# Learning Dynamic Regimes of Event-Based Substructures In EEG Data Using Graph Kernel Koopman Embedding

by

Rashmi Nagawara Muralinath

Bachelor of Engineering, Visvesvaraya Technological University, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

| | |
|---|---|
| **Supervisor:** | Prabhat K. Mahanti, Ph.D., Computer Science |
| **Examining Board:** | Janet Light, Ph.D., Computer Science, Chair |
| | Jong-Kyou Kim, Ph.D., Computer Science |
| | Dongmin Kim, Ph.D., Faculty of Business |

This thesis is accepted by the Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

January, 2025

© Rashmi Nagawara Muralinath, 2025

# Abstract

Understanding brain activity requires analyzing EEG data, which is challenging due to the high noise levels, non-linearity, non-stationarity, and individual variability. This thesis introduces a novel methodology using Graph Kernel Koopman Embedding (GKKE) methodology by representing time-evolving brain connectivity as low-dimensional, meta-stable regimes. The study focuses on two critical applications: detecting epileptic seizures(CHB-MIT dataset) and assessing cognitive workload (Cognitive Mental Workload dataset).

This research attempts to classify cognitive and neurological states using various combinations of connectivity measures, graph kernels, and classifiers. The results demonstrate that the method has a good classification accuracy of above 85% for both datasets, thus demonstrating its potential to identify intricate patterns. The suggested method involves preprocessing the raw EEG data through which the connectivity matrix is obtained by calculating corelation coefficients and generating gram matrices. Next, we use kernel PCA to simplify the graph features by reducing their dimensions. After that, we test how well they work with machine learning classifiers.

# Acknowledgments

I am deeply grateful to my supervisor, Dr. Prabhat Mahanti, for his invaluable guidance, encouragement, and unwavering support throughout this research. His mentorship has been pivotal in enabling me to explore and contribute to the study of epileptic seizures.

I also extend my appreciation to Dr. Vishwambhar Pathak, Assistant Professor in the Department of Computer Science & Engineering at Birla Institute of Technology Mesra Jaipur Off-Campus, India, for his insightful inputs and collaborative efforts, which significantly contributed to shaping and publishing my research findings.

I sincerely thank the University of New Brunswick for granting me this invaluable opportunity, providing essential resources, and fostering an environment that encouraged academic growth and innovation.

Lastly, I thank my family for their constant support and encouragement.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| EEG | Electroencephalogram |
| MRI | Magnetic Resonance Imaging |
| CT | Computed Tomography |
| PET | Positron Emission Tomography |
| GKKE | Graph Koopman Kernel Embedding |
| SPH | Seizure Prediction Horizon |
| SVC | Support Vector Classifier |
| WL | Weisfeiler-Lehman Kernel |
| DNN | Deep Neural Networks |
| CNN | Convolutional Neural Networks |
| RNN | Recurrent Neural Network |
| SVM | Support Vector Machines |
| PCA | Principal Component Analysis |
| BCI | Brain-Computer Interface |
| mPLV | Mean Phase Lock Value |
| CHB | Children's Hospital Boston |
| MIT | Massachusetts Institute of Technology |

# Chapter 1

# Introduction

Analyzing Electroencephalogram (EEG) data is pivotal for understanding the intricate workings of the brain. EEG is a non-invasive method to track brain activity over time, helping us understand how the brain works. However, analyzing EEG data is difficult because it is complex, has many dimensions, and changes over time. Traditional models fall short in capturing the dynamic and non-linear patterns within EEG signals, which are essential for understanding brain connectivity and state transitions.

This thesis introduces a novel method called Graph Kernel Koopman Embedding (GKKE) to address these challenges in EEG data analysis. The GKKE framework leverages graph-based models to represent dynamic brain connectivity, providing a structured and scalable approach to analyze and classify different brain states. By mapping EEG data onto graph structures, this method enables the exploration of relationships and patterns that are not easy to detect using other traditional techniques. Advanced graph kernels, such as Weisfeiler-Lehman, Spectral Decomposition, and Random Walk kernels, play a central role in identifying stable patterns within EEG signals, significantly enhancing classification accuracy.

The study achieves robust performance by combining graph-based connectivity measures, advanced graph kernels, and machine learning classifiers, including Decision Trees, Random Forests, and Support Vector Classifiers (SVC). Additionally, cost-sensitive learning techniques are employed to optimize accuracy, especially in imbalanced datasets. These innovations not only enhance classification outcomes but also pave the way for practical applications in diagnosing neurological and cognitive disorders.

This work holds profound implications for neuroscience research and clinical applications, as it enables the analysis of EEG data across diverse datasets and patient populations. By uncovering complex brain activity patterns, the GKKE framework contributes to a deeper understanding of brain functionality, fostering advancements in both fundamental neuroscience and applied domains [1] such as neuro-rehabilitation, brain-computer interfaces, and cognitive diagnostics.

## 1.1 Problem Statement

Deep learning and machine learning have helped the analysis of complex data, including EEG signals, by enabling advanced prediction and classification tasks. Machine learning methods excel at identifying patterns in large datasets, even when the data is noisy or non-linear. In the context of EEG analysis, these techniques have improved the prediction of brain activity by focusing on synchronizations across different frequency bands. However, challenges remain in interpreting the extracted features due to the variability and complexity of EEG data.

Many existing approaches rely heavily on statistical properties while overlooking the dynamic structures that emerge from the temporal evolution of EEG signals [2]. This limitation hinders their effectiveness in standard tasks such as pattern recognition,

dimensionality reduction, and feature learning for time-series data with complex dynamic features, often leading to suboptimal results [3, 4]. The Koopman operator offers a promising solution.

Koopman Operator is a mathematical framework that, while linear, can capture the evolution of non-linear systems by embedding them into an infinite-dimensional space. Finite approximations of the Koopman operator allow for predicting the evolution of intrinsic functions [5], making it a powerful tool for understanding dynamic data.

This thesis introduces a novel methodology called Graph Kernel Koopman Embedding (GKKE). By integrating graph kernels with the Koopman operator, GKKE provides a robust way to model time-evolving brain connectivity, focusing on metastable states for learning dynamic connectivity patterns in EEG data.

To validate the proposed method, GKKE is applied to datasets from two publicly available repositories. The first is the CHB-MIT Scalp EEG Database hosted on PhysioNet [6], which contains EEG recordings from epilepsy patients. The second is the Cognitive Mental Workload EEG Dataset, available on GitHub [7], which includes EEG data related to cognitive workload tasks. These datasets demonstrate the versatility and effectiveness of GKKE in addressing both clinical and cognitive applications in EEG analysis.

The thesis is organized into five chapters, each providing a structured exploration of the research process and findings. Below is an overview of each chapter:

- **Chapter 2: Background**

  This chapter lays the foundation for the study by providing an overview of key concepts and topics relevant to the research. It begins with an introduction to epileptic seizures, including their clinical significance and the challenges associated with their detection and analysis using EEG data.

- **Chapter 3: Methodology**

  This chapter details the research methodology employed in the study, offering a step-by-step explanation of the approaches and techniques used. It begins by describing the datasets selected for the experiments, including the CHB-MIT Scalp EEG Database and the Cognitive Mental Workload EEG Dataset.

- **Chapter 4: Results**

  This chapter presents the outcomes of the experiments conducted using the GKKE methodology on the selected datasets.

- **Chapter 5: Conclusion**

  The final chapter discusses the broader implications of the research findings for neuroscience and clinical applications. It highlights the key contributions of the study, such as improved classification accuracy and insights into brain connectivity patterns. The chapter also outlines potential directions for future work, including extending the methodology to other neurological disorders, exploring alternative graph kernel functions, and integrating deep learning models with GKKE for enhanced performance.

# Chapter 2

# Background

This chapter provides a detailed overview of the foundational concepts necessary to understand the study which includes epileptic seizures, including their causes, characteristics, and the importance of early detection and monitoring in medical research. Also introduces Koopman embedding and kernel functions, a key concept in machine learning and data analysis.

## 2.1 Epilepsy and EEG

The term *epilepsy* originated from the Latin and Greek word *epilepsia* which means 'seizure' or 'to seize upon.' It is a severe neurological disorder with unique characteristics, tending to recurrent seizures caused by abnormal brain electrical activity [8, 9].

According to the World Health Organization, epilepsy affects approximately 50 million people worldwide. It is defined by the frequent occurrence of seizures, which result from sudden, excessive surges of electrical activity in clusters of brain cells [10]. These discharges can originate in various areas of the brain [11], leading to different types of seizures.

Seizures vary significantly in severity and duration. They may manifest as brief lapses of attention or mild muscle twitches, while others can result in intense, prolonged convulsions. The frequency of seizures also varies widely among individuals, ranging from less than once a year to several times a day [12]. These variations underscore the complexity of epilepsy and highlight the need for effective tools to monitor and analyze brain activity, such as EEG.

Different diagnostic tools are available, such as magnetic resonance imaging (MRI), computed tomography (CT) scan, positron emission tomography (PET), ultrasound, and Electroencephalogram (EEG). MRI, CT, and Ultrasound are costly and cannot be used for long-term evaluation. On the other hand, EEG is a low-cost and non-invasive tool that can be used for long-term evaluation. Therefore, EEG is the most helpful tool for diagnosing epilepsy [13].

During an EEG, electrodes are attached to the scalp of the brain via adhesive gel. If a person has epilepsy, their EEG recording and EEG pattern will be different from standard patterns. EEG plays a crucial role in epilepsy detection by measuring voltage fluctuations across scalp electrodes, capturing ionic currents in neurons to provide both temporal and spatial insights into brain activity. Beyond its use in initial diagnosis, EEG is also essential in surgical planning for epilepsy treatment [14]. EEG is also used for continuous monitoring to evaluate treatments' effectiveness and identify subclinical seizures that may occur without apparent physical symptoms.

## 2.2 Seizure Phases and Neuro Markers

The events observed in the EEG can be grouped into three categories: ictal events, which occur during a seizure; preictal events, which occur before a seizure; and interictal events, which are all other events that are not part of the ictal or preictal phase. The duration of the preictal phase can vary widely [15, 8], from a few minutes to several hours.

Scientists have been studying how different brain parts communicate to detect epileptic seizures. They've found interesting patterns where, during seizures, specific brain regions become more active and communicate with each other. Also, they observed that the focus of activity could shift from one area to another, and the intensity of brain signals could change in specific frequency ranges. These patterns help us identify when a seizure might occur, thus helping us understand brain communication, which helps detect and predict epileptic seizures [16].

In neuroscience, a biomarker is a measurable biological characteristic or indicator that provides information about the state or function of the nervous system. Neuromarkers play a crucial role in understanding neurological disorders, monitoring disease progression, assessing treatment efficacy, and predicting clinical outcomes.

Sophisticated computational methods facilitate the exploitation of dynamic properties of epileptiform waveforms which play a crucial role in the diagnosis and management of epilepsy to quantify and characterize seizure dynamics suitably employing methods of multivariate time series analysis of bio-physiological signals. Epileptiform waveforms also provide important diagnostic information, guide treatment decisions, and monitor response to therapy [16].

## 2.3 Machine Learning in Seizure Detection

Machine learning has transformed medical science by offering advanced tools to process complex data and improve patient care. Using data-driven techniques, it enables predictions, supports disease diagnosis, tailors treatment plans to individual needs, and streamlines healthcare management. Its applications range from predicting disease outbreaks and analyzing medical images to improving treatment strategies.

Machine learning has extensively been used for disease prediction and detection. ML has shown promise, offering potential benefits in terms of early intervention, improved patient care, and a better understanding of epilepsy. Epilepsy, characterized by repeated seizures due to irregular brain activity, needs prompt and accurate diagnosis for proper management. Conventional methods of detecting seizures, which rely on manually analyzing EEG signals, are time-consuming and can be error-prone [17].

Epilepsy, marked by recurrent seizures caused by abnormal brain activity, requires timely and accurate diagnosis for effective management. Traditional seizure detection methods, which involve manually reviewing EEG signals, are labor-intensive and prone to error. ML offers automated, efficient, and accurate solutions to overcome these challenges.

Finding an answer to "which brain portions cause epileptic seizure" requires causal models capable of estimating spatial (between brain regions) and temporal (spread of effect in time) relations in different frequency bands of the multichannel signals. Synchronizations of certain segregated brain-regions do happen during and between epileptic seizures [16].

The alternative workflows harnessing the Koopman operators for epileptic seizures are characterized by abnormal, synchronized neuronal activity. The changing connections in the brain are key to how seizures start and spread. Recent research highlights the importance of studying how these connections change before a seizure, as this could offer important clues about what triggers seizures [18, 5, 19].

There are some available research work already done using Deep Neural Networks (DNNs) which learn the features automatically, even in a noise-invariant manner, and thus release from the expensive data-preprocessing and feature extraction steps required in traditional machine learning models. DNN models employing combinations of CNN, RNN, and auto encoders have been applied to the problems of EEG classification, speech recognition, motor-event recognition, language recognition and prediction, etc. [16, 20, 21].

Another research team, they focused on a single channel extracted from multichannel EEG signals, [22] which are filtered and divided into short-duration segments. The approach utilized three deep learning algorithms—stacked autoencoder (SA), recurrent neural network (RNN), and convolutional neural network (CNN)—for classification. Their findings revealed that the CNN model outperformed the other classifiers.

In one other proposed method for diagnosing epileptic seizures, the research was based on nonlinear dynamic features [23]. In this approach, discrete wavelet transform (DWT) is employed for pre-processing and noise removal. The EEG signals are characterized using nonlinear dynamic features, and a support vector machine (SVM) is utilized to classify these features and detect epileptic seizures.

A significant amount of research has been conducted on diagnosing epileptic seizures using EEG signals and machine learning techniques, with new approaches emerging each year to address this challenge [24]. Prior studies have explored various biomarkers, genetic factors, and imaging modalities in the context of epilepsy prediction [14].

However, there is still a need for detailed investigations that integrate different data sources, considering the complex nature of epilepsy. By addressing the limitations of current methods through the proposed GKKE framework, this research seeks to find a new path for EEG data analysis [25].

### 2.3.1 Kernel Functions, Kernel Methods, Graph Kernels

In many domains, describing relations between objects or individuals cannot be interpreted as vectors or fixed grids, instead; they are naturally represented by graphs. Hence, it becomes necessary for algorithms to exploit the rich information inherent to the graph structure and annotations associated with their vertices and edges [26, 27].

#### Graph Overview

A graph G, could be defined as a triplet $(V, E, \ell)$, where V is the set of vertices, E is the set of undirected edges, and $\ell : V \to \Sigma$ is a function that assigns labels from an alphabet $\Sigma$ to nodes in the graph [28].

The **neighbourhood** N (v) of a node v is the set of nodes to which v is connected by an edge, that is: $N(v) = v'|(v, v') \in E$.
Assuming that every graph has n nodes, m edges, and a maximum degree of d. The size of G is defined as the **cardinality** of V.

