6CCS3PRJ Final Year
Task-fMRI Classification using Temporal Hypergraphs

Author: Valentin Magis

Supervisor: Dr. David Kohan Marzagao

Student ID: 20076746

April 12, 2024

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Valentin Magis

April 12, 2024

Acknowledgements

I express my sincere gratitude to Dr. David Kohan Marzagão for his encouragement to delve into uncharted territories and his support in navigating through uncertainties. My thanks also go to Dr. Ítalo Karmann Aventurato for providing valuable guidance on the neuroscientific aspects of this project.

Contents

Introduction

This report presents a project focused on advancing the field of functional brain network analysis through the adaptation of a novel machine learning (ML) model for the classification of task functional Magnetic Resonance Imaging (t-fMRI) data. The project leverages "Hypergraph Anomaly Detection in Brain" (HADiB), proposed by Behrouz et al. [6], which is designed for unsupervised anomaly detection using temporal hypergraphs. By adapting it to classify cognitive tasks using t-fMRI data from the Human Connectome Project (HCP) [45], this research aims to evaluate the model's performance in the classification of distinct activities undertaken by subjects during an fMRI scan, such as discerning the movement of the left or right hand.

Understanding the brain's functional dynamics during specific tasks has profound implications. Firstly, a model grounded in neuroscientific hypotheses that performs effectively does not only validate these hypotheses, but also enriches our comprehension of the brain's high-level dynamics. This advancement can, in turn, guide neuroscientific research into new directions, enhancing medical diagnostics and informing the design of intelligent systems.

The evolution of ML-based brain network analysis shows how neuroscientific insights have guided the development of models, from using conventional ML to sophisticated approaches that incorporate the brain's representation as a network, temporal dynamics, and higher-order interactions.

Initial ML-Based Approaches involved the use of Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) for classifying neurological disorders, demonstrating the utility of fMRI data in understanding neural patterns [98, 37, 21, 79].

Network Neuroscience: Graph-Based Approaches emerged from the insight of representing the brain as a graph with nodes representing regions in the brain and edges functional connections between them [4]. This led to the adoption of Graph Convolutional Networks (GCNs) and Graph Isomorphism Networks (GINs) for disease classification, highlighting the

significance of the brain's network structure [2, 74, 58, 56, 94, 60, 103].

The Importance of the Temporal Dimension: Temporal Approaches were developed in response to the recognition of how the brain's functional connectivity changes dynamically over time [12]. Techniques such as Recurrent Neural Networks (RNNs) have shown the power of integrating temporal dynamics [22, 101, 73, 34, 105]

Higher-Order Relationships: Hypergraph Approaches stem from the insight that multiple areas in the brain collaborate to achieve certain tasks, which can not be represented by conventional graphs [82]. Hypergraph-based methods have provided new insights into brain disorders by analysing resting-state fMRI (r-fMRI) data, showcasing the utility of capturing higher-order correlations [102, 99, 61, 51, 95]. Their application to the analysis of t-fMRI data represents a gap in the research.

However, despite the advancements brought about by these approaches, none has fully integrated all of these neuroscientific insights simultaneously. To bridge this gap, the concepts of temporal graphs (for the temporal dimension) and hypergraphs (for higher-order relationships) have been merged, creating temporal hypergraphs. HADiB exemplifies this integration by leveraging this representation of functional connectivity.

As HADiB performs remarkably well, it forms the basis for this project and will be adapted for a supervised approach to t-fMRI task classification. By undertaking this adaptation, the project endeavours to fill a gap in current research in two crucial ways: firstly, by employing temporal hypergraphs to analyse t-fMRI data, thereby leveraging the model's capability to encapsulate higher-order relationships and temporal dynamics; secondly, by aiming to classify more granular tasks, a novel approach as most existing models primarily focus on the classification of broader, higher-level tasks.

To this end, this report initially lays the groundwork with an introduction to crucial neuroscience concepts, graph analysis of brain networks and temporal hypergraphs. This foundation is followed by an exploration of related works, pinpointing gaps that HADiB addresses. An overview of the HCP1200 task dataset [45] sets the stage for the analysis. The design section critically analyses HADiB and proposes an adaption for t-fMRI classification. The implementation delves into the practical challenges encountered during this adaptation process. The evaluation section presents a critical analysis of the model's performance, contrasting it with existing approaches and discussing its limitations. Ethical considerations pertinent to neuroimaging research are addressed and finally, the report concludes with a reflection on the findings and suggestions for future research.

Background

2.1 Neuroscience

2.1.1 Brain Anatomy for Neuroimaging

Understanding the intricacies of brain anatomy is paramount for the field of neuroimaging, as it provides the foundational knowledge necessary to interpret the complex signals captured during various neuroimaging procedures. At the core of brain function and structure are neurons, the principal cells responsible for processing and transmitting information throughout the nervous system, as illustrated in Figure 2.1. Each neuron consists of a cell body (soma), dendrites, and an axon, which facilitate the transmission of electrical signals. The soma is the neuron's nucleus-containing part, dendrites receive signals from other neurons, and the axon transmits signals away from the neuron. Axons may be coated in a myelin sheath, a fatty substance that accelerates signal transmission, highlighting the division between gray matter, primarily made up of neuronal cell bodies and dendrites, and white matter, comprising myelinated axons. The division between grey and white matter is illustrated in Figure 2.2 [49].

The brain is also categorised into several lobes: frontal, parietal, occipital, and temporal, each associated with different functions and cognitive processes. These lobes are distinguished from one another by specific gyri (ridges) and sulci (grooves) that create the brain's folded surface, a characteristic feature that increases its surface area and allows for a higher density of neurons within a limited space.

One of the significant challenges in analysing neuroimaging data stems from individual variability in cortical folding patterns. This variability can complicate the task of accurately mapping mental activities across different individuals due to the unique anatomical features of each person's brain. To address this issue, neuroimaging researchers often rely on atlases,

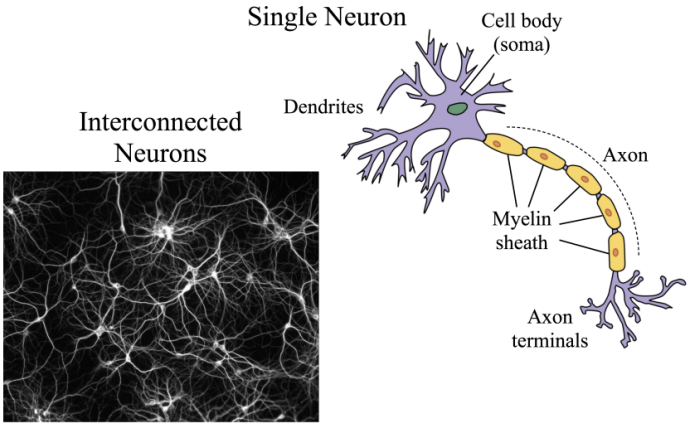


Figure 2.1: Schematic of a neuron [49].

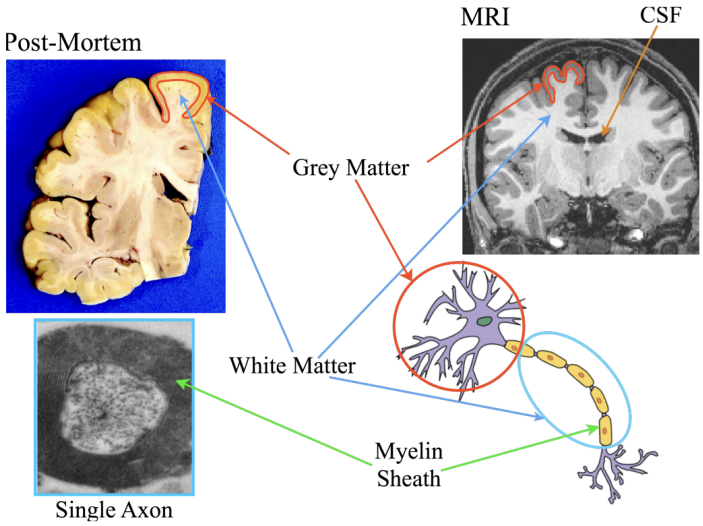


Figure 2.2: Gray and white matter tissue in a postmortem brain [49]

which are standardised representations of the brain. These atlases serve as reference points for comparing individual brain scans and are crucial for identifying specific brain regions and their associated functions across different individuals [49].

2.1.2 Neuroimaging

Neuroimaging plays a pivotal role in neuroscience, offering invaluable insights into the structural, functional, and molecular aspects of the human brain. This section provides an overview of the primary neuroimaging modalities—Electroencephalography (EEG), and Magnetic Resonance Imaging (MRI) with its subdivisions: Structural MRI, fMRI, and Diffusion MRI—elucidating their relationship to neuroanatomy, the significance of the Blood Oxygenation Level Dependent (BOLD) signal, the hemodynamic response function, and the challenges of directionality and causality in fMRI studies [49].

Types of Neuroimaging and Their Relation to Neuroanatomy

Neuroimaging modalities differ significantly in their methodologies and the type of information they reveal about the brain:

Electroencephalography (EEG) employs electrodes placed on the scalp to measure the electrical activity generated by neuron firing. It excels in temporal resolution, capturing the dynamics of brain activity in real-time. However, its spatial resolution is limited, making it challenging to pinpoint the exact locations of neuronal activity within the brain's complex neuroanatomy [72].

Magnetic Resonance Imaging (MRI) is a non-invasive technique for obtaining high-resolution images of the brain's structure and function. It operates on the principles of nuclear magnetic resonance, where a strong magnetic field and radiofrequency pulses interact with hydrogen atoms in the body to produce signals that are converted into images. It encompasses several techniques that provide detailed images of the brain's structure and function:

Magnetic Resonance Imaging (MRI) encompasses several techniques that provide detailed images of the brain's structure and function:

- *Structural MRI* offers high-resolution images of the brain's anatomy, enabling the visualisation of brain tissue, ventricles, and other structural features with great detail. It is

fundamental for identifying the brain's morphological characteristics and detecting abnormalities [50].

- *Functional MRI (fMRI)* focuses on the brain's metabolic changes associated with neuronal activity, leveraging the BOLD signal to map brain areas involved in cognitive tasks or resting state networks as it has been shown to be coupled to underlying neuronal activity [43, 55, 88]. A key parameter to fMRI is the Repetition Time (TR), which is the interval between successive pulse sequences applied to the same slice. TR is critical as it influences the temporal resolution and contrast of the images, thus affecting the quality of the structural and functional data acquired. Despite its powerful insight into brain function, fMRI is challenged by the indirect nature of the BOLD signal, reflecting blood flow changes rather than direct neuronal activity, and by the temporal delay inherent in the hemodynamic response [64].

- *Diffusion MRI* visualises the diffusion process of water molecules in the brain, providing insights into the microstructural organisation of brain tissue, particularly white matter tracts. This modality is crucial for understanding the brain's connectivity and the integrity of neural pathways [71].

Functional MRI: BOLD Signal and Hemodynamic Response

The BOLD signal underpins fMRI studies, offering a window into brain function by detecting changes in blood oxygenation and flow that occur in response to neural activity. However, interpreting this signal requires careful consideration of its relationship to neuronal events and the timing of the hemodynamic response, which introduces a delay between neuronal activity and observable BOLD changes. These aspects present challenges in deciphering the direct causes of observed functional patterns, necessitating careful preprocessing to infer neuronal activity from the BOLD response accurately [64].

At the microscopic level, neuronal activity, such as the generation of action potentials and synaptic transmission, leads to localised increases in blood flow [9]. This results in an increase in blood oxygenation levels, which BOLD fMRI measures as an indirect and secondary marker of neuronal activity. The hemodynamic response is relatively slow reaching its peak approximately 5–6 seconds after the onset of neuronal activity as illustrated in Figure 2.3, underscoring the indirect nature of BOLD signal measurements in capturing neural dynamics [9].

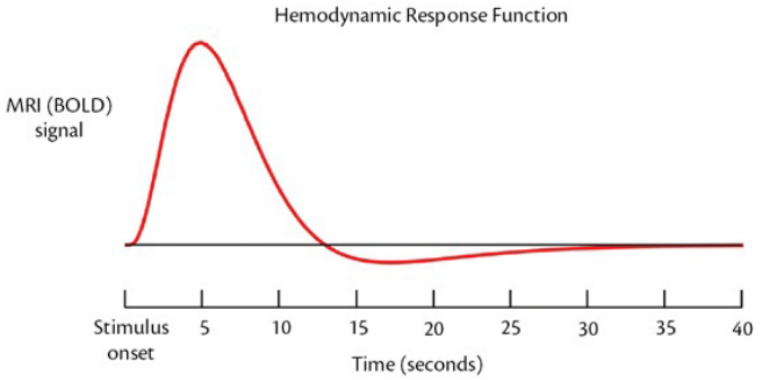


Figure 2.3: This illustration presents a typical hemodynamic response function (HRF) observed in the adult human brain, depicted through changes in MRI signal intensity, here referred to as "BOLD activation", over time in response to a brief stimulus. The graph highlights the delayed nature of the vascular response, peaking approximately 6 seconds post-stimulus and taking over 20 seconds to revert to baseline levels. In stark contrast, neuronal electrical activity is transient, lasting merely tens to hundreds of milliseconds. [9]

Challenges in fMRI: Directionality, Causality, and Application

FMRI studies confront issues of directionality and causality, as the BOLD signal does not directly measure neural firing but rather the secondary responses of the vascular system to neuronal activity. This limitation complicates the interpretation of functional connectivity and the determination of cause-effect relationships in brain processes. Despite these challenges, fMRI remains a powerful tool for investigating brain function, with applications ranging from understanding the neural basis of cognitive tasks in t-fMRI to exploring the brain's intrinsic connectivity networks in r-fMRI. This versatility extends to two primary experimental conditions under which subjects are usually scanned: resting state and task performance. R-fMRI captures brain activity while the subject is not engaged in any specific task, facilitating "mind wandering." This modality is instrumental in constructing models for disease detection and making predictions about different phenotypes, such as age or sex, by analysing the brain's default network activity. On the other hand, t-fMRI involves scanning the brain while subjects perform designated tasks, ranging from simple motor activities to complex cognitive functions. This approach allows researchers to identify and classify brain activity patterns associated with specific tasks, offering insights into the functional organisation of the brain and the neural underpinnings of various cognitive processes [27].

2.1.3 Preprocessing of fMRI data

Preprocessing is a crucial step in fMRI analysis, including task-based and resting-state studies. It involves a series of procedures aimed at correcting the data for artifacts and preparing it for subsequent analysis. This process is essential for ensuring the accuracy and reliability of the results derived from fMRI data. The primary reasons preprocessing is necessary are the correction for head movement, hardware noise, and physiological noise.

Motion Correction

The first step in the conventional preprocessing pipeline for fMRI data is motion correction, also known as realignment. This step is fundamental because even minute head movements can significantly distort the fMRI signal, leading to false results. Motion correction algorithms typically involve spatially aligning each brain volume in a time series to a reference volume, often chosen as the first volume or a volume acquired in the middle of the scanning session. This alignment is done using rigid body transformations, which include translations and rotations, to correct for head motion. The process ensures that the time series data are spatially consistent, allowing for accurate voxel-wise comparisons across time. Additionally, motion correction algorithms provide motion parameters that describe the detected movements. These parameters can be used in further analyses to model and remove motion-related artifacts from the fMRI data [9].

Slice Timing Correction

The primary objective of slice timing correction is to amend temporal discrepancies arising from the sequential acquisition of image slices within each TR cycle. Given the varying acquisition times for different slices—some at the commencement of the TR and others subsequently—the correction aims to standardise the timepoint of data acquisition across all slices. This adjustment is especially pertinent when employing sequences with longer TRs, such as 3 seconds, where the acquisition time disparity can be substantial. However, the advent of accelerated multiband EPI sequences, reducing the TR to approximately 1 second or less, diminishes the impact of these timing differences due to the inherently slow hemodynamic response. The correction employs temporal interpolation to align the BOLD timecourses, albeit at the cost of slight temporal smoothing and the consequent loss of high-frequency information. Moreover, the interaction of slice timing correction with other preprocessing steps, such as motion correction and spatial smoothing, introduces complexity that may not be fully rectifiable. Con-

sequently, the decision to apply slice timing correction is contingent upon the specific TR, the study's objectives, and the subsequent analytical procedures. In scenarios where precise timing is integral to the study hypothesis, even with fast TRs, slice timing correction might be deemed necessary [9].

Spatial Smoothing

Spatial smoothing is a preprocessing step applied to fMRI data to enhance the signal-to-noise ratio (SNR) and to account for anatomical variations across subjects. It involves convolving the data with a Gaussian kernel, which averages the signal over neighboring voxels based on the kernel's size, specified by its full width at half maximum. While smoothing can increase the SNR and improve statistical power, it also reduces spatial resolution and may blur important anatomical details. The degree of smoothing applied should be carefully considered based on the goals of the study and the spatial scale of the neural activations of interest. For instance, studies focusing on small brain structures may require less smoothing to preserve spatial specificity [9].

The necessity of preprocessing steps, such as motion correction, slice timing correction, and spatial smoothing, underscores their importance in the fMRI data analysis pipeline. By addressing motion artifacts, temporal discrepancies in slice acquisition, and enhancing SNR, these steps ensure that the data are optimally prepared for extracting reliable and meaningful neural activity patterns.

2.2 Graphs for Brain Network Analysis

2.2.1 The Brain as a Graph

The standard process of analysing functional connectivity involves deriving networks from fMRI data, calculating connectivity between nodes, and utilising these measures for downstream tasks.

On a high level, the brain can be conceptualised as a dynamic functional system comprised of smaller regions that interact with each other over time. The activation of these brain regions is quantifiable through fMRI scans, which measure BOLD signals. These signals reflect local changes in brain blood flow and oxygenation, indicators closely linked to neuronal activity [43, 88, 55].

A graph representation of the brain's functional connectivity can be constructed from the

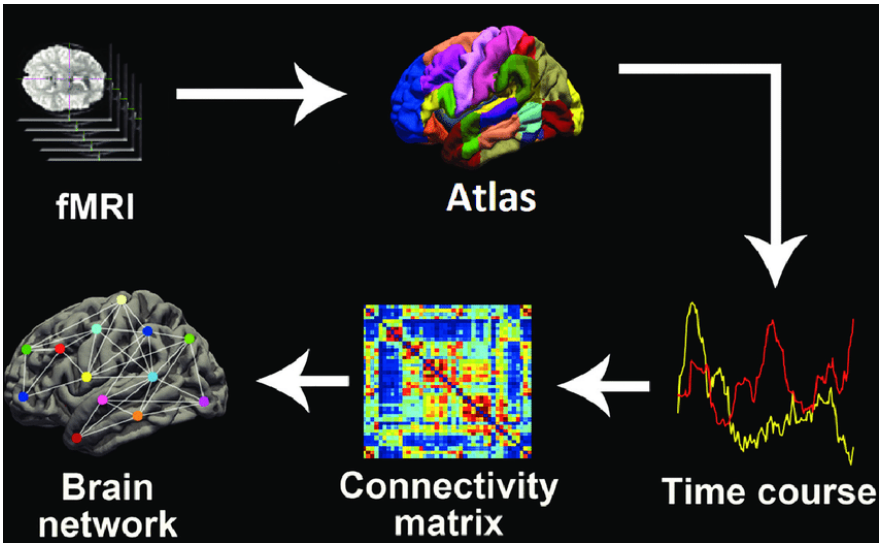


Figure 2.4: From fMRI to Brain Connectivity: This image depicts the process of transforming fMRI data into a visual representation of the brain's network. First, fMRI data is collected in a scanner, then the data is overlaid on a brain atlas to map activity to brain regions. The correlation between brain regions is computed to generate a connectivity matrix, representing the functional connections between different regions. Finally, the connectivity is visualised as a graph, with coloured nodes representing regions and edges indicating functional connections. Image: [19].

multivariate time series data acquired via fMRI scans. In this graph, nodes represent the brain's regions of interest (ROIs), as defined by a specific atlas, and the edges denote the pairwise connectivity between these ROIs. The determination of edges' weights is typically based on Pearson correlation, offering a quantitative measure of the functional relationships between different brain regions. This graph-based approach to analysing brain connectivity is illustrated in Figure 2.4, which outlines the sequence from acquiring fMRI data, mapping this data to specific brain regions using a brain atlas, calculating the connectivity matrix through correlation analysis, and ultimately visualising the functional network of the brain as a graph. The visual representation encompasses nodes colour-coded to represent different regions and edges that signify the functional connections between these areas [54]. This graph representation of the brain can then be used for downstream analysis tasks like diagnostics based on topological graph measures, graph kernels and machine learning models like graph neural networks (GNNs).

2.2.2 Limitations of Graphs and Hypergraphs as Solution

Graph-based models, while robust in delineating pairwise interactions, are limited when it comes to modelling complex, multi-entity interactions. Traditional graphs treat connections

Figure 2.5: A conventional graph is ambiguous when trying to model relationships between groups of nodes. Image adjusted from [53].

between entities as binary, inherently losing information on group dynamics. The inadequacies of such an approach become particularly evident in the context of functional brain connectivity. Studies have illuminated the presence of higher-order interactions within brain functionality, suggesting that the interactions between brain regions are not exclusively pairwise but often involve complex, simultaneous connectivity between multiple regions [82].

To capture the essence of these higher-order interactions, a more nuanced data structure is required. Hypergraphs offer a solution to this through the concept of hyperedges, which can encompass relationships between any number of nodes, extending beyond the binary scope of traditional graphs. This capability allows for a more accurate representation of group collaborations, as seen in the example of co-authorship, where traditional graphs would fail to differentiate a single collaboration between three authors and three different collaborations between separate pairs of authors, leading to the same pairwise connections 2.5.

2.2.3 Temporal Modelling using Graphs and Hypergraphs

Understanding brain function requires not just a snapshot of neural activity but a dynamic perspective that captures the brain's temporal fluctuations. Recent advances in fMRI analysis highlight the significance of these temporal dynamics, revealing the brain's functional connectivity as a vibrant and evolving network. Traditional static models fall short by ignoring these fluctuations, which embody the brain's adaptive responses to cognitive tasks and environmental changes. [54]

Key to this understanding is the method of sliding window correlation (SWC), which analyses changes in connectivity over specified time intervals as shown in Figure 2.6. This technique marks a departure from static analysis models by allowing for the observation of the brain's functional network on a more granular temporal scale. By examining the variations in connectivity across different windows, researchers can identify transient brain states—specific periods where the connectivity pattern is distinct from the overall average. These states are crucial

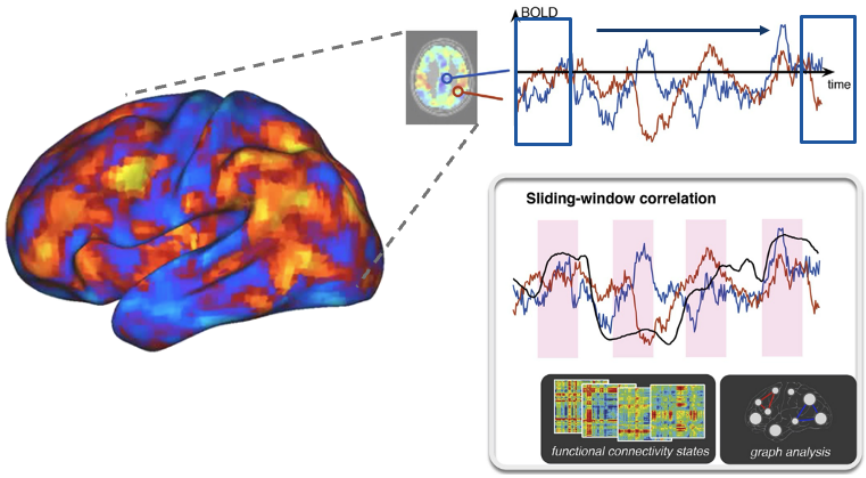


Figure 2.6: Visualisation of SWC applied to fMRI BOLD signals. The process involves analysing these extracted signals over successive time windows to capture the dynamic changes in functional connectivity. The lower panel illustrates how these time-windowed correlations are used to construct a temporal sequence of functional connectivity states. Image: [54]

for mapping out how the brain navigates between different cognitive functions or responds to external stimuli.

Temporal modelling, thus, offers a window into the brain's adaptive mechanisms, showcasing how dynamic reconfigurations of networks facilitate cognitive flexibility and how disruptions in these dynamics may correlate with neurological conditions. It emphasises the brain's non-static nature, urging a shift towards analytical models that can take the temporal dimension into account.

However, considering that activity patterns over time transcend pairwise interactions, hypergraphs emerge as a useful model for capturing spatiotemporal higher-order patterns. Temporal hypergraphs can be generated by applying a technique analogous to SWC for constructing hypergraphs for time steps. Behrouz et al. introduce HADiB, which leverages temporal hypergraphs to capture these spatiotemporal higher-order patterns precisely [6]. Its architecture will be discussed in detail throughout the report.

2.3 Introduction to Temporal Hypergraphs

Temporal hypergraphs are an extension of hypergraphs that allow modelling complex relationships that evolve over time. Unlike traditional graphs, where edges represent binary relationships between pairs of nodes, hypergraphs generalise this concept by using hyperedges that can connect any number of nodes. This section provides an overview of the formal definitions and

concepts underpinning temporal hypergraphs.

2.3.1 Temporal Hypergraph Definition

A temporal hypergraph is a sequence of hypergraph snapshots over discrete time intervals, formally defined as follows [6]:

Definition 1 *A **temporal hypergraph** $\mathcal{G}$ is a sequence of hypergraph snapshots $\mathcal{G} = \{G^{(t)}\}_{t=1}^{T_{max}}$ for time steps $t = 1, 2, \ldots, T_{max}$, where each snapshot $G^{(t)} = (V, E^{(t)}, X^{(t)})$ consists of:*

- *A set of nodes V, consistent across all snapshots.*
- *A set of hyperedges $E^{(t)}$ at time t, with each hyperedge $e_i \subseteq V$, i.e. connecting a subset of nodes.*
- *A matrix of node attributes $X^{(t)} \in \mathbb{R}^{|V| \times f}$, where each row corresponds to a node's encoding of size f at time t.*

Each hypergraph snapshot captures the connectivity and attribute information at a specific time point, facilitating the analysis of dynamic systems with higher-order relations. Note that this definition assumes that the nodes are static and only the hyperedges change over time.

2.3.2 Incidence Matrix

The incidence matrix is a binary matrix that represents the connectivity of a hypergraph at a specific time t, defined as follows [6]:

Definition 2 *The **incidence matrix** $A^{(t)}$ of a hypergraph snapshot $G^{(t)}$ at time t is a $|V| \times |E^{(t)}|$ binary matrix where each row corresponds to a node in V and each column corresponds to a hyperedge in $E^{(t)}$. The entries of $A^{(t)}$ are defined as follows:*

$$A_{ue}^{(t)} = \begin{cases} 1 & \text{if node } u \text{ is part of hyperedge } e \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

2.3.3 Node Neighbourhood in Temporal Hypergraphs

In a temporal hypergraph, the concept of neighbouring nodes is defined based on the connectivity through shared hyperedges at a given time t. This relationship is crucial for understanding

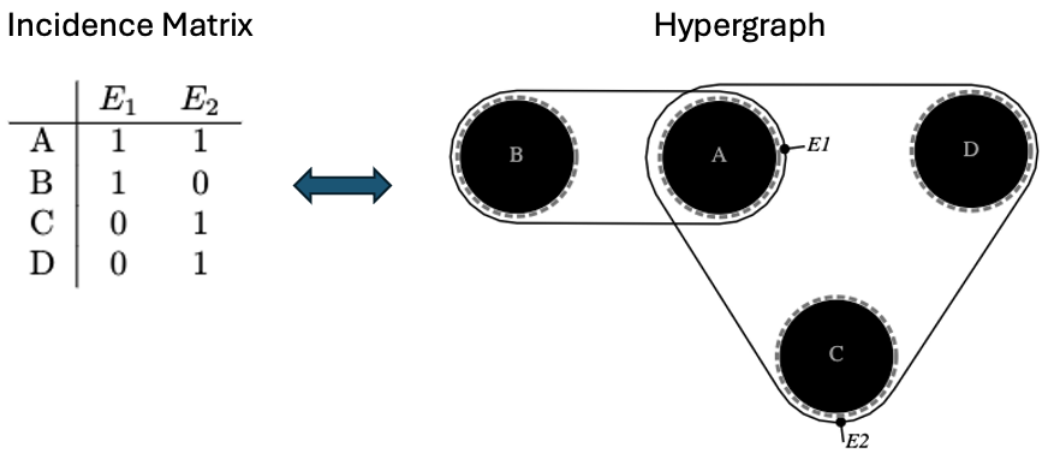


Figure 2.7: An incidence matrix representing a hypergraph with four nodes and two hyperedges (left) and the corresponding hypergraph representation (right). Hypergraph generated with [44].

the structure and dynamics of interactions within the hypergraph. The neighbourhood of a node is defined as follows

Definition 3 *Let $\mathcal{G} = \{G^{(t)}\}_{t=1}^{T}$ denote a temporal hypergraph, where each $G^{(t)} = (V, E^{(t)}, X^{(t)})$ represents a hypergraph snapshot at time t. Two nodes $u, v \in V$ are said to be* ***neighbours at time*** *t, denoted as $u \sim_t v$, if and only if there exists a hyperedge $e \in E^{(t)}$ for which $u, v \in e$. Formally, the neighbourhood relation $\sim_t$ at time t is defined by the existence of such an e, as follows:*

$$u \sim_t v \iff \exists e \in E^{(t)} : u \in e \wedge v \in e \wedge u \neq v.$$

2.3.4 Converting a Hypergraph into a Bipartite Graph

Transforming a hypergraph $H = (V, E)$ into a bipartite graph $B = (U, W, E')$ enables its analysis with bipartite graph techniques, in the case of this report particularly relevant in the context of message passing. A bipartite graph, or bigraph, consists of two disjoint sets of vertices, U and W, with edges only between vertices from different sets. In the conversion process, U represents H's original node set V, while W stands for H's hyperedges. Each hyperedge $e \in E$, connecting nodes $v_1, v_2, \ldots, v_n$, is represented in B by edges $(v_1, e), (v_2, e), \ldots, (v_n, e)$, linking nodes in U to their respective hyperedge counterparts in W, as visualised in Figure 2.8. This approach maintains the incidence relationships, facilitating the representation of hypergraph properties within a bipartite framework. [15].

