.pl?id=MACIM&proxyId=&u=https%3A%2F%2Fdx.doi.org%2F10.1111%2Fj.1933-1592.2010.00481.x.

Mane, A.R., Biradar, P.S.D. and Shastri, P.R.K., 2015. Review paper on Feature Extraction Methods for EEG Signal Analysis. *International journal of emerging trend in engineering and basic sciences (ijeebs)*, 2(1), pp.545–552.

McKinney Wes, 2010. Data Structures for Statistical Computing in Python. In: Stéfan van der Walt and Jarrod Millman, eds. *Proceedings of the 9th Python in Science Conference* [Online]. pp.56 – 61. Available from: https://doi.org/10.25080/Majora-92bf1922-00a.

Meijer, G.T., Mertens, P.E., Pennartz, C.M., Olcese, U. and Lansink, C.S., 2019. The circuit architecture of cortical multisensory processing: Distinct functions jointly operating within a common anatomical network. *Progress in neurobiology* [Online], 174, pp.1–15. Available from: https://doi.org/https://doi.org/10.1016/j.pneurobio.2019.01.004.

Miranda, Í.M., Aranha, C. and Ladeira, M., 2019. Classification of EEG Signals using Genetic Programming for Feature Construction. *Gecco 2019 - proceedings of the 2019 genetic and evolutionary computation conference* [Online], pp.1275–1283. 1906.04403, Available from: https://doi.org/10.1145/3321707.3321737.

Molla, M.K.I., Shiam, A.A., Islam, M.R., Tanaka, T., Tanaka, T. and Tanaka, T., 2020. Discriminative Feature Selection-Based Motor Imagery Classification Using EEG Signal. *Ieee access* [Online], 8, pp.98255–98265. Available from: https://doi.org/10.1109/ACCESS.2020.2996685.

Nanay, B., 2010. Perception and imagination: amodal perception as mental imagery. *Philosophical studies: An international journal for philosophy in the analytic tradition* [Online], 150(2), pp.239–254. Available from: http://www.jstor.org/stable/40856553 [Accessed 2022-05-10].

Olejniczak, P., 2006. Neurophysiologic Basis of EEG. *J clini neurophysiol* [Online], 23, pp.186–189. Available from: https://doi.org/https://doi.org/10.1097/01.wnp.0000220079.61973.6c.

Oostenveld, R. and Praamstra P, 2001. The five percent electrode system for high-resolution EEG and ERP measurements. *Clin neuriphysiol* [Online], 12, pp.713–719. Available from: https://doi.org/https://doi.org/10.1016/S1388-2457(00)00527-7.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds. *Advances in neural information processing systems*

32 [Online]. Curran Associates, Inc., pp.8024–8035. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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. and Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12, pp.2825–2830.

Perky, C.W., 1910. An Experimental study of imagination. *The american journal of psychology* [Online], 21(3), pp.422–452. Available from: https://doi.org/https://doi.org/10.2307/1413350.

Pion-Tonachini, L., Kreutz-Delgado, K. and Makeig, S., 2019. ICLabel: An automated electroencephalographic independent component classifier, dataset, and website. *Neuroimage* [Online], 198, sep, pp.181–197. Available from: https://doi.org/10.1016/j.neuroimage.2019.05.026.

Ramachandran, P., Zoph, B. and Le, Q.V., 2017. Searching for activation functions [Online]. Available from: https://doi.org/10.48550/ARXIV.1710.05941.

Reaves, J., Flavin, T., Mitra, B. and Mahantesh, K., 2021. Assessment And Application of EEG : A Literature Review Journal of Applied Assessment And Application of EEG : A Literature Review. *J appl bioinforma comput biol*, 10:7(August).

Rivet*, B., Souloumiac, A., Attina, V. and Gibert, G., 2009. xdawn algorithm to enhance evoked potentials: Application to brain–computer interface. *Ieee transactions on biomedical engineering* [Online], 56(8), pp.2035–2043. Available from: https://doi.org/10.1109/TBME.2009.2012869.

Robbins, H. and Monro, S., 1952. A stochastic approximation method. *The annals of mathematical statistics*, pp.400–407.

Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T.H. and Faubert, J., 2019. Deep learning-based electroencephalography analysis: A systematic review. *Journal of neural engineering* [Online], 16(5). 1901.05498, Available from: https://doi.org/10.1088/1741-2552/ab260c.

Rumelhart, D., Hinton, G. and Williams, R., 1986. Learning representations by back-propagating errors. *Nature* [Online], 323, pp.533–536. Available from: https://doi.org/10.1038/323533a0.

Rushton, J. and Ankney, C., 2009. Whole Brain Size and General Mental Ability: A Review. *Int j neurosci* [Online], 119(5), pp.692–732. Available from: https://doi.org/https://dx.doi.org/10.1080%2F00207450802325843.

Russel, S. and Norvig, P., 2002. *Artificial Intelligence: A Modern Approach* [Online]. Available from: https://www.cin.ufpe.br/$\sim$tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf.

Rybář, M., Poli, R. and Daly, I., 2021. Decoding of semantic categories of imagined concepts of animals and tools in fNIRS. *Journal of neural engineering* [Online], 18(4). Available from: https://doi.org/10.1088/1741-2552/abf2e5.

Santana, L.M.Q.D., Santos, R.M., Matos, L.N. and Macedo, H.T., 2012. Deep Neural Networks for Acoustic Modeling in the Presence of Noise. *Ieee latin america transactions* [Online], 16(3), pp.918–925. Available from: https://doi.org/10.1109/TLA.2018.8358674.

Santoso, I., 2020. Epileptic EEG Signal Classification Using Convolutional Neural Networks Based on Optimum Window Length and FFT's Length. *Icccm* [Online], pp.87–91. Available from: https://doi.org/https://doi.org/10.1145/3411174.3411179.

Schmidhuber, J., 2015. Deep Learning in neural networks: An overview. *Neural networks* [Online], 61, pp.85–117. 1404.7828, Available from: https://doi.org/10.1016/j.neunet.2014.09.003.

Shoemaker, S., 1994. Phenomenal Character. *Wiley*, 28(1), pp.21–38.

Shorten, C. and Khoshgoftaar, T.M., 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of big data* [Online]. Available from: https://doi.org/10.1186/s40537-019-0197-0.

Siclari, F., Baird, B., Perogamvros, L., Bernardi, G., LaRocque, J.J., Riedner, B., Boly, M., Postle, B.R. and Tononi, G., 2017. The neural correlates of dreaming [Online]. 20(6), pp.872–878. Available from: https://doi.org/10.1038/nn.4545.

Song, L., Kolar, M. and Xing, E., 2009. Time-Varying Dynamic Bayesian Networks. *Advances in neural information processing systems* [Online], 22. Available from: https://proceedings.neurips.cc/paper/2009/file/a67f096809415ca1c9f112d96d27689b-Paper.pdf.

Srivastava, N. and Hinton, G., 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of machine learning research* [Online], 15, pp.1929–1958. Available from: https://www.cs.toronto.edu/%7B$\sim$%7Drsalakhu/papers/srivastava14a.pdf.

Sun, K.T., Hsieh, K.L. and Syu, S.R., 2020. Towards an accessible use of a brain-computer interfaces-based home care system through a smartphone. *Computational intelligence and neuroscience* [Online], 2020, pp.16–18. Available from: https://doi.org/10.1155/2020/1843269.

Teng, L., 2021. Cognitive Penetration: Inference or Fabrication? *Cognitive penetration: Inference or fabrication?* [Online], 99, pp.547–563. Available from: https://doi.org/https://doi.org/10.1080/00048402.2020.1812095.

Torse, D. and Desai, V., 2016. Design of adaptive EEG preprocessing algorithm for neurofeedback system. *Iccsp* [Online]. Available from: https://doi.org/https://doi.org/10.1109/ICCSP.2016.7754164.

Waskom, M.L., 2021. seaborn: statistical data visualization. *Journal of open source software* [Online], 6(60), p.3021. Available from: https://doi.org/10.21105/joss.03021.

Wiesel, T.N., 1968. Receptive fields and functional architecture of monkey striate cortex. *J. physiol* [Online], 195, pp.215–243. Available from: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1968.sp008455.

Wu, M.C., David, S.V. and Gallant, J.L., 2006. Complete functional characterization of sensory neurons by system identification. *Annual review of neuroscience* [Online], 29(September 2015), pp.477–505. Available from: https://doi.org/10.1146/annurev.neuro.29.051605.113024.

Wu, X., Zhang, Y. and Xiaojun, W., 2019. An Improved Noise Elimination Model of EEG Based on Second Order Volterra Filter. *Proceedings of the 2019 3rd international conference on digital signal processing* [Online], pp.60–64. Available from: https://doi.org/https://doi.org/10.1145/3316551.3316565.

Xie, S., Kaiser, D. and Cichy, R.M., 2020. Visual imagery and perception share neural representations in the alpha frequency band. *Curr. biol.*, 30(13), July, pp.2621–2627.e5.

Yamins, D.L.K., Hong, H., Cadieu, C.F., Solomon, E.A., Seibert, D. and DiCarlo, J.J., 2014. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences of the united states of america* [Online], 111(23), pp.8619–8624. Available from: https://doi.org/10.1073/pnas.1403112111.

Zeman, A., Dewar, M. and Della Sala, S., 2015. Lives without imagery - congenital aphantasia. *Cortex* [Online], 73, pp.378–380. Available from: https://doi.org/10.1016/j.cortex.2015.05.019.

Zeman, A., Logie, R., Della Salla, S., Torrens, L., Gountouna, V. and McGonigle, D., 2010. Loss of imagery phenomenology with intact visuo-spatial task performance: A case of 'blind imagination'. *Neuropsychologia* [Online], 48(1), pp.145–155. Available from: https://doi.org/10.1016/j.neuropsychologia.2009.08.024.

