university of groningen

faculty of science and engineering

Revisiting the Pattern Recognition Methods Employed in the Study of Cognitive Processing Stages

Master's Thesis

Bharath Kumar Musuadhi Rajan

August 2022

Supervisor: Dr. Jelmer P. Borst
Artificial Intelligence
University of Groningen, The Netherlands

Table of Contents

Abstract

To perform cognitive tasks, models of the ACT-R cognitive architecture process information in multiple stages, mimicking the different cognitive stages that humans undergo while performing such tasks. For evaluating these models' processing stages, cognitive stages underlying the tasks are first predicted by performing multivariate pattern analysis over the features extracted from the brain data of multiple participants and then compared with the models' stages to constrain them. However, if the techniques used to extract the features from brain data do not consider – 1) the alignment issues incurred while combining participants' neuroimaging data and 2) the nonlinear relation between the experimental task and the brain activity, the resulting features might not provide a proper representation of the neural activities required for the analysis. To investigate these two potential issues, two feature extraction techniques, Multiset Canonical Correlation Analysis (MCCA) and Deep Multiset Canonical Correlation (DMCCA), were applied to the EEG dataset of a recent cognitive study that used Principal Component Analysis (PCA) for extracting the features from the brain data of 26 participants. The techniques were evaluated by performing two multivariate pattern analyses – 1) stimuli classification using the Ridge classifier model and 2) cognitive stage prediction using the Hidden semi-Markov Model (HsMM). The classifier model's sensitivity metric showed MCCA's ability in finding nuanced patterns of the experimental stimuli throughout the entire duration of the trial. And the stage predictions of the HsMM model with high probability using MCCA features confirmed the classifier's result. The classifier model's sensitivity also indicated DMCCA's ability in finding unique patterns of the stimuli that were different from the other two linear techniques confirming the presence of nonlinear patterns in the EEG data. Results from the analyses obtained from the two techniques suggest that further investigations – a) Applying MCCA on high-density EEG data, and 2) Updating the architecture of DMCCA, are required to find better evidence supporting the applicability of the two techniques with respect to the inter-subject alignment and nonlinearity respectively.

Chapter I. Introduction

The success of modern-day artificial intelligence dates back to the 1940s when scientists took inspiration from the functional and structural aspects of the brain to build the first mathematical model of a neuron. This idea of replicating the biological neural network of the brain to build a computational neural network has given rise to state-of-the-art neural network architectures such as Convolutional Neural Networks which take inspiration from the brain's visual cortex and breaks down the complex features of an image similar to how a visual stimulus is processed along the ventral streams of the human brain. Neural Network architectures have enabled us to develop a wide range of AI-based applications in the field of Computer Vision, Natural Language Processing, Machine Translation etc., yet these architectures are limited to the specific tasks in their respective fields, that is, a model trained for image recognition would not perform well in natural language processing and vice versa.

On the other hand, cognitive architectures such as SOAR (Laird et al., 1987) and ACT-R (Anderson, 1993) act as a framework for developing cognitive models and were built to perform a multitude of cognitive tasks in a way a human would perform these tasks. Unlike the development of neural network architectures, which requires only a representative biological neural network model, for once, to draw inspiration while designing the architecture, the development of cognitive architectures requires conducting multiple cognitive experiments to evaluate the architectures using brain data of multiple participants collected from these experiments. But, until recently, the evaluation methods were based solely on the goodness-of-fit to the behavioural data and restricted the potential of cognitive architectures. The cognitive neuroscience revolution in the 1990s gave rise to the new field of Computational Cognitive Neuroscience influencing researchers to include neuroimaging data to evaluate and constrain cognitive architectures (Ashby & Helie, 2011, Anderson, 2007). This approach also required researchers to update their analysis methods from univariate to multivariate pattern analysis (MVPA) to accommodate brain activities from multiple regions (Anderson et al., 2012).

Performing analyses over multiple variables of the brain data is compute-intensive and time-consuming. To alleviate these problems, feature extraction techniques are employed prior to performing MVPA. This additional step in the analysis helps to reduce the number of variables in the brain data without compromising the quality of the original data and the information contained in it. PCA (Pearson, 1901) is one such technique that has been used by researchers to extract features from different types of brain data including fMRI (Anderson & Fincham, 2014), EEG (Borst & Anderson, 2015b), and MEG (Borst et al., 2016). However, there are two key concerns regarding this century old feature extraction technique that could potentially limit the amount of information available in the extracted features and in-turn affect the analyses performed over these features.

1. Inter-Subject Alignment

While combining brain data from multiple participants, the differences in the structural and functional characteristics of the participants' brains, and the positional differences incurred while recording the brain activity, could make the resulting data prone to inter-subject noise (Zhang et al., 2017). If PCA is applied on this combined data, the extracted features might be unsuitable for performing group analysis. For example, Zhang et al. (2017) applied MCCA, a technique that aligns participants' brain data before extracting the features, to the MEG data of 18 participants performing a cognitive task and showed that the results from stimuli classification analysis were comparatively better than PCA.

2. Nonlinearity

The second key consideration that has to be taken into account before applying PCA to brain data is its inability in finding nonlinear patterns. Using the multiple variables of brain imaging data for evaluating cognitive architecture implies that we are dealing with the data that directly reflects the complex nature of the brain dynamics (Stam, 2005). So, if there is a nonlinear relation between the brain activity and the experimental task/stimulus, linear techniques such as PCA will not be able to recognize the nonlinear patterns present in the brain data, and the extracted features might not be a complete representation of the brain activities.

In the current study, these two potential issues were investigated by applying two feature extraction techniques, MCCA and DMCCA, to the EEG data of 26 participants performing a Lexical Decision Task (LDT) of a cognitive experiment conducted by Berberyan et al., (2021). DMCCA is a deep learning variant of the multiset canonical component analysis that preserves the nonlinearity in the individual sets while supporting multiset data. The results from the two techniques were then compared using two multivariate pattern analyses techniques. Better results from the analyses performed over MCCA features would suggest that the alignment issue has to be addressed before performing similar inter-subject analyses on data obtained via EEG recording, and better results from DMCCA features would substantiate the presence of a nonlinear relation between the experimental stimulus and the brain response in lexical decision tasks, advocating the use of nonlinear analyses for similar studies.

The next section provides an overview of the ACT-R cognitive architecture and the value-addition of neuroimaging data in evaluating such an architecture.

1.1 ACT-R Cognitive Architecture

Described as "*a theory for simulating and understanding human cognition*" by the author, Adaptive Control of Thought-Rational (ACT-R) is a cognitive architecture proposed by Anderson (1993) that was built on the strong foundations of cognitive theories to be able to perform a wide range of cognitive tasks. The framework of this architecture consists of multiple modules to replicate the different basic functionalities of the human brain involved while performing a cognitive task. For example, ACT-R 4.0 (Anderson & Lebiere, 1998) included modules like Declarative Memory, Procedural Memory, Perception and Action (see Fig. 1). The actions required to complete a particular task are not only designed to be executed by these specific modules but were also made to go through a series of stages (Conflict Resolution, Declarative Retrieval and Production Execution) to mimic the cognitive processing stages that are assumed to be undergone by humans while performing a cognitive task. Until ACT-R 5.0, the architecture was evaluated by comparing the behavioural data of participants performing a specific cognitive task with the model's performance obtained from the respective cognitive task. But evaluating an architecture that was designed to imitate the complex functionalities of the brain using only a few factors such as response time and accuracy, failed to properly constrain the architecture (Borst et al., 2015).

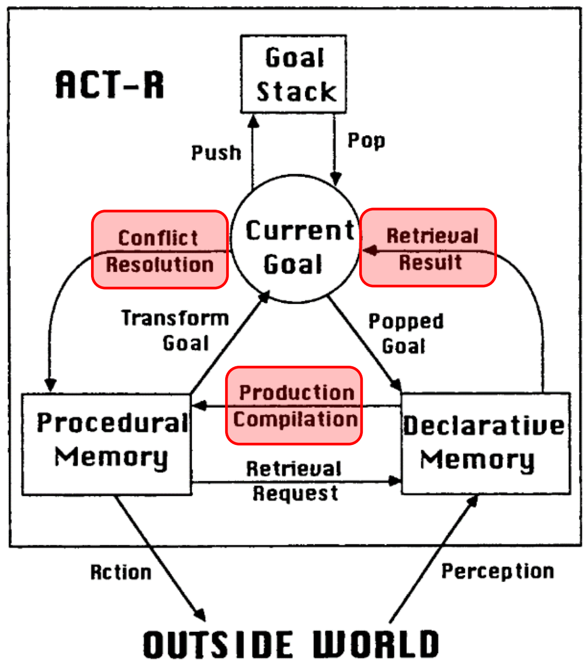


Figure 1: Different modules of the ACT-R 4.0 architecture (Anderson & Lebiere, 1998) with the stages of the architecture being highlighted in red. The modules and stages of the architecture until this version were evaluated solely based on the behavioural data.

1.1.1 Evaluation of ACT-R using Neuroimaging Data

Advancements in neuroimaging techniques and easier accessibility of neuroimaging devices enabled researchers to include brain imaging data for evaluating the ACT-R cognitive architecture. This additional constraint of including neuroimaging data in evaluating the cognitive architecture limits the possible theories involved in designing the different modules and stages and helps in converging the research focus to build a robust cognitive architecture. For example, Anderson et al., (2003) used fMRI data of participants performing two cognitive tasks, mapped the modules of the ACT-R model to the corresponding brain regions, and evaluated the modules by comparing the model activations with the activations seen in the corresponding brain regions. Fig. 2 shows an illustration of the mapping of ACT-R 6.0 modules based on their functionalities to the corresponding brain regions as seen in Borst & Anderson, (2015). Apart from this spatial constraint of the model based on the brain regions, the processing stages of the model were also evaluated to temporally constrain the ACT-R architecture.

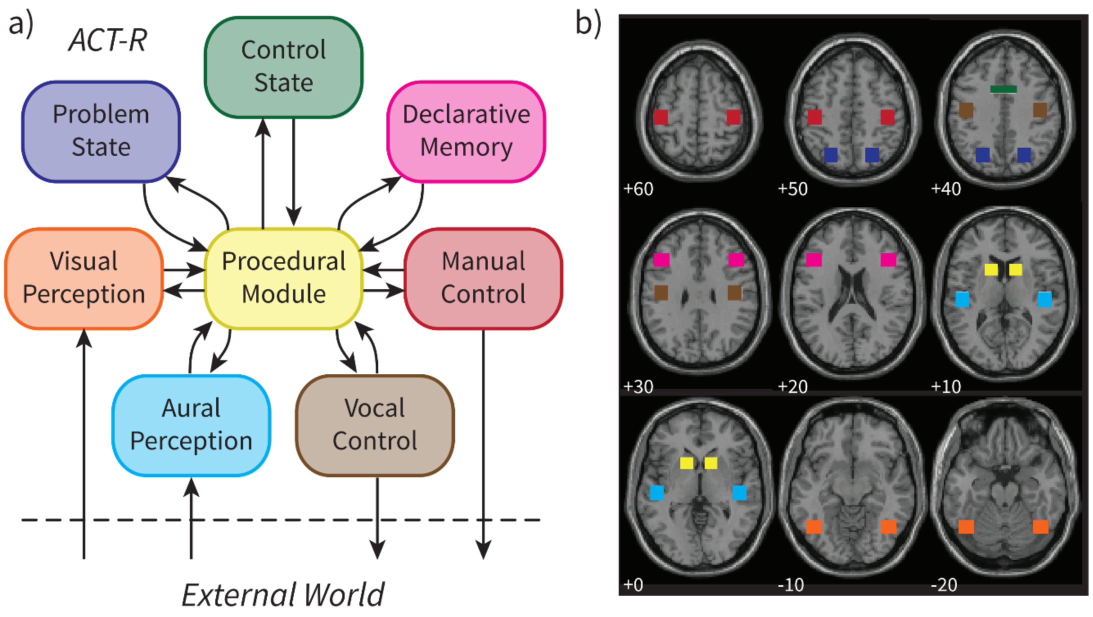


Figure 2: a) ACT-R 6.0 architecture with its modules updated based on fMRI based analysis. b) Mappings of the ACT-R model in the brain regions of an fMRI scan (Borst & Anderson, 2015a).

Decoding Brain via Multi-Variate Pattern Analysis

For further evaluating the ACT-R model's processing stages, Anderson et al., (2012) performed a Hidden semi-Markov Model - Multi-Variate Pattern Analysis (*HsMM-MVPA*) over the multiple voxels of fMRI data collected from students working on an intelligent tutoring system, to decode their brain data and predict the cognitive states based on the activity patterns observed in the fMRI data. The poor temporal resolution of fMRI data limited the evaluation of the model stages to tasks that occur in the order of seconds. To overcome this limitation, Borst et al., (2013) applied a *Ridge Classifier* model over the multiple channels of EEG data collected from participants performing an associative recognition task, to find *activity patterns related to the experimental stimuli* at different time windows of the trials. Based on the analysis of the classification results from each time window, the authors were able to identify the cognitive processing stages of the experimental task in the order of milliseconds. Making further utilization of this high temporal resolution of EEG data, Borst & Anderson (2015a) performed a HsMM-MVPA to find *stimuli-independent activity patterns* in single-trial EEG data and predicted the cognitive processing stages in this associate recognition task (see Fig. 3). In their following research, Anderson et al., (2016) performed an extension of the HsMM-MVPA to find *activity patterns based on two kernel functions*. Recently, Berberyan et al., (2021) applied this HsMM-MVPA method in a language experiment to identify cognitive processing stages involved in a lexical decision task.

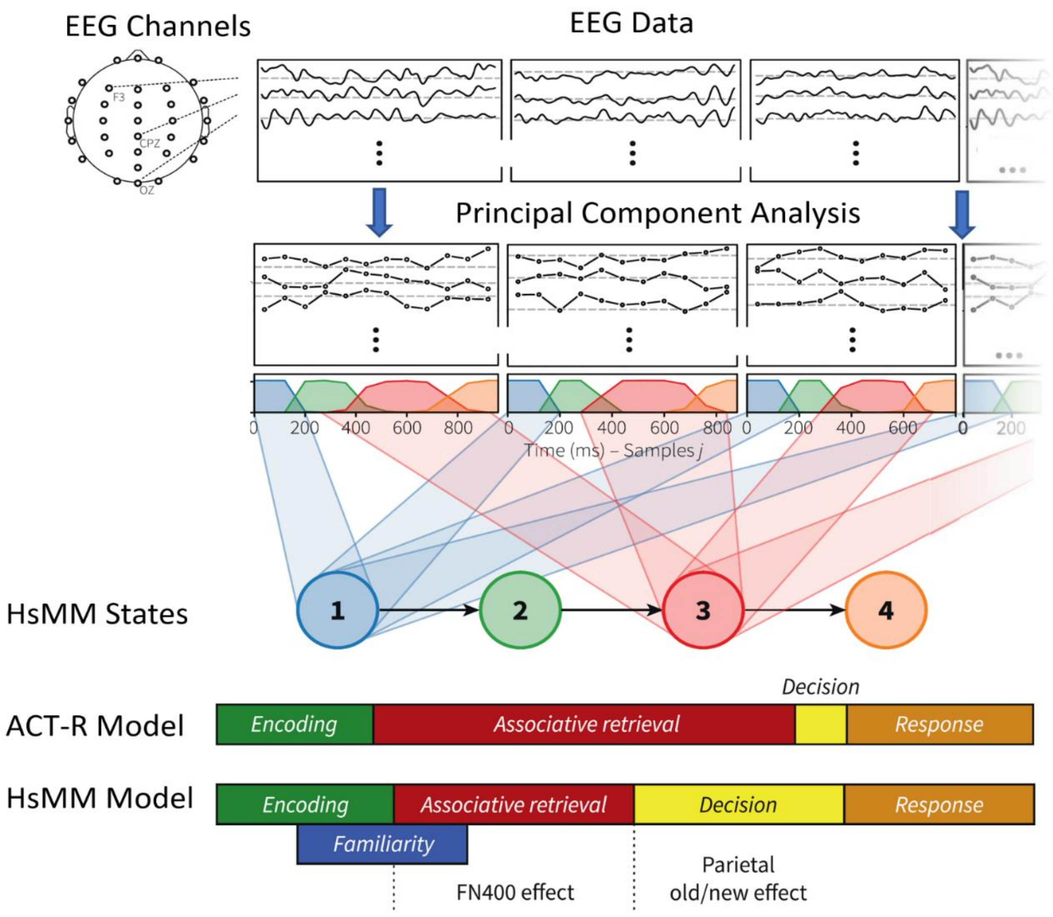


Figure 3: Stages of the ACT-R model being compared with the cognitive processing stages (referred to as HsMM States) predicted via HsMM-MVPA method implemented by Borst & Anderson, 2015.

The remainder of this report is as follows: The next chapter provides a theoretical background of the feature extraction techniques and the Multi-Variate Pattern Analyses required for this project; followed by the 'Methods' chapter with the details of the three feature extraction techniques and the two Multi-Variate Pattern Analyses implemented in this project using the EEG data from the Lexical Decision Task (LDT) the results of the analyses performed on the features extracted from the three techniques are presented in the 'Results' chapter, followed by its discussion, and the 'Conclusion' chapter.

Chapter II. Theoretical Background

In this chapter, the first section provides an intuition behind the dimensionality reduction of M/EEG data, followed by a theoretical background for Principal Component Analysis, Canonical Component Analysis, Generalized Canonical Component Analysis, and Multilayer Nonlinear Canonical Correlation Analysis. The second section provides the theoretical background for the Ridge Classifier model and the HsMM-MVPA model. For conceptual coherence, the different techniques are explained using a single overall theme of maximizing the correlation between two vectors, the predictor, and the criterion. Criterion is the dependent variable, $\boldsymbol{y}$, related to the independent variables, $\boldsymbol{X}$, by some function, $\boldsymbol{f(X)}$, and the predictor, $\boldsymbol{\hat{y}}$, is a statistical model defined by some hypothesis function, $\boldsymbol{h(X)}$. Fitting the model to match the data distribution pattern observed in the independent variables, $\boldsymbol{X}$, is the process of maximizing the correlation between these two vectors, $\boldsymbol{y}$, and $\boldsymbol{\hat{y}}$, for finding the coefficients of the hypothesis function, $\boldsymbol{h(X)}$. Coefficients of this hypothesis function can then be used to transform the existing data to a new feature space (in case of feature extraction) or applied to unseen data for classifying stimuli information or applied to the channel activities for predicting cognitive stages.

2.1 Dimensionality Reduction in M/EEG data

The purpose of doing feature selection before an analysis is to find the relevant features from a dataset and thereby reduce the dimensions of the original dataset. The dataset with these newly obtained feature-rich variables helps to improve the quality (due to the omission of irrelevant features) and computing efficiency (due to reduced dimensions) of the analyses.

Note: In the examples of M/EEG datasets used in this report, the ***rows*** indicate the ***observations*** made at a particular time (discrete time points when the sensor/electrode values are sampled), and the ***columns*** primarily represent the ***channels*** of the recording device but are also referred to as 'dimensions', 'independent variables' and 'feature vectors' based on the context. Hence, each cell of the M/EEG dataset represents a digital sample of a sensor's/electrode's value sampled at the corresponding time point, referred to as ***data points***.

Consider an inordinately simplified EEG dataset (see Fig. 4 Left) obtained as a result of recording the values (and averaged over a specific time duration) from 2 electrodes placed at two different parts of the participant's scalp during an experiment that was repeated for 10 trials with 2 different stimuli. We can understand the significance of feature selection using an example of a stimulus classification analysis. For the given data, if we assume only one of the regions is activated for the stimuli, that is, either one of the electrodes is influenced by the stimuli, then we can use the data points of each electrode to represent the stimuli (represented as green and red dots) in a one-dimensional plot as seen in Fig. 4 Centre. If we consider that the stimuli influence both the electrodes, then we can represent the stimuli as a combination of the data points from the corresponding rows of the electrodes **E1** and **E2** via a 2-D scatter plot as seen in Fig. 4 (Right).

From the 1D plots, it can be seen that the stimuli can be classified based on the pattern (positive/negative) observed solely from the electrode E2. And from the 2D plot, it could also be seen that the data provided by the E1 electrode acts as noise rather than a relevant feature for the classification analysis performed in this example. Hence, by selecting only the E2 electrode, we would be able to reduce the dimension of the data, and at the same time have relevant noise-free information for the classification analysis.

Trial	Stim	E1	E2
x1	y1	0.3	0.2
x2	y2	0.1	-0.1
x3	y2	0.1	-0.8
x4	y1	0.2	0.1
x5	y2	0.4	-0.2
x6	y1	0.2	0.6
x7	y1	0.1	0.4
x8	y2	0.3	-0.3
x9	y1	0.1	0.8
x10	y2	0.1	-0.5

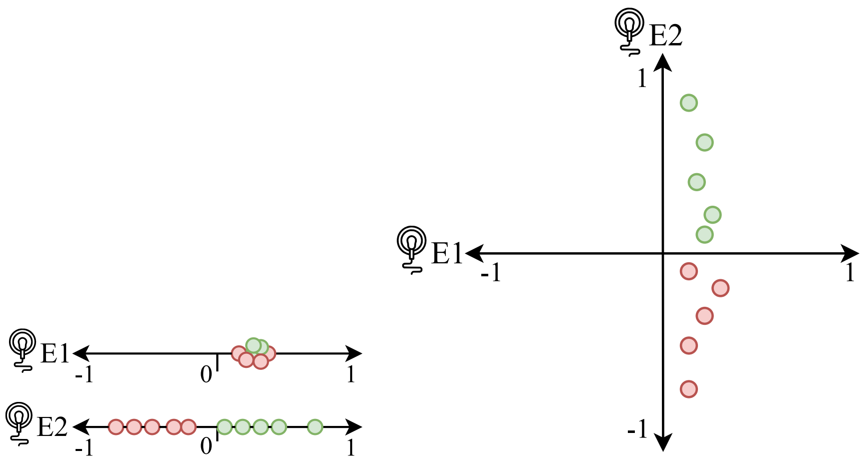


Figure 4: Left – Example EEG data with electrode activities for the corresponding stimulus. **Centre:** Distribution of the electrode values by treating each channel as an individual coordinate vector of its own variable space. **Right:** A 2-dimensional scatter plot representing the common *variable space* formed by the two variables, E1 and E2, of the EEG data.

This forms the basic idea behind feature selection: selecting the features that provide meaningful information and avoiding the features that could either be less significant and/or could act as noise in the analyses to be performed. For analyses in neuroscience experiments, this is equivalent to the selection of channels representing brain areas covering specific regions of interest. For example, to analyse the specific brain activities related to visual stimuli, channels representing regions other than the occipital lobe can be dropped to remove their interference in the analysis and to speed up the computation process.

However, selecting electrodes based on known regions of interest might not be a viable choice for all kinds of studies. For example, in the cognitive stage analyses mentioned earlier, one of the objectives itself is to identify the brain regions responsible for the corresponding stages. Hence, analyses have to be performed on all channels to find the regions of interest. To reduce the dimensions of M/EEG data for these kinds of analyses, feature extraction techniques such as Principal Component Analysis (PCA) can be employed. Unlike feature selection, feature extraction techniques create a new set of variables from the original dataset based on the relevant information from all channels. Analyses can then be performed on the top variables that explain the maximum information, allowing us to achieve maximum quality and efficiency in the analyses.

2.1.1 Principal Component Analysis

The motivation behind PCA (Pearson, 1901) is that before using a set of independent variables to make a prediction of a dependent variable, it is preferable to correct the errors present in the independent variables. The errors, here, refer to the potential correlation or dependency between the independent variables. Given a data distribution comprised of erroneous independent variables, PCA determines a new set of vectors that best fit the data in the variable space of the original data that are orthogonal to each other. Hence, the vectors obtained via this process are uncorrelated/independent and represent a unique pattern observed in the data. The process can be seen as a form of multiple regression. In multiple regression, the independent variables are combined together, acting as input for the regression model, to predict an external dependent variable. In PCA, the independent variables from within the input space are combined together to predict each other resulting in error-free independent principal components. When applied to the M/EEG data, PCA determines a new set of principal components from the sensor/electrode space. Selecting the top components would allow for creating a dataset with a) the most relevant features, b) reduced dimensions, and c) features that are less susceptible to noise. This section provides an intuition behind PCA using a stimuli classification example. The process involved in performing principal component analysis via eigendecomposition can be found in the 'Methods' chapter.