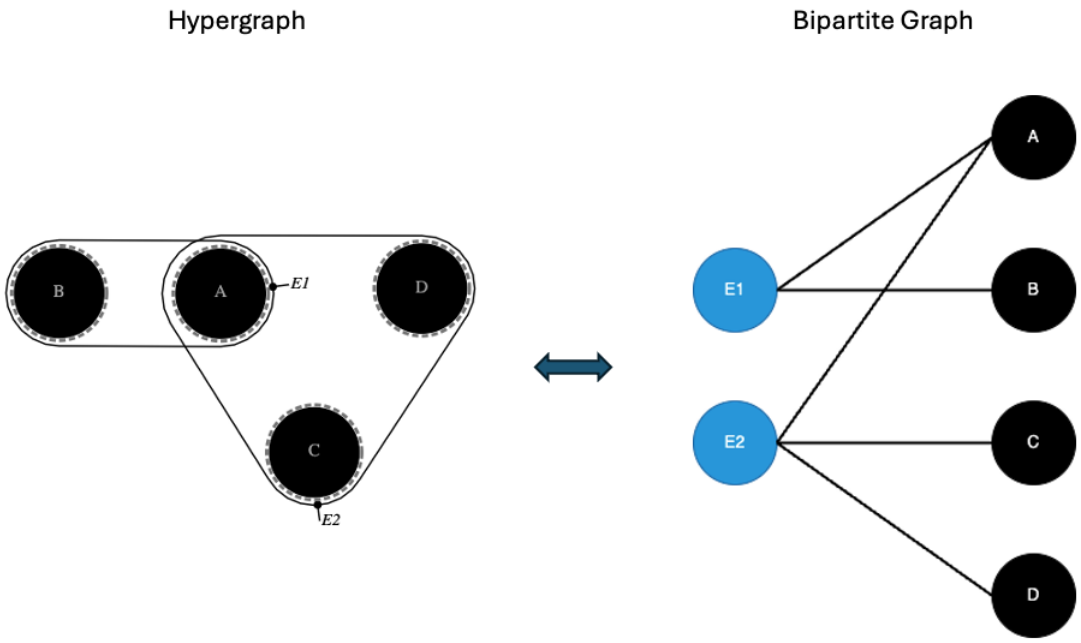


Figure 2.8: Conversion of a hypergraph to a bipartite graph. The hypergraph (left) consists of nodes A, B, C, and D, with hyperedges $E1$ and $E2$. The corresponding bipartite graph (right) represents nodes as one node set and hyperedges as another. Nodes in the bipartite graph are linked to hyperedge-nodes if they are contained in the hyperedge, exemplifying the preservation of incidence relations between nodes and hyperedges during the transformation. Hypergraph generated with [44].

2.3.5 Message Passing on Hypergraphs and its Simplification through Bipartite Graph Conversion

Message passing is a fundamental technique in graph theory and network analysis, playing a critical role in areas such as GNNs and more. It facilitates a wide range of applications, from node classification and graph clustering to the advanced modelling of interactions within complex systems. Fundamentally, message passing involves the iterative exchange and aggregation of information between nodes, enabling the refinement of node representations based on the local neighbourhood structure.

In the context of hypergraphs, the message-passing mechanism introduces additional complexity due to the nature of hyperedges, which connect an arbitrary set of nodes. This complexity is manifested in the management of multi-node interactions within a single message-passing step, which can entail sending messages from nodes to hyperedges (to aggregate node information at the hyperedge level), from hyperedges to nodes (to distribute aggregated information back to the nodes), or directly between nodes (to exchange information among neighbouring nodes).

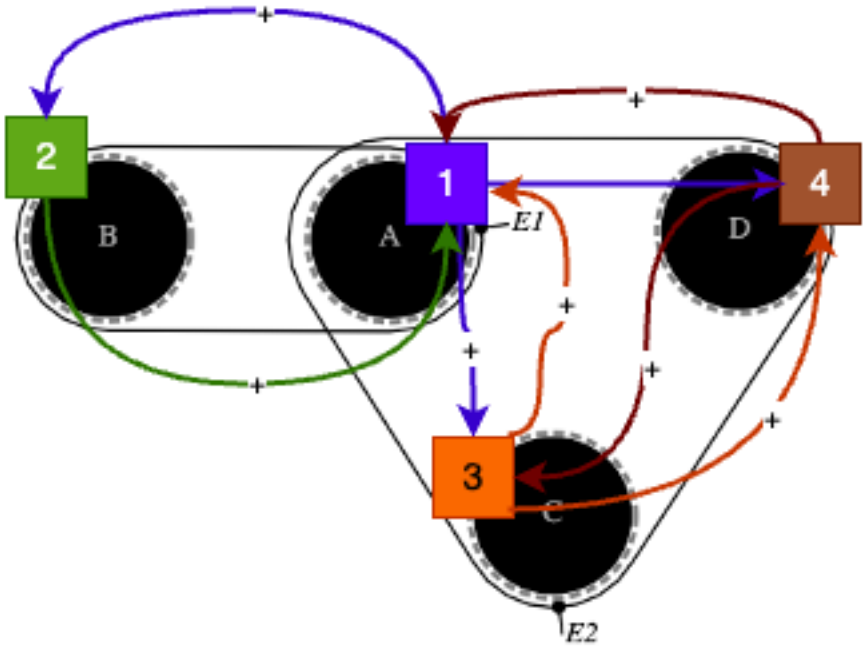


Figure 2.9: Schematic Representation of Message Passing in a Hypergraph. This image demonstrates the message-passing mechanism where each node sends its encoded value (denoted by the coloured numbers) to all other nodes within the same hyperedge. The received values are then aggregated. For instance, node A with an encoding '1' broadcasts this value to B in hyperedge E1, and C and D in hyperedge E2, and simultaneously receives the encodings from these nodes. The culmination of this process for node A is the sum of its original value and the encodings received from B, C, and D, signifying the collective informational influence of the connected nodes. The process exemplifies how message passing in hypergraphs integrates neighbourhood encodings to update each node's representation.

Formal Definition of the Node-to-Node Message Passing Objective

Consider a hypergraph where each node's encoding is represented by a single numerical value. The aim of one iteration of message passing is to update each node's encoding to reflect not only its own information but also the aggregated information of all other nodes within its neighbourhood. For a node v at time step t, the updated encoding v' after one round of message passing is defined as:

$$v' = v + \sum_{u \in \mathcal{N}(v)} u$$

where $\mathcal{N}(v)$ denotes the set of v's neighbouring nodes, representing the collective influence of the immediate network around each node. This can be computed as described in algorithm 1. The execution of a single round of message passing is illustrated in Figure 2.9.

The requirement of finding the neighbours of a node based on an incidence matrix can pose problems when integrated into a ML-model, which will be discussed in more detail in section 6.7. Performing message passing on bigraphs instead can rectify this issue.

Algorithm 1 Node-to-Node Message Passing in Hypergraphs

Input: Hypergraph $H = (V, E, X)$
Output: Updated node encodings X' after message passing
for each node $v \in V$ **do**
    Initialise v' to v
end for
for each hyperedge $e \in E$ **do**
    for each pair of nodes (u, v) in e **do**
        $v' \leftarrow v' + u$
        $u' \leftarrow u' + v$
    end for
end for
for each node $v \in V$ **do**
    Update v with v'
end for

Simplification through Bipartite Graph Conversion

Converting a hypergraph into a bipartite graph simplifies the message passing process by representing the original complex multi-node interactions in a more manageable form. In this bipartite representation, one set of nodes corresponds to the original nodes of the hypergraph, while another set represents the hyperedges. This setup facilitates a straightforward application of message passing techniques developed for traditional graphs.

Message Passing on Bipartite Graph Representation: In the bipartite graph context, message passing from nodes to nodes, as represented in a hypergraph, translates into a two-round process. The first round involves passing messages from the original nodes to the hyperedge-nodes, followed by a second round where messages are passed back from the hyperedge-nodes to the original nodes. Effectively, the neighbourhood of a node in this bipartite structure extends two steps away, encompassing the nodes connected indirectly through shared hyperedge-nodes [42]. The corresponding algorithm and a visual example are provided in 2 and 2.10, respectively.

This approach to message passing leverages the bipartite graph's structure to effectively mirror the hypergraph's higher-order interactions with a methodology that is both computationally efficient and conceptually straightforward.

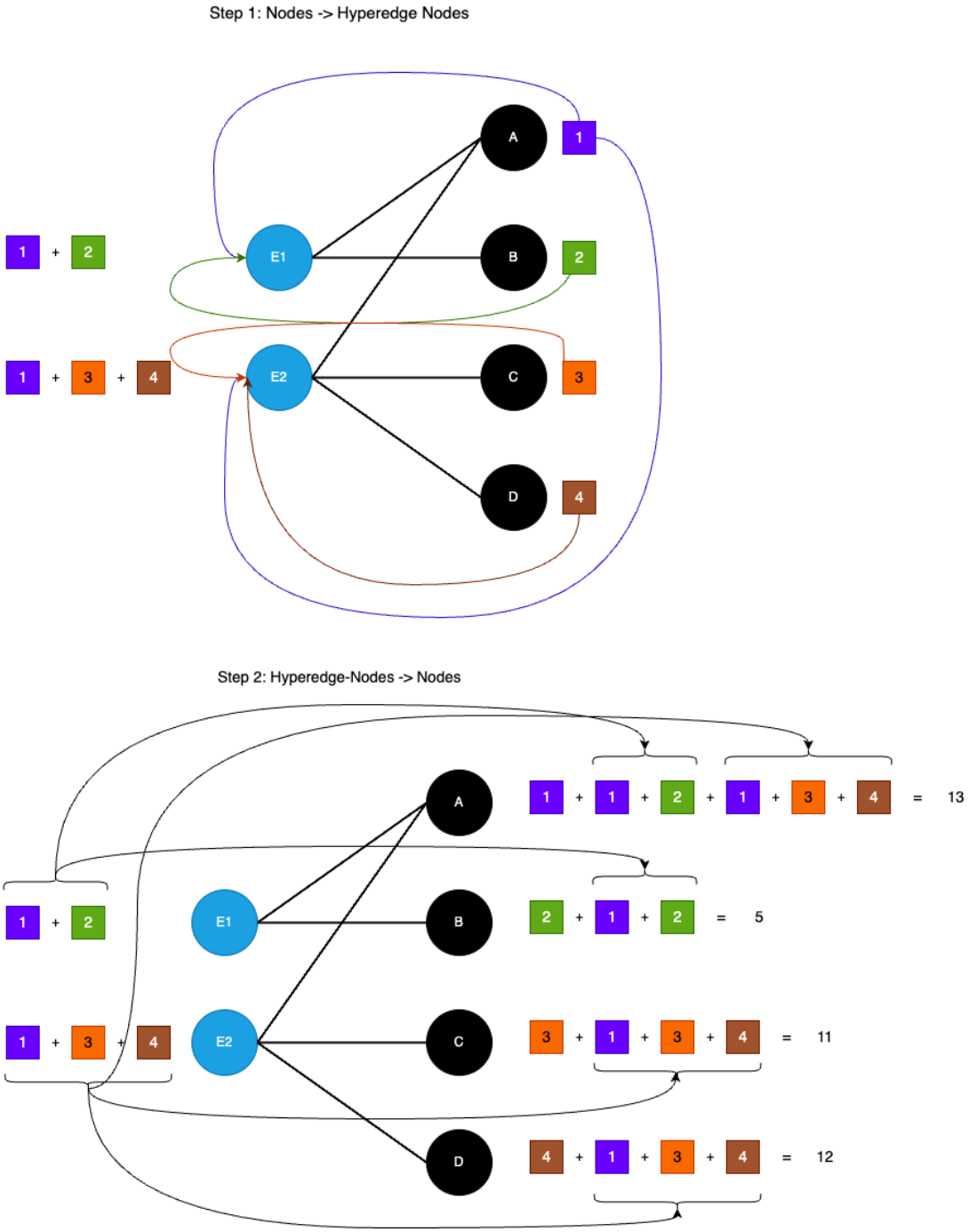


Figure 2.10: Illustration of Message Passing in a Bipartite Graph representing a Hypergraph. This diagram delineates the two-step message passing process, with Step 1 depicting the aggregation of messages from individual nodes to their respective hyperedge-nodes (E1 and E2), and Step 2 showing the distribution of these aggregated messages back to the nodes. Notably, node A uniquely receives updates from two distinct neighbourhoods via hyperedge-nodes E1 and E2, showcasing the capability of this approach to incorporate multi-node interaction dynamics into the node's final encoding.

Algorithm 2 Simplified Message Passing on Bipartite Graph

Input: Bipartite graph representation of a hypergraph $B = (U, W, E')$
Output: Updated node encodings after message passing
Initialise node encodings in U based on the original hypergraph
Step 1: Nodes to Hyperedge-Nodes
for each node $u \in U$ **do**
    Aggregate encoding of u at connected hyperedge-nodes in W
end for
Step 2: Hyperedge-Nodes to Nodes
for each hyperedge-node $w \in W$ **do**
    Distribute aggregated encoding from w back to connected nodes in U
end for
Each node in U now contains the aggregated encodings of all nodes within its two-step neighbourhood

Related Work

3.1 Overview

The advent of fMRI has revolutionised the study of the human brain, providing a window into its dynamic function in health and disease. This chapter reviews methods for fMRI analysis. The overview is not exhaustive and intends to categorise the methodologies that have been applied, with a focus on graph-based approaches. It distinguishes between the use of r-fMRI for detecting or classifying phenotypical information such as neurological disorders, sex, age and cognitive abilities, and the use of t-fMRI for classifying the performed tasks. For both types of fMRI, it provides an overview of *non-graph-based, graph-based and hypergraph-based* methods. Notably, while hypergraph-based methods have shown promise in r-fMRI analysis, their application in t-fMRI remains unexplored. This gap presents an opportunity to investigate their efficacy in classifying task-related brain activity.

3.2 Non-Graph-Based Approaches

Early research in fMRI analysis has largely focused on methodologies that treat the data as Euclidean, structured within a 2-dimensional grid, like images. Within this sphere, two primary categories emerge, based on the processing of spatial and temporal dimensions: non-temporal and temporal methods, both applied to r- and t-fMRI.

3.2.1 Resting-State fMRI

Non-Temporal Approaches

Non-temporal approaches primarily leverage the Functional Connectivity Matrix (FCM), a construct that gauges similarity between time series extracted from ROIs. The FCM is akin to

an adjacency matrix of a graph, albeit its utilisation in this context does not extend to graph-based analytics [98]. This methodological framework has seen applications in neural disorders classification, including autism spectrum disorder [37, 21] and attention deficit hyperactivity disorder [79] through the deployment of MLPs and CNNs.

A significant limitation associated with FCMs is their dependency on a predefined similarity measure, introducing potential biases. The Pearson correlation coefficient, commonly employed to measure co-activation between ROIs, is restrictive to linear associations, potentially skewing the analysis [22, 101].

Temporal Approaches

Temporal approaches in r-fMRI analysis have gained traction, particularly with the rising interest in dynamic functional connectivity [12]. This line of inquiry utilises time-windowed FCMs, accommodating the evolving nature of brain connectivity. The integration of CNNs with RNNs exemplifies this approach, aiding in the prediction of fluid intelligence [22] and the identification of major depressive episodes [101]. However, similar to non-temporal methods, these approaches are hampered by the arbitrary selection of similarity measures and window lengths (see Sliding Window Correlation in 2.2.3), complicating their application [46].

An emergent strategy diverging from FCM representations involves processing the entire four-dimensional brain volume time series via convolutional RNNs. This method, although comprehensive, tends to overlook the empirical evidence suggesting that functional brain activity is often task-localised and exhibits strong spatial correlations [86, 7, 30, 75]. The resultant feature learning, while extensive, is prone to computational inefficiencies and redundancy.

3.2.2 Task-fMRI

The analysis of t-fMRI data employs both spatial and spatiotemporal methods to interpret or classify brain activity measured while subjects perform tasks.

Spatial Approaches

Spatial methods analyse connectivity between brain regions based on spatial data, disregarding temporal information. Ertuğrul et al. (2019) demonstrated this approach by using Support Vector Machines (SVMs) to classify cognitive states within the HCP1200 t-fMRI dataset [45], based on encodings of the local connectivity patterns of the brain. Achieving 77.49% accuracy, this study highlights the efficacy of spatial analysis in task classification [73].

Spatiotemporal Approaches

Spatiotemporal methods, on the other hand, integrate both spatial and temporal data to capture the dynamic nature of brain activity. Gui et al. demonstrated the potential of integrating spatial frequency information of fMRI data at each time point with an improved circle-EMD (Empirical Mode Decomposition) algorithm. This method is designed to enhance classification efficiency by filtering out low-frequency and redundant noise, using logistic regression, SVMs, and deep neural networks for classification, with the deep neural network approach achieving the highest accuracy with around 80%. This approach underscores the value of considering the spatiotemporal complexity of brain activity for task classification [34].

Building on the importance of temporal dynamics, Zhao et al. (2022) introduced a novel Group Deep Bidirectional RNN (Group-DBRNN) to decode task sub-type states from individual fMRI volume data points. Their model capitalises on the temporal dependencies between adjacent fMRI data points through a bidirectional RNN layer and a multi-task interaction layer. This complex model structure facilitates the capture of latent temporal dependencies across different tasks, thereby enhancing the discriminative learning of functional brain states under various sub-type tasks. Impressively, their model achieved an average classification accuracy of 91.34% across 24 sub-type brain states and an average seven-task classification accuracy of 95.55%, indicating a significant improvement over existing state-of-the-art methods. This highlights the potential of deep learning models, especially those that integrate spatiotemporal data, to advance the understanding of task-related brain activity [105]. Due to its high accuracy on sub-task classification, Group-DRBNN is one of the main competitors for the model developed in this paper

3.3 Graph-Based Approaches

Graph-based analysis offers a structured perspective on fMRI data, treating the brain as a network of interconnected regions. This section outlines the distinction between spectral and spatial methods in graph theory, and their application to r- and t-fMRI data.

3.3.1 Advantages of Graph-Based Approaches

The shift from non-graph-based to graph-based methodologies in fMRI analysis marks a significant advance, directly addressing the former's limitations through the comprehensive capabilities of graph theory. This approach unfolds several critical advantages:

- **Comprehensive Network Insights:** Graph-based analysis transcends local pairwise interactions by offering a global view of the brain's network by for instance using topological graph measures, revealing how individual regions contribute to overall brain function. This holistic perspective enhances the understanding of functional integration and segregation, crucial for interpreting the brain's complex dynamics [85].

- **Enhanced Efficiency and Resilience:** Graph theory evaluates the brain's network for efficiency in information transfer and resilience against disruptions. These analyses provide insights into the underlying architecture that supports the brain's operational robustness and adaptability [1].

- **Dynamic Connectivity Analysis:** Emphasising the brain's network variability, graph-based methods can capture changes in connectivity patterns over time using temporal graphs. This approach aligns with the view of the brain as a dynamically evolving system, offering a more nuanced understanding of neural plasticity and functional reorganisation [12].

By addressing the limitations of non-graph-based methods, graph-based approaches enrich fMRI analysis, providing a deeper, network-centric insight into the brain's functional organisation and its capacity for dynamic change.

3.3.2 Spectral vs. Spatial Methods

Graph-based learning on graphs predominantly employs either spectral-based convolution or spatial-based message passing techniques. Spectral-based convolution methods, exemplified by GCNs [57] and Hypergraph Neural Networks (HGNN) [24], leverage a Laplacian Matrix to perform feature smoothing across vertices, embedding both low-order and high-order structural information into vertex features. Conversely, spatial-based message passing methods, including GraphSAGE [35], Graph Attention Networks (GAT) [93], and HGNN+ [29], facilitate message passing mechanisms to encode structural information into node features. These embedded features are instrumental for downstream tasks like classification, regression, and link prediction [23]. These methods form the theoretical basis of graph-based approaches to fMRI analysis.

3.3.3 Resting-state fMRI

Spectral Approaches

Spectral graph convolutions have been successfully applied to r-fMRI data analysis for various classification tasks. Arslan et al. (2018) utilised GCNs to identify brain regions critical for sex classification, demonstrating high accuracy and neurobiological relevance [2]. Similarly, Parisot et al. (2017) leveraged GCNs for population-based disease prediction by integrating imaging and non-imaging data, achieving superior prediction accuracy [74]. Ktena et al. (2018) introduced a siamese GCN for learning graph similarity metrics, significantly enhancing classification performance by processing fMRI data through spectral graph convolutions [58].

Spatial Approaches

Spatial methods like the Graph Isomorphism Networks (GINs) have been applied to classify subjects' sex based on brain functional connectivity, offering improved classification performance and interoperability [56]. Wang et al. (2022) proposed a novel methodology for extracting the default mode network from r-fMRI data using GraphSAGE, demonstrating robustness and reliability over traditional methods [94].

Spatiotemporal Approaches

Dynamic graph representation learning models, such as BrainTGL introduced by Liu et al. (2023), leverage temporal graph pooling to capture spatial and temporal correlations within functional brain networks, significantly enhancing diagnostic tasks [63]. The Spatio-Temporal Graph Convolutional Network (ST-GCN) by Gadgil et al. (2020) effectively captures the dynamic nature of functional connectivity, improving predictions of gender and age [28].

3.3.4 Task-fMRI

Spatial Approaches

BrainGNN, developed by Li et al. (2021), employs ROI-aware graph convolutional layers for identifying neurological biomarkers, providing insights into brain region importance for specific conditions with applications in individual-level and group-level analysis [60].

Spatiotemporal Approaches

Ye et al. (2023) proposed the Spatial Temporal-pyramid Graph Convolutional Network (STpGCN), leveraging multi-scale spatial-temporal pathways to capture dynamic brain activities. This framework notably improves decoding performance and explainability by pinpointing task-related brain regions through sensitivity analysis [103]. With 92% accuracy in sub-task classification in the HCP1200 task dataset [45], it is the currently best performing model.

3.4 Hypergraph-Based Methods

Recent advances in neuroimaging analysis have highlighted the usefulness of hypergraph-based methods for elucidating complex Functional Connectivity Networks (FCNs) within the brain. These methods have been instrumental in advancing the understanding of various neurological conditions, including autism spectrum disorders (ASD), mild cognitive impairment (MCI), and attention deficit hyperactivity disorder (ADHD). All approaches mentioned in this section are spatial and non-temporal. The studies primarily use r-fMRI data, although in some cases additional modalities, such as t-fMRI, are also employed. The objective of all studies is to classify a neurological disease or phenotype, such as age or learning ability.

Yang et al. (2023) presented an innovative approach for constructing high-order FCNs using hypergraphs for the diagnosis ASD, leveraging r-fMRI data from the ABIDE dataset [37]. Their methodology showcased superior diagnostic accuracy through the integration of high-order FCNs and traditional low-order FCNs [102].

Xiao et al. (2020) developed a multi-hypergraph learning framework to analyse brain functional connectivity from r- and t-fMRI data obtained from the Philadelphia Neurodevelopmental Cohort, aiming at classifying individuals based on learning abilities assessed by the Wide Range Achievement Test scores (WRAT) [84]. Their results outperformed conventional graph-based and unweighted hypergraph-based methods, underlining the efficacy of weighted hypergraphs in representing brain functional connectivity [99].

Li et al. (2019) improved MCI classification by integrating resting-state BOLD and arterial spinal labeling (ASL) fMRI data into hyper-connectivity networks using a functionally-weighted LASSO algorithm. This approach harnesses the high temporal resolution of BOLD and the physiological detail from ASL to accurately represent brain functions. The resulting multimodal networks showed enhanced ability to distinguish MCI, outperforming standard unimodal and

pairwise connectivity methods. This demonstrates the value of fusing fMRI modalities for more precise neurodegenerative disease diagnostics [61].

Jie et al. (2016) focused on diagnosing brain diseases, including MCI and ADHD, by estimating the hyper-connectivity network of brain functions from resting-state fMRI data. Their approach highlighted the importance of high-order relationships among brain regions, resulting in improved diagnostic accuracy [51].

Wang (2023) introduced a dynamic weighted hypergraph convolutional network (dwHGCN) for analysing the brain functional connectome, utilising both resting-state and task-based fMRI data. This method significantly advanced the classification tasks, attributing its success to the dynamic updating of hyperedge weights, which captured discriminative high-order relationships [95].

3.5 Limitations of Methods in fMRI Analysis

Despite the advancements highlighted by the aforementioned studies, the reviewed approaches all have limitations, which are summarised by Behrouz et al. (2023) [6]:

1. **Capturing Linear Correlation and Ignoring Temporal Order**

   Most spatiotemporal models use SWC and compute correlations between nodes within each window. This approach fails to capture temporal dependencies fully as shuffling the time within each window leads to the same outcome. In addition, Pearson correlation has been shown to not be able to comprehensively capture functional connectivity [70].

2. **Missing Higher-Order Dependencies**

   Empirical evidence suggests that interactions in neuroscience extend beyond dyadic relationships, involving complex group dynamics among nodes [82, 80, 106]. Current models often limit their scope to pairwise interactions, overlooking the potential for higher-order patterns in brain functional activity. This simplification neglects the network context at the level of individual regions, which necessitates the consideration of higher-order interactions for a more accurate representation of functional connectivity. Hypergraphs offer a promising alternative, as they generalise traditional graph models to enable edges connecting any number of vertices, thus better capturing the collective actions of groups of nodes.

3. **Neglecting the Brain's Hierarchical Structure**

   The human brain exhibits a hierarchical organisation, comprising functional systems

where groups of ROIs operate as integrated networks [41]. Conventional approaches without hypergraphs often disregard these functional communities, crucial for understanding the brain's functional organisation. ROIs within the same functional modules tend to exhibit similar behaviours and clustered representations, indicating that neglecting the hierarchical structure may lead to incomplete understanding of functional dependencies among ROIs [40, 83].

4. **Overlooking Network Dynamics**

   Brain dynamics vary over time. Existing studies frequently overlook the temporal variations in functional connectivity, which can dynamically change, even in resting states [12]. It is vital for models to capture these dynamics accurately, including variations across different timescales, to provide a comprehensive understanding of brain functionality [39, 62].

3.6 Overview of HADiB

HADiB an unsupervised hypergraph-based spatiotemporal approach to using r-fMRI for anomaly detection, addresses the limitations identified in section 3.5. By integrating a hierarchical feature extraction, a temporal hypergraph constructor and a tetra-stage message passing mechanism, HADiB addresses the critical challenges of capturing temporal dependencies, higher-order interactions, the brain's hierarchical structure, and dynamics over varying timescales.

To address limitations 1) and 2), HADiB introduces a *temporal hypergraph constructor* designed to *learn* the higher-order temporal dependencies between ROIs in the brain. This method surpasses the limitations of simple window-based correlation calculations, which fail to fully encapsulate the brain's complex connectivity patterns. The use of a hypergraph allows for the inclusion of edges connecting multiple nodes, thereby capturing the collective interactions among groups of ROIs [6].

To address limitations 3) and 4), HADiB employs a *tetra-stage message passing mechanism*. This method operates at different levels of granularity, encompassing ROIs, functional systems (FS), and overall brain encodings. By doing so, HADiB takes full advantage of the hierarchical organisation of the brain, where groups of ROIs operate as integrated networks to perform various tasks [6].

3.7 Project Aim and Scope

This project seeks to expand the frontier of neuroscience research by applying the principles of HADiB to a novel domain: the multiclass classification of cognitive tasks in t-fMRI. The objective is to explore the potential of hypergraph models, a concept not previously employed in this context, for the accurate identification of specific motor or social sub-tasks performed during t-fMRI scanning sessions. This initiative ventures into uncharted territory, as t-fMRI has been utilised identifying learning disabilities measured by the WRAT [99], but has yet to be applied for discerning the task undertaken by subjects during scans.

The rationale behind repurposing HADiB, originally designed for anomaly detection, for task classification lies in its adeptness at capturing complex temporal patterns across varying timescales. By adopting brain-network representation techniques akin to those used in HADiB, there is a promising opportunity to substantially improve the fidelity and granularity of task classification models. This approach not only aims to pinpoint which temporal brain activity patterns correspond to specific tasks but also to address the limitations observed in prior models while acknowledging those introduced by HADiB itself.