Zhang, H., Wang, Z., Yu, Y., Yin, H., Chen, C. and Wang, H., 2022. An improved eegnet for single-trial eeg classification in rapid serial visual presentation task. *Brain science advances* [Online], 8(2), pp.111–126. https://doi.org/10.26599/BSA.2022.9050007, Available from: https://doi.org/10.26599/BSA.2022.9050007.

Zhang, M., 2020. Classification of Vigilance Based on EEG. *Annu int conf ieee eng med biol soc* [Online], pp.76–80. Available from: https://doi.org/https://doi.org/10.1145/3406971.3406987.

Zhao, S. and Rudzicz, F., 2015. Classifying phonological categories in imagined and articulated speech.

2015 ieee international conference on acoustics, speech and signal processing (icassp) [Online]. pp.992–996. Available from: https://doi.org/10.1109/ICASSP.2015.7178118.

Zotev, V., Phillips, R., Yuan, H., Misaki, M. and Bodurka, J., 2014. Self-regulation of human brain activity using simultaneous real-time fMRI and EEG neurofeedback. *Neuroimage* [Online], 85, pp.985–995. 1301.4689, Available from: https://doi.org/10.1016/j.neuroimage.2013.04.126.

Özbeyaz, A. and Arıca, S., 2018. Familiar/unfamiliar face classification from eeg signals by utilizing pairwise distant channels and distinctive time interval. *Signal, image and video processing* [Online], 12, 09. Available from: https://doi.org/10.1007/s11760-018-1269-x.

Appendix A

Misc

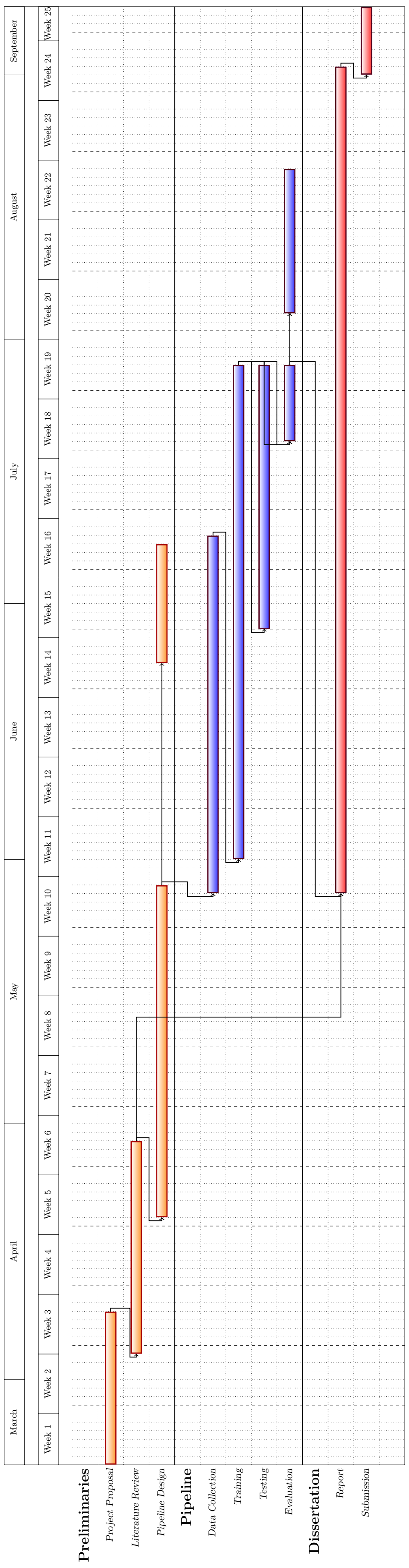


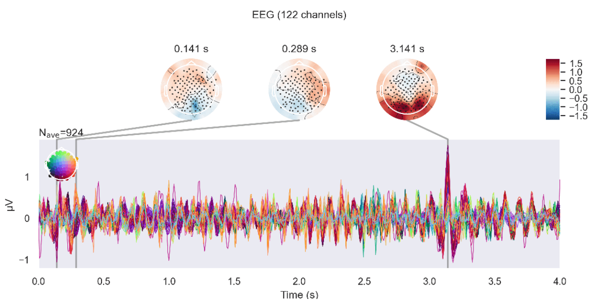
Figure A.1: Initial timeline proposal for the machine learning pipeline.

Appendix B

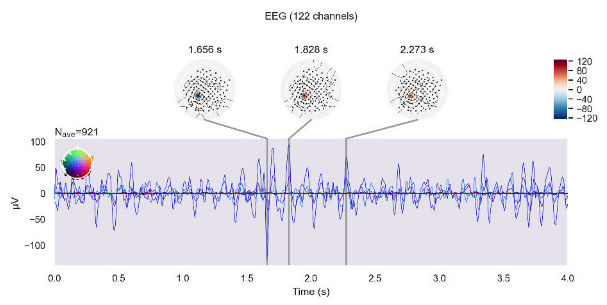
Results

B.1 Ablation Study

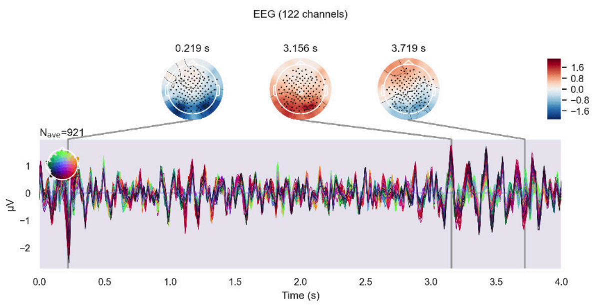
B.1.1 Before Ransac/Autoreject



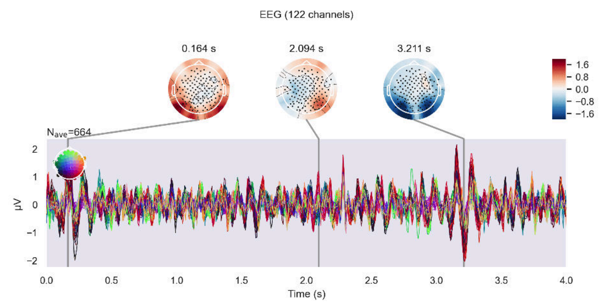
(a) Participant 8 - Session 2



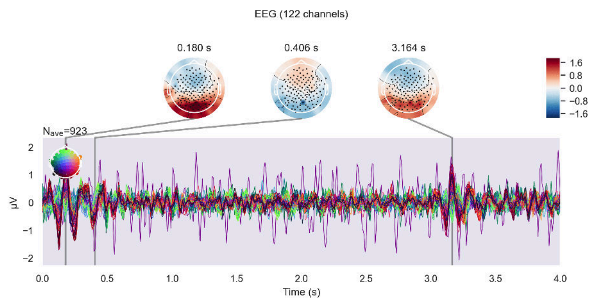
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

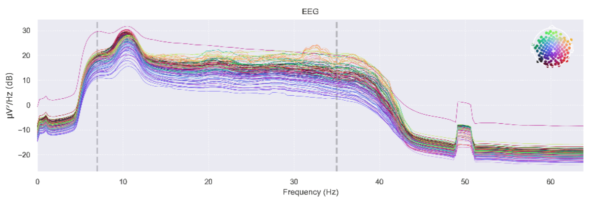


(d) Participant 15 - Session 1

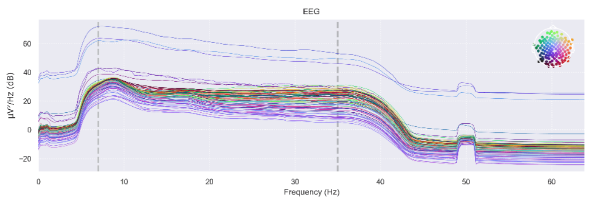


(e) Participant 17 - Session 1

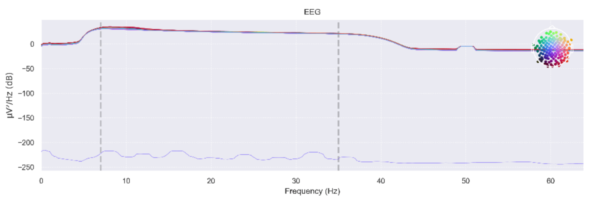
Figure B.1: 10-1 appears to have a rogue channel which is an order of magnitude greater than the others. Strong event related potentials seem to occur in all subjects around the 0.3 and 3 second mark. 8-2 and 14-2 look the cleanest, yet it was noted during the recording that event timings were corrupted which will compound timing errors with future processing.