Consider another dataset as seen in Fig. 5 Left. Similar to the previous example, if we now represent the stimuli in a one-dimensional plot (see Fig. 5 Centre), there is no distinct pattern in either of the electrodes. And from the two-dimensional plot (Fig. 5 Right), it can be seen that the variance observed in the data, explaining the channel activities related to the stimuli, is now dependent on the values from both electrodes, implying that running a classification analysis using either one of the electrodes would not provide a meaningful result as it requires both the electrodes. But the plot also suggests that there is a correlation/dependency between the two electrodes i.e., as the values in E1 increase, the values in E2 also increase.

Trial	Stim	E1	E2
x1	y1	0.6	0.7
x2	y2	0.3	0.6
x3	y2	0.1	-0.4
x4	y1	0.4	0.3
x5	y2	-0.3	0.1
x6	y1	0.2	-0.1
x7	y1	-0.4	-0.3
x8	y2	-0.2	-0.6
x9	y1	-0.1	0.4
x10	y2	-0.8	-0.8

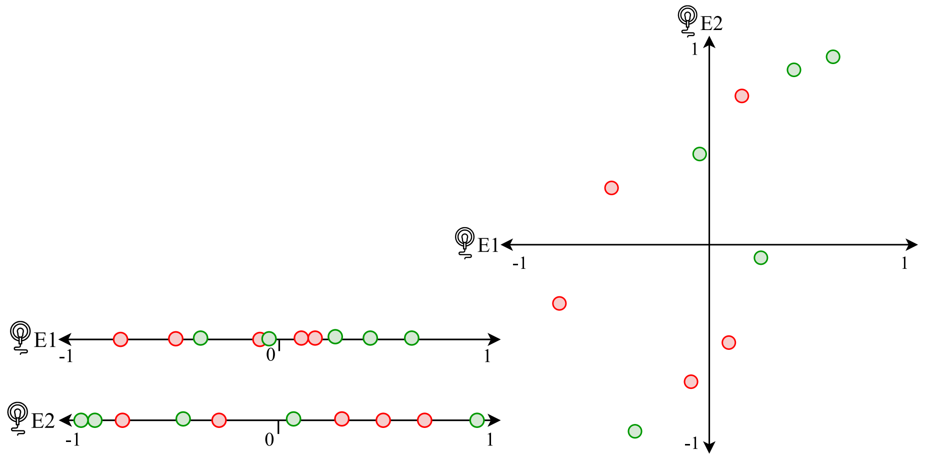


Figure 5: Example EEG data demonstrating the distribution of correlated data between two different electrodes.

If we consider electrode ***E1*** as the independent variable, $\boldsymbol{x}$, electrode ***E2*** as the dependent variable, $\boldsymbol{y}$, and assume both variables are related by a linear function $\boldsymbol{y = f(x) = mx + c}$, then using a statistical model, $\boldsymbol{\hat{y}}$, with hypothesis function, $\boldsymbol{h(x) = \alpha x + b}$, we can fit the data to find the best fitting model by maximizing the correlation between $\boldsymbol{y}$ and $\boldsymbol{\hat{y}}$. The vector obtained from the best-fitting model would point in the direction of the maximum variance observed in the data. And the new variable obtained by projecting the original data points onto this vector could now be used as the only feature in our classification analysis to distinguish the two stimuli (See Fig. 6 Top Left). This new variable containing the maximum relevant information of the two electrodes is the first principal component (***P1***), and the variable obtained from the second best-fitting model that is orthogonal to the first, gives us the second principal component (***P2***).

Real-world M/EEG data with multiple channels will be spread across multiple dimensions, and there would be several strong patterns indicating the directions of the maximal variances observed in the data, resulting in more than one best-fitting model. We determine all of the orthogonal vectors that best fit the data, rank them in the order of their explained variance, select the top $\boldsymbol{p}$ vectors and calculate the principal components required for the new dataset.

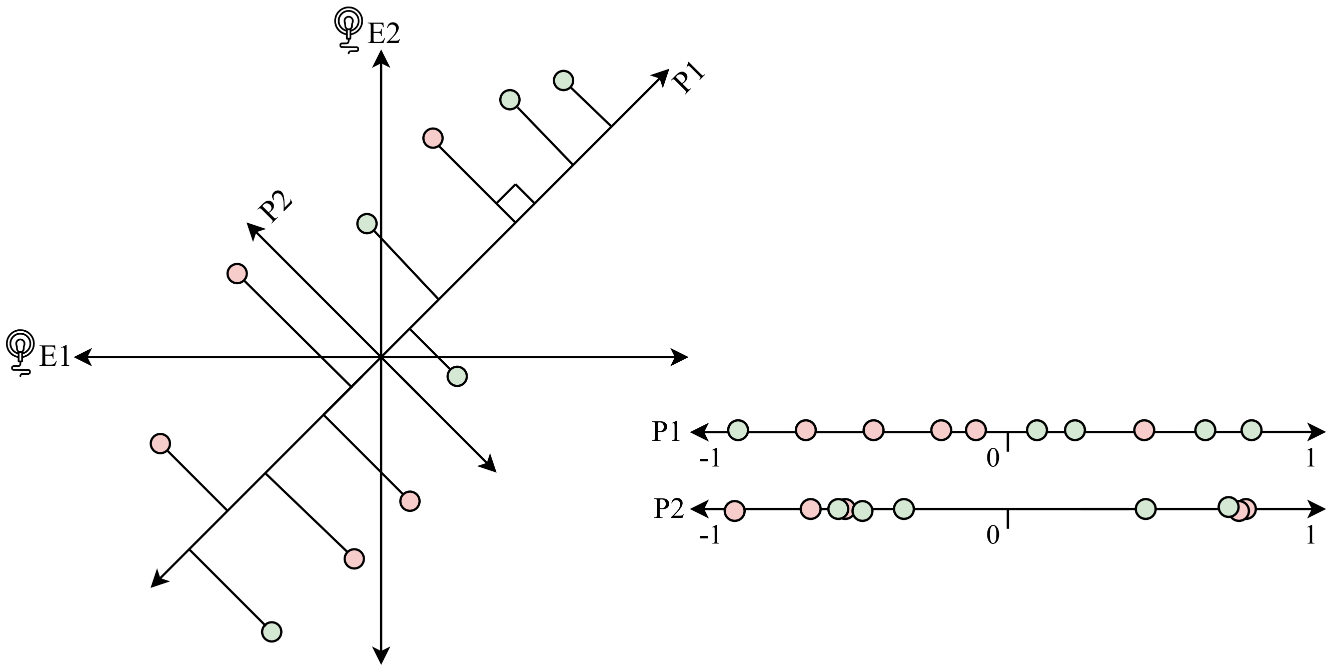


Figure 6: Left: An illustration of Principal Component Analysis in the context of its applicability in a supervised classification problem. **Right:** 1D plot showing the distribution of newly obtained data points in the individual principal component's vector space.

Note: The distinction between the data points based on stimuli (red and green) is made for illustrative purposes. PCA itself is an *unsupervised learning algorithm*. The process remains the same regardless of the presence of stimuli information, and the algorithm finds the maximum variances in the data.

Limitation: Inter-subject Alignment

When performing inter-subject analyses, if we have to extract the features from the brain data of all the participants, we could apply PCA either on the combined data of the participants (*Across-Subjects PCA*) or apply the algorithm on each participant's data separately and then combine the data for analysis (*Within-Subjects PCA*). An intuition behind how either of these approaches could be suboptimal in the presence of inter-subject alignment issues can be seen as follows.

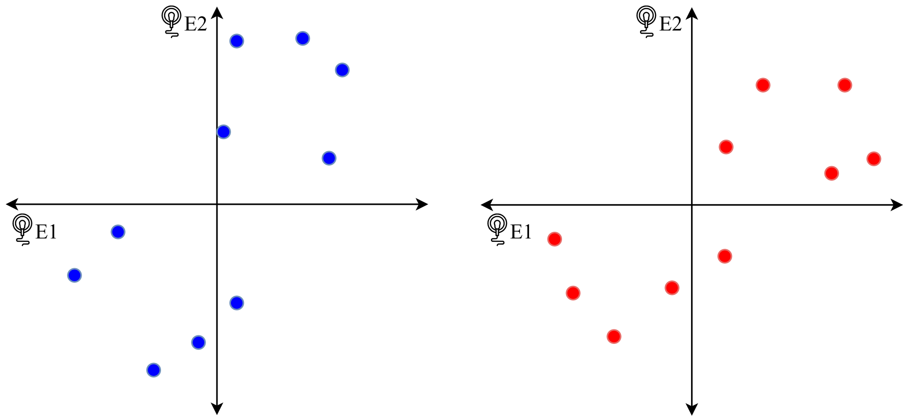


Figure 7: An illustration of the inter-subject alignment issue in the data distribution pattern of EEG data collected from 2 participants.

Consider a similar example dataset as seen in the previous section, but with additional data collected from another participant (Fig. 7). In the *Within-Subjects* approach (Fig. 8 Left), principal components obtained from each subject individually would be free from the intra-subject noise, but they would be uncorrelated and not aligned in the common subject space, making the new dataset prone to the inter-subject noise. The complication involved in applying PCA across subjects (Fig. 8 Right) is that, although it results in a seemingly noise-free dataset (with uncorrelated variables), the principal components themselves would be obtained from overly noisy (accumulated over multiple subjects' intra-subject noise) data. So, the lower the signal-noise ratio of the brain data, the poorer will be the quality of the principal components. An alternative technique that addresses this correlation issue is Canonical Correlation Analysis (CCA).

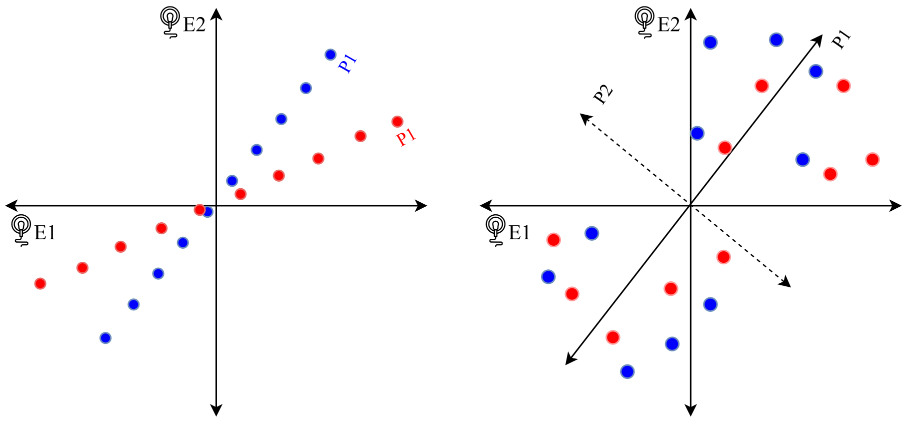


Figure 8: An illustration of *Within-Subjects PCA* and *Across-Subjects PCA*. **Left:** Shows the inter-subject noise after combining the PCs extracted individually from two participants. **Right:** Shows the increase in intra-subject noise (i.e., increased variance in the secondary direction of the data) of the combined data prior to applying PCA.

2.1.2 Canonical Correlation Analysis

CCA (Hotelling, 1936) is a technique that can be used to find the features that are maximally correlated between two sets of data variables. It can be seen as a generalization of the PCA to two sets of data but with the objective of increasing the correlation between the linear best fits of the two datasets, or as an extension of multiple regression analysis in which the variables of the first dataset are treated as the predictors and are linearly combined to predict the linear best fits of the criterion variables in the second, and vice versa. This objective makes CCA slightly different from PCA conceptually and computationally.

CCA can be intuitively understood as the process of determining the canonical variates (linear combinations of the original data variables) from two subjects' M/EEG data and projecting them onto each other such that the canonical angle between the projected and the native canonical variate is minimal (see Fig. 9). In other words, the objective is to maximize the cosine of the angle representing the correlation between the canonical variates (See Appendix 6.1 for details). CCA tries to find all such best-fitting linear combinations from each dataset that are maximally correlated between the two datasets.

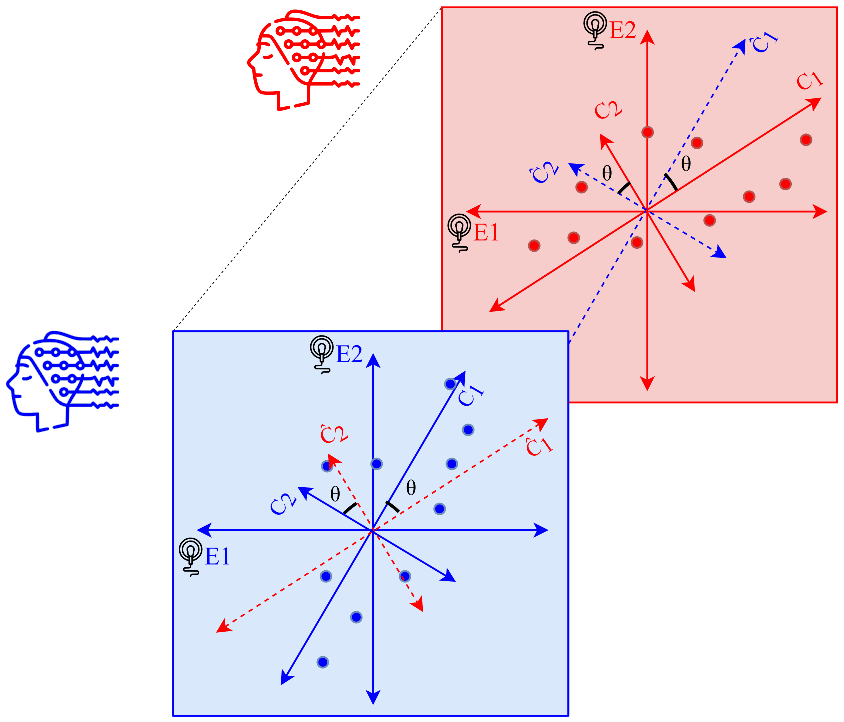


Figure 9: An illustration of CCA. Projecting the canonical variates obtained from two subjects' data onto each other to minimize their canonical angle.

The computational difference to PCA is that the factorization is performed on the cross-correlation matrix of the two datasets rather than the auto-covariance matrix of a single dataset. While the maximum number of components determined by the PCA algorithm is equal to the number of variables in the original dataset, the maximum number of canonical components obtained from the CCA algorithm is equal to the least number of variables among the two original datasets.

Limitation: Intra-subject correlation

In the process of maximizing the inter-subject correlation of the linear combinations of the two datasets, the CCA algorithm disregards the correlation that exists between the canonical components within a subject's data, in other words, the CCA algorithm does not include the orthogonality constraint. Since the determined canonical components need not be uncorrelated within the dataset, this method restricts the canonical variates of a dataset from maximizing and explaining all the unique variances observed within a subject's data. Hence the CCA method, unlike PCA, is not a lossless process and the components obtained via this method are still prone to within-subject noise. It should also be noted that the original CCA algorithm is constrained to only two sets of data.

2.1.3 Generalized Canonical Correlation Analysis

Generalized variants of CCA (see Horst, (1961) for the different variants) allow one to apply CCA to more than two datasets and also provide an optimization solution that adds the orthogonality constraint to retain the maximum variance within a dataset while finding features that correlate between multiple datasets. In this method, instead of projecting the calculated canonical variates between each pair of datasets, the canonical variates are projected onto a common space shared between multiple datasets. And from this view independent representation, the correlations between the canonical variates are maximized. Furthermore, the canonical variates themselves are not a combination of the original variables, but a linear combination of the principal components obtained by performing PCA on the individual datasets. In this section, an intuition behind the second model of Horst, (1961) is provided. A detailed description of the implementation of this technique applied over the M/EEG data (referred to as Multiset Canonical Correlation Analysis, Zhang et al., (2017)) using the optimization solution provided by Via et al., (2007) can be seen in the Methods chapter.

Consider the datasets as seen in Fig. 10. The first step in GCCA involves the extraction of the principal components of the M/EEG data from the channel space of each subject using PCA. Next, in the principal component space of each subject (similar to the electrode space, but using the principal components as coordinate vectors), a linear best fit is calculated. The newly obtained canonical variates from each subject are then projected onto the subject space, a common representational space formed by treating the canonical variates from each subject as its coordinate vectors. In this subject space, the correlation between the canonical variates is maximized. The transformed canonical variates are projected back to the PCA space of the corresponding subjects, and the data points from the principal components of each subject are projected onto the individual subject's canonical variate to obtain the first canonical component. This process is repeated $\boldsymbol{n}$ times, where $\boldsymbol{n}$ denotes the number of variables in the set with the least number of variables.

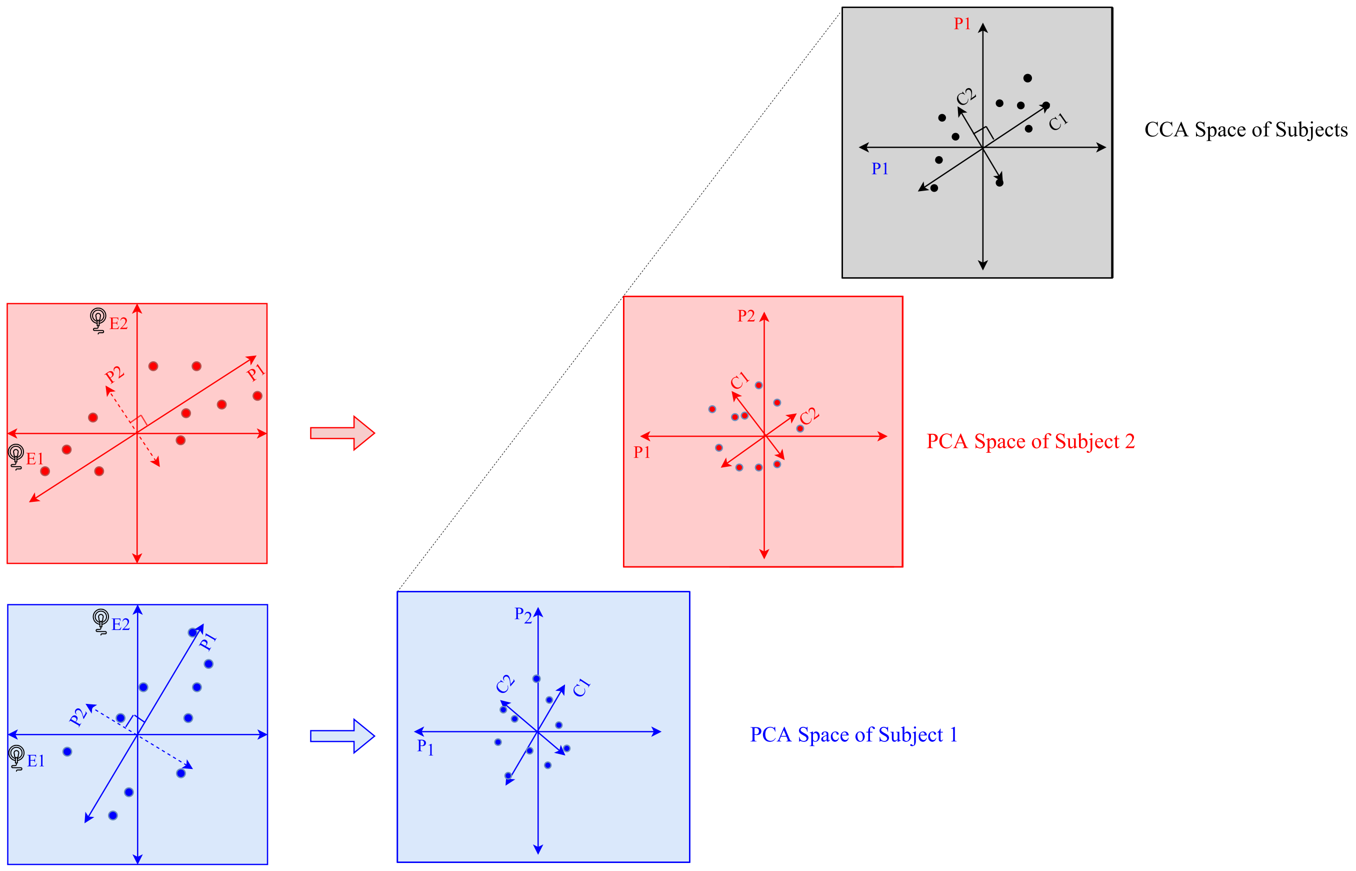


Figure 10: An illustration of GCCA using EEG data from two subjects (indicated in red and blue). **Left:** Extraction of principal components from the M/EEG data of each subject. **Right:** Canonical variates are calculated from the PCA space of the subjects. **Right Top:** Canonical variates from each subject are projected onto the subject space and their correlation is maximized in this common representational space.

Limitation: Linearity

PCA and GCCA, when applied to M/EEG data, extracts features that provide a fair representation of the original data with reduced dimensions when there is a linear relationship between the experimental task/stimulus and its response (brain activity), that is when the experimental task/stimulus could be defined using a linear combination of the electrodes representing the brain activities from a different region. But if we were to assume a nonlinear relation between the channel activities, these techniques will not be able to explain the information contained in the brain data comprehensively.

Consider a dataset similar to the previous sections but with the assumption that the data points explaining the brain activity for the presented stimulus have a nonlinear relationship in the channel space (see Fig. 11). Fitting a linear model to this data using PCA/GCCA would result in the components having a poor goodness-of-fit to the data, explaining only partial variance of the data. An alternative solution for extracting the features in these scenarios is to apply nonlinear models to fit the data.

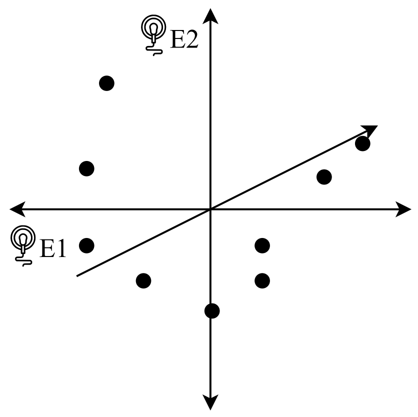


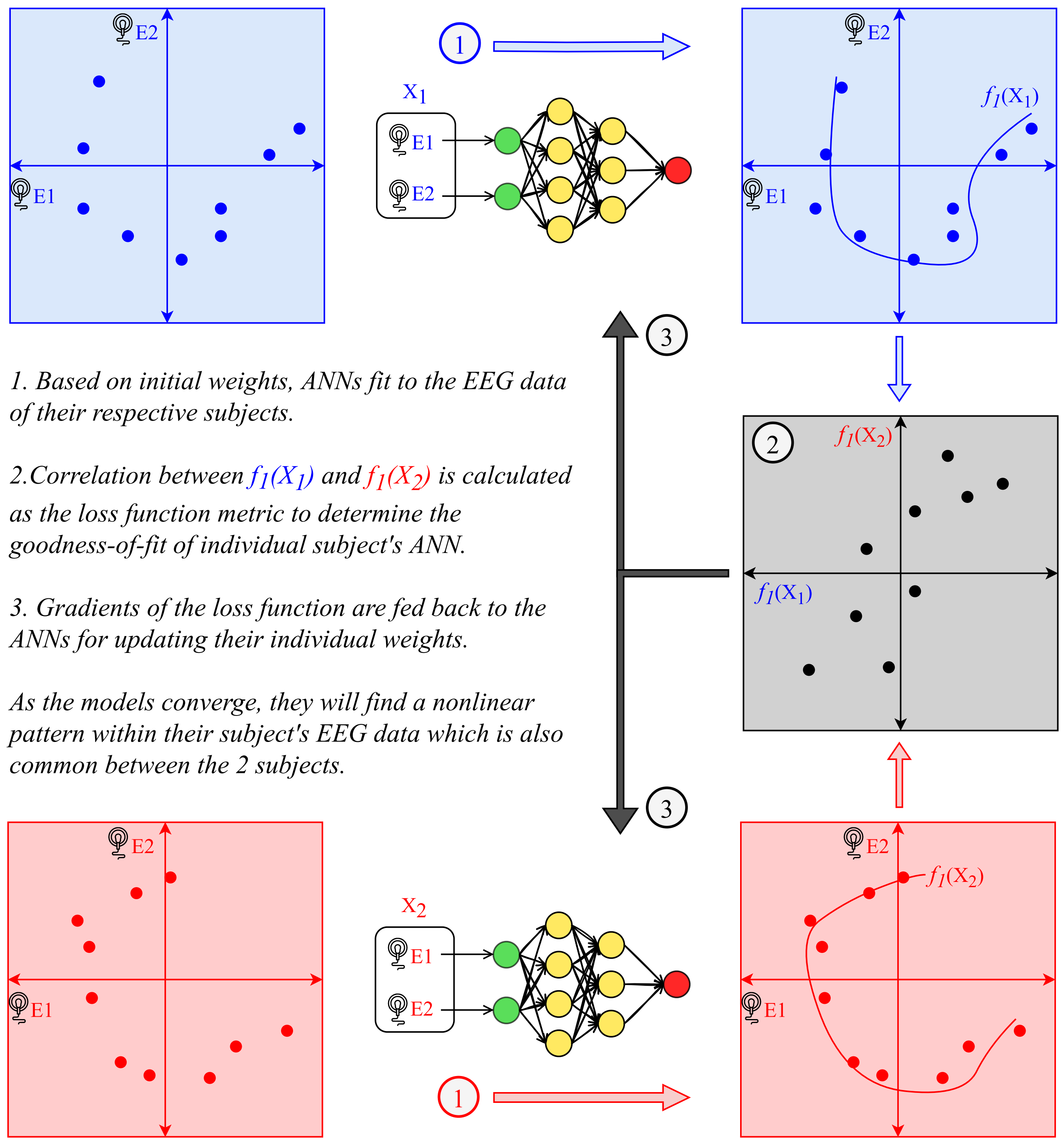
Figure 11: Fitting a linear model to a data distribution that has a nonlinear pattern.

