Imperial College London

MEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Memo-GNN: Memory-Aware Graph Neural Networks for 4D Brain Connectivity Forecasting

Author:
Scarlet Xiao

Supervisor:
Dr. Islem Rekik

Second Marker:
Dr. Pedro Mediano

June 17, 2024

Abstract

The connection patterns of brain neural circuits form a complex and dynamic network. By observing the alteration of these networks over time, we can detect brain disorders. Recent studies have demonstrated the potential to predict the evolution of brain circuits from a single observation using graph neural networks. This capability allows for the early-stage diagnosis of brain diseases, helping to prevent them from progressing into more severe conditions such as Alzheimer's disease. However, none of the existing research has explored the potential of generating brain connectomes that are cognitively aligned to the real ones. To address this gap in literature we introduce our memory-aware graph neural network model-MemoGNN. Our main contributions consist of (i) Developing and implementing the first GNN model that is capable of forecasting 4D brain connectivity that is memory aware. (ii) Assessing the MemoGNN's performance on two distinct memory capacity tasks: the regular memory capacity task defined in the literature, and the newly introduced language-infused memory capacity task. (iii) Benchmarking MemoGNN on various GNN models and datasets. Our results demonstrate that MemoGNN models outperform others in generating brain connectivities that closely align with actual memory functions. Although the MemoGNN model trained with the regular memory capacity task did not improve the generated graph's MAE, it exhibited fewer topological errors compared to its non-memory-aware counterparts. Conversely, the MemoGNN model trained with the language-infused memory capacity task surpassed the baseline model's MAE.

Acknowledgments

First, I would like to extend my gratitude to my supervisor, Dr. Islem Rekik, for her deep insights in the area of graph neural networks. She has taught me many lessons across the board, from research skills to work ethics. Our weekly meetings were really helpful as they kept me on track and guided me through the complexities of my research. I also wish to thank my second marker, Dr. Pedro Mediano, for his helpful feedback and suggestions. Next, I would like to thank my personal tutor, Dr. Maria Valera Espina, for her regular check-ins throughout my four years at Imperial. Finally, I am very grateful to my family, friends, and partner for their support throughout this project and my journey at Imperial. It has been a rollercoaster ride, and I wouldn't have been here without each one of you.

Publications

Some of the work in this report has been submitted for publication in the following venue:

- Authors: **Scarlet Xiao** and Islem Rekik
- Title: *Memo-GNN: Memory-Aware Graph Neural Networks for 4D Brain Connectivity Forecasting*
- Venue: The MICCAI 2024 Workshop

Contents

Chapter 1

Introduction

Graph-based models offer a more effective way of understanding the brain's complexities. This is particularly the case when detecting brain disorders where the traditional image-based approaches may fall short in capturing the complex interactions and dysconnectivity [1]. A detailed mapping of the connections across the different regions of the brain can be created by converting Magnetic resonance imaging (MRI) scans into graphs [2]. Such brain graphs are invaluable for predicting changes in brain structure and function. This is an important area of research, as the early-stage diagnosis and treatment of conditions like mild cognitive impairment is required to prevent it from developing into more severe conditions like Alzheimer's disease [3].

Studies have employed Graph Neural Networks (GNNs) to solve this problem. Nebli et al. [4] developed EvoGraphNet, a model comprising a series of graph-generative adversarial networks (gGANs), each predicting the subsequent time point's brain graph. It utilises a generator and a discriminator, with the generator creating synthetic samples of the next time point and the discriminator differentiating between real and generated data. However, although EvoGraphNet is effective at predicting brain connectomes, it requires a considerable amount of computational resources. To improve computational efficiency, Tekin et al. [5] introduced the Recurrent Brain Graph Mapper (RBGM), which uses a series of recurrent graph neural networks for predicting brain graphs at various time points. This approach significantly reduced the training time compared to EvoGraphNet but with some performance trade-offs. Further expanding the scope, Demirbilek et al. [6] proposed the Recurrent Multigraph Integrator Network (ReMI-Net*), which focuses on multigraph populations and employs a recurrent graph convolution approach for predicting Connectome-Based Templates. Lastly, Gürler et al. [7] proposed the 4D-FED-GNN+ model to address the data scarcity issue in longitudinal datasets. This approach combines GNNs with federated learning to enhance data availability across different hospitals. While these models have made significant advancements in brain graph prediction, they have not yet explored the generation of brain graphs that closely resemble the cognitive function performance of actual connectomes.

Reservoir Computing (RC) has been used as a tool to integrate biological aspects into networks. By mapping the brain graph structure onto the reservoir, we can create a biologically instantiated neural network [8]. This model can be trained to perform various neuroscience tasks, which can then be used to assess the impact of neurological diseases based on performance [9]. Previous studies have evaluated memory tasks across different primate species using this RC paradigm [10]. Further work has been done to develop a toolbox that can explore the effects of network subdivisions on cognitive functions. This is done by configuring input and output nodes in RC to represent the different brain systems [11]. Building on these insights, our work aims to investigate the effects of integrating biological neural networks with GNNs for 4D brain connectivity forecasting, an area that has not yet been explored in the literature.

1.1 Objectives & Contributions

1. **Developing and implementing the first GNN model that is capable of forecasting 4D brain connectivity that is memory aware.**

   We present **MemoGNN: Memory-aware graph neural network**, a novel model for 4D brain connectivity forecasting that produces memory-aware brain graphs. By co-training a GNN with a biological reservoir network, MemoGNN generates brain graphs that better reflect actual cognitive memory functions. Our loss function minimises the difference in memory capacity between predicted and actual brain connectomes, enhancing the model's fidelity in capturing the brain's intricate dynamics.

2. **Assessing the MemoGNN's performance on two distinct memory capacity tasks: the regular memory capacity task defined in the literature, and the newly introduced language-infused memory capacity task.**

   In the memory capacity task introduced by Jaeger [12], the training signals received by the reservoir are randomly chosen values from a uniform distribution within the range [-0.5, 0.5]. We propose a new language-infused memory capacity task, where the values used are derived from word embeddings of texts. This approach aims to more closely align with the way the human brain processes and stores information, potentially enhancing the reservoir's capability to simulate memory capacity and thus, improve the MemoGNN's ability to predict brain activity patterns more accurately.

3. **Conducting extensive experiments on MemoGNN using various base GNN models and datasets.**

   To validate the MemoGNN model, we applied it to different types of graph neural networks: Chebyshev Graph Convolutional Gated Recurrent and Long Short-Term Unit [13], Topology Adaptive Graph Convolutional Networks [14], and the Recurrent Brain Graph Mapper [5]. We evaluated performance using mean absolute error (MAE) and various network topology measures on the

OASIS-2 and simulated datasets. Our evaluation indicates that the MemoGNN trained with the regular memory capacity task exhibited fewer errors in certain topological aspects compared to non-memory-aware counterparts. This finding aligns with our preliminary experiment, which showed a strong correlation between the regular memory capacity and some network topologies. However, it did not achieve a lower MAE than the baseline models. On the other hand, MemoGNN trained with the language-infused memory capacity task achieved a lower MAE than the baseline models, demonstrating that the language-infused task could be more biologically representative.

1.2 Ethical Considerations

Ethical handling of personal data is important in this project. While the datasets used are publicly available and include MRI scans with assumed participant consent, we ensured that all additional participant data does not allow for the identification of the individual, providing full anonymity. The citations of the datasets are done to allow for the reproducibility of the project's methodology.

Other than the usage of personal data, the development and use of automated prediction tools like our project's models come with significant ethical considerations. Given the sensitive nature of medical applications, where inaccuracies could have detrimental effects on patients, it's crucial to recognise that our models are experimental. The direct application of the model in clinical treatment without comprehensive validation and regulatory compliance is not recommended.

Furthermore, the environmental impact of machine learning is acknowledged. The usage of GPU to train models for a long period requires high computational and energy demands. This aspect should be taken into consideration during the execution of the project as it contributes to climate change.

1.3 Preliminary Experiments

This section details some preliminary experiments we conducted to explore the nature of the **regular memory capacity**. The concept and mathematical formulation of memory capacity are explained in Section 3.2.

1.3.1 Memory Capacity of Two Groups

In this experiment, we compared the memory capacity of two distinct subject groups to determine if there is a significant difference between their average memory capacities. The following hypotheses were tested:

- **(H0)** The average memory capacity of the two populations is not significantly different.

- **(H1)** The average memory capacity of the two populations is significantly different.

Evaluation Dataset

We used the Connectomics in NeuroImaging Transfer Learning Challenge (CNI-TLC) dataset [15] for this experiment. This dataset includes 340 individuals, divided equally into 170 neurotypical controls and 170 individuals with Attention-Deficit/ Hyperactivity Disorder (ADHD), aged between 8 and 12 years. Each participant underwent magnetic resonance imaging (MRI) to acquire resting-state fMRI (rs-fMRI) time series data. Prior to analysis, brain graphs were constructed for each scan using the method described in Section 2.1, and the Harvard Oxford parcellation was employed to identify 112 regions of interest.

Experimental Set-Up

We used the `ESNRegressor` from the `echoes` library as our Echo State Network (ESN) for this experiment. The concepts of ESN are explained in Section 2.3.1. We chose to use this library as it allows us to instantiate the W_{res} of the network, works well with `numpy`, and is computationally efficient. To determine the memory capacity of each biologically instantiated network, we used 4000 time steps for training and 1000 time steps for testing.

Hyperparameter tuning was performed using grid search to find the optimal values to achieve the best memory capacity for the ESN model. They were then fixed across all networks. The hyperparameters explored were: spectral radius of the reservoir connectivity matrix $p = \{0.8, 0.9, 1.0\}$, input scaling $e = \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$, and leakage rate $\alpha = \{0.6, 0.8, 1.0\}$. The optimal combination of hyperparameters we found is as follows: spectral radius $= 0.9$, input scaling $= 10^{-8}$, and leakage rate $= 1.0$.