A **walk** in a graph is defined as a sequence of nodes where an edge connects each consecutive pair of nodes. On the other hand, a **path** is a walk that includes only distinct nodes. A **subtree** of a graph G can be described as a connected subset of different nodes in G, structured as a tree. The **height** of a subtree refers to the most significant distance from its root to any other node.

While a walk generalizes the concept of a path by allowing nodes to repeat, the idea of subtrees can similarly be broadened to include subtree patterns (also referred to as tree walks). These patterns permit nodes to appear multiple times, treating each repetition as a unique instance, thereby maintaining a cycle-free tree structure. Importantly, subtree kernels are designed to compare subtree patterns across two graphs rather than strictly comparing subtrees.

**Kernel Functions**: They measure similarity between data points in a higher dimensional feature space without explicitly transforming the data into space. Kernel-based methods are machine learning algorithms that learn by comparing any pair of data points using similarity measures called **kernel functions** [29]. Some of the kernel functions used in many kernel-based methods is the Gaussian Kernel, Linear Kernel, Polynomial Kernel etc.

Mathematically, a kernel function $K(x.x')$ computes the inner product of the feature space mappings of two input vectors x and x' without needing to compute the actual transformation $\theta(x)$ or $\theta(x')$. This allows machine learning algorithms to operate efficiently in high-dimensional spaces without explicitly performing the transformations.

In practical applications, kernel functions are extensively used to find optimal hyperplanes for classification tasks in high-dimensional spaces, enabling the separation of data that is not linearly separable in the original space. They are also employed in Kernel Principal Component Analysis (kernel PCA) for non-linear dimensionality reduction and various clustering algorithms to capture complex relationships within data [30].

**Kernel Methods**: Kernel methods refer to machine learning algorithms that learn by comparing pairs of data points using particular similarity measures—kernels.

Kernel methods have been developed for most machine learning paradigms, e.g., Support Vector Classifier (SVC) for classification, Gaussian processes (GP) for regression, kernel PCA, k-means for unsupervised learning and clustering, and kernel density estimation (KDE) for density estimation [31].

An essential concept in kernel methods is the Gram matrix K, defined concerning a finite set of data points $x_1, ..., x_m \in X$.

The **Gram matrix** of a kernel k has elements $K_{ij}$, for $i, j \in \{0, ..., m\}$ equal to the kernel value between pairs of data points, i.e., $K_{ij} = k(x_i, x_j)$.
Examples of a few Kernel methods are SVC, Kernel PCA, Gaussian Process, etc. Kernel methods allow for the application of linear techniques in non-linear domains, improving algorithms' ability to capture complex patterns in the data.

**Graph Kernels**:
In the graph space G, a graph is denoted as G = (V, E), where V is the set of nodes and E is the edge set of G. Given two graphs $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$ in G, the graph kernel $K_G(G_i, G_j)$ measures the similarity between them.

Graph kernels have become an established and widely used technique for solving classification tasks on graphs. Graph kernels are kernels defined on graphs to capture graph similarity, which can be used in kernel methods—such as support vector machines (SVM) and principal component analysis (PCA)—for graph classification. This capability is particularly valuable in bioinformatics, chemoinformatics, and social network analysis, where data is inherently graph-structured.

A popular approach to learning with graph-structured data is to use graph kernel functions that measure the similarity between graphs plugged into a kernel machine [31]. In the past 15 years, numerous graph kernels have been proposed, motivated either by their theoretical properties or by their suitability and specialization to particular application domains [32].

Several different graph kernels [33] have been defined in machine learning which can be categorized into three classes: graph kernels based on walks and paths like Random Walk Kernels and Shortest Path Kernels, graph kernels based on limited-size subgraphs like Graphlet Kernels, and graph kernels based on subtree patterns like Weisfeiler-Lehman (WL) Subtree Kernel [28] and Tree-Walk Kernels.

### 2.3.2 Koopman Embedding

Graph theory has proven invaluable in modeling and analyzing brain connectivity networks. Researchers can efficiently quantify graph structures by employing graph kernels, such as node centrality or edge density. These methods have facilitated the identification of specific connectivity patterns associated with neurological conditions, as demonstrated in recent studies on epilepsy [34].

Koopman theory provides a theoretical foundation for understanding the evolution of observables in a dynamical system. Embedding techniques based on Koopman theory have shown promise in capturing the underlying dynamics of complex systems. Applying Koopman embedding to graph kernels enables the representation of evolving network structures in a high-dimensional space, allowing for the detection of subtle changes in connectivity patterns.

A comprehensive understanding of the challenges in analyzing time-evolving graphs in microbiome research illustrates the effectiveness of the GKKE approach in capturing the dynamics and metastable behavior by representing time-evolving graphs as fixed-length feature vectors [29]. GKKE combines the concepts of graph theory, machine learning and koopman operator theory to analyze brain connectivity.

The GKKE approach used in [29], combined with the Weisfeiler–Lehman graph kernel, outperformed other state-of-the-art methods in accurately identifying metastable states in both benchmark and real-world datasets for microbiome analysis done in paper [29]. It captured the temporary changes in the time-evolving graph. WL graph kernel in terms of runtime on large graphs, was shown to outperform other kernels, like random walk kernels and graphlet kernels [28] . The method used the Koopman operator to capture the temporal changes in the time-evolving graph. Koopman operator is an emerging methodology for data-driven identification of high-dimensional and nonlinear dynamical systems [35].

The paper [29] shows by approximating the eigenfunctions of transfer operators, the GKKE approach can learn a low-dimensional representation of the time-evolving graph that preserves the metastable behavior. It's effective in capturing temporary changes in the time-evolving graph, maintaining all dynamic properties in such a way that it is possible to detect metastable states in the low-dimensional space.

The GKKE approach extracts the substructures of a graph by learning the embedding of the time-evolving graph using the spectral analysis of transfer operators and graph kernels. The method is based on the approximation of eigenfunctions of transfer operators, such as the Koopman operator, using graph kernels. The low-dimensional representation obtained through GKKE preserves the original dynamics of the graph, allowing for the extraction of important substructures and the detection of distinct states within the graph.

## 2.4 Multivariate Time Series

A time series is a collection of observations made chronologically. The nature of time series data includes: large in data size, high dimensionality and update continuously. Moreover time series data, which is characterized by its numerical and continuous nature [36, 37], is always considered as a whole instead of individual values. It comprises multiple interdependent time-dependent variables recorded over a period. Unlike univariate time series, which analyze a single variable [38].

Analyzing multivariate time series data involves understanding temporal dependencies within each variable and cross-dependencies between variables. Multivariate time series data analysis is computationally intensive due to the high dimensionality of the data. The growing availability of high-resolution multivariate datasets, coupled with advances in machine learning, continues to expand the potential of multivariate time series data analysis for solving real-world problems [39].

### 2.4.1 CHB dataset

The CHB-MIT Scalp EEG Database, is a collection of EEG recordings of 22 subjects with intractable seizures. A team of investigators from Children's Hospital Boston (CHB) and the Massachusetts Institute of Technology (MIT) created and contributed this database to PhysioNet. The files named chbNN-summary.txt contain information about the dataset used for each recording, and the elapsed time in seconds from the beginning of each .edf file to the beginning and end of each seizure contained in it. The folder structure is shared as in figure 2.1 Only the Preictal and Ictal segments from the dataset are utilized for the experiments.


Figure 2.1: CHB Data Preictal-Ictal Information

### 2.4.2 Cognitive Workload dataset

The EEG DATA has been captured at BCI-HCI lab, the data captured is not tagged / marked due to privacy reasons. There are 5 experiments conduced and data with respect to each subject are considered. The dataset has 5 tasks per subject (Idle, 1-Back, 2-Back, Dual-1-Back, Dual-2-Back). The Idle and Dual-1-Back workload tasks are utilized from the dataset for the experiments.

The experiments were different cognitive tasks used to assess an individual's cognitive workload or mental effort. Idle condition serves as a baseline for comparison where the participant is not actively performing any task. The n-back tasks (1-back, 2-back) and dual-task conditions (dual-1-back, dual-2-back) represent progressively increasing levels of cognitive demand. Dual-task conditions test the participant's ability to process multiple streams of information, revealing insights into multitasking and divided attention.

# Chapter 3

# Methodology

This chapter explains the methods used in this study, including the steps taken to process and analyze the data.

The subsequent sections provide a step-by-step explanation, beginning with the data collection and preparation process. This is followed by a discussion of how advanced graph-based algorithms were applied to study the data and classify various brain states. The primary objective is to develop a reliable system capable of identifying essential patterns in EEG data and making accurate predictions.

## 3.1 Introduction

The proposed thesis workflow has been represented in Figure 3.1.

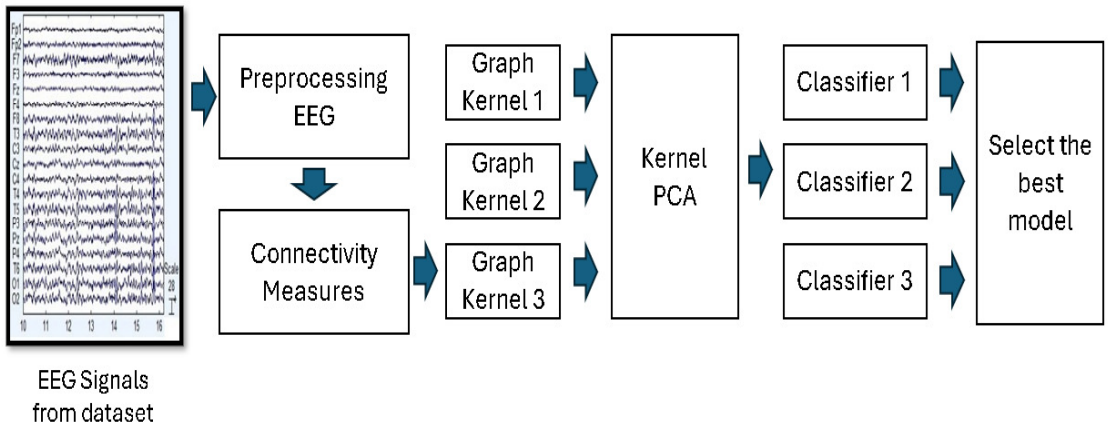


Figure 3.1: Methodology - Workflow

This workflow showcases processing EEG data to develop and select the best model for seizure detection. It utilizes Graph Kernels to generate Gram Matrices and obtain optimal clusters. The Raw EEG data collected from the datasets (CHB-MIT dataset and BCI-HCI dataset) serves as input.

Preprocessing is done on the dataset to clean and prepare the EEG data for analysis, including noise filtering which is further elaborated in next section. The structural or functional connectivity between EEG channels using correlation matrices and mPLV in the case of CHB dataset [6] and using correlation matrices for Cognitive Load dataset [7] was determined. The EEG channels are represented nodes, and the connections are weighted edges. Implementing the WL kernel, Random Walk kernel, and Spectral Decomposition kernel to capture the brain connectivity in Gram matrices was done. Kernel PCA is utilized to reduce the dimensionality of the EEG graph features. ML classifiers like Decision Tree, Support Vector Classifier(SVC), and Random Forest are applied to the reduced feature set. Each classifier is trained and evaluated on the data to determine performance. Finally, an optimal seizure detection model is selected based on the evaluation metrics like accuracy, precision, recall, and F1-score.

This thesis builds upon the foundational work on GraphKKE for Human Micro-

biome Analysis, where the researchers [29] investigated microbial communities over long periods, and a time-evolving graph describes the relationships between their members.

Inspired by their approach, this study applies a similar methodology to different datasets, specifically in terms of EEG data [29]. The key point was representing the time-evolving graphs as fixed-length feature vectors preserving the original dynamics. They proposed embedding the time-evolving graph based on the spectral analysis of transfer operators and graph kernels [29].

## 3.2 Data Collection and Preprocessing

This study obtained EEG data from two primary sources: the CHB-MIT Scalp EEG Database and the Cognitive Mental Workload EEG Dataset.

### 3.2.1 CHB-MIT Scalp EEG Database

The **CHB-MIT dataset** [6], available on PhysioNet, was developed by the Children's Hospital Boston and the Massachusetts Institute of Technology for the study of epilepsy and seizure prediction. It contains scalp EEG recordings from 23 pediatric patients with intractable seizures, recorded at a sampling rate of 256 Hz specified in the dataset.

The dataset contained multiple cases but the case considered here was chb02 which contains 35 continuous .edf (European Data Format ) files from a single subject. This dataset was selected for its high-quality annotations of seizure events, making it ideal for training and evaluating seizure detection models. Considering only the preictal and ictal phases of the EEG signals for the experiments [6].

### 3.2.2 Cognitive Mental Workload EEG Dataset

The **Cognitive load Dataset or BCI-HCI Dataset** [7] was obtained from the BCI-HCI (Brain-Computer Interface - Human-Computer Interaction) research group and is publicly available on GitHub. This dataset is designed to study cognitive mental workload during various tasks and provides EEG recordings from multiple subjects performing different cognitive activities.

The EEG data was captured at the BCI-HCI Lab, IIT Kharagpur, using the EMOTIV Epoc+ device. This device records from 14 sensors placed at locations such as AF3, AF4, F3, and others, with a sampling rate of 128 Hz was considered as per the guidelines about the data was provided. Data includes recordings from subjects labeled S01 to S05. Each dataset is a matrix (37 rows $\times$ M columns), where columns 3 to 16 correspond to 14 EEG channels.

Both datasets underwent preprocessing to filter out artifacts from noise ensuring consistent feature extraction for analysis. The raw data is also segmented after noise removal into time epochs and assigned labels. Later we calculated the connectivity between brain regions. An adjacency matrix is used as input features for the machine learning models. The adjacency matrix are constructed using correlation coefficients and mPLV, then are binarized over a threshold. This forms the basis for training and validating our Koopman kernel seizure detection algorithm.

## 3.3 Theoretical Background

### 3.3.1 Koopman Operator Theory, Eigendecomposition

The Koopman operator allows for handling nonlinear systems through a globally linear representation [26, 40]. The Koopman operator provides a framework for analyzing nonlinear dynamical systems by lifting them into an infinite-dimensional space where their evolution can be represented linearly. Instead of focusing on the state variables directly, the Koopman operator acts on observable functions of the system's state, transforming the nonlinear system into a linear operator acting on these observables [27, 5].

Its eigendecomposition offers valuable insights into the system's behavior by identifying invariant modes and their growth rates or frequencies.