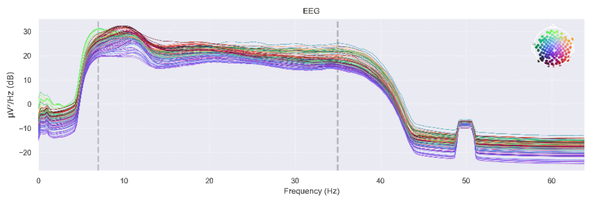
(a) Participant 8 - Session 2



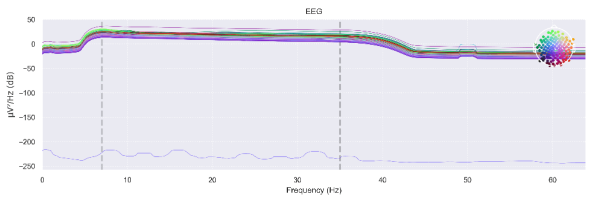
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

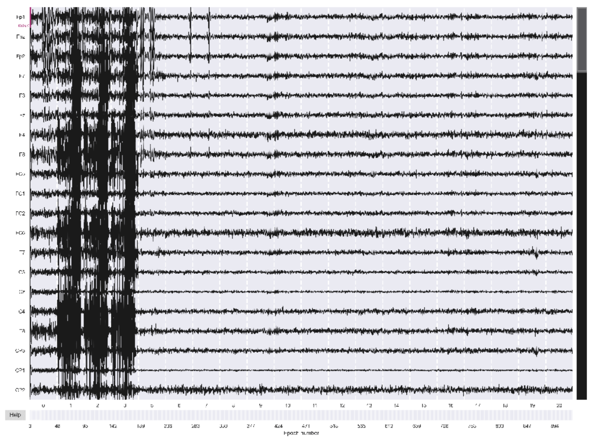


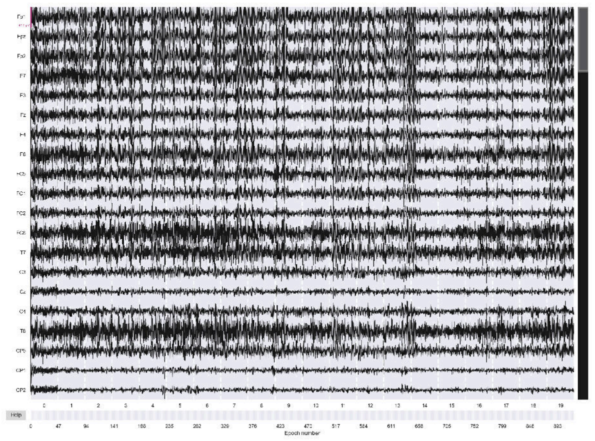
(d) Participant 15 - Session 1

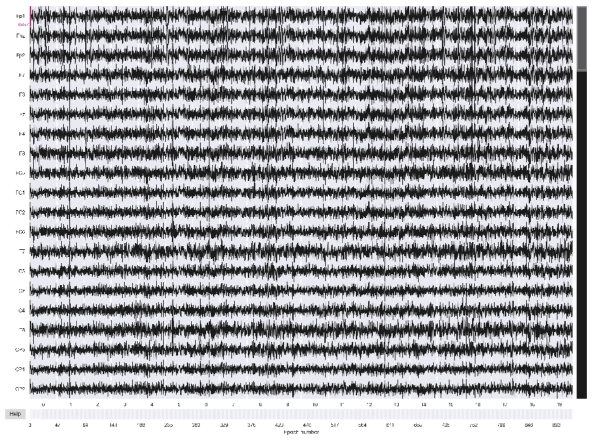


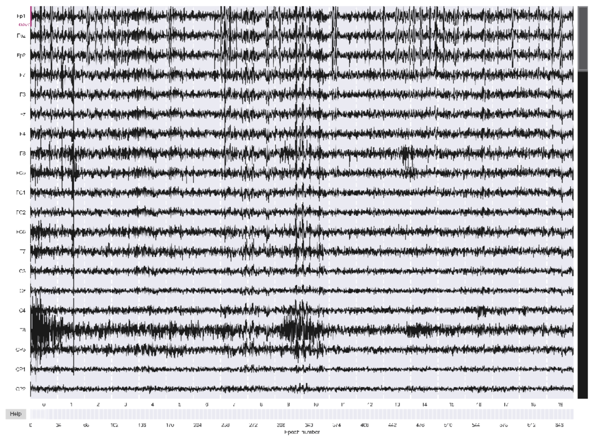
(e) Participant 17 - Session 1

Figure B.2: 14-2 and 17-1 appear to have a rogue channel which is likely caused by a faulty sensor with poor scalp contact.


(a) Participant 8 - Session 2


(b) Participant 10 - Session 1


(c) Participant 14 - Session 2


(d) Participant 15 - Session 1

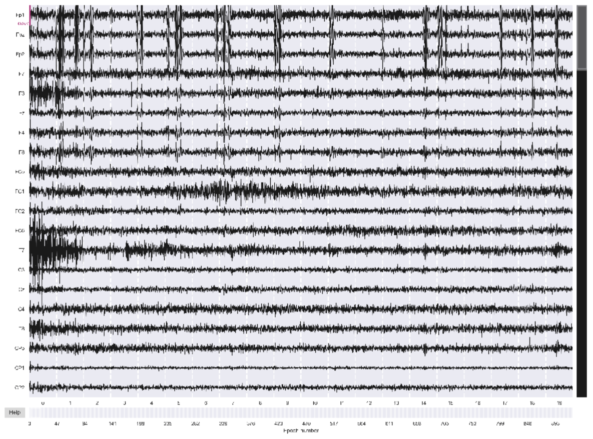
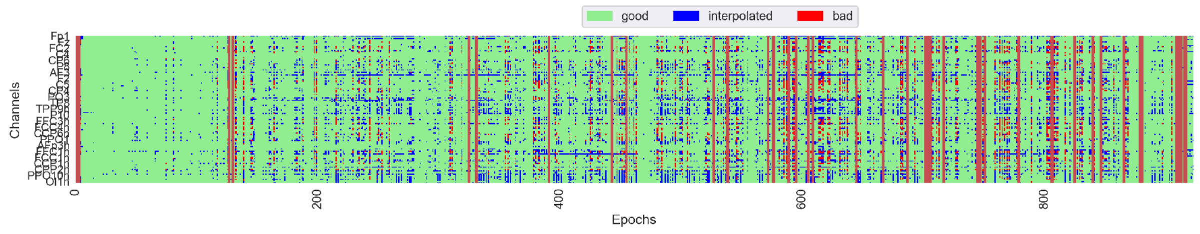
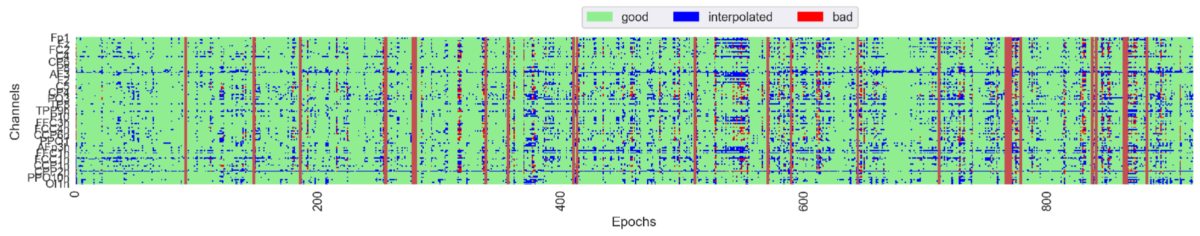

(e) Participant 17 - Session 1

Figure B.3: At a glance, we can see similarities across the different subjects raw data, which at this stage is more indicative of the artifacts present in the shared environment they were recorded in.

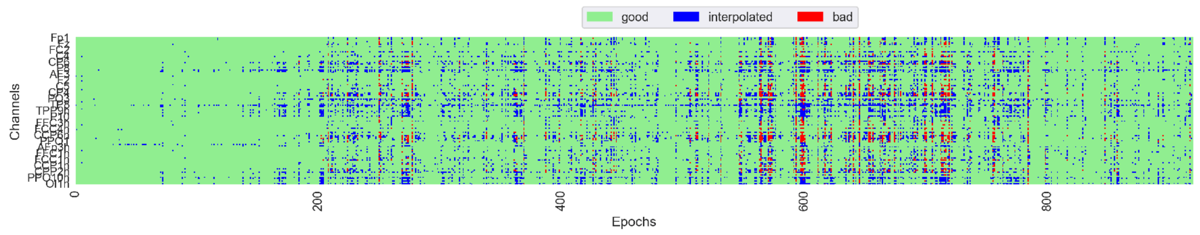
B.1.2 After Ransac/Autoreject



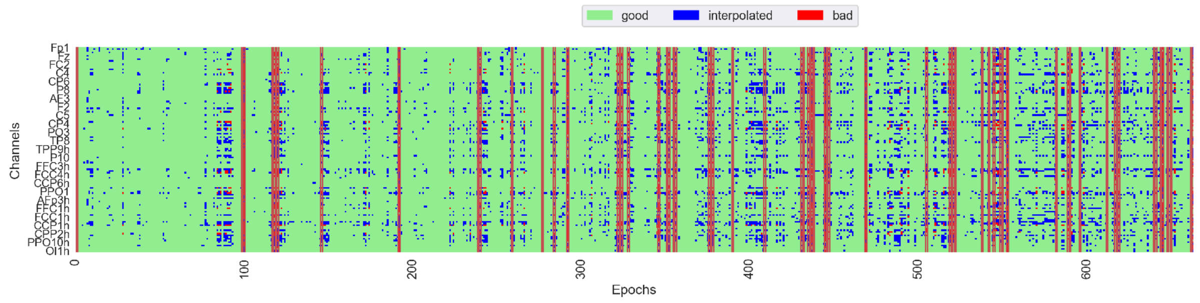
(a) Participant 8 - Session 2



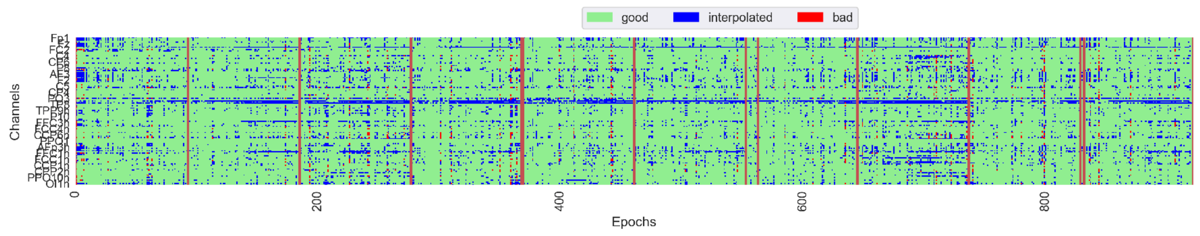
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