After fixing the hyperparameters, we calculated the memory capacity for each biologically instantiated network. This is done by changing the W_{res} to each subject's connectivity matrix from the dataset, using the same training data to train the network and the testing data for the computation of memory capacity.

Results and Analysis

The average of the memory capacity across the two populations as well as the standard deviation is calculated. We then plotted a distribution graph shown in Figure 1.1 which displays the memory capacities distribution of the two groups. Lastly, we carried out a t-test on the results and got a p-value of 0.003. Therefore, we can reject H0 and accept H1. This implies that there is a statistically significant difference in the average memory capacities between the two groups. Observation can be made from the figure that the memory capacity of individuals with ADHD is lower than that of neurotypical controls.

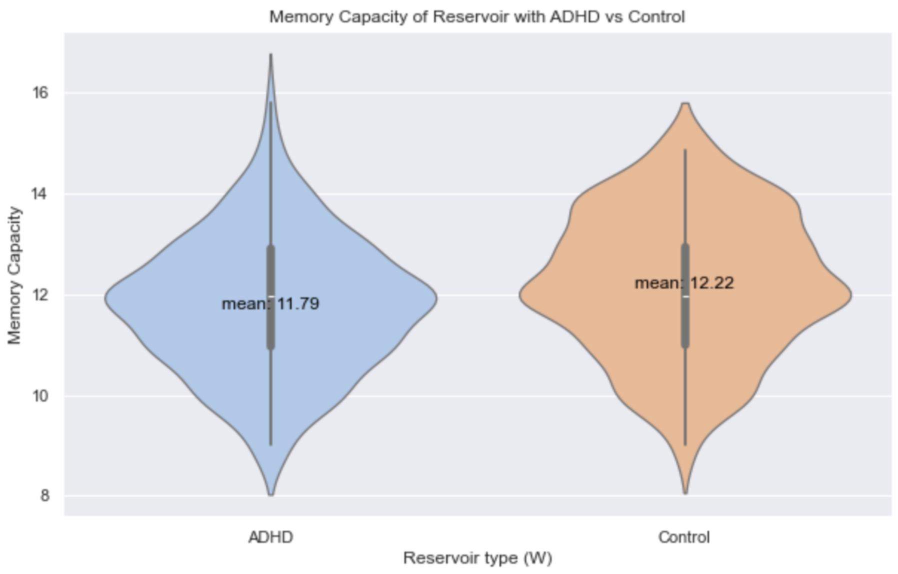


Figure 1.1: Violin plots comparing the memory capacity between the ADHD and Control groups.

The significant difference in memory capacity observed can be attributed to the variations in brain network connectivity characteristic of ADHD. This demonstrates that the biologically instantiated reservoir could potentially capture individual cognitive differences.

1.3.2 Memory Capacity and Network Topologies

We explore the correlation between memory capacity and various network topologies in this experiment.

Evaluation Dataset

The evaluation dataset contains network graphs with topological traits across the spectrum. This is such that we can thoroughly study the relationship between the different network topologies and the memory capacity.

This dataset includes real brain graphs from the OASIS-2 dataset, which will be covered in Section 4.2. We incorporated all the brain graphs from each subject across multiple time points.

To extend the dataset, two techniques were used - rewiring, and edge deletion to generate disrupted brain graphs. A visualisation of these disrupted graphs is shown in Figure 1.2.

To introduce noise by rewiring the graph, we perform random edge weight swaps for a set amount of times. The more swaps carried out, the more disrupted the network is. For each brain graph in the OASIS-2 dataset, we disrupt graphs by rewiring 100, 200, and 300 edges and we add these graphs to the datasets.

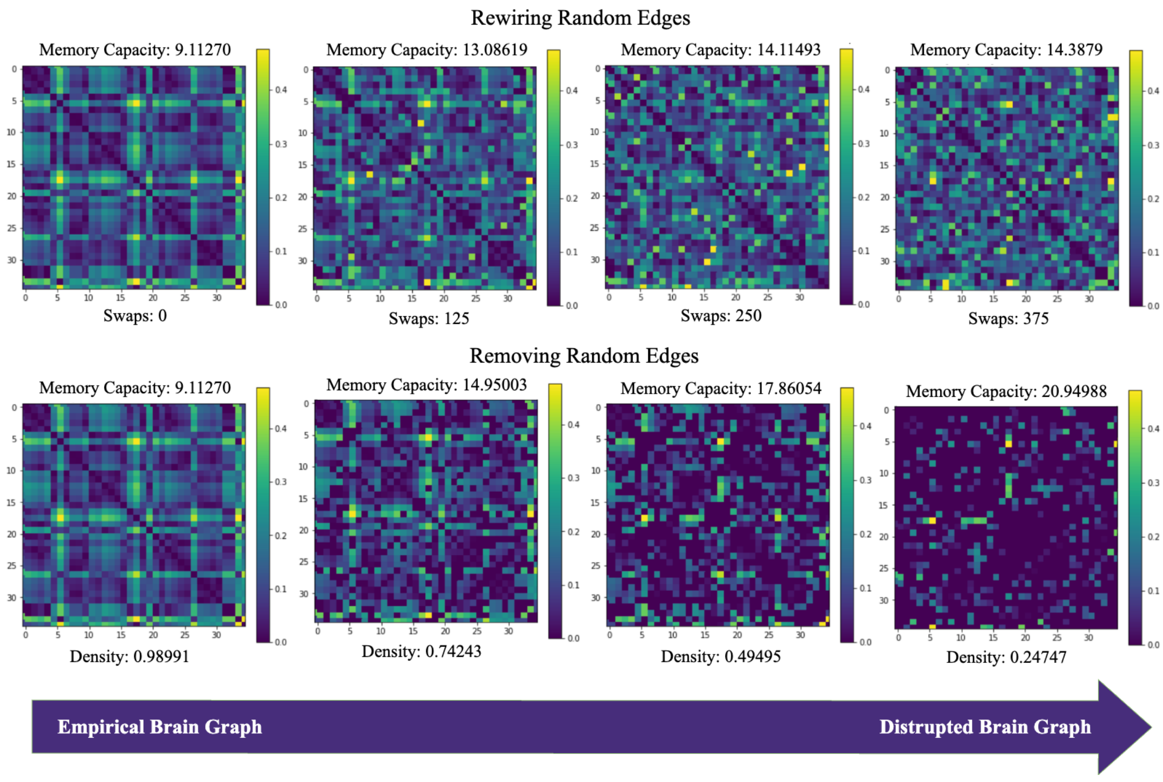


Figure 1.2: *Visualisation of brain graphs with rewiring and edge deletion disruptions*. The top row shows the effects of rewiring random edges with varying numbers of swaps, while the bottom row illustrates the impact of removing random edges to achieve different densities. Each subplot includes the memory capacity value and the respective disruption parameter (number of swaps or density).

To adjust the density of the graph G by removing edges, we first calculate the maximum possible number of edges in a complete graph with N nodes:

$$E_{\text{max}} = \frac{N(N-1)}{2}$$

Next, determine the target number of edges E_t needed to achieve the desired density d:

$$E_t = d \cdot E_{\text{max}} = d \cdot \frac{N(N-1)}{2}$$

Compare the current number of edges E_c in G with the target number E_t. If E_c exceeds E_t, randomly select and remove edges until the number of edges matches E_t. For each original brain graph, we created disrupted graphs by removing random edges to achieve densities $d = \{0.98, 0.79, 0.60, 0.41, 0.22, 0.1\}$ and added them to the dataset.

Additionally, the dataset includes a variety of synthetically generated graphs using multiple well-established models, such as Erdős-Rényi, Barabási-Albert, and modular networks. A visualisation of networks generated with different models and parameters is shown in Figure 1.3.