Mathematical Representation: Given a dynamical system's state at time t, $x(t) \in \mathbb{R}^n$, a flow map $\phi^t$ defined as: $x(t+1) = \phi^t(x(t))$ governs the evolution of the state. The Koopman operator defined as in equation (1) maps the observable $g : \mathbb{R}^n \to \mathbb{R}$ to a new function that describes the evolution of g under the dynamics of the system.

$$Kg(x) = g(\phi^t(x)) \tag{3.1}$$

The eigendecomposition of the Koopman operator can be expressed as:
$Kg(x) = \lambda\phi(x)$ where $\phi$ represent the eigenfunctions and $\lambda$ represent the eigenvalues. For a set of eigenfunctions $\{\phi_i\}$, any observable $g$ can be expressed as a linear combination of these eigenfunctions:
$g(x) = \sum_i c_i\phi_i(x)$ where $c_i$ are coefficients.

### 3.3.2 Graph Kernels Overview

EEG data, when represented as functional connectivity graphs (where nodes are EEG channels and edges represent connectivity measures like correlation or mPLV), requires a robust method for comparison. Graph kernels serve as similarity functions, enabling efficient analysis and classification of these graphs [41].

Graph kernels capture structural properties and dynamics by producing eigenvalues and eigenvectors and comparing the spectra of graph-related matrices e.g., adjacency matrix, Laplacian matrix or other graph-related matrices. The popular graph kernels operate differently and there is a trade-off between accuracy performance and computational complexity and the optimal choice depends strongly on the dataset, as noted in [21] [32].

Several different graph kernels have been defined in machine learning which can be categorized into three classes: graph kernels based on walks and paths, graph kernels based on limited-size subgraphs, and graph kernels based on subtree patterns.

The first class, graph kernels on walks and paths, compute the number of matching pairs of random walks (respective paths) in two graphs. Example, the common walk kernel is based on the concept to compare all possible walks starting from all vertices in two. Another example would be of the shortest-Path Kernel, its built on the comparison of shortest paths between any pair of vertices in two graphs. A shortest-paths graph contains the same set of vertices as the original graph, while there is an edge between all vertices which is labeled by the shortest distance between these two vertices.

The second class, graph kernels based on limited-size subgraphs, includes kernels based on so called graphlets, which represent graphs as counts of all types of subgraphs.

In the third class of subtree kernels, to compare graphs G and G', this kernel iteratively compares all matching between neighbors of two nodes v from G and v' from G'. In other words, for all pairs of nodes v from G and v' from G', it counts all pairs of matching substructures in sub-tree patterns rooted at v and v'.

Traditional static brain networks cannot capture the dynamic nature of brain activity. In contrast, evolving brain networks—sequences of networks over time—can effectively represent these temporal changes. Graph kernel methods are particularly useful for assessing differences between networks, making them well-suited to analyze the evolving structures of brain connectivity. This approach holds significant potential for understanding the dynamic variations in brain function over time.

#### 3.3.2.1 Weisfeiler-Lehman Kernel

The Weisfeiler-Lehman (WL) kernel is a widely used graph-based technique in machine learning, designed to assess graph similarity, particularly in tasks like graph classification and clustering. It operates by iteratively refining node labels based on the labels of their neighbors, effectively encoding the structural characteristics of the graphs. This refinement generates a histogram of node labels across iterations, which is then compared to compute graph similarity and construct a Gram matrix. The WL kernel's ability to efficiently capture structural features makes it well-suited for large-scale graph learning applications [28].

#### 3.3.2.2 Random Walk Kernel

The Random Walk Kernel is a graph-based technique that evaluates graph similarity by counting the number of common random walks between two graphs. This method captures structural similarities by analyzing sequences of node transitions, making it applicable for tasks like graph classification and clustering. It is especially

effective at identifying local structural patterns. However, the kernel can become computationally expensive for large graphs, requiring the use of efficient algorithms for practical implementation [33].

#### 3.3.2.3 Spectral Decomposition Kernel

The Spectral Decomposition Kernel is a machine learning method used to analyze and break data down into its basic frequencies or components. It is especially useful for working with high-dimensional data and tasks involving manifold learning. This approach leverages the frequency characteristics of kernel matrices to uncover the hidden structure of the data.

Spectral decomposition means finding the eigenvalues and eigenvectors of the kernel matrix, which can then be used for reducing the number of dimensions, grouping data, or other types of analysis.

This approach is commonly applied in methods like Kernel PCA (Principal Component Analysis) and Diffusion Maps, which use spectral properties to reveal the data's intrinsic geometry [42].

### 3.3.3 Koopman Kernel Algorithm Implementation

Steps for the GKKE algorithm is presented below:

**Algorithm 1** GKKE for EEG best model classification

**Input:** EEG signals from dataset
**Output:** Best classification model
**procedure** EEG_Processing_Pipeline
**Step 1: Preprocess EEG Data**
Remove noise from raw EEG signals
Divide them into smaller Epochs
Assign labels
**Step 2: Compute Connectivity Measures**
Calculate connectivity matrices (e.g., using correlation or mPLV)
Binarize the generated matrix
Convert each to networkx graphs
**Step 3: Apply Graph Kernels**
**for** each graph kernel type $k_i \in \{$
$WeisfeilerLehmanGraphKernel$,
$SpectralDecompositionKernel$,
$RandomWalkKernel\}$ **do**
Compute graph kernel representation based on Gram Matrices $G_{k_i}$
**end for**
**Step 4: Dimensionality Reduction using Kernel PCA**
Apply Kernel PCA to graph kernel features for dimensionality reduction
**Step 5: Train Classifiers**
**for** each classifier $c_i \in \{DecisionTree, SVC, RandomForest\}$ **do**
Train $c_i$ on reduced feature set along with Cost-Sensitive Learning
**end for**
**Step 6: Model Selection**
Evaluate each classifier based on performance metrics (e.g., accuracy, precision etc)
Select the best-performing model
**end procedure**

### 3.3.4 Software and Packages Used

- Python
- jupyter IDE
- MNE package - Open-source Python package for exploring, visualizing, and analyzing human neurophysiological data
- Scikit-learn package - for working with Graph Kernels

### 3.3.5 Model Training and Evaluation

The systematic approach to training, tuning, and evaluating machine learning models—specifically, Support Vector Machines (SVC), Random Forests, and Decision Trees—for classifying EEG data are broken down below:

#### 3.3.5.1 Data Preparation and Splitting

The process begins with splitting the dataset into training and testing subsets using train_test_split function from Scikit-learn. The input data (X) and corresponding labels (y) are partitioned, with 20% reserved for testing. This split ensures the model's performance can be evaluated on unseen data, crucial for assessing generalization.

#### 3.3.5.2 Model Initialization and Parameter Tuning

Three classifiers are initialized: Support Vector Classifier (SVC), Random Forest Classifier (RF), Decision Tree Classifier (DT). Each model is paired with a parameter grid for hyperparameter tuning. For example: SVC parameters include regularization strength (C), kernel type, and gamma value. Random Forest and Decision Tree grids focus on the number of estimators, tree depth(max_depth), and minimum samples(min_samples_split) required for a split.

Cost-Sensitive Learning, is performed which takes the costs of prediction errors (and potentially other costs) into account when training the model. Python sklearn provides support for cost-sensitive learning for most baseline classifiers in the form of class_weight parameter. By strongly penalizing mistakes on the minority class, cost-sensitive learning helps improve their importance during the classifier training step and thus allows to improve generalization on the minority class.

GridSearchCV is used to exhaustively search through these combinations, optimizing for accuracy. The cross-validation (cv=5) ensures robustness by splitting the training data into five folds, training, and validating on each to reduce bias.

#### 3.3.5.3 Model Training and Hyperparameter Optimization

Each classifier undergoes training using GridSearchCV. This process identifies the best hyperparameters for each model by evaluating performance across the grid. And also stores the optimal configurations (best_svc, best_rf, best_dt) for subsequent evaluation.

#### 3.3.5.4 Model Evaluation Metrics

The models are assessed on the test set using Performance metrics:

Accuracy measures the overall correctness. Accuracy which is defined as the ratio between the correctly classified samples and the total number of samples in the evaluation dataset was above 90% for both datasets for the best performance dataset. Precision was evaluated and cross verified too. Precision which calculats the proportion of correctly predicted positive cases among total predicted positives. Confusion Matrix for the both datasets are also generated to further analyse the precision.
Recall, AUC and F1-Score to assesses the model's ability to identify all actual positives, provide a balanced measure of precision and recall, especially valuable for imbalanced datasets are also considered for evaluation of the model.

For probabilistic models like SVC and Random Forests, predict_proba is employed to output probability scores, enabling more detailed performance analysis.

With this model an accuracy of above 95% A.3 for Cognitive Load dataset [7] for the WL Kernel - RF Classifier combination and above of 90% A.1 for CHB dataset [6] for the WL Kernel - DT Classifier and mPLV as connectivity measure was achieved.

The results derived from these methods are presented in the following chapter, offering insights into the performance of the proposed GKKE framework EEG datasets.

# Chapter 4

# Results

This chapter presents the results of applying the methodology described in Chapter 3 to two distinct datasets: the CHB-MIT Scalp EEG Database and the Cognitive Mental Workload EEG Dataset.

## 4.1 Dataset 1: CHB - MIT Dataset

### 4.1.1 Understanding the Dataset

The raw EEG data was preprocessed to prepare it for analysis. First, the data was segmented into three categories: ictal (seizure), preictal (before seizure), and interictal (regular periods). However, for our study, only 30 epoch segments of ictal and preictal periods are considered as computational complexity was huge while entire dataset was included. The plot in figure 4.1 shows a comparison between preictal and ictal data for Channel 1, where the preictal data is shown in blue, and the ictal data is overlaid on top of it in orange. The x-axis shows time in samples, and the y-axis shows how strong the signal is. The overlay makes it easy to compare EEG activity before and during a seizure, showing the differences in signal patterns that help tell apart preictal (before a seizure) and ictal (during a seizure) states.

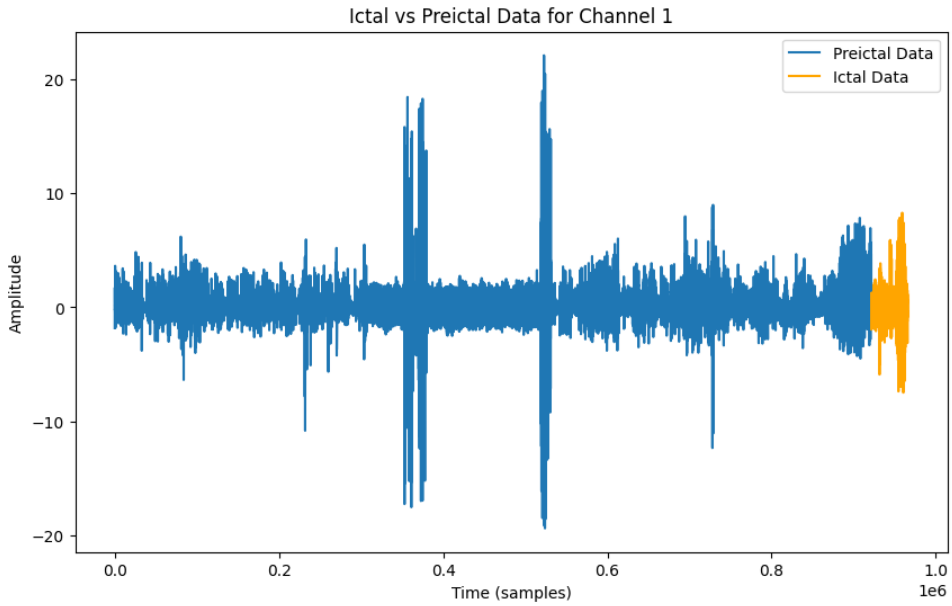


Figure 4.1: CHB Data Preictal-Ictal Information

In the EEG data, the channel names refer to the specific electrode placements on the scalp that are used to record electrical brain activity. These placements are typically standardized according to the 10-20 International System or its variants, which provide a consistent and reproducible method for electrode positioning. In the CHB dataset, it has 23 channels, as represented in Figure 4.2

The circular layout graph illustrates the connectivity between the EEG channels. The nodes represent individual EEG electrodes, and the edges denote significant connections based on the correlation or coherence between channels. The color-coded dots show where the electrodes are placed on the scalp. The scattered connections show that only a few electrodes are strongly linked, which could point to focused brain activity or specific network patterns. This helps find important groups or regions in the brain, which could be helpful for detecting seizures or analyzing brain states. It helps identify functional clusters or regions of interest, potentially highlighting areas critical for further seizure detection or brain state classification analysis.

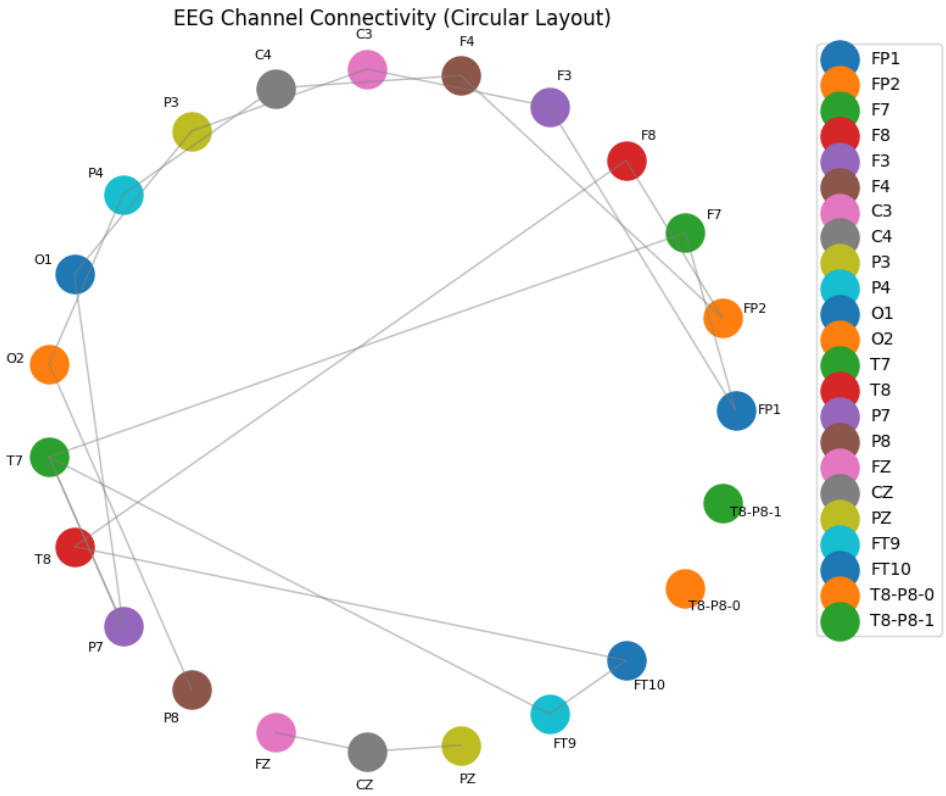


Figure 4.2: CHB Data Channel Details

Filters are applied to remove noise, including a notch filter to eliminate power line interference and a band-pass filter to focus on relevant brainwave frequencies. The data was modified to maintain consistency between patients and channels. It was then divided into 2-second segments, labeled as either preictal (before a seizure) or ictal (during a seizure), to build a training dataset for machine learning models. This step ensured the data was clean, uniform, and properly structured for analysis [16].