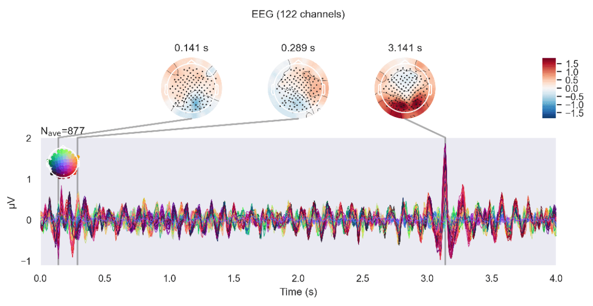


(d) Participant 15 - Session 1

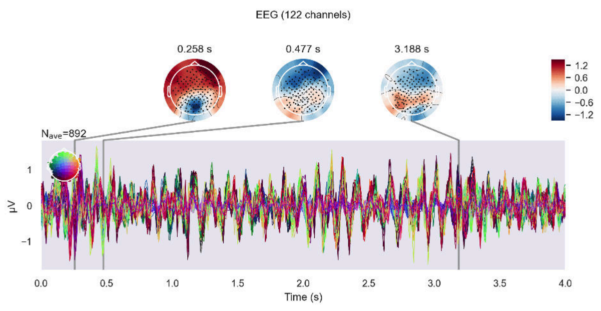


(e) Participant 17 - Session 1

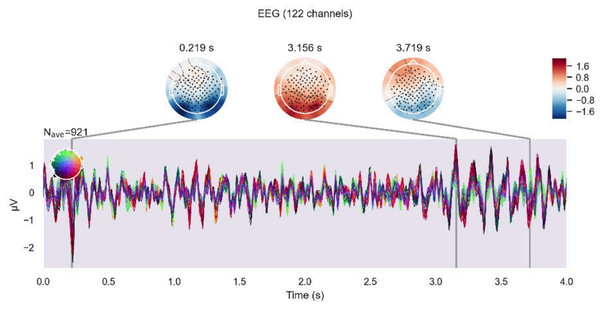
Figure B.4: Very few bad epochs and channels are detected in participant 14, in agreement with previous analysis suggesting that this trial had less artifacts present.



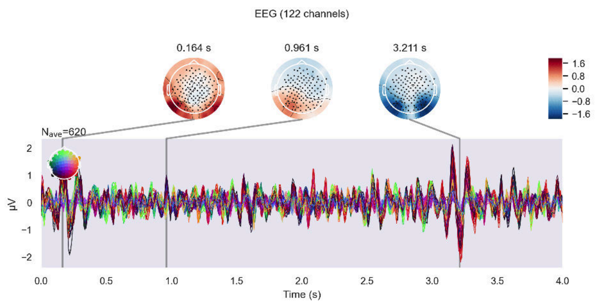
(a) Participant 8 - Session 2



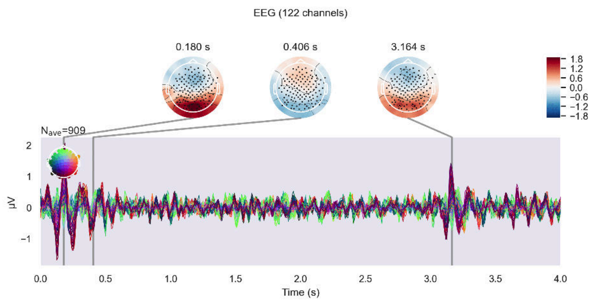
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

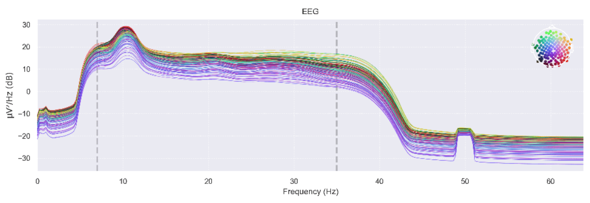


(d) Participant 15 - Session 1

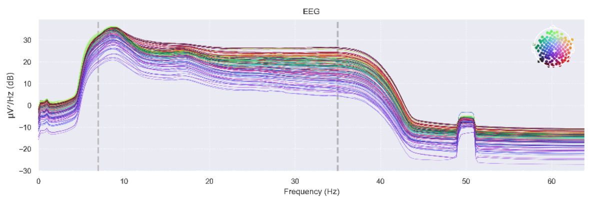


(e) Participant 17 - Session 1

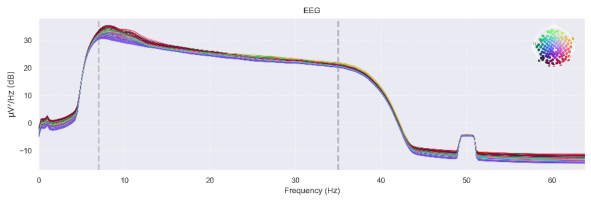
Figure B.5: After ransac and autoreject, we observe that the signals are much more constrained and share similar event timings between participants.



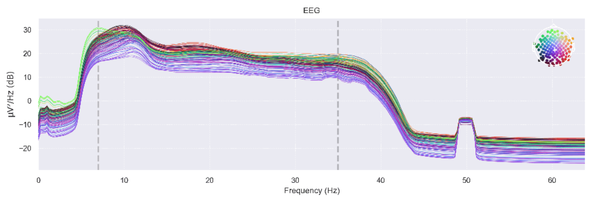
(a) Participant 8 - Session 2



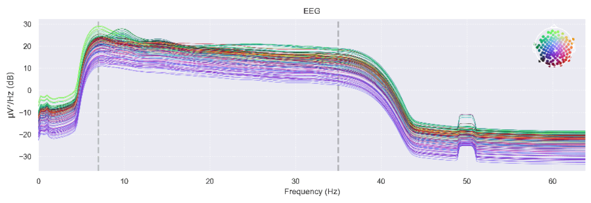
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

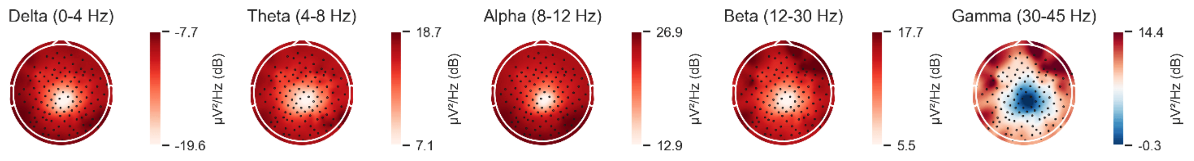


(d) Participant 15 - Session 1

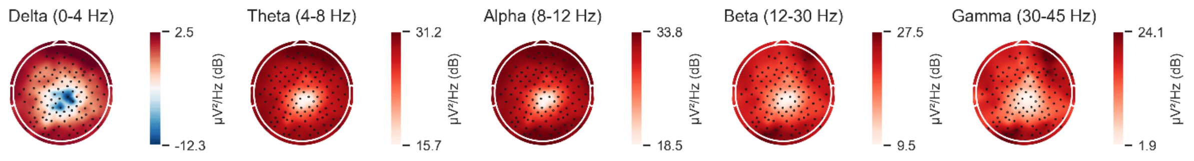


(e) Participant 17 - Session 1

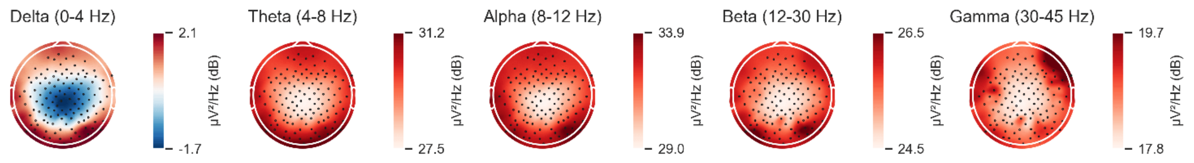
Figure B.6: The variance between channels in participant 14 are well constrained. In comparison to the PSD before ransac (Figure B.2) we can see that the automated cleaning process has had a positive effect in cleaning rogue channels.



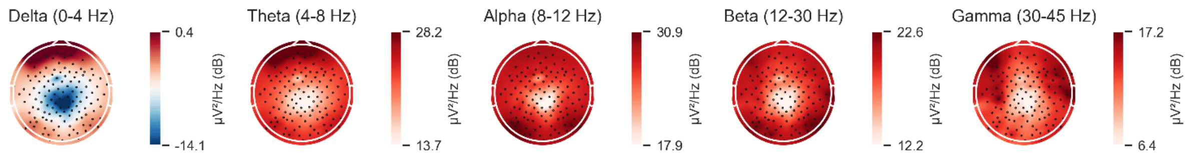
(a) Participant 8 - Session 2



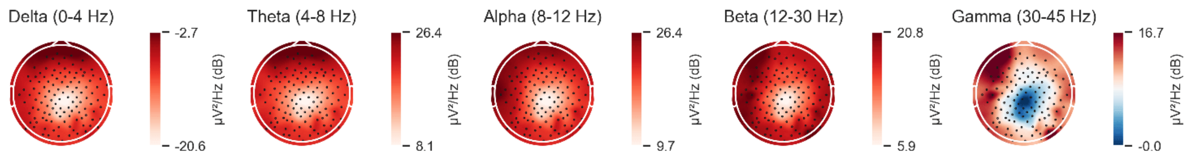
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