This project begins by exploring the Human Connectome Project 1200 task dataset, detailing the tasks involved and the protocol employed in acquiring the data. It then transitions into a comprehensive examination of HADiB, identifying its limitations and areas for improvement. The discussion progresses to the implementation phase, undertaken from scratch due to the absence of publicly available code, highlighting the challenges encountered during this process. Subsequent evaluation of the developed approach focuses on accuracy and comparative analysis with existing methodologies. Ethical considerations, particularly the implications of using artificial intelligence in interpreting brain scans, the consequences of erroneous diagnoses and social implications of BCIs are addressed. The conclusion synthesises the findings and suggests avenues for further research.

Data

The research employs a subset of the Human Connectome Project 1200 (HCP) [45] dataset, which comprises t-fMRI data captured using 3T scanners. For the purposes of this study, data from 339 subjects curated by the Neuromatch Academy [68] were chosen. This selection was informed by several pragmatic considerations:

- **Preprocessing Requirements:** Given that the HCP data undergoes only minimal preprocessing [33], considerable effort is required to prepare it for task-based analysis. The Neuromatch Academy's provision of preprocessed and structured task-data aligns with the project's focus on fMRI analysis, allowing for immediate commencement of the analysis phase without additional preprocessing stages.
- **Data Size:** The original HCP dataset's volume, extending to several terabytes, presents significant challenges in terms of data handling and computational resources. The selected subset is of a more manageable size due to a reduced number of subjects and preprocessing already applied.
- **Reproducibility:** The adoption of a well-defined and processed subset of data enhances reproducibility. The streamlined dataset from Neuromatch Academy offers a consistent framework for verification and replication of results across studies.

For the analysis, the 7 tasks are divided into 24 sub-tasks as shown in Figure 4.1.

In summary, the utilisation of this specific subset not only aligns with the logistical and methodological requirements of the research but also promotes the precision and reproducibility of the analytical outcomes.

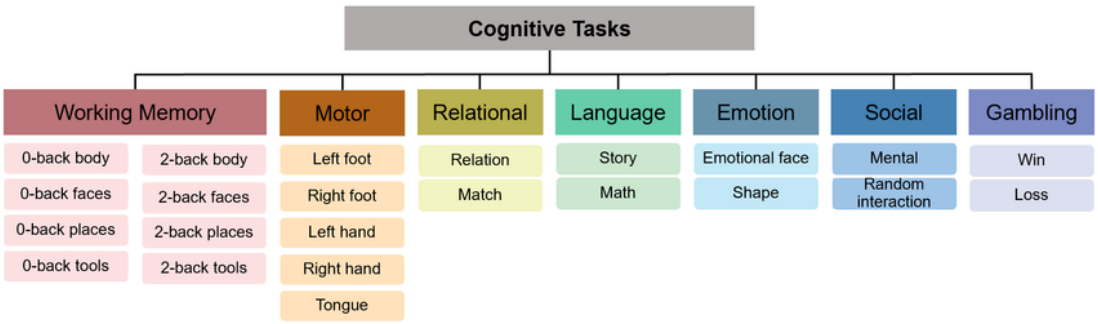


Figure 4.1: Description of the tasks and corresponding sub-tasks performed in the HCP. This includes 24 sub-tasks belonging to 7 higher-level tasks [77]. The "MOTOR_cue" sub-task, which involves signalling to a subject to get ready for a certain movement is not included in the image but will be included in the classification.

4.1 Tasks

The tasks within the dataset are designed to probe a range of cognitive functions, delineated as follows [36]:

1. **Working Memory (WM)**: This includes tests on category-specific representation and working memory combined into a single task paradigm, involving stimuli types such as pictures of places, tools, faces, and body parts.

2. **Gambling**: This task entails a card guessing game where participants predict the value of a mystery card to win or lose money.

3. **Motor**: Participants perform movements of fingers, toes, or tongue upon visual cues, to map motor areas. Each movement is preceded by a cue to prepare the participant for the movement at the right time.

4. **Language Processing**: Using scripts from Binder et al., participants are exposed to auditory stories and math problems, requiring them to solve comprehension and arithmetic tasks [10].

5. **Social Cognition (Theory of Mind)**: Participants view short videos of shapes interacting and judge the nature of these interactions.

6. **Relational Processing**: Participants discern relational aspects of objects differing in shape or texture.

7. **Emotion Processing**: Participants match faces or shapes based on emotional expressions.

	ROI_1	ROI_2
t_1	12	2
t_2	2	1
t_3	16	19
t_4	3	6

Table 4.1: A sample time series with two ROIs and 4 time steps. At time t_3, both ROIs show elevated activity at the same time.

4.2 Preprocessing

The data provided by the Neuromatch Academy is preprocessed according to the HCP minimal preprocessing pipeline. It comprises several stages, including spatial distortion removal, realignment, registration to structural data, bias field reduction, global mean intensity normalisation, brain masking, and surface-based processing [33].

The dataset is then parcellated using the Glasser Atlas [32], delineating the brain into 360 functionally relevant ROIs, 180 per hemisphere. The data are structured in a 2D matrix of shape Time $\times$ ROI, and the BOLD intensity values as the measured variable for each value, Table 4.1 illustrates a simple example time series.

4.3 Exploratory Analysis

An exploratory analysis was conducted to provide an initial characterisation of the dataset's structure, with a particular focus on the distribution and frequency of the different tasks. The histogram presented in Figure 4.2 showcases the variability in the number of examples for each task category within the dataset, showing that some tasks have been repeated multiple times for each subject. Figure 4.3 illustrates the variation in time series lengths across different sub-tasks, indicating a degree of inconsistency. Notably, sub-tasks under the categories 'LANGUAGE_story' and 'LANGUAGE_math' demonstrate a distinct disparity in time series lengths.

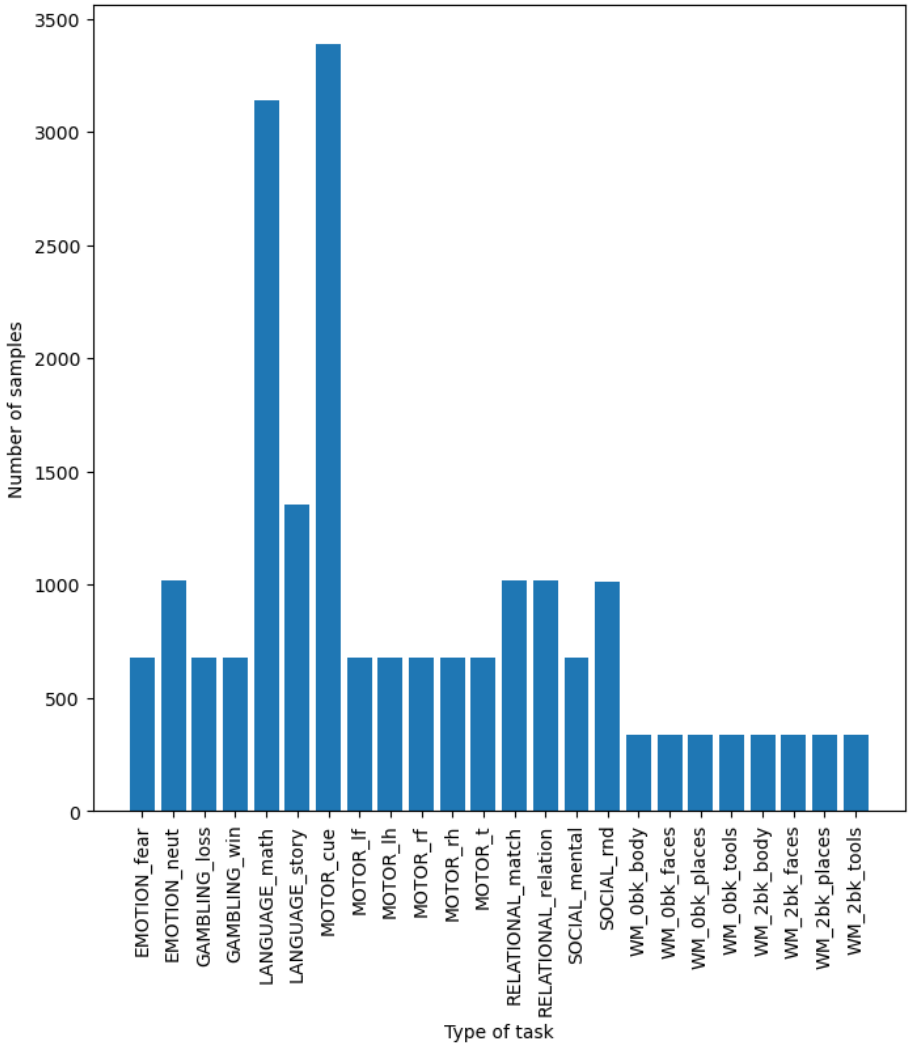


Figure 4.2: The frequency of occurrence for each cognitive task sub-category in the Neuromatch Academy subset of the HCP1200 task data [68].

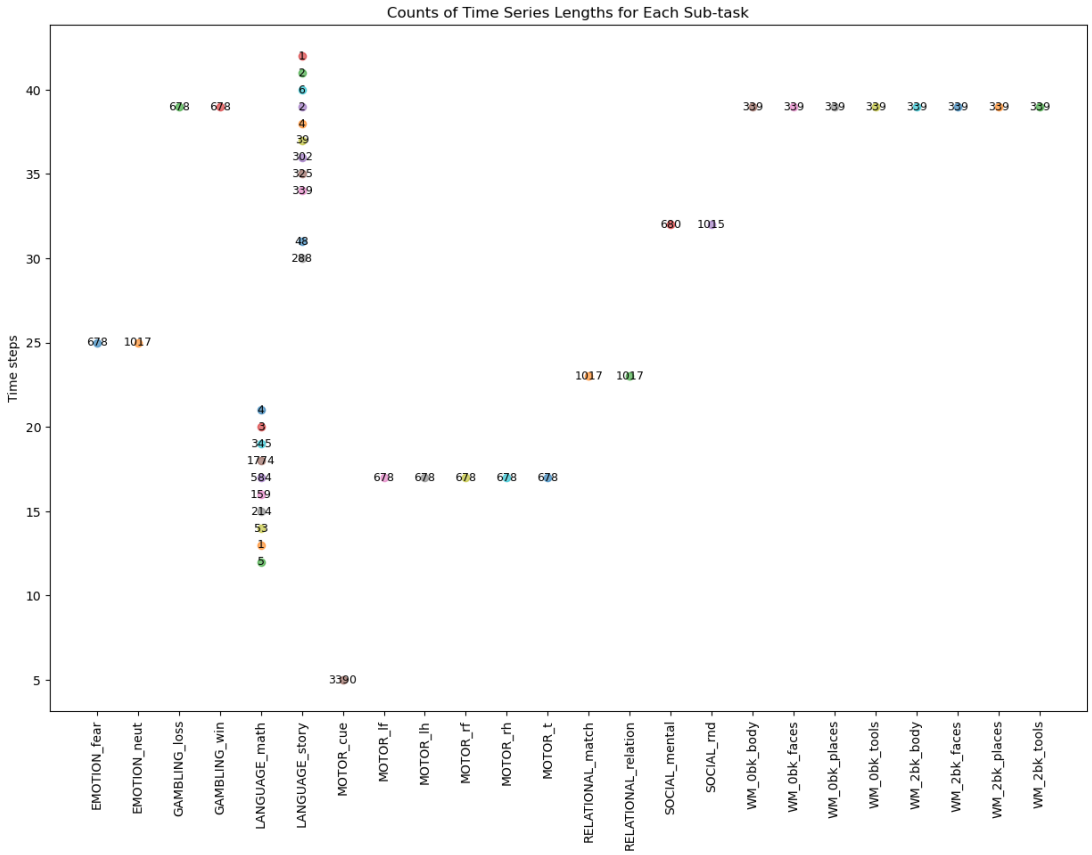


Figure 4.3: The length of time series for all sub-tasks in the Neuromatch Academy subset of the HCP1200 task data [68]. Each point is labelled with the number of occurrences of that length.

Design

To effectively adapt the methodology proposed by Behrouz et al. for t-fMRI classification, a thorough understanding of their end-to-end model, HADiB, is required. Figure 5.1 shows that on a high level, it is composed of three primary modules: (1) Feature Extraction, (2) Hypergraph Construction, and (3) Hypergraph Learning. In adapting this model, the original modules are preserved to the greatest extent feasible, while areas of ambiguity within the published work are clarified and components unsuitable for the classification task are replaced.

As depicted in Figure 5.2, the adapted pipeline for t-fMRI classification maintains the core architecture of HADiB. However, it integrates a data loader to prepare the data, skips FS-related message passing and replaces the unsupervised anomaly detection mechanism with a classification head.

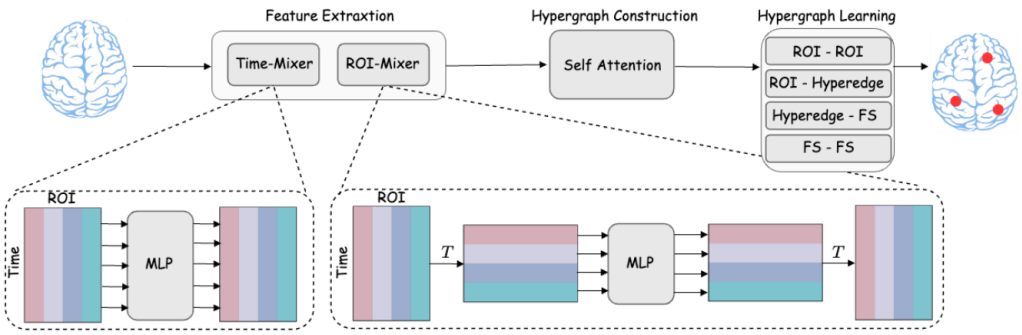


Figure 5.1: Schematic of HADiB, showing the end-to-end model comprising Feature Extraction using an MLP-Mixer, Hypergraph Construction using self-attention and Hypergraph Learning using message passing [6].

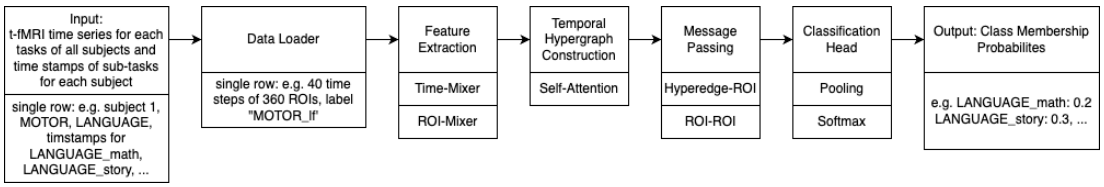


Figure 5.2: Adapted pipeline from HADiB for t-fMRI classification, incorporating data loading, feature extraction, hypergraph construction, message passing, and classification phases.

5.1 Data Loader

The data loader serves as the entry point to the pipeline, responsible for preparing and structuring the t-fMRI data to conform to the requirements of subsequent processing stages. The data, structured by the Neuromatch Academy [68], demands an implementation that is attuned to the specific format required by the model. The data loader accomplishes two principal functions:

1. **Sub-task Time Series Extraction:** The input to the data loader consists of timestamps delineating the commencement and conclusion of each sub-task, coupled with a time series for each group of tasks. The data loader extracts the time series pertinent to each sub-task, thereby segmenting the fMRI data into discrete instances that align with the temporal boundaries of cognitive or motor tasks being performed by subjects. For example, a time series for the language task is segmented multiple LANGUAGE_math and LANGUAGE_story time series, one for each repetition of each sub-task.

2. **Data Organisation:** Beyond extraction, the data loader of the Neuromatch Academy was extended to execute organisational tasks. It divides the entire dataset into training, validation, and test sets. This partitioning is pivotal for supervised learning, facilitating model training on a designated subset of the data (training set), hyperparameter tuning and model adjustment on another (validation set), and unbiased evaluation of the final model performance on unseen data (test set).

5.2 Feature Extraction

Feature extraction takes as input the fMRI time series of a single subject and uses an MLP-Mixer architecture [91] to encode cross-time and cross-ROI dependencies. It outputs sub-segments of the time series which have been 'mixed' for the subsequent construction of a temporal hypergraph.

5.2.1 Creating a Hierarchical Time Series

Temporal analysis of brain dynamics based on an fMRI time series requires accounting for dynamic changes over time [62]. In HADiB, this is achieved by constructing a hierarchical time series, which involves segmenting the series into varying window sizes corresponding to scales within a predefined set S.

The largest window size, $s_{\max} = \max(S)$, forms the basis for constructing network snapshots. For each non-negative integer t satisfying $0 \leq t \leq \left\lfloor \frac{T}{s_{\max}} \right\rfloor$, the growing segment $X_{1:(t+1)s_{\max}}$ is defined, with zero-padding applied where necessary to ensure the series length is a multiple of $s_{\max}$.

Each segment is further divided into sub-segments of size $s \in S$. Formally, this results in a hierarchical time series $\mathbb{X} = \{X^{(s)}\}_{s \in S}$, where each $X^{(s)}_{1:T_s} = (x^{(s)}_1, \ldots, x^{(s)}_{T_s})$, and the quantity $T_s = \left\lceil \frac{T}{s} \right\rceil$ indicates the number of windows into which the time series is partitioned.

Example

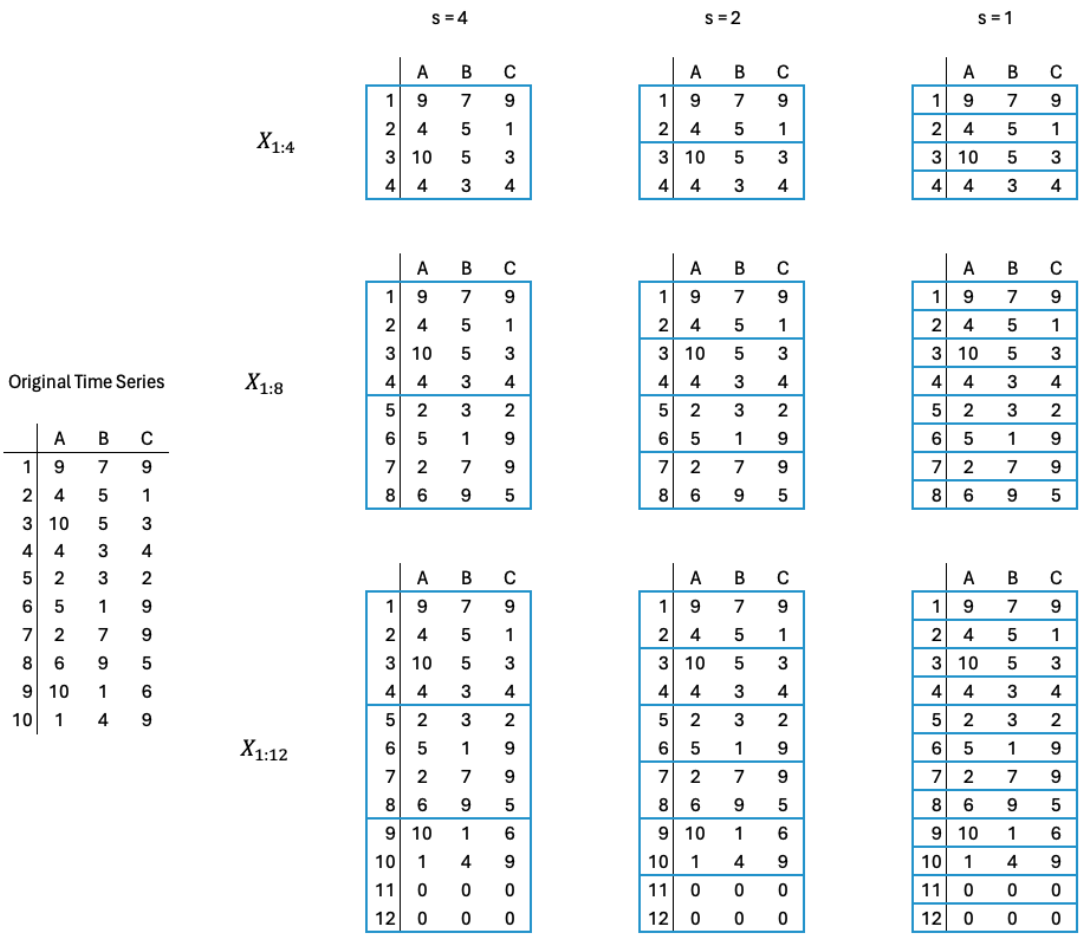
The process of creating a hierarchical time series is illustrated in Figure 5.3. It presents the decomposition of an original time series matrix into growing segments and subsequently into sub-matrices at multiple scales, delineating how each larger matrix is segmented into smaller matrices according to the scale $s \in S$. The window sizes demonstrated are $s = 4$, $s = 2$, and $s = 1$, where each descending scale provides a finer resolution of the time series data. Zero-padding is employed in cases where the time series does not evenly divide into the window size, ensuring the equal size of the segmentation across all scales.

The decomposition process simplifies the analysis of brain dynamics by providing a multi-scale representation that can reflect temporal evolution at different levels of granularity.

5.2.2 MLP-Mixer

The MLP-Mixer is an architecture designed to process visual data by leveraging multi-layer perceptrons rather than the conventional convolutional or attention-based mechanisms common in other models like CNNs or Vision Transformers [18]. This approach is rooted in the idea that MLPs, given sufficient scale and a carefully considered structure, can effectively encode spatial hierarchies and relationships within images.

As shown in Figure 5.4, the MLP-Mixer first partitions an image into a grid of fixed-size patches. These patches are then linearly embedded and arranged into a sequence, serving as the



Original Time Series

	A	B	C
1	9	7	9
2	4	5	1
3	10	5	3
4	4	3	4
5	2	3	2
6	5	1	9
7	2	7	9
8	6	9	5
9	10	1	6
10	1	4	9

$X_{1:4}$

	s = 4			s = 2			s = 1		
	A	B	C	A	B	C	A	B	C
1	9	7	9	9	7	9	9	7	9
2	4	5	1	4	5	1	4	5	1
3	10	5	3	10	5	3	10	5	3
4	4	3	4	4	3	4	4	3	4

$X_{1:8}$

	s = 4			s = 2			s = 1		
	A	B	C	A	B	C	A	B	C
1	9	7	9	9	7	9	9	7	9
2	4	5	1	4	5	1	4	5	1
3	10	5	3	10	5	3	10	5	3
4	4	3	4	4	3	4	4	3	4
5	2	3	2	2	3	2	2	3	2
6	5	1	9	5	1	9	5	1	9
7	2	7	9	2	7	9	2	7	9
8	6	9	5	6	9	5	6	9	5

$X_{1:12}$

	s = 4			s = 2			s = 1		
	A	B	C	A	B	C	A	B	C
1	9	7	9	9	7	9	9	7	9
2	4	5	1	4	5	1	4	5	1
3	10	5	3	10	5	3	10	5	3
4	4	3	4	4	3	4	4	3	4
5	2	3	2	2	3	2	2	3	2
6	5	1	9	5	1	9	5	1	9
7	2	7	9	2	7	9	2	7	9
8	6	9	5	6	9	5	6	9	5
9	10	1	6	10	1	6	10	1	6
10	1	4	9	1	4	9	1	4	9
11	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0

Figure 5.3: Hierarchical segmentation of a time series into scales $s = 4$, $s = 2$, and $s = 1$. The original time series is represented at the left, with subsequent columns illustrating the segmentation process at decreasing window sizes and rows representing the sub-segment growing by $s_{\max}$. Zero-padding is applied in $X_{1:12}$.

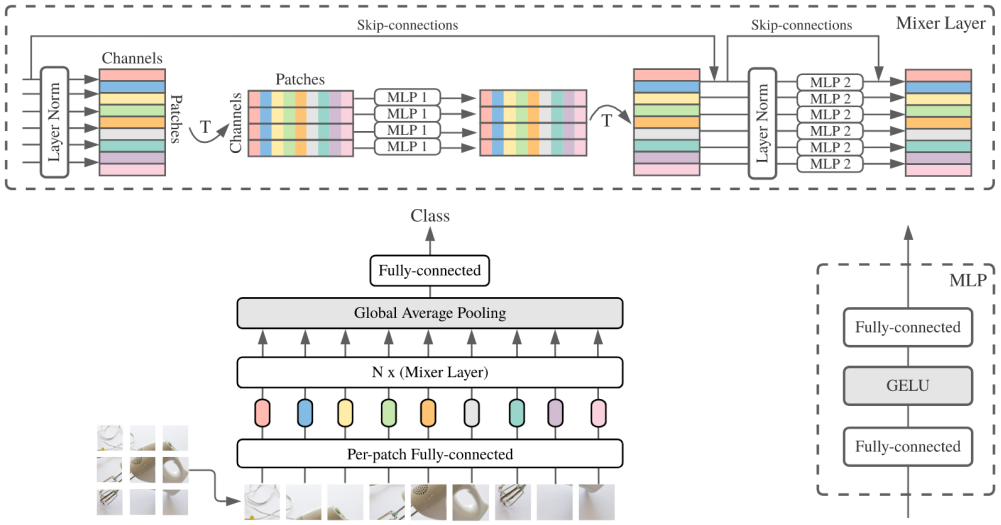


Figure 5.4: High-level schematic of MLP-Mixer for image classification [91].

input for the subsequent mixer layers. Each mixer layer comprises two core MLP components: one operates across the spatial dimension (token-mixing MLP) and the other across the channel dimension (channel-mixing MLP). This dual structure enables the MLP-Mixer to integrate information both locally (within patches) and globally (across patches) without relying on the more complex operations found in CNNs and Transformers.

The token-mixing MLP aggregates and recombines features from different spatial locations, allowing the model to learn spatial relationships across the image. Conversely, the channel-mixing MLP focuses on interactions between different feature channels within the same spatial location, enabling the model to refine its understanding of the local content within individual patches. The alternation between these two mixing mechanisms across layers allows the MLP-Mixer to build a rich, hierarchical representation of the image.

This architecture is distinguished by its simplicity and scalability, relying solely on MLPs and their inherent operations (matrix multiplication, non-linear activations, and normalisation) without convolutions or attention mechanisms. This simplicity facilitates the scaling of the model to large datasets and computational resources [91].

This method can be adapted for spatiotemporal data with ROIs being treated as channels and sub-segments of time as patches. This enables the capture of dependencies and interactions over time and across different ROIs, facilitating a more nuanced understanding of spatiotemporal dynamics.

5.2.3 HADiB Feature Extraction in Detail

High-Level Overview

At the core of HADiB's feature extraction lies the objective to encode the dynamic interplay between different brain regions and temporal segments, facilitating the construction of a temporal hypergraph. This is achieved through a hierarchical decomposition of the fMRI time series, enabling the capture of spatiotemporal dependencies at varying scales. The use of an MLP-Mixer architecture empowers HADiB to process these dependencies without relying on conventional convolutional or attention-based mechanisms. [1]

Step-by-Step Walkthrough

The feature extraction process in HADiB encapsulates a sequential application of time and ROI mixers to encode dependencies across both temporal and spatial dimensions of fMRI data. This section provides a detailed step-by-step walkthrough of the process, visualised in Figure 5.5.

1. **Hierarchical Time Series Extraction:** The fMRI time series data of a subject is decomposed into a hierarchical structure, as described on denoted as $\mathbb{X}$, facilitating the analysis at different temporal scales. This step forms the foundation for capturing dynamic brain activities with varying granularity.

2. **Application of Time and ROI Mixers:** For each temporal scale s in the set of predefined scales S, the time series of the segment t, $X^{(t)^{(s)}}$, undergoes processing by time and ROI mixers, aimed at capturing the dependencies across time points and ROIs.

   (a) **Time Mixer:** The time mixer operates on the hierarchical time series data, $X^{(t)^{(s)}}$ to encode temporal dependencies as per the equation:

   $$H_{TIME}^{(t)^{(s)}} = X^{(t)^{(s)}} + W_{TIME}^{(1)^{(s)}} \sigma \left(W_{TIME}^{(2)^{(s)}} LayerNorm \left(X^{(t)^{(s)}} \right) \right) \tag{5.1}$$

   In this operation, the input time series $X^{(t)^{(s)}}$ for each scale s is first subjected to *LayerNorm(.)*, a normalisation step that ensures consistent training dynamics and facilitates model convergence [3]. Following this, the normalised data passes

[1] To ensure clarity in the intricate feature extraction process of HADiB, the mathematical framework is examined as presented by Behrouz and Hashemi [6]. The equations are directly taken from their paper and expanded by myself to clarify the authors' probable rationale for their methodological choices.

through a sequence of operations: a dense layer (implemented as $W_{TIME}^{(1)^{(s)}}$), a non-linear activation function known as GELU [38], and another dense layer ($W_{TIME}^{(2)^{(s)}}$). This sequence is designed to encode the temporal dependencies within the data effectively. A critical aspect of this process is the use of shared weights in the dense layers for all sub-segments of identical length s, which not only reduces the model's complexity but also enhances its ability to learn generalised features across different segments. The inclusion of a skip connection, by adding the initial time series of segment t and the currently processed sub-segment of size s $X^{(t)^{(s)}}$ to the output of the second dense layer, ensures that the original information is not lost, thereby preserving the integrity of the input data. Through these steps, the MLP-Mixer operates across the rows of the matrix, to encode temporal relationships within the fMRI time series.

(b) **ROI Mixer:** Following the time mixer, the ROI mixer processes the output of the Time Mixer in the same manner to encode spatial dependencies between ROIs. The input is initially transposed to align ROIs with the matrix rows, ensuring that the MLP-Mixer's operations apply across the spatial dimension:

$$H_{ROI}^{(t)^{(s)}} = H_{TIME}^{(t)^{(s)}} + \left(W_{ROI}^{(1)^{(s)}} \sigma \left(W_{ROI}^{(2)^{(s)}} \, LayerNorm \left(\left(H_{TIME}^{(t)^{(s)}} \right)^{\intercal} \right) \right) \right)^{\intercal} \tag{5.2}$$

This process ensures the MLP-Mixer effectively integrates spatial information by operating across the transposed matrix. After processing, the data is transposed back to its original orientation, preserving the integrity of temporal and spatial encodings.