2.1.4 Multilayer Nonlinear Canonical Correlation Analysis

Utilizing the potential of Artificial Neural Networks (ANN) for finding nonlinear patterns, Becker & Hinton (1992) designed an ANN architecture with ***two*** individual neural network modules[1] to train ***two*** adjacent patches of an image. The weights of these networks were optimized such that both the modules learnt the same features (common aspects of the image) from the two patches. The optimization solution used for updating the weights of their models is similar to the solution seen in the GCCA methods, that is, maximizing the correlation of the feature vectors (canonical variates) obtained from the two patches. A description of this architecture in the context of its ability to find nonlinear patterns in individual subject data is provided in this section. An overview of the architectural design implemented in this project can be found in the Methods chapter.

In the previous section (2.1.3 - Limitation), we saw that by fitting a linear model to the data having a nonlinear pattern, we fail to capture the maximum variance of the original data. If the nonlinear pattern formed by the variables is known to an extent, then we would be able to apply kernel methods to find a nonlinear fit of the data. But, when there is no prior knowledge of the data or possible patterns formed by the multiple variables, we could apply ANNs to fit the data. ANNs use nonlinear functions such as sigmoid and hyperbolic tangent, to activate the nodes in the network's hidden layers (also referred to as dense layers) to fit the data. Depending on the type of pattern spanned in the multidimensional space, the individual nodes in the hidden layer either activate or deactivate to form a nonlinear combination of these activation functions that match the pattern of the original data.

[1] The authors also implemented a network architecture that consists of 10 modules in their following work (See Becker & Hinton, 1992 for details).



1. Based on initial weights, ANNs fit to the EEG data of their respective subjects.

2.Correlation between $f_1(X_1)$ and $f_1(X_2)$ is calculated as the loss function metric to determine the goodness-of-fit of individual subject's ANN.

3. Gradients of the loss function are fed back to the ANNs for updating their individual weights.

As the models converge, they will find a nonlinear pattern within their subject's EEG data which is also common between the 2 subjects.

Figure 12: An illustration of neural networks finding nonlinear patterns in individual subjects' data. **Left:** Distribution of data from 2 subjects having a nonlinear pattern in the electrode space. **Centre:** ANNs of individual subjects with input, dense, and output layers indicated by green, yellow, and red circles respectively. Red and Blue arrows indicate transformation of individual subject's data, and grey arrow indicates the weight update based on the inter-subject correlation. **Right:** Model outputs $\boldsymbol{f_1(X_1)}$ and $\boldsymbol{f_1(X_2)}$ fitting the nonlinear pattern of each individual subject's data.

Consider a distribution of data from two subjects as seen in Fig. 12 (Left). If we take the electrode values of a subject's EEG data as $\boldsymbol{X}$, and the function defining the nonlinear pattern observed in the data as $\boldsymbol{y} = \boldsymbol{f(x)}$, then, a neural network model, $\hat{\boldsymbol{y}}$, with a combination of activation functions as its hypothesis function, $\boldsymbol{h(x)}$, can be fitted to match the nonlinear data pattern. During every iteration of the training process, the correlation between vectors $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ is calculated, and the weights (coefficients of activation functions) are updated to maximize their correlation. As the model approaches the shape of the EEG data, their correlation will be at its maximum, and the weights of this hypothesis function can be then used to transform the original data to create a new variable $\boldsymbol{f_1(X)}$. The new variable $\boldsymbol{f_1(X)}$ is akin to principal component $\boldsymbol{P1}$ seen in section 2.1.2, but explains the *nonlinear pattern* of the individual subject data. This is the working mechanism of a regular ANN that could be applied on an individual subject's EEG data. If we include another separate neural network model to process the second subject's EEG data, and update the weights such that the correlation between the $\boldsymbol{f_1(X)}$ obtained from the two subjects is maximum, then the weights of individual subjects' $\boldsymbol{f_1(X)}$ can then be used to transform the original data of the corresponding subjects. This new variable explains the nonlinear pattern that is common across subjects.

Limitation:

Overfitting of Original Data: The nonlinear functions used in the neural networks are not restricted to any particular function like the fixed transformation functions used in the kernel variants of PCA. This flexibility tends to make the model tend to fit all the data points, which could result in overfitting, that is, while PCA gives more weightage to the data points falling along the direction of maximum variance and less priority to the noisy data points deviating from the principal direction, the neural network's nonlinear functions could bend the fit towards the data points with less significance and/or include the noise present in the original data. This in turn could affect the process of maximizing the correlation between the subjects' canonical variates in the common representational space during the CCA optimization step.

The Nonlinearity of Canonical Variables: The intuition behind this limitation is the same as seen in the GCCA section. Though it captures the nonlinear pattern present in the original data, while trying to find the correlation among the canonical variates, it still uses the linear GCCA function to optimize the original weights. And if the canonical variates between the subjects are related by a nonlinear function, then the optimization solution would not be able to find a proper nonlinear transfer function.

2.2 Multivariate Pattern Analysis

Referring to their 2001 study on finding the brain activity patterns of object recognition by performing statistical analysis over multiple voxels of fMRI data, Haxby et al., (2012) coined the term Multivariate Pattern Analyses (MVPA) for a generic representation of the usage of multiple variables in neuroimaging analysis. For example, Peters et al., (1998) used a Laplace Filter and an Autoregressive model to extract 30 features based on the spatio-temporal pattern from 56-channel electrode data, and then applied 30 individual ANNs on these extracted features to classify the finger movement of the participants performing the experiment. Hence the 'variables' in MVPA differ depending on the type of brain data being used for the analysis including voxels in fMRI, electrodes in EEG, and sensors in MEG. The increase in computing power and the development of sophisticated analysis algorithms in recent years, when combined with MVPA, enables us to gain a better in-depth understanding of the neural activities observed across brain regions.

Decoding of brain activity using MVPA includes both – supervised learning (for example, predicting the stimulus from the neuroimaging data), and unsupervised learning (for example, finding patterns in the brain activities of an experimental task). The features extracted using the techniques implemented in this project were evaluated by models belonging to both categories. The Ridge classifier method, previously implemented by Borst et al., (2013) in their study on cognitive stages comes under the former category, while the HsMM-MVPA method used in the original experiment (Berberyan et al., 2021) belongs to the latter. The next two sections provides an overview of the underlying mechanism of these two models in the context of finding patterns from the extracted features. Procedural details to implement these models for cognitive stage analysis can be found in the Methods chapter of this report.

2.2.1 Ridge Classifier

For any given data with a set of input and corresponding output variables, a straightforward way of evaluating any set of features extracted from these input variables would be to compare the prediction performance of a multiple regression model by training the model separately over the original variables and the extracted features. Decoding of a participant's response to its stimulus based on the corresponding brain data can be seen as a typical multiple regression (or classification, as in our case) problem. The multi-variable inputs of the model here refers to the M/EEG data's channel activities / extracted features, and the output variables are given by the stimuli information of the corresponding trials. Several supervised machine learning techniques are available to achieve this including Logistic Regression, Support Vector Machines, Multilayer Perceptron etc. The ridge classifier used in this project is a simple, robust, and computationally efficient model based on the ridge regression algorithm proposed by Hoerl and Kennard, (1970).

The model mechanism can be explained in two parts – 1) Training session / Fitting the data and 2) Testing session / Predicting the data. Prior to applying the model, the M/EEG data along with the stimuli information is split into training and test data. In the training session, the model is fitted to the data points of the input variables (X) containing the channel activities such that the trained model would have learnt a function, $h(\mathrm{X})$, to map the channel activities to its corresponding stimuli information in the output variable (y) with minimal error (ϵ). At every iteration during the training process, the model validates the error of its output, $\hat{y}$, and updates the coefficients (β) of the input variables based on a cost function (J). The ridge model is defined by the same function as a linear model, which is of the form

$$\hat{y} = h(\mathrm{X}) = \mathrm{X}\beta$$

where $\mathrm{X} = x_1, x_2 \ldots x_\mathrm{n}$ are values of training data, $\beta = \beta_0, \beta_1, \beta_2 \ldots \beta_\mathrm{n}$ are the weights of the model.

The cost function used by the Ridge classifier defines the error using the sum of squared errors (sum of the squares of the distance between the data points of the original and the model output variables) and a regularization factor that contains an L2 norm (sum of the square of the coefficients) penalized by a lambda parameter, λ. This regularization parameter enables us in controlling the model from being influenced by the presence of multicollinear variables[2]. The model is optimized by minimizing this error using a cost function $J(\beta)$. The cost function and error is defined as follows

$$\epsilon = |y - \hat{y}|, \text{ where } y = f(\mathrm{X}) = y_1, y_2 \ldots y_n \text{ are the stimuli of the model.}$$

$$_{\min} J(\beta) = {}_{\min}\epsilon = {}_{\min}(||y - \hat{y}||^2 + \lambda \, ||\beta||^2)$$

During the testing session, the model uses its function with the updated weights to predict the stimuli information for its corresponding channel activities. If there is a correlation between the experimental stimuli and M/EEG data (i.e., if the influence of the stimulus on the brain activity is captured by the neural data), and if it can be defined using a linear combination of the channel activities, then the trained model

[2] Although the extracted features are supposed to be noncollinear, the data samples from specific time windows (see 3.3.1) are selected and split for the training/testing process, and these specific data samples need not necessarily be correlated.

should be able to find the pattern in the brain activity that distinguishes between the experimental stimuli and predict them with minimal error.

It has to be noted that the fitting of this model to an output variable with binary labels (i.e., two unique stimuli) differs from the fitting process described in the previous sections. While the algorithms mentioned in the feature extraction section referred to a process where both the predictor and the criterion variables had continuous values, the predictor variables in this process has continuous values and predict a criterion variable with either 1 or -1 (correct, or incorrect stimulus). If there are more than two stimuli, then the model uses a one-vs-all or one-vs-rest technique, but the core mechanism still remains the same of predicting a binary output. Hence this process is technically a classification process and the hypothesis function used by this classifier model is given by

$$\hat{y} = h(\mathrm{X}) = \begin{cases} +1, & \mathrm{X}\beta > 0 \\ -1, & \mathrm{X}\beta < 0 \end{cases}$$

2.2.2 HsMM-MVPA

Neuroimaging data obtained via M/EEG recording during an experiment contains vital temporal information that enables researchers to analyse the cognitive activities occurring at specific time periods of an experimental trial, and it is crucial to verify that the feature extraction techniques do not distort this temporal information. The extracted features are evaluated for this criterion based on their performance from the HsMM-MVPA method (Anderson et al., 2016), a MVPA technique that allows us to find patterns in brain activities along the dimension of time. The HsMM-MVPA method used in this project is built on the foundation of two theoretical concepts – 1) The 'Classical' theory of ERP (Shah et al., 2004) which provides basis for the '*bump kernel*' that is used in predicting the onset of cognitive stages in single trial M/EEG data, and 2) The Markov Chains theory used by the Hidden semi-Markov Model (HsMM) to calculate the probabilities of states and their duration in a stochastic process that is assumed to have satisfied the Markov property.

Hidden semi-Markov Models can be used to model systems that randomly transitions through different hidden states with random transition times if they satisfy the Markov properties – a) the probability of the system's future state is dependent only its current state, and b) the system has an observable process emitted at each state whose outcomes are influenced by that state. In the HsMM implementation[3] of the cognitive stage analysis, we consider brain as the system, cognitive stages that the brain goes through while

[3] Assumptions made to incorporate HsMM in finding the cognitive stages can be found in Anderson et al., (2016)

a participant performs an experimental task as the hidden states, amount of time spent in each stage as the transition time / state duration, and the brain activities elicited during the corresponding cognitive stages as the observations. But the cognitive stage activities themselves are not directly observable in the M/EEG data as they are obscured by other ongoing brain activities. Hence, prior to applying HsMM, these concealed cognitive stage activities are first detected using MVPA and the observed outcomes are then used to find the hidden stages and their durations. In this section, an intuition behind the pattern matching mechanism involved in predicting the onset of a cognitive stage using trial data from one feature as an example is provided. The 'Methods' chapter details a step-by-step operational procedure for estimating the parameters (state probabilities and durations) of the HsMM model.

According to the 'Classical' theory of ERP, during the onset of significant cognitive events (stages), neurons oscillating at a specific frequency with same phase but different amplitudes spike with a peak activity (referred to as *bump* in this report) for a short period of time (indicate as a blue sine wave in Fig. 13b). The macroscopic activities seen in the M/EEG data are a combination of these phasic bursts and the ongoing activities from other neurons oscillating at different phases, frequencies, and amplitudes. To find these cognitive stages that are latent in the M/EEG data, Anderson et al., (2016) modelled the peak activity of stage onset using a 10 Hz half-sine wave (*bump kernel*), and the duration of stage (*flat*) using gamma distribution with a shape value of 2 (*flat kernel*) (see Fig. 13c). Scale parameter of the flat kernel, and probability of the bump given the flat is calculated for each data sample of a trial during the multi-channel analysis of the single trial M/EEG data using Hidden semi-Markov Model. Time stamps of the data samples having the highest bump probabilities determine the location of the bump in the trial. Details of the bump and flat kernel calculation can be seen Appendix 6.3, and their design choices in finding peak activity pattern and duration, respectively, can be found in Anderson et al., (2016).

Probability estimation of the bumps in an ongoing M/EEG signal can be understood as similar to the calculation of correlation between two vectors as seen in the previous sections. Consider an experimental trial of EEG data sampled at ***100 Hz*** with a duration of ***800 ms*** in which the participants undergo a sequence of cognitive processing stages to perform a cognitive task. If we take data samples of the first ***50 ms (or 5 samples)*** of an electrode (or an extracted feature) from this trial as ***x(t)***, and plot it along with the ***5 samples*** of the bump kernel, ***h(t)***, placed at time point, $\boldsymbol{t_1}$, of the trial, in a 2-dimensional plot (see Fig. 13d left), we would be able to calculate the correlation between the EEG signal and the bump kernel for that time point, $\boldsymbol{t_1}$. Similarly, if we continue to calculate the correlation for each time point till ***100th ms*** by moving the bump kernel one point at a time, we would be able to calculate the correlations of the EEG signal and the bump kernel for the corresponding time points.

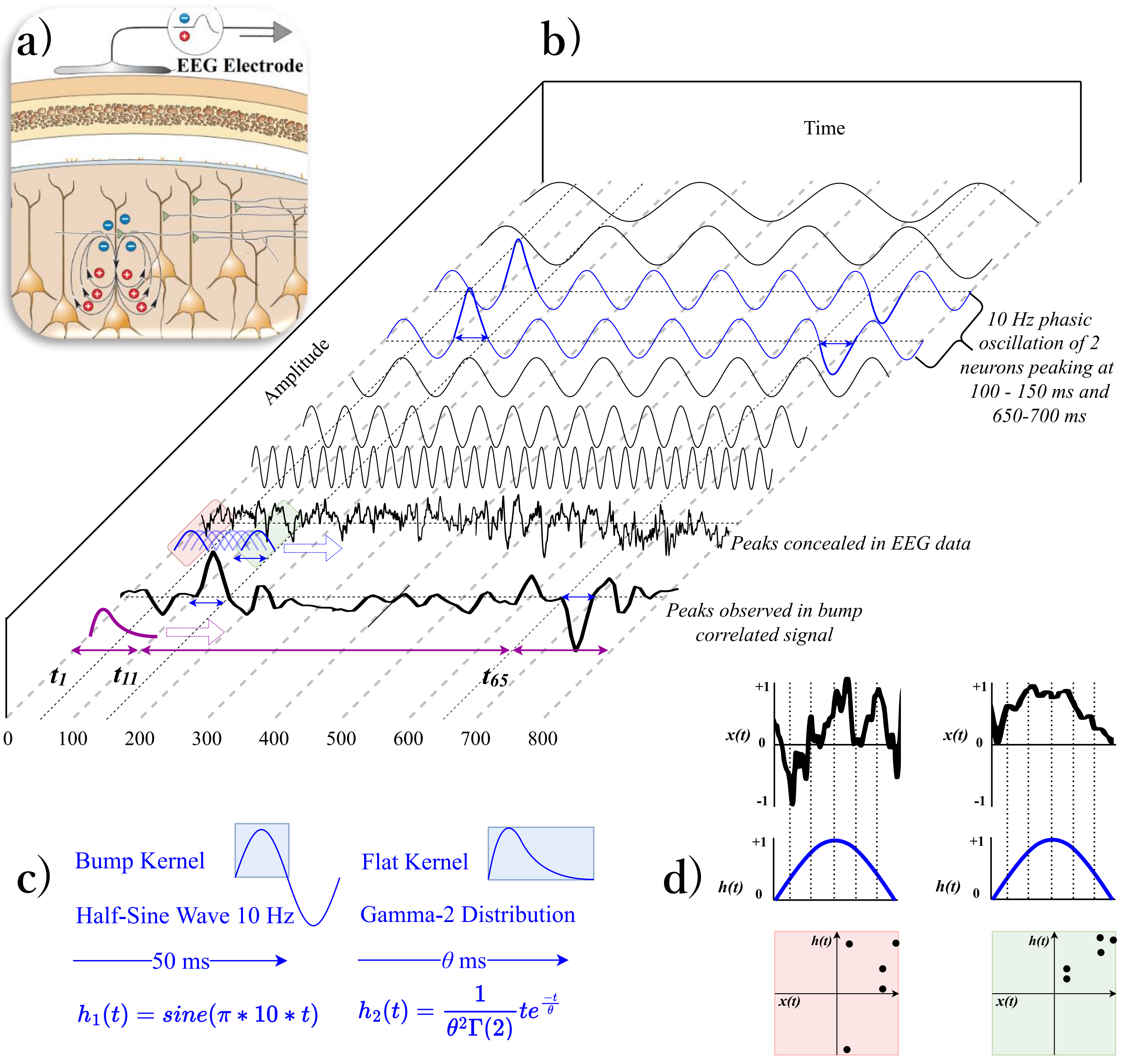

a)
EEG Electrode
b)
Time
Amplitude
10 Hz phasic oscillation of 2 neurons peaking at 100 - 150 ms and 650-700 ms
Peaks concealed in EEG data
Peaks observed in bump correlated signal
t_1
t_{11}
t_{65}
0
100
200
300
400
500
600
700
800
c)
Bump Kernel
Half-Sine Wave 10 Hz
50 ms
$h_1(t) = sine(\pi * 10 * t)$
Flat Kernel
Gamma-2 Distribution
θ ms
$h_2(t) = \frac{1}{\theta^2\Gamma(2)} te^{\frac{-t}{\theta}}$
d)
x(t)
h(t)
+1
0
-1

Figure 13: An illustration of a cognitive event prediction using bump kernel. **a)** EEG electrode recording the combined electrical activity of neurons (source: Bear et al., 2016). **b)** A Time-Frequency-Amplitude representation of the neurons oscillating at different frequencies concealing the peak-activity in the final EEG signal. **c) Left:** Shape and function of the bump kernel used as a 50 ms window function to filter out the non-peak activities. **Right:** Shape and function of the gamma kernel used as a window function with a duration of θ ms to determine duration of each stage observed in the bump correlated signal. **d)** 2-D scatter plot showing the correlation between EEG signal and bump kernel at t_1 (left) and t_{11} (right).

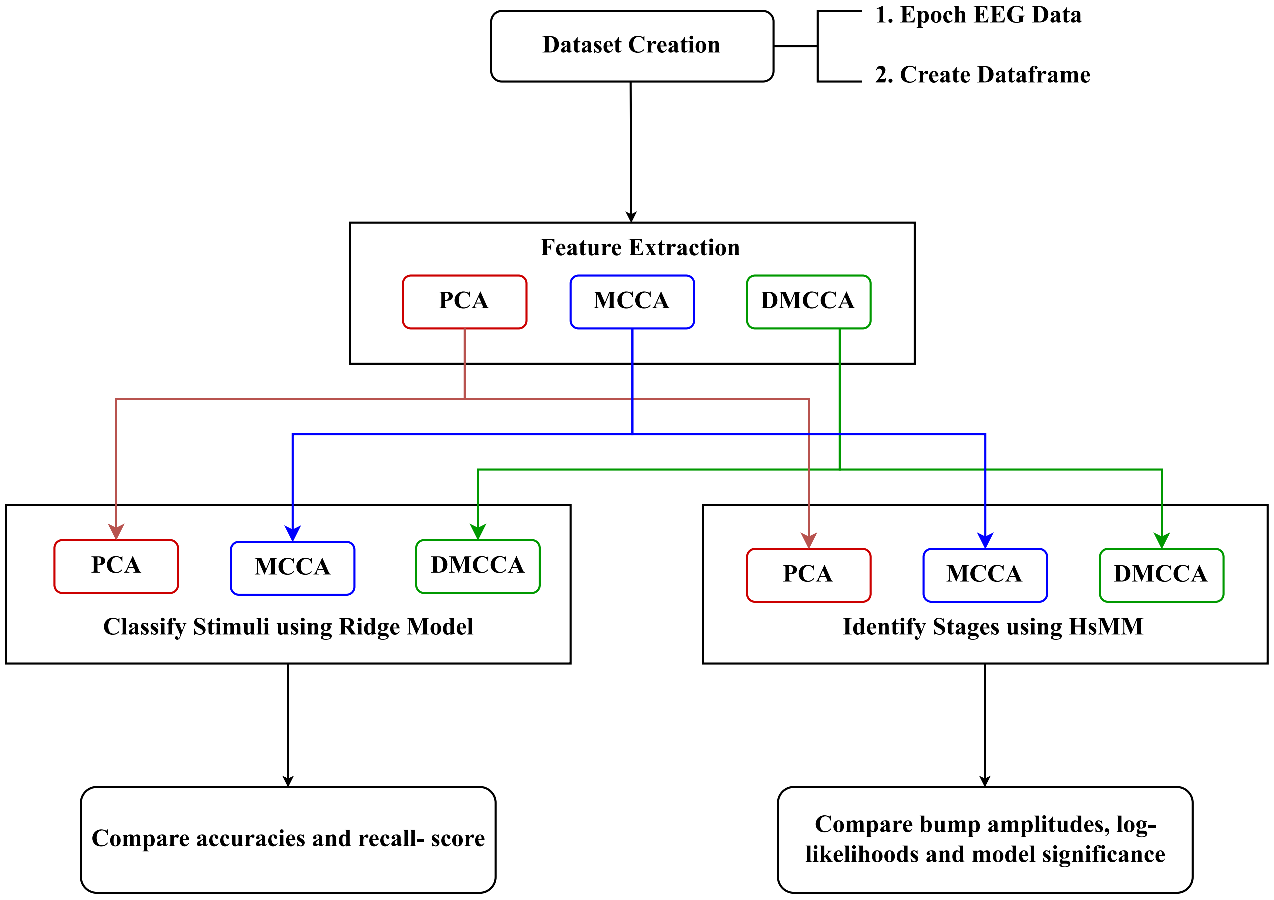
And at time point, $\boldsymbol{t_{11}}$, the EEG signal, now having a shape influenced by the peak activity elicited by the neurons, will correlate maximum with the bump kernel that was modelled to represent the shape of the peak activity (see Fig. 13d right). As we continue, the correlation gradually decreases as the kernel moves along the '*time*' axis. And at time point, $\boldsymbol{t_{65}}$, the signal will again have a high correlation (but negative) between the bump kernel. The bump correlated signal, thus obtained by cross correlating the EEG signal and the bump kernel, is then sequentially correlated[4] with the flat kernel to calculate the ***bump probabilities*** at each time point, $\boldsymbol{t_i}$, given the probability[4] of the ***flat kernel*** at that time, $\boldsymbol{t_i}$. The likelihood of the estimated bump probabilities are then maximized using HsMM's Expectation Maximation (EM) algorithm, and the weights of the model's parameters, state probabilities and state durations, are updated based on the maximized values.