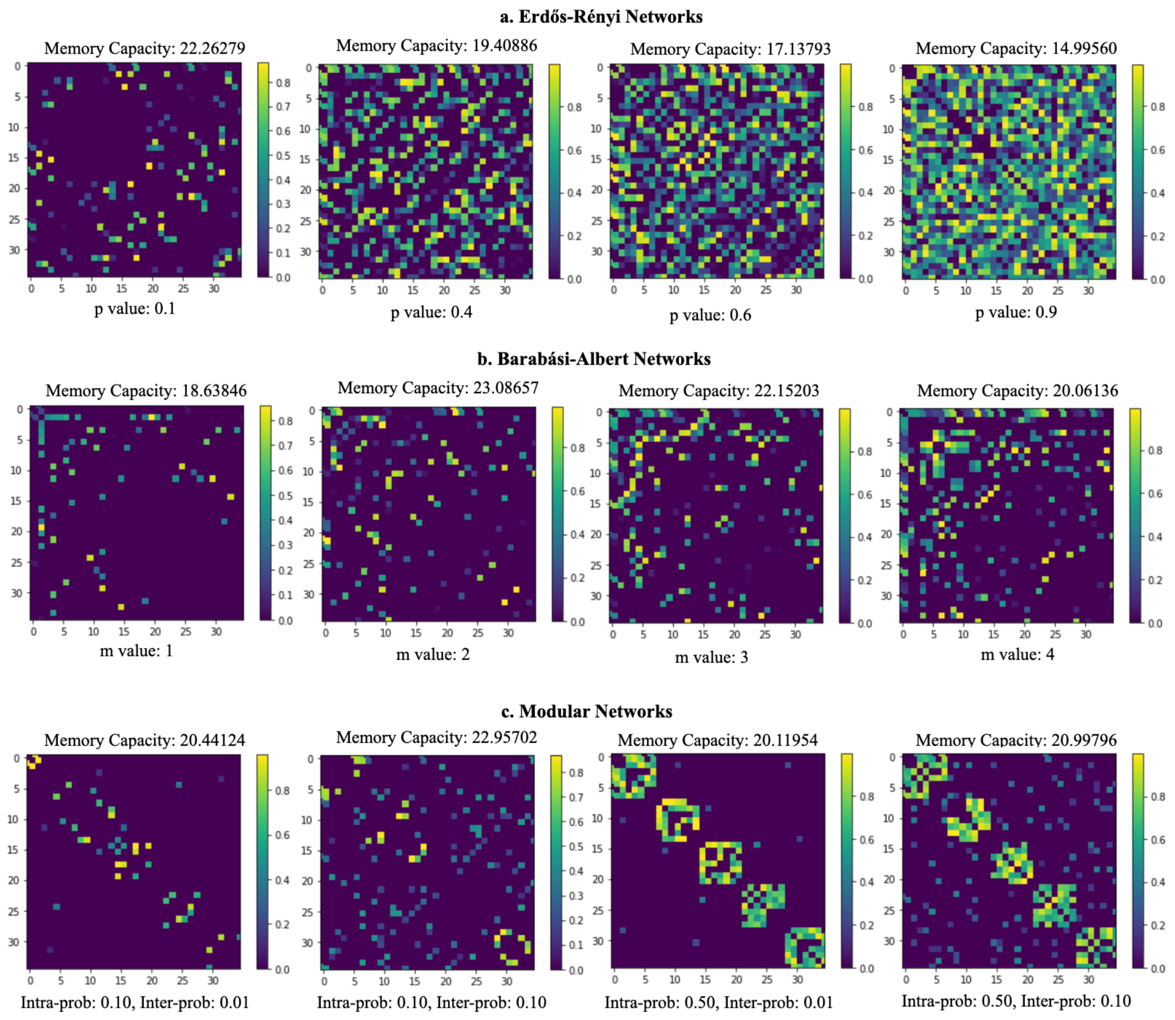


Figure 1.3: *Examples of synthetically generated network graphs using different models and parameters.* (a) Erdős-Rényi networks with varying probabilities p for edge creation. (b) Barabási-Albert networks with different m values. (c) Modular networks with varying intra-community and inter-community connection probabilities. Each subplot includes the memory capacity value and the respective model parameter.

In an Erdős-Rényi (ER) graph $G(n, p)$, n nodes are connected by edges, with each pair of nodes having an independent probability p of being connected. We generated 300 ER graphs for each $p = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$.

In a Barabási-Albert (BA) graph, nodes are added sequentially, and each new node is connected to m existing nodes with a probability proportional to the number of links that the existing nodes already have, a process known as preferential attachment. We generated 200 BA graphs for m values ranging from 1 to 15.

In modular networks, there are distinct communities where nodes within the same community are more likely to be connected than nodes in different communities. We divided the total number of nodes, N, into C communities, each containing $\frac{N}{C}$ nodes. Within each community, edges are created with a high probability p_{intra}, and between

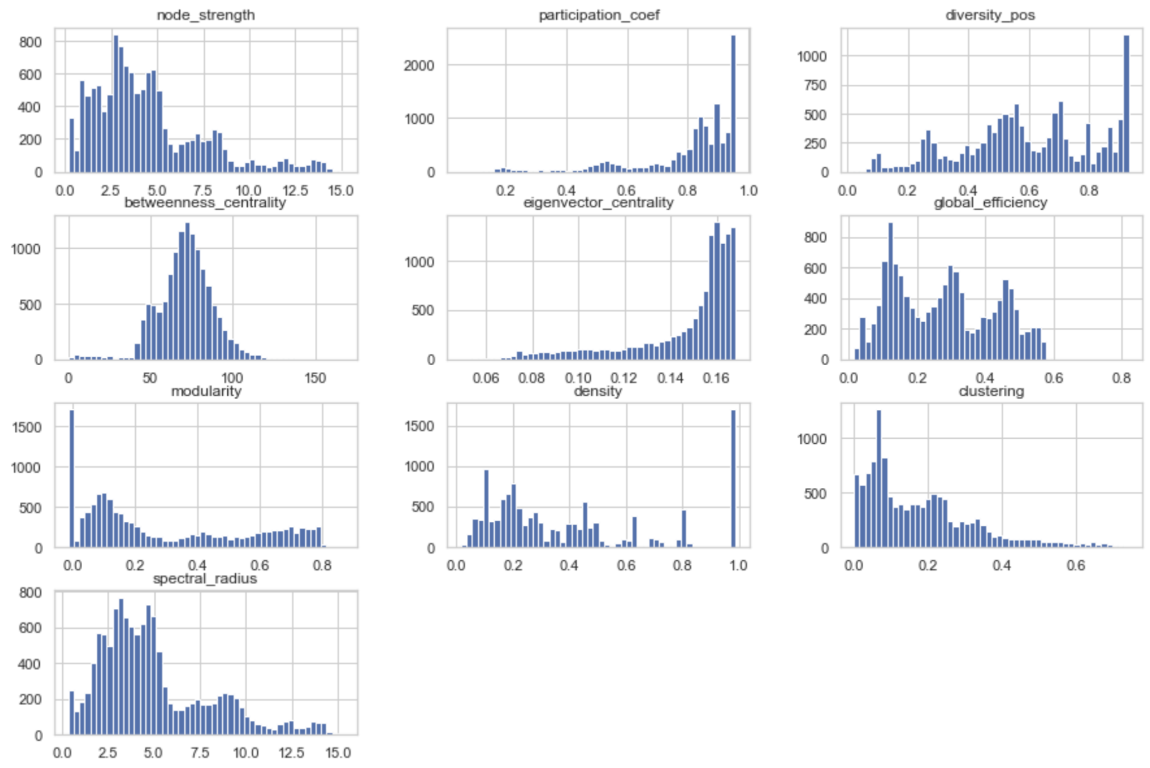


Figure 1.4: *Distribution of various network topologies across the evaluation dataset*. The histograms illustrate the range and frequency of values for node strength, participation coefficient, diversity, betweenness centrality, eigenvector centrality, global efficiency, modularity, density, clustering coefficient, and spectral radius.

communities with a low probability p_{inter}. We generated 150 modular networks with varying $p_{\text{intra}} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $p_{\text{inter}} = \{0.01, 0.0325, 0.055, 0.0775, 0.1\}$.

Experimental Setup

The experimental setup explores the predictive relationship between network topologies and memory capacity. Initially, a suite of topology measurements is computed for each network graph in the dataset. These include node strength, participation coefficient, diversity coefficient, betweenness centrality, eigenvector centrality, density, clustering coefficient, and spectral radius. Figure 1.4 displays the distribution of network topologies in the evaluation datasets. Networkx and bctpy libraries are used to generate the graphs and compute the topologies.

Next, we compute the memory capacity of all the networks. We used the same set of training and testing data, as well as hyperparameters as the first experiment for the reservoir. Now that the network topologies and memory capacity are calculated, a supervised learning model which aims to predict memory capacity based on the network characteristics is set up. Linear Regression model from sklearn was used to carry out this task.

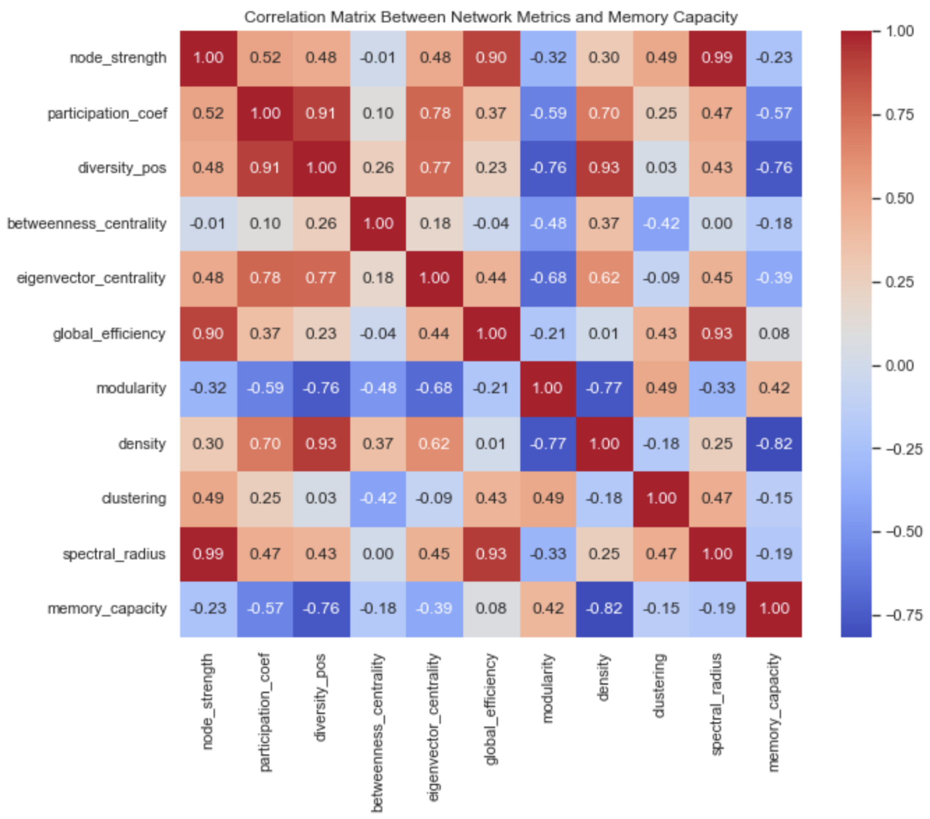


Figure 1.5: *Correlation matrix illustrating the relationships between various network topologies and memory capacity.* This visualisation helps identify which topological features are most strongly associated with memory capacity, providing insights into the structural properties that influence cognitive function.