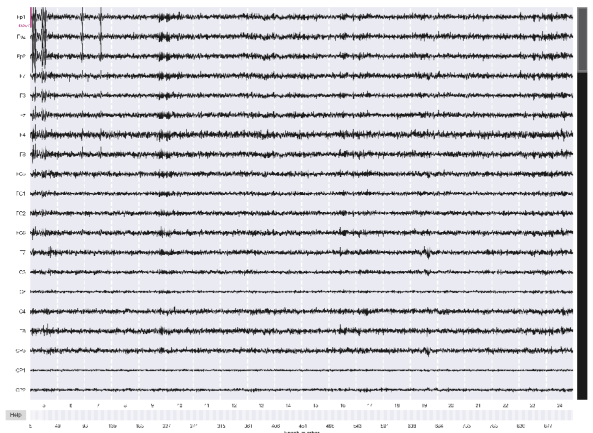


(d) Participant 15 - Session 1

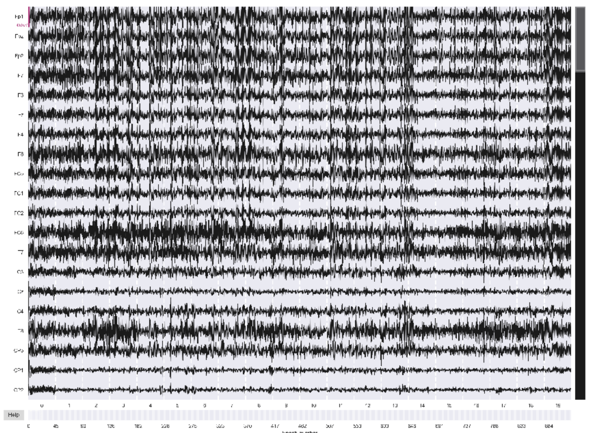


(e) Participant 17 - Session 1

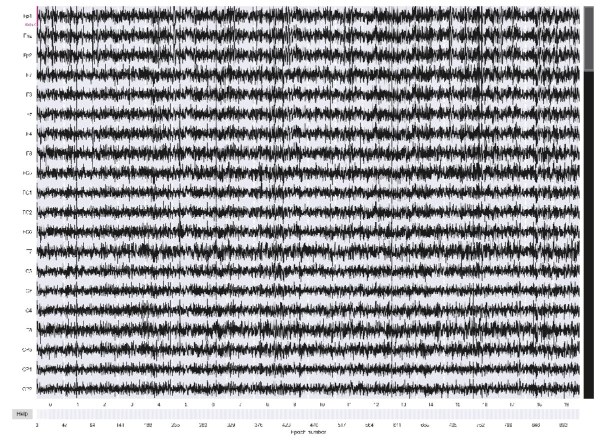
Figure B.7: Participant 14 appears to have a smoother transition in its signal to noise, and exhibits much more constraint as previously shown. The others have significantly more variance in this regard.



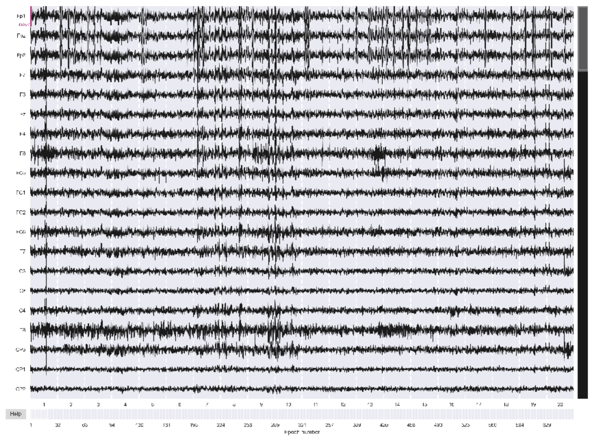
(a) Participant 8 - Session 2



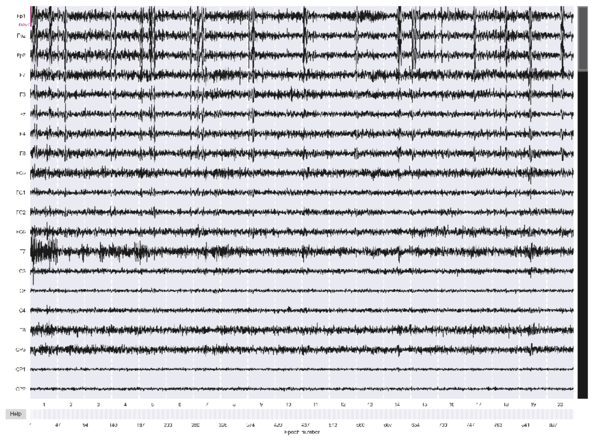
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2



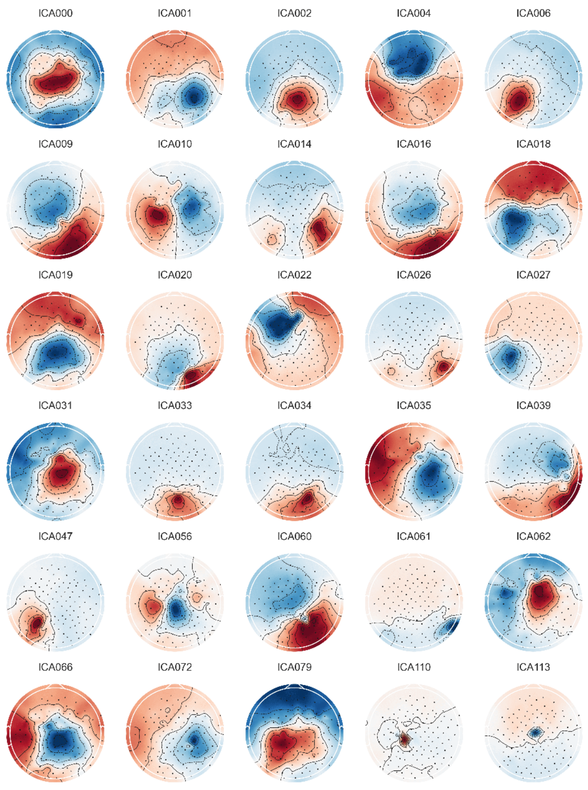
(d) Participant 15 - Session 1



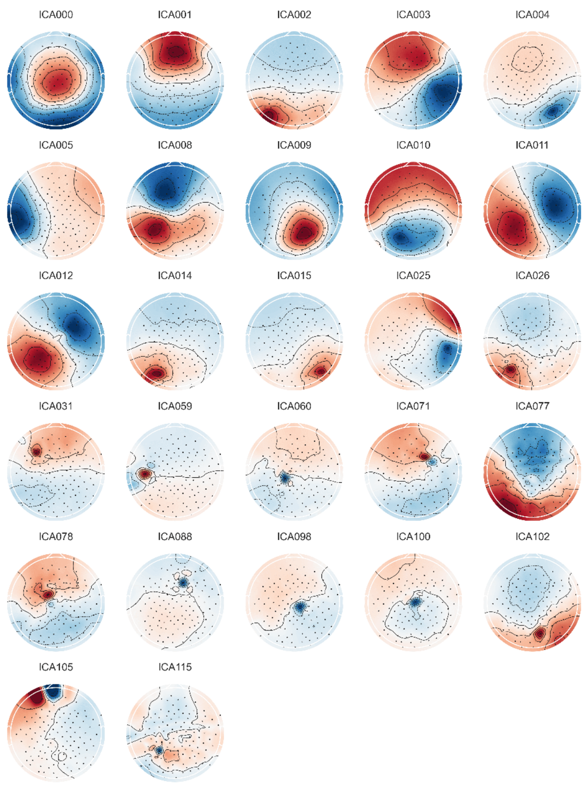
(e) Participant 17 - Session 1

Figure B.8: Again, participant 14 can be seen to have consistent signals across the channels. Visual inspection suggests we can see possible patterns but still the signal to noise is quite poor.