The mPLV generates the adjacency matrices as the connectivity measure represents different brain states: a preictal epoch (before a seizure) and an ictal epoch (during a seizure). The adjacency matrix was plotted in figure 4.3 to understand the sample preictal epoch's connectivity between various regions in the brain, using a heatmap to indicate the strength of connections. Similarly, the adjacency matrix for the ictal epoch displays the brain's connectivity during a seizure.

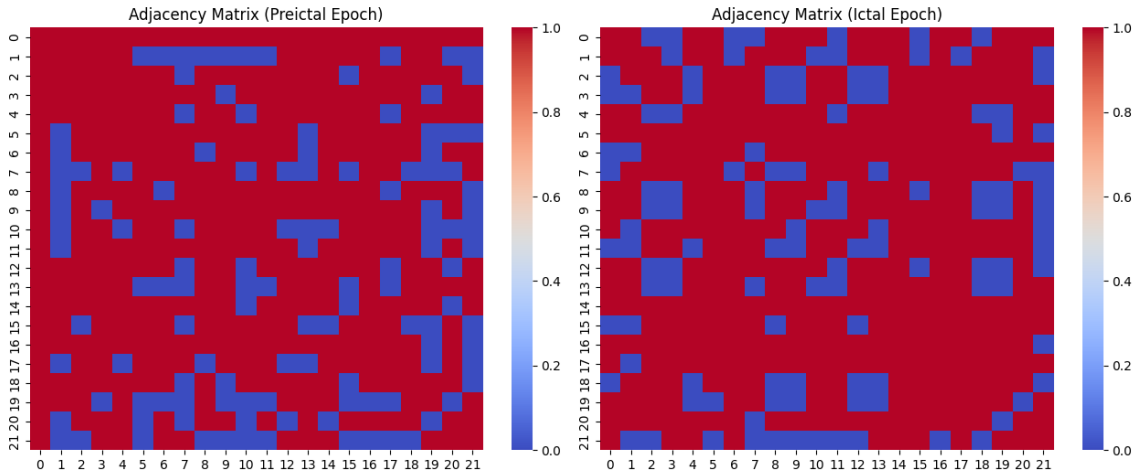


Figure 4.3: CHB data Gram Matrices

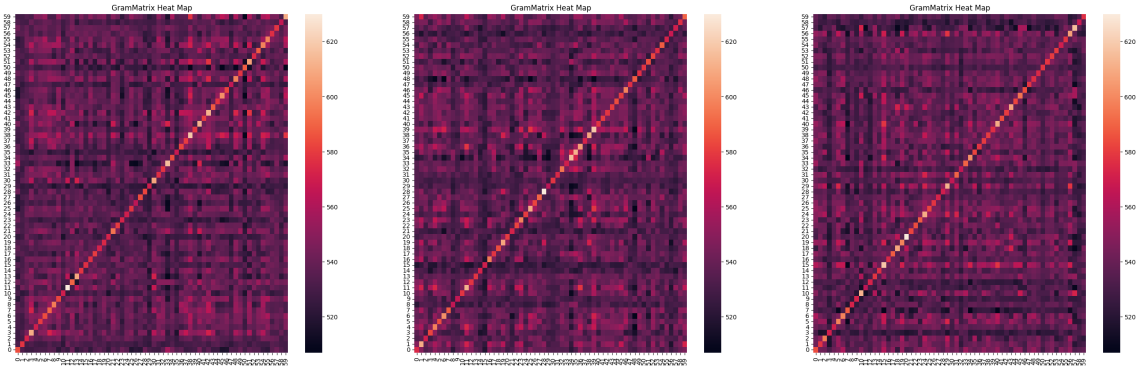


Figure 4.4: CHB data Gram Matrices

Correlation Heatmap was plotted to compare channel-wise correlations for ictal and preictal stages and analyze connectivity differences as in figure 4.4

### 4.1.2 Understanding Classification Accuracy and Feature Importance

The data set was trained with the kernel-classifier for the ictal and preictal data obtained from raw EEG. The Raw EEG data was extracted, filtered, and normalized after determining the Ictal and preictal segments, further splitting them into 2-second

epochs, and preparing them for training a model to predict seizures or classify seizure states. Time parameters like SPH (seizure prediction horizon) was utilized to define the window before a seizure to classify as preictal.

The average accuracy of around 83% was obtained for Weisfeiler-Lehman(WL) Kernel across all classifiers - Decision Tree, Random Forest, SVC with mPLV as connectivity measure . It outperformed the other kernels namely Spectral Decomposition Kernel and Random Walk Kernels. The values of Accuracy and AUC of the WL kernel across different combinations of connectivty measure which are mPLV and corelation-coefficient and kernels and classifiers with and without Cost-sensitive learning is displayed in Figure 4.5. Also the results obtained without Cost-sensitive Learning is available in A.2

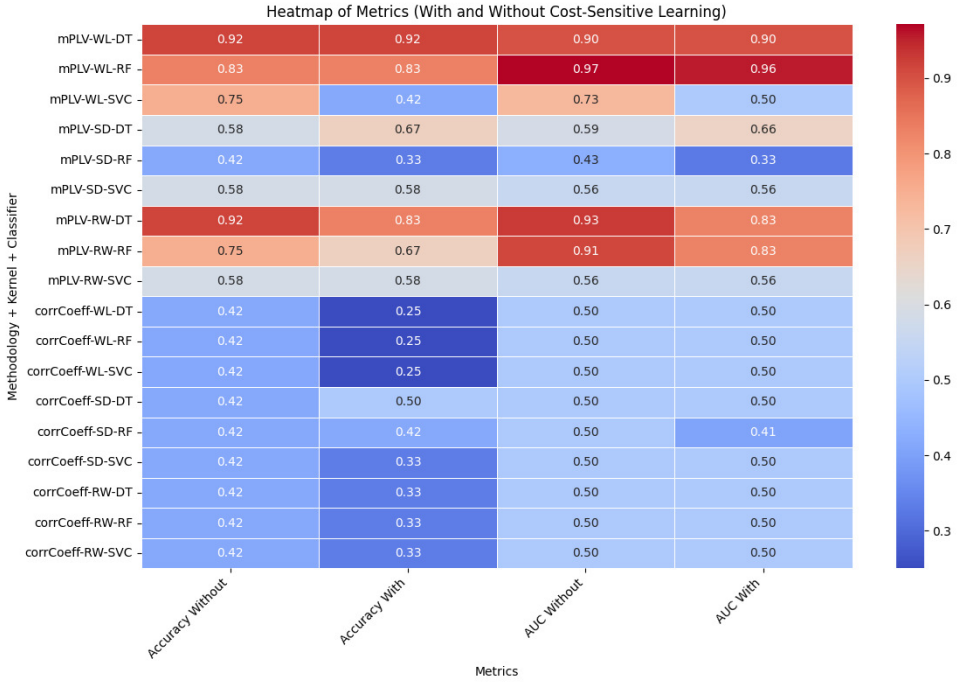


Figure 4.5: CHB Data - Performance Results

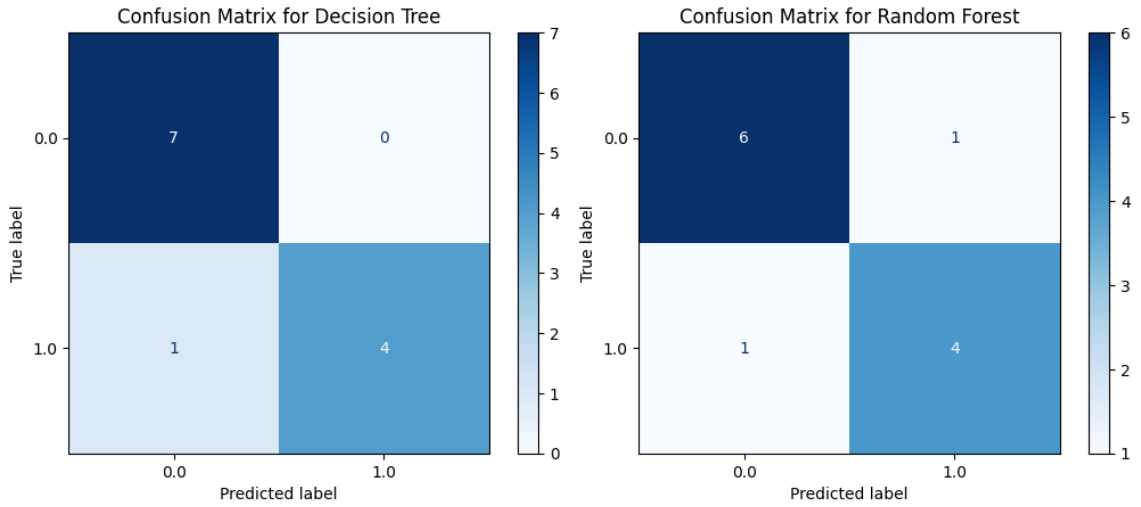


Figure 4.6: CHB data WL Kernel - Confusion Matrix

Table A.1 on Page 55 displays the complete performance metrics for the various combinations of Connectivity measures, kernels included Weisfeiler-Lehman Kernel (K1), Spectral Decomposition Kernel(K2) and Random Walk Kernel(K3) along with classifiers Decision Tree(C1), Random Forest(C2) and SVC(C3).

To assess the performance of the classifiers, confusion matrices are analyzed for three models: Decision Tree, Random Forest, and SVC. The confusion matrices provide a detailed breakdown of predictions into four categories: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The Decision Tree classifier emerged as the most reliable model for this dataset, offering robust predictions with minimal errors. The Confusion Matrix of the best performed kernel - WL Kernel for classifier combination is shown in the Figure 4.6

The ROC Curve was plotted to evaluate the kernel in evaluating a classifier's performance by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). The plot in figure 4.7 shows the ROC (Receiver Operating Characteristic) curves for three different models: Decision Tree, Random Forest, and SVC. The

AUC (Area Under the Curve) is of 0.90 indicating a high-performance with a good balance between sensitivity and specificity. The ROC curve for the Random Forest model achieves an AUC of 0.96. This is the best among the three models, reflecting a very high-performance level and excellent classification capabilities. For SVC model is depicted in green, with an AUC of 0.50 indicating poor classification performance for this dataset.

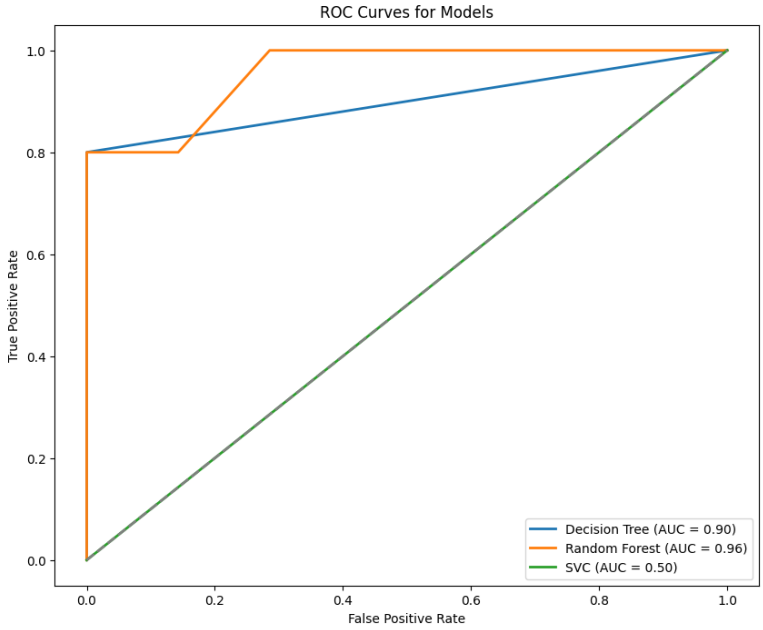


Figure 4.7: CHB Data - ROC Curves for WL Kernel

### 4.1.3 Clustering

The optimal number of clusters for K-means clustering was determined using the Elbow Method [43]. The plot generated as in figure 4.8 visualizes this. The plot helps identify the "elbow" point, where the graph starts to decrease more slowly. This point suggests the optimal number of clusters, balancing and avoiding overfitting. Identifying the elbow and the appropriate number of clusters for our K-means clustering model. Assuring a balance between model complexity and accuracy as in

figure 4.8.

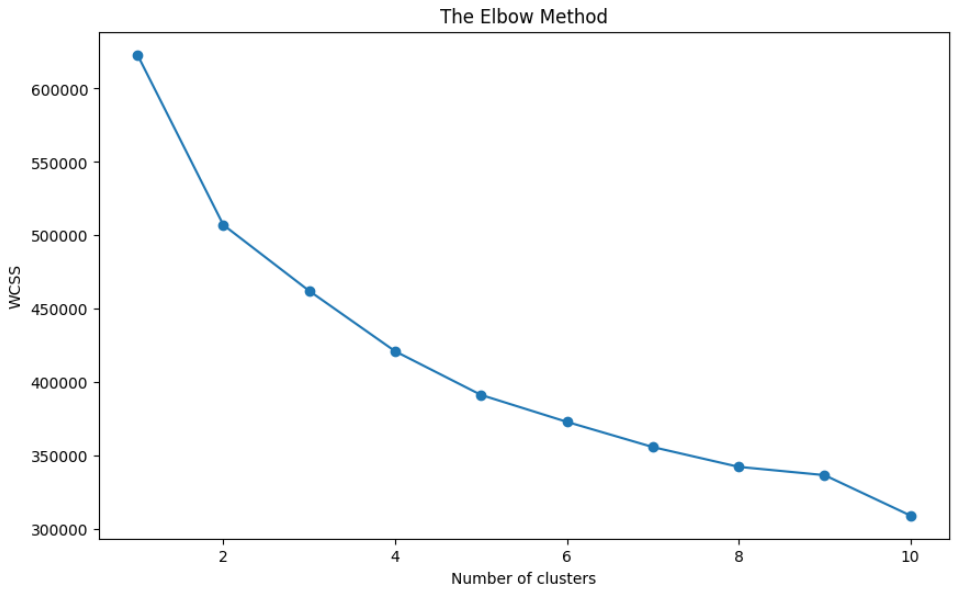


Figure 4.8: CHB Data Channel Information

The connectivity matrices corresponding to the segments are processed using graph kernels, resulting in their Gram matrices. Eigenanalysis was then performed on the Gram matrices, and the optimal number of clusters for the eigenvectors was determined based on the sum of squared error (SSE) and silhouette scores.