[4] After each maximization step, the bump correlated signal and the flat kernels are updated with the state probabilities and the state durations weights obtained from the previous iteration of the HsMM model, respectively, before sequentially correlating with each other.

Chapter III. Methods

The workflow of the project implementation can be described in 4 phases – Dataset Creation, Feature Extraction, Cognitive Stage Analysis, and Results Comparison. The details of the first three phases are provided in this chapter of the report and the inference from the comparison of the results can be found in the next chapter.

The Dataset section provides information regarding the experimental task and the pre-processing of the EEG data carried out by the authors of the original experiment, followed by the details of the event selection, epoching, and creation of the dataset. The details of the implementation of PCA, MCCA and DMCCA are described in the Feature Extraction section. An overview of the implementation of the Ridge Classifier and HsMM model, used to decode the brain data for cognitive stage analysis and validate the extracted features, is provided in the last section.



Dataset Creation
1. Epoch EEG Data
2. Create Dataframe
Feature Extraction
PCA
MCCA
DMCCA
PCA
MCCA
DMCCA
Classify Stimuli using Ridge Model
PCA
MCCA
DMCCA
Identify Stages using HsMM
Compare accuracies and recall- score
Compare bump amplitudes, log-likelihoods and model significance

Figure 14: A block diagram containing the different modules in each phase of the workflow

3.1 Dataset

3.1.1 EEG Task, Recording and Pre-processing

In the original experiment, participants were shown a sequence of letters and asked to respond if the displayed string is a word or non-word. The words (and non-words) were chosen randomly from the Dutch Lexicon Project containing 14,000 words and 14,000 non-words. The items displayed were based on four conditions - high frequency words (HF), low frequency (LF) words, pseudowords and random non-words. The 4-block experiment consisted of a total of 960 trials, split into 60 trials for each of the four conditions in a block. The trial duration consisted of a 400 – 600 ms fixation period, a maximum response time of 2000 ms, and an incorrect feedback duration of 2000 ms. EEG data was collected using 32 Ag-AgCl electrodes placed in a 10-20 system from 29 participants at a sampling rate of 512 Hz. The authors processed the raw EEG signals from 26 participants (1 faulty, and 2 noisy data were dropped) using the EEGLAB toolbox in MATLAB. The pre-processing steps included a) re-referencing to the average of mastoids, b) applying high-pass (0.5 Hz) and low-pass (40 Hz) filters, c) manual artefact rejection, d) Independent Component Analysis to remove eye blinks and muscle movements, and e) annotating the raw data with event info.

3.1.2 Event selection, Epoching and Dataset Creation

The dataset required for the project was created using the ***mne*** package for python. The pre-processed EEG data was first resampled to ***100 Hz***, and then using the obtained event information, the duration of each trial, t_{dur}, was calculated from the stimulus onset event till the response event. Next, the EEG data were baseline corrected from ***-400 ms to 0 ms*** (400ms prior to the onset of stimuli), and epoched using a minimum ($t_{min} = -400$) and maximum duration, t_{max}, equal to the baseline duration and the duration of the longest trial, respectively. This step was followed by the conversion of epoched data to a ***pandas*** dataframe for a convenient analysis and visualization of the data. In the next step, the excess duration datapoints in each trial were removed from the dataframe (from -400 ms to 0 ms, and from t_{dur} to t_{max}). In the final step, '***time window***' information required for the Ridge classifier was added to the dataframe. This was done by adding time window labels (ranging from 25, 50, 75, . . ., 675) to the data samples falling under the corresponding time windows.

3.2 Feature Extraction

In all three feature extraction techniques implemented for this project, 10 features extracted from the original data were retained. This allowed for a fair comparison with the feature extraction method used in the original study, where the top 10 principal components (PCs) explaining ~90% of the data were retained.

3.2.1 Principal Component Analysis

PCA was implemented as seen in the original study, and also on individual subject data (Within-PCA) and the combined data of all subjects (Across-PCA). The last two variants of PCA were implemented to get a better insight in understanding the inter-subject and intra-subject noise, respectively. A description of the general procedure to perform PCA is provided below, followed by the variations implemented for this project.

The first step in a principal component analysis is to find the covariance matrix of the channel variables in the M/EEG dataset. If the dataset is of shape ***m x n***, with ***m*** observations and ***n*** channels, then the resulting covariance matrix will be of the shape ***n x n***. The diagonal elements of this matrix contains the variances of the features, and the remaining elements of the matrix consist of the pairwise covariances between the corresponding pairs of features in the data. The next step involves factorizing the covariance matrix using eigen decomposition to determine the eigen vectors and eigen values of the covariance matrix. Each of these eigen vectors represents the directions of the observed variance and the eigen values seen along the main diagonal of the matrix contains the magnitude of the variance. In the third step, we arrange the eigen vectors by sorting its eigen values from the largest to the smallest magnitude. After this ordering, the amount of variance explained by the eigen vectors descends from left to right. These eigen vectors, also known as loadings or principal component coefficients, acts as weights while calculating the principal component scores. In the fourth step, we find the product of the data matrix and the weight matrix to get the principal component scores. The final step is to select the first ***p*** components (left to right) that explain the maximum variance observed in the data. It should be noted that PCA is a lossless algorithm, that is, by using all of the extracted principal components, the original features can be reconstructed fully.

PCA (Berberyan et al., 2021)

This is the implementation of PCA performed in the original study. The purpose of using the dataset obtained from this method is to keep this data as a reference while comparing the results from the HsMM-MVPA method. In their study, each subject's data is first scaled to the mean variance of the overall data.

Next, the covariance matrix is calculated for each trial. Then, PCA is applied on the mean covariance matrix of all trials across all subjects. In the last step, the data from each trial is standardized.

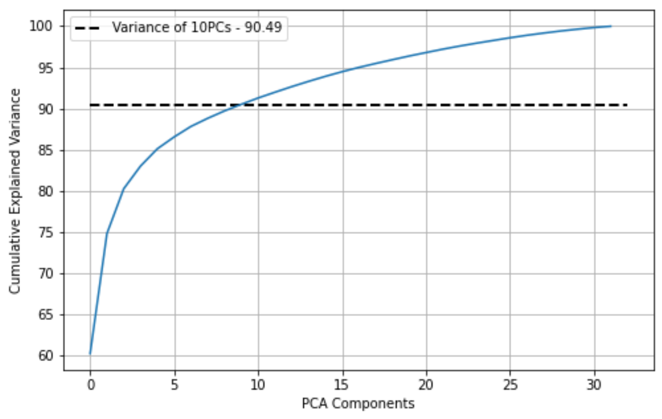


Figure 15: Plot explaining the total variance accumulated over all of the 32 components. The curve gets flatter as the amount of variance explained by the low-ranked components decreases, implying that the vectors of the low-ranked components represent the direction with less variance, and could be considered as noise or less relevant for the analysis.

For the next two methods, the PCA class available in the matrix decomposition module of Python's scikit-learn library was used. The core mechanism of PCA is the same as explained earlier. But, instead of factorizing the data's covariance matrix using eigen decomposition, the library's default algorithm – randomized SVD algorithm was used to decompose the data matrix.

Within-Subject PCA

The PCA-SVD algorithm was applied by treating each subject's data as an individual dataset, and the resulting PCA data of each subject was standardized individually. The new dataset created using this method is robust for individual subject analyses, but the performance metrics from Between-Subject classification can be compared with the scores of Across-PCA to demonstrate the effect of alignment issues caused by combining principal components from different subjects (as described in Section 1.1.1). The average (across subjects) explained variance obtained from the first 10 PCs from this method was 93.53%.

Across-Subjects PCA

In this method, data from all the participants were combined, and then PCA-SVD was applied on this combined dataset. The data were standardized across subjects. However, for comparison purpose, both the datasets (Within-Subject PCA and Across-Subjects PCA) were also standardized per trial and per subject. The first 10 PCs of this method explained 90.55% of the total variance.

3.2.2 Multiset Canonical Correlation Analysis

To extract features from the MEG data of multiple subjects, Zhang et al. (2017) implemented the MCCA technique that allows for the extraction of features that are free from the intra-subject noise while addressing the inter-subject alignment issues pertaining to the differences in sensor locations, as well as the structural and functional differences in the participants' brains.

The MCCA method being a combination of two techniques, requires a selection of some ***p*** PCA components in the first part of the process before creating the final dataset with ***k*** MCCA components. Unlike the PCA implementation seen in the previous section where the model was fit to the dataset containing the datapoints from all the trials, in MCCA, the model is fitted to the data where the trials are averaged over the time points for each condition. The weights obtained from the model trained on this trial-average data are then used to transform the original single trial dataset. While the selection of the top ***p*** components allows to reduce the spatial noise related to the electrodes, the trial-averaging process allows to reduce the inter-trial noise. These two filtering processes ensures that the data from the individual subjects have minimal intra-subject noise. The next section provides the details of the MCCA procedure applied on the EEG dataset used for this project, the shape of the matrices in each step, and their notations as seen in the Figure 11 (mentioned in parenthesis).

The trial-averaged data for the **4 conditions** with a duration of **60 samples**[5] (600 ms at a sampling rate of 100 Hz) from the **32 electrodes** had a shape of ***240 x 32*** for each of the ***26*** subjects (referred as ***m x n*** and $S_1, S_2, \ldots, S_M$ in the figure). PCA was applied to these datasets individually and the dimension was reduced to 20 PCs[6] from the original 32 electrodes of each subject's EEG data. The PCA coefficients matrices of shape ***32 x 20*** obtained from each subject's PCA model gives us the PCA weights (W_{PCA}) of each subject, and are saved for combining with the weights from the MCCA model in the next part.

Since PCA factorizes the *covariance* matrix of the electrodes, the obtained principal components, though uncorrelated within each subject, still vary in terms of variance. To maintain uniform variance of the components across the subjects prior to the subject alignment, and to satisfy the condition required for applying the solution from Vía et al., 2007, the subjects' PCA data were scaled to unit variance. These

[5] The process involves stacking of the subjects' PCA data for finding the eigen vectors of the intra-/inter-subject covariance matrices, and the shape of the rows were made uniform across all participants by taking the first 600 ms of trials. Average response time of the participants was found to be ~600 ms from the behavioural data (see Berberyan et al., 2021), trials below 500 ms and above 700 ms were dropped before averaging, and average data was cropped to a duration of 600 ms.

[6] The value of ***p*** is constrained to the condition $p>k$ and is kept as 20, but can be experimented by changing its value based on the amount of variance we want to retain per subject / the amount of noise we wish to remove from each subject.

whitened PCA data of shape ***240 x 20*** (referred as ***m x p***) from ***26* subjects** were then stacked horizontally to form a data matrix of shape ***240 x 520*** (referred as ***m x pM***). Next, the covariance of this stacked PCA data is calculated to obtain a block matrix ($\boldsymbol{R_{PCA}}$) of shape ***520 x 520*** (referred as ***pM x pM***). The diagonal elements (submatrices of shape ***20 x 20***) of this block matrix contain the covariance of the principal components within subjects, and the off-diagonal elements have the cross-covariances of the principal components between subjects.

To apply the solution from Vía et al, 2007, two separate block matrices from the main $\boldsymbol{R_{PCA}}$ block matrix were created. First, the diagonal elements (***20x20*** submatrices) of $\boldsymbol{R_{PCA}}$ were saved in a separate block matrix (referred as $\boldsymbol{R_W}$). This diagonal block matrix $\boldsymbol{R_W}$ is then subtracted from $\boldsymbol{R_{PCA}}$ to obtain a hollow block matrix $\boldsymbol{R_B}$. As mentioned earlier, the diagonal and the off-diagonal ***20x20*** submatrix elements of $\boldsymbol{R_W}$ and $\boldsymbol{R_B}$ contains the auto-covariance and the cross-covariances of the subjects (a sample structure of these matrices obtained from 3 subjects' data with 5 PCs is shown in the Table 2 & 3 of the Appendix). The two block matrices satisfy the condition required for applying the solution to the generalized eigen value problem (Vía et al, 2007, equation 9).

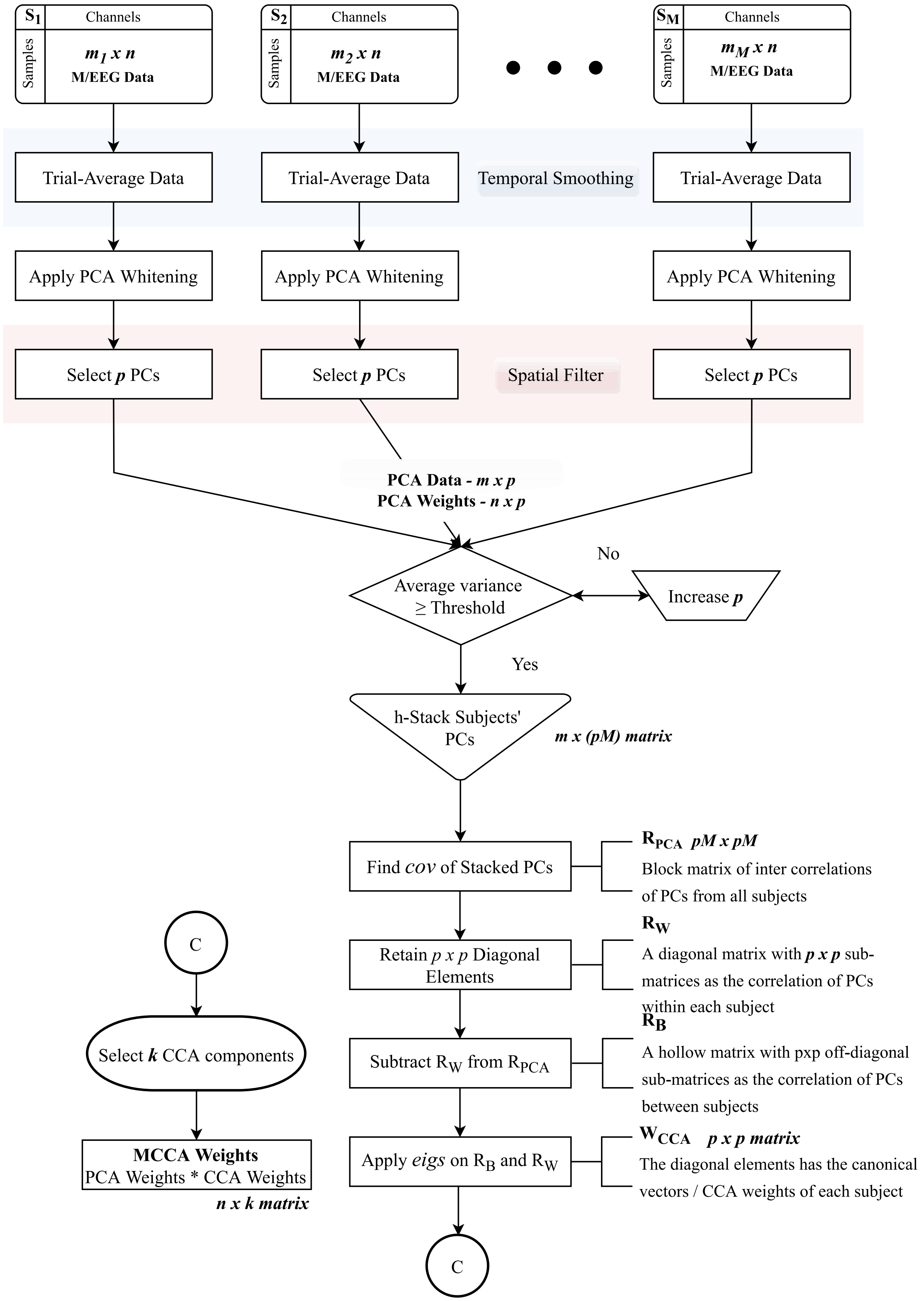


Figure 16: Workflow of the MCCA algorithm applied on the trial-averaged M/EEG data of M subjects.

3.2.3 Deep Multiset Canonical Correlation Analysis

The workflow of DMCCA can be understood similar to that of the MCCA model. But, instead of applying *PCA* to individual subject data, we use the dense layers of individual subject's neural network to apply a nonlinear transformation on their data. The transformed data with 10 features from each subject are then stacked horizontally in the output layer of the network before being fed to the loss function of the model for the GCCA optimization objective. When compared with a regular neural network, the architecture of DMCCA differs in terms of the number of input layers and their corresponding list of dense layers. And functionally, the difference with that of a typical neural network can be noticed in the way how the weights of the individual networks are updated based on a common loss function. The details of the DMCCA architecture implemented for the EEG dataset used in this project is detailed as follows.

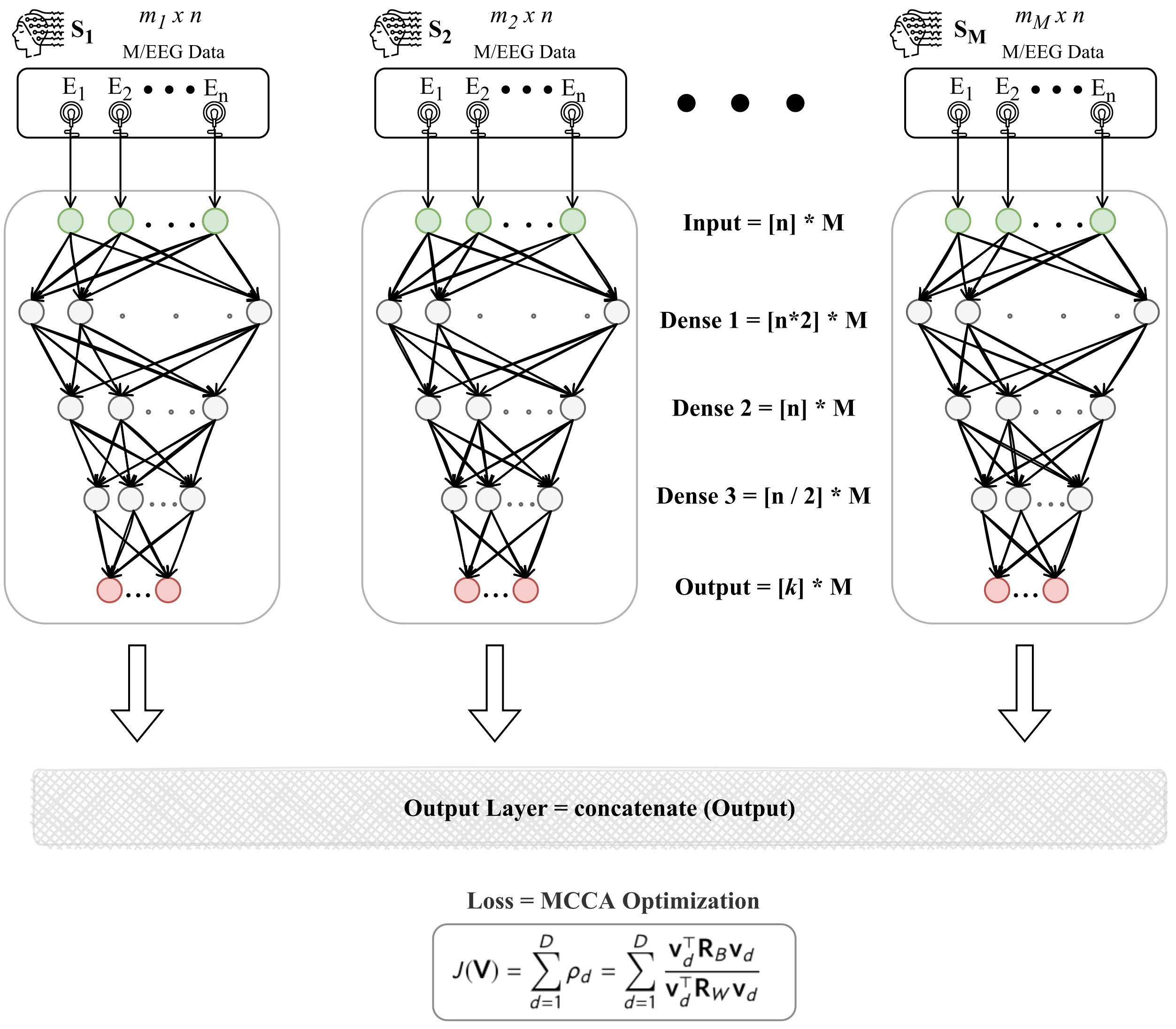


Figure 17: Design of the DMCCA architecture implementation for the M/EEG data of M subjects.

The model was implemented in Python using Keras library. The trial-averaged data, same as seen in MCCA, was used for training. The input layer of the model was initialized as a list of 26 sub-input layers each with 32 nodes reflecting the 26 subjects and the 32 electrodes respectively (referred as **[*n*] * M**). The first dense layer has 64 nodes i.e., twice the input nodes (**[*n* * 2] * M**), the second dense layer has the same dimensions as the input nodes (**[*n*] * M**), and the third dense layer has 16 nodes, that is, half the size of the input dimension (**[*n* / 2] * M**). The output layer has the dimensions equal to the number of ***k*** required features, in this case 10 (**[*k*] * M**). The elements of the output layer list are concatenated (similar to the horizontal stack in the MCCA method) before calling the custom MCCA loss function. The activation functions ***'tanh'*** and ***'sigmoid'*** were used to activate the nodes in the dense and the output layer of the network, respectively. To optimize the process of minimizing loss, '***Adam'*** optimizer was used with varying learning rates.

Computational details of the MCCA loss function used for optimizing the network can be found in Somandepalli et al., (2019), and the theoretical details of the loss function can be found in Parra et al., (2018). The core mechanism of the loss function is similar to the MCCA optimization solution seen in the previous section. Since the neural network model is designed to reduce the value returned by the loss function, and the MCCA optimization solution maximizes the correlation during every iteration, negative value of the MCCA solution is returned as the actual loss value. So, in every epoch, the model tries to reduce the loss value by increasing the ratio of the correlation between subjects and within subjects. The weights of the fully trained model, $\boldsymbol{W_{DMCCA}}$, of the shape ***32x10*** was used to transform the original data similar to the MCCA method[7]. A converged model will obtain an optimal balance of the inter- and intra-subject correlation.

[7] The 'model.predict()' method of this architecture requires the shape of the dataset to be of same size for each subject. Since the number of observations differed between each subject in the LDT dataset, weights of the trained model were calculated separately and applied on the dataset manually. An implementation of the 'predict()' method was also done and applied on a truncated dataset but with no significant differences between both the results.

3.3 Multivariate Pattern Analysis

3.3.1 Ridge Classifier

The primary objective of including a supervised classifier model in cognitive stage analysis is to check if there is a discernible pattern of brain activity that is unique to each of the stimuli conditions. If the model is able to find the pattern and classify the stimuli with an above chance accuracy, we could then identify the different time periods of the distinguishable activity pattern, representative of the different cognitive processing stages, that led to the decision made by the participants. To this end, the model is trained and tested separately in 14 time window segments[8], each consisting of 50 ms of the trial duration. Apart from the accuracy of the model, the 'recall score' of the model was also used as an evaluation metric to observe the sensitivity of the model to the stimuli conditions as each stimulus could have different distinct patterns at different time periods of the trial.