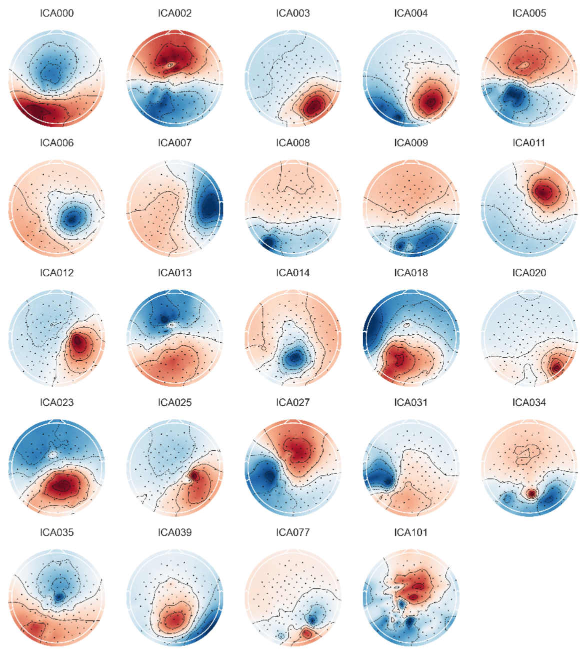
B.1.3 After ICA



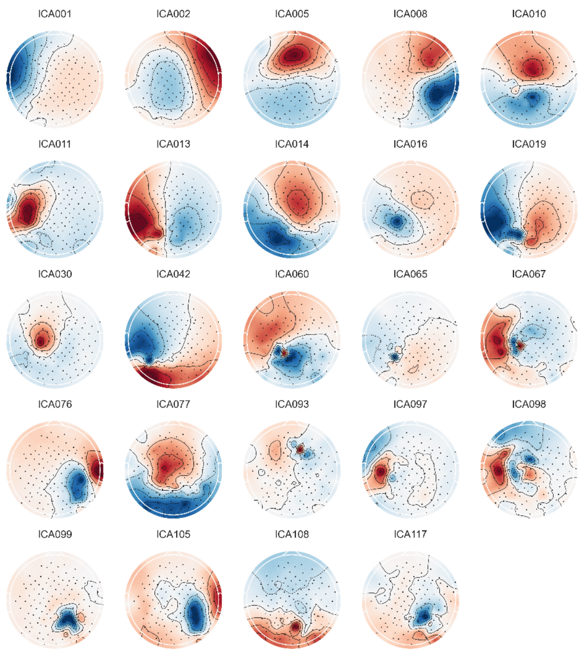
(a) Participant 10 - Session 1



(b) Participant 14 - Session 2

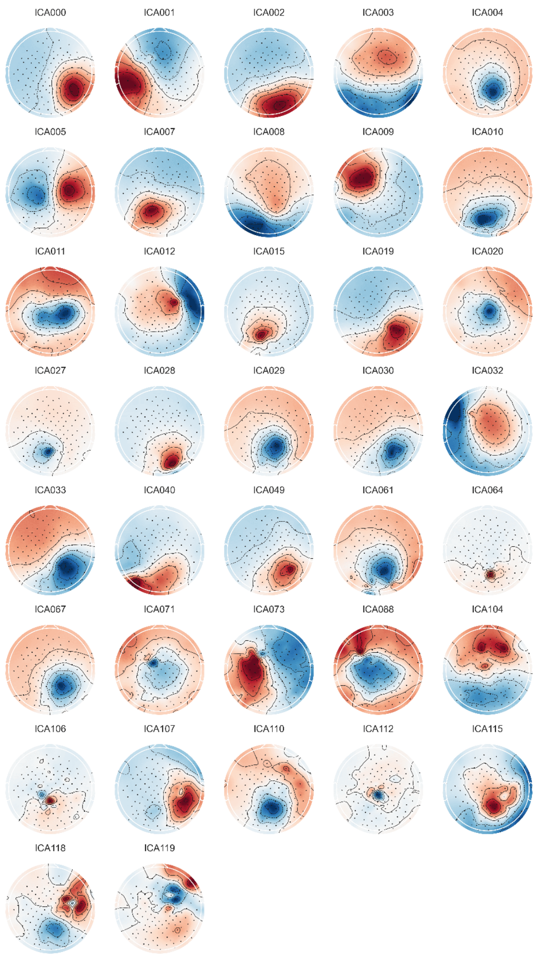


(c) Participant 15 - Session 1



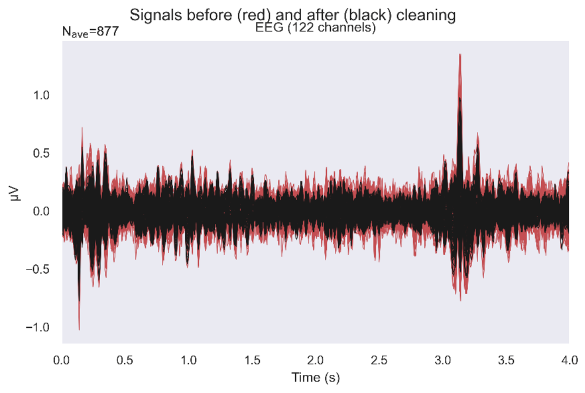
(d) Participant 17 - Session 1

Figure B.9: The components marked as brain activity, automatically classified using IC-Label.

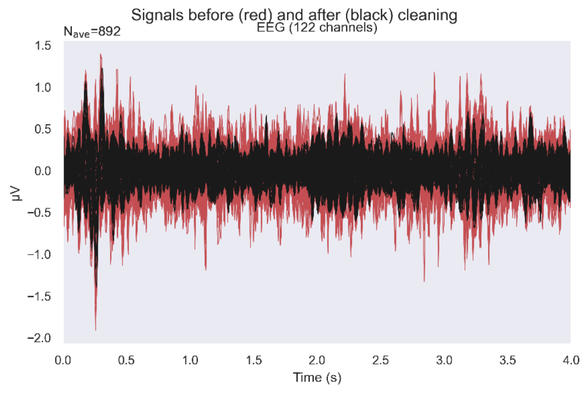


(a) Participant 8 - Session 2

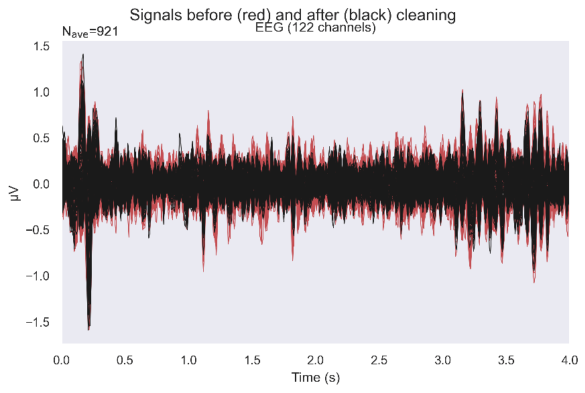
Figure B.10: The components marked as brain activity, automatically classified using IC-Label.



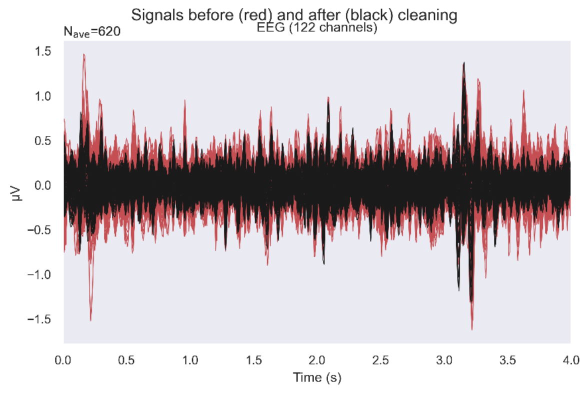
(a) Participant 8 - Session 2



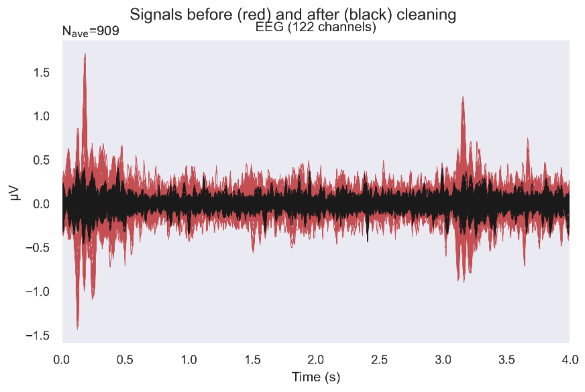
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

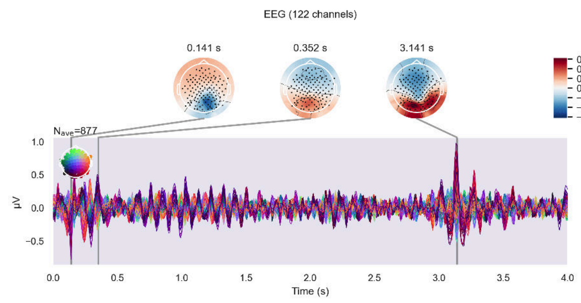


(d) Participant 15 - Session 1

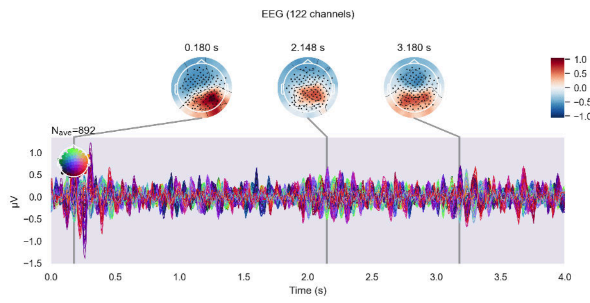


(e) Participant 17 - Session 1

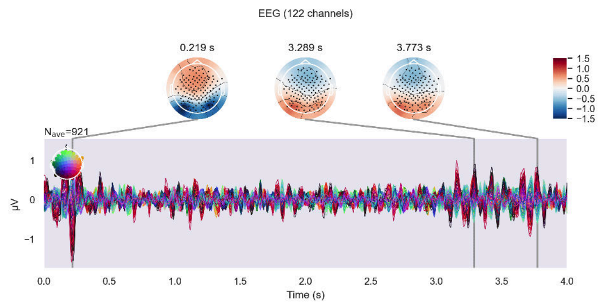
Figure B.11: The effect of ICA decomposition. Marked in red are the signals before, and black after cleaning. Strong spikes remain in all but participant 17.