Results and Analysis

The results are analysed to assess the predictive capability of the supervised learning models in estimating memory capacity based on network topologies. The primary performance metric for this evaluation is the coefficient of determination (R^2). The high R^2 value obtained in this study ($R^2 = 0.8937$) demonstrates that the network topology measurements can effectively predict memory capacity, suggesting a strong correlation between memory capacity and these network characteristics. To support this conclusion, we further present a correlation heat map and the topological distribution of the dataset. The heat map illustrates the relationships between various network measurements and memory capacity. It shows that there are significant correlations between them. In particular, measurements such as participation coefficient, diversity coefficient, and density show stronger correlations with memory capacity. Out of all of them, density has the highest correlation of -0.82. The bar chart in Figure 1 provides a clearer comparison of the correlation values. Overall, these findings allow us to gain insights into the impact of generating memory-aware brain connectomes on network topologies.

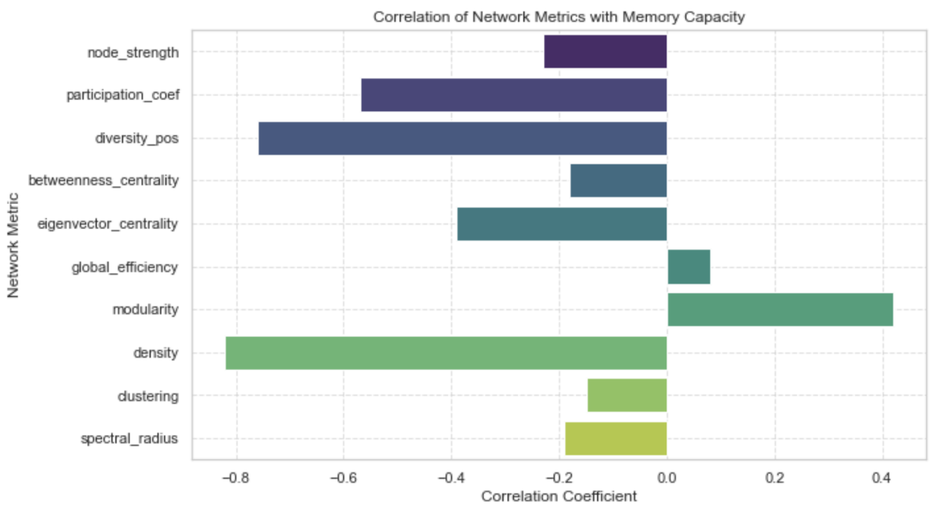


Figure 1.6: *Bar chart displaying the correlation coefficients between various network topology metrics and memory capacity*. This visualisation highlights the strength and direction of the relationship between each network metric and memory capacity.

Chapter 2

Background

2.1 Brain Graph Construction

In neuroimaging, Magnetic Resonance Imaging (MRI) scans can be converted into graph structures, using adjacency matrices for representation [16]. This process begins by defining nodes as Regions of Interest (ROIs) based on a chosen parcellation method [17]. The approach to constructing a graph from MRI data varies with the type of MRI:

- **Functional MRI (fMRI)**: fMRI measures changes in blood flow, particularly the blood-oxygen-level dependent (BOLD) signal, which correlates with neural activity. When brain regions become active, they consume more oxygen, leading to increased blood flow and higher levels of oxygenated haemoglobin [18]. For each ROI, the average BOLD signal over time is calculated to represent neural activity within that region. A functional connectivity matrix is then created by computing correlations or other statistical measures between these time-series data from the ROIs. This process is shown in Figure 2.1.

- **Structural MRI**: This type of MRI provides detailed visualisations of brain tissues, including grey matter, white matter, and cerebrospinal fluid, capturing the physical shape, and size of these structures [19]. Structural connectivity can be analysed through direct diffusion-based anatomical connections (using techniques like diffusion tensor imaging) or through indirect, morphology-based statistical interdependencies across different populations [20].

2.2 Neural Networks for Sequential Data

2.2.1 Recurrent Neural Network

Neural Networks are machine learning methods that aim to learn the underlying patterns in data. Their structure is inspired by the human brain, where they are made up of neurons interconnected with each other. Each neuron contains these

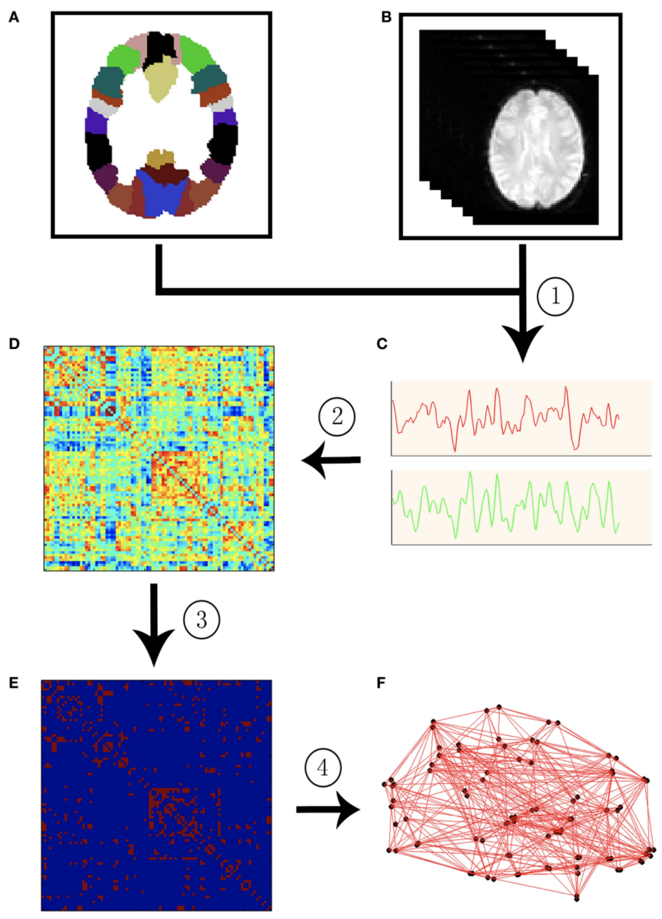


Figure 2.1: Flowchart for the conversion of R-fMRI to a brain graph representation taken from [17]

three main components:

1. Features (x): These are the data points fed into each neuron.
2. Parameters (w, b): These are learnable and adjusted during training to improve performance.
3. Activation Function (g): This function introduces non-linearity, allowing the neurons to handle the complex patterns in the data.

Mathematically, a single neuron's operation can be described by the formula: $\hat{y} = g(\sum_{i=0}^{m} x_i w_i + b)$, where $\hat{y}$ represents the neuron's output. This formula is used in feedforward neural networks, where each input is viewed independently, meaning it cannot retain past information.

To address this limitation, Recurrent Neural Networks (RNNs) incorporate a looping structure in their design [21], illustrated in Figure 2.2. This architecture endows RNNs with a memory-like capability by including dependency between the input data. This means that RNNs can be used to process time series data. Specifically, RNNs maintain a state h_t, updated based on the previous time step's state h_{t-1} and

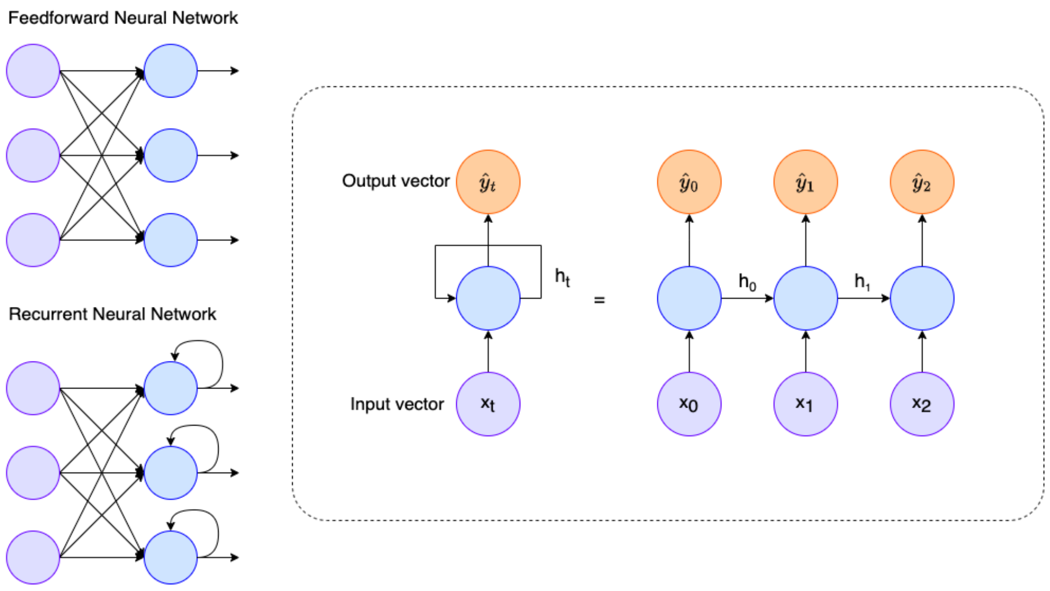


Figure 2.2: Comparison of a feedforward neural network to a recurrent neural network with the looping structure

the current time step's input x_t. This calculation is mathematically expressed as $h_t = g_h(W_h h_{t-1} + W_x x_t + b_h)$. The computed hidden state is then used to determine the output $\hat{y}$ (via $\hat{y} = g_y(W_y h_t + b_y)$), with all weights being shared across time steps.

To update the weights to minimise the difference between the predicted data and the actual data, most RNNs use backpropagation through time (BPTT). Unlike the standard backpropagation used in feedforward networks, in BPTT, the error is propagated back through each time step, and shared weights are updated based on the accumulated gradients from the entire sequence. However, this method can trigger the vanishing gradient problem, where the continuous multiplication of the hidden state weight matrix across time steps leads to gradients that either explode or significantly diminish [22]. Another effect of this repeated multiplication is that updating a single parameter in the network requires significant computational effort [23], leading to an extended period of training time.