The classifier model used for this purpose is the Ridge classifier from the Scikit-learn package for python. The parameter 'class_weight' was set to 'balanced' to compensate the differences in the class information (availability of data points from each stimulus condition). Two types of classification were done using this model – ***Per-Subject*** classification and ***LOOCV*** classification. By classifying the data for each subject individually, a reference result to compare the actual inter-subject aligned features is obtained. The two classification types have the same procedure but different train / test data for model fitting and predicting respectively. In the *Per-Subject* classification, each subject's data was split in the ratio of ***0.7:0.3*** for ***train:test*** data, and in the *LOOCV* classification, the data from all subjects except m^{th} subject were used for training and the m^{th} subject's data were used for testing, where $m = 1,2,3 \ldots 26$. In both the scenarios, the data was split during the classification of each time window.

3.3.2 HsMM-MVPA

In a typical semi-Markov process where a HsMM is used to find the transition probabilities of hidden states, the number of states is known prior to estimating the parameters of the HsMM model. Since the number of hidden states (cognitive stages) present in an experimental trial are unknown, the HsMM implementation for cognitive stage analysis calculates the probabilities of all possible N state models that could be fit into the shortest trial in the experiment. These N state models were then evaluated using LOOCV procedure, and the model that has the highest log-likelihood over significant number of the subjects determines the number of processing stages involved in the given cognitive experimental task. A

[8] Since the trials of some stimuli conditions are of short length, the class information is extremely unbalanced beyond 700 ms.

step-by-step implementational procedure applied on the EEG dataset used in this project is provided below. A detailed description of the implementational choice of this method including the assumptions made to incorporate HsMM in finding the cognitive stages can be found in Anderson et al., 2016.

→ ***Step 1: Calculate the maximum number of possible states and assign initial values to the state probabilities.***

Variable: 'state_probability'; Shape: (n_features, n_bumps) || Initial Values: Zeroes

The maximum possible stages were calculated based on the duration of the shortest trial and the width of the bump kernel. For this step, the duration of flats was assumed 0 ms to accommodate maximum possible bumps. For the LDT EEG dataset, the shortest trial had a duration of ***300 ms*** (***30 samples***) and the width of the bump kernel was ***fixed to 50 ms*** (***5 samples***). Hence the maximum possible bumps were ***6***. Each possible bump scenario was modelled by a HsMM model, referred to as ***n-bump*** model[9], where ***n = 1,2 . . . 6***. The ***state_probability*** variable provides us with the estimated weights of each bump in an *n-bump* model for a given feature of the EEG data. The initial values of this variable were assigned as zeros. So, the *state_probability* variables of *1-bump, 2-bump . . . 6-bump* models had the shape *(10, 1), (10,2) . . . (10,6)*, respectively.

→ ***Step 2: Assign initial values to the bump kernels and calculate the bump correlated data.***

Variable: 'bump_kernel'; Shape: (n_bumpKernel_samples, n_bumps)

Variable: 'eeg_data'; Shape: (n_eegSignal_samples, n_features)

Variable: 'bump_correlated_data': Shape: (n_bumpCorrelated_samples, n_features)

Since a fixed ***bump_kernel*** was used to predict the probability of all peaks and across all trials, the same initial values, [0.309, 0.809, 1.000, 0.809, and 0.309], were used for the *bump_kernel* of each bump in an *n-bump* model. These values were then normalized to [0.123, 0.323, 0.4, 0.323, 0.123] and then cross-correlated with the data samples from each of the ***10*** features in the ***eeg_data*** to obtain the ***bump_correlated_data***.

→ ***Step 3: Assign initial values to the state durations and generate the flat kernels.***

Variable: 'state_duration'; Shape: (n_flats, 2) || Variable: 'flat_kernel'; Shape (n_flatKernel_samples, n_flats)

The two columns of the ***state_duration*** variable indicates the ***shape*** and ***scale*** parameters of the gamma distribution used by the ***flat_kernel*** to model the durations of each state. The values of the *shape* parameter were ***fixed to 2*** for all flats in an *n-bump* model and the values of its *scale* parameter were initialized based on half the length of the longest trial duration, which was ***156 samples***. So, a *1-bump* model had [2 39; 2 39]

[9] In this method, the initial or 0th bump of a trial is assumed as the trial onset. So, a trial with N states will have N flats and N-1 bumps. For the LDT EEG Data, the maximum possible bumps were 6 implying a maximum of 7 possible cognitive stages.

as the values of its 2 flats, a *2-bump* model with 3 flats had its values as [2 26; 2 26; 2 26], etc. The flat parameters from this *state_duration* variable were then used to generate the gamma distributions of the corresponding flats in the *flat_kernel* variable.

→ Step 4: Calculate the gains per trials using the state probability weights

Variable: 'gains_per_trial'; Shape: (n_gain_samples, n_trials, n_bumps)

During the first iteration of the EM algorithm, the initial weights assigned to the *state_probability* variable in *Step 1* were applied on the *bump_correlated_data* to calculate the gains for each feature[10] and then summated over all the features. The exponential gains of this *bump_correlated_data* were then calculated and split separately for each trial to calculate the ***gains_per_trial*** obtained at each time point of a trial for each bump in an *n-bump* model. In the later iterations, the updated *state_probability* weights from ***Step 7*** were used to calculated ***gains_per_trial*** for those corresponding iterations.

→ Step 5: Calculate the bump probabilities given the flat kernel.

Variable: 'bump_probability'; Shape: (n_bumpProbability_samples, n_trials, n_bumps)

The *flat_kernel* obtained from *Step 3* is sequentially correlated with the *gains_per_trial* obtained from the *Step 4* to calculate the values of the ***bump_probability*** variable. Each value in this variable indicates the probability of placing a bump kernel, that is, the probability of a cognitive stage onset at that time point in a given trial for the given bump.

→ Step 6: Apply EM algorithm to maximize the estimated bump probabilities

The EM algorithm used in this method for maximizing the bump probabilities is the ***forward-backward*** algorithm (For implementational details, see Yu & Kobayashi, 2003). The algorithm was looped through the *n-bump* models, and the probabilities of each bump in an *n-bump* model were maximized individually.

→ Step 7: Use the maximized bump probabilities to update state probabilities and state durations

Variable: 'likelihood'; Shape: (n_n-bumpModels, 1)

The maximized bump probabilities of each trial obtained from Step 6 were used to – a) sequentially correlate with the *bump_correlated_data* (calculated at *Step 2*) of each feature and average their values across trials, for each bump, to update the *state_probability* variable, b) update the *scale* parameter of the *state_duration* variable in each trial, and c) calculate the ***likelihood*** of the *n-bump* models of the shape ***(n_n-bumpModels, 1)***.

[10] See Appendix – Equation 4, Anderson, 2016 for the details of the calculation of gains.

→ Step 8: Evaluate the n-bump models using LOOCV and calculate the significance using sign test.

Variable: best-likelihood; Shape: (n_subjects, 1) || Variable: model_significance; Shape: (n_n-bumpModels, 1)

The models were evaluated using a LOOCV procedure similar to the *LOOCV Classification* of the Ridge classifier described in the previous section, that is, the data from all subjects except ***m**th* subject were used for estimating the parameters of the n-bumps models and the ***m**th* subject's data were used to calculate the log-likelihood of the models for that subject. This process is repeated to obtain the ***best likelihood*** of the n-bump models from all subjects (***n_subjects, 1***), and the quality of this metric was determined by calculating their significance across subjects using a sign test (***n_n-bumpModels, 1***).

Chapter IV. Results

The first two sections of this chapter provide the results obtained from the Ridge classifier model and the HsMM-MVPA method. The objective of analysing the results is to verify the applicability of MCCA and DMCCA techniques on the LDT EEG dataset, and quantitatively assess the features extracted from the two techniques in comparison to the PCA implementation of Berberyan et al., (2021). Hence, inferences of the results are made informing these techniques' abilities in finding patterns of the experimental stimuli within and across subjects, and not with respect to the cognitive aspects of the Lexical Decision Task. A short discussion on the outcome of these results is provided in the final section.

4.1 Stimuli Classification

The Ridge classifier model was applied to the datasets obtained from all feature extraction techniques. The ability of the model to perform the classification analysis within seconds[11] enabled a comprehensive evaluation of different variations of the feature extraction techniques (different normalization, sampling rates, DMCCA features obtained with different hyperparameters, features extracted from RAW EEG data etc.,), and the relevant results are shared in this report. To evaluate and compare the performance of the model trained on the features from different techniques, two metrics of the model were used.

1. ***Accuracy***: Mean accuracies of the model in predicting the labels in a given time window → indicates how precise the model was able to distinguish the stimuli using the train/test data available in that window.

2. ***Recall score***: Indicates the model's sensitivity to a particular stimulus. The more the model learns about a stimulus (fits to the data distribution pattern of the stimulus) during the training session, the higher will be its sensitivity towards that stimulus (tendency to predict the test data as that stimulus) in the testing session.

Note 1: The time limit of 700 ms seen in the plots does not reflect the duration of the actual trials, but rather the time windows in which the model was trained. For example, if the duration of a 'pseudoword' trial was 1200 ms, the data samples of the first 700 ms of that trial were used for the training. Since the duration of trials was dependent on the type of stimulus, dropping them directly impacted the class balance of the stimuli, and the trials were therefore retained in the final input of the model. The trial count of each stimulus based on its maximum duration (segmented by time windows) is shown in Appendix 6.4.

[11] The combined training/prediction time of Per Subject and LOOCV classification for a dataset with ~1M observations and 10 features was ~10s and ~50s, respectively, using an Intel Xeon CPU with 2 cores running at 2.20GHz.

Note 2: The early above chance prediction made by the classifier model need not necessarily indicate the emergence of a pattern at those time points, but rather a result of combining information across all time samples in that window. When the model was trained using the data samples from each time point separately, this effect was not noticed (see Appendix 6.5).

To verify the advantages of PCA over the EEG data as described in section 2.2.1, the model was also applied to the pre-processed EEG data in addition to the extracted features. Since the Ridge classifier can handle multi-collinear variables, the model's performance should not be affected by the correlation between EEG electrodes (see Appendix 6.6 for an illustration of EEG correlation as observed in the LDT dataset), and the results should solely represent the amount of relevant information available in the EEG data (and the model's ability to capture it).

→ If the EEG data was prone only to the intra-subject noise, then the performance of the model from both methods, *Per-Subject* and LOOCV classification, trained on the PCA datasets should be relatively similar to its performance obtained from the MCCA and DMCCA datasets, but better than its performance from EEG dataset.
→ If the EEG data contained intra- and inter-subject noise, then the *Per-Subject* classification results across the feature extraction techniques should be similar, but the LOOCV classification results from MCCA and DMCCA datasets should be better than the LOOCV results from PCA and EEG datasets.

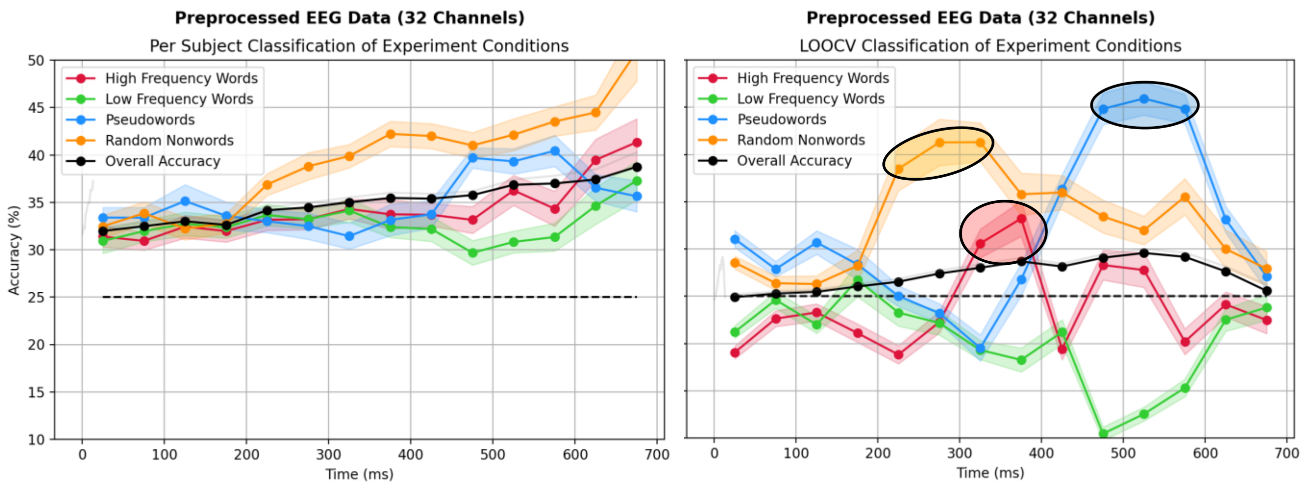


Figure 18: *Per-Subject* and LOOCV classification results for the 4 stimuli conditions of the original experiment obtained from EEG data. Coloured and black dots indicate the average recall score of the stimuli and average accuracy of the model, respectively, and the coloured shades on the lines indicate the SEM. Dots are plotted at the centre of the window duration, that is, the average calculated from 0 – 50 ms is plotted at 25 ms, 50 – 100 ms at 75 ms etc. The dashed line, marked horizontally at 25% accuracy, indicates the chance level accuracy. Coloured circles mark the model's peak sensitivity towards a particular stimulus in the corresponding windows.

Performance of the *Per-Subject* classification model and LOOCV classification model trained over the data from 32 channels of the pre-processed EEG dataset can be seen in Fig. 18. These results serve as a baseline and the inferences of these results are provided in the following sections, by comparing them with the results obtained from the feature extraction techniques.

4.1.1 Within-Subject PCA

Per-Subject classification results obtained from the Within-Subject PCA (Fig. 19-Left) data showed no significant improvement over the classifier results obtained from the EEG dataset. On the contrary, the EEG dataset had better results than the Within-Subject PCA dataset with 10 components. From this result, it can be inferred that the individual subject's EEG data had a good signal-to-noise ratio, as the presence of high intra-subject noise would have made the classifier model learn less information from the noisy channels of the EEG dataset compared to the relatively noise-free Within-Subject PCA components. So, in this scenario, Within-Subject PCA data helped only in reducing the dimensions of the dataset but not in improving the performance of the classification.

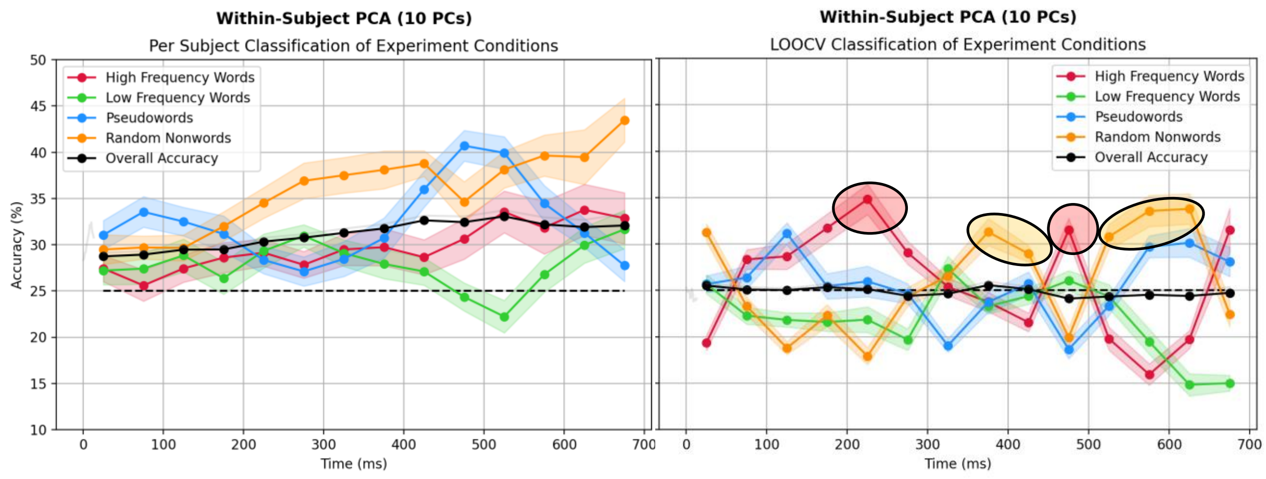


Figure 19: *Per-Subject* and LOOCV classification results obtained from the Within-Subject PCA data using the first 10 principal components.

When the model was applied to multiple subjects' data via the LOOCV classification method, the mean accuracy obtained from the *Within-Subject* PCA data was poor and the overall accuracy did not cross the above chance level (see Fig.19-Right). This was to be expected since training the model over the data created by combining the PCs of individual subjects without maximizing their individual correlation only made it worse than the original EEG dataset[12]. But the model's ability to predict the '*High-Frequency Words*' at 200 – 250 ms, 450 – 500 ms, and '*Random Nonwords*' at 350 – 400 ms, 500 – 650 ms with above-chance accuracy could indicate that the data samples of these two stimuli (in the corresponding windows) obtained from the Within-Subject PCA components of 26 participants were already correlated (across subjects), and did not require any further transformation. In other words, the activity patterns of these two stimuli were strong enough to be picked up by the Within-Subject PCA and the transformed components of individual participants were consistent enough to overcome the inter-subject noise.

Key Findings from Within-Subject PCA

1. Similar performance of *Per-Subject* classification on Within-Subject PCA and EEG datasets show that the EEG Dataset was relatively free from intra-subject noise
2. The results also indicate the Ridge classifier's ability in handling correlated data of the EEG dataset.
3. The high sensitivity of the LOOCV model towards the '*High-Frequency Words*' and '*Random Nonwords*' in certain time windows raises the question if the MCCA technique, which utilizes 20 components of WPCA, would be able to retain these patterns observed within subjects.

4.1.2 Across-Subjects PCA

Per-Subject classification results obtained from Across-Subjects PCA (Fig. 20-Left) were similar to the results obtained from Within-Subject PCA. This confirms the inference made regarding the signal-to-noise ratio of individual subjects' EEG data in the previous section. As described in section 2.1.1 (see Fig 8-Right), while applying PCA to the combined data of multiple participants, the algorithm will be challenged to find patterns from a data distribution that would also have the combined intra-subject noise from multiple participants. If the individual's signal-to-noise ratio had been poor, the *Per-Subject* classification results obtained from Across-Subjects PCA features would have been poor compared to the performance of the model trained on features obtained from Within-Subject PCA, as the latter deals with the intra-subject noise.

The performance of LOOCV model trained on Across-Subjects PCA components (Fig. 20-Right) was better when compared to its performance obtained from Within-Subject PCA components. This indicates that this method (applying PCA across subjects) is more suitable for group analysis than applying PCA to individual subjects. Although the overall accuracy of the LOOCV model from this method was similar to the results from the EEG dataset, it has to be noted that the model was more sensitive to the '*Random*

[12] Maximizing correlation of PCs here refers to the PCs of one subject to the corresponding PCs of another subject (for example, Subject1 PC1, Subject 2 PC1, Subject 3 PC1 etc.) and not the correlation of PCs within a subject which is expected to be uncorrelated (see 2.1.1 Fig. 8 Left for an illustration of uncorrelated PCs between subjects).

Nonwords', '*High-Frequency Words*', and '*Pseudowords*' stimuli at 250 – 350 ms, 300 – 400 ms, and 450 – 700 ms time windows respectively. Since the class weights were balanced for each window prior to training the model, a higher sensitivity towards these stimuli indicates that the model was able to find more information specific to these stimuli (in the corresponding windows) from the first 10 principal components than it could find from the combined data of the 32 EEG channels. In other words, these three stimuli had more variances in the corresponding window durations, and PCA was able to capture these variances, enabling the classifier model to learn more about these stimuli in the corresponding windows. The existence of these patterns across subjects also made the LOOCV model trained on the EEG dataset and the Across-Subjects PCA data to perform better than the *Per-Subject* model in the corresponding time windows. The LOOCV model's high sensitivity towards nonwords (*pseudowords* and *random nonwords*) also resulted in better overall accuracy at 150 – 200 ms and 600 – 700 ms when compared to its accuracy obtained from the EEG dataset, but at the expense of poorly classifying the other two stimuli.

Key Findings from Across-Subjects PCA

1. LOOCV results showed that, for group analysis, applying PCA across the combined data of participants is a better option than applying PCA individually for each subject and then combining those PCs.
2. The Per-Subject classification results confirms the high signal-to-noise ratio of the EEG dataset.

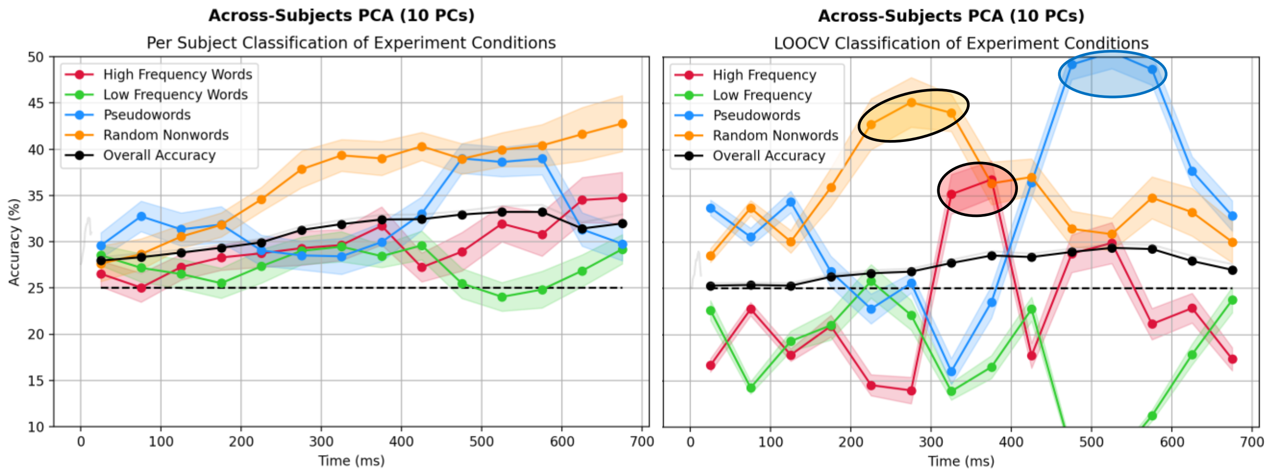


Figure 20: *Per-Subject* and LOOCV Classification results obtained from Across-Subjects PCA Data.

4.1.3 MCCA

Fig. 21 shows the *Per-Subject* and LOOCV classification results obtained from the MCCA features. Performance trend of the model seen in the PCA datasets continued with the MCCA dataset as well, that is, in both classification types the mean accuracy of the model trained using the MCCA features were similar to the results obtained from the EEG dataset and the Across-Subjects PCA features. But a close inspection of the LOOCV results shows that the model was equally sensitive to all the stimuli (except for the '*Pseudoword*' stimulus in the 450 – 600 ms range, a behaviour that is also observed in the other two datasets) when compared with Across-Subjects PCA. This indicates that there could have been nuanced stimuli patterns in individual subjects that were lost during the Across-Subjects PCA, but the within-subject PCA step of MCCA picked up the pattern and the GCCA step made them prominent by maximising the inter-subject correlation. Yet, the results are not representative of the MCCA method's theoretical soundness. A discussion regarding this can be found in section 4.3.

Key Findings from MCCA

1. Similar mean accuracies of the LOOCV model obtained from the MCCA features and Across-Subjects PCA data could imply that the inter-subject noise was relatively low.
2. Yet, the LOOCV model's uniform sensitivity towards the different experimental stimuli showed MCCA's ability in capturing nuanced patterns across subjects, which could be beneficial while performing HsMM-MVPA on single-trial MCCA features (as the model will predict better for each individual stimulus).