3. **Projection:** Post mixing, the output for each time scale, $H_{ROI}^{(t)^{(s)}}$, is projected to a uniform size encoding using an MLP: $H_{PROJ}^{(t)^{(s)}} = MLP(H_{ROI}^{(t)^{(s)}})$.

4. **Producing the final Encoding** The projections then are concatenated along the ROI dimension, aggregating the encoded features across different temporal resolutions.

$$H_F^{(t)} = \text{MLP-MIXER} \left(\bigoplus_{s \in S} H_{PROJ}^{(t)^{(s)}} \right) \tag{5.3}$$

The precise operation applied to the concatenated $H_{PROJ}^{(t)^{(s)}}$ is slightly ambiguous and will be discussed in 5.2.4.

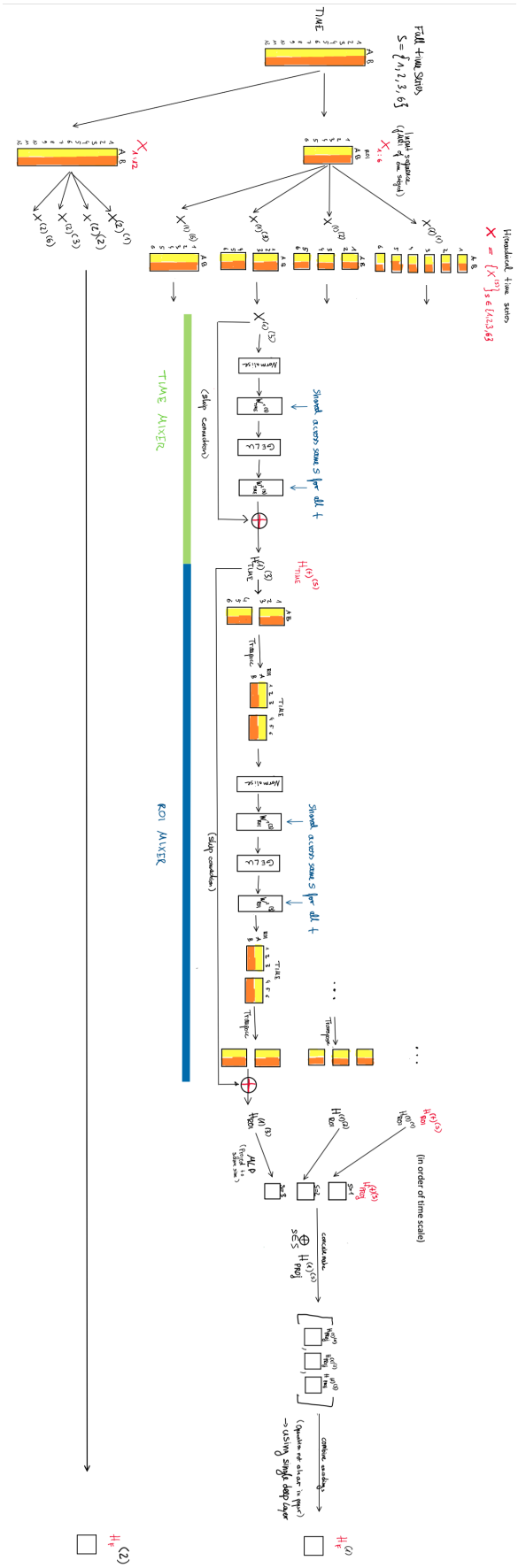


Figure 5.5: Spatiotemporal feature extraction in HADiB using the MLP-Mixer architecture

5.2.4 Open Questions and Ambiguities

1. **Applicability to short time series:** HADiB is designed for analysing r-fMRI data, which typically involves longer time series in the range of 1000 time steps. This is considerably longer than 5-40 time steps of the tasks in t-fMRI dataset. The difference in time series length raises questions about HADiB's effectiveness in capturing dynamic brain changes in t-fMRI data since the number of possible window sizes $s \in S$ for constructing the hierarchical time series is limited.

2. **Growing Sub-segments Approach:** A distinctive aspect of HADiB is its approach of employing growing segments ($X_{1:(t+1)s_{\max}}$) to capture the brain's temporal dynamics. While this method offers a novel perspective on r-fMRI data analysis in comparison to the conventional SWC, its effectiveness has not been demonstrated extensively in the literature.

3. **MLP-Mixer Operation Clarification:** The operation specified by *'MLP-MIXER'* in the HADiB extraction process, as indicated in equation 5.3, is ambiguous as to whether an actual MLP-Mixer is applied to the concatenated $H_{PROJ}^{(t)}$ or it just denotes a summarisation of the previous operations, which essentially already constitute an MLP-Mixer. To avoid the risk of overfitting that might arise from a more complex model, a straightforward approach was adopted, utilising a single dense layer for producing the final feature representation ($H_F^{(t)}$). This choice reflects a commitment to maintaining the model's simplicity and ease of interpretation.

5.3 Temporal Hypergraph Construction

5.3.1 Overview

The construction of a temporal hypergraph in HADiB involves the application of a modified self-attention mechanism to capture the dynamic brain connectivity across time series of growing length. Specifically, the t-th snapshot of the hypergraph is generated through the following way:

$$A^{(t)} = \text{SIGMOID}\left(\frac{Q^{(t)}K^{(t)}}{\sqrt{K^{(T)^\top}}}\right), \tag{5.4}$$

where $Q^{(t)} = H_F^{(t)}W_{\text{ATTN}}^{(1)}$ and $K^{(t)} = H_F^{(t)^\top}W_{\text{ATTN}}^{(2)}$ represent the query and key matrices

respectively, with $H_F^{(t)}$ denoting the output of the feature extraction module at time t, $W_{\text{ATTN}}^{(1)}$ and $W_{\text{ATTN}}^{(2)}$ being trainable weight matrices, and $\sqrt{K^{(T)^\top}}$ is the normalisation factor to prevent the dot products from growing too large. The ambiguous definition of the normalisation factor is discussed in 5.3.2.

The sigmoid function applied to the self-attention mechanism serves to constrain the values within a (0,1) range, thus inducing a binarising effect on the resultant attention scores. This effectively interprets high attention scores as logical '1's, signifying a strong functional connection and thus inclusion within the hyperedge, while low scores approach '0', indicating the absence of such a connection. Behrouz et al. interpret $A^{(t)}$ as the incidence matrix for the hypergraph, representing the brain's functional connectivity in a near-binary fashion for the t-th snapshot [6].

5.3.2 Interpreting the Hypergraph

The construction of the hypergraph raises fundamental questions regarding the interpretation of its nodes and hyperedges, particularly in the context of the attention mechanism and the underlying feature extraction process. Intuitively, nodes should correspond to ROIs and hyperedges to detected co-activation patterns.

Feature Extraction

The feature extraction process, particularly the use of growing windows, introduces a non-uniform analysis across the time series

The extracted features $H_F^{(t)}$ do not equate to a consistent hypergraph representation across identical window sizes, unlike the methodology observed in SWC. This results in a hypergraph where earlier time steps are analysed with finer granularity compared to later ones, which contradicts the approach of uniform granularity across the series. An alternative method, employing a sliding window approach to construct a hierarchical time series for each window and subsequently a hypergraph for each, could offer a more consistent representation of brain dynamics across time. This would result in a more intuitive temporal hypergraph where each snapshot reflects the connectivity within a specific window.

Attention Mechanism

The application of the attention mechanism in constructing the incidence matrix of a hypergraph introduces several considerations. Firstly, employing an attention mechanism for hyper-

graph construction is relatively unusual, highlighting the need for further exploration within the literature. Second, the normalisation term is essential for maintaining the scale of the `softmax` operation, typically being the square root of the dimensionality of the key vector ($\sqrt{d_k}$) as noted in related works. However, the paper's proposed normalisation factor, $\sqrt{K^{(T)^\top}}$, lacks a clear definition and seems to rely on the variable T, which denotes the length of the entire time series, suggesting a dynamic normalisation factor, which itself is not too common. This might be a typographical error, implying that the well-defined $K(t)$ was the intended term. Empirical evaluations indicate that the choice between a dynamic normalisation factor and the conventional approach does not significantly impact performance, prompting the preference for the latter to avoid a speculative implementation and ensure simplicity.

Conclusion

Given the aforementioned considerations, the interpretation of the hypergraph within HADiB presents a challenge. The nodes, ideally corresponding to ROIs, and the hyperedges, representing complex connectivity patterns, necessitate a more nuanced understanding to accurately reflect the underlying brain dynamics. In giving the authors the benefit of the doubt, and in the absence of alternative interpretative frameworks, it is assumed that the proposed methodology is effective. This assumption is further supported by reference to the ablation study conducted by Behrouz et al., which demonstrates the effectiveness of the attention mechanism [6]. However, it is acknowledged that further research is necessary to deepen the understanding of this approach.

5.4 Hypergraph Learning

5.4.1 Overview

After the temporal hypergraph has been constructed using the MLP-Mixer and self-attention mechanism, the graph structure is utilised to learn higher-order interactions between ROIs. There are two ways the graph structure can be used: either by computing hand-crafted relevant topological features of a hypergraph such as length and width of walks or the size of edge intersections [52] or by letting the model learn relevant features on its own using message passing. Since HADiB is designed as an end-to-end model that minimises the need for manual adjustment, the authors propose a tetra-stage message passing mechanism to learn the dependencies between ROIs at different spatiotemporal scales. The following sections describe the

message passing proposed by Behrouz et al. [6], which is subsequently critically analysed.

5.4.2 ROI-Hyperedge Phase

The encoding of a hyperedge within the hypergraph message passing framework is essential for capturing the global interactions among ROIs. The strength of the connection that each ROI has to the hyperedge varies, necessitating a weighted approach to account for the varying importance of each ROI's message. This is achieved through an attention mechanism, as denoted in equitation 5.4, that learns to emphasise the most relevant information:

$$\zeta_e^{(t)(\ell)} = \sum_{u \in e} A_{u,e}^{(t)} \hat{h}_u^{(t)(\ell)}, \tag{5.5}$$

In the above equation, $\zeta_e^{(t)(\ell)}$ signifies the encoding of hyperedge e at time t and layer ℓ, while $A_{u,e}^{(t)}$ denotes the attention coefficient that scales the influence of node u's encoding within the hyperedge. The attention coefficients $A_{u,e}^{(t)}$ are obtained using a sigmoid function wrapping around the attention mechanism. In the context of this model, this mostly results in values that are close to either 0 or 1, effectively creating a binary-like effect. Multiplying these with the node encodings, consequently, the encoding of a hyperedge primarily consists of the encodings of its constituent nodes, in accordance with the binary nature of the hypergraph's incidence matrix.

5.4.3 ROI-ROI Phase

To model the local dependencies between ROIs within a hypergraph structure, the ROI-ROI phase involves an iterative process of message passing. Given a snapshot at time t, for two ROIs u and v within the same hyperedge $e \in E^{(t)}$, the message passing and encoding update at the ℓ-th neural network layer is formalised as follows:

$$m_{u \to v}^{(\ell)} = \mathbf{W}_{ROI}^{(\ell)} \cdot \text{CONCAT}\left(\hat{h}_u^{(t)(\ell-1)}, \hat{h}_v^{(t)(\ell-1)}, \zeta_e^{(t)(\ell-1)}\right), \tag{5.6}$$

$$\hat{h}_u^{(t)(\ell)} = \sum_{v \in \mathcal{N}^{(t)}(u)} m_{v \to u}^{(\ell)} + \hat{h}_u^{(t)(\ell-1)}, \tag{5.7}$$

where $\mathbf{W}_{ROI}^{(\ell)}$ denotes a learnable matrix specific to the ℓ-th layer, $\zeta_e^{(t)(\ell-1)}$ represents the encoding strength of the hyperedge e from the previous layer, and $\mathcal{N}^{(t)}(u)$ is the neighbourhood of u, i.e. the set of nodes connected to it by at least one hyperedge in the t-th snapshot. The initial state $\hat{h}_u^{(t)(0)}$ is set to be $h_F^{(t)}(u)$, corresponding to the row of the matrix $H_F^{(t)}$ associated

with ROI u. $H_F^{(t)}$ is the output of the feature extraction module, as described in equation 5.3. Looking ahead to the implementation section, there was an issue with generating the hyperedge encodings and therefore, each message will be computed without them:

$$m_{u\to v}^{(\ell)} = \mathbf{W}_{ROI}^{(\ell)} \cdot \text{CONCAT}\left(\hat{h}_u^{(t)(\ell-1)}, \hat{h}_v^{(t)(\ell-1)}\right), \tag{5.8}$$

5.4.4 Hyperedge-FS Phase

In the Hyperedge-FS phase, each FS is an assemblage of ROIs. What exactly constitutes an FS is not clarified by Behrouz et al. This issue is elaborated in section 5.4.6. It is worth noting that due to this ambiguity, the message passing related to FS could not be implemented and is elaborated in order to be able to discuss the implications of their exclusion.

A naive approach to encode an FS would be to simply aggregate the encodings of its constituent ROIs [6]. However, this method would fail to capture the dependencies between the ROIs within each FS. To address this, a pooling function is introduced, denoted as POOL$(\cdot)$, which aggregates the encodings of hyperedges associated with each FS. This ensures that the resulting FS encoding encapsulates not only the information from different ROIs but also the dependencies between them and their roles in the system's dynamics.

Let F denote the complete set of functional systems, and for each $f \in F$, let E_f represent the set of hyperedges contained within FS f. The matrix $Z_f^{(t)(\ell)}$ is then defined as the aggregation of the encodings of hyperedges in E_f at time t and layer ℓ:

$$Z_f^{(t)(\ell)} = \bigoplus_{e\in E_f} \zeta_e^{(t)(\ell)}, \tag{5.9}$$

where $\bigoplus$ signifies the aggregation operation. The FS encoding is computed by processing $Z_f^{(t)(\ell)}$ through a Gated Recurrent Unit (GRU) [13] to capture temporal dynamics:

$$\hat{Z}_f^{(t)(\ell)} = \text{GRU}\left(Z_f^{(t)(\ell)}, Z_f^{(t-1)(\ell)}\right), \tag{5.10}$$

The encoding is further refined to incorporate cross-feature dependencies:

$$\tilde{Z}_f^{(t)(\ell)} = \hat{Z}_f^{(t)(\ell)} + \sigma\left(\text{Softmax}\left(\text{LayerNorm}\left(\hat{Z}_f^{(t)(\ell)}\right)^\top\right)^\top\right), \tag{5.11}$$

Finally, the pooled FS encoding, $\psi_f^{(t)(\ell)}$, is obtained by averaging the refined encoding along the hyperedge dimension, and applying a non-parametric feature binding via the Softmax

function to reduce the number of parameters:

$$\psi_f^{(t)(\ell)} = \text{MEAN}\left(\tilde{Z}_f^{(t)(\ell)} + \sigma\left(\text{LayerNorm}\left(\tilde{Z}_f^{(t)(\ell)}\right) W_P^{(1)}\right) W_P^{(2)}\right), \tag{5.12}$$

where $\text{MEAN}(\cdot)$ computes the mean along the hyperedge dimension, $W_P^{(1)}$ and $W_P^{(2)}$ are trainable weight matrices, and σ represents a non-linear activation function.

This pooling method emulates the feature extraction architecture to assimilate both cross-hyperedge and cross-feature dependencies. The inclusion of a GRU cell in the computation sequence allows for the temporal evolution of the FS encodings, thereby capturing the dynamic characteristics of the functional systems.

5.4.5 FS-FS Phase

In the final stage of message passing, structural learning is undertaken at the level of FSs. This phase aims at capturing the dependencies between different functional systems, thereby facilitating the derivation of brain-level encodings that reflect higher-order interactions of brain activity. It operates under the assumption that all functional systems are interconnected and employs the same self-attention mechanism as in 5.4 to ascertain the strength of these interconnections.

Given $\psi_f^{(t)(\ell)}$ for each $f \in F$, let $\Psi^{(t)(\ell)} \in \mathbb{R}^{|F| \times d}$ be the matrix comprising rows of $\psi_f^{(t)(\ell)}$ for each functional system. The self-attention mechanism is then defined as:

$$A_F^{(t)} = \text{SIGMOID}\left(\frac{Q_F^{(t)} {K_F^{(t)}}^\top}{\sqrt{{K_F^{(T)}}^\top}}\right), \tag{5.13}$$

where $Q_F^{(t)} = \Psi^{(t)(\ell)} W_F^{(1)}$, $K_F^{(t)} = \Psi^{(t)(\ell)} W_F^{(2)}$, and $W_F^{(1)}$ and $W_F^{(2)}$ are learnable matrices.

The brain-level encoding, denoted by $\Upsilon^{(t)(\ell)}$, is obtained through the weighted aggregation of the encodings of the functional systems:

$$\Upsilon^{(t)(\ell)} = \sum_{f \in F} A_{f,f}^{(t)} \psi_f^{(t)(\ell)}. \tag{5.14}$$

This mechanism allows for the dynamic capturing of the structural and functional relationships between different FS, thus yielding a comprehensive brain-level representation that encapsulates the overarching patterns of neural interactions.

5.4.6 Investigation into the Definition of Functional Systems

The delineation of what precisely constitutes a functional system remains ambiguous in the work by Behrouz et al. This presents a challenge for implementing FS-related message passing within the HADiB framework. There are two options for how to decide what nodes are part of what FS:

1. **Pre-defined Atlas Mapping:** The Glasser Atlas offers a pre-defined mapping of its 360 ROIs to a set of 22 segments, as shown in 5.6. For instance, it demarcates distinct regions such as the visual cortex (1-5) and the sensorimotor area (6-9), amongst others. Each of these 22 segments could be considered to be one FS. However, there are two problems with this interpretation:

    - **Coarseness:** The coarseness of this parcellation could be prohibitive for task classification. For example, the different motor tasks such as the movement of the left or right hand could be hard to distinguish since the 26 ROIs associated with motor function get mapped to only five FSs [31].
    - **Interpretation of ROIs:** After the process of feature extraction, hypergraph construction and the first two rounds of message passing, it becomes problematic to ascertain whether ROIs can still be attributed to specific FSs as defined by the atlas, similarly to the issue elaborated in section 5.3.2.

2. **Connected Components of the Hypergraph:** An alternative definition of an FS could be the disconnected components of a hypergraph. This approach posits that the hypergraph may contain multiple disconnected components, suggesting the presence of multiple functional systems. However, this is contradicted by my implementation of the model proposed by Behrouz et al., which learns a fully connected hypergraph, as depicted in Figure 5.7, implying the existence of a singular, encompassing FS.

Given these considerations, both the Hyperedge-FS and the FS-FS phase, which depend on a concrete definition of what constitutes an FS, could not be effectively implemented. The implications of this are discussed in 5.6.2

5.5 Classification Head

The adaptation of HADiB from an unsupervised model for anomaly detection to a supervised model for t-fMRI classification necessitates the development of a new component, the classi-

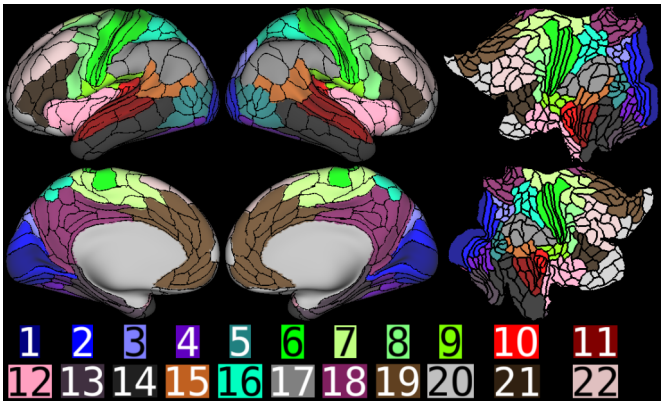


Figure 5.6: The Glasser Atlas mapping of ROIs to FSs [32].

fication head. Unlike the feature extraction, temporal hypergraph construction, and message passing mechanisms, which largely follow HADiB's original framework with only minor design decisions in cases of ambiguity, the classification head is a novel addition specifically designed for this application. It serves as the terminal layer where the encoded spatiotemporal patterns are mapped to specific cognitive tasks. This section discusses two possible approaches and explains the decision to use hypergraph encodings directly instead of first extracting topological measures as inputs to the classification head.

5.5.1 Considered Options for Classification

Two principal strategies were evaluated for their potential in capturing the higher-order spatiotemporal patterns representative of brain activities during cognitive tasks: (1) the extraction and utilisation of topological measures, and (2) the direct feeding of hypergraph encodings into the classification head.

Topological Measures

Topological measures—quantitative metrics that capture the global and local structural properties of hypergraphs—present an common approach to extract features from a graph. These measures, like for example centrality, node degree distribution, and temporal motif counting [59], offer insights into the brain's topology and its evolution during task performance. However, a significant limitation emerges in the form of differentiability. The end-to-end learning model necessitates a framework where gradients can be computed and backpropagated, a requirement that many topological measures, being inherently discrete and non-differentiable, do not satisfy.

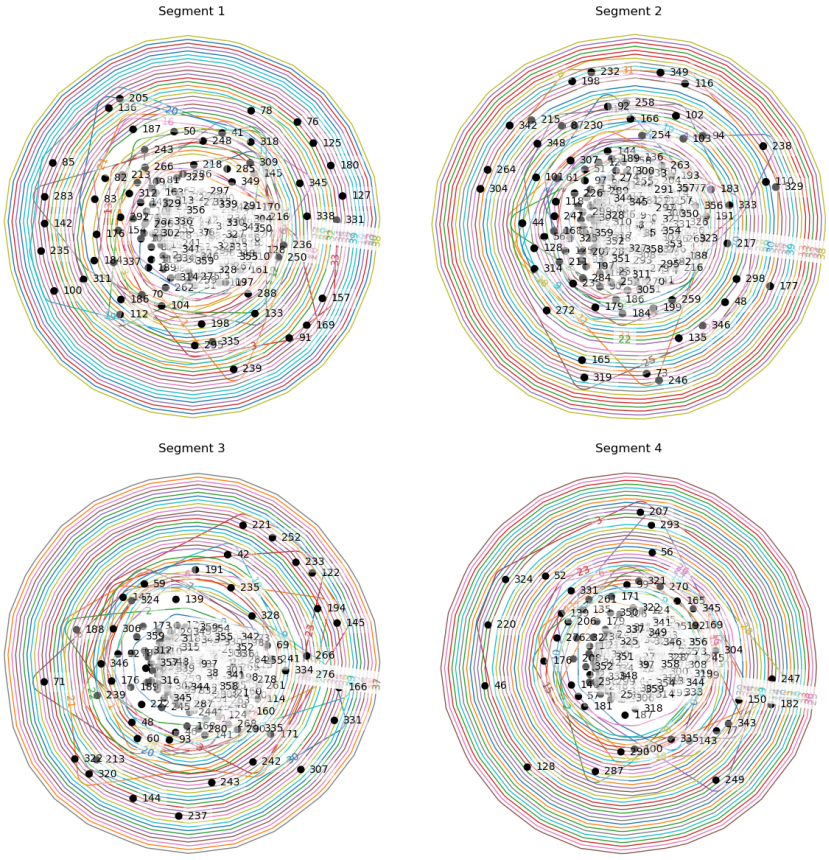


Figure 5.7: The hypergraphs generated by the temporal hypergraph construction layer, contain 360 nodes and 40 hyperedges each. The model was evaluated using window sizes of 5 and 10, with a total sequence length (`T`) being 40. The tasks classified were GAMBLING and SOCIAL. Further default parameters can be found in Appendix A.1.

Direct use of Hypergraph Encodings

Given the constraints associated with topological measures, the strategy to directly input hypergraph encodings into the classification head was adopted. This approach uses the node encodings updated based on the structure of the hypergraph—without necessitating a preliminary reduction to topological metrics. The classification head flattens the result of the message passing and applies a dense layer followed by a `softmax` activation function [11] to output class membership probabilities.

5.5.2 Decision Rationale

The decision to integrate hypergraph encodings directly into the classification head is primarily influenced by the need for simplicity in the model's design, as well as the practicality of constructing an end-to-end differentiable framework. This approach circumvents the complexities

of deriving differentiable topological features, streamlining the development process and emphasising raw data utilisation for task classification. While the direct use of encodings facilitates model construction and learning, it introduces considerations regarding the interpretability of the results. Incorporating topological measures into the classification could enhance the model's explainability, potentially verifying whether the constructed hypergraphs are consistent with established neuroscientific evidence.

5.6 Summary

In the adaptation of the HADiB architecture for task-based fMRI classification, the design adheres closely to the original framework while navigating ambiguities necessitated certain design decisions. The adapted model utilises a hypergraph-based approach, leveraging HADiB's foundational components: feature extraction, hypergraph construction, and hypergraph learning, complemented by an integrated classification head for supervised learning tasks. The following summarises the purpose of the different components and highlights where design decisions needed to be made due to ambiguities in the paper by Behrouz et al.

5.6.1 Design Adaptations and Specifications

The model retains the essence of HADiB, aiming to encode multivariate time series data at varied temporal scales to capture both higher-level cross-time and cross-ROI dependencies.

1. **The feature extraction** process utilises an MLP-Mixer architecture, ensuring comprehensive encoding by blending temporal and spatial features. Ambiguities encountered within the original HADiB design, particularly concerning the operation for generating the final encoding HF (Equation 5.3), necessitated a design decision to adopt a single dense layer to consolidate the extracted features into a coherent format suitable for subsequent analysis.

2. **The construction of the temporal hypergraph**, central to modelling the complex, non-pairwise interactions among brain regions, similarly follows HADiB's methodology but incorporates a design choice regarding the normalisation term of the attention mechanism defined in Equation 5.4. Opting for the conventional term, the square root of the dimensionality of the key vector, the model aims to capture higher-order spatiotemporal dependencies crucial for understanding brain connectivity dynamics.

3. **The hypergraph learning component** incorporates a tetra-stage message-passing mechanism designed to discern dependencies across various levels of granularity, spanning from individual ROIs to broader FSs and the entire brain structure. However, ambiguities in defining what constitutes an FS and implementation challenges necessitated a focus on only ROI-ROI message passing. The lack of FS-based message passing possibly does not lead to a disadvantage in the context of t-fMRI, where the analysis is more concerned with activities at the level of ROIs, such as the differentiation between left and right-hand movements in motor tasks, rather than working with the less granular FSs. Consequently, the Hyperedge-FS and FS-FS message passing layers for generating brain-level encodings can be skipped. However, the exclusion of Hyperedge-FS message passing might still be relevant and presents an opportunity for possibly enhancing the model.

4. **The classification head** determines the class membership of a given time series. This stage marks the transformation of the HADiB framework from its original unsupervised anomaly detection application to a supervised t-fMRI classification model.

5.6.2 Open Questions

The adaptation of HADiB for task-based fMRI classification, while comprehensive, raises several open questions important for further refinement and understanding:

- The explicit consideration of temporal dynamics might be insufficient, as currently only the MLP-Mixer does this. The removal of the Hyperedge-FS message-passing step might be hindering as it is the only other component that explicitly incorporates the temporal aspect using a GRU [13].

- The explicit consideration of ”higher-order spatiotemporal patterns” presents an open question as well. Currently treated in an implicit manner using a hypergraph, a more explicit extraction and analysis of these patterns could enhance the model’s explainability and contribute to a deeper understanding of brain function.

Implementation

6.1 Overview

This section provides the implementation for the model described in the design section. Given the unavailability of the original code, the development was undertaken from the ground up. The discussion includes an introduction to TensorFlow, along with a comprehensive analysis of the implementation process for each component. This encompasses challenges faced, solutions adopted and persisting limitations. Selected code snippets are included for clarity, and further relevant bits of source code in C are referred to. An iterative development methodology was employed, allowing for the evaluation and adjustment of the model at various stages. Instances where the model's performance influenced development choices are specifically addressed. To ensure reproducibility, random parameters were set to constant values which are detailed in Appendix Section A.1.

6.2 TensorFlow

TensorFlow (TF) is a comprehensive, open-source framework for machine learning developed by Google. At its core, TF operates on the concept of computational graphs, which are directed graphs where nodes represent operations, and edges denote the flow of data, or tensors, between these operations. This structure facilitates the representation of ML models in a manner that is highly efficient for computation.

A symbolic tensor, in the context of TF, is essentially a multidimensional array that may have a defined shape and data type but does not hold values itself. Instead, it serves as a description of the data that flows through the graph. Operations within the graph can have any number of inputs and outputs, ranging from simple mathematical equations to complex variable

manipulations and control flow directives. Trainable parameters of models are encapsulated as TF variables (`tf.Variable`), characterised by their name, type, shape, and initialisation parameters [104].

The use of computational graphs offers several key advantages [104]:

- **Dependency-driven scheduling**: The execution order is determined by the data dependencies within the graph, allowing for operations that do not depend on each other to be processed in parallel.
- **Graph optimisations**: TF can optimise computational graphs through methods such as eliminating common subgraphs, thereby enhancing performance.
- **Automatic differentiation**: By describing computations as graphs, TF simplifies the calculation of gradients through reverse mode auto-differentiation. This approach computes the gradient of any tensor with respect to any other in a single pass, crucial for training ML models [104].