Researchers have employed two main strategies to address the vanishing gradient problem. First, they have tried to employ much more advanced learning algorithms beyond simple stochastic gradient descent. One such technique is gradient clipping, which sets a threshold value for gradients. When a gradient exceeds this threshold, it is scaled down to prevent it from becoming too large or too small. This approach helps stabilise training by preventing exploding gradients, thereby indirectly addressing the vanishing gradient issue. Second, researchers have implemented more sophisticated activation functions. Instead of the traditional affine transformation followed by a simple element-wise nonlinearity, they use gating units. These units include the Long Short-Term Memory (LSTM) unit and the Gated Recurrent Unit (GRU), both of which incorporate mechanisms to better manage long-term dependencies and mitigate the vanishing gradient problem.

2.2.2 Long Short-Term Memory

The Long Short-Term Memory (LSTM) network [24] is a specialised form of Recurrent Neural Network (RNN) designed to handle long-term dependencies in sequential data, effectively addressing the vanishing gradient problem. An LSTM unit consists of a cell state, an input gate, a forget gate, and an output gate. These components interact to regulate the flow of information through the unit.

- **Forget Gate**: The forget gate determines which part of the cell state to forget. It is defined as:
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
- **Input Gate**: The input gate controls how much of the new information is stored in the cell state. It has two parts:
  - *Input Gate Activation*:
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
  - *Candidate Cell State*:
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
- **Cell State Update**: The cell state is updated by combining the old state, partially forgotten by the forget gate, and the new candidate values, modulated by the input gate:
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$
- **Output Gate**: The output gate determines the next hidden state, which is used for both the current output and the next time step's input:
  - *Output Gate Activation*:
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
  - *Hidden State*:
$$h_t = o_t \cdot \tanh(C_t)$$

In these equations, σ represents the sigmoid function, and $\tanh$ represents the hyperbolic tangent function. The variables W and b are the weight matrices and biases, respectively. The subscripts f, i, C, and o refer to the forget gate, input gate, candidate cell state, and output gate, respectively. h_{t-1} is the previous hidden state, x_t is the current input, and C_{t-1} and C_t are the previous and current cell states. The output y_t can be produced by a one layer neural network similar to the simple RNN case: $\hat{y} = g_y(W_y h_t + b_y)$.

2.2.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) [25] is designed to maintain and utilise long-term dependencies in sequential data with a simpler architecture than Long Short-Term Memory (LSTM) networks. A GRU unit consists of an update gate and a reset gate, which regulate the flow of information.

- **Update Gate**: The update gate determines how much of the past information needs to be passed along to the future. It is defined as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

- **Reset Gate**: The reset gate decides how much of the past information to forget. It is defined as:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

- **Candidate Hidden State**: The candidate hidden state, which represents new information to be added to the unit, is calculated as:

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

- **Hidden State Update**: The final hidden state is a combination of the previous hidden state and the candidate hidden state, modulated by the update gate:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$

2.3 Reservoir Computing

2.3.1 Echo State Networks

Echo State Networks (ESN) [26] is a new approach that modifies the RNN architecture to address its shortcomings [27], it comes under the reservoir computing umbrella. It consists of the input, reservoir, and readout layers [8]. The input layer receives a signal and passes it to the reservoir. The reservoir is an RNN with fixed untrained weights that can map the input into a higher-dimensional space. This step is crucial as it transforms the complex, non-linearly separable data into a more manageable, linearly separable form. The readout layer is a straightforward linear model that transforms the reservoir states into the target signal. This layer is typically trained using basic methods like linear regression. By focusing on training just the readout layer, RC avoids the issues usually encountered with gradient descent in standard RNN training [27].

Mathematically, the input vector $x(t) \in \mathbb{R}^{N_x}$ is processed through an input matrix $W_{in} \in \mathbb{R}^{N_r \times N_x}$, where N_r and N_x denote the reservoir size and input size, respectively. An input scaling factor can be applied to calibrate the magnitude of the input signals in relation to the reservoir's internal dynamics. The reservoir's state is updated as per the following equations:

$$h'(t) = f(W_{in}x_t + W_r h(t-1) + b) \tag{2.1}$$

$$h(t) = \alpha h'(t) + (1-\alpha)W_r h(t-1) \tag{2.2}$$

where $W_r \in \mathbb{R}^{N_r \times N_r}$ represents the connectivity matrix between reservoir neurons, $b \in \mathbb{R}^{N_r}$ is the bias vector, f is the activation function and α is the leak rate. The leak rate α governs the extent to which the neuron's new state permeates into its state at each time step. A higher leak rate, approaching 1, implies a more aggressive state update based on new inputs, enhancing the network's responsiveness to recent inputs but potentially diminishing its long-term information retention capacity. Additionally, the spectral radius is used to scale the weights W_r, influencing the balance between the network's memory and stability.

The output readout vector $y(t) \in \mathbb{R}^{N_y}$ is calculated as follows:

$$y(t) = W_{out}h(t) \tag{2.3}$$

As mentioned before, training is confined to the readout weights W_{out}, which can be efficiently determined using rapid linear regression models. Assuming the internal states and desired outputs are stored in matrices X and Y, the readout weights W_{out} are computed by solving a least squares problem:

$$||W_{out}X - Y||_2^2 \tag{2.4}$$

2.4 Machine Learning on Graphs

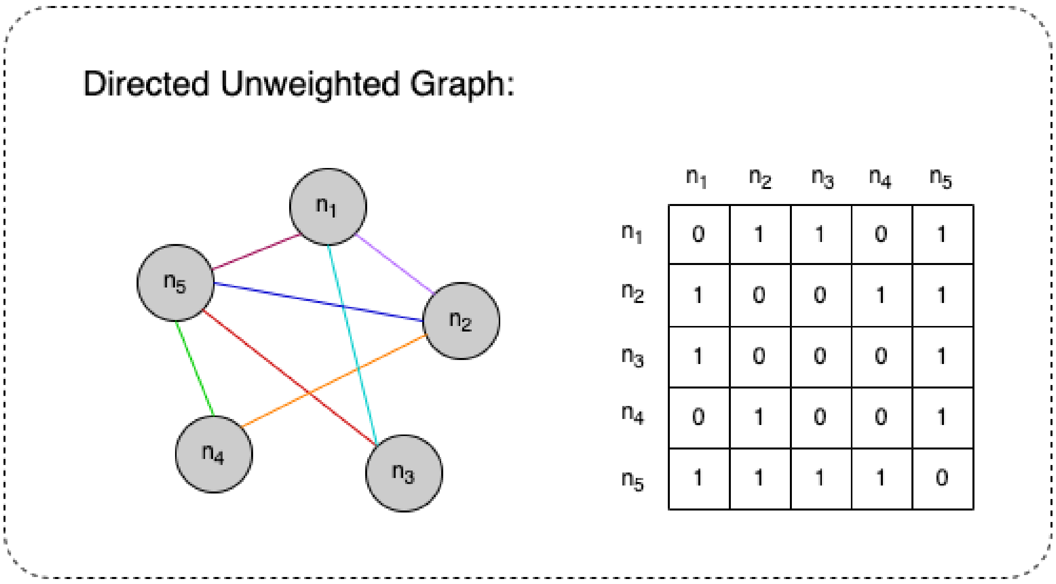
2.4.1 Graphs

A graph G is defined as an ordered pair $G=(V, E)$ consisting of a set V of vertices (or nodes) and a set E of edges. Each edge is a pair (x, y) where x and y are vertices. An edge represents a connection or relation between the vertices it connects with attributes like weights or directions. Graphs can also be represented using an adjacency matrix, in which the matrix elements indicate whether there is an edge between the pairs of vertices in the graph. An example of this is shown in Figure 2.3, where a directed unweighted graph's adjacency matrix would be asymmetric.

Graph structures are widely used to model complex relationships across various domains, including areas like text and image analysis. This utility has inspired the creation of machine learning architectures that capitalise on the unique properties of graphs. In our research, we specifically focus on leveraging these graph-based machine learning architectures to work with the brain graph data.

2.4.2 Graph Neural Networks

Graph Neural Networks (GNNs) are specialised neural networks designed to process structured data represented as graphs. In GNNs, both nodes and edges have associated feature vectors/learned embeddings that contain specific information used



	n_1	n_2	n_3	n_4	n_5
n_1	0	1	1	0	1
n_2	1	0	0	1	1
n_3	1	0	0	0	1
n_4	0	1	0	0	1
n_5	1	1	1	1	0

Figure 2.3: A directed unweighted graph and its asymmetric adjacency matrix

for learning purposes. The objective of a GNN is to learn a mapping function that transforms these input feature vectors into output feature vectors.

GNNs can solve various tasks and they are categorised into three levels:

- **Node-level predictions** focus on forecasting specific node attributes or classifications. An example is predicting the changes in the state or attributes of individual nodes in the brain graph (like changes in the activity or function of specific brain regions).

- **Edge-level predictions** aim to determine the existence of a link between two nodes, like predicting the formation or dissolution of connections (edges) between different regions or nodes (neurons, neuron clusters, or functional areas) in the brain.

- **Graph-level predictions** involves classifying an entire graph or predicting its overall attributes, such as classifying the brain graph into different categories (e.g., healthy vs. diseased).

When constructing a typical GNN model, different types of computational blocks are combined based on the task at hand. There are three main types [28]:

1. **Propagation Block**: This block facilitates the transfer of information between nodes, enabling the aggregation of data that reflects both the features and the topology of the graph. Propagation typically involves convolution operators to gather neighbouring information. This is somewhat similar to convolutional layers in Convolutional Neural Networks (CNNs), where a kernel or filter slides across an image to create a feature map (Figure 2.4). However, the convolutional operations differ between GNNs and CNNs.