The scatter plot in figure 4.9 displays data points in a two-dimensional space defined by the first two principal components (Principal Component 1 and Principal Component 2) obtained from Kernel PCA. Each data point is according to the clusters identified by the KMeans algorithm, it helps in visualizing how the KMeans algorithm has grouped the data into distinct clusters within the reduced feature space. We see separability in the PCA-reduced feature space which shows distinct brain connectivity patterns.

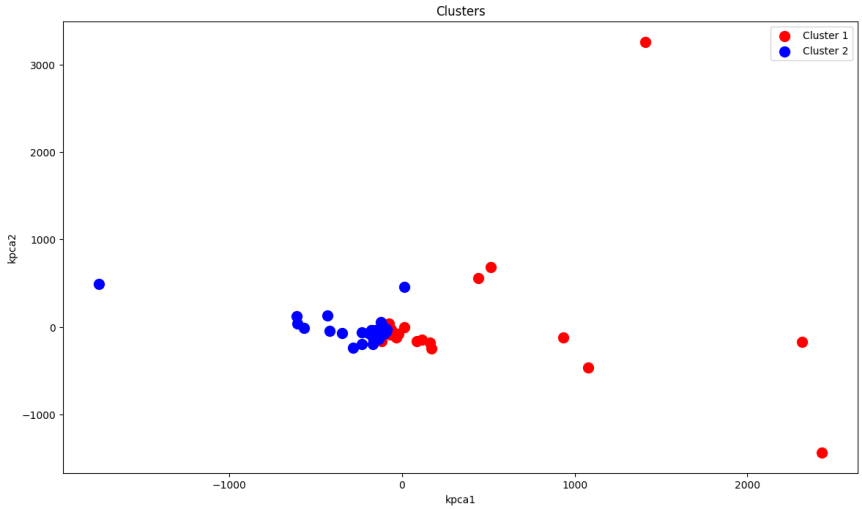


Figure 4.9: CHB Data - Clustering in Kernel PCA Feature Space

## 4.2 Dataset 2: Cognitive Load Dataset

### 4.2.1 Understanding the Dataset

The EEG data processing pipeline involved extracting graph representations for different cognitive states and calculating graph kernels for classification tasks. Only the cognitive states "Idle" and "Dual-1-Back," was considered which are already labeled in the datasets. The dataset contained 5 subjects for analysis. Using the MNE library, EEG data was preprocessed by loading the files, selecting relevant channels, and removing channels with all-zero data.

The continuous EEG signals are segmented into epochs of 3-second intervals. For each epoch, a correlation matrix was computed to capture the connectivity between EEG channels. This matrix was thresholded and binarized to form adjacency matrices, which are subsequently converted into graph representations. These graphs, along with their associated labels, are used to compute the Weisfeiler-Lehman kernel matrix. This kernel captures structural similarities between graphs and serves as a feature set for downstream machine learning tasks.

The EEG data is plotted using the MNE library, as in the Figure 4.10 which shows the EEG signals are measured as potential differences relative to a reference electrode also referred to as channels. This step ensures that the data is analyzed relative to a consistent and meaningful baseline. The 14 channels(only CHANNEL#3 - CHANNEL#16) being considered is plotted on the Y-axis and the time is plotted over X-axis.

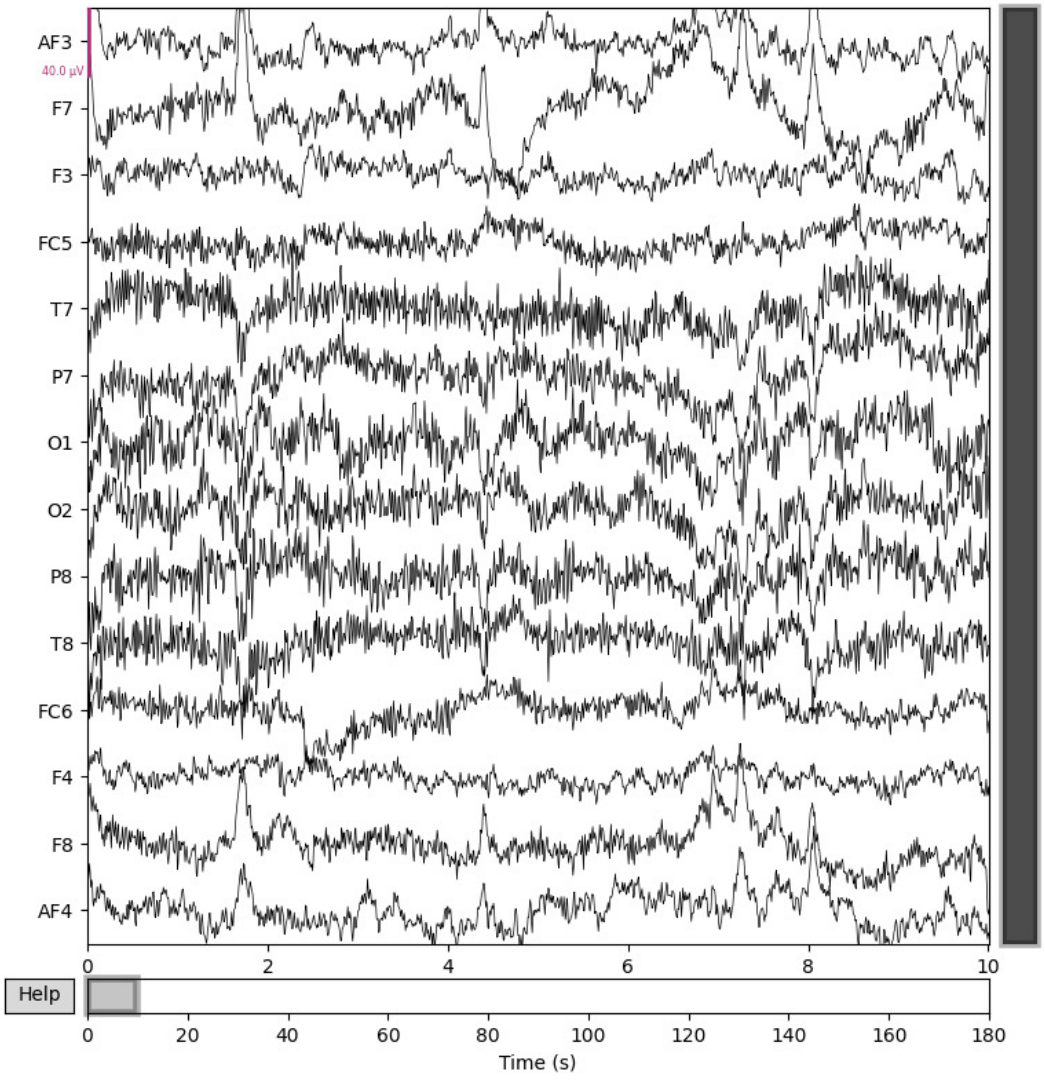


Figure 4.10: Cognitive Load Data - Channel Information

### 4.2.2 Understanding Classification Accuracy and Feature Importance

An average accuracy of around 85% was obtained for Weisfeiler-Lehman Kernel across different classifiers - Decision Tree, Random Forest, SVC and considered only the Idle and Dual-1-Back states for our analysis. Random Walk Kernel also performed better for some subjects only. The Accuracy Score is illustrated in Figure 4.11 shows across comparison of classifier performance metrics (accuracy) with and without cost-sensitive learning across multiple subjects, kernels, and classifiers(only Decision Tree and Random Forest shown). The results of evaluation before Cost-sensitive learning is documented in A.4.

It is seen by performing Cost-Sensitve Learning in most cases, accuracy improved significantly. It is noticed that some subject-kernel combinations (e.g., S01-RW-RF and S02-WL-RF) show consistently high accuracy, while others (e.g., S03-SD-DT and S04-SD-RF) exhibit lower accuracy, even with cost-sensitive learning. Random Forest(RF) performed better than Decision Tree(DT) also SVC as a classifier has consistent poor scores. The entire performance score across different combinations is also available in page 57.

The models ability to distinguish between positive and negative classes is analyzed using the ROC(Receiver Operating Characteristic) curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) [30] as in figure 4.12 The curves are for three different models: Decision Tree, Random Forest, and SVC. The Decision Tree model shows a good performance with an AUC of 0.84. Random Forest model performs with an AUC of 1.00, suggesting excellent classification capabilities. SVC model performs no better than random guessing with an AUC of 0.50. SVC had a consistent poor performance across various subjects and kernel combination.

To further evaluate the performance the confusion matrices is plotted of the three different classification models: SVC (Support Vector Classifier), Decision Tree, and Random Forest as shown in figure 4.13 SVC classified all true negatives (12) but fails to classify any true positives. All 13 positive cases are misclassified as negatives, indicating a severe bias towards the negative class. Decision Tree model shows better performance than SVC, correctly identifying majority of true negatives and true positives. However, it misclassified few negative cases as positives and positive case as negative, reflecting a moderate level of error. But Random Forest demonstrates the best performance among the three. It had only 1 misclassification in each class. This indicates a balanced and effective classification capability.

Overall, the Random Forest model is the most accurate and balanced classifier in this comparison, while SVC struggles significantly with positive case classification.

### 4.2.3 Clustering

The EEG signals are preprocessed and computed into connectivity matrices using corelation coefficients corresponding to the segments and are processed by graph kernels which returned their gram matrices. Subsequent step performed eigenanalysis of the gram matrices and optimal number of clusters of the eigenvectors was determined. Scatter plot was generated as in figure 4.14 to visualize data points using principal components, which was obtained from PCA (Principal Component Analysis) which is a dimensionality reduction technique.

We observe the clusters identifying dynamic states in brain activity. Help identify transitions (preictal $\rightarrow$ ictal). Though some overlap between the two classes is noticed, there are distinct clusters, which showcases that the principal components are effective in differentiating the classes.