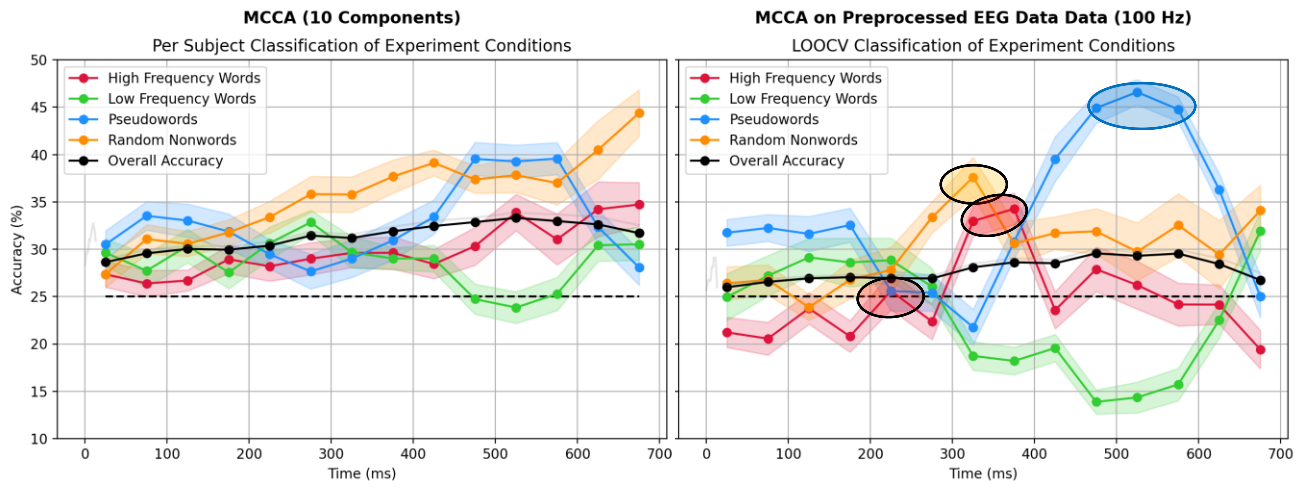


Figure 21: *Per-Subject* and LOOCV Classification results obtained from MCCA dataset.

4.1.4 DMCCA

Unlike linear feature extraction techniques which produce one standard transformation of the data, the DMCCA data differs[13] depending on the architectural preferences including the number of layers per

[13] Apart from the differences in transformation (finding patterns), methodological differences such as 'Conditions' used for the trial-averaged dataset and the inclusion of trial durations with varying lengths that apply to the other feature techniques also apply to DMCCA.

network (with respect to a subject), number of nodes per layer, activation function used in the dense layers etc., and hyperparameter settings such as learning rate and batch size. Furthermore, the DMCCA model's ability to fit the data using a transformation function that is free from any particular shape makes it harder to get the same data even with fixed settings. The classification results shown in Fig. 22-Top represent the performance of a DMMCA dataset that was obtained using one such unknown transformation function. Concluding that the DMCCA technique is able to capture more information related to one particular stimulus depending on the model's sensitivity will not be ideal. For example, the results of a DMCCA dataset (Fig. 22 Bottom) obtained using an architecture with 2 additional layers show completely different behaviour.

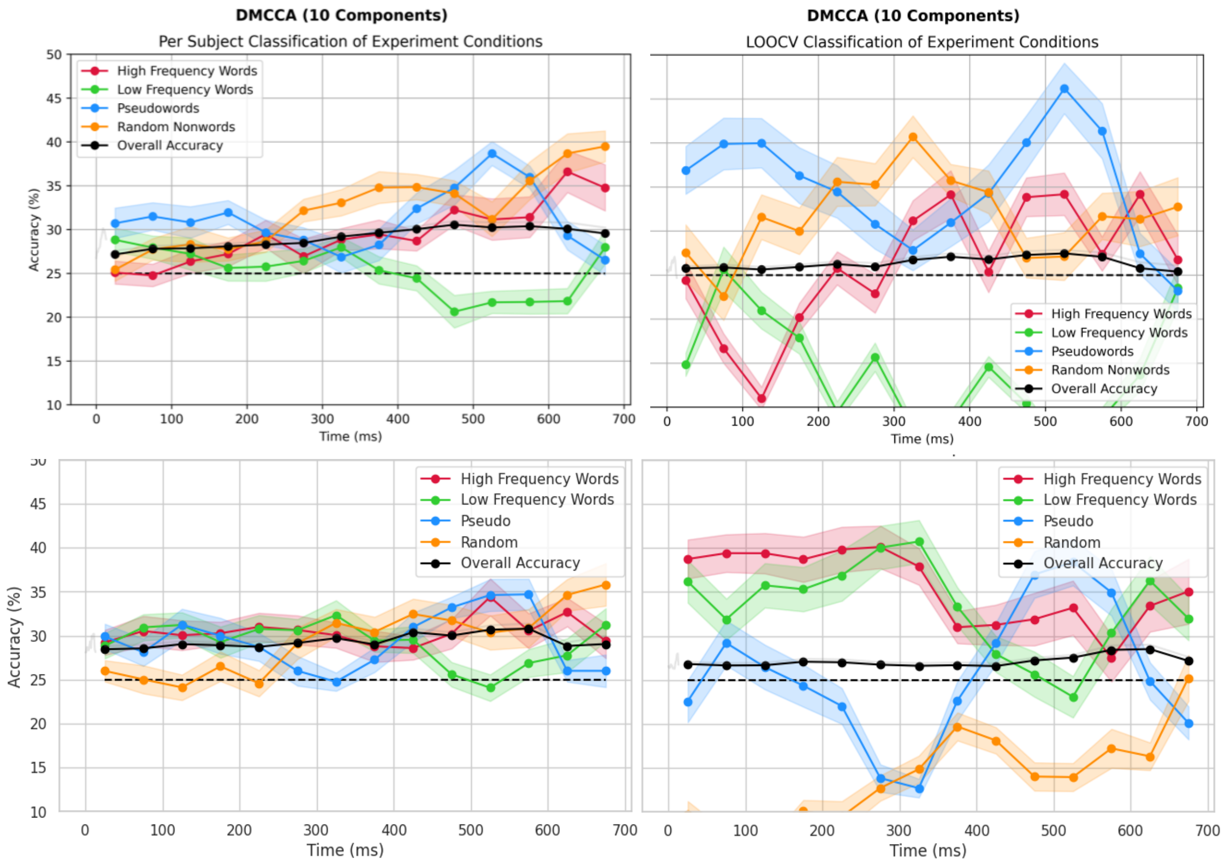


Figure 22: *Per-Subject* and LOOCV Classification results of two DMCCA datasets (top and bottom) obtained using different architecture.

However, the classifier model's sensitivity metric indicates that the DMCCA technique does have the potential in finding unique patterns (different from the patterns found by linear feature extraction techniques) that are common across subjects. If we could converge the results by improving the stability of the model, then the resulting data could provide new insights in the studies. A discussion regarding the same can be found in the Discussion section.

Key Findings from DMCCA

1. DMCCA's ability to find unique patterns different from linear techniques shows the presence of nonlinear relation between the experimental stimuli and the brain activities.
2. The technique, based on its current architecture, is not a viable option to perform any analyses due to its unstable nature.

4.1.5 Classification using Features from RAW EEG Data

One of the key findings from the PCA classification results was that the EEG dataset had a good signal-to-noise ratio within individual subject's data. And the results from MCCA features also indicated a relatively low inter-subject noise. To verify these two findings, features were extracted from the raw unprocessed data using all three techniques and the results obtained from their stimuli classification analysis were compared. The intuition behind this approach is that the steps involved in the pre-processing of EEG data, along with the removal of EEG artefacts, could have reduced the noise related to inter-subject alignment as well.

Initial results from the datasets obtained by applying MCCA/DMCCA on the RAW EEG data were poor compared to the results obtained from the pre-processed EEG data. But, upon dropping the trials that were too short and too long, there was a noticeable improvement in the performance of the classifier model with respect to the MCCA/ DMCCA datasets. Fig. 28 shows the classification results of the RAW EEG dataset (epoched using the same method as described in 3.1.2, but without any preprocessing and resampling of the data) and the datasets obtained by applying PCA/MCCA/DMCCA on the RAW EEG data. While the classifier models trained on RAW and PCA datasets were sensitive towards certain stimuli in specific time windows, models trained on the MCCA/DMCCA dataset were equally sensitive to all the stimuli across all time windows (except for the '*Random Nonwords*' stimulus from 400 – 700 ms for MCCA) and were able to have a better overall accuracy throughout all the time windows.

Note: Across-Subjects PCA was also fit to the same trial-averaged data and the weights from this trial-averaged data were used to transform the RAW EEG data to see if it improves its results, but there was no noticeable improvement in the results.

Key Findings

1. LOOCV results from RAW EEG Data and PCA indicated that existence of inter-subject noise in the LDT EEG dataset.
2. Results from MCCA and DMCCA features showed that the techniques are capable of finding patterns in EEG data that are prone to inter-subject noise.

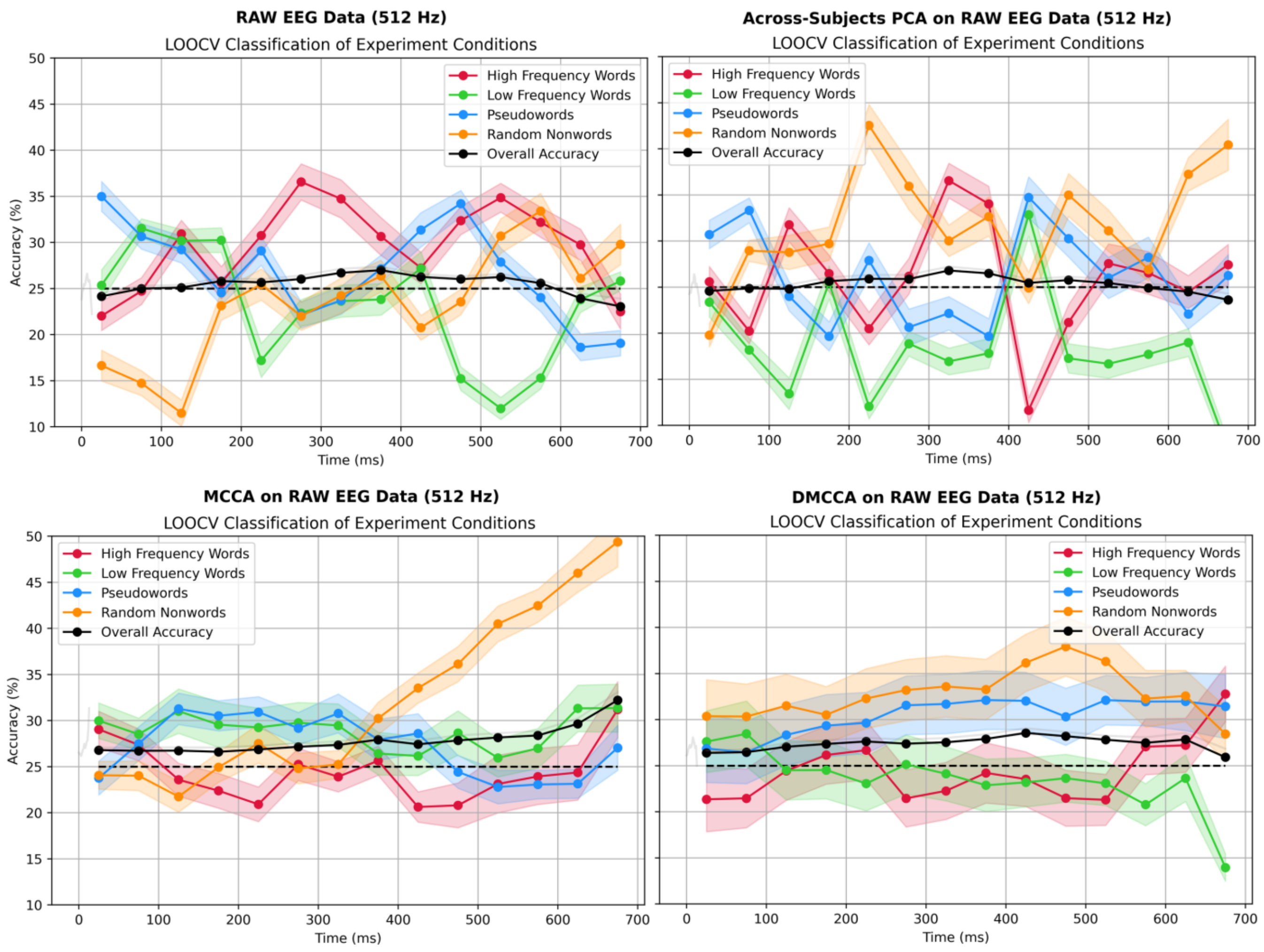


Figure 28: Classification results obtained from the model trained on the RAW EEG data and the PCA, MCCA, and DMCCA data obtained by extracting the features from RAW EEG data.

4.2 Cognitive Stage Prediction

Given the similar performance of classifier models obtained from different feature extraction techniques' data, the results from HsMM-MVPA were also expected to be the same. But the differences in hypothesis function (linear vs bump + flat kernel), time duration (continuous time samples vs fixed time window), and core objective (stage prediction vs stimuli classification) of HsMM-MVPA motivated to perform this analysis for evaluating the feature extraction techniques' ability in retaining the information that will be required by the HsMM model for predicting the cognitive stages. Analysis of two techniques – a) Within-Subject PCA, and b) DMCCA are omitted in this report, the former due to its poor performance in inter-subject classification and the latter due to many possible datasets obtained from the technique. Results obtained from the PCA dataset of Berberyan et al., (2021) are used as a reference to compare the Across-Subject PCA and MCCA implementation.

HsMM-MVPA results were assessed using three metrics,

1.Bump amplitudes[14] – product of the electrodes' amplitudes and the bump probabilities (summed over all features) at each time sample of a trial averaged over all time samples. → Indicates the correlation strength between the PCA/MCCA signal and the model's bump and gamma kernel.

2.Mean log-likelihood – best log-likelihoods obtained via the LOOCV procedure for each *n-bump* model are averaged over the subjects. → Indicates the probability of having ***'N'*** cognitive stages in the LDT experiment as predicted by the corresponding n-bump models.

3.Model significance – number of subjects in which the *n-bump* model's mean log-likelihood was determined to be significant ($p < 0.05$) using a sign-test. → Indicates the generalizability of the model across subjects.

Results from the two PCA implementations (Fig. 23) showed no significant difference and were almost identical. But the results obtained from the MCCA dataset (Fig. 24 & Fig. 25) were slightly different in all three metrics. Firstly, the mean log-likelihoods of each *n-bump* model obtained from MCCA features were better than the corresponding *n-bump* models of PCA dataset. Furthermore, the mean log-likelihood of the 4-bump model (~ -750) trained on MCCA features was higher compared to the dataset's 5-bump model (~ -850), which is different from the PCA results, where the mean log-likelihood of the 5-bump model (~ -2000) was higher than its 4-bump model (< - 2000). Second, the 4-bump model of MCCA dataset was also significant across majority of the subjects (19/26 vs 6/26), which is again in contrast with the generalizability of the 4-bump and 5-bump models of the PCA dataset. Third, the bump amplitudes of each bump in an *n-bump* model obtained from the MCCA dataset was higher compared to the respective bumps in the corresponding *n-bump* models of the PCA dataset. For example, the bump amplitude range of the 5-bump model obtained from MCCA dataset was 11 compared to the 5-bump model of the PCA dataset, which was ~ 8.

[14] As this metric is influenced by the electrodes' amplitudes, it is used to analyse the regions in which the peak activity was prominent. But the analysis in this report focuses on the other factor (bump probabilities) that influences this metric.

The mean log-likelihood results indicate that the HsMM model, based on the information available in the top 10 features of the MCCA dataset, predicted that a presence of 5 cognitive stages is more likely than the presence of 6 stages. Also the strong bump amplitudes of the *n-bump* models trained using MCCA features show the models' abilities in predicting the peaks with high probability, that is, HsMM model trained using MCCA features was able to capture the activity patterns of the cognitive stages better than PCA albeit with poor generalizability across subjects (due to their low significance score).

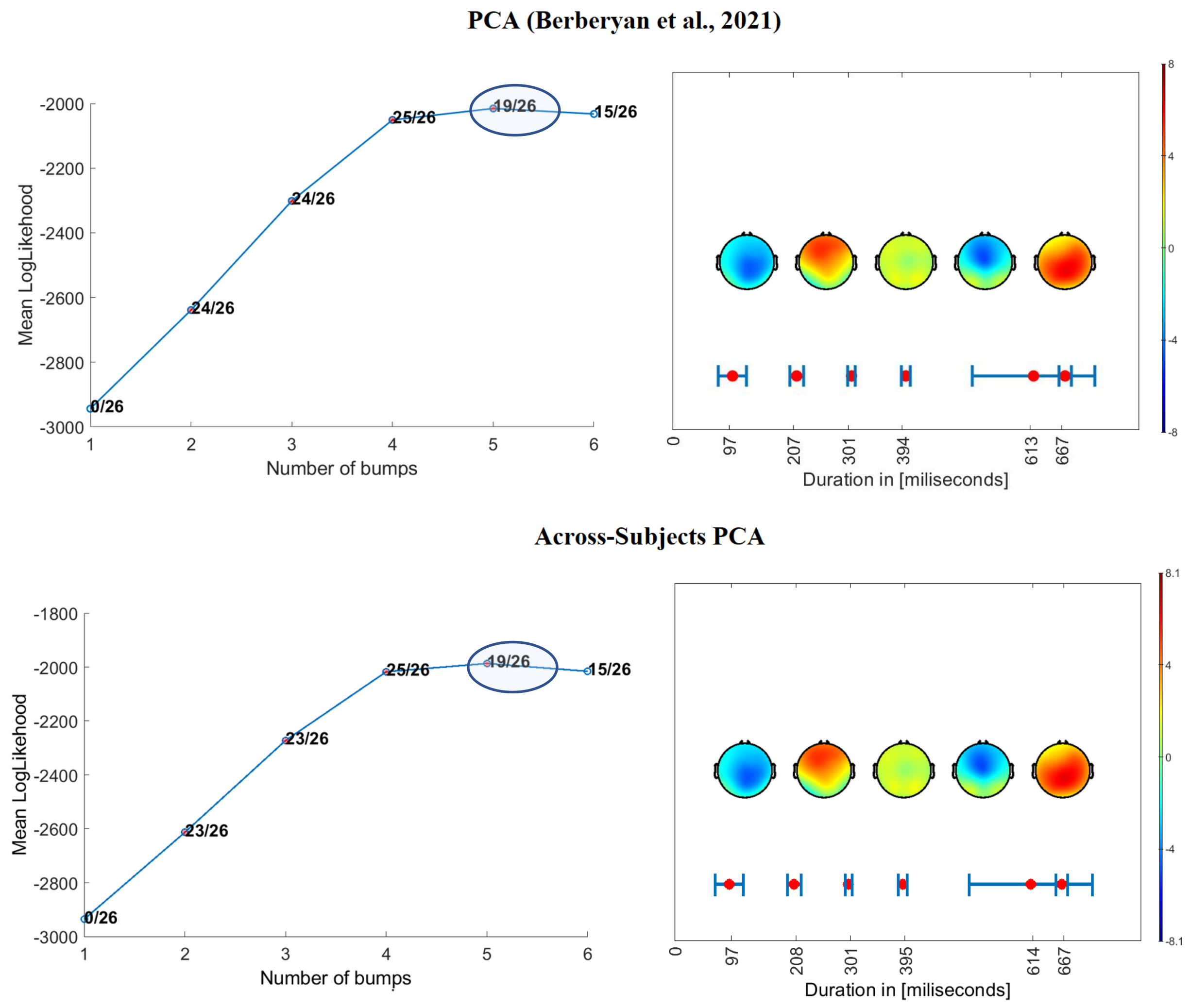


Figure 23: Cognitive stage prediction using HsMM-MVPA. **Left Column:** Mean log-likelihood plot of the *n-bump* models. Models' significance across subjects are mentioned beside their corresponding mean log-likelihood values and circled values corresponds to the *5-bump* models in the right column. **Right Column:** Topoplot of electrodes' bump amplitudes and duration of each stage obtained from the *5-bump* model. Range of the probability values are indicated in the colour-bar. Note: Negative value indicates the probability of a peak activity having negative amplitude.

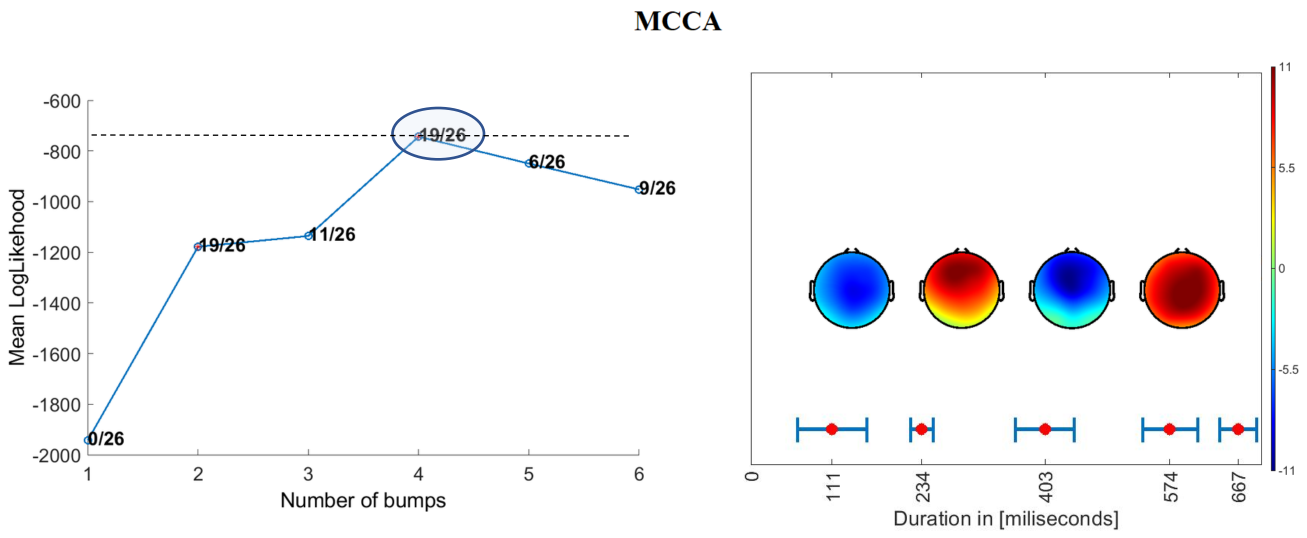


Figure 24: Left: Mean log-likelihoods of the n-bump models obtained from the MCCA dataset. Electrodes' bump amplitudes of the 4-bump model (top right) and 5-bump model (bottom right) obtained from the first 10 components of the MCCA dataset. are shown in the left column.

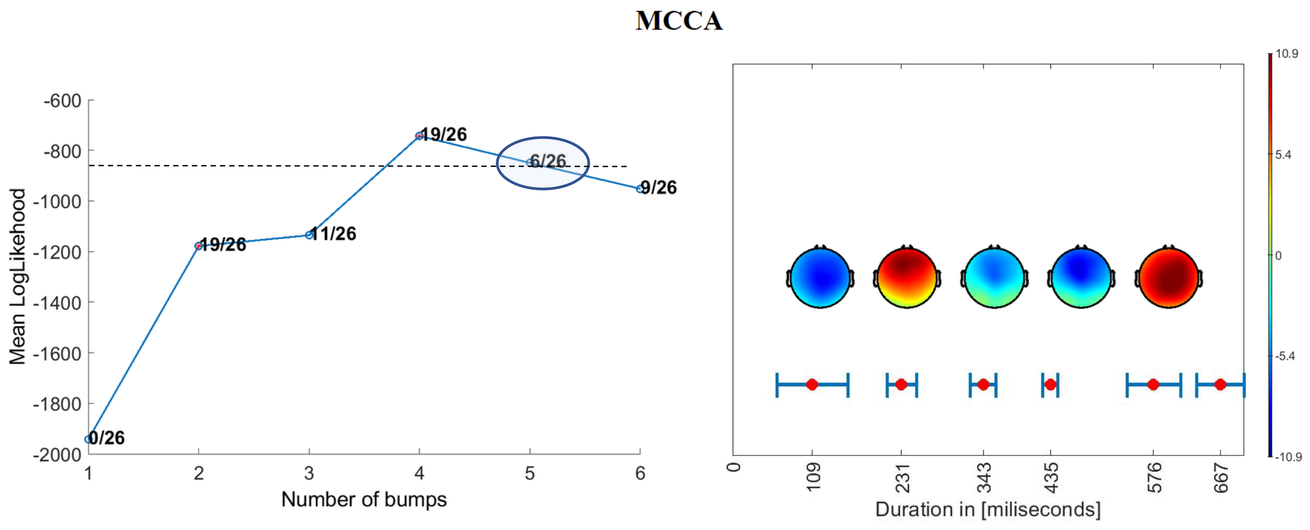


Figure 25: Electrodes' bump amplitudes of the 4-bump model (top right) and 5-bump model (bottom right) obtained from the first 10 components of the MCCA dataset. Mean log-likelihood of the models are shown in the left column.