2. **Sampling Block**: In scenarios involving large graphs, sampling blocks become

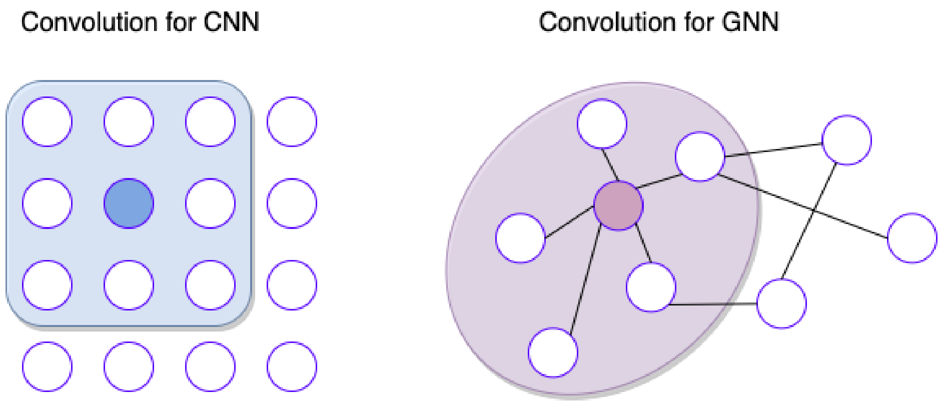


Figure 2.4: Convolution operation in CNN vs in GNN

essential. They enable efficient propagation across these extensive networks and are often used in conjunction with the propagation module.

3. **Pooling Block**: When it's necessary to derive representations for higher-level subgraphs or entire graphs, pooling modules come into play. They are crucial for extracting comprehensive information from nodes.

In particular, for the propagation block, the convolutional operators have been categorised into two domains, spatial and spectral.

Spectral Methods

Spectral Methods rely on the principles of graph spectral theory [29]. The basis of this approach is the graph Laplacian L, which is defined as $L = D - A \in \mathbb{R}^{n \times n}$. Here, n is the number of vertices in the graph, A represents the adjacency matrix of the graph, and D is the degree matrix, a diagonal matrix where D_{ii} indicates the degree of node i. The central part of spectral methods is the eigen-decomposition of the graph Laplacian, expressed as $L = U\Lambda U^T$, where $U \in \mathbb{R}^{n \times n}$ is the matrix of eigenvectors and $\Lambda \in \mathbb{R}^{n \times n}$ is the diagonal matrix containing eigenvalues. In graph-based data, defining a meaningful translation operator in the vertex domain presents challenges; hence, the convolution operation $*_\mathcal{G}$ is performed in the Fourier domain. A graph signal $x \in \mathbb{R}^{n \times d_x}$ is filtered using a non-parametric kernel defined by $g_\theta(\Lambda) = \text{diag}(\theta)$, where $\theta \in \mathbb{R}^n$ represents the vector of Fourier coefficients. The convolution is expressed as:

$$y = g_\theta *_\mathcal{G} x = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x \in \mathbb{R}^{n \times d_x}, \qquad (2.5)$$

However, non-parametric filters have two main limitations: first, their global scope conflicts with the local nature of traditional convolution, which should ideally focus only on neighbouring nodes; second, evaluating the formula is computationally expensive due to the multiplication of U, especially for large graphs, resulting in a learning complexity of $O(n)$. These challenges are mitigated by using a polynomial filter [30], where g_θ is parameterised as a truncated series expansion of Chebyshev polynomials T_k, up to order $K-1$:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tag{2.6}$$

Here, $\theta \in \mathbb{R}^K$ are the Chebyshev coefficients, and the scaled Laplacian is $\tilde{\Lambda} = 2\Lambda/\lambda_{\text{max}} - I_n$. The graph filtering operation is:

$$y = g_\theta *_{\mathcal{G}} x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x, \tag{2.7}$$

with $\tilde{L} = 2L/\lambda_{\text{max}} - I_n$. This method addresses the first limitation by focusing on nodes within K hops from the central node, thus ensuring a local scope. The second limitation is resolved by the recurrence relation for the Chebyshev polynomials, $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, which allows for efficient computation linear to the number of edges.

Spatial Methods

On the other hand, spatial Methods carry out the operations in the spatial domain. These methods focus on aggregating features from the immediate neighbours of each node [31]. The aggregation process for a node v is mathematically formulated as:

$$h_v^{(k+1)} = AGGREGATE^{(k)}(\{h_u^{(k)} : u \in N(v)\}) \tag{2.8}$$

where $h_v^{(k)}$ represents the feature vector of node v at the k-th layer, and $N(v)$ denotes the set of neighbours of v. Post-aggregation, a transformation is applied, typically in the form of:

$$h_v^{(k+1)} = \sigma(W^{(k)} \cdot h_v^{(k+1)}) \tag{2.9}$$

with $W^{(k)}$ being a layer-specific learnable weight matrix and σ a non-linear activation function. This approach is quite similar to a message-passing scheme where nodes exchange and update features based on their local neighbourhoods. The advantage of spatial methods is their computational efficiency, especially when dealing with varying graph structures. However, they may require careful design to be able to capture the global graph properties well.

2.5 Related Work

This section first reviews the state-of-the-art models that aim to predict brain graph evolution using GNNs. It then examines research in the area of biological reservoir computing. This review highlights key advancements, methodologies, and findings, and identifies any current limitations and future research directions in these fields.

2.5.1 Brain graph evolution prediction

Nebli et al. [4] introduced EvoGraphNet, which is composed of a sequential chain of graph-generative adversarial networks (gGANs), with each link in the chain dedicated to the prediction of the brain graph of the subsequent time point. Central to the architecture of EvoGraphNet is the generative adversarial network (GAN), consisting of two primary components: a generator, denoted as G, and a discriminator, D. The generator, functioning as an autoencoder, given a brain graph of the current time point, is designed to fabricate synthetic samples representative of the following time point. Concurrently, the discriminator refines its ability to discern between the genuine, ground-truth data and the fabricated data produced by the generator. The training paradigm of this network allows for a dual advancement. The generator progressively enhances its capacity to create more precise and convincing samples, while the discriminator simultaneously improves its proficiency in differentiating between the actual data and the generated samples. EvoGraphNet demonstrates notable success in this domain, achieving an average MAE of 0.05495 at t_1 and 0.08048 at t_2. However, the model's complexity and the sequential nature of its generator-discriminator stacks demand substantial computational resources and extended training durations.

Addressing the computational efficiency challenge, Tekin et al. [5] proposed the Recurrent Brain Graph Mapper (RBGM). This model comprises a series of mappers, each corresponding to a distinct time point. Each mapper employs a graph neural network that executes recurrent dynamic edge-filtered convolution, where the hidden state is propagated through all the mappers, and the output node embeddings are transformed into a predicted brain graph. The model also leverages the teacher-forcing method during training to enhance the quality of the generated brain graph. RBGM has shown improved performance over EvoGraphNet in terms of mean MAE at t_1 on the same dataset, but trailing slightly behind EvoGraphNet at t_2. On the other hand, RBGM achieves a 46% reduction in training time for t_2, such, RBGM is able to match the performance whilst reducing the training time drastically. However, both EvoGraphNet and RBGM are limited to predictions on single-modal data.

Expanding the scope to multigraph populations, Demirbilek et al. [6] focused on forecasting and integrating longitudinal multigraph datasets by learning time-dependent Connectome-Based Templates (CBTs). Their model, Recurrent Multigraph Integrator Network (ReMI-Net*), implements a recurrent graph convolution approach, using the message-passing paradigm. This technique embeds dependencies at both the node neighbourhood and temporal levels into a unified node embedding matrix, representing consecutive time points for an individual subject. Subsequently, a transformation layer is applied to these embeddings to predict the CBT. This approach parallels RBGM but extends to multigraph scenarios, incorporating normalisation operations across views to account for scale variations in view-specific features.

Targeting the sample size scarcity issue in longitudinal datasets for brain connectomes, Gürler et al. [7] developed a federated learning framework, 4D-FED-GNN+, that allows the training of GNN models across multiple hospitals without the need for centralised data storage. The training process involves iteratively updating local models at each hospital and aggregating the updates on a central server to refine a global model.

Collectively, these studies have addressed diverse challenges in predicting brain graph evolution. However, a notable gap in the existing literature is the integration of biological traits into the predictive models. Consequently, the focus of this project will be to explore this uncharted territory by combining biological reservoir computing with one of these existing methodologies, potentially opening new avenues in brain graph evolution prediction.

2.5.2 Biological Reservoir Computing

The biological feasibility of RC has been extensively researched for its ability to replicate the brain's handling of temporal data [8]. A notable early RC model aimed to decipher brain mechanisms governing eye movements in varying scenarios [32]. This model centres on the prefrontal cortex, which is involved in complex cognitive tasks (such as decision-making, problem-solving, and social behaviour) and its connection to the striatum. The prefrontal cortex neurons formed a fixed network (reservoir), with adaptable connections to the striatum (readout). This aided in understanding visual perception and eye movement coordination. Subsequent research expanded this model's applications to include learning patterns in sequential inputs [33], integrating it with language processing models for grammar learning [34] and learning algorithm enhancement [35].

Advancements in imaging have enabled detailed mapping of human brain circuits. This allows the traditional RC models to be enhanced biologically by incorporating structured connectivity based on actual brain networks, thus, allowing us to explore the brain's computational capabilities. Damicelli et al. [10] integrated real brain connectomes into Echo State Networks (ESNs), creating BioESNs through the bio2art framework, which maps and scales up real connectomes for use in recurrent ANNs. The study evaluated the performance of connectomes from different primate species: humans, macaques, and marmosets. Various surrogate network conditions were also created to compare with the empirical connectome-based BioESNs, including different random wiring configurations and connection densities. The study employed two types of memory tasks: the Memory Capacity Task and the Memory Sequence Recall Task. It found that BioESNs with real connectome-based reservoirs performed comparably to classical ESNs with random connectivity, provided there was enough randomness in the connections. Additionally, the study demonstrated that larger reservoirs with heterogeneous connectivity patterns could overcome the memory performance limitations inherent to the underlying connectivity. Despite these advancements, the study has several limitations. It focused only on memory tasks framed as regression problems. Thus, future research should include classification tasks and more ecologically realistic tasks. Additionally, considering entire connectomes as single networks limits the exploration of network subdivisions corresponding to different brain systems.