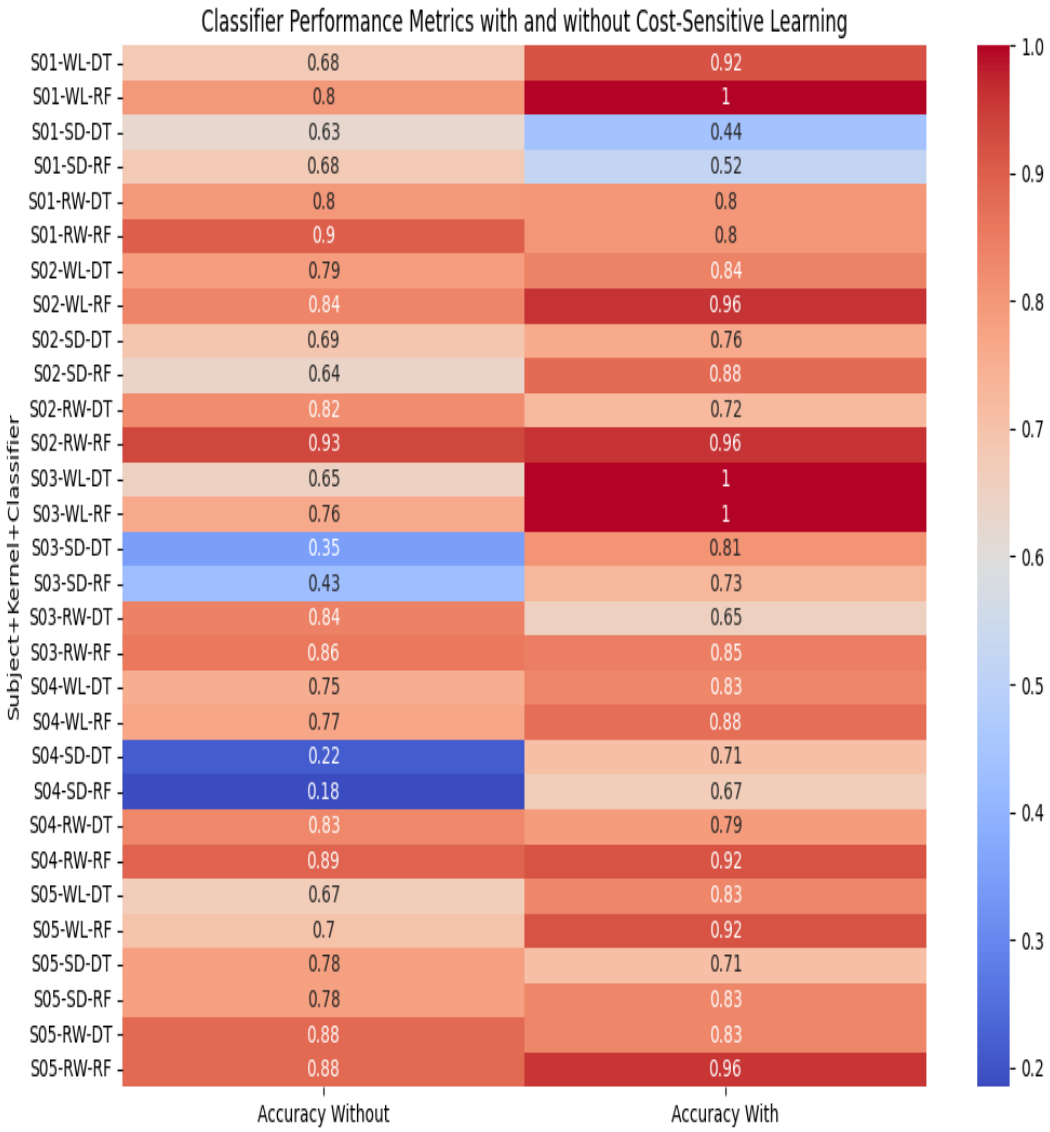


Figure 4.11: Cognitive Load Data - Performance Evaluation

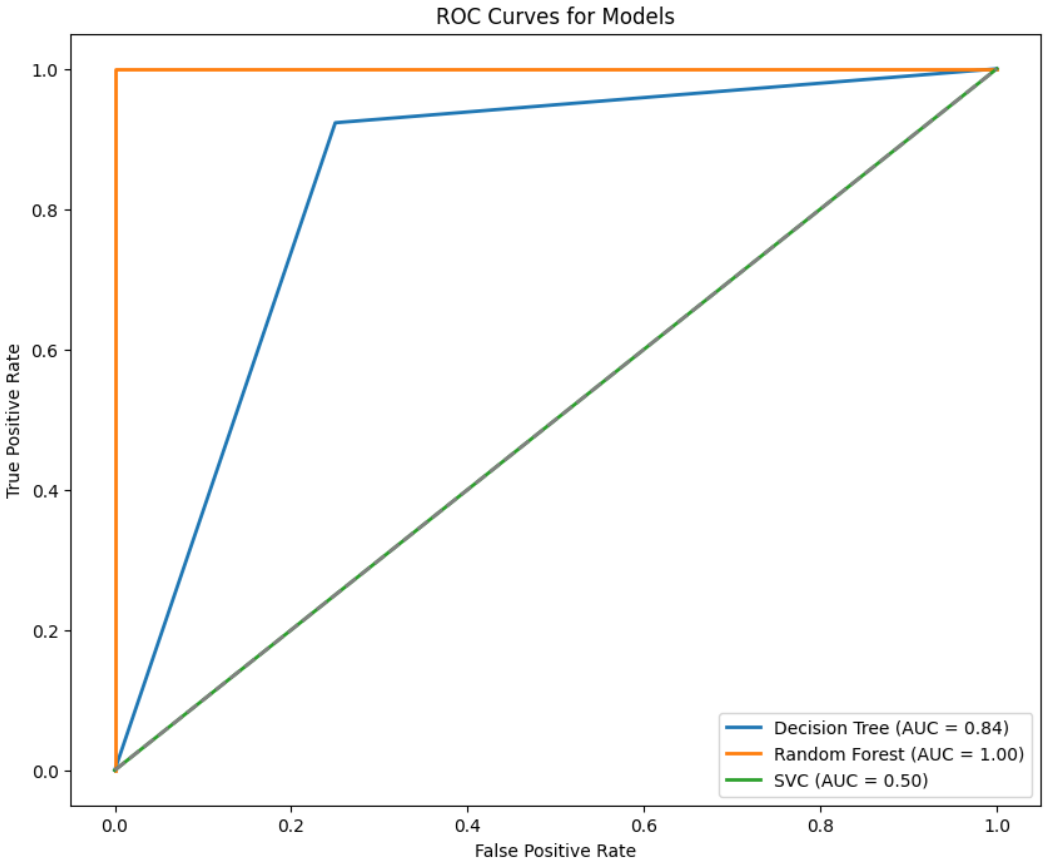


Figure 4.12: Cognitive Load Data - ROC Curve

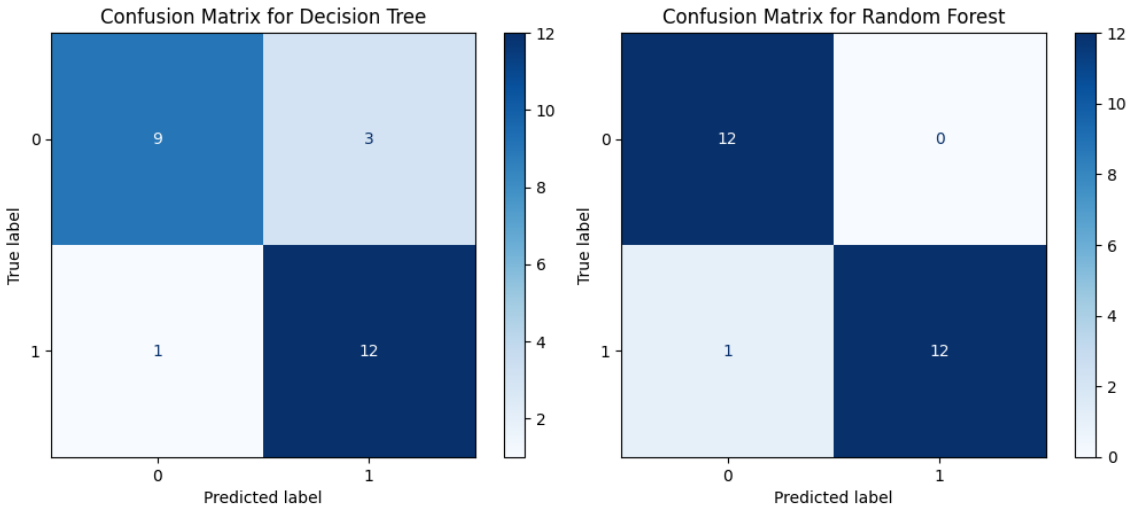


Figure 4.13: Cognitive Load Data - WL Kernel - Confusion Matrix

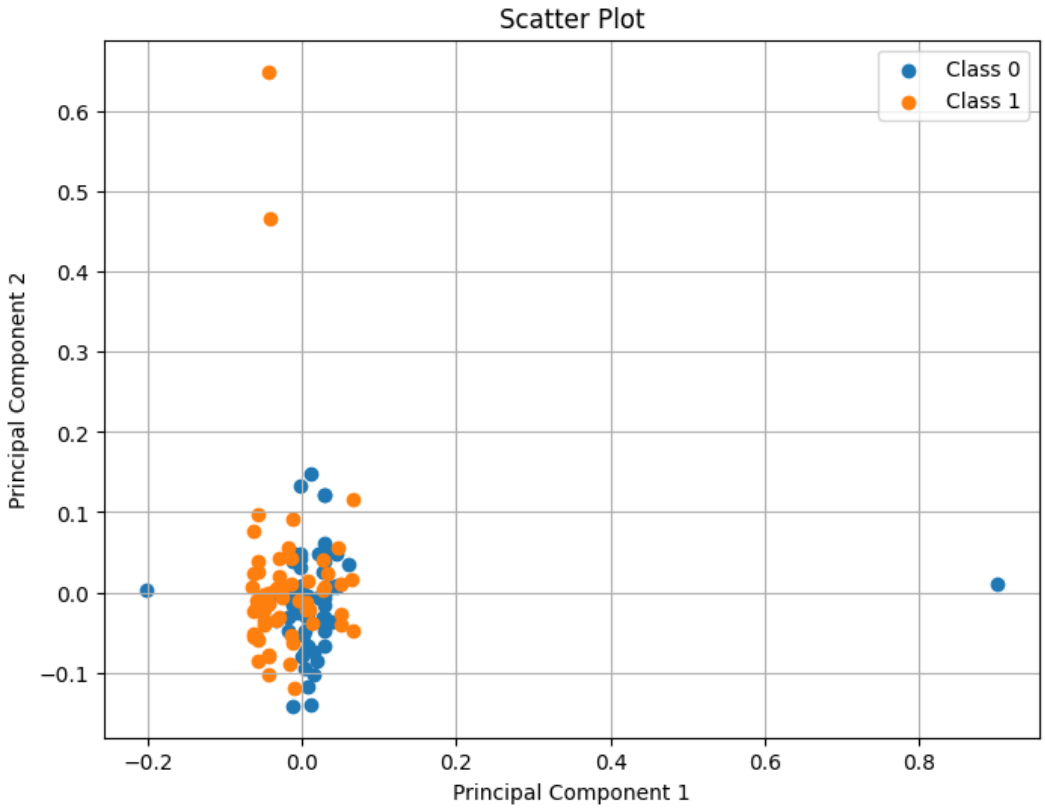


Figure 4.14: Cognitive Load Data - PCA Scatter Plot - Clustering

# Chapter 5

# Conclusion

This thesis presents a novel method for classifying cognitive states by leveraging multiple biophysiological signals, with a primary focus on EEG data. The core innovation lies in the use of Koopman transfer operators, which enable the extraction of spectral connectivity features in a simplified yet effective manner. This approach enhances both the accuracy and reliability of cognitive state classification.

The proposed model demonstrates strong performance, achieving an average classification accuracy exceeding 85%, showcasing its potential for practical applications in neuroscience and related fields.

## 5.1 Future Work

- **Dataset Expansion**: While the current results are good in the two datasets, one with Seizure data and one with workload data, testing the model on a more extensive and diverse datasets is essential to improve generalization. This would make it easier to apply the findings to different groups of people and tasks, ensuring the model works well and is reliable in various situations.

- **Classifiers and Kernel Tuning**:

  More research could focus on improving how kernels and classifiers work together. Exploring other kernels and kernel variations within random walk kernels, Weisfeiler-Lehman kernels, or graph-based deep learning techniques might reveal better configurations for specific datasets.

- **Deep Learning Integration**: Incorporating deep learning-based classifiers presents a significant opportunity. Techniques such as Graph Convolutional Networks (GCNs) or Recurrent Neural Networks (RNNs) could enhance the model's ability to capture complex, non-linear relationships in the EEG data. This could lead to more accurate and scalable solutions for cognitive state detection.

- **Hybrid Architectures**: Developing hybrid models that combine graph-based methods with traditional machine learning or deep learning architectures (like LSTM networks for sequence data) could offer more flexibility.

## 5.2 Limitations

- **Generalization Issues**: The current model has been tested on a limited number of subjects and cognitive tasks. The model's performance can differ depending on the EEG datasets or experimental conditions used. Thorough testing on larger and more diverse datasets is essential to ensure its reliability.

- **Kernel-Classifier Dependency**: The effectiveness of the model heavily relies on the selected kernels and classifiers. Poor parameter tuning or an unsuitable kernel choice could lead to less effective results.

- **Computational Complexity**: Using spectral embedding and graph-based calculations requires a lot of computing power, especially with high-dimensional EEG data. This might make it harder to use the model for real-time analysis or very large datasets.

- **Interpretability Challenges**: The Koopman operator method makes things easier to explain, but understanding how spectral embeddings work still needs expert knowledge. Connecting the model's results to useful clinical insights is still a challenge.

- **Ethical Considerations**: Real-time seizure prediction systems raise concerns regarding data privacy, fairness, and societal impact. Ensuring anonymity, addressing biases in dataset representation, and evaluating the societal implications are critical steps for ethical implementation.

# Bibliography


[1] Frank P. Elsen. "Chatting with a neuron: patch-clamp technology allows precise investigation of neural mechanisms". en. In: *Genetic Engineering & Biotechnology News* 35.21 (Dec. 2015), pp. 16–17. ISSN: 1935-472X, 1937-8661. DOI: 10.1089/gen.35.21.09.

[2] Congzhong Sun and Chaozhou Mou. "Survey on the research direction of EEG-based signal processing". English. In: *Frontiers in Neuroscience* 17 (July 2023), p. 8. ISSN: 1662-453X. DOI: 10.3389/fnins.2023.1203059.

[3] F. Mormann et al. "Seizure prediction: the long and winding road". en. In: *Brain* 130.2 (Feb. 2007), pp. 314–333. ISSN: 0006-8950, 1460-2156. DOI: 10.1093/brain/awl241.

[4] Zhichao Liang et al. "Online Learning Koopman Operator for Closed-Loop Electrical Neurostimulation in Epilepsy". In: *IEEE Journal of Biomedical and Health Informatics* 27.1 (Jan. 2023), pp. 492–503. ISSN: 2168-2194, 2168-2208. DOI: 10.1109/JBHI.2022.3210303.

[5] Shaodi Qian and Chun-An Chou. "A Koopman-operator-theoretical approach for anomaly recognition and detection of multi-variate EEG system". In: *Biomedical Signal Processing and Control* 69 (Aug. 2021), p. 102911. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2021.102911.

[6] Shoeb Ali. *CHB-MIT Scalp EEG Database*. https://physionet.org/content/chbmit. 2010. DOI: 10.13026/C2K01R.

[7] IIT KHARAGPUR. *EEG DATA, CAPTURED @BCI-HCI LAB.* https://github.com/BCI-HCI-IITKGP/Cognitive-Mental-workload-EEG-Data.

[8] Mohammad Khubeb Siddiqui et al. "A review of epileptic seizure detection using machine learning classifiers". en. In: *Brain Informatics* 7.1 (Dec. 2020), p. 5. ISSN: 2198-4018, 2198-4026. DOI: 10.1186/s40708-020-00105-1.

[9] World Health Organization. *Epilepsy: a public health imperative*. en. Geneva: World Health Organization, 2019. ISBN: 9789241515931.

[10] B. Jedrzejczyk-Goral, H. Flisiak-Antonijczuk, and R. Kalinowski. "1925 – The frequency of different types of seizures in children suffered from autism and epilepsy". en. In: *European Psychiatry* 28 (Jan. 2013), p. 1. ISSN: 09249338. DOI: 10.1016/S0924-9338(13)76871-8.

[11] Arnaud Gaudinat et al. "Health search engine with e-document analysis for reliable search results". en. In: *International Journal of Medical Informatics* 75.1 (Jan. 2006), pp. 73–85. ISSN: 13865056. DOI: 10.1016/j.ijmedinf.2005.11.002.

[12] Swati Banerjee and Viktor Jirsa. "A review of epileptic markers: from ion channels, astrocytes, synaptic imbalance to whole brain network dynamics". en. In: *Exploration of Neuroscience* 3.5 (Sept. 2024), pp. 478–492. ISSN: 2834-5347. DOI: 10.37349/en.2024.00060.

[13] Apu Nandy et al. "Feature Extraction and Classification of EEG Signals for Seizure Detection". In: *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*. Dhaka, Bangladesh: IEEE, Jan. 2019, pp. 480–485. DOI: 10.1109/ICREST.2019.8644337.

[14] Mengni Zhou et al. "Epileptic Seizure Detection Based on EEG Signals and CNN". In: *Frontiers in Neuroinformatics* 12 (Dec. 2018), p. 95. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00095.

[15] Marco Zurdo-Tabernero et al. "An overview of machine learning and deep learning techniques for predicting epileptic seizures". en. In: *Journal of Integrative Bioinformatics* 20.4 (Jan. 2024), p. 20230002. ISSN: 1613-4516. DOI: 10.1515/jib-2023-0002.

[16] Vishwambhar Pathak et al. "Exploring the Efficacy of Explainable Deep Learning in Identifying Neuromarkers for Precise Prediction of Epilepsy and Causal Connectivity Analysis". en. In: *2023 The 15th International Conference on Computer Modeling and Simulation*. Dalian China: ACM, June 2023, pp. 126–132. ISBN: 9798400707919. DOI: 10.1145/3608251.3608292.

[17] Chiyuan Zhang et al. "Understanding deep learning (still) requires rethinking generalization". en. In: *Communications of the ACM* 64.3 (Mar. 2021), pp. 107–115. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3446776.

[18] Leslie Hannah. "WilliamLazonick and David J.Teece, eds., Management innovation: essays in the spirit of Alfred D. Chandler, Jr. (New York: Oxford University Press, 2012. Pp. xii + 378. 15 figs. 9 tabs. ISBN 9780199695683 Hbk. £55)". en. In: *The Economic History Review* 66.2 (May 2013), pp. 684–685. ISSN: 0013-0117, 1468-0289. DOI: 10.1111/1468-0289.12015_30.

[19] Ioannis K. Gallos et al. "Data-driven modelling of brain activity using neural networks, diffusion maps, and the Koopman operator". en. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 34.1 (Jan. 2024), p. 013151. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/5.0157881.

[20] N. Venkata Saichand and Gopiya Naik. S. "Epileptic seizure detection using novel Multilayer LSTM Discriminant Network and dynamic mode Koopman

decomposition". en. In: *Biomedical Signal Processing and Control* 68 (July 2021), p. 102723. ISSN: 17468094. DOI: 10.1016/j.bspc.2021.102723.

[21] Natasza Marrouch et al. "Data-driven Koopman operator approach for computational neuroscience". en. In: *Annals of Mathematics and Artificial Intelligence* 88.11-12 (Dec. 2020), pp. 1155–1173. ISSN: 1012-2443, 1573-7470. DOI: 10.1007/s10472-019-09666-2.