Given these capabilities, TF was selected as the framework of choice for this project, particularly due to its official implementation of TSMixer, an MLP-Mixer tailored for time-series data analysis [20], which served as starting point for the feature extraction. The decision was further reinforced by TF's flexibility, especially when utilising the Keras Functional API, which supports complex model architectures with shared layers and integration with libraries such as TensorFlow Graph Neural Networks (tf-gnn) for message passing.

6.3 Data Loader

The Data Loader facilitates preprocessing and preparing the HCP t-fMRI data in form of a TF dataset for model training, consisting of extracted time series of single tasks and a corresponding label. This section elaborates on the implementation challenges, features, and methodologies of the Data Loader, using efficient and structured access to a subset of the HCP dataset provided by the Neuromatch Academy [68].

6.3.1 Challenges Encountered

The HCP dataset presents a significant challenge due to its complexity and the voluminous nature of neuroimaging files. Initially, the extraction of time series data from the myriad of available neuroimaging files was not straightforward. Moreover, the substantial size of the

dataset, approximately 7GB per subject, imposed limitations on storage capacity for personal computing resources. To address these challenges, the Data Loader utilises a pre-processed subset of the HCP dataset curated by the Neuromatch Academy. This subset significantly reduces the file size for all subjects' time series data to around 3.5 GB, thereby mitigating storage capacity concerns. The Data Loader leverages specific code snippets provided by Neuromatch Academy [68] for extracting time series data for each task, streamlining the data preparation process. This approach focuses the analysis on pattern recognition within task fMRI time series data, bypassing the intricacies of preprocessing neuroimaging data.

6.3.2 Features and Methodology

Input Parameters: The Data Loader is designed to accommodate various input parameters, including the selection of experiments (e.g., "GAMBLING" and "SOCIAL"), which run direction of the scan should be used ('LR' or 'RL'), subject IDs, the batch size determining how many instances should be processed by the model in parallel to facilitate training speed, and the maximum window size (s_{max}) used for padding. The inclusion of a random state parameter facilitates data shuffling during the train-test-validation split process. The Data Loader utilises the constant TR (0.72 seconds) and timestamps for sub-tasks to accurately segment the time series data corresponding. Labels are encoded using the sklearn LabelEncoder, converting task descriptions into numerical representations.

Train-Validation-Test Split: The data is divided into training, validation, and testing sets using sklearn's `train_test_split` function, applied sequentially to ensure proportional representation of data across splits and optionally enabling a fixed random state for reproducibility.

Padding Strategy: The `pad_to_smax` function computes and applies necessary padding to each time series, ensuring that the length of each sequence is a multiple of s_{max}. This step is required as the hierarchical time series construction in the feature extraction expects time series to be multiples of s_{max}. For instance, consider the scenario where s_{max} is set to 10. In the context of different task-related experiments within the HCP dataset, the length of time series data varies, as illustrated in Figure 4.3. Taking the 'MOTOR' experiment as an example, which may have a sequence length of 17, the `pad_to_smax` function will extend this sequence to 20, aligning it with the nearest multiple of s_{max}. Similarly, for the 'EMOTION' experiment with a sequence length of 25, the function will pad the sequence to 30. This padding is achieved by appending zeros to the end of the sequence, ensuring that the integrity of the original data

is maintained while adhering to the uniform sequence length requirement.

Dataset Construction: The `make_dataset` function transforms split and padded time series data and corresponding encoded labels into a format compatible with TF. By leveraging `tf.data.Dataset.padded_batch`, the function ensures that data batches are dynamically padded to match the longest sequence within each batch, maintaining consistency in input shape and size. This approach allows for the inclusion of variable-length time series within the same batch, further optimised by specifying padding values and shapes. Considering a batch that contains time series data from the 'MOTOR' and 'EMOTION' experiments, initially padded to lengths of 20 and 30 respectively by pad_to_smax, to conform with the s_{max} parameter. Within the context of this batch, the `make_dataset` function will further adjust the padding for the 'MOTOR' experiment sequences from 20 to 30, ensuring that all time series in a batch are of the same length.

In summary, the Data Loader abstracts away the complexities of data preprocessing, providing a streamlined pipeline for loading and partitioning the HCP task fMRI dataset into structured formats suitable for the model.

6.4 ML Model Overview

The construction of the model is contingent upon the specified window sizes. As the granularity of the windows increases, the number of learnable parameters within the model grows. The `build_model` function orchestrates the construction of the TensorFlow computational graph. It accepts several parameters: `input_shape` to define the input dimensions, a list of window sizes (`S`) for feature extraction, `ff_dim` specifying the output size of certain neural network layers and `n_classes` for the number of target classes in the classification task. The process begins with a masking layer to handle zero-padding, followed by feature extraction to generate hierarchical time-series data. Subsequently, it constructs a temporal hypergraph for each segment of the time series, transforming these into a format suitable for the application of a message passing layer. Finally, a classification head processes the features to produce predictions for subtasks. Dropout layers are placed at strategic locations in the model in order to prevent overfitting [87]. The architecture reflects a modular design, with distinct functions or custom layers for each component, promoting clarity and flexibility in model construction. Each of the components' implementation and challenges will be discussed subsequently.

6.5 Feature Extraction

This section elaborates on the implementation of the feature extraction component. It uses a hierarchical time series encoder designed to process inputs of the form [Batch, Time, ROI]. The encoder creates a hierarchical time series based on varying window sizes, performs time and ROI mixing, and returns a tensor for each of the segments growing by steps of size s_{max}. The implementation faced the challenge of implementing shared layers for the same window sizes as specified in equation 5.1.

6.5.1 Challenge: Shared Layers

The initial implementation of the encoder without shared layers resulted in a substantial model size, approximately 2.1 million parameters for window sizes $S = 10, 20$ and a time series length of $T = 40$, leading to prolonged training durations. To address this, the architecture was revised to include shared layers as per the HADiB specifications, which reduced the model size to around 1.2 million parameters. This adjustment not only streamlined the model but also enhanced its training efficiency without compromising the learning capacity.

6.5.2 Implementation

The hierarchical time series encoder employs a structured approach to segmenting the input time series into growing segments and further dividing these into smaller sub-segments, as depicted in Figure 5.3. This process leverages shared learnable matrices for feature extraction across different window sizes, enabling a robust representation of the time series data.

Segmentation Strategy

Initially, the input time series is divided into segments growing at steps of $s_{\max}$. These segments are then further divided into smaller sub-segments, with sizes corresponding to all provided window sizes.

Pseudocode Overview

```
Define hierarchical_time_series_encoder(input, S, ff_dim):
    s_max = max(S)
    Initialise shared res_blocks for each window size s,
    using create_shared_layers(S, ff_dim, input_shape)
```

```
    For each growing segment from 1:T, step size s_max
        compute end index
        For each window size s in S:
            Divide the segment into sub-segments of size s, up to the end index
            For each sub-segment:
                Apply residual block pertinent to window size s for feature extraction
            Project sub-segment features to a uniform size
        Concatenate projected features from all sub-segments
        Apply dense layer to combine features, producing H_F_t
    Stack H_F_t for all segments to form the output tensor H_F_ts_stacked

Return H_F_ts_stacked
```

This pseudocode illustrates the approach to dissecting the input time series into segments and sub-segments, applying shared dense layers for feature extraction, and ultimately synthesising a comprehensive representation. The use of a residual block (res_block) for each sub-segment, with shared layers for each window size, ensures that the model captures both temporal and spatial dynamics effectively.

By projecting the features of sub-segments to a uniform size and concatenating these projections, the encoder preserves the integrity of the temporal structure while enhancing the feature space. The final encoding, $H_F^{(t)}$, for each segment t is obtained by combining the mixed features through a dense layer, as outlined in Section 3. The encodings are then stacked to satisfy TensorFlow's requirements for tensor outputs.

6.5.3 Time and ROI Mixing

Within the hierarchical time series encoder, the residual block, referred to as `res_block`, integrates time and ROI mixing. This block utilises shared dense layers to learn spatiotemporal patterns in the inputs of the same window size s.

Dense Layers

A dense layer performs a linear transformation of the input features into a space of potentially different dimensionality, represented by the equation $output = activation(dot(input, kernel) + bias)$. Here, $kernel$ and $bias$ stand for the weight matrix and bias vector, respectively. The lin-

ear transformation is optionally followed by the application of an activation function, introducing non-linearity to the model's operations. This enhancement is crucial for the model's ability to capture complex patterns within the data. For the activation function, the Gaussian Error Linear Unit (GELU) is employed, defined as $f(x) = 0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3)))$. The GELU function offers a more refined non-linearity than ReLU, assisting in alleviating the vanishing gradient issue and enabling more effective model training by better accommodating the data's distribution [38].

Implementing the Mixing Operations

In the TensorFlow framework, dense layers act upon the last dimension of the input tensor. For an input tensor shaped $[Batch, Time, ROI]$, it becomes necessary to transpose the tensor to $[Batch, ROI, Time]$ to allow the dense layer to act on the temporal dimension. This transposition enables the model to treat time steps as features, facilitating "Time mixing".

The implementation of this concept is demonstrated in the residual block function, `res_block`, presented in Code Block 6.1. This function employs shared dense layers for both time and ROI mixing within the hierarchical time series encoder, accommodating inputs of consistent time steps. The process begins with a transposition of the input tensor for time mixing, where it undergoes normalisation and is processed through two dense layers, $W1\_time$ and $W2\_time$. Following this, the tensor is transposed back to its original shape, allowing for the integration of the time-mixed features with the original inputs through a residual connection. Subsequent to this time mixing, ROI mixing is performed on the resultant tensor using an analogous approach with two additional dense layers, $W1\_roi$ and $W2\_roi$, post-normalisation. The `res_block` thus outputs a tensor that has been enriched with features learned from both temporal and spatial dimensions, maintaining the input's original shape $[Batch, Time, ROI]$.

```
def res_block(inputs, W1_time, W2_time, W1_roi, W2_roi):
    """
    Residual block with time and ROI mixing.
    The layers are shared for processing inputs of the same shape (i.e. the
    ↪ same number of time steps)
    Args:
        inputs (tensor): Input tensor. Shape [Batch, Time, ROI]
        W1_time (layer): Dense layer for time mixing
        W2_time (layer): Dense layer for time mixing
        W1_roi (layer): Dense layer for ROI mixing
        W2_roi (layer): Dense layer for ROI mixing

```

```
    Returns:
        tensor: With time and roi-mixing applied. Shape [Batch, Time, ROI]
    """
    layer_norm_time = layers.LayerNormalization(axis=-1)
    layer_norm_roi = layers.LayerNormalization(axis=-1)

    # Time mixing
    x = tf.transpose(inputs, perm=[0, 2, 1])  # Transpose for time mixing
    x = layer_norm_time(x)
    x = W1_time(x)
    x = W2_time(x)
    x = tf.transpose(x, perm=[0, 2, 1])  # Transpose back for ROI mixing
    res = x + inputs

    # ROI mixing
    x = layer_norm_roi(res)
    x = W1_roi(x)
    x = W2_roi(x)
    return x + res
```

Code Block 6.1: The simplified `res_block` implementation for time and ROI-mixing.

6.5.4 Limitations

The implementation of the feature extraction component encounters some limitations that warrant consideration:

1. The implementation only works if all window sizes are multiples of smaller window sizes. In cases where this condition is not met, the division into smaller sub-segments does not work. A conceivable solution involves zero-padding these segments to fit the model's input requirements. However, dynamically adjusting the tensor dimensions to incorporate zero-padding presents a challenge within the TF, due to its strict requirements for the computational graph.

2. The current architecture utilises a single residual block (`res_block`) for each sub-segment within the hierarchical structure. This design choice limits the depth of feature extraction and mixing, potentially constraining the model's capacity to learn complex temporal and spatial relationships. Incorporating multiple `res_block` layers for each sub-segment could enhance model performance but would necessitate the creation of distinct shared layers for each block, increasing model complexity and computational requirements.

3. The feedforward dimension (`ff_dim`) parameter, is a remnant of the TSMixer implementation. It represents the output dimensionality of one of the shared layers within the `res_block`. Identifying a method to eliminate or dynamically adjust `ff_dim` based on the data could avoid possibly unnecessary complexity.

6.6 Hypergraph Construction

The implementation of the attention mechanism for hypergraph construction in the proposed model is relatively straightforward, leveraging the outputs from the feature extraction phase, specifically the stacked H_F_ts with shape [Batch, Segment, Time, ROI]. This phase transitions the feature extraction output into an incidence matrix representing the hypergraph. It employs a modular design, integrating two custom layers to achieve this transformation. An outer layer unstacks H_F_ts, converts each individual H_F_t into a hypergraph and then re-stacks them to form a temporal hypergraph of shape [Batch, Segment, Node, Hyperedge]. The number of ROIs corresponds to the number of ROIs and the number of Hyperedges corresponds to the number of time steps in H_F_t.

6.6.1 Custom Layers for Hypergraph Construction

The outer `TemporalHypergraphLayer` and the inner `HypergraphConstructionLayer` are implemented as custom TF layers, promoting maintainability and modularity.

Tailored Transformations

Custom layers enable the creation of tailored transformations specific to the requirements of temporal hypergraph construction. The process of translating fMRI data into a hypergraph format representing spatiotemporal interactions between ROIs demands specialised operations, notably the modified self-attention mechanism within `HypergraphConstructionLayer`.

Integration with TensorFlow's Ecosystem

Embedding these operations within custom layers facilitates seamless integration with TensorFlow's computational graph, allowing for optimised performance and efficient training. Custom layers leverage TensorFlow's capabilities for automatic differentiation and backpropagation, ensuring that the model's parameters are accurately updated during the learning process. This integration is essential for harnessing the full power of TensorFlow's optimisation algorithms and

for exploiting GPU acceleration, thereby enhancing the model's scalability and performance.

Isolation and Modularity

Wrapping the hypergraph construction transformations in custom layers isolates them from the rest of the model's architecture, promoting modularity. This isolation simplifies the model's complexity, making it more maintainable and allowing for focused experimentation and adjustment within each layer without impacting other components. Such modularity is crucial for the iterative development process, enabling rapid testing of new ideas and adjustments to the hypergraph construction mechanism to improve model accuracy and efficiency.

Parameter Management and Experimentation

Custom layers provide a structured framework for managing trainable parameters, such as the learnable matrices and well-structured handling of fixed parameters such as the dimension of the attention vector (`attn_dim`) or the number of hyperedges to be constructed (`hyperedge_dim`). This approach not only simplifies parameter tuning but also allows for more systematic experimentation with different configurations. Adjusting parameters becomes straightforward, facilitating the exploration of various setups to identify the most effective model configuration for capturing the nuances of brain connectivity.

In conclusion, the necessity of utilising custom layers for the hypergraph construction process arises from the need for tailored transformations, seamless integration with TensorFlow, and the benefits of isolation and modularity.

6.6.2 Incidence Matrix Creation

The incidence matrix $A^{(t)}$ of a segment t of the temporal hypergraph is constructed using the attention mechanism detailed in Section 5.4. A sample output of the IMs for each segment is shown in Figure 6.1. Using window sizes $S = 5, 10$ and the zero-padded length of all time series $T = 40$, four segments are created in the feature extraction.

6.6.3 Limitations

Duplicate Hyperedges

As depicted in the hypergraph visualisation of the IM's in Figure 6.2, there are a lot of duplicate hyperedges which might indicate a lack of power of expression of the hypergraph. One approach

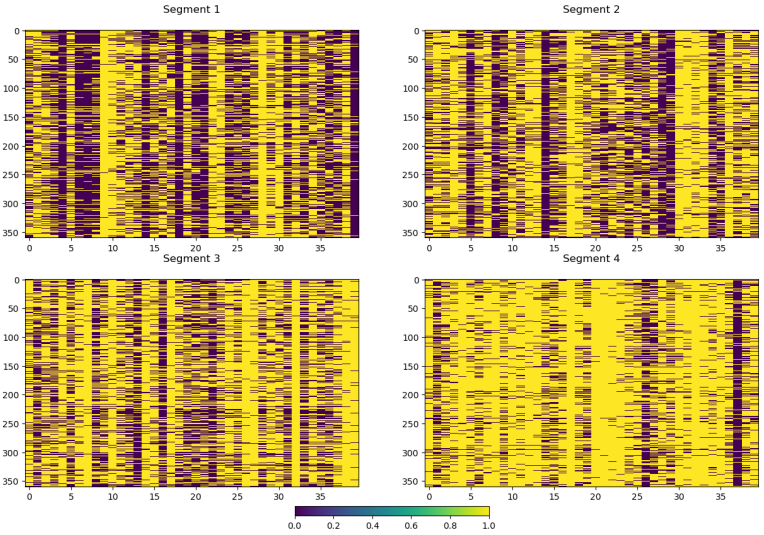


Figure 6.1: IMs representing the output of the temporal hypergraph construction. Each IM corresponds to one segment extracted by the feature extraction component.

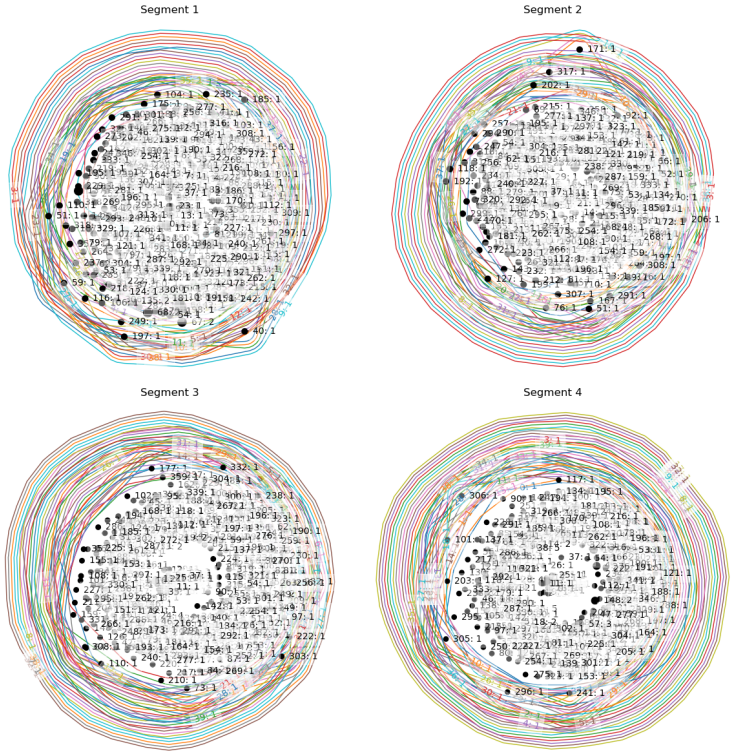


Figure 6.2: Hypergraph representation of the IMs in Figure 6.1. Each graph corresponds to one IM. Notably, there are a lot of duplicate hyperedges in segment 4.

to tackle this issue is to remove duplicate hyperedges after the construction of the hypergraph. To comply with TF's computational graph, this was implemented using TF operations, as shown in Code Block 6.2.

```
def _remove_duplicate_hyperedges(self, im):
    num_rows = self.roi_dim

    # Create an expanded version for comparison
    expanded_im = tf.expand_dims(im, 2) # [num_rows, num_cols, 1]
    comparisons = tf.equal(expanded_im, tf.transpose(expanded_im, [0, 2, 1]))
    ↪ # Shape: [num_rows, num_cols, num_cols]

    # Determine if columns are identical
    identical = tf.reduce_all(comparisons, axis=0) # [num_cols, num_cols]

    # Reduce to identify unique columns
    lower_triangle = tf.linalg.band_part(identical, -1, 0)  # Keep lower
    ↪ triangle and diagonal
    unique_mask = tf.reduce_sum(tf.cast(lower_triangle, tf.int32), axis=0) == 1
    ↪  # True for unique columns

    # Apply mask to original incidence matrix
    im_unique = tf.where(tf.tile(tf.expand_dims(unique_mask, 0), [num_rows,
    ↪ 1]), im, tf.zeros_like(im))

    return im_unique
```

Code Block 6.2: TensorFlow implementation for removing duplicate hyperedges in a hypergraph's incidence matrix, utilising boolean comparisons and tensor operations to identify and exclude identical columns, thereby ensuring each hyperedge is unique.

The algorithm presented for removing duplicate hyperedges from a hypergraph incidence matrix involves a sequence of operations designed to identify and exclude redundant hyperedges, thereby enhancing the hypergraph's expressiveness. Initially, the algorithm expands the incidence matrix to facilitate pairwise column comparison, identifying identical columns through a boolean equality check. This process relies on the discrete comparison operation `tf.equal`, which assesses whether elements in one column exactly match those in another, yielding a binary (true/false) matrix indicating identical columns. Subsequently, a masking strategy is employed to isolate unique columns, leveraging the fact that unique columns would have a distinct pattern in the resulting binary matrix, particularly when observed in the lower triangle

to avoid duplicate comparisons. By applying this mask, duplicate hyperedges are excluded, retaining only unique representations in the hypergraph.

However, the discrete nature of the comparison operation (`tf.equal`) introduces non-differentiability into the process, presenting challenges for gradient-based optimisation methods. Since the operation produces a binary output without a gradient, it cannot directly contribute to the computational graph in a manner that supports backpropagation. To address this limitation, a soft approximation approach could be considered, such as computing the negative Euclidean distance or cosine similarity between columns. These measures, being continuous and differentiable, offer gradients that can be propagated through the computational graph. Additionally, incorporating a sparsity measure of the graph into the loss function could further refine the model by penalising overly dense hypergraph representations. Nonetheless, to remain aligned with the HADiB implementation, such modifications were not pursued.

Predefined Number of Hyperedges

A notable limitation within the model's design is the necessity to predetermine the number of hyperedges, which must be fixed due the IM being represented as a tensor of the shape [ROI, Hyperedges]. This requirement is imposed to ensure compatibility with TF's computational graph, which necessitates predefined tensor dimensions for efficient computation and backpropagation. The imposition of a fixed hyperedge quantity can potentially restrict the model's adaptability and precision in representing complex, dynamic brain connectivity patterns, as it imposes a uniform hyperedge limit across all instances, regardless of the inherent variability in connectivity complexity. Exploring alternatives, such as the utilisation of `tf.RaggedTensor`, which accommodates tensors of varying shapes, could offer a pathway to more dynamically adjusting the number of hyperedges in accordance with the data's intrinsic characteristics. However, the adoption of `tf.RaggedTensor` introduces its own set of computational and implementation challenges, particularly in maintaining efficiency and compatibility within TensorFlow's ecosystem. This complexity has, thus far, deterred its application within the current model framework, highlighting a trade-off between flexibility and computational efficiency.

6.7 Hypergraph Learning

This section delves into the intricacies of implementing message passing within the context of learning dependencies of temporal patterns of brain activity. The primary objective is to facilitate an understanding of how message passing can be integrated into a TF model. The

discussion is structured to provide an overview, outline the challenges encountered, explore a custom implementation along with its limitations, present a solution utilising tf-gnn, and finally discuss the limitations of the approach.

6.7.1 Overview

The message passing mechanism is designed to refine the spatiotemporal dependencies detected by feature extraction and hypergraph construction. It aims at performing roi-hyperedge and roi-roi message passing, essential for understanding the complex dynamics of brain activity over time. Through the interaction between ROIs and hyperedges, message passing facilitates communication within the hypergraph, enabling the model to capture nuanced relationships and dependencies, which are crucial for a comprehensive analysis of temporal patterns.

6.7.2 Challenges

The implementation of message passing within the model posed several challenges:

1. The integration of hypergraph construction within the model presents a departure from the conventional approach, where the graph's construction is typically handled during preprocessing, often employing SWC. This deviation raises critical questions regarding the seamless integration of hypergraph construction and message passing within the model's architecture, specifically focusing on the necessity for employing solely differentiable operations on tensors. The challenge lies in ensuring that the entire process—from hypergraph construction to message passing—is fully compatible with the TF computational graph.

2. The need to aggregate encodings of ROIs and hyperedges require a storage solution that is both efficient and compatible with the model's architecture.

3. The absence of readily available libraries for hypergraph message passing that are both compatible with the TF functional API and support differentiable operations significantly increased the complexity of implementation. Initial investigations into libraries such as Deep Graph Library (DGL), Deep Hypergraph Library (DHG), Pytorch Geometric (PyG) and tf-gnn [96, 16, 26, 25] highlighted compatibility and integration issues with TF:

    b) Among the considered libraries, PyG and DHG are tailored for PyTorch, necessitating a complete rewrite of the model to integrate either library.

    c) DGL, while offering TF support, is optimised for PyTorch, resulting in difficulties during integration attempts.

d) Tf-gnn demonstrates good integration with TF but lacks support for hypergraphs, rendering it unsuitable for direct application in hypergraph message passing.

6.7.3 Custom Implementation

To address the identified challenges, a custom implementation was developed initially, encapsulating a novel approach to hypergraph message passing within the TensorFlow framework. This section details the design and functionality of the custom implementation, which is composed of the `TensorHypergraph` class and a bespoke message-passing layer.

TensorHypergraph Class

The core of the custom implementation is the `TensorHypergraph` class, which represents the hypergraph structure using three tensors:

1. The incidence matrix tensor, delineating the connections between nodes and hyperedges.
2. A tensor for hyperedge encodings, capturing the attributes or features associated with each hyperedge.
3. A tensor for node encodings, representing the features of each node.

These tensors are initialised based on the input tensors A_ts and H_F_ts, which represent the adjacency time slices and the feature time slices, respectively, as detailed in the design sections 5.2 and 5.3. This structure allows for the dynamic representation of hypergraphs, facilitating updates to node and hyperedge encodings through message passing. The `TensorHypergraph` provides an API for operations on the hypergraph such as retrieving neighbours of a node. Its code is provided in Appendix C.1.

Message Passing Custom Layer

The message-passing functionality is encapsulated within a custom TensorFlow layer, designed to operate on the three tensors described above. This layer's input comprises the incidence matrix, hyperedge encoding, and node encoding tensors, with shapes $[Batch, Segment, Node, Hyperedge]$, $[Batch, Segment, Hyperedge, Encoding]$, and $[Batch, Segment, Node, Encoding]$, respectively. Its implementation is provided in C.1.

The computation of messages between nodes and hyperedges is orchestrated by two nested `tf.map_fn` functions. The outer `map_fn` iterates over each element in the batch dimension, reducing the tensors' dimensions from $[Batch, Segment, X, Y]$ to $[Segment, X, Y]$. Subsequently,

the inner `map_fn` further reduces the dimensions from $[Segment, X, Y]$ to $[X, Y]$, effectively applying the message-passing operations to each time slice independently. The code is provided in C.1.

This nested application of `tf.map_fn` enables the processing of each batch and segment, facilitating the generation of updated node and hyperedge encodings through message passing.

Custom Message Passing Implementation

The custom message passing implementation within the `TensorHypergraph` is designed to leverage TensorFlow's computational graph for efficient and differentiable operations on tensors representing the hypergraph structure. The primary goal is to update the encodings of nodes based on the interactions among nodes, hyperedges, and their corresponding encodings. This process is detailed in the function `compute_messages_for_single_t`, which is integral to the custom layer responsible for message passing.

```
def compute_messages_for_single_t(args):
    """
        Node encodings: [Node, Encoding]
        Hyperedge encodings: [Hyperedge, Encoding]
        Incidence matrix: [Node, Hyperedge]
    """
    node_encodings, hyperedge_encodings, incidence_matrix, W_roi = args

    # Assuming TensorHypergraph correctly initializes from the provided args
    hypergraph = TensorHypergraph(incidence_matrix, node_encodings,
↪ hyperedge_encodings)
    new_node_encodings = []

    for u in range(incidence_matrix.shape[0]): # Iterate over each node
        neighbors_indices = hypergraph.get_connected_nodes(u)
        messages_for_u = []

        for v in neighbors_indices: # Iterate over neighbors of node 'u'
            u_enc = tf.gather(params=node_encodings, indices=u)
            v_enc = tf.gather(params=node_encodings, indices=v)
            common_hyperedges = hypergraph.get_hyperedges_connecting_nodes(u, v)
            e_encs = tf.gather(params=hyperedge_encodings,
↪ indices=common_hyperedges)
            e_enc = tf.reduce_sum(e_encs, axis=0) # sum across all hyperedges
↪ that connect two nodes
```

```
            concatenated_encoding = tf.concat([u_enc, v_enc, e_enc], axis=0)
            # Reshape for multiplication
            concatenated_encoding = tf.reshape(concatenated_encoding, [1, -1])
            # Message computation
            message = tf.matmul(concatenated_encoding, W_roi)

            # Squeeze out the singleton dimension introduced by reshaping
            # Specify axis=0 to only remove the first singleton dimension
            message = tf.squeeze(message, axis=0)

            messages_for_u.append(message)

        aggregated_message = tf.reduce_sum(messages_for_u, axis=0)

        updated_node_encoding = u_enc + aggregated_message
        new_node_encodings.append(updated_node_encoding)

    # Returning the updated node encodings
    return tf.stack(new_node_encodings, axis=0)
```

Code Block 6.3: Implementation of ROI-ROI message passing.