Suárez et al. [36] also conducted experiments on biologically instantiated reservoirs using human connectomes. The study investigated the effects of the reservoir in stable (spectral radius of connectomes < 1), critical (spectral radius of connectomes $= 1$), and chaotic (spectral radius of connectomes > 1) states. The findings reveal that memory capacity is optimal at the edge of chaos, where the network exhibits critical dynamics. Moreover, the study identifies that network structure significantly influences memory capacity, with moderate to high correlations observed. At criticality, memory capacity relies more on global network dynamics than on topological features. Whereas in stable and chaotic regimes, the topology plays a greater role. Later, Suárez et al. [11] presented the development of the conn2res toolbox, an open-source Python framework designed to implement biological neural networks as artificial neural networks for performing cognitive tasks. The toolbox allows the selection of input and readout nodes, so researchers can configure the arbitrary network architectures with different types of local dynamics. For example, by setting the input nodes from the visual system and output nodes from the somatomotor system. This flexibility allows researchers to explore the effects of network subdivisions on cognitive functions.

Chapter 3

Methodologies

This section outlines the key steps of our MemoGNN for predicting brain graph evolution from a single time point.

3.1 Overview

In the longitudinal brain graph population **dataset**, we have a set of longitudinal tensors $\{\mathcal{T}_0^t, \ldots, \mathcal{T}_{n_s}^t\}_{t=1}^{n_t}$ with each tensor $\{\mathcal{T}^t\}_{t=1}^{n_t}$ representing a subject. During **preprocessing**, all brain connectivity matrices are set as the weight matrices of an echo state network to create biologically instantiated reservoirs. We subsequently train the reservoirs to remember the lagged version of the X_{train} input signal, Y_{train}, and then measure their memory capacity based on performance on X_{test}. This process yields the memory capacity for all brain connectivity matrices in the dataset. Our proposed model includes an **arbitrary graph neural network** g that processes a brain connectivity matrix $\mathcal{T}_s^t$ to predict the brain connectivity matrix $\mathcal{T}_s^{t+1}$. We then create a **biologically instantiated reservoir** and measure the memory capacity using the same methodology as in the preprocessing stage. We compare the memory capacity of the predicted brain graph $\hat{m}_s^{t+1}$ with that of the actual brain graph m_s^{t+1}, obtained during preprocessing. The mean absolute error (MAE) between the two memory capacities serves as one component of the **loss function**, along with the MAE between the predicted and actual brain graphs. The overall model flow is shown in Figure 3.1, and the main mathematical notations used are detailed in Table 3.1. In the next sections, we dive into the mathematical formulation behind the main component of our model, the biologically instantiated reservoir, as well as the loss function we introduced for minimising the cognitive closeness of the produced brain graphs.

3.2 Biologically Instantiated Reservoir

The main component of our model consists of the biologically instantiated reservoir. We implemented the Echo State Network (ESN) introduced in Section 2.3.1

Notation	Definition
n_r	Total number of nodes (regions of interests) in the network
n_s	Total number of subjects in the dataset
n_t	Total number of time points in the dataset
$\mathcal{T}_s^t$	Brain connectivity matrix of subject s at time point $t \in \mathbb{R}^{n_r \times n_r}$
$\hat{\mathcal{T}}_s^t$	Predicted brain connectivity matrix of subject s at time point $t \in \mathbb{R}^{n_r \times n_r}$
m_s^t	Memory capacity of the brain connectivity matrix of subject s at time point t
$\hat{m}_s^t$	Memory capacity of predicted brain connectivity matrix of subject s at time point t
n_x	The number of features in the input signal
n_{res}	The number of neurons in the reservoir of a neural network
n_o	The number of features in the output signal
ts	The number of time steps in the input/output signal
$\mathbf{x}_t$	Input signal to the reservoir, at time step t $\in \mathbb{R}^{n_x}$
$\mathbf{W}_{in}$	Input weight matrix, dimension $\in \mathbb{R}^{n_{res} \times n_x}$
$\mathbf{W}_{res}$	Reservoir weight matrix $\in \mathbb{R}^{n_{res} \times n_{res}}$
$\mathbf{h}_t$	State vector at time step $t \in \mathbb{R}^{n_{res}}$
$\tanh$	Hyperbolic tangent activation function
$[;]$	Concatenation of two vectors
$\mathbf{S}$	Combined states matrix $\in \mathbb{R}^{ts \times (n_{res}+n_x)}$
$\mathbf{T}$	Target output matrix $\in \mathbb{R}^{ts \times n_o}$
$\mathbf{S}^+$	Pseudoinverse of $\mathbf{S} \in \mathbb{R}^{(n_{res}+n_x) \times ts}$
$\mathbf{W}_{out}$	Output weight matrix, dimension $\in \mathbb{R}^{n_o \times (n_{res}+n_x)}$
$\mathbf{s}_t$	Combined states vector at time step $t \in \mathbb{R}^{n_{res}+n_x}$
$\mathbf{y}_t$	Actual output at time step $t \in \mathbb{R}^{n_o}$
$\hat{\mathbf{y}}_t$	Predicted output at time step $t \in \mathbb{R}^{n_o}$
λ_1	Adjustable coefficient for the l_1 loss
λ_2	Adjustable coefficient for the memory capacity loss
$\mathcal{L}_{l1}$	MAE between the actual and predicted brain graphs
$\mathcal{L}_{mem}$	MAE of the memory capacity between the actual brain graph and the predicted brain graph
$\mathcal{L}_{total}$	Overall loss computed for the training sample s

Table 3.1: *Major parameters used in MemoGNN.* We use capital letters X to denote matrices, bold lower case letters $\mathbf{x}$ to denote vectors and lower case letters x to denote scalar values

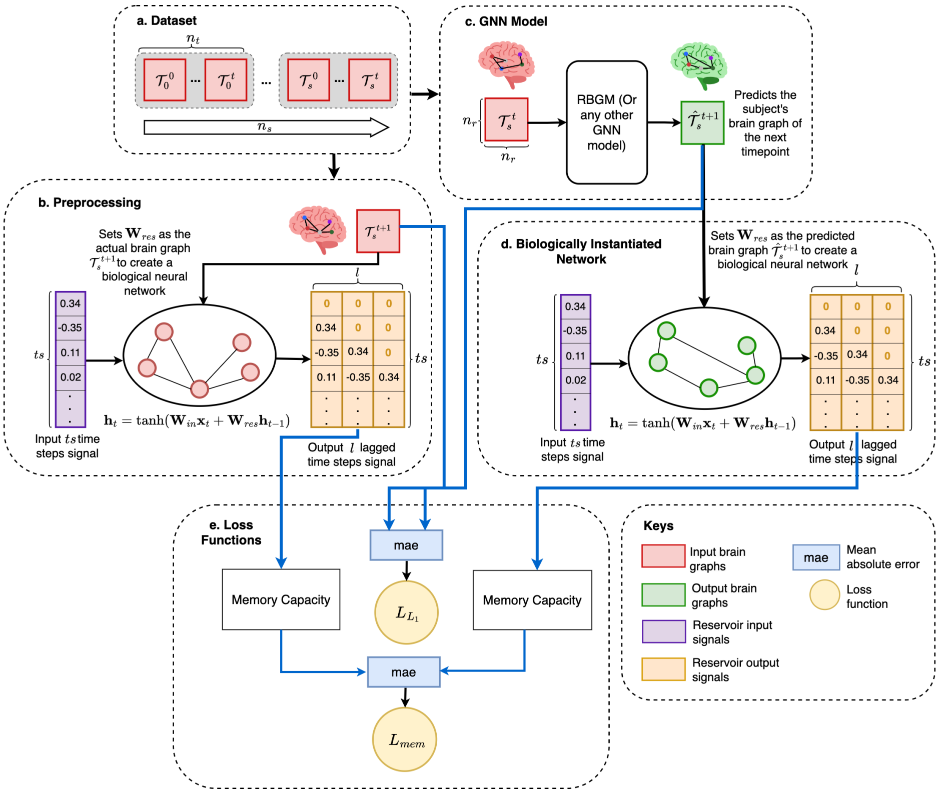

a. Dataset
b. Preprocessing
c. GNN Model
RBGM (Or any other GNN model)
Predicts the subject's brain graph of the next timepoint
d. Biologically Instantiated Network
Sets $\mathbf{W}_{res}$ as the actual brain graph $\mathcal{T}_s^{t+1}$ to create a biological neural network
Sets $\mathbf{W}_{res}$ as the predicted brain graph $\hat{\mathcal{T}}_s^{t+1}$ to create a biological neural network
$\mathbf{h}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{res}\mathbf{h}_{t-1})$
Input ts time steps signal
Output l lagged time steps signal
e. Loss Functions
Memory Capacity
mae
L_{L_1}
L_{mem}
Keys
Input brain graphs
Output brain graphs
Reservoir input signals
Reservoir output signals
Mean absolute error
Loss function

Figure 3.1: *MemoGNN Architecture* **a. Dataset** consisting of longitudinal brain graph tensors, each representing a subject's brain connectivity at different time points. **b. Preprocessing** where brain connectivity matrices are set as the weight matrices of an echo state network to create biologically instantiated reservoirs. These reservoirs are trained to remember lagged versions of input signals and evaluated on the test signals, yielding memory capacities for all brain graphs. **c. GNN model** predicts the next time point's brain graph from the current graph. **d. Biologically instantiated reservoir** created using the predicted graph, and its memory capacity is measured similarly. **e. Loss functions** include the mean absolute error (MAE) between the memory capacities of the predicted and actual brain graphs, and the MAE between the predicted and actual brain graphs themselves.

from scratch, tailoring it to our application by setting reservoir weights to match the predicted brain graph. The implementation includes several key components, as described below.

State Update

The state update equation is given by:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{res}\mathbf{h}_{t-1})$$

Since this is a biologically instantiated reservoir, we set $\mathbf{W}_{res}$ to a brain connectivity matrix $\mathcal{T}_s^t$. Hence the equation becomes:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathcal{T}_s^t\mathbf{h}_{t-1})$$

This step transforms the complex, non-linearly separable data into a more manageable, linearly separable form.