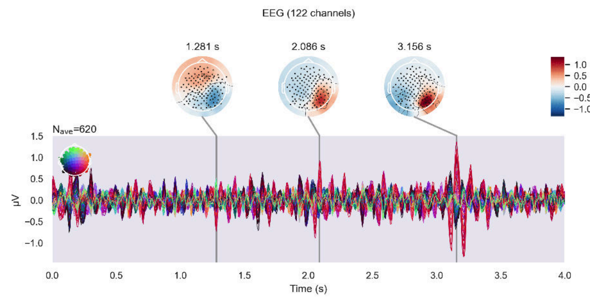
(a) Participant 8 - Session 2



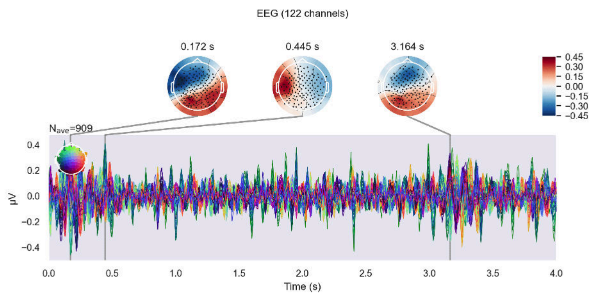
(b) Participant 10 - Session 1



(c) Participant 14 - Session 2

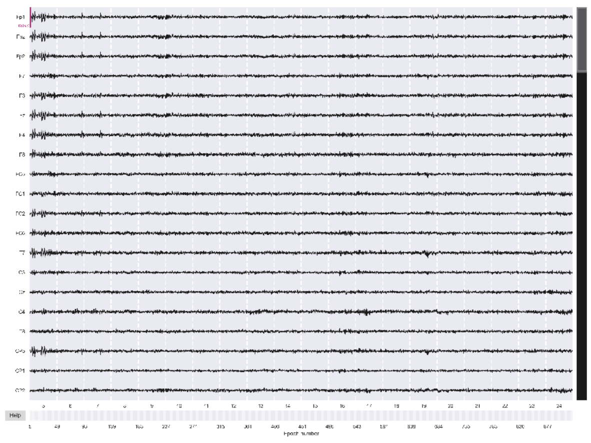


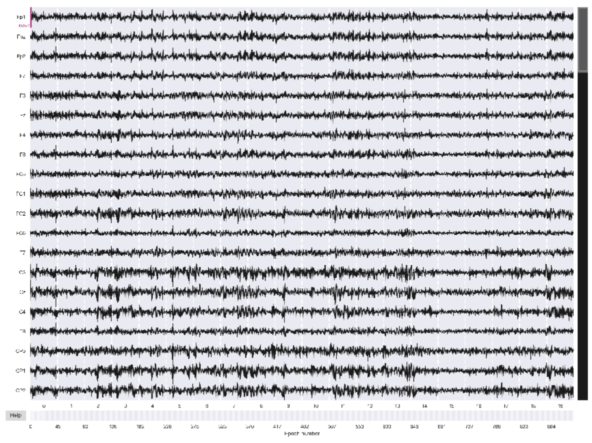
(d) Participant 15 - Session 1

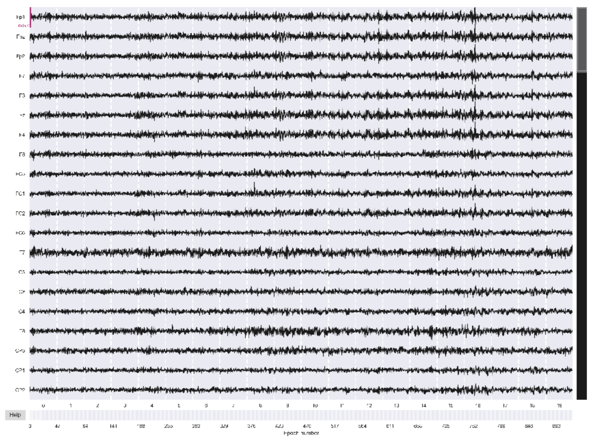


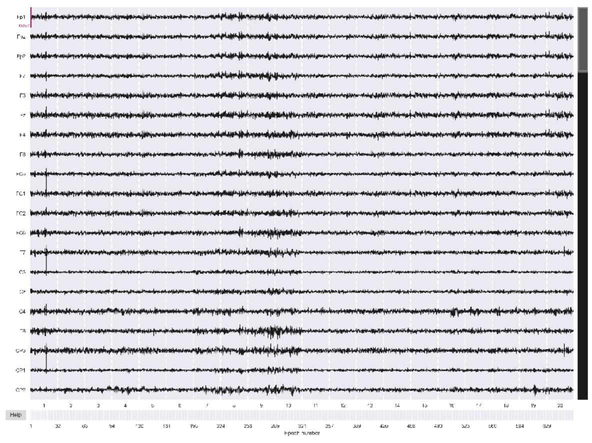
(e) Participant 17 - Session 1

Figure B.12: The signal to noise issues present before ICA show improvement from before, indicating that many of the artifacts showing long term trends have been removed, leaving us primarily with non-stationary data.


(a) Participant 8 - Session 2


(b) Participant 10 - Session 1


(c) Participant 14 - Session 2


(d) Participant 15 - Session 1

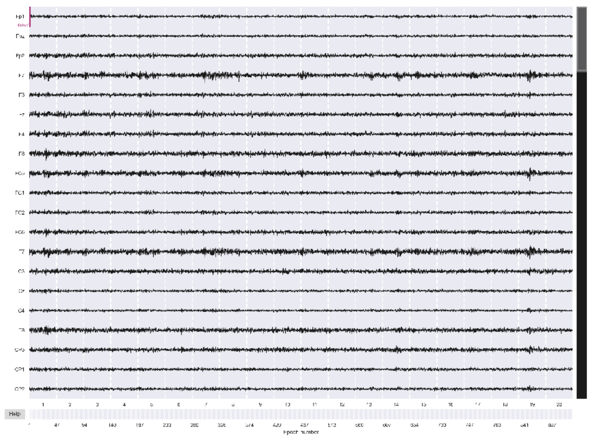

(e) Participant 17 - Session 1

Figure B.13: The events listed here show the impact of the entire pipeline in comparison to Figure B.3, before any pre-processing. It can be seen that the differences are quite drastic, and much of the artifacts appear to have been removed. Its possible this approach is too aggressive, but the nature of this dataset in particular is a challenge for decoders to see any separation between the classes.

Appendix C

Code

C.1 Pre-Processing

```python
import mne
from mne import preprocessing
import pandas as pd
from sklearn.model_selection import train_test_split
from mne.io import read_raw_eeglab
from mne import channels
from mne.io import eeglab
import numpy as np
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
import seaborn as sns
sns.set_theme('paper', style='dark')
import warnings
warnings.filterwarnings("ignore")
from mne import compute_raw_covariance
from mne_icalabel import label_components

from autoreject import AutoReject

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Montage
ant_montage = channels.read_custom_montage(
    'eeg_data/edit.loc', coord_frame= 'head', head_size=0.08)
raw.set_montage(ant_montage)
# raw.plot_sensors(show_names=True)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Remove/Clean known bad channels
if cap_size == 'large':
    raw.info['bads'] += ['CCP1h']
# picks can then be taken into epochs
picks = mne.pick_types(raw.info, exclude='bads')
raw.filter(7, 35, n_jobs=12)
# Alpha 8-12Hz
# raw.filter(8, 12, n_jobs=12)
# Clean Power Line
# raw.notch_filter(freqs=None, method='spectrum_fit', n_jobs=12)
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# Epoching for ICA
```

```
    events, event_ids = mne.events_from_annotations(
        raw, verbose = False)
    epochs = mne.Epochs(
        raw=raw,
        events=events,
        event_id=event_ids,
        preload=True,
        tmin=0,
        tmax=4,
        baseline=None,
        event_repeated='merge',
        decim=8,
    )
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
    # Ransac/Autoreject
    rs = Ransac(n_jobs=12)
    epochs = rs.fit_transform(epochs)
    ar = AutoReject(n_jobs=12)
    epochs, reject_log = ar.fit_transform(epochs, return_log=True)
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
    # ICA
    epochs.set_eeg_reference()
    _, chans, _ = epochs.get_data().shape
    ica = mne.preprocessing.ICA(
        method='picard', fit_params=dict(extended=True, ortho=False)
    )
    ica.fit(epochs)
    # ICA Labels
    ic_labels = label_components(epochs, ica, method='iclabel')
    labels = ic_labels["labels"]
    exclude_idx = [
        i for i, label in enumerate(labels) if label not in ["brain"]
    ]
    picks_idx = [
        i for i, label in enumerate(labels) if label in ["brain"]
    ]
    print(f"Excluding these ICA components: {exclude_idx}")
    # Apply ICA
    ica.apply(epochs, include=picks_idx, exclude=exclude_idx)
```

C.2 Utilities

```
import tensorflow as tf
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
import seaborn as sns
sns.set_theme('paper', style='dark')

from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix

def conf_matrix(y_imag, y_pred, modality):
        """Plot the Confusion Matrix"""
        names = ['Flower', 'Guitar', 'Penguin']
        cm = confusion_matrix(y_imag, y_pred)
```

```
            ConfusionMatrixDisplay(cm, display_labels=names).plot(colorbar=False)
            acc2 = np.mean(y_pred == y_imag)
            plt.title(f'{modality}, Accuracy: {acc2:.2f}')
            plt.show()

    def history(df):
        fig, ax = plt.subplots(figsize=(10,10))
        epoch_ = range(len(df))
        plt.plot(epoch_, df['loss'], label='Train Loss')
        plt.plot(epoch_, df['val_loss'], label='Val Loss')
        plt.plot(epoch_, df['accuracy'], label='Train Accuracy')
        plt.plot(epoch_, df['val_accuracy'], label='Val Accuracy')
        plt.legend()
        plt.ylabel('Loss/Accuracy')
        plt.xlabel('Epoch')

    class Reshape(BaseEstimator, TransformerMixin):
        """Reshapes to (trials, channels, samples, kernel)."""
        def __init__(self):
            pass
        def fit(self, X, y=None):
            return self
        def transform(self, X, y=None):
            kernels = 1
            trials, chans, samples = X.shape
            return X.reshape(trials, chans, samples, kernels)
```