4.2.1 Selection of CCA Components

To analyse this poor generalizability of the MCCA models, HsMM-MVPA was performed on the first 5 components of the MCCA dataset (see Fig. 26). The significance of these models across subjects were better compared to the MCCA dataset with 10 components (17/26 vs 6/26 for the 5-bump models obtained from 5 vs 10 MCCA components) confirming that the inter-subject correlation diminishes as we increase the number of MCCA components. This is in accordance with the discussion made regarding the selection of MCCA (referred to as 'CCA' in their article) components by the authors of this technique (see Zhang et al., 2017 – subsection 'Application of MCCA'). The authors recommend two possible solutions for the selection of MCCA components – 1) By manually verifying the average correlation of the MCCA components[15] (between subjects) or 2) By setting a fixed threshold value.

However, cutting down the number of components (regardless of the two selection methods mentioned above) would also mean that the amount of relevant information will be reduced along with the reduction of inter-subject noise (a typical issue of maintaining signal-to-noise ratio). This is indicated by the HsMM models' prediction of the bumps with lower probabilities compared to the strong prediction ability of the models trained with 10 MCCA components. Moreover, these models (obtained from 5 MCCA components) still lagged behind the strong generalizability of the models obtained from 5 PCs (5-bump model had a bump amplitude range of 8.1 and generalized across 18/26. Results not shown here).

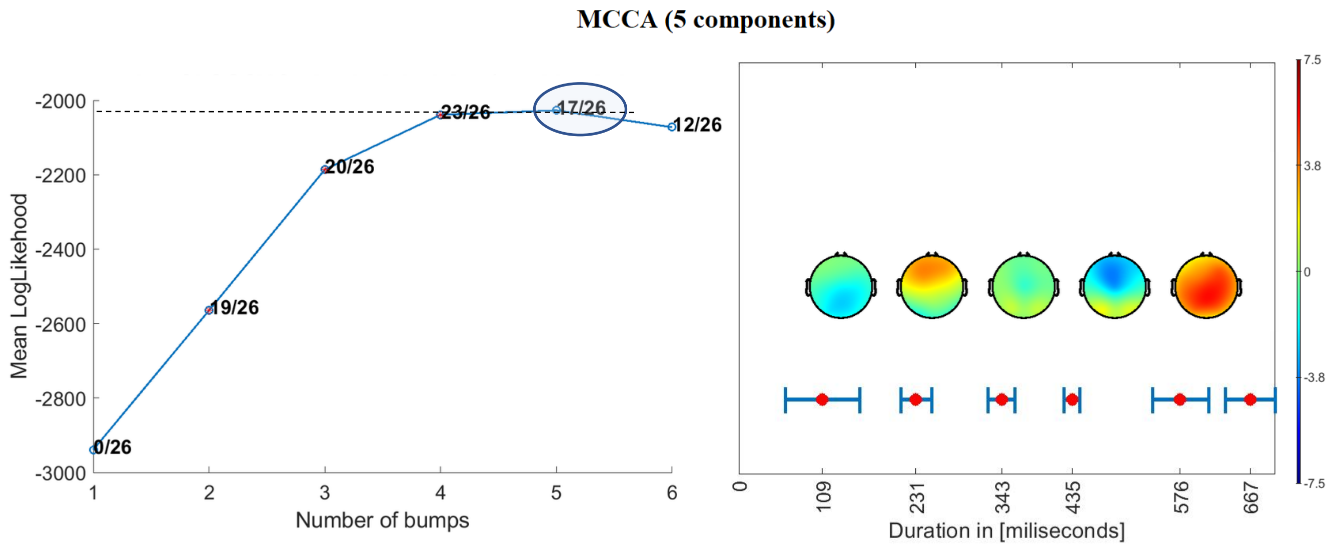


Figure 26: HsMM-MVPA results obtained from the first 5 components of the MCCA dataset.

4.2.2 Effect of Trial-Averaging in MCCA

One of the implementational steps involved in the MCCA method is to average the trials over *Subjects*, *Conditions*, and *Time*. The "*Conditions*" here refers to the stimulus conditions set during the design of the experiment. This temporal smoothing done based on the experimental conditions could benefit the HsMM model if it is trained separately for each condition, as it was performed in the original experiment. But for

[15] The authors recommend splitting the averaged data (used for fitting the model) into train/test data, and making the selection of CCA components based on the inter-subject correlation obtained from the test data.

a general model (as seen in this report) trained across the trial data from all conditions, MCCA dataset averaged on these experimental conditions need not necessarily be beneficial especially due to the differences in the mean duration of the trials in each condition (see Berberyan et al., 2021 for the mean RT per condition). To verify this effect, trials were averaged across a different set of stimuli – ***Words*** (Combination of '*High-Frequency*' and '*Low-Frequency*' words) and ***Non-Words*** ('Pseudowords' and 'Non-Words'). Weights obtained by fitting the MCCA model to this averaged data were used to create a new word/non-word averaged MCCA dataset, and HsMM-MVPA was performed on this new '***wordness***' dataset.

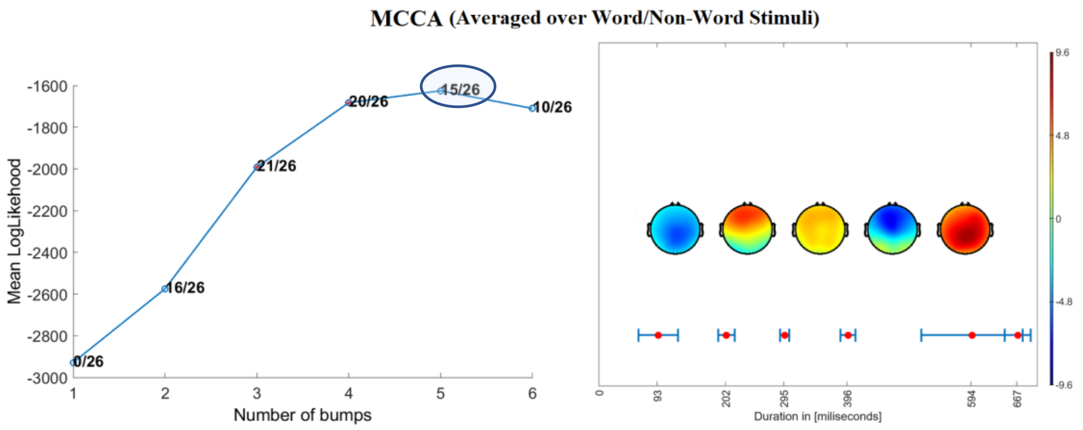


Figure 27: HsMM-MVPA results of the MCCA dataset obtained by fitting MCCA model to the data averaged over 'Word/Non-Word' stimuli condition.

Despite the low number of training samples (165 vs 323 samples per subject in *wordness* vs *experimental conditions* dataset), MCCA model was able to capture the overall pattern that was common across subjects. This can be seen in the results obtained from the first 10 components of this dataset (see Fig. 27). Although the mean log-likelihood and the bump magnitudes were lower compared to the first MCCA dataset with 10 features (~ -1600 & 9.6 vs ~ -800 & 11), the HsMM models were able to generalize well across subjects, especially the 5-bump model (15/26 subjects vs 6/26 subjects). This could be due to a) the smoothing effect of trial-averaging using less conditions could have removed more 'inter-subject' noise from the trials and/or b) averaging over the *wordness* stimuli could have kept more relevant information (with respect to word/non-word) that were prominent across subjects. The latter is more plausible due to the higher bump probabilities observed in the 3rd bump of the 5-bump model, which is not noticed in the bump topology of the PCA dataset.

4.3 Discussion

The first part of this section provides 4 methodological considerations, based on the results obtained from this particular project, that could potentially improve the two techniques' (MCCA and DMCCA) ability in finding features that generalize better across subjects. The second part of this section provides 3 techniques for improving the stability of the DMCCA model in general.

4.3.1 Considerations for MCCA and DMCCA

The common step involved in the implementation of MCCA and DMCCA in this project was to train both models using the trial-averaged data. Although this step helped in removing the inter-trial noise, there were two factors affecting the performance and consistency of the feature extraction techniques.

1. Filtering trials before trial-averaging

The trials in the LDT EEG dataset used in this project were not of equal duration (including trials from the same experimental condition) and this directly impacted the models' ability in finding patterns that were uniform across time windows. For example, if the majority of the trials from a particular stimulus were of the same duration, the activity patterns related to that stimulus in the corresponding time points can be easily detected by the model. But, if the trial durations for a particular stimulus are distributed across a wide range of time (see Appendix 6.4), so will the data distribution of the activity patterns related to a particular phenomenon that occurs in specific time points. And training the model using the overlapped information combined from different time periods would make it hard to find the pattern related to that phenomenon. Hence, a better solution will be to drop the trials (for each experimental condition) that fall outside specific time duration or to train and fit the model separately for trials that fall under specific time windows.

2. Selecting conditions for trial-averaging

Results from HsMM-MVPA also showed the effect of the choice of 'condition' used in the trial-averaging step. But, this factor could affect the consistency of the model's results more, than it could affect the performance of the model. And the selection of conditions also largely depends on the type of analysis performed. Hence setting a standard rule for selecting conditions before the trial-averaging step would not reflect well across all analyses and should be varied depending on the analyses performed.

3. Selection of PCA components

The average variance of the first 20 PCs from the Within-Subject PCA used in this project was 98.38% of the total data. But the preprocessed dataset used in this project was already having a good signal-to-noise ratio. In such scenarios, any reduction of components could directly impact the amount of information available for the MCCA technique. Hence, while selecting PCs in the MCCA technique, consideration should not only be based on the variance explained by the PCs, but also if the selected PCs could discard the actual noise.

4. Selection of CCA components

Zhang et al. (2017) discussed in detail the importance of this step and also provided two recommendations for the selection of the CCA components. However, a third option of verifying inter-subject correlation to detect outliers can also be considered. This would help in situations where data from a minority of the subjects did not correlate well with the remaining subjects' data.

4.3.2 Stabilizing DMCCA

LOOCV Classification results from the DMCCA features showed that the technique has the potential in finding nonlinear patterns and extracting features containing vital information that could otherwise be lost by using linear feature extraction techniques. However, the inability of the model, based on its current design choices, to converge and provide stable results made it unusable for further analysis. To overcome this issue and constrain the neural nets to provide stable results, the following changes could be made in the current method.

1. Using Within-Subject PCA as a preprocessing step to remove intra-subject noise.

By removing the noise that exists within individual subjects' data, the number of possible transformations to be deduced by the neural networks will be significantly reduced. The number of PCs to be selected prior to applying the technique should be similar to the selection of PCs in the MCCA method, as in both cases the main intention of this step is to remove the noise and retain the maximum possible information of the M/EEG data.

2. Using a 'Dropout' mechanism to randomly deactivate network nodes.

As mentioned in the previous section, the DMCCA method used in this project was trained using the trial-averaged data, and the weights obtained from this trial-averaged data were then applied to the individual subjects' EEG data. During this step, brain activities containing vital information about the experimental stimuli but faint in the EEG data would have been smoothened out. If the distribution pattern of these data samples were unique from what was learnt by the model during training, the transformation of these data samples to the lower dimensional space would not be a proper representation of those samples. This effect will become worse if the model overfits the training data. Using 'Dropout' (Srivastava et al., 2014) as a hyperparameter in the neural network will prevent the model from overfitting to the training data and help in regularizing the model for achieving a better transformation of the unseen data samples.

3. Constraining individual subjects' ANNs using MSE as an additional loss function.

In the current DMCCA architecture, the ratio of the inter-/intra-subject correlation values obtained from the GCCA optimization solution was used to update the weights of the neural networks. However, this loss function need not necessarily restrict the ANNs to fit their individual subjects' data, as the model will converge as long as the inter-subject correlation is maximized. Katthi & Ganapathy (2021) implemented a variation of the DMCCA technique by combining the GCCA loss function with the MSE loss function to extract features from the EEG data of participants performing an audio listening task. The MSE loss, calculated on the decoded output of the individual ANNs helps to constrain the output of the individual ANNs and better fit the data of individual subjects.

Chapter V. Conclusion

In view of the increasing interest in evaluating cognitive models using neuroimaging data, the primary objective of this project was set to investigate two issues, inter-subject alignment, and nonlinearity, that could potentially affect the analyses performed in these studies. To this end, two feature extraction techniques, MCCA and DMCCA, were implemented on the EEG data collected from multiple participants performing a cognitive task.

Results from the stimuli classification analysis performed on 10 features extracted using PCA and MCCA indicated that the MCCA technique's ability in reducing inter-subject noise was better than PCA when dealing with RAW EEG data. But when fed with pre-processed EEG data, MCCA and PCA applied across subjects had similar overall performance in classifying the stimuli, with minor differences in their sensitivity towards the stimuli. Owing to its operational characteristic of finding principal components within subjects and maximizing their correlation across subjects, MCCA was still able to find and maximize nuanced patterns observed across subjects, while PCA captured the activities that were prominent across subjects but not the patterns that could have got lost in the noise as a result of combining the subjects' data. Cognitive stage analysis, performed on the PCA and MCCA features extracted from the pre-processed EEG data, showed that the models trained on the PCA dataset generalize better across subjects compared to the MCCA dataset. However, the models trained on the MCCA dataset were able to predict the peak activities with higher probability than they could with the information available from the PCA components.

Having the same ability as MCCA in maximizing the correlation of patterns observed across subjects, DMCCA has the additional advantage of preserving the nonlinearity of the dataset. But this advantage of neural networks also meant that the number of possible 'best' fits is not as standard as the linear feature extraction techniques, resulting in more than one dataset. Yet, given that the DMCCA architecture used in this project had previously not been applied to EEG data, and the architecture consists only of 3 dense layers with a total of 112 nodes, the results obtained from the classification analysis show the technique's potential in finding patterns that are nonlinear and are observed across subjects.

Results from the two multivariate pattern analyses indicate the potential of both techniques' applicability in alleviating inter-subject alignment issues, and results from the classifier model showed the applicability of DMCCA in extracting features from EEG data. 4 methodological considerations for further improving the two techniques' generalizability and 3 techniques to improve the stability of the DMCCA model were provided.

References

Anderson, J.R. (1993). Rules of the Mind (1st ed.). Psychology Press. https://doi.org/10.4324/9781315806938

Anderson, J. R. (2007). How can the human mind occur in the physical universe? Oxford University Press.

Anderson, J. R., Betts, S., Ferris, J. L., & Fincham, J. M. (2012). Tracking children's mental states while solving algebra equations. Human Brain Mapping, 33(11), 2650–2665. https://doi.org/10.1002/hbm.21391

Anderson, J. R., & Fincham, J. M. (2014). Discovering the sequential structure of thought. Cognitive Science, 38(2), 322–352. https://doi.org/10.1111/cogs.12068

Anderson, J.R., & Lebiere, C.J. (1998). The Atomic Components of Thought (1st ed.). Psychology Press. https://doi.org/10.4324/9781315805696

Anderson, J. R., Qin, Y., Sohn, M.-H., Stenger, V. A., & Carter, C. S. (2003). An information-processing model of the BOLD response in symbol manipulation tasks. Psychonomic Bulletin & Review, 10(2), 241–261. https://doi.org/10.3758/bf03196490

Anderson, J. R., Zhang, Q., Borst, J. P., & Walsh, M. M. (2016). The discovery of processing stages: Extension of Sternberg's method. Psychological Review, 123(5), 481–509. https://doi.org/10.1037/rev0000030

Ashby, F. G., & Helie, S. (2011). The neurodynamics of cognition: A tutorial on computational Cognitive Neuroscience. Journal of Mathematical Psychology, 55(4), 273–289. https://doi.org/10.1016/j.jmp.2011.04.003

Bear, M. F., Connors, B. W., & Paradiso, M. A. (2016). Neuroscience: exploring the brain. Fourth edition. Philadelphia: Wolters Kluwer.

Becker, S., & Hinton, G. E. (1992). Self-organizing neural network that discovers surfaces in random-dot stereograms. Nature, 355(6356), 161–163. https://doi.org/10.1038/355161a0

Berberyan, H. S., van Rijn, H., & Borst, J. P. (2021). Discovering the brain stages of lexical decision: Behavioral effects originate from a single neural decision process. Brain and Cognition, 153(105786), 105786. https://doi.org/10.1016/j.bandc.2021.105786

Borst, J. P., & Anderson, J. R. (2015a). The discovery of processing stages: analyzing EEG data with hidden semi-Markov models. NeuroImage, 108, 60–73. https://doi.org/10.1016/j.neuroimage.2014.12.029

Borst, J. P., & Anderson, J. R. (2015b). Using the ACT-R cognitive architecture in combination with fMRI data. In An Introduction to Model-Based Cognitive Neuroscience (pp. 339–352). Springer New York.

Borst, J. P., Ghuman, A. S., & Anderson, J. R. (2016). Tracking cognitive processing stages with MEG: A spatio-temporal model of associative recognition in the brain. NeuroImage, 141, 416–430. https://doi.org/10.1016/j.neuroimage.2016.08.002

Borst, J. P., Nijboer, M., Taatgen, N. A., van Rijn, H., & Anderson, J. R. (2015). Using data-driven model-brain mappings to constrain formal models of cognition. PloS One, 10(3), e0119673. https://doi.org/10.1371/journal.pone.0119673

Borst, J. P., Schneider, D. W., Walsh, M. M., & Anderson, J. R. (2013). Stages of processing in associative recognition: evidence from behavior, EEG, and classification. Journal of Cognitive Neuroscience, 25(12), 2151–2166. https://doi.org/10.1162/jocn_a_00457

Haxby, J. V. (2012). Multivariate pattern analysis of fMRI: the early beginnings. NeuroImage, 62(2), 852–855. https://doi.org/10.1016/j.neuroimage.2012.03.016

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. Technometrics: A Journal of Statistics for the Physical, Chemical, and Engineering Sciences, 12(1), 55. https://doi.org/10.2307/1267351

Horst, P. (1961). Generalized canonical correlations and their applications to experimental data. Journal of Clinical Psychology, 17(4), 331–347. https://doi.org/10.1002/1097-4679(196110)17:4<331::aid-jclp2270170402>3.0.co;2-d

Hotelling, H. (1936). Relations between two sets of variates. Biometrika, 28(3/4), 321. https://doi.org/10.2307/2333955

Katthi, J. R., & Ganapathy, S. (2021). Deep multiway canonical correlation analysis for multi-subject EEG normalization. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. Artificial Intelligence, 33(1), 1–64. https://doi.org/10.1016/0004-3702(87)90050-6

Parra, L. C., Haufe, S., & Dmochowski, J. P. (2018). Correlated Components Analysis - extracting reliable dimensions in multivariate data. https://doi.org/10.48550/ARXIV.1801.08881

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London Edinburgh and Dublin Philosophical Magazine and Journal of Science, 2(11), 559–572. https://doi.org/10.1080/14786440109462720

Peters, B. O., Pfurtscheller, G., & Flyvbjerg, H. (1998). Mining multi-channel EEG for its information content: an ANN-based method for a brain-computer interface. Neural Networks: The Official Journal of the International Neural Network Society, 11(7–8), 1429–1433. https://doi.org/10.1016/s0893-6080(98)00060-4

Shah, A. S., Bressler, S. L., Knuth, K. H., Ding, M., Mehta, A. D., Ulbert, I., & Schroeder, C. E. (2004). Neural dynamics and the fundamental mechanisms of event-related brain potentials. Cerebral Cortex (New York, N.Y.: 1991), 14(5), 476–483. https://doi.org/10.1093/cercor/bhh009

Somandepalli, K., Kumar, N., Travadi, R., & Narayanan, S. (2019). Multimodal representation learning using deep multiset canonical correlation. https://doi.org/10.48550/ARXIV.1904.01775

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research: JMLR, 15(1), 1929–1958. https://dl.acm.org/doi/10.5555/2627435.2670313

Stam, C. J. (2005). Nonlinear dynamical analysis of EEG and MEG: review of an emerging field. Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology, 116(10), 2266–2301. https://doi.org/10.1016/j.clinph.2005.06.011

Vía, J., Santamaría, I., & Pérez, J. (2007). A learning algorithm for adaptive canonical correlation analysis of several data sets. Neural Networks: The Official Journal of the International Neural Network Society, 20(1), 139–152. https://doi.org/10.1016/j.neunet.2006.09.011

Yu, S., & Kobayashi, H. (2003). An efficient forward-backward algorithm for an explicit-duration hidden Markov model. IEEE Signal Processing Letters, 10, 11-14.

Zhang, Q., Borst, J. P., Kass, R. E., & Anderson, J. R. (2017). Inter-subject alignment of MEG datasets in a common representational space. Human Brain Mapping, 38(9), 4287–4301. https://doi.org/10.1002/hbm.23689

Zhang, Q., van Vugt, M., Borst, J. P., & Anderson, J. R. (2018). Mapping working memory retrieval in space and in time: A combined electroencephalography and electrocorticography approach. NeuroImage, 174, 472–484. https://doi.org/10.1016/j.neuroimage.2018.03.039

Appendix

6.1 Angle Between Flats, Cosine Similarity and Correlation

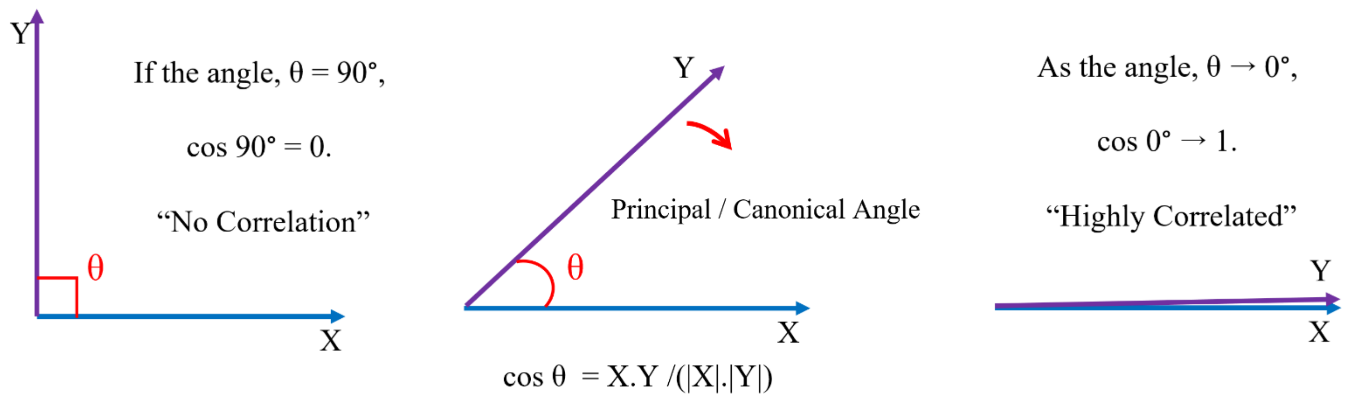
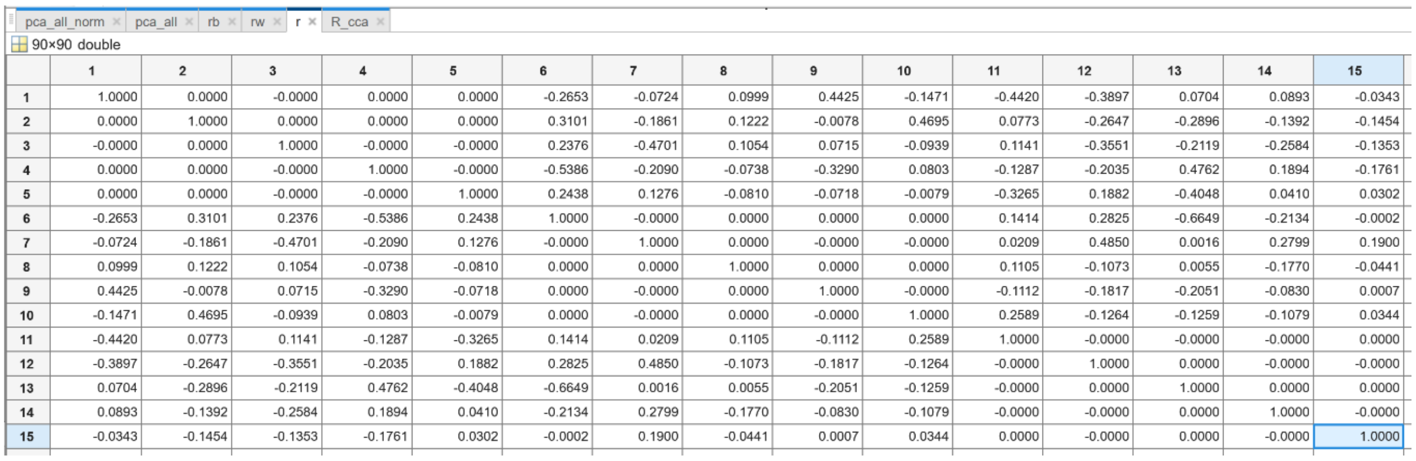


Figure A1: Geometric interpretation of correlation between two vectors, X and Y, using the cosine of angle between the two vectors (also called as principal / canonical angle). For X and Y two be highly correlated, the cosine of their angle should be maximum. This cosine similarity is calculated as the dot product of X and Y divided by the product of their magnitudes.