This implementation in Code Block 6.3 illustrates the sequential steps involved in the message passing algorithm, using TF operations. The key operations and their implications in the algorithm are as follows:

- The `tf.gather` operation is employed to retrieve specific node and hyperedge encodings based on indices, facilitating targeted manipulation of tensor slices that represent nodes and hyperedges.
- The function `get_connected_nodes` identifies the indices of nodes that are directly connected to a given node. This selection is crucial for local message aggregation and is made possible by the incidence matrix representation within the `TensorHypergraph` class.
- The `get_hyperedges_connecting_nodes` function retrieves the indices of hyperedges connecting two nodes, providing the basis for incorporating hyperedge information into node-to-node communication.
- The `tf.reduce_sum` operation aggregates encodings across hyperedges and messages for a node, ensuring that the influence of multiple hyperedges and the cumulative effect of messages from neighboring nodes are accounted for in the updated node encoding.

- The use of `tf.concat` and `tf.reshape` operations prepares the tensors for matrix multiplication with `W_roi`, a learnable weight matrix that transforms the concatenated encodings into a message vector.
- Finally, `tf.stack` aggregates the updated node encodings into a tensor, completing the update process for the entire batch of nodes within the hypergraph.

Problems with the Custom Implementation

The custom implementation of message passing in the `TensorHypergraph` framework encounters significant challenges.

Non-differentiable Operations The core issue revolves around the inability to circumvent the use of non-differentiable operations for key functionalities within the `TensorHypergraph`. Specifically, the `get_connected_nodes` and `get_hyperedges_connecting_nodes` (C.1) functions, essential for identifying the neighbours of a node and determining all hyperedges that connect two nodes, respectively, rely on `tf.logical_and`. This reliance inherently restricts the gradient flow during backpropagation, rendering the model's training suboptimal. As a consequence, the model exhibits difficulties in adequately learning from the training data.

Training Efficiency and Accuracy The training process for this implementation is slow, with a duration of approximately 65 minutes for only 32 subjects across window sizes $S = \{10, 20\}$ for GAMBLING and SOCIAL tasks, comparing to around one minute the final solution takes to train on these parameters. This inefficiency is further compounded by a marked decrease in accuracy of the validation data, indicating that the model struggles to generalise from the training data effectively, as shown in Figure 6.3. A pivotal observation is the considerable acceleration in training speed over successive epochs—from 10 minutes for the first epoch to a mere 3 seconds for the 35th epoch. An investigation showed that the IM evolved to depict no connections by hyperedges. Consequently, with fewer neighbors to consider, the demand for message passing diminished, thereby expediting computation. However, this "speedup" inadvertently highlights the model's regression in learning—progressing towards a hypergraph with no hyperedges.

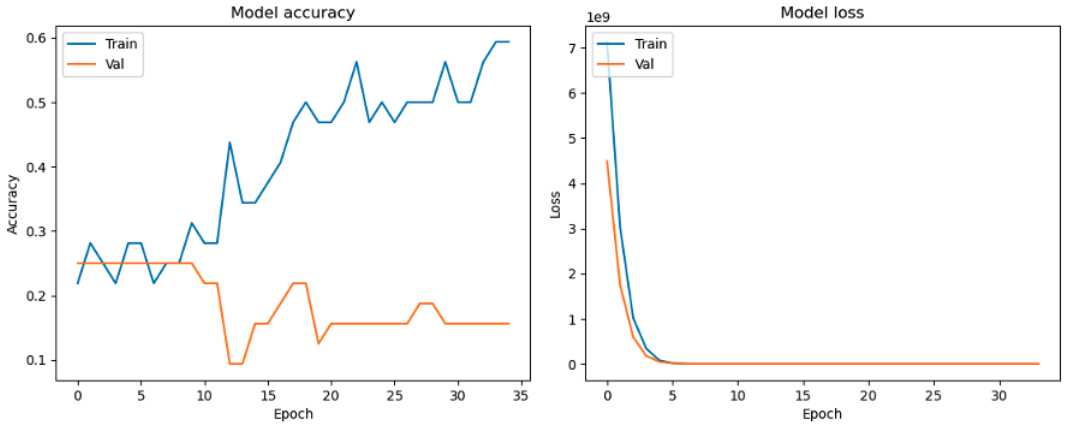


Figure 6.3: Performance of the model using the custom message passing implementation.

6.7.4 Solution Using TensorFlow Graph Neural Networks (tf-gnn)

To overcome the limitations of the custom implementation, a solution using tf-gnn is proposed. This approach involves converting the hypergraph to a bipartite graph, enabling the application of message passing using tf-gnn's capabilities. The theory of conversion to and message passing on bipartite graphs is explained in 2.3.5.

Converting a Hypergraph to a Bipartite Graph in TensorFlow

The conversion of the IM to a bipartite graph is differentiable. In order to create a `tf.GraphTensor` object, one can specify the types of nodes (hyperedge nodes and ROI nodes) and provide an array with the indices of the connections between these nodes.

```
@tf.function
def incidence_matrix_to_graph_tensor_with_roi_features(incidence_matrix, ...)
    num_col_nodes = incidence_matrix.shape[1]
    ...
    # Create edges from non-zero entries of the incidence matrix
    nonzero_indices = tf.where(tf.reshape(incidence_matrix, [-1]) > 0)

    # Compute source (row) and target (column) indices for edges
    # Flatten the incidence matrix and get indices of non-zero entries
    flat_nonzero_indices = nonzero_indices[:, 0]
    # Compute row indices: divide the flat indices by the number of columns
    row_indices = flat_nonzero_indices // num_col_nodes
    # Compute column indices: get the remainder of the division
    col_indices = flat_nonzero_indices % num_col_nodes
```

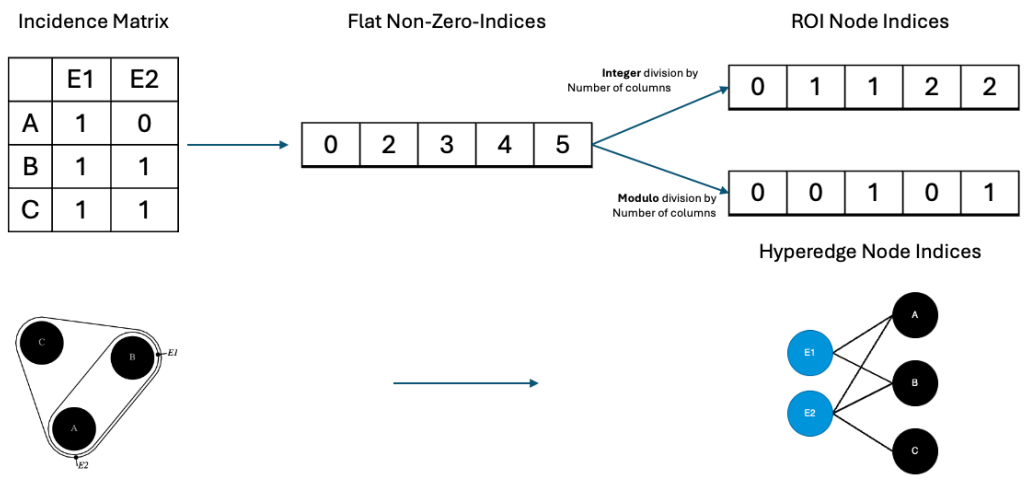


Figure 6.4: Illustration of the transformations applied to convert an IM to a bipartite graph.

```
15      ...
```

Code Block 6.4: Excerpt from the function for converting an IM to a bipartite graph, showing the code for the conversion

This object then serves as the basis for ROI-to-hyperedge and hyperedge-to-ROI message passing, utilising tf-gnn's low-level message passing API.

The usage of the modulo operator within the function for converting an incidence matrix to a bipartite graph, as shown in line 14 in code excerpt 6.4, raises questions about differentiability in the context of TensorFlow's computational graph. The modulo operation is inherently non-differentiable. However, in this specific application this does not impede the model's learning process. This is due to the operation is encapsulated within a `@tf.function`, which compiles a Python function into a callable TensorFlow graph [90]. This conversion facilitates optimisation and execution efficiency but, more importantly, abstracts away certain non-differentiable operations. TensorFlow is designed to handle such operations gracefully, bypassing them during the backward pass of backpropagation when computing gradients, assuming these operations are not directly involved in the loss computation or any gradient-based updates [48].

Bipartite Graph Message Passing

In TensorFlow, the transformation of a hypergraph to a bipartite graph structure enables the execution of message passing using tf-gnn. The `roi_roi_message_passing` function implements a two-stage message passing mechanism in this bipartite representation. Initially, messages are propagated from ROI nodes to hyperedge nodes, followed by a reverse transmis-

sion from hyperedge nodes back to ROI nodes. This bidirectional flow ensures that messages from all adjacent nodes are aggregated. Each stage involves a sequence of two operations: `tfgnn.broadcast_node_to_edges()` disseminates node attributes across connected edges, and `tfgnn.pool_edges_to_node()` aggregates these edge attributes back to the nodes. The heterogeneous bipartite graph distinguishes between two node types: ROI and hyperedge nodes. The initial stage enriches hyperedge nodes with the aggregated data from connected ROI nodes, while the subsequent stage redistributes this aggregated information back to the ROI nodes, facilitating indirect communication among ROI nodes via hyperedge-node intermediaries. The implementation details are provided in Code Block 6.5.

```
  def roi_roi_message_passing(graph, roi_node_set_name, he_node_set_name,
↪ hidden_state_name, edges_name):
  """Performs two passes of message passing in a bipartite hypergraph
↪ representation.
  1. From roi_nodes to he_nodes (hyperedges).
  2. From he_nodes (hyperedges) back to roi_nodes.
  This accumulates the messages from all neighbours of a node in the
↪ hypergraph representation.

  Args:
      graph (tfgnn.GraphTensor):
      roi_node_set_name (string): The name of the nodes in the graph
↪ representing ROIs.
      he_node_set_name (string): The name of the nodes in the graph
↪ representing Hyperedges.
      edge_set_name (string): The name of edge set name. In the bipartite
↪ graph there is only one type and defaults to tfgnn.EDGES constant.
  """
  # First pass: roi_nodes to he_nodes
  sender_features_to_he = tfgnn.broadcast_node_to_edges(
      graph, edges_name, tfgnn.SOURCE, feature_name=hidden_state_name)
  he_new_features = tfgnn.pool_edges_to_node(
      graph, edges_name, tfgnn.TARGET, reduce_type="sum",
      feature_value=sender_features_to_he)

  # Update he_node features with the result of the first pass
  he_updated_features = {he_node_set_name: {hidden_state_name:
↪ he_new_features}}
  graph = graph.replace_features(node_sets=he_updated_features)

```

```
    # Second pass: he_nodes back to roi_nodes
    sender_features_to_roi = tfgnn.broadcast_node_to_edges(
        graph, edges_name, tfgnn.TARGET, feature_name=hidden_state_name)
    roi_new_features = tfgnn.pool_edges_to_node(
        graph, edges_name, tfgnn.SOURCE, reduce_type="sum",
        feature_value=sender_features_to_roi)

    # Get original roi_node features for skip connection
    original_roi_node_features =
↪ graph.node_sets[roi_node_set_name].features[hidden_state_name]

    # Add (sum) the original roi_node features to the new roi_node features for
↪ skip connection
    updated_roi_node_features = roi_new_features + original_roi_node_features

    # Update roi_node features with the result of the second pass
    roi_updated_features = {roi_node_set_name: {hidden_state_name:
↪ updated_roi_node_features}}
    updated_graph = graph.replace_features(node_sets=roi_updated_features)

    return updated_graph
```

Code Block 6.5: Function for doing one round of roi-roi message passing

Integration of tf-gnn Solution into the ML Model

Integrating the tf-gnn solution into the model addresses the challenge of handling 4-dimensional tensors with dimensions corresponding to $[Batch, Segment, Node, Hyperedge]$. Utilising a strategy akin to the custom solution, two nested `tf.map_fn` transform the incidence matrices of the temporal hypergraph (A_ts) and the mixed time series segments ($H\_F\_ts\_T$) for bipartite graph construction and message passing. Unlike the custom implementation, this method converts input tensors into a `tf.GraphTensor`, performs message passing, and outputs the updated node encodings as a tensor within the inner `tf.map_fn`, complying with its required output signature.

6.7.5 Limitations

The current tf-gnn implementation exhibits following limitations:

1. As shown in Equation 5.5, the first step of message passing is the ROI-Hyperedge phase, which introduces the hyperedge encodings based on the nodes it contains. Subsequently,

in the ROI-ROI-phase, each message sent to an ROI should also contain this hyperedge encoding. Due to issues around differentiability, the computation of hyperedge encodings could not be implemented.

2. As shown in Equation 5.6, HADiB specifies a learnable matrix $W_{ROI}^{(\ell)}$ to be multiplied with each message, which is currently not implemented and might hamper the complexity of patterns the model can learn.

3. There is presently only one round of message passing. While this facilitates initial inter-node communication, it may not suffice for deeper learning of complex patterns. The architecture of HADiB suggests the potential for multiple layers of message passing, akin to practices in standard GNNs, to capture a richer representation of relationships within the data.

4. The lack of message passing between different segments of the temporal hypergraph restricts the model's capacity to incorporate the temporal dimension comprehensively. Introducing inter-segment message passing could significantly enhance the model's ability to learn from and exploit the temporal dynamics inherent in the data.

6.8 Classification Head and Loss

6.8.1 Overview

The classification head serves as the final component, responsible for transforming the feature representations learned by the model into predictions of class membership probabilities. This transformation is accomplished through a series of operations that include flattening or pooling the updated node encodings in order to combine them into a single tensor, followed by a dense layer equipped with a softmax activation function. The choice between using a flattening operation and global average pooling hinges on the model's ability to learn patterns effectively within the training data, and the implications this has for its performance on validation and test sets.

6.8.2 Implementation Details

Both flattening and global average pooling were evaluated as methods to condense the feature representations into a form suitable for classification. The findings indicated that while both approaches yielded an identical accuracy of 89% on the test set, the training accuracy was

notably higher when employing the flattening operation (71% and 91%). This suggests that the flattening approach allows the model to capture and learn from the patterns in the training data more effectively than global average pooling. The learnable parameters for both configurations amounted to approximately 4.1 million [1], excluding model complexity from influencing the decision. The fixed random parameters detailed in A.1 ensure reproducibility. Consequently, the model was configured to utilise the flattening operation in the classification head, with a subsequent dense layer activated by a `softmax` function to output the probabilities of class membership.

6.8.3 Loss Function: Sparse Categorical Accuracy

The selection of the loss function for the model was informed by the structure of the target data, which is encoded using label encoding. Given this context, the sparse categorical cross-entropy loss function was deemed the most appropriate choice. This loss function compares the predicted probability distribution for each class against the true distribution, where the true class label is represented as an integer. Sparse categorical cross-entropy is computationally efficient and particularly suited for scenarios where the target classes are mutually exclusive [89].

The computation of the sparse categorical accuracy involves determining the class with the highest predicted probability and comparing it against the true class label for each instance in the dataset. The accuracy metric, therefore, reflects the proportion of instances for which the model's predictions match the actual labels, providing a straightforward and interpretable measure of the model's performance on the classification task [89].

6.9 Summary

The implementation of HADiB started with the selection of TF as the foundational framework due to its robust support for computational graphs and machine learning operations. The data loader component addressed the challenges of handling the complexity and volume of the HCP t-fMRI dataset, employing strategies such as utilising a pre-processed subset and extracting time series data efficiently.

The ML model's construction was delineated into several key components: feature extraction, hypergraph construction, and the learning mechanism through message passing. The feature extraction process harnessed hierarchical time series encoding, overcoming challenges

[1] Using window sizes $S = \{5, 10\}$ and a total sequence length $T = 40$.

related to shared layers and the implementation of time and ROI mixing to enhance feature representation. Hypergraph construction translated these features into a hypergraph format, utilising custom TF layers for flexibility and modular design. Despite challenges in implementing non-differentiable operations within the custom message passing solution, a transition to utilising tf-gnn provided a viable pathway for integrating message passing into the model.

The classification head, pivotal in translating learned features into class membership probabilities, demonstrated the effectiveness of flattening operations over global average pooling in capturing training data patterns. Sparse categorical cross-entropy was employed as the loss function, aligning with the nature of the target data and facilitating a clear interpretation of the model's classification accuracy.

Evaluation

7.1 Performance Evaluation

The performance of the model is assessed through its ability to classify cognitive tasks using t-fMRI time series from the Neuromatch Academy subset of the HCP1200 dataset. The evaluation encompasses analyses across different parameters, such as window sizes and the number of classes, with a focus on understanding the model's performance under varying conditions.

7.1.1 Runs with Different Parameters

The model was trained multiple times using with unique combinations of window sizes and task categories, excluding language tasks due to the model's current limitations in processing these due to their variable length. This selection strategy was employed to evaluate the impact of temporal dynamics and task complexity on the model's performance, with a focus on a subset of runs. This approach was necessitated by constraints on computational resources and to present a clearer overview of the model's capabilities. The runs use the same random state as detailed in Appendix A.1.

Tasks (Number of Sub-Tasks)	Window Size	Accuracy
MOTOR (6)	20	0.85
	5, 10	0.96
	10, 20	0.96
	5, 10, 20	0.97
GAMBLING, SOCIAL (4)	20	0.88
	5, 10	0.90

Continued on next page

Tasks	Window Size	Accuracy
	10, 20	0.90
	5, 10, 20	0.89
GAMBLING, WORKING MEMORY (WM) (10)	20	0.73
	5, 10	0.83
	10, 20	0.83
	5, 10, 20	0.81
EMOTION, GAMBLING, RELATIONAL, SOCIAL (8)	20	0.84
	5, 10	0.88
	10, 20	0.88
	5, 10, 20	0.89
EMOTION, GAMBLING, RELATIONAL, SOCIAL,	20	0.85
MOTOR, WM (22)	5, 10	-
	10, 20	0.87
	5, 10, 20	0.89

Table 7.1: A selected collection of runs of the model showing the tasks included, the window sizes and the accuracy. '-' indicates that the model crashed when training on this particular combination.

Influence of Window Sizes

A notable observation was the model's enhanced performance with an increased number of window sizes. This improvement suggests that multiple window sizes enable the model to capture a broader range of temporal dynamics, which seems to be crucial for accurately classifying tasks based on fMRI data. However, the performance did not change when comparing the use of two window sizes (5, 10) and (10, 20), indicating that not the granularity of the windows but the number of windows is an indicator for performance.

Effect of Number of Sub-Tasks

As the number of high-level tasks increases, there is a noticeable decline in model performance, suggesting that the model's complexity may not adequately capture the nuances of various tasks. Interestingly, the model demonstrates proficiency in distinguishing between different motor movements, encompassing 6 sub-tasks, whereas it struggles with the GAMBLING and SOCIAL tasks, which involve only 4 sub-tasks. The confusion matrix reveals a pronounced difficulty in

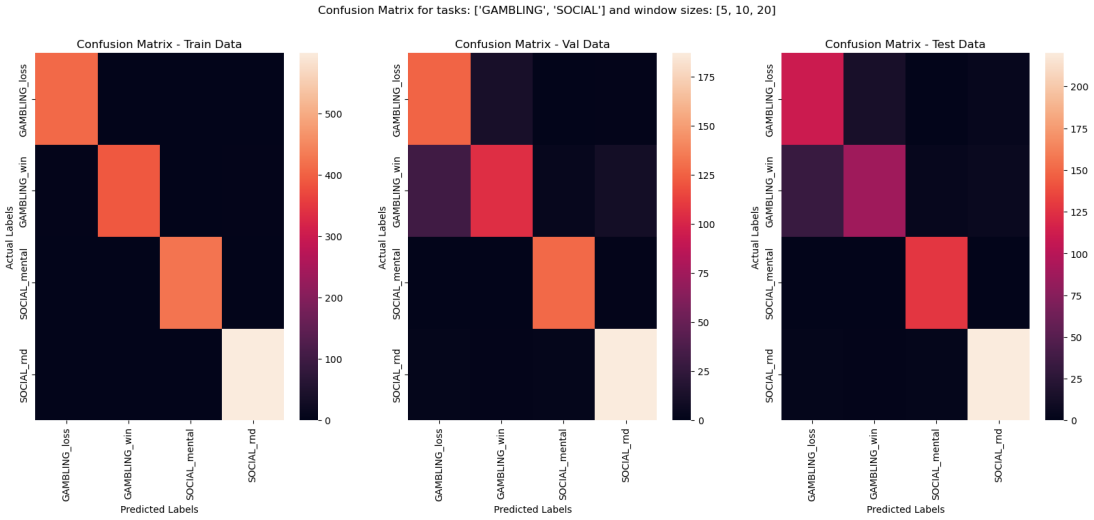


Figure 7.1: Confusion matrix for train, validation and test data for a model trained on GAMBLING and SOCIAL tasks using window sizes [5, 10, 20]. The main source of inaccuracy is the misclassification of the gambling sub-tasks.

distinguishing between 'gambling win' and 'gambling loss' outcomes, a challenge that extends to the combination of WM and GAMBLING tasks. This pattern indicates a broader issue with accurately differentiating between the types of GAMBLING-related activities, which has also been reported in the classification of the 7 high-level tasks by Etrugrul et al [73].

7.1.2 Comparison across Task Combinations

An intriguing aspect of the model's performance is its varying accuracy across different combinations of tasks. For instance, when GAMBLING tasks are combined with working memory (WM) tasks, there is a notable decrease in classification accuracy, approximately 10% worse than when GAMBLING is combined with social tasks. This pattern underscores a significant challenge in accurately classifying these combined tasks. The inclusion of a broader spectrum of tasks, as observed in the final experiment which integrates all tasks, leads to an improvement in performance. Upon inspection of the confusion matrix 7.3, it can be observed that the source of inaccuracy are the gambling, relational and motor tasks. The increase in overall accuracy in comparison to models trained on fewer tasks, but including e.g. gambling, can be attributed to the introduction of more easily classifiable tasks, pushing up the average. It would be interesting to run more combinations and investigate whether the accuracy of classifying a specific task changes when combined with other tasks, revealing similarities in brain activity to perform those tasks.

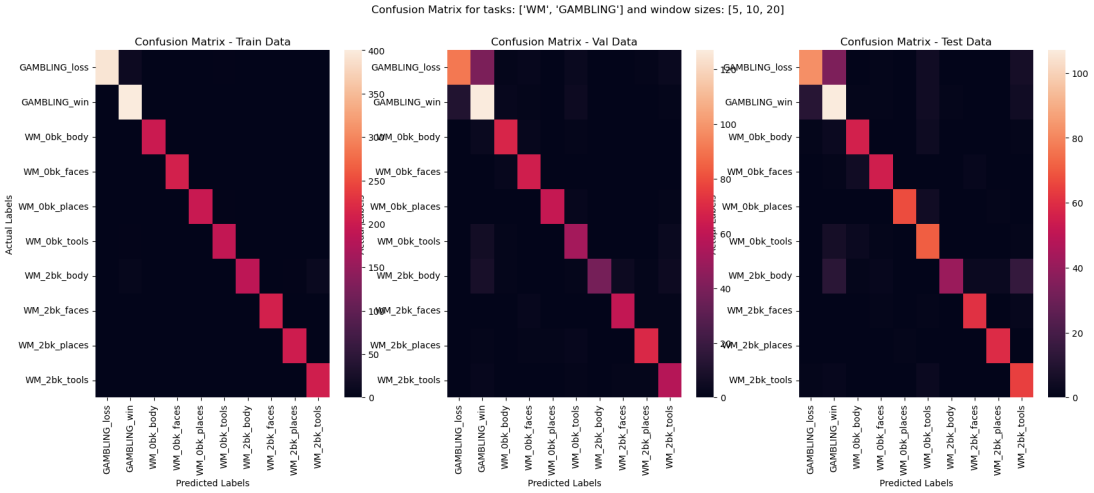


Figure 7.2: Confusion matrix for train, validation and test data for a model trained on GAMBLING and WM tasks using window sizes [5, 10, 20]. Again, the model has difficulty with classifying the gambling sub-tasks.

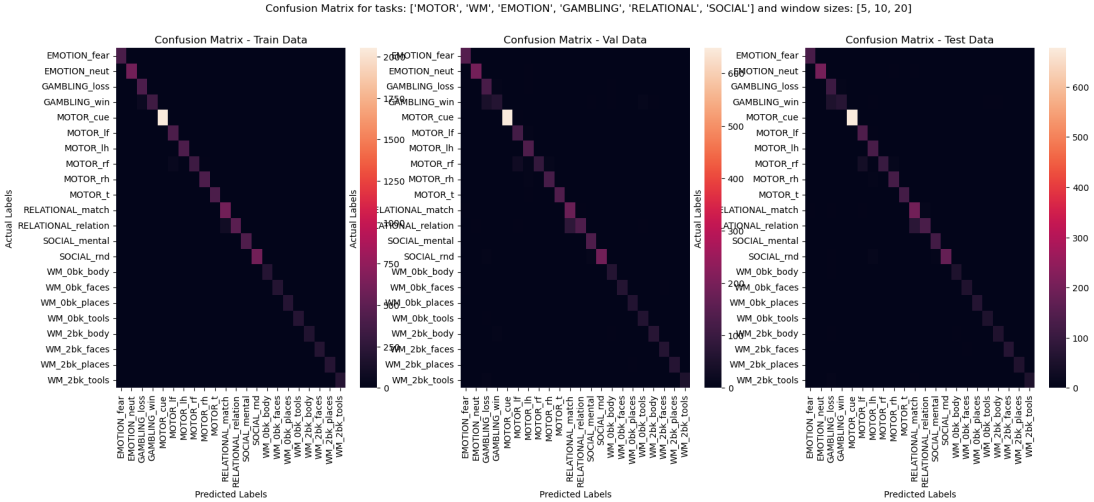


Figure 7.3: The confusion matrix for the combination of all tasks.

7.1.3 Conclusion

The evaluation reveals that the model's performance is significantly influenced by the configuration of window sizes, the complexity of task categories and the types of tasks included. While the model demonstrates a robust ability to capture temporal dynamics, its efficacy varies across when trained across different cognitive tasks. Future work should focus on optimising the selection of window sizes and improving the model's sensitivity to better distinguish between hard-to-classify tasks.

7.2 Comparison to Other Methods

The comparison of the current model's performance with that of existing methods is a critical component of this research. The field has seen various approaches for classifying cognitive tasks using task-based fMRI data, yet few directly align with the present work's scope, which includes a fine-grained classification of sub-tasks.

7.2.1 Challenges in Comparison

A fundamental issue in benchmarking the current model against others is the scarcity of comparable studies. Most existing methods restrict their scope to classifying a set of the 7 high-level tasks without delving into sub-task differentiation.

This distinction introduces a considerable complexity because, while high-level tasks can be classified by identifying distinct patterns of neural activity across different groups of ROIs, sub-tasks require a more refined analysis. High-level tasks involve distinct sets of ROIs, enabling classification through the differentiation of these groups. In contrast, sub-tasks involve variations in neural activation within the same brain areas, necessitating a higher degree of granularity in analysis.

Figure 7.4 illustrates the distinct groups of ROIs activated by different high-level tasks, demonstrating that classification at this level requires distinguishing between these groups. However, for sub-tasks associated with a specific high-level task, the challenge lies in discerning how the same brain areas are activated in nuanced ways. For instance, gambling-related tasks engage both the amygdala and the orbitofrontal cortex [14], while language tasks activate Broca's area [66]. This implies that, although the same groups of ROIs may be involved in all sub-tasks of a given high-level task, the specific patterns of activity within these groups must be identified to accurately classify sub-tasks.

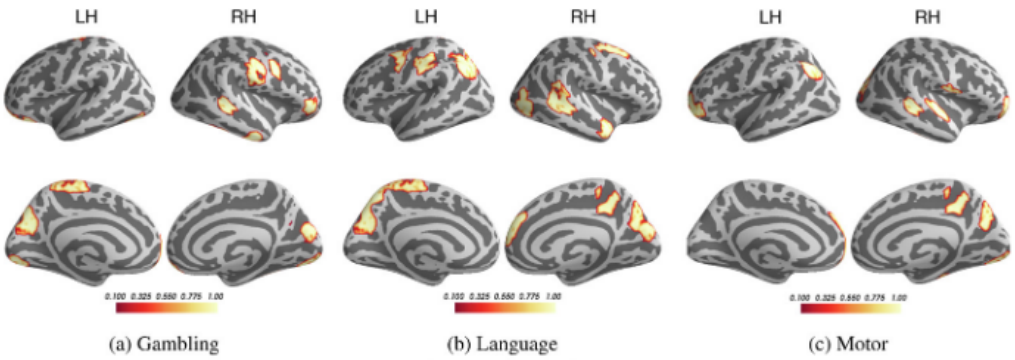


Figure 7.4: The brain areas relevant for classifying the Gambling, Language and Motor tasks with the importance scores shown in the colour bar [60].

Said et al. provide a comprehensive analysis of the high-level approaches in their work, "NeuroGraph: Benchmarks for Graph ML in Brain Connectomics", demonstrating an accuracy rate of approximately 95% for the 7 tasks in the HCP for several models [81].

7.2.2 Comparison of Model Performance on Sub-Tasks

The results presented in this section are derived from the comprehensive evaluations conducted in two papers: one by Zhao et al., introducing Group-DBRNN, and another by Ye et al., introducing STp-GCN [105, 103]. These studies provide an in-depth analysis of various computational approaches for sub-task classification within the HCP dataset, segmented into approaches using traditional ML, deep learning and GNNs. The segmentation aids in elucidating the distinct contributions of each model type to the decoding of brain activities, alongside their comparative efficacies.