[22] Kuldeep Singh and Jyoteesh Malhotra. *Smart neurocare approach for detection of epileptic seizures using deep learning based temporal analysis of EEG patterns - Multimedia Tools and Applications — link.springer.com*. https://link.springer.com/article/10.1007/s11042-022-12512-z. [Accessed 03-01-2025].

[23] Shanen Chen et al. "Automatic Diagnosis of Epileptic Seizure in Electroencephalography Signals Using Nonlinear Dynamics Features". In: *IEEE Access* (2019), pp. 61046–61056. DOI: 10.1109/ACCESS.2019.2915610.

[24] Mahshid Dastgoshadeh and Zahra Rabiei. "Detection of epileptic seizures through EEG signals using entropy features and ensemble learning". In: *Frontiers in Human Neuroscience* 16 (Feb. 2023), p. 1084061. ISSN: 1662-5161. DOI: 10.3389/fnhum.2022.1084061.

[25] Stefan Klus and Nataša Djurdjevac Conrad. "Koopman-Based Spectral Clustering of Directed and Time-Evolving Graphs". en. In: *Journal of Nonlinear Science* 33.1 (Feb. 2023), p. 8. ISSN: 0938-8974, 1432-1467. DOI: 10.1007/s00332-022-09863-0.

[26] Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. "Koopman operator dynamical models: Learning, analysis and control". en. In: *Annual Reviews in Control* 52 (2021), pp. 197–212. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2021.09.002.

[27] Francesco Zanini and Alessandro Chiuso. "Estimating Koopman operators for nonlinear dynamical systems: a nonparametric approach". en. In: *IFAC-PapersOnLine* 54.7 (2021), pp. 691–696. ISSN: 24058963. DOI: 10.1016/j.ifacol.2021.08.441.

[28] Nino Shervashidze et al. "Weisfeiler-Lehman Graph Kernels". In: *Journal of Machine Learning Research* 12.77 (2011), pp. 2539–2561. ISSN: 1533-7928.

[29] Kateryna Melnyk et al. "GraphKKE: graph Kernel Koopman embedding for human microbiome analysis". en. In: *Applied Network Science* 5.1 (Dec. 2020), p. 96. ISSN: 2364-8228. DOI: 10.1007/s41109-020-00339-2.

[30] Intisar Shadeed Al-Mejibli et al. "Performance Evaluation of Kernels in Support Vector Machine". In: *2018 1st Annual International Conference on Information and Sciences (AiCIS)*. Nov. 2018, pp. 96–101. DOI: 10.1109/AiCIS.2018.00029.

[31] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. "A survey on graph kernels". en. In: *Applied Network Science* 5.1 (Jan. 2020), p. 6. ISSN: 2364-8228. DOI: 10.1007/s41109-019-0195-3.

[32] Linlin Jia, Benoit Gaüzère, and Paul Honeine. "Graph kernels based on linear patterns: Theoretical and experimental comparisons". en. In: *Expert Systems with Applications* 189 (Mar. 2022), p. 116095. ISSN: 09574174. DOI: 10.1016/j.eswa.2021.116095.

[33] S.V.N. Vishwanathan et al. "Graph Kernels". In: *Journal of Machine Learning Research* 11.40 (2010), pp. 1201–1242.

[34] Nino Shervashidze et al. "Efficient graphlet kernels for large graph comparison". en. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2009, pp. 488–495.

[35] Daniel Lehmberg, Felix Dietrich, and Gerta Köster. "Modeling Melburnians—Using the Koopman operator to gain insight into crowd dynamics". en. In: *Transportation Research Part C: Emerging Technologies* 133 (Dec. 2021). ISSN: 0968090X. DOI: 10.1016/j.trc.2021.103437.

[36] Tak-chung Fu. "A review on time series data mining". en. In: *Engineering Applications of Artificial Intelligence* 24.1 (Feb. 2011), pp. 164–181. ISSN: 09521976. DOI: 10.1016/j.engappai.2010.09.007.

[37] Yakir Yehuda, Daniel Freedman, and Kira Radinsky. "Self-supervised Classification of Clinical Multivariate Time Series using Time Series Dynamics". en. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Long Beach CA USA: ACM, Aug. 2023, pp. 5416–5427. ISBN: 9798400701030. DOI: 10.1145/3580305.3599954.

[38] William W. S. Wei. *Multivariate Time Series Analysis and Applications — Wiley*. [Online; accessed 2025-01-03].

[39] Kasun Mendis, Manjusri Wickramasinghe, and Pasindu Marasinghe. "Multivariate Time Series Forecasting: A Review". In: *Proceedings of the 2024 2nd Asia Conference on Computer Vision, Image Processing and Pattern Recognition*. CVIPPR '24. Xiamen, China: Association for Computing Machinery, 2024. ISBN: 9798400716607. DOI: 10.1145/3663976.3664241.

[40] Wei Zhang, Yao-Chi Yu, and Jr-Shin Li. "Dynamics reconstruction and classification via Koopman features". en. In: *Data Mining and Knowledge Discovery* 33.6 (Nov. 2019), pp. 1710–1735. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00639-x.

[41] Abdulkadir Celikkanat, Yanning Shen, and Fragkiskos Malliaros. "Multiple Kernel Representation Learning on Networks". In: *IEEE Transactions on*

*Knowledge and Data Engineering* (2022), pp. 1–1. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2022.3172048.

[42] Moshe Salhov et al. "Learning from patches by efficient spectral decomposition of a structured kernel". en. In: *Machine Learning* 103.1 (Apr. 2016), pp. 81–102. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-015-5538-4.

[43] Dhendra Marutho et al. "The Determination of Cluster Number at k-Mean Using Elbow Method and Purity Evaluation on Headline News". In: *2018 International Seminar on Application for Technology of Information and Communication*. Sept. 2018, pp. 533–538. DOI: 10.1109/ISEMANTIC.2018.8549751.

# Appendix A

# Performance Metrics of Graph Kernels

This Appendix presents tables comparing various Subjects, Kernels, and Classifiers, as discussed in Section 3.1. The comparison encompasses six different Subjects: three kernels, which comprise Weisfeiler-Lehman, Spectral Decomposition, and Random Walk Kernel, and three classifiers, which comprise Decision Tree, SVM, and Random Forest classifiers.

Table A.1 shows the comparison for the CHB dataset, with cost-sensitive learning applied, A.2 is without cost-sensitive learning implemented. Similarly, the evaluation scores with and without cost-sensitive for Cognitive Load Dataset are shown in A.3 and A.4, respectively.

## A.1 CHB dataset performance scores

### A.1.1 With Cost-Sensitive Learning

Table A.1: Performance metrics of Graph Kernels - Classifiers for CHB Dataset with Cost-Sensitive Learning

| **Subject** | **Kernel, Classifier** | **Accuracy** | **AUC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|---|---|
| mPLV | WL, DT | 0.9167 | 0.9 | 0.9271 | 0.9167 | 0.9148 |
| | WL, RF | 0.8333 | 0.9517 | 0.8333 | 0.8333 | 0.8333 |
| | WL, SVC | 0.4167 | 0.5 | 0.1736 | 0.4167 | 0.2451 |
| | SD, DT | 0.6667 | 0.6571 | 0.6667 | 0.6667 | 0.6667 |
| | SD, RF | 0.3333 | 0.3286 | 0.3333 | 0.3333 | 0.3333 |
| | SD, SVC | 0.5833 | 0.5571 | 0.5729 | 0.5833 | 0.5741 |
| | RW, DT | 0.8333 | 0.8286 | 0.8333 | 0.8333 | 0.8333 |
| | RW, RF | 0.6667 | 0.8286 | 0.6667 | 0.6667 | 0.6667 |
| | RW, SVC | 0.5833 | 0.5571 | 0.5729 | 0.5833 | 0.5833 |
| Corelation Coefficient | WL, DT | 0.25 | 0.5 | 0.0625 | 0.25 | 0.1 |
| | WL, RF | 0.25 | 0.5 | 0.0625 | 0.25 | 0.1 |
| | WL, SVC | 0.25 | 0.5 | 0.0625 | 0.25 | 0.1 |
| | SD, DT | 0.5 | 0.5 | 0.5556 | 0.5 | 0.5143 |
| | SD, RF | 0.4167 | 0.4062 | 0.5556 | 0.4167 | 0.3963 |
| | SD, SVC | 0.3333 | 0.5 | 0.1111 | 0.3333 | 0.1667 |
| | RW, DT | 0.3333 | 0.5 | 0.1111 | 0.3333 | 0.1667 |
| | RW, RF | 0.3333 | 0.5 | 0.1111 | 0.3333 | 0.1667 |
| | RW, SVC | 0.3333 | 0.5 | 0.1111 | 0.3333 | 0.1667 |

### A.1.2 Without Cost-Sensitive Learning

Table A.2: Performance metrics of Graph Kernels - Classifiers for CHB Dataset without Cost-Sensitive Learning

| **Subject** | **Kernel, Classifier** | **Accuracy** | **AUC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|---|---|
| mPLV | WL, DT | 0.9167 | 0.90 | 1 | 0.80 | 0.8889 |
| | WL, RF | 0.8333 | 0.9714 | 0.80 | 0.80 | 0.80 |
| | WL, SVC | 0.75 | 0.7286 | 0.75 | 0.60 | 0.6667 |
| | SD, DT | 0.5833 | 0.5857 | 0.50 | 0.60 | 0.5455 |
| | SD, RF | 0.4167 | 0.4286 | 0.3333 | 0.40 | 0.3636 |
| | SD, SVC | 0.5833 | 0.5571 | 0.50 | 0.40 | 0.4444 |
| | RW, DT | 0.9167 | 0.9286 | 0.8333 | 1 | 0.9091 |
| | RW, RF | 0.75 | 0.9143 | 0.75 | 0.60 | 0.6667 |
| | RW, SVC | 0.5833 | 0.5571 | 0.50 | 0.40 | 0.4444 |
| Corelation Coefficient | WL, DT | 0.4167 | 0.50 | 0.4167 | 1 | 0.5882 |
| | WL, RF | 0.4167 | 0.50 | 0.4167 | 1 | 0.5882 |
| | WL, SVC | 0.4167 | 0.50 | 0.4167 | 1 | 0.5882 |
| | SD, DT | 0.50 | 0.50 | 0.50 | 1 | 0.6667 |
| | SD, RF | 0.50 | 0.50 | 0.50 | 1 | 0.6667 |
| | SD, SVC | 0.50 | 0.50 | 0.50 | 1 | 0.6667 |
| | RW, DT | 0.3333 | 0.50 | 0 | 0 | 0 |
| | RW, RF | 0.3333 | 0.50 | 0 | 0 | 0 |
| | RW, SVC | 0.3333 | 0.50 | 0 | 0 | 0 |

## A.2 Cognitive Load Dataset with 5 different subjects

### A.2.1 With Cost-Sensitive Learning

Table A.3: Performance metrics of Graph Kernels - Classifiers for Cognitive Load Dataset with Cost-Sensitive Learning

| **Subject** | **Kernel, Classifier** | **Accuracy** | **AUC** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|---|---|
| S01 | WL, DT | 0.92 | 0.9286 | 0.9323 | 0.92 | 0.9203 |
| | WL, RF | 1 | 1 | 1 | 1 | 1 |
| | WL, SVC | 0.56 | 0.5 | 0.3136 | 0.56 | 0.4021 |
| | SD, DT | 0.44 | 0.4351 | 0.4492 | 0.44 | 0.4418 |
| | SD, RF | 0.52 | 0.6169 | 0.5297 | 0.52 | 0.5215 |
| | SD, SVC | 0.48 | 0.4675 | 0.4747 | 0.48 | 0.4766 |
| | RW, DT | 0.8 | 0.7922 | 0.8 | 0.8 | 0.7987 |
| | RW, RF | 0.8 | 0.8831 | 0.8 | 0.8 | 0.7987 |
| | RW, SVC | 0.64 | 0.6104 | 0.6470 | 0.64 | 0.6143 |
| S02 | WL, DT | 0.84 | 0.8365 | 0.8480 | 0.84 | 0.8384 |
| | WL, RF | 0.96 | 0.1 | 0.9631 | 0.96 | 0.96 |
| | WL, SVC | 0.4 | 0.5 | 0.2304 | 0.48 | 0.3114 |
| | SD, DT | 0.76 | 0.7404 | 0.76 | 0.76 | 0.76 |
| | SD, RF | 0.88 | 0.9551 | 0.8821 | 0.88 | 0.8796 |
| | SD, SVC | 0.88 | 0.9551 | 0.8821 | 0.88 | 0.8796 |
| | RW, DT | 0.72 | 0.7212 | 0.7223 | 0.72 | 0.72 |
| | RW, RF | 0.96 | 1 | 0.9631 | 0.96 | 0.96 |
| | RW, SVC | 0.88 | 0.8814 | 0.8828 | 0.88 | 0.88 |

| Subject | Kernel, Classifier | Accuracy | AUC | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|
| S03 | WL, DT | 1 | 1 | 1 | 1 | 1 |
| | WL, RF | 1 | 1 | 1 | 1 | 1 |
| | WL, SVC | 0.5385 | 0.5 | 0.2899 | 0.5385 | 0.3769 |
| | SD, DT | 0.8077 | 0.8095 | 0.8107 | 0.8077 | 0.8080 |
| | SD, RF | 0.7308 | 0.8214 | 0.7483 | 0.7308 | 0.7296 |
| | SD, SVC | 0.5769 | 0.5774 | 0.5799 | 0.5769 | 0.5775 |
| | RW, DT | 0.6538 | 0.6488 | 0.6527 | 0.6538 | 0.6523 |
| | RW, RF | 0.8462 | 0.9286 | 0.8846 | 0.8462 | 0.8443 |
| | RW, SVC | 0.5385 | 0.5 | 0.2889 | 0.5385 | 0.3769 |
| S04 | WL, DT | 0.8333 | 0.8392 | 0.8450 | 0.8333 | 0.8333 |
| | WL, RF | 0.8750 | 0.9231 | 0.8785 | 0.875 | 0.8752 |
| | WL, SVC | 0.4583 | 0.5 | 0.211 | 0.4583 | 0.2881 |
| | SD, DT | 0.7083 | 0.7098 | 0.7118 | 0.7083 | 0.7088 |
| | SD, RF | 0.6667 | 0.014 | 0.670 | 0.6667 | 0.6667 |
| | SD, SVC | 0.5417 | 0.5699 | 0.6354 | 0.5417 | 0.4869 |
| | RW, DT | 0.7917 | 0.8007 | 0.8149 | 0.7917 | 0.7906 |
| | RW, RF | 0.9167 | 0.9371 | 0.9167 | 0.9167 | 0.9167 |
| | RW, SVC | 0.5833 | 0.5804 | 0.5833 | 0.5833 | 0.5833 |
| S05 | WL, DT | 0.8333 | 0.8429 | 0.8472 | 0.8333 | 0.8345 |
| | WL, RF | 0.9167 | 0.9714 | 0.9306 | 0.9167 | 0.9172 |
| | WL, SVC | 0.6667 | 0.6 | 0.7879 | 0.6667 | 0.5626 |
| | SD, DT | 0.7083 | 0.6929 | 0.7056 | 0.7083 | 0.7057 |
| | SD, RF | 0.8333 | 0.8857 | 0.8704 | 0.8333 | 0.8229 |
| | SD, SVC | 0.7917 | 0.7786 | 0.7907 | 0.7917 | 0.7898 |
| | RW, DT | 0.8333 | 0.8286 | 0.8333 | 0.8333 | 0.8333 |

| Subject | Kernel, Classifier | Accuracy | AUC | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|
| | RW, RF | 0.9583 | 0.9893 | 0.9611 | 0.9583 | 0.9580 |
| | RW, SVC | 0.7083 | 0.6786 | 0.7094 | 0.7083 | 0.6967 |