C.3 Neural Networks

This extension to *EEGNet* called *TSGL-EEGNet* (Deng et al., 2021) adds a regularization layer. The rest of the base pipeline remained the same in analysis from Appendix C.

```
    class TSG(Regularizer):
        '''Regularizer for TSG regularization.'''
        def __init__(self, l1=0., l21=0., tl1=0.):
            self.l1 = K.cast_to_floatx(l1)
            self.l21 = K.cast_to_floatx(l21)
            self.tl1 = K.cast_to_floatx(tl1)
        def __call__(self, x):
            if not self.l1 and not self.l21 and not self.tl1:
                return K.constant(0.)
            regularization = 0.
            if x.shape[0] == 1:
                ntf = tf.squeeze(x, 0)
            elif x.shape[1] == 1:
                ntf = tf.squeeze(x, 1)
            elif len(x.shape) == 2:
                ntf = tf.expand_dims(x, axis=0)
            else:
                ntf = x
            if self.l1:
                regularization += self.l1 * tf.reduce_sum(tf.abs(ntf))
            if self.l21:
                regularization += self.l21 * tf.reduce_sum(
                    tf.sqrt(
                        tf.multiply(tf.cast(ntf.shape[2], tf.float32),
                                    tf.reduce_sum(tf.square(ntf), [0, 1]))))
            if self.tl1:
                regularization += self.tl1 * tf.reduce_sum(
                    tf.abs(tf.subtract(ntf[:, :-1, :], ntf[:, 1:, :])))
            return regularization

        def get_config(self):
            return {
                'l1': float(self.l1),
                'l21': float(self.l21),
                'tl1': float(self.tl1)
            }

    def l2_1(l21=0.01):
        '''Group lasso'''
        return TSG(l21=l21)

    def tsc(tl1=0.01):
        '''
        Temporal constrained to preserve the temporal smoothness, for
        activity_regularizer.
        '''
        return TSG(tl1=tl1)

    def sgl(l1=0.01, l21=0.01):
        '''Sparse group lasso, for kernel_regularizer'''
        return TSG(l1=l1, l21=l21)
    def tsgl(l1=0.01, l21=0.01, tl1=0.01):
        '''
        Temporal constrained sparse group lasso, use tsc + sgl instead.
        '''
        return TSG(l1=l1, l21=l21, tl1=tl1)

    def eeg_model(nb_classes, chans, samples):
        """EEGNet."""
        def EEGNet(
            nb_classes, chans=64, samples=128,
            dropout_rate=0.5, kern_length=64, FSLength=16, F1=8,
            D=2, F2=16, norm_rate=0.25
            ):

            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Input
            input = tf.keras.layers.Input(shape=(chans, samples, 1))
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Block 1
            x = tf.keras.layers.Conv2D(
                F1, (1, kern_length), padding='same',
                input_shape=(chans, samples, 1),
                use_bias=False)(input)
            x = tf.keras.layers.BatchNormalization()(x)
            x = tf.keras.layers.DepthwiseConv2D(
                (chans, 1), use_bias=False,
                depth_multiplier=D, depthwise_constraint=max_norm(1.))(x)
            x = tf.keras.layers.BatchNormalization()(x)
            x = tf.keras.layers.Activation('swish')(x)
            x = tf.keras.layers.AveragePooling2D((1, 4))(x)
            x = tf.keras.layers.Dropout(dropout_rate)(x)
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Block 2
            x = tf.keras.layers.SeparableConv2D(
                F2, (1, FSLength), use_bias=False, padding='same')(x)
            x = tf.keras.layers.BatchNormalization()(x)
            x = tf.keras.layers.Activation('swish')(x)
            x = tf.keras.layers.AveragePooling2D((1, 8))(x)
            x = tf.keras.layers.Dropout(dropout_rate)(x)
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Output
            x = tf.keras.layers.Flatten(name='flatten')(x)
            x = tf.keras.layers.Dense(nb_classes, name='dense',
                kernel_constraint=max_norm(norm_rate))(x)
            output = tf.keras.layers.Activation('softmax', name='softmax')(x)
            return tf.keras.models.Model(inputs=input, outputs=output)

        model = EEGNet(
            nb_classes, chans, samples,
            dropout_rate=0.5, kern_length=32, F1=8, D=2, F2=16,
            )
        model.compile(
            loss='sparse_categorical_crossentropy',
            optimizer='adam', #tf.keras.optimizers.Adam(learning_rate=0.0001, amsgrad=True),
            metrics=['accuracy']
        )
        return model

```

```python
    def tsgl_eeg_model(nb_classes, chans, samples):
        """TSGL-EGNet."""
        def TSGLEEGNet(
            nb_classes, chans, samples, colors=1,
            dropout_rate=0.5, kern_length=64, FSLength=16, F1=9,
            D=4, F2=32, l1=1e-4, l21=1e-4, tl1=1e-5, norm_rate=0.25,
            ):
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Input
            input = tf.keras.layers.Input(shape=(chans, samples, colors), dtype=tf.float32)
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Block 1
            x = tf.keras.layers.Conv2D(
                F1, (1, kern_length), padding='same',
                use_bias=False)(input)
            x = tf.keras.layers.BatchNormalization(axis=-1)(x)
            x = tf.keras.layers.DepthwiseConv2D(
                (chans, 1), use_bias=False,
                depth_multiplier=D, depthwise_constraint=max_norm(1.))(x)
            x = tf.keras.layers.BatchNormalization(axis=-1)(x)
            x = tf.keras.layers.Activation('swish')(x)
            x = tf.keras.layers.AveragePooling2D((1, 4))(x)
            x = tf.keras.layers.Dropout(dropout_rate)(x)
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Block 2
            x = tf.keras.layers.Conv2D(F2, (1, FSLength),
                   use_bias=False,
                   padding='same',
                   kernel_regularizer=sgl(l1, l21),
                   activity_regularizer=tsc(tl1))(x)
            x = tf.keras.layers.BatchNormalization(axis=-1)(x)
            x = tf.keras.layers.Activation('swish')(x)
            x = tf.keras.layers.AveragePooling2D((1, 8))(x)
            x = tf.keras.layers.Dropout(dropout_rate)(x)
            # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
            # Output
            x = tf.keras.layers.Flatten(name='flatten')(x)
            x = tf.keras.layers.Dense(nb_classes, name='dense',
                kernel_constraint=max_norm(norm_rate))(x)
            output = tf.keras.layers.Activation('softmax', name='softmax')(x)

            return tf.keras.models.Model(inputs=input, outputs=output)

        model = TSGLEEGNet(
            nb_classes, chans, samples,
            dropout_rate=0.5, kern_length=32, F1=8, D=4, F2=16,
            )
        model.compile(
            loss='sparse_categorical_crossentropy',
            optimizer='adam', #tf.keras.optimizers.Adam(learning_rate=0.0001, amsgrad=True),
            metrics=['accuracy']
        )
        return model
```

C.4 Pipeline

```python
from pipeline import eeg_model, tsgl_eeg_model, conf_matrix, history, Reshape
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'
import seaborn as sns
sns.set_theme('paper', style='dark')

from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
# from scikeras.wrappers import KerasClassifier
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras import utils as np_utils
from sklearn.ensemble import RandomForestClassifier, VotingClassifier, AdaBoostClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.gaussian_process.kernels import RBF, WhiteKernel
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn.cross_decomposition import CCA
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import VarianceThreshold

from pyriemann.utils.viz import plot_embedding
from pyriemann.spatialfilters import Xdawn, AJDC
from pyriemann.estimation import XdawnCovariances, ERPCovariances, Shrinkage, Covariances
from pyriemann.tangentspace import TangentSpace, FGDA
from pyriemann.classification import MDM, TSclassifier, FgMDM
from pyriemann.clustering import Potato, Kmeans
from pyriemann.embedding import SpectralEmbedding
from pyriemann.channelselection import ElectrodeSelection

from mne.decoding import CSP, Scaler, Vectorizer, PSDEstimator, UnsupervisedSpatialFilter
from picard import Picard

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
# Choose model
model = 'Logistic Regression'
standard = {
    'Support Vector Machine':LinearSVC(),
    'Logistic Regression':LogisticRegression(solver='liblinear'),
    'Random Forest':RandomForestClassifier(),
    'Gaussian Process':GaussianProcessClassifier(),
    'Stochastic Gradient Descent':SGDClassifier(),
    'Multi Layer Perceptron':MLPClassifier(),
    'Linear Discriminant Analysis':LinearDiscriminantAnalysis(),
}
nets = {
    'EEGNet':KerasClassifier(eeg_model),
    'TSGL-EEGNet':KerasClassifier(tsgl_eeg_model)
}
```

```
    models = list(standard) + list(nets)
    assert model in models, f'Model must be one of {models}.'
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
    # Choose modality
    epoch_modality = audviz_epochs
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
    # Split data
    y = epoch_modality.events[:, -1]
    nb_classes = len(set(y))
    if nb_classes > 4:
        nb_classes = 3
    y = y % nb_classes
    X = epoch_modality.get_data() * 1e6
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, stratify=y
    )
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
    # Pipeline (Standard Models)
    if model in standard:
        pipe = Pipeline([
            ('pca', UnsupervisedSpatialFilter(PCA())),
            ('xDawnCov', XdawnCovariances()), # out dim 3
            ('TangentSpace', TangentSpace()), # out dim 2
            # ('csp', CSP()),
            # ('var', VarianceThreshold(threshold=0.1)),
            ('scale', StandardScaler()),
            ('clf', standard[model])
        ])
        # Hyperparameters
        param_grid = {}
    # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
    # Pipeline (Neural Networks)
    # Only transform the validation set (Avoid leakage)
    if model in nets:
        pipe = Pipeline([
        ('pca', UnsupervisedSpatialFilter(PCA())),
        ('xDawn', Xdawn()),
        # ('csp', CSP()),
        ('scale', Scaler(scalings='mean')),
        ('reshape', Reshape())
        ])
        # Create Validation Set
        X_train, X_valid, y_train, y_valid = train_test_split(
            X_train, y_train, test_size=0.33, stratify=y_train
        )
        pipe.fit(X_train, y_train)
        X_valid = pipe.transform(X_valid)
        _, chans, samples, _ = X_valid.shape
        # Add Model
        pipe.steps.append(['clf', nets[model]])
        # Hyperparameters
        param_grid = {
        'clf__nb_classes':[nb_classes],
        'clf__chans':[chans],
        'clf__samples':[samples],
        'clf__epochs':[250],
        'clf__batch_size':[16],
        'clf__validation_data':[(X_valid, y_valid)],
```

```
            'clf__verbose':[2],
            'clf__kern_length':[32],
            'clf__F1':[8],
            'clf__F2':[16],
            'clf__D':[2],
            }
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ #
        # Plot the Covariances
        plot_pipe = Pipeline([
            ('pca', UnsupervisedSpatialFilter(PCA())),
            ('xDawn', XdawnCovariances()),
        ])
        covs = plot_pipe.fit(X_train, y_train).transform(X_test)
        #transdict = {0:'Flower', 1:'Guitar', 2:'Penguin'}
        #phoenetic = np.array([transdict[idx] for idx in y_test])
        plot_embedding(covs, y_test, title="Embedding of Covariances")
        # Grid search with cross validation
        # Set n_jobs to 1 if running on GPU
        grid = GridSearchCV(pipe, param_grid, cv=3, n_jobs=3)
        results = grid.fit(X_train, y_train)
        if model in nets:
            hist = pd.DataFrame(results.best_estimator_['clf'].model.history.history)
            history(hist)
        y_pred = grid.predict(X_test)
        acc = accuracy_score(y_pred, y_test)
```