- **Traditional Machine Learning Techniques:**
  - **Support Vector Machine (SVM):** Implemented in the studies by Zhao et al. and as SVM with Radial Basis Function (SVM-RBF) by Ye et al., SVM is foundational in traditional machine learning for classification tasks. The SVM-RBF variant particularly excels at handling nonlinear data relationships via the radial basis function kernel [67].
  - **Multi-Voxel Pattern Analysis (MVPA):** This approach, referenced by Zhao et al., applies pattern recognition and statistical learning to the analysis of data across multiple voxels [65].
  - **Softmax Regression:** Included in the analysis by Zhao et al., softmax regression extends logistic regression to accommodate multiple classes, proving useful in

classification tasks [105].

- **Deep Learning Approaches:**
  - **Deep Neural Network (DNN):** A comparison with DNNs is highlighted by Zhao et al., showcasing their potential in learning hierarchical representations of data [97].
  - **MLP-Mixer:** The MLP-Mixer serves as a state-of-the-art model in computer vision and natural language processing, functioning effectively in Euclidean space [91].
  - **Group-DBRNN:** This model combines deep learning with RNNs to efficiently decode brain activities [105].
- **Graph Neural Network Approaches:**
  - **Graph Convolutional Network (GCN):** This model simplifies graph convolutions to emulate convolutional neural networks in graph data, facilitating an intuitive approach to graph-based learning [57].
  - **Graph Isomorphism Network (GIN):** By extending the Weisfeiler-Lehman graph isomorphism test, GIN enhances the discriminative power among graph neural networks, making it a potent tool for graph-based analysis [100].
  - **Graph Attention Network (GAT):** Introduces an attention mechanism that allows for differential weighting of nodes within a neighborhood [93].
  - **STpGCN:** This model aims to encapsulate the spatial-temporal graph representation of functional brain activities through a multi-scale spatial-temporal pathway, enhancing the exploitation of the hierarchical nature of brain processing within a graph neural network framework [103].

Table 7.2 summarises the comparisons conducted by both, showing the accuracy of different models on sub-tasks within the HCP dataset and includes the HADiB-Adaption proposed in this paper:

7.2.3 Discussion of Model Performance and Comparison

The comparison of the HADiB-Adaption's performance with existing methodologies reveals significant insights into the complexities and challenges associated with sub-task classification in t-fMRI data. The analysis underscores the nuanced nature of this classification task, which not only demands distinguishing between broad categories of neural activities but also necessitates a granular understanding of subtle variations within the same group of ROIs across sub-tasks.

Model	Accuracy
SVM	15%
MVPA	17%
GCN	61%
DNN	80%
GIN	80%
SVM-RBF	81%
GAT	82%
HADiB-Adaption	89%
MLP-Mixer	90%
Group-DBRNN	91%
STpGCN	92%

Table 7.2: Comparison of classification accuracy of the sub-tasks in the HCP dataset, including results reported by Zhao et al. and Ye et al. [105, 103].

Insights from the Comparison

The performance metrics highlight several critical aspects of the current state of computational approaches for sub-task classification within the HCP dataset:

1. The HADiB-Adaption implemented in this report demonstrates a commendable accuracy of 89%, positioning it among the top-performing models in the domain of sub-task classification using the HCP dataset. However, it exhibits a marginal lag in performance behind the MLP-Mixer, Group-DBRNN, and STpGCN models, which report accuracies of 90%, 91%, and 92%, respectively. A distinguishing characteristic of these leading models is their incorporation of temporal dynamics in the analysis. Only HADiB and STpGCN explicitly consider higher-order interactions, but they suggest that this consideration is essential to their performance in ablation studies [6, 103]. These observation suggests that capturing the temporal evolution is paramount and when using graph representations, the complex inter-relationships within the brain's functional networks are essential for achieving high accuracy in sub-task classification tasks.

2. The analysis further reveals that the utility of graph-based approaches in this domain is nuanced. While the top-performing STpGCN model leverages graph representations to achieve superior outcomes, the MLP-Mixer and Group-DBRNN models attain comparable success without relying on graph-based data structures. However, when employing graphs, the consideration of higher-order relationships through hypergraphs is paramount, as indicated by the lower performance of methods employing normal conventional graphs (GCN, GAT, GIN).

3. Another noteworthy observation is the significant disparity in performance between tra-

ditional ML models, particularly the SVM variants employed by Zhao et al. and Ye et al. This discrepancy highlights the critical role of kernel choice and parameter optimisation in traditional machine learning approaches, suggesting that careful tuning could substantially impact model efficacy.

Limitations and Future Directions

While the HADiB-Adaption represents a solid approach in the field of cognitive task classification using fMRI data, there are several limitations that merit attention:

- The necessity for a more comprehensive validation approach for the HADiB-Adaption, including k-fold cross-validation, is apparent. This is especially relevant due to the small size of the dataset and would offer a more robust assessment of model performance and generalisability.
- The divergent performances of graph-based versus non-graph-based models suggest an avenue for further exploration into the fundamental mechanisms through which these models capture and process neural activity patterns. Understanding these mechanisms could inform the development of hybrid models that harness the strengths of both approaches.
- The current study's limited computational resources constrained the extent of hyperparameter tuning and exploration of larger datasets. Future work could address these limitations by leveraging more extensive computational infrastructure, potentially unveiling further enhancements in model performance.

In conclusion, the comparative analysis of the HADiB-Adaption model with existing approaches offers valuable insights into the current landscape of sub-task classification. The slightly lower performance of the model presents an opportunity to explore the effect of varying hyperparameters, the utilisation of the full HCP dataset and enhancements of the implementation to potentially enhance performance and exceed the current best models.

7.3 Limitations and Open Questions

The adaptation and implementation of the HADiB model for t-fMRI classification have highlighted several limitations and open questions, spanning from data handling and feature extraction to temporal hypergraph construction, hypergraph learning, the classification head, and the

model's general framework. Work on these present an opportunity surpass the performance of the best models presented in the previous section.

7.3.1 Data

1. **Length of t-fMRI Time Series:** The HADiB model, originally designed for longer r-fMRI time series with 1000 time steps, has been adapted for shorter t-fMRI time series of approximately 40 time steps. This adaptation raises concerns about its effectiveness in capturing dynamic brain changes inherent in task-based fMRI data due to the limited number of possible window sizes $s \in S$ for constructing the hierarchical time series.

2. **Limited Dataset Size:** The use of a subset of the HCP1200 dataset provided by the Neuromatch Academy restricts the number of instances available for training and evaluation, going from 1200 to 339 subjects. A more extensive dataset could potentially improve the model's performance and generalisability.

7.3.2 Feature Extraction

1. **Design Concerns:** The original HADiB model's operation, specified as 'MLP-MIXER' for producing the final feature representation H_F, was ambiguous. The decision to utilise a single dense layer might limit the model's complexity and ability to capture nuanced features.

2. **Implementation Challenges:** The hierarchical time series encoder's performance is contingent on the selection of window sizes S. The requirement for all window sizes to be a multiple of smaller window sizes might restrict the model's flexibility in capturing brain dynamics.

7.3.3 Temporal Hypergraph Construction

1. **Design Ambiguities:** The interpretation of the hypergraph, particularly in the context of using an attention mechanism for construction and the feature extraction using growing windows, remains unclear. This ambiguity impacts the ability to interpret the hypergraph accurately.

2. **Implementation Limitations:** The necessity to predetermine the number of hyperedges may not accurately capture the brain connectivity's complexity and variability.

7.3.4 Hypergraph Learning

1. **Design Clarification:** The definition of what constitutes an FS within the HADiB framework remains unclear. This ambiguity impacts the implementation of FS-related message passing.

2. **Adaptation Limitations:** The current solution utilising tf-gnn only performs ROI-ROI message passing without an additional learnable matrix and omits the ROI-Hyperedge phase which both could provide enhanced model performance.

7.3.5 Classification Head

The simplicity of the classification head, while effective, leaves room for experimentation with more sophisticated structures that might enhance the model's predictive performance.

7.3.6 General

1. **Temporal Dynamics:** The model incorporates temporal information through the MLP-Mixer but may not fully exploit the temporal dynamics present in t-fMRI data due to the absence of mechanisms like recurrent layers or additional temporal feature engineering.

2. **Errors with certain Tasks:** The model currently can not run the 'LANGUAGE' task, as it does not successfully account for the different lengths among the sub-tasks.

3. **Errors with certain Window Sizes:** For certain combinations of tasks, it was not possible to train the model using certain window sizes. For example the combination [EMOTION, GAMBLING, RELATIONAL, SOCIAL, MOTOR, WM] could not be trained with window sizes [5, 10].

7.3.7 Explainability

Explainability in ML models for neuroimaging is crucial for several reasons. For BCIs, the goal extends beyond mere prediction; the aim is to understand and interpret the brain's activity patterns. The challenge with fMRI is its relatively low spatial resolution, which often hampers the ability to pinpoint specific neuronal activities with the precision required for BCIs. However, fMRI data can still provide valuable insights into the broader neuroscientific context, such as understanding the coordination and interaction of various brain regions during certain tasks.

Understanding Spatiotemporal Patterns

In neuroscientific research, determining the spatiotemporal patterns that underlie specific tasks, like the movement of the left foot, is of high interest. Identifying these patterns could lead to a more nuanced understanding of the brain's functional organisation. It could also aid in the design of BCIs by correlating particular patterns with intended actions.

Current Model's Limitation

The present implementation lacks explainability, since it is unclear what spatiotemporal higher-order patterns it detects. Optimally, the output would include how groups of ROIs collaborate over time to achieve a specific task. Models like BrainGNN [60] offer a comparative framework, providing both accurate predictions and insights into the specific brain regions involved in each task. BrainGNN leverages GNNs to capture the relationships between regions and clustering to highlight the regions that are key for its predictions.

Improving explainability in ML model for fMRI analysis is not merely about enhancing the performance of a predictive model. It is also about augmenting our understanding of the brain's functioning, which can lead to breakthroughs in neuroscience and inform the development of BCIs. Incorporating explainability features into the current model is essential for making it a valuable tool for both practical applications and scientific discovery.

7.3.8 Conclusion

These limitations and open questions underscore the need for further research and development to enhance the model's effectiveness, flexibility, and applicability in the context of t-fMRI classification.

Legal, Social, Ethical and Professional Issues

8.1 AI and Brain Scans: Legal and Ethical Considerations

The integration of AI in brain imaging presents significant legal and ethical challenges, primarily around data privacy and the potential for misdiagnosis. These concerns are amplified in medical applications, such as disease classification, where inaccurate outcomes can have profound implications for patients.

8.1.1 Data Privacy

The use of AI in analysing brain scans raises substantial data privacy concerns. Brain imaging data contains sensitive personal information, making it imperative to ensure the confidentiality and security of such data [76]. The General Data Protection Regulation (GDPR) in the European Union sets forth stringent requirements for the processing of personal data, demanding robust measures to protect data against unauthorised access [78]. They have been adhered to by using the Open Access HCP dataset and accepting its terms of use [92].

8.1.2 Risk of Misdiagnosis

The application of AI in medical diagnostics, including brain imaging, introduces the risk of misdiagnosis. This risk is associated with the accuracy of the AI models employed. Models trained on limited or biased datasets may not generalise well across diverse populations, leading to potential misdiagnoses. The ethical imperative is to ensure AI systems are thoroughly validated across diverse datasets to minimise this risk [69].

8.2 Brain-Computer Interfaces: Ethical and Social Implications

Thinking further ahead, the development of BCIs leveraging insights from brain scans presents ethical issues concerning autonomy, consent, and potential misuse. As BCIs blur the lines between technology and human cognition, they raise questions about the extent to which individuals maintain control over their thoughts and actions when interfaced with AI [47].

8.2.1 Ethical Concerns Around BCIs

The use of BCIs involves ethical considerations related to autonomy and the potential for coercion. The ability of BCIs to interpret and act upon users' neural signals raises concerns about the voluntary nature of such interactions and the potential for BCIs to be used in ways that might compromise individual autonomy [17].

8.2.2 Social Implications

The advancement of BCIs also harbors social implications, particularly regarding accessibility and the digital divide. There is a risk that such technologies may exacerbate existing social inequalities, with access to potentially life-enhancing BCI technologies being limited to those who can afford them [8].

8.3 Professional Responsibilities

This work has been conducted in accordance with the code of conduct issued by the British Computer Society (BCS). The BCS Code of Conduct emphasises adherence to ethical considerations, the safeguarding of data privacy, and especially relevant for this project, importance of maintaining the accuracy and reliability of AI systems. The principles outlined by the BCS serve as a foundational framework ensuring that professionals uphold the highest standards of professional responsibility and ethical practice [5].

8.4 Conclusion

The integration of AI into brain imaging and the way it could potentially inform the development of BCIs raise complex legal, social, ethical, and professional issues. Addressing these

concerns requires a multidisciplinary approach, incorporating legal safeguards, ethical frameworks, and technological solutions to protect individuals' privacy, ensure the accuracy of medical diagnoses, and consider the broader social implications of these technologies.

Conclusion and Future Work

This research embarked on the journey to classify cognitive task sub-types using task-based t-fMRI data, leveraging the complex domain of temporal higher-order patterns within brain network dynamics. The study introduced a novel approach by adapting and extending the HADiB framework to analyse t-fMRI data, overcoming significant challenges inherent in the existing methodologies for fMRI analysis. The study's core achievements and learnings are outlined below:

1. **Adaptation and Extension of HADiB:** While there are some limitations to the developed model, the research still successfully adapted most of the HADiB framework, previously applied for anomaly detection in r-fMRI data, to classify cognitive sub-tasks in t-fMRI data. This adaptation harnessed higher-order temporal patterns in brain network dynamics, addressing significant challenges in existing fMRI analysis methodologies.

2. **Insights:** The study supports the neuroscientific premise that multiple brain areas collaborate over time to accomplish tasks. This can be seen in the fact that Temporal Hypergraphs seem to be a better representation of functional activity than conventional graphs.

3. **Overcoming technical Challenges:** The integration of message passing into a machine learning model, although challenging, was addressed by converting hypergraphs into bipartite graphs. This methodological advancement underscores the complexity and potential of leveraging graph theory concepts in neuroimaging data analysis.

4. **Overcoming Ambiguity - The Importance of Code Accessibility:** Reconstructing a model from the ground up presents substantial difficulties, particularly when the corresponding publication harbors ambiguities, and the code is not made available. It is recommended that researchers consider publishing their code or addressing inquiries

regarding any ambiguities in their work. This practice significantly facilitates the research community's ability to replicate, validate, and build upon existing work, thereby accelerating progress and enhancing the reliability of scientific advancements.

Future Directions

Several key areas have been identified for further development and exploration. These areas, detailed below, represent vital directions for advancing the model's efficacy, better understanding its capabilities and enhancing its explainability.

1. **Design and Implementation:** Future work should focus on resolving ambiguities in the model's design and refining the implementation, particularly verifying the functionality of the self-attention mechanism for hypergraph construction and findings ways to incorporate more levels of message passing.

2. **Comprehensive Evaluation:** Expanding the evaluation of the model, testing different preprocessing methods, training on on the full HCP1200 dataset, using k-fold cross validation to verify the model's generalisability, and performing hyperparameter tuning to find the optimal batch size and combination of window size, is crucial for further validation and improvement.

3. **Explainability:** The pursuit of model explainability stands as a paramount objective, with potential to guide neuroscience research into promising directions. Understanding the patterns learned by the model and validating them against established neuroscience literature could offer insights on how to build better ML models for fMRI analysis and inform new directions of neuroscientific research.

In conclusion, this research represents a significant stride towards understanding the complex temporal dynamics of brain networks through the lens of t-fMRI data. The innovative methodologies developed and the insights gained lay a solid foundation for future explorations, promising to enhance our understanding of cognitive processes and their neural substrates.

References

[1] Sophie Achard and Ed Bullmore. "Efficiency and cost of economical brain functional networks". In: *PLoS computational biology* 3.2 (Feb. 2, 2007), e17. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.0030017.

[2] Salim Arslan et al. *Graph Saliency Maps through Spectral Convolutional Networks: Application to Sex Classification with Brain Connectivity*. June 5, 2018. DOI: 10.48550/arXiv.1806.01764. arXiv: 1806.01764[cs]. URL: http://arxiv.org/abs/1806.01764 (visited on 03/24/2024).

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. July 21, 2016. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450[cs,stat]. URL: http://arxiv.org/abs/1607.06450 (visited on 12/12/2023).

[4] Danielle S. Bassett and Olaf Sporns. "Network neuroscience". In: *Nature Neuroscience* 20.3 (Mar. 2017). Number: 3 Publisher: Nature Publishing Group, pp. 353–364. ISSN: 1546-1726. DOI: 10.1038/nn.4502. URL: https://www.nature.com/articles/nn.4502 (visited on 11/27/2023).

[5] *BCS Code of Conduct for members - Ethics for IT professionals — BCS*. URL: https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/ (visited on 04/10/2024).

[6] Ali Behrouz and Farnoosh Hashemi. "Learning Temporal Higher-order Patterns to Detect Anomalous Brain Activity". In: NeurIPS 2023 AI for Science Workshop. Oct. 28, 2023. URL: https://openreview.net/forum?id=i7jjLwOOQm (visited on 01/29/2024).

[7] Marcel Bengs, Nils Gessert, and Alexander Schlaefer. *4D Spatio-Temporal Deep Learning with 4D fMRI Data for Autism Spectrum Disorder Classification*. Apr. 21, 2020. DOI:

10.48550/arXiv.2004.10165. arXiv: 2004.10165[cs,eess]. URL: http://arxiv.org/abs/2004.10165 (visited on 03/25/2024).

[8] Guillermo Bernal, Sean M. Montgomery, and Pattie Maes. "Brain-Computer Interfaces, Open-Source, and Democratizing the Future of Augmented Consciousness". In: *Frontiers in Computer Science* 3 (Apr. 14, 2021). Publisher: Frontiers. ISSN: 2624-9898. DOI: 10.3389/fcomp.2021.661300. URL: https://www.frontiersin.org/articles/10.3389/fcomp.2021.661300 (visited on 04/10/2024).

[9] Janine Bijsterbosch. *Introduction to resting state fMRI functional connectivity*. First edition. Oxford neuroimaging primers. Oxford, United Kingdom ; New York, NY: Oxford University Press, 2017. 138 pp. ISBN: 978-0-19-880822-0.

[10] Jeffrey R. Binder et al. "Mapping Anterior Temporal Lobe Language Areas with FMRI: A Multi-Center Normative Study". In: *NeuroImage* 54.2 (Jan. 15, 2011), pp. 1465–1475. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2010.09.048. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2997157/ (visited on 04/12/2024).

[11] John Bridle. "Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters". In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper/1989/hash/0336dcbab05b9d5ad24f4333c7658a0e-Abstract.html (visited on 04/06/2024).

[12] Vince D. Calhoun et al. "The Chronnectome: Time-Varying Connectivity Networks as the Next Frontier in fMRI Data Discovery". In: *Neuron* 84.2 (Oct. 22, 2014), pp. 262–274. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2014.10.015. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4372723/ (visited on 03/25/2024).

[13] Kyunghyun Cho et al. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. Oct. 7, 2014. DOI: 10.48550/arXiv.1409.1259. arXiv: 1409.1259[cs,stat]. URL: http://arxiv.org/abs/1409.1259 (visited on 04/05/2024).

[14] Luke Clark et al. "Pathological Choice: The Neuroscience of Gambling and Gambling Addiction". In: *Journal of Neuroscience* 33.45 (Nov. 6, 2013). Publisher: Society for Neuroscience Section: Symposium and Mini-Symposium, pp. 17617–17623. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3231-13.2013. URL: https://www.jneurosci.org/content/33/45/17617 (visited on 04/10/2024).

[15] Qionghai Dai and Yue Gao. "Mathematical Foundations of Hypergraph". In: *Hypergraph Computation*. Ed. by Qionghai Dai and Yue Gao. Singapore: Springer Nature, 2023, pp. 19–40. ISBN: 978-981-9901-85-2. DOI: 10.1007/978-981-99-0185-2_2. URL: https://doi.org/10.1007/978-981-99-0185-2_2 (visited on 03/22/2024).

[16] Qionghai Dai and Yue Gao. "The DeepHypergraph Library". In: Jan. 17, 2023, pp. 237–240. ISBN: 978-981-9901-84-5. DOI: 10.1007/978-981-99-0185-2_12.

[17] Erika J. Davidoff. "Agency and Accountability: Ethical Considerations for Brain-Computer Interfaces". In: *The Rutgers journal of bioethics* 11 (2020), pp. 9–20. ISSN: 2475-6431. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7654969/ (visited on 04/10/2024).

[18] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929[cs]. URL: http://arxiv.org/abs/2010.11929 (visited on 04/02/2024).

[19] Anand Eijlers et al. "Functional Network Dynamics on Functional MRI: A Primer on an Emerging Frontier in Neuroscience". In: *Radiology* 292 (June 25, 2019), p. 194009. DOI: 10.1148/radiol.2019194009.

[20] Vijay Ekambaram et al. "TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Aug. 6, 2023, pp. 459–469. DOI: 10.1145/3580305.3599533. arXiv: 2306.09364[cs]. URL: http://arxiv.org/abs/2306.09364 (visited on 02/07/2024).

[21] Taban Eslami et al. "ASD-DiagNet: A Hybrid Learning Approach for Detection of Autism Spectrum Disorder Using fMRI Data". In: *Frontiers in Neuroinformatics* 13 (2019), p. 70. ISSN: 1662-5196. DOI: 10.3389/fninf.2019.00070.

[22] L. Fan et al. "A Deep Network Model on Dynamic Functional Connectivity With Applications to Gender Classification and Intelligence Prediction". In: *Frontiers in Neuroscience* 14 (2020). ISSN: 1662-4548. DOI: 10.3389/fnins.2020.00881.

[23] Yifan Feng. *Structures in DHG — DHG 0.9.5 documentation*. URL: https://deephypergraph.readthedocs.io/en/latest/start/structure.html#introduction (visited on 03/25/2024).

[24] Yifan Feng et al. "Hypergraph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.1 (July 17, 2019). Number: 01, pp. 3558–3565. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33013558. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4235 (visited on 03/11/2024).

[25] Oleksandr Ferludin et al. *TF-GNN: Graph Neural Networks in TensorFlow*. July 23, 2023. DOI: 10.48550/arXiv.2207.03522. arXiv: 2207.03522[physics,stat]. URL: http://arxiv.org/abs/2207.03522 (visited on 04/08/2024).

[26] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. Apr. 25, 2019. DOI: 10.48550/arXiv.1903.02428. arXiv: 1903.02428[cs,stat]. URL: http://arxiv.org/abs/1903.02428 (visited on 04/08/2024).

[27] Michael D. Fox and Marcus E. Raichle. "Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging". In: *Nature Reviews Neuroscience* 8.9 (Sept. 2007). Publisher: Nature Publishing Group, pp. 700–711. ISSN: 1471-0048. DOI: 10.1038/nrn2201. URL: https://www.nature.com/articles/nrn2201 (visited on 03/20/2024).

[28] Soham Gadgil et al. "Spatio-Temporal Graph Convolution for Resting-State fMRI Analysis". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*. Ed. by Anne L. Martel et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 528–538. ISBN: 978-3-030-59728-3. DOI: 10.1007/978-3-030-59728-3_52.

[29] Yue Gao et al. "HGNN+: General Hypergraph Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (Mar. 2023). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 3181–3199. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3182052. URL: https://ieeexplore.ieee.org/document/9795251 (visited on 03/25/2024).

[30] Ahmed El-Gazzar et al. *A Hybrid 3DCNN and 3DC-LSTM based model for 4D Spatio-temporal fMRI data: An ABIDE Autism Classification study*. Feb. 14, 2020. DOI: 10.48550/arXiv.2002.05981. arXiv: 2002.05981[cs]. URL: http://arxiv.org/abs/2002.05981 (visited on 03/25/2024).

[31] Matthew F Glasser et al. "A multi-modal parcellation of human cerebral cortex". In: *Nature* 536.7615 (Aug. 11, 2016), pp. 171–178. ISSN: 0028-0836. DOI: 10.1038/nature18933.

URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4990127/ (visited on 04/05/2024).

[32] Matthew F. Glasser et al. "The Human Connectome Project's Neuroimaging Approach". In: *Nature neuroscience* 19.9 (Aug. 26, 2016), pp. 1175–1187. ISSN: 1097-6256. DOI: 10.1038/nn.4361. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6172654/ (visited on 11/14/2023).

[33] Matthew F. Glasser et al. "The minimal preprocessing pipelines for the Human Connectome Project". In: *NeuroImage* 80 (Oct. 15, 2013), pp. 105–124. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2013.04.127.

[34] Renzhou Gui, Tongjie Chen, and Han Nie. "Classification of Task-State fMRI Data Based on Circle-EMD and Machine Learning". In: *Computational Intelligence and Neuroscience* 2020 (2020), p. 7691294. ISSN: 1687-5273. DOI: 10.1155/2020/7691294.

[35] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. Sept. 10, 2018. DOI: 10.48550/arXiv.1706.02216. arXiv: 1706.02216[cs,stat]. URL: http://arxiv.org/abs/1706.02216 (visited on 03/25/2024).

[36] *HCP_3T Task fMRI Protocol Overview*. URL: https://www.humanconnectome.org/hcp-protocols-ya-task-fmri (visited on 03/26/2024).

[37] Anibal Sólon Heinsfeld et al. "Identification of autism spectrum disorder using deep learning and the ABIDE dataset". In: *NeuroImage : Clinical* 17 (Aug. 30, 2017), pp. 16–23. ISSN: 2213-1582. DOI: 10.1016/j.nicl.2017.08.017. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5635344/ (visited on 04/12/2024).

[38] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. June 5, 2023. DOI: 10.48550/arXiv.1606.08415. arXiv: 1606.08415[cs]. URL: http://arxiv.org/abs/1606.08415 (visited on 12/12/2023).

[39] Leanna M. Hernandez et al. "Neural signatures of autism spectrum disorders: insights into brain network dynamics". In: *Neuropsychopharmacology: Official Publication of the American College of Neuropsychopharmacology* 40.1 (Jan. 2015), pp. 171–189. ISSN: 1740-634X. DOI: 10.1038/npp.2014.172.

[40] Martijn P. van den Heuvel and Hilleke E. Hulshoff Pol. "Exploring the brain network: A review on resting-state fMRI functional connectivity". In: *European Neuropsychopharmacology* 20.8 (Aug. 1, 2010), pp. 519–534. ISSN: 0924-977X. DOI: 10.1016/j.

euroneuro.2010.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0924977X10000684 (visited on 03/25/2024).

[41] Martijn P. van den Heuvel et al. "Functionally linked resting-state networks reflect the underlying structural connectivity architecture of the human brain". In: *Human Brain Mapping* 30.10 (Oct. 2009), pp. 3127–3141. ISSN: 1097-0193. DOI: 10.1002/hbm.20737.

[42] Sajjad Heydari and Lorenzo Livi. "Message Passing Neural Networks for Hypergraphs". In: vol. 13530. 2022, pp. 583–592. DOI: 10.1007/978-3-031-15931-2_48. arXiv: 2203.16995[cs]. URL: http://arxiv.org/abs/2203.16995 (visited on 03/24/2024).

[43] Elizabeth M.C. Hillman. "Coupling Mechanism and Significance of the BOLD Signal: A Status Report". In: *Annual review of neuroscience* 37 (July 8, 2014), pp. 161–181. ISSN: 0147-006X. DOI: 10.1146/annurev-neuro-071013-014111. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4147398/ (visited on 11/29/2023).

[44] *hnx-widget 0.1.0-beta.4 Demo*. URL: https://pnnl.github.io/HyperNetX/index.html (visited on 03/22/2024).

[45] *Human Connectome Project - Homepage*. URL: https://www.humanconnectome.org/ (visited on 11/30/2023).

[46] R. Matthew Hutchison et al. "Dynamic functional connectivity: Promise, issues, and interpretations". In: *NeuroImage*. Mapping the Connectome 80 (Oct. 15, 2013), pp. 360–378. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2013.05.079. URL: https://www.sciencedirect.com/science/article/pii/S105381191300579X (visited on 03/25/2024).

[47] Marcello Ienca, Pim Haselager, and Ezekiel J. Emanuel. "Brain leaks and consumer neurotechnology". In: *Nature Biotechnology* 36.9 (Sept. 6, 2018), pp. 805–810. ISSN: 1546-1696. DOI: 10.1038/nbt.4240.

[48] *Introduction to gradients and automatic differentiation — TensorFlow Core*. TensorFlow. URL: https://www.tensorflow.org/guide/autodiff (visited on 04/08/2024).

[49] Mark Jenkinson and Michael Chappell. "Brain Anatomy for Neuroimaging". In: (2018).

[50] Mark Jenkinson et al. "FSL". In: *NeuroImage* 62.2 (Aug. 15, 2012), pp. 782–790. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2011.09.015.

[51] Biao Jie et al. “Hyper-connectivity of functional networks for brain disease diagnosis”. In: *Medical image analysis* 32 (Aug. 2016), pp. 84–100. ISSN: 1361-8415. DOI: 10.1016/j.media.2016.03.003. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5333488/ (visited on 03/24/2024).

[52] Cliff A. Joslyn et al. “Hypernetwork Science: From Multidimensional Networks to Computational Topology”. In: *Unifying Themes in Complex Systems X*. Ed. by Dan Braha et al. Cham: Springer International Publishing, 2021, pp. 377–392. ISBN: 978-3-030-67318-5. DOI: 10.1007/978-3-030-67318-5_25.

[53] KAIST Data Mining Lab. *Mining of Real-world Hypergraphs: Concepts, Patterns, and Generators (Tutorial) (Full)*. Oct. 25, 2022. URL: https://www.youtube.com/watch?v=0mD7zOBx10M (visited on 03/22/2024).