### A.2.2 Without Cost-Sensitive Learning

Table A.4: Performance metrics of Graph Kernels - Classifiers for Cognitive Load Dataset without Cost-Sensitive Learning

| Subject | Kernel, Classifier | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| S01 | WL, DT | 0.6780 | 0.6977 | 0.6780 | 0.6837 |
| | WL, RF | 0.7966 | 0.8070 | 0.7966 | 0.7981 |
| | WL, SVC | 0.6441 | 0.6700 | 0.6441 | 0.6487 |
| | SD, DT | 0.6271 | 0.6247 | 0.6271 | 0.6197 |
| | SD, RF | 0.6780 | 0.6733 | 0.6780 | 0.6697 |
| | SD, SVC | 0.5932 | 0.6334 | 0.5932 | 0.5766 |
| | RW, DT | 0.7966 | 0.8257 | 0.7966 | 0.8043 |
| | RW, RF | 0.8983 | 0.9072 | 0.8983 | 0.8969 |
| | RW, SVC | 0.6271 | 0.6642 | 0.6271 | 0.6170 |
| S02 | WL, DT | 0.7869 | 0.7970 | 0.7869 | 0.7902 |
| | WL, RF | 0.8361 | 0.8533 | 0.8361 | 0.8402 |
| | WL, SVC | 0.9016 | 0.9138 | 0.9016 | 0.9025 |
| | SD, DT | 0.6885 | 0.7027 | 0.6885 | 0.6710 |
| | SD, RF | 0.6393 | 0.6220 | 0.6393 | 0.6175 |
| | SD, SVC | 0.5246 | 0.5002 | 0.5246 | 0.4917 |
| | RW, DT | 0.8197 | 0.8323 | 0.8197 | 0.8185 |

| Subject | Kernel, Classifier | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| | RW, RF | 0.9344 | 0.9361 | 0.9344 | 0.9339 |
| | RW, SVC | 0.5738 | 0.5381 | 0.5738 | 0.5389 |
| S03 | WL, DT | 0.6508 | 0.6803 | 0.6508 | 0.6513 |
| | WL, RF | 0.7619 | 0.8105 | 0.7619 | 0.7615 |
| | WL, SVC | 0.7778 | 0.8293 | 0.7778 | 0.7768 |
| | SD, DT | 0.3492 | 0.3483 | 0.3492 | 0.3458 |
| | SD, RF | 0.4286 | 0.3730 | 0.4286 | 0.3941 |
| | SD, SVC | 0.3333 | 0.3627 | 0.3333 | 0.2982 |
| | RW, DT | 0.8413 | 0.8653 | 0.8413 | 0.8485 |
| | RW, RF | 0.8571 | 0.8868 | 0.8571 | 0.8638 |
| | RW, SVC | 0.3810 | 0.3030 | 0.3810 | 0.3221 |
| S04 | WL, DT | 0.7538 | 0.7971 | 0.7538 | 0.7594 |
| | WL, RF | 0.7692 | 0.7750 | 0.7692 | 0.7652 |
| | WL, SVC | 0.7385 | 0.7492 | 0.7385 | 0.7396 |
| | SD, DT | 0.2154 | 0.2284 | 0.2154 | 0.2213 |
| | SD, RF | 0.1846 | 0.1894 | 0.1846 | 0.1805 |
| | SD, SVC | 0.2615 | 0.1272 | 0.2615 | 0.1693 |
| | RW, DT | 0.8308 | 0.8386 | 0.8308 | 0.8313 |
| | RW, RF | 0.8923 | 0.8996 | 0.8923 | 0.8918 |
| | RW, SVC | 0.2615 | 0.6840 | 0.2615 | 0.1084 |
| S05 | WL, DT | 0.6667 | 0.6611 | 0.6667 | 0.6557 |
| | WL, RF | 0.7000 | 0.7304 | 0.7000 | 0.7078 |
| | WL, SVC | 0.3167 | 0.5711 | 0.3167 | 0.2870 |
| | SD, DT | 0.7833 | 0.7925 | 0.7833 | 0.7779 |
| | SD, RF | 0.7833 | 0.7929 | 0.7833 | 0.7748 |
| | SD, SVC | 0.3000 | 0.1822 | 0.3000 | 0.2054 |

| Subject | Kernel, Classifier | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| | RW, DT | 0.9833 | 0.9848 | 0.9833 | 0.9834 |
| | RW, RF | 0.9833 | 0.9848 | 0.9833 | 0.9834 |
| | RW, SVC | 0.2833 | 0.1848 | 0.2833 | 0.2017 |

# Appendix B

# Code

The implementation source code of the proposed functionality is presented below. The code is modularized and snapshots provided are based on the functionality and dataset.

## B.1 EEG Preprocessing

Listing B.1: Preprocessing of the EEG data from BCI-HCI repository

```
# Considering only Idle, Dual1Back data for our processing
label_map  = {"Idle": 0, "Dual-1-Back": 1}

all_files = {
    "Idle": listdir(join(data_path, "Idle")),
    "Dual-1-Back": listdir(join(data_path, "Dual-1-Back"))
}

graphs = []
labels_list = []
# Constants
fs = 128                    # Sampling frequency
```

```
idle_duration = 5 * fs     # 5 seconds of idle, converted to samples

for label, files in all_files.items():
  for filename in files:
    file_path = join(data_path, label, filename)
    edf = mne.io.read_raw_edf(file_path)
    edf.load_data()

    channel_data = edf.get_data()
    noiserow=[]
    for i in np.arange(0, channel_data.shape[0]):
         if np.sum(channel_data[i])==0:
            noiserow.append(i)
    channel_data=np.delete(channel_data, noiserow, axis=0)
    num_channels=channel_data.shape[0]

    experiment_data = channel_data
    NEpoch=int(experiment_data.shape[1]/(3*128))
    data=np.zeros((NEpoch,num_channels*num_channels))
    for i in range(NEpoch):
      epoch = experiment_data[:,3*i*128:3*(i+1)*128]
      cor_mat = np.corrcoef(epoch)
      if np.isnan(cor_mat).any():
          continue

      # Binarization of covarience matrix
      th=np.percentile(cor_mat.flatten(),35)
      cor_mat[cor_mat<th]=0
      cor_mat[cor_mat>=th]=1

      graph = nx.from_numpy_array(cor_mat)
      graphs.append(graph)
      labels_list.append(label_map[label])
```

## B.2 Graph kernel methods implementation

Listing B.2: Graph Kernel methods existing in the gklearn package to producing gram kernel matrices code for either of the matrices is run at a time the same code snippet was utilized for both CHB dataset and Cognitive Workload dataset

```
from gklearn.kernels.weisfeilerLehmanKernel import
   weisfeilerlehmankernel
kernel = 'weisfeilerlehmankernel'
kernel_matrix = weisfeilerlehmankernel(
    graphs,
    node_label='atom',
    edge_label='bond_type',
    height=3,
    base_kernel='subtree',
    parallel=None,
    n_jobs=None,
    verbose=True)

from gklearn.kernels import SpectralDecomposition
import multiprocessing

kernel = 'SpectralDecomposition'
graph_kernel = SpectralDecomposition(
    ds_infos=graphs,
    weight=0.1,
    p=None,
    q=None,
    edge_weight=None,
    sub_kernel='exp'
)
```

```
kernel_matrix = graph_kernel.compute(
    graphs,
    parallel=None,
    n_jobs=multiprocessing.cpu_count(),
    verbose=True
)


kernel = 'RandomWalk'
kernel_matrix = randomwalkkernel(
        graphs,
        compute_method=compute_method,
        weight=0.001,
        p=None,
        q=None,
        edge_weight=None,
        node_kernels=sub_kernels,
        edge_kernels=sub_kernels,
        node_label='atom',
        edge_label='bond_type',
        sub_kernel=sub_kernel,
        n_jobs=multiprocessing.cpu_count(),
        verbose=True
    )
```

## B.3 kPCA calculation

Listing B.3: Gram kernels produced is then to reduce dimentionality we process it with kernelPCA

```
from sklearn.decomposition import PCA, KernelPCA
kpca = KernelPCA(kernel='rbf', gamma=0.1)
gram_kpca = kpca.fit_transform(kernel_matrix[0])

X=gram_kpca
y=labels_list
```

## B.4 Model Training

Listing B.4: To obtain the performance scores below code snippet was utlized for both CHB dataset and Cognitive Workload datasets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=25)

param_grid_svc = {
  'C': [0.01, 0.1, 1, 10, 100],
  'kernel': ['linear', 'rbf', 'poly'],
  'gamma': ['scale', 'auto'],
  'class_weight': ['balanced', None]
}

param_grid_rf = {
  'n_estimators': [50, 100, 200],
  'max_depth': [None, 10, 20, 30],
  'min_samples_split': [2, 5, 10],
```

```
  'max_features': ['sqrt', 'log2'],  # Number of features to
     consider for splits
  'class_weight': ['balanced', None]
}

param_grid_dt = {
  'max_depth': [None, 10, 20, 30],
  'min_samples_split': [2, 5, 10],
  'class_weight': ['balanced', None]
}

svc = SVC(class_weight='balanced', random_state=42)
rf = RandomForestClassifier(oob_score=True, class_weight='balanced',
    random_state=42)
dt = DecisionTreeClassifier(class_weight='balanced', random_state
   =42)

# Perform grid search for each model
grid_search_svc = GridSearchCV(svc, param_grid_svc, cv=5, scoring='
   accuracy', n_jobs=-1)
grid_search_rf = GridSearchCV(rf, param_grid_rf, cv=5, scoring='
   accuracy', n_jobs=-1)
grid_search_dt = GridSearchCV(dt, param_grid_dt, cv=5, scoring='
   accuracy', n_jobs=-1)

# Fit the grid search
grid_search_svc.fit(X_train, y_train)
grid_search_rf.fit(X_train, y_train)
grid_search_dt.fit(X_train, y_train)

# Get the best estimators
best_svc = grid_search_svc.best_estimator_
best_rf = grid_search_rf.best_estimator_
```

```
best_dt = grid_search_dt.best_estimator_

print('best_svc', best_svc)
print('best_rf', best_rf)
print('best_dt', best_dt)

# Evaluate each model and print accuracy and AUC
models = { 'Decision Tree': best_dt, 'Random Forest': best_rf, 'SVC'
    : best_svc }
results = {}

for name, model in models.items():
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:, 1] if hasattr(
        model, "predict_proba") else y_pred

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    auc_score = roc_auc_score(y_test, y_pred_proba)

    # Display metrics
    results[name] = {'accuracy': accuracy, 'auc_score': auc_score, '
        precision': precision, 'recall': recall, 'f1': f1}
    print(f"{name} - Accuracy: {accuracy:.4f}, AUC: {auc_score:.4f},
        precision:{precision:.4f}, recall: {recall:.4f}, f1: {f1:.4
        f}")

    scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    print(f"{name} Cross-Validation Accuracy: {scores.mean():.4f}")
```

## B.5 mPLV corelation calculation for CHB dataset

Listing B.5: For CHB-MIT dataset we utilized mPLV calculation for obtaining corelation coefficients

```
def compPlv(epoch):
    nc,ns = epoch.shape
    plvk = np.zeros([nc,nc])
    data = sig.hilbert(epoch) #Compute the analytic signal, using
       the Hilbert transform.
    data = np.array(data)
    data = np.divide(data,abs(data))
    plvk = abs((np.inner(data[:,:], np.conj(data[:,:]))) / ns)
    return plvk
```

Listing B.6: For CHB-MIT dataset post preprocessing the data we consider only 30 ictal & preictal samples for our experiements and obtain a corelation matrix.

```
fs=256
num_channels=preictalepochs.shape[1]

nanct=0
# We considered only 30 epochs for the processing
NEpoch=30

adj_matrix = np.zeros((NEpoch*2, num_channels, num_channels))
labels=np.zeros(NEpoch*2)

k=0
for i in np.arange(0,NEpoch):
    epoch = preictalepochs[i,:,:] #channel_data[:,3*i*fs:3*(i+1)*fs]
    # the below line is either compPlv or corcoff based on the
```

```
        experiement being run. mPLV provided higher accuracy.
    cor_mat = compPlv(epoch)
    if(np.isnan(cor_mat).any()):
        nanct=nanct+1
        continue
    adj_matrix[k] = cor_mat
    #data[k]=cor_mat.flatten()
    labels[k]=0
    k=k+1

nanct=0
# We considered only 30 epochs for the processing
NEpoch=30

for i in np.arange(0,NEpoch):
    epoch = ictalepochs[i,:,:] #channel_data[:,3*i*fs:3*(i+1)*fs]
     # the below line is either compPlv or corcoff based on the
        experiement being run. mPLV provided higher accuracy.
    cor_mat = compPlv(epoch)
    if(np.isnan(cor_mat).any()):
        nanct=nanct+1
        continue
    adj_matrix[k] = cor_mat
    labels[k]=1
    k=k+1

#Binarizing Adjacency matrices before feeding for graph preparation
for i in np.arange(0,k):
    th=np.percentile(adj_matrix[i].flatten(),25)
    adj_matrix[i][adj_matrix[i]<th]=0
    adj_matrix[i][adj_matrix[i]>=th]=1
```

# Vita

**Candidate's full name:**

Rashmi Nagawara Muralinath

**University attended** (with dates and degrees obtained):

Bachelor of Computer Science & Engineering, Visvesvaraya Technological University, Bengaluru, 2010

**Publications:**

"Optimizing Spectral Graph Kernels for Time-Delay Embedding of Multichannel EEG Data." 8th International Conference on Algorithms, Computing and Systems (ICACS 2024), October 11-13, 2024. Hong Kong.

ACM ISBN 979-8-4007-1830-4/24/10

https://doi.org/10.1145/3708597.3708607