6.2 Cross-correlation Matrices of Multiple Subjects

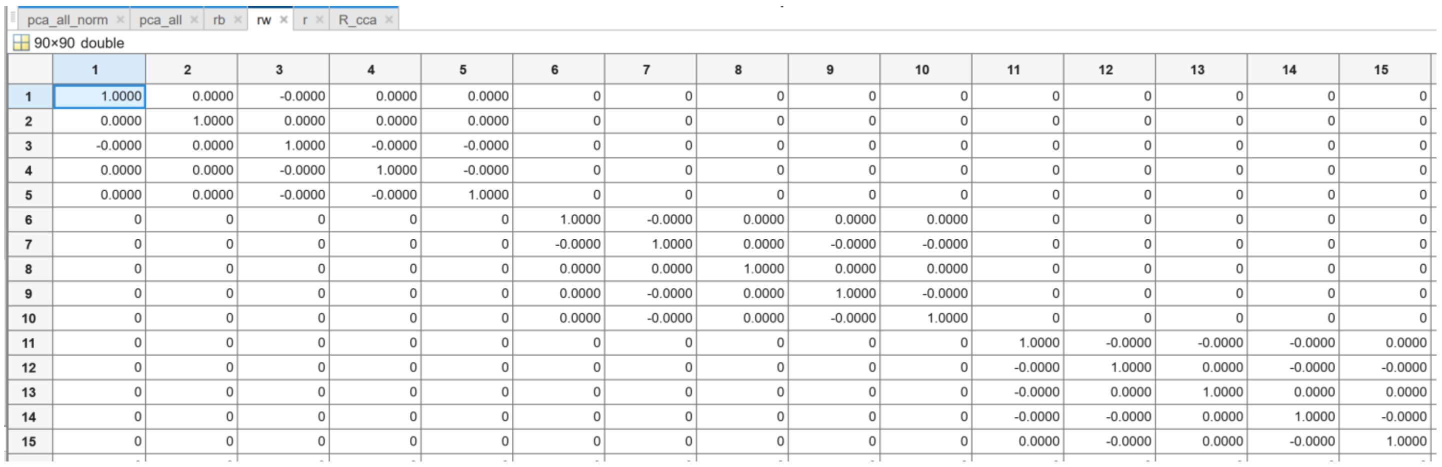
The tables below show the structure (partially) of the matrices created during the Multiset Canonical Analysis.

Table 1: Multisubject Cross-Covariance Block Matrix, R



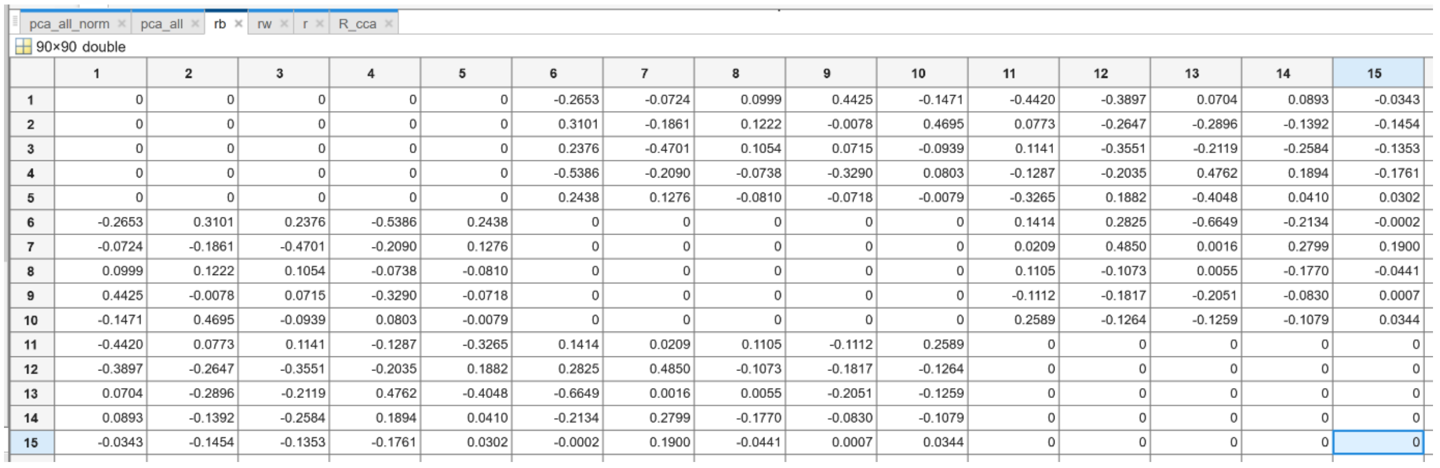
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.0000	0.0000	-0.0000	0.0000	0.0000	-0.2653	-0.0724	0.0999	0.4425	-0.1471	-0.4420	-0.3897	0.0704	0.0893	-0.0343
2	0.0000	1.0000	0.0000	0.0000	0.0000	0.3101	-0.1861	0.1222	-0.0078	0.4695	0.0773	-0.2647	-0.2896	-0.1392	-0.1454
3	-0.0000	0.0000	1.0000	-0.0000	-0.0000	0.2376	-0.4701	0.1054	0.0715	-0.0939	0.1141	-0.3551	-0.2119	-0.2584	-0.1353
4	0.0000	0.0000	-0.0000	1.0000	-0.0000	-0.5386	-0.2090	-0.0738	-0.3290	0.0803	-0.1287	-0.2035	0.4762	0.1894	-0.1761
5	0.0000	0.0000	-0.0000	-0.0000	1.0000	0.2438	0.1276	-0.0810	-0.0718	-0.0079	-0.3265	0.1882	-0.4048	0.0410	0.0302
6	-0.2653	0.3101	0.2376	-0.5386	0.2438	1.0000	-0.0000	0.0000	0.0000	0.0000	0.1414	0.2825	-0.6649	-0.2134	-0.0002
7	-0.0724	-0.1861	-0.4701	-0.2090	0.1276	-0.0000	1.0000	0.0000	-0.0000	-0.0000	0.0209	0.4850	0.0016	0.2799	0.1900
8	0.0999	0.1222	0.1054	-0.0738	-0.0810	0.0000	0.0000	1.0000	0.0000	0.0000	0.1105	-0.1073	0.0055	-0.1770	-0.0441
9	0.4425	-0.0078	0.0715	-0.3290	-0.0718	0.0000	-0.0000	0.0000	1.0000	-0.0000	-0.1112	-0.1817	-0.2051	-0.0830	0.0007
10	-0.1471	0.4695	-0.0939	0.0803	-0.0079	0.0000	-0.0000	0.0000	-0.0000	1.0000	0.2589	-0.1264	-0.1259	-0.1079	0.0344
11	-0.4420	0.0773	0.1141	-0.1287	-0.3265	0.1414	0.0209	0.1105	-0.1112	0.2589	1.0000	-0.0000	-0.0000	-0.0000	0.0000
12	-0.3897	-0.2647	-0.3551	-0.2035	0.1882	0.2825	0.4850	-0.1073	-0.1817	-0.1264	-0.0000	1.0000	0.0000	-0.0000	-0.0000
13	0.0704	-0.2896	-0.2119	0.4762	-0.4048	-0.6649	0.0016	0.0055	-0.2051	-0.1259	-0.0000	0.0000	1.0000	0.0000	0.0000
14	0.0893	-0.1392	-0.2584	0.1894	0.0410	-0.2134	0.2799	-0.1770	-0.0830	-0.1079	-0.0000	-0.0000	0.0000	1.0000	-0.0000
15	-0.0343	-0.1454	-0.1353	-0.1761	0.0302	-0.0002	0.1900	-0.0441	0.0007	0.0344	0.0000	-0.0000	0.0000	-0.0000	1.0000

Table 2: Multi-Intrasubject Auto-Covariance Diagonal Block Matrix, R_W


pca_all_norm
pca_all
rb
rw
r
R_cca
90×90 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.0000	0.0000	-0.0000	0.0000	0.0000	0	0	0	0	0	0	0	0	0	0
2	0.0000	1.0000	0.0000	0.0000	0.0000	0	0	0	0	0	0	0	0	0	0
3	-0.0000	0.0000	1.0000	-0.0000	-0.0000	0	0	0	0	0	0	0	0	0	0
4	0.0000	0.0000	-0.0000	1.0000	-0.0000	0	0	0	0	0	0	0	0	0	0
5	0.0000	0.0000	-0.0000	-0.0000	1.0000	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1.0000	-0.0000	0.0000	0.0000	0.0000	0	0	0	0	0
7	0	0	0	0	0	-0.0000	1.0000	0.0000	-0.0000	-0.0000	0	0	0	0	0
8	0	0	0	0	0	0.0000	0.0000	1.0000	0.0000	0.0000	0	0	0	0	0
9	0	0	0	0	0	0.0000	-0.0000	0.0000	1.0000	-0.0000	0	0	0	0	0
10	0	0	0	0	0	0.0000	-0.0000	0.0000	-0.0000	1.0000	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1.0000	-0.0000	-0.0000	-0.0000	0.0000
12	0	0	0	0	0	0	0	0	0	0	-0.0000	1.0000	0.0000	-0.0000	-0.0000
13	0	0	0	0	0	0	0	0	0	0	-0.0000	0.0000	1.0000	0.0000	0.0000
14	0	0	0	0	0	0	0	0	0	0	-0.0000	-0.0000	0.0000	1.0000	-0.0000
15	0	0	0	0	0	0	0	0	0	0	0.0000	-0.0000	0.0000	-0.0000	1.0000

Table 3: Multi-Intersubject Cross-Covariance Hollow Block Matrix, R_B


pca_all_norm
pca_all
rb
rw
r
R_cca
90×90 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	-0.2653	-0.0724	0.0999	0.4425	-0.1471	-0.4420	-0.3897	0.0704	0.0893	-0.0343
2	0	0	0	0	0	0.3101	-0.1861	0.1222	-0.0078	0.4695	0.0773	-0.2647	-0.2896	-0.1392	-0.1454
3	0	0	0	0	0	0.2376	-0.4701	0.1054	0.0715	-0.0939	0.1141	-0.3551	-0.2119	-0.2584	-0.1353
4	0	0	0	0	0	-0.5386	-0.2090	-0.0738	-0.3290	0.0803	-0.1287	-0.2035	0.4762	0.1894	-0.1761
5	0	0	0	0	0	0.2438	0.1276	-0.0810	-0.0718	-0.0079	-0.3265	0.1882	-0.4048	0.0410	0.0302
6	-0.2653	0.3101	0.2376	-0.5386	0.2438	0	0	0	0	0	0.1414	0.2825	-0.6649	-0.2134	-0.0002
7	-0.0724	-0.1861	-0.4701	-0.2090	0.1276	0	0	0	0	0	0.0209	0.4850	0.0016	0.2799	0.1900
8	0.0999	0.1222	0.1054	-0.0738	-0.0810	0	0	0	0	0	0.1105	-0.1073	0.0055	-0.1770	-0.0441
9	0.4425	-0.0078	0.0715	-0.3290	-0.0718	0	0	0	0	0	-0.1112	-0.1817	-0.2051	-0.0830	0.0007
10	-0.1471	0.4695	-0.0939	0.0803	-0.0079	0	0	0	0	0	0.2589	-0.1264	-0.1259	-0.1079	0.0344
11	-0.4420	0.0773	0.1141	-0.1287	-0.3265	0.1414	0.0209	0.1105	-0.1112	0.2589	0	0	0	0	0
12	-0.3897	-0.2647	-0.3551	-0.2035	0.1882	0.2825	0.4850	-0.1073	-0.1817	-0.1264	0	0	0	0	0
13	0.0704	-0.2896	-0.2119	0.4762	-0.4048	-0.6649	0.0016	0.0055	-0.2051	-0.1259	0	0	0	0	0
14	0.0893	-0.1392	-0.2584	0.1894	0.0410	-0.2134	0.2799	-0.1770	-0.0830	-0.1079	0	0	0	0	0
15	-0.0343	-0.1454	-0.1353	-0.1761	0.0302	-0.0002	0.1900	-0.0441	0.0007	0.0344	0	0	0	0	0

6.3 Bump and Flat Kernels

The bump kernel is given by a half-sine wave, where the amplitude, **A**, is fixed as 1, phase shift, $\boldsymbol{\varphi}$, is set as 0, angular frequency, $\boldsymbol{\omega} = 2\pi$ f, using a frequency of 10Hz, and radians of π (half sine). Hence, the final bump kernel is calculated as

$$h_1(t) = sine(t, \omega, \phi, A) = A * sine(\omega t + \phi)$$

$$\boxed{h_1(t) = sine(t) = sine\ (\pi * 10 * t)}$$

The flat kernel is given by a gamma distribution where the shape parameter, $\boldsymbol{k}$, is fixed as 2 and the scale parameter $\boldsymbol{\theta}$ is estimated during the model fitting.

$$h_2(t) = gamma(t, k, \theta) = \frac{t^{k-1} * e^{-\frac{t}{\theta}}}{\theta^k * \Gamma(k)}$$

$$\boxed{h_2(t) = gamma(t, 2, \theta) = \frac{t * e^{-\frac{t}{\theta}}}{\theta^2 * \Gamma(2)}}$$

6.4 Maximum Durations of Trials

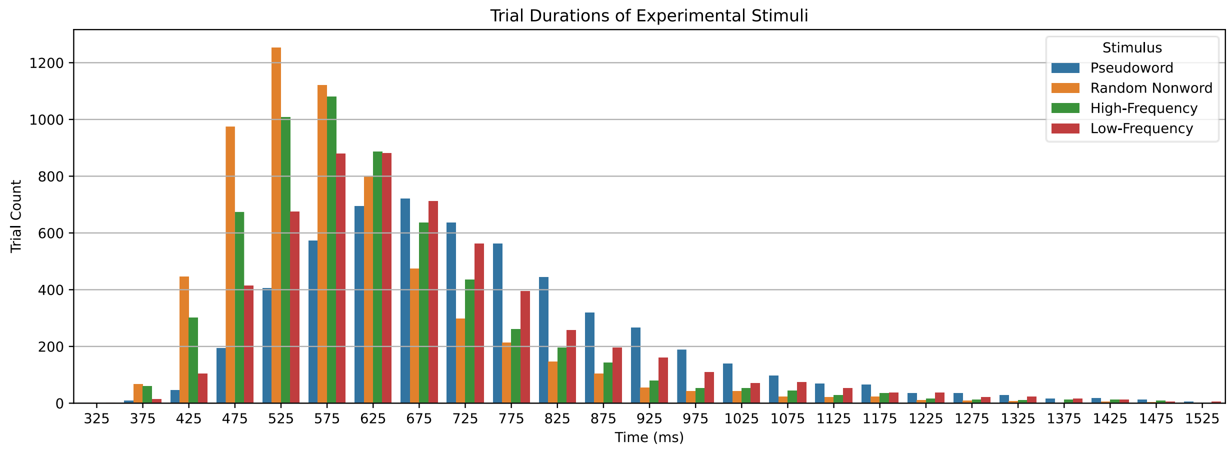


Figure A4: Maximum durations of the trials were grouped in 50 ms time windows for better visualization in the plot. For example. yellow bar at 525 ms shows the total number of trials with 'Random Nonword' as its experimental stimulus having a maximum duration of 500 – 500 ms (which is above 1200 trials). Overall plot shows that the trials of the 'High Frequency' and 'Random Nonword' stimuli tend to have short durations compared to the trial durations of the 'Low Frequency' and 'Pseudoword' stimuli. Read the plot as "*number of trials (y-axis) with a maximum duration of **t** ms (x-axis) belonging to stimuli **x** (see legend)*".

6.5 Stimuli Classification (Time Samples) of Preprocessed EEG Data

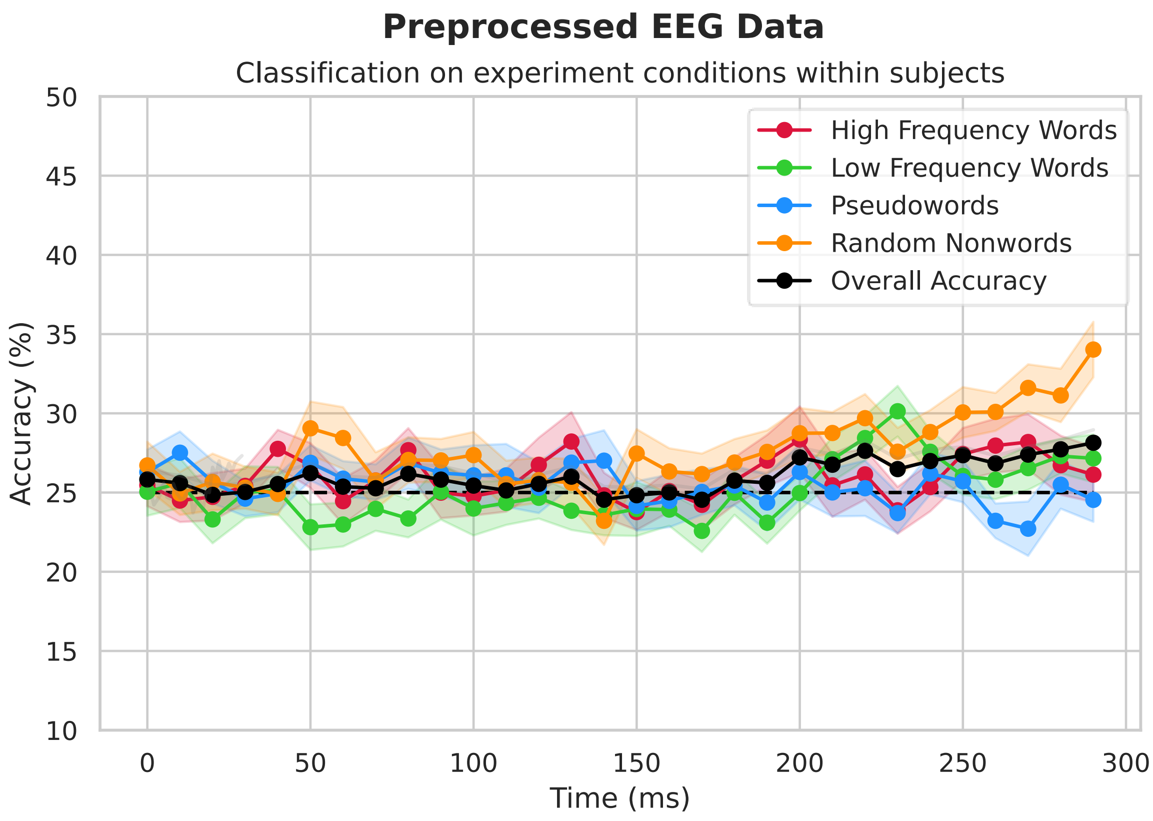


Figure A5: Stimuli classification results obtained by training (and predicting) the Ridge model separately over the data samples available at every time point for the first 300 ms.

6.6 Correlation of EEG Electrodes

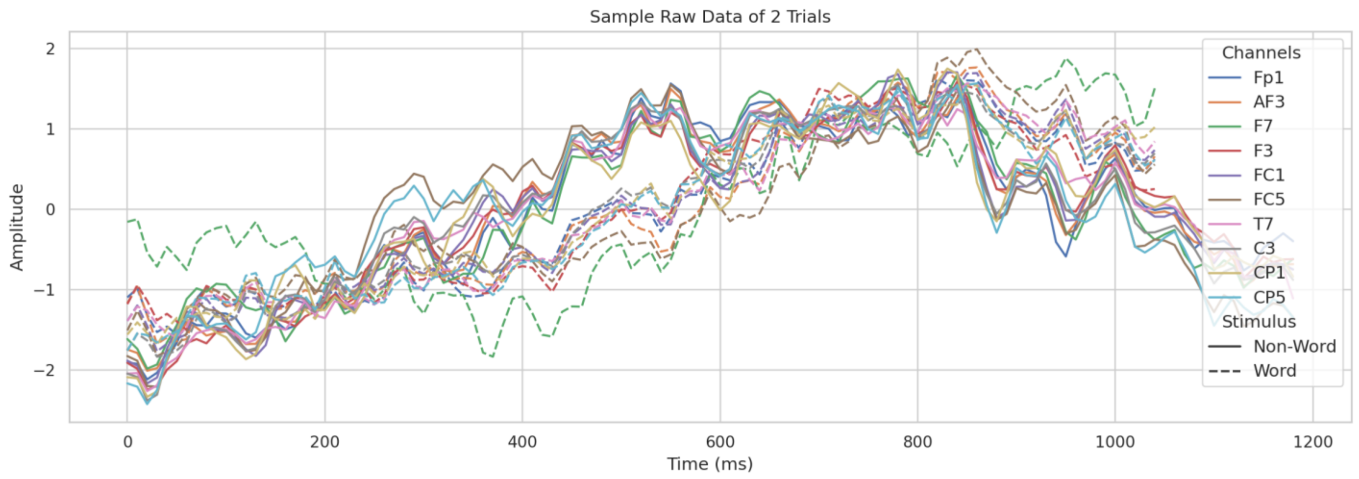


Figure A2: EEG data of 10 electrodes from two random trials showing the correlation between the electrodes. Electrodes placed closer to each other were picked for a clear illustration of the correlation.

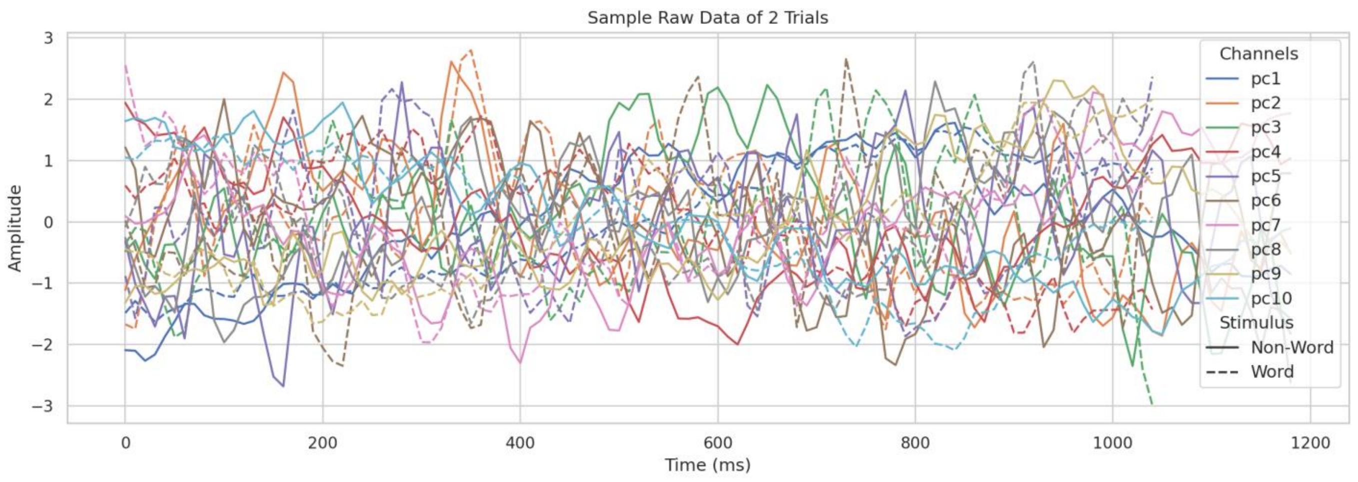


Figure A3: Illustration of PCA orthogonality using 10 principal components from the two trials.