[54] Fikret Işık Karahanoğlu and Dimitri Van De Ville. “Dynamics of large-scale fMRI networks: Deconstruct brain activity to build better models of brain function”. In: *Current Opinion in Biomedical Engineering*. New Developments in Biomedical Imaging 3 (Sept. 1, 2017), pp. 28–36. ISSN: 2468-4511. DOI: 10.1016/j.cobme.2017.09.008. URL: https://www.sciencedirect.com/science/article/pii/S2468451117300417 (visited on 11/08/2023).

[55] Shella Dawn Keilholz. “The neural basis of time-varying resting-state functional connectivity”. In: *Brain Connectivity* 4.10 (Dec. 2014), pp. 769–779. ISSN: 2158-0022. DOI: 10.1089/brain.2014.0250.

[56] Byung-Hoon Kim and Jong Chul Ye. *Understanding Graph Isomorphism Network for rs-fMRI Functional Connectivity Analysis*. May 24, 2020. DOI: 10.48550/arXiv.2001.03690. arXiv: 2001.03690[cs,stat]. URL: http://arxiv.org/abs/2001.03690 (visited on 03/24/2024).

[57] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. Feb. 22, 2017. DOI: 10.48550/arXiv.1609.02907. arXiv: 1609.02907[cs,stat]. URL: http://arxiv.org/abs/1609.02907 (visited on 03/25/2024).

[58] Sofia Ira Ktena et al. “Metric learning with spectral graph convolutions on brain connectivity networks”. In: *NeuroImage* 169 (Apr. 1, 2018), pp. 431–442. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2017.12.052. URL: https://www.sciencedirect.com/science/article/pii/S1053811917310765 (visited on 03/24/2024).

[59] Geon Lee and Kijung Shin. *THyMe+: Temporal Hypergraph Motifs and Fast Algorithms for Exact Counting*. Sept. 16, 2021. DOI: 10.48550/arXiv.2109.08341. arXiv: 2109.08341[cs]. URL: http://arxiv.org/abs/2109.08341 (visited on 03/11/2024).

[60] Xiaoxiao Li et al. "BrainGNN: Interpretable Brain Graph Neural Network for fMRI Analysis". In: *Medical Image Analysis* 74 (Dec. 1, 2021), p. 102233. ISSN: 1361-8415. DOI: 10.1016/j.media.2021.102233. URL: https://www.sciencedirect.com/science/article/pii/S1361841521002784 (visited on 11/14/2023).

[61] Yang Li et al. "Multimodal hyper-connectivity of functional networks using functionally-weighted LASSO for MCI classification". In: *Medical Image Analysis* 52 (Feb. 2019), pp. 80–96. ISSN: 1361-8423. DOI: 10.1016/j.media.2018.11.006.

[62] Raphaël Liégeois et al. "Resting brain dynamics at different timescales capture distinct aspects of human behavior". In: *Nature Communications* 10.1 (May 24, 2019). Number: 1 Publisher: Nature Publishing Group, p. 2317. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10317-7. URL: https://www.nature.com/articles/s41467-019-10317-7 (visited on 11/30/2023).

[63] Lingwen Liu et al. "BrainTGL: A dynamic graph representation learning model for brain network analysis". In: *Computers in Biology and Medicine* 153 (Feb. 2023), p. 106521. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2022.106521. URL: https://linkinghub.elsevier.com/retrieve/pii/S001048252201229X (visited on 11/27/2023).

[64] N. K. Logothetis et al. "Neurophysiological investigation of the basis of the fMRI signal". In: *Nature* 412.6843 (July 12, 2001), pp. 150–157. ISSN: 0028-0836. DOI: 10.1038/35084005.

[65] Abdelhak Mahmoudi et al. "Multivoxel Pattern Analysis for fMRI Data: A Review". In: *Computational and Mathematical Methods in Medicine* 2012 (2012). Publisher: Hindawi Limited. DOI: 10.1155/2012/961257. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3529504/ (visited on 04/09/2024).

[66] Saima Malik-Moraleda et al. "An investigation across 45 languages and 12 language families reveals a universal language network". In: *Nature Neuroscience* 25.8 (Aug. 2022). Publisher: Nature Publishing Group, pp. 1014–1019. ISSN: 1546-1726. DOI: 10.1038/s41593-022-01114-5. URL: https://www.nature.com/articles/s41593-022-01114-5 (visited on 04/10/2024).

[67] F. Melgani and L. Bruzzone. "Classification of hyperspectral remote sensing images with support vector machines". In: *IEEE Transactions on Geoscience and Remote Sensing* 42.8 (Aug. 2004). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 1778–1790. ISSN: 1558-0644. DOI: 10.1109/TGRS.2004.831865. URL: https://ieeexplore.ieee.org/document/1323134 (visited on 04/10/2024).

[68] Patrick Mineault, Carsen Stringer, and Jorge Aurelio Menendez. *hcp_task curated by Neuromatch Academy*. Publisher: Open Science Framework. June 6, 2024. URL: https://osf.io/https://osf.io/s4h8j (visited on 03/27/2024).

[69] Mirja Mittermaier, Marium M. Raza, and Joseph C. Kvedar. "Bias in AI-based models for medical applications: challenges and mitigation strategies". In: *npj Digital Medicine* 6.1 (June 14, 2023), 113, s41746–023–00858–z. ISSN: 2398-6352. DOI: 10.1038/s41746-023-00858-z. URL: https://www.nature.com/articles/s41746-023-00858-z (visited on 04/10/2024).

[70] Rosaleena Mohanty et al. "Rethinking Measures of Functional Connectivity via Feature Extraction". In: *Scientific Reports* 10.1 (Jan. 28, 2020). Number: 1 Publisher: Nature Publishing Group, p. 1298. ISSN: 2045-2322. DOI: 10.1038/s41598-020-57915-w. URL: https://www.nature.com/articles/s41598-020-57915-w (visited on 11/30/2023).

[71] Susumu Mori and J. Donald Tournier. *Introduction to diffusion tensor imaging: And higher order models: Second edition*. Elsevier Inc., Sept. 2013. ISBN: 978-0-12-398398-5. DOI: 10.1016/C2011-0-07607-X. URL: https://pure.johnshopkins.edu/en/publications/introduction-to-diffusion-tensor-imaging-and-higher-order-models--3 (visited on 03/20/2024).

[72] Ernst Niedermeyer and F. H. Lopes da Silva. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Google-Books-ID: tndqYGPHQdEC. Lippincott Williams & Wilkins, 2005. 1342 pp. ISBN: 978-0-7817-5126-1.

[73] Itir Onal Ertugrul, Mete Ozay, and Fatos T. Yarman Vural. "Encoding the local connectivity patterns of fMRI for cognitive task and state classification". In: *Brain Imaging and Behavior* 13.4 (Aug. 2019), pp. 893–904. ISSN: 1931-7557, 1931-7565. DOI: 10.1007/s11682-018-9901-5. URL: http://link.springer.com/10.1007/s11682-018-9901-5 (visited on 03/24/2024).

[74] Sarah Parisot et al. *Spectral Graph Convolutions for Population-based Disease Prediction*. June 21, 2017. DOI: 10.48550/arXiv.1703.03020. arXiv: 1703.03020[cs,stat]. URL: http://arxiv.org/abs/1703.03020 (visited on 03/24/2024).

[75] Harshit Parmar et al. "Spatiotemporal feature extraction and classification of Alzheimer's disease using deep learning 3D-CNN for fMRI data". In: *Journal of Medical Imaging* 7.5 (2020). Publisher: SPIE, p. 056001. DOI: 10.1117/1.JMI.7.5.056001. URL: https://doi.org/10.1117/1.JMI.7.5.056001.

[76] Monique Pyrrho, Leonardo Cambraia, and Viviane Vasconcelos. "Privacy and Health Practices in the Digital Age". In: *The American Journal of Bioethics* 22 (Mar. 7, 2022), pp. 1–10. DOI: 10.1080/15265161.2022.2040648.

[77] Youzhi Qu et al. "Transfer Learning to Decode Brain States Reflecting the Relationship Between Cognitive Tasks". In: Nov. 29, 2022, pp. 110–122. ISBN: 978-981-19822-1-7. DOI: 10.1007/978-981-19-8222-4_10.

[78] *Regulation - 2016/679 - EN - gdpr - EUR-Lex*. Doc ID: 32016R0679 Doc Sector: 3 Doc Title: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) Doc Type: R Usr_lan: en. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (visited on 04/10/2024).

[79] Atif Riaz et al. "DeepFMRI: End-to-end deep learning for functional connectivity and classification of ADHD using fMRI". In: *Journal of Neuroscience Methods* 335 (Apr. 1, 2020), p. 108506. ISSN: 1872-678X. DOI: 10.1016/j.jneumeth.2019.108506.

[80] Gideon Rosenthal et al. "Mapping higher-order relations between brain structure and function with embedded vector representations of connectomes". In: *Nature Communications* 9.1 (June 5, 2018). Number: 1 Publisher: Nature Publishing Group, p. 2178. ISSN: 2041-1723. DOI: 10.1038/s41467-018-04614-w. URL: https://www.nature.com/articles/s41467-018-04614-w (visited on 12/13/2023).

[81] Anwar Said et al. *NeuroGraph: Benchmarks for Graph Machine Learning in Brain Connectomics*. Nov. 21, 2023. DOI: 10.48550/arXiv.2306.06202. arXiv: 2306.06202[cs, q-bio]. URL: http://arxiv.org/abs/2306.06202 (visited on 02/19/2024).

[82] Andrea Santoro et al. “Higher-order organization of multivariate time series”. In: *Nature Physics* 19.2 (Feb. 2023). Number: 2 Publisher: Nature Publishing Group, pp. 221–229. ISSN: 1745-2481. DOI: 10.1038/s41567-022-01852-0. URL: https://www.nature.com/articles/s41567-022-01852-0 (visited on 11/29/2023).

[83] Stephen M Smith et al. “Functional connectomics from resting-state fMRI”. In: *Trends in cognitive sciences* 17.12 (Dec. 2013), pp. 666–682. ISSN: 1364-6613. DOI: 10.1016/j.tics.2013.09.016. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4004765/ (visited on 03/25/2024).

[84] Alisa J. Snelbaker et al. “Wide Range Achievement Test 3 (wrat3)”. In: *Understanding Psychological Assessment*. Ed. by William I. Dorfman and Michel Hersen. Boston, MA: Springer US, 2001, pp. 259–274. ISBN: 978-1-4615-1185-4. DOI: 10.1007/978-1-4615-1185-4_13. URL: https://doi.org/10.1007/978-1-4615-1185-4_13 (visited on 04/12/2024).

[85] Olaf Sporns. “Network attributes for segregation and integration in the human brain”. In: *Current Opinion in Neurobiology* 23.2 (Apr. 2013), pp. 162–171. ISSN: 1873-6882. DOI: 10.1016/j.conb.2012.11.015.

[86] Olaf Sporns. “The human connectome: a complex network”. In: *Annals of the New York Academy of Sciences* 1224 (Apr. 2011), pp. 109–125. ISSN: 1749-6632. DOI: 10.1111/j.1749-6632.2010.05888.x.

[87] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 1533-7928. URL: http://jmlr.org/papers/v15/srivastava14a.html (visited on 04/12/2024).

[88] Enzo Tagliazucchi et al. “Dynamic BOLD functional connectivity in humans and its electrophysiological correlates”. In: *Frontiers in Human Neuroscience* 6 (Dec. 28, 2012), p. 339. ISSN: 1662-5161. DOI: 10.3389/fnhum.2012.00339. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531919/ (visited on 11/30/2023).

[89] Juan Terven et al. *Loss Functions and Metrics in Deep Learning*. Sept. 6, 2023. arXiv: 2307.02694[cs]. URL: http://arxiv.org/abs/2307.02694 (visited on 04/09/2024).

[90] *tf.function — TensorFlow v2.16.1*. TensorFlow. URL: https://www.tensorflow.org/api_docs/python/tf/function (visited on 04/08/2024).

[91] Ilya Tolstikhin et al. “MLP-Mixer: An all-MLP Architecture for Vision”. In: Advances in Neural Information Processing Systems. Nov. 9, 2021. URL: https://openreview.net/forum?id=EI2KOXKdnP (visited on 12/11/2023).

[92] *Use terms for HCP Young Adult - Connectome*. URL: https://www.humanconnectome.org/study/hcp-young-adult/data-use-terms (visited on 04/12/2024).

[93] Petar Veličković et al. *Graph Attention Networks*. Feb. 4, 2018. DOI: 10.48550/arXiv.1710.10903. arXiv: 1710.10903[cs,stat]. URL: http://arxiv.org/abs/1710.10903 (visited on 03/25/2024).

[94] Donglin Wang, Qiang Wu, and Don Hong. “Extracting default mode network based on graph neural network for resting state fMRI study”. In: *Frontiers in Neuroimaging* 1 (Sept. 7, 2022). Publisher: Frontiers. ISSN: 2813-1193. DOI: 10.3389/fnimg.2022.963125. URL: https://www.frontiersin.org/articles/10.3389/fnimg.2022.963125 (visited on 03/24/2024).

[95] Junqi Wang et al. “Dynamic weighted hypergraph convolutional network for brain functional connectome analysis”. In: *Medical Image Analysis* 87 (July 1, 2023), p. 102828. ISSN: 1361-8415. DOI: 10.1016/j.media.2023.102828. URL: https://www.sciencedirect.com/science/article/pii/S1361841523000889 (visited on 03/25/2024).

[96] Minjie Wang et al. *Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks*. Aug. 25, 2020. DOI: 10.48550/arXiv.1909.01315. arXiv: 1909.01315[cs,stat]. URL: http://arxiv.org/abs/1909.01315 (visited on 04/08/2024).

[97] Xiaoxiao Wang et al. “Decoding and mapping task states of the human brain via deep learning”. In: *Human Brain Mapping* 41.6 (Apr. 4, 2020). Publisher: Wiley-Blackwell, p. 1505. DOI: 10.1002/hbm.24891. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7267978/ (visited on 04/09/2024).

[98] Ze Wang et al. “Brain Entropy Mapping Using fMRI”. In: *PLOS ONE* 9.3 (Mar. 21, 2014). Publisher: Public Library of Science, e89948. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0089948. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0089948 (visited on 03/25/2024).

[99] Li Xiao et al. “Multi-Hypergraph Learning Based Brain Functional Connectivity Analysis in fMRI Data”. In: *IEEE transactions on medical imaging* 39.5 (May 2020), pp. 1746–

1758. ISSN: 0278-0062. DOI: 10.1109/TMI.2019.2957097. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7376954/ (visited on 01/25/2024).

[100] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* Feb. 22, 2019. arXiv: 1810.00826[cs,stat]. URL: http://arxiv.org/abs/1810.00826 (visited on 04/10/2024).

[101] Baoyu Yan et al. "Quantitative Identification of Major Depression Based on Resting-State Dynamic Functional Connectivity: A Machine Learning Approach". In: *Frontiers in Neuroscience* 14 (2020). ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00191. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2020.00191.

[102] Jie Yang et al. "Constructing high-order functional networks based on hypergraph for diagnosis of autism spectrum disorders". In: *Frontiers in Neuroscience* 17 (2023). ISSN: 1662-453X. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2023.1257982 (visited on 03/03/2024).

[103] Ziyuan Ye et al. "Explainable fMRI-based brain decoding via spatial temporal-pyramid graph convolutional network". In: *Human Brain Mapping* 44.7 (Feb. 28, 2023), pp. 2921–2935. ISSN: 1065-9471. DOI: 10.1002/hbm.26255. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10089104/ (visited on 01/16/2024).

[104] Yusup. *TensorFlow 1.0 vs 2.0, Part 1: Computational Graphs.* AI³ — Theory, Practice, Business. Nov. 5, 2019. URL: https://medium.com/ai%C2%B3-theory-practice-business/tensorflow-1-0-vs-2-0-part-1-computational-graphs-4bb6e31c1a0f (visited on 04/07/2024).

[105] Shijie Zhao et al. "Decoding Task Sub-type States with Group Deep Bidirectional Recurrent Neural Network". In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022*. Ed. by Linwei Wang et al. Cham: Springer Nature Switzerland, 2022, pp. 241–250. ISBN: 978-3-031-16431-6. DOI: 10.1007/978-3-031-16431-6_23.

[106] Chen Zu et al. "Identifying High Order Brain Connectome Biomarkers via Learning on Hypergraph". In: *Machine learning in medical imaging. MLMI (Workshop)* 10019 (Oct. 2016), pp. 1–9. DOI: 10.1007/978-3-319-47157-0_1. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6014614/ (visited on 03/25/2024).

Appendix

A.1 Fixed Random Parameters

The model configuration and environment were standardised to ensure reproducibility and deterministic behaviour across runs. Parameters include a fixed random seed for `sklearn`'s `train_test_split` (`42`) and TensorFlow (`tf.random.set_seed(42)`) to maintain consistency in data splitting and model initialization. The Python and TF environments were configured for deterministic operations, with `os.environ['PYTHONHASHSEED'] = '0'` to disable hash randomisation and `os.environ['TF_DETERMINISTIC_OPS'] = '1'` to eliminate nondeterministic behavior in TF, respectively. The run direction 'RL' and a batch size of 16 were used for all runs and if not stated otherwise, all subjects were used as input. The train val test split defaults to 60% train and 20% for both validation and test set.

A.2 List of Abbreviations

Abbreviation	**Term**
AD	Alzheimer's Disease
ADHD	Attention Deficit Hyperactivity Disorder
ASD	Autism Spectrum Disorder
ASL	Arterial Spinal Labeling
BOLD	Blood Oxygenation Level Dependent
CNN	Convolutional Neural Network
EMD	Empirical Mode Decomposition
EEG	Electroencephalography
FCN	Functional Connectivity Network
FCM	Functional Connectivity Matrix
fMRI	functional magnetic resonance imaging
FS	Functional System
GAT	Graph Attention Networks
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Network
GNN	Graph Neural Network
HADiB	Hypergraph Anomaly Detection in Brain
HCP	Human Connectome Project
HGNN	Hypergraph Neural Networks
HRF	Hemodynamic response function
MCI	Mild cognitive impairment
MLP	Multi Layer Perceptron
MRI	Magnetic Resonance Imaging
r-fMRI	resting state functional magnetic resonance imaging
RNN	Recurrent Neural Network
ROI	Region of Interest
SNR	Signal-to-noise ratio
ST-GCN	Spatiotemporal Graph Convolutional Network
STpGCN	Spatial Temporal-pyramid Graph Convolutional Network
SVM	Support Vector Machine
SWC	Sliding Window Correlation
t-fMRI	task functional magnetic resonance imaging
tf-gnn	Tensorflow Graph Neural Networks
TR	Repetition Time
WRAT	Wide Range Achievement Test Scores

Table A.1: List of Abbreviations used throughout the report

User Guide

B.1 Project Setup and Execution Guide

This guide provides step-by-step instructions for setting up and running the tFMRI Classification project. It encompasses everything from environment setup to running the project, including data acquisition, dependency installation, and model operations.

B.1.1 Setting Up the Required Technology

Before diving into the project, ensure that you have the necessary tools and environment to run a Jupyter Notebook, which offers an interactive interface for running code blocks and inspecting their outputs. You can use the Jupyter Notebook interface by following the installation guide available at `https://docs.jupyter.org/en/latest/install.html#install`. Alternatively, for Visual Studio Code users, the "Jupyter" extension can be installed directly within the IDE to facilitate notebook execution. Instructions for setting Jupyter up in VSCode can be found under `https://code.visualstudio.com/docs/datascience/jupyter-notebooks`.

B.1.2 Getting the Data

The project's data must be retrieved and correctly positioned within the project structure. Follow these steps based on your setup preference:

1. If running the project with the source code provided in the submission, download the dataset from `https://osf.io/s4h8j/`. Given its size (1.37GB), it must be downloaded separately. After downloading, unzip the file and place it in the root directory, ensuring the folder structure is ‘./data/hcp/hcp_task‘ to align with the predefined paths in the code.

2. Alternatively, a GitHub repository containing both the source code and data is available at `https://github.com/MagisV/HADiB-for-tFMRI-Classification`. This method simplifies the setup process by eliminating the need to download and unzip the data and possibly the need to adjust data paths manually.

B.1.3 Installing Dependencies

The project is compatible with Python version 3.9 or newer (¿= 3.9). To manage dependencies efficiently and create an isolated environment, 'virtualenv' is recommended. Note that 'Conda' is not suitable for this project due to compatibility issues with TensorFlow Graph Neural Networks ('tf-gnn'). Follow these steps to set up your environment:

1. Install 'virtualenv' via pip with the command: `pip3 install virtualenv`.
2. Navigate to the project folder and create a virtual environment with: `virtualenv venv`. To specify a Python version, use: `virtualenv -p /path/to/usr/bin/python3.9 venv`.
3. Activate the virtual environment with: `source venv/bin/activate`. To deactivate, simply run: `deactivate`.
4. Install the required Python packages with: `pip3 install -r requirements.txt`.

B.1.4 Running the Project

With the data in place and dependencies installed, you can proceed to run the project as follows:

1. Set the data directory in the code by adjusting the `HCP_DIR` constant if necessary.
2. By default, randomness is disabled to ensure reproducibility. To enable non-deterministic behaviour, comment out the respective code segment.
3. Configure central parameters such as tasks, `s_max`, batch size, and subject IDs in the `DataLoader`.
4. To train the model, execute all code cells up to the `build_model` function. Training duration varies based on the parameter combination, typically ranging from 3 to 10 minutes.
5. After setting the window sizes and compiling the model, fit the model by running the cells under "Fitting the Model".
6. Test the model's generalisability using the provided test set by executing the code under "Testing".

7. Optionally, the model can be trained on different combinations of window sizes and tasks. Additional instructions for this are provided within the project documentation.

This guide should facilitate a smooth setup and execution of the project, ensuring that all components are correctly configured and operational. For further details or troubleshooting, refer to the detailed documentation provided within the source code.

Code Extracts

C.1 Custom Message Passing

The custom implementation of the message passing layer, as described in 6.7.3.

```
class MessagePassingLayer(tf.keras.layers.Layer):

    def __init__(self, **kwargs):
        super(MessagePassingLayer, self).__init__(**kwargs)


    def build(self, input_shape):
        concat_encoding_size = input_shape[0][-1] * 2 + input_shape[1][-1] #
        ↪  The size of the concatenated node and hyperedge encodings
        node_encoding_size = input_shape[0][-1]  # The size of a single node
        ↪  encoding
        # Create the learnable weight matrix W_roi
        # Shapes:
        #  - Input: Size of the concatenated encodings (2 * node_encoding_size
        ↪  + hyperedge_encoding_size)
        #  - Output: Size of a single node encoding (as this is what is
        ↪  updated in the ROI-ROI message passing)
        self.W_roi = self.add_weight(
                    shape=(concat_encoding_size, node_encoding_size),
                    initializer='glorot_uniform',
                    trainable=True,
                    name='W_roi')
```

```
    def call(self, inputs):
        # Unpack the inputs
        node_encodings, hyperedge_encodings, incidence_matrix = inputs

        # Compute messages using the provided compute_messages function
        updated_node_encodings = compute_all_messages(node_encodings,
        ↪  hyperedge_encodings, incidence_matrix, self.W_roi)

        return updated_node_encodings
```

A function applying nested `tf.map_fn` to perform message passing using the matrices representing each hypergraph 6.7.3

```
def compute_all_messages(node_encodings, hyperedge_encodings,
↪  incidence_matrix, W_roi):
    """
        Node encodings: [Batch, Time, Node, Encoding]
        Hyperedge encodings: [Batch, Time, Hyperedge, Encoding]
        Incidence matrix: [Batch, Time, Node, Hyperedge]
    """

    def process_single_subject(args):
        """
            Node encodings: [Time, Node, Encoding]
            Hyperedge encodings: [Time, Hyperedge, Encoding]
            Incidence matrix: [Time, Node, Hyperedge]
        """
        node_encodings_b, hyperedge_encodings_b, incidence_matrix_b = args

        # The inner map_fn now processes each time step, W_roi is passed
        ↪  through each call
        updated_node_encodings = tf.map_fn( # --> Apply this to each time step
        ↪  for one element in the batch
```

```
            lambda x: compute_messages_for_single_t((x[0], x[1], x[2],
            ↪  W_roi)),
            (node_encodings_b, hyperedge_encodings_b, incidence_matrix_b),
            fn_output_signature=tf.float32
        )
        return updated_node_encodings

    # The outer map_fn iterates over batches --> Apply this to each batch
    result = tf.map_fn(
        process_single_subject,
        (node_encodings, hyperedge_encodings, incidence_matrix),
        fn_output_signature=tf.float32
    )

    return result
```

The custom implementation of a `TensorHypergraph`. Computes the Hyperedge encodings on creation, if they are not provided. Provides functions for computing neighbours of a node and the hyperedges connecting nodes. Note that `tf.logical_and` is not differentiable. The TensorHypergraph is briefly described in 6.7.3

```
class TensorHypergraph:
    def __init__(self, incidence_matrix, node_encodings=None,
    ↪  hyperedge_encodings=None):
        # Ensure incidence_matrix is a TensorFlow tensor
        self.incidence_matrix = incidence_matrix
        self.node_encodings = node_encodings
        self.hyperedge_encodings = hyperedge_encodings

    def build(self, node_features):
        """Initialises the node and hyperedge encodings."""
        # Initialise or update node_encodings as a trainable variable if it's
        ↪  None; otherwise, assign the new values
        if self.node_encodings is None:
```

```
            self.node_encodings = node_features
        else:
            self.node_encodings.assign(node_features)

        # Initialise or update hyperedge_encodings based on the current
        ↪ node_encodings
        self.update_edge_encodings()

    def update_node_encodings(self, new_node_encodings, update_edges=False):
        """Updates the node encodings, with an option to update hyperedge
        ↪ encodings."""
        self.node_encodings.assign(new_node_encodings)
        if update_edges:
            self.update_edge_encodings()

    def _compute_hyperedge_encoding(self):
        """
        Computation of hyperedge encoding by directly multiplying the
        ↪ incidence matrix
        with H_F values for each ROI and time step, then summing across ROIs
        ↪ and time steps.

        Parameters:
        - incidence_matrix: A tf.Tensor of shape [ROI, Edge] representing the
        ↪ incidence matrix.
        - H_F: A tf.Tensor of shape [Time, ROI] representing the values over
        ↪ time for each ROI.

        Returns:
        - A tf.Tensor of shape [Edge, 1] representing the encoding for each
        ↪ hyperedge.
        """
```

```
        # Multiply incidence matrix by H_F for each ROI, summing across all
        ↪  times for each edge
        product = tf.matmul(self.incidence_matrix, self.node_encodings,
        ↪  transpose_a=True, transpose_b=False)
        encodings = tf.reduce_sum(product, axis=1, keepdims=True)

        return encodings

    def update_edge_encodings(self):
        """Updates hyperedge encodings based on the current node encodings."""
        updated_hyperedge_encodings = self._compute_hyperedge_encoding()
        # If hyperedge_encodings is None, initialise it; otherwise, update it
        if self.hyperedge_encodings is None:
            self.hyperedge_encodings = updated_hyperedge_encodings
        else:
            self.hyperedge_encodings.assign(updated_hyperedge_encodings)

    def get_node_encodings(self):
        """Returns the current node encodings."""
        return self.node_encodings

    def get_hyperedge_encodings(self):
        """Returns the current hyperedge encodings."""
        return self.hyperedge_encodings

    def get_connected_nodes(self, node_index):
        # Convert the incidence matrix to a boolean tensor
        incidence_bool = tf.cast(self.incidence_matrix, tf.bool)

        # Get the hyperedges that node 'u' is a part of as a boolean tensor
        node_hyperedges = incidence_bool[node_index]
```

```
        # Expand dimensions of node_hyperedges to align with incidence_bool
        ↪ for broadcasting
        node_hyperedges = tf.reshape(node_hyperedges, [1, -1])

        # Perform a logical AND operation across the hyperedges to determine
        ↪ connections
        connected_nodes_matrix = tf.logical_and(incidence_bool,
        ↪ node_hyperedges)

        # Reduce across the hyperedges to find if a node is connected to node
        ↪ 'u' via any hyperedge
        connected_nodes_vector = tf.reduce_any(connected_nodes_matrix, axis=1)

        # Get indices of the connected nodes, excluding node 'u' itself
        connected_nodes_indices = tf.where(connected_nodes_vector)[:, 0]
        connected_nodes_indices = tf.boolean_mask(connected_nodes_indices,
        ↪ connected_nodes_indices != node_index)
        return connected_nodes_indices

    def get_hyperedges_connecting_nodes(self, u, v):
        # Convert the incidence matrix to a boolean tensor
        incidence_bool = tf.cast(self.incidence_matrix, tf.bool)

        # Get the hyperedges that node 'u' is a part of as a boolean tensor
        node_u_hyperedges = incidence_bool[u]

        # Get the hyperedges that node 'v' is a part of as a boolean tensor
        node_v_hyperedges = incidence_bool[v]

        # Perform a logical AND operation across the hyperedges to determine
        ↪ connections
        connected_hyperedges_matrix = tf.logical_and(node_u_hyperedges,
        ↪ node_v_hyperedges)
```

```
# Get indices of the connected hyperedges
connected_hyperedges_indices =
↪  tf.where(connected_hyperedges_matrix)[:, 0]

return connected_hyperedges_indices
```