Forward Pass

In the forward pass, the state update function is used to compute the hidden states over time. Then the combined states, which incorporate both the hidden states and the original input, are then returned:

$$\mathbf{s}_t = [\mathbf{h}_t; \mathbf{x}_t]$$

where $[;]$ denotes concatenation. Combining the hidden states with the raw input allows the model to leverage both the transformed features from the reservoir and the original input features. This provides a richer and more comprehensive feature representation. On top of that, this balances the memory of past states captured by the reservoir with the influence of the current input.

Training Output Weights

The output weights $\mathbf{W}_{out}$ are trained using the pseudoinverse method:

$$\mathbf{W}_{out} = (\mathbf{S}^{+}\mathbf{T})^{\top}$$

We chose the pseudoinverse to solve for $\mathbf{W}_{out}$ in the readout layer over gradient decent training as it is much more computationally efficient and provides a more accurate prediction. This is because it calculates a direct solution to the problem, whereas in gradient descent, iterative updates and hyperparameter tuning need to be carried out. The memory capacity for each subject s and each time point t is calculated during training, therefore this process has to be as efficient as possible.

Prediction

The prediction equation is:

$$\hat{\mathbf{y}}_t = \mathbf{W}_{out}\mathbf{s}_t$$

gives us the predicted signal from the biologically instantiated reservoir, which can then be used to calculate its memory capacity.

Memory Capacity Task

In this task, the reservoir's goal is to learn and predict delayed versions of the input signal. We define two memory capacity tasks, the one introduced by Jaegar [12], which we termed the regular memory capacity task, and the language-infused memory capacity task, which we have introduced. The difference between them lies in how the signal X is generated.

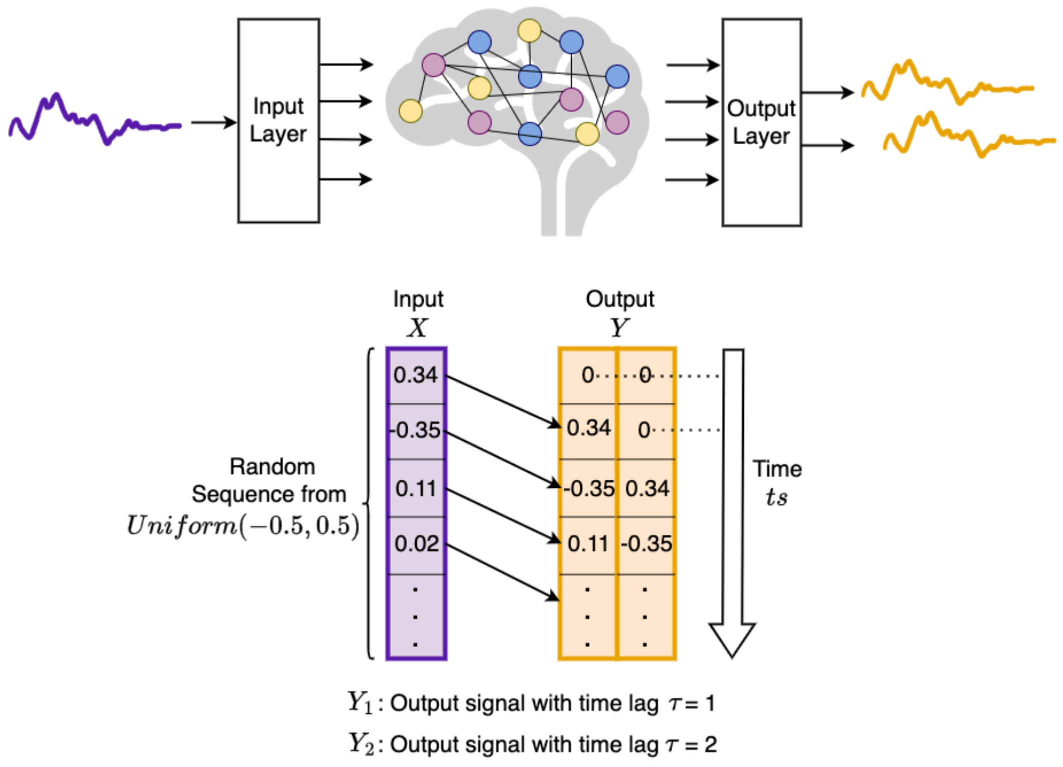

Input Layer
Output Layer
Input X
Output Y
Random Sequence from $Uniform(-0.5, 0.5)$
0.34
-0.35
0.11
0.02
0
0
0.34
0
-0.35
0.34
0.11
-0.35
Time ts
Y_1: Output signal with time lag τ = 1
Y_2: Output signal with time lag τ = 2

Figure 3.2: A biologically instantiated reservoir performing the regular memory capacity task with max time lag set to 2

In the **regular memory capacity task**, each value of X termed x_t is chosen randomly from a uniform distribution between -0.5 and 0.5 and is passed through an input neuron. Formally, we can write:

$$x_t \sim \text{Uniform}(-0.5, 0.5)$$

In the **language-infused memory capacity task**, x_t is the word embedding of a token, chosen from a text corpus. We obtain this word embedding using the Word2vec algorithm.

For each delay time τ, there's an output Y_τ that predicts the input X at the time $t - \tau$. Then, the reservoir tries to generate an output sequence $\hat{Y}_\tau$ which aims to approximate the delayed input sequence Y_τ. The predicted output at time step t can be expressed as:

$$\hat{y}_\tau(t) \approx x_{t-\tau}$$

The overall performance which is termed the Memory Capacity, is calculated by summing up the squared Pearson correlation coefficient (ρ) across all outputs (from no lag to the maximum lag set), reflecting the reservoir's ability to remember and predict the input at different time lags. The Pearson correlation coefficient ρ between the true output y_τ and the predicted output $\hat{y}_\tau$ for a given delay τ is defined as:

$$\rho(y_\tau, \hat{y}_\tau) = \frac{\sum_t (y_\tau(t) - \bar{y}_\tau)(\hat{y}_\tau(t) - \bar{\hat{y}}_\tau)}{\sqrt{\sum_t (y_\tau(t) - \bar{y}_\tau)^2 \sum_t (\hat{y}_\tau(t) - \bar{\hat{y}}_\tau)^2}}$$

where $\bar{y}_\tau$ and $\bar{\hat{y}}_\tau$ are the mean values of the true and predicted outputs, respectively. The Memory Capacity (MC) can then be expressed as:

$$\text{MemoryCapacity} = \sum_\tau \rho^2(y_\tau, \hat{y}_\tau) \tag{3.1}$$

This formula indicates that Memory Capacity is the sum of the squared Pearson correlation coefficients over all considered time lags. A visual representation of the regular memory capacity task is shown in Figure 4.1.

3.3 Loss function

We decided to use the L_1 loss as part of the loss function. Given the current time point t and the predicted brain graph at time point $t+1$ for subject s as $\hat{\mathcal{T}}_s^{t+1}$, we define the L_1 loss as follows:

$$\mathcal{L}_{L_1}(\hat{\mathcal{T}}_s^{t+1}, \mathcal{T}_s^{t+1}) = \|\hat{\mathcal{T}}_s^{t+1} - \mathcal{T}_s^{t+1}\|_1$$

We chose the L_1 loss because it is resilient against outliers, which makes it suitable for brain graphs that are sparse and on top of that, exhibit high variability. We also implemented the memory capacity loss by using Formula 3.1, which measures the brain graph's ability to retain and utilise past information to make accurate forecasts. During the preprocessing stage, we obtained the memory capacity m_s^{t+1} for the actual brain graph. Thus while training, we only need to compute the memory capacity $\hat{m}_s^{t+1}$ for the predicted brain graph. We then calculate the absolute difference between the two:

$$\mathcal{L}_{mem}(\hat{\mathcal{T}}_s^{t+1}, \mathcal{T}_s^{t+1}) = \left|\hat{m}_s^{t+1} - m_s^{t+1}\right|$$

We chose to implement this loss to ensure that the predicted brain graphs not only match the ground-truth graphs in terms of connectivity values but also in their functional memory capacity. This will allow for maintaining the cognitive closeness of the brain network across different time points, resulting in more biologically plausible predictions.

Overall, the total loss function L_{total} would be as follows:

$$\mathcal{L}_{total}(s) = \sum_{i=1}^{s} (\lambda_1 \mathcal{L}_1(\hat{\mathcal{T}}_s^{t+1}, \mathcal{T}_s^{t+1}) + \lambda_2 \mathcal{L}_{mem}(\hat{\mathcal{T}}_s^{t+1}, \mathcal{T}_s^{t+1}))$$

where λ_1 and λ_2 are hyperparameters that need to be tuned to adjust to the two losses.

Chapter 4

Results and Evaluation

In this chapter, we present our experimental results, evaluating the performance of various graph neural network models in their ability to forecast 4D brain connectivity. Our analysis compares the baseline models with their reservoir-enhanced counterparts (MemoGNNs), on two types of memory capacity tasks. We begin by outlining the evaluation measures and their mathematical definitions. We then present the two datasets used during the evaluation. Lastly, we discuss the results obtained for the two memory tasks and provide further analysis to explain the underlying patterns.

4.1 Evaluation Measures

In this section, we discuss the evaluation measures used in our analysis, clarifying the mathematical definitions and their significance in the context of our study.

MAE

We will use the Mean Absolute Error (MAE) as a measurement for comparison between the predicted brain graph matrix ($\hat{Y}$) and the actual brain graph matrix (Y). The MAE is defined mathematically as:

$$MAE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}|Y_{ij} - \hat{Y_{ij}}|$$

This metric will be applied across all time points. Subsequently, these results will be compared with those obtained from the baseline models to evaluate whether our model demonstrates a reduction in MAE.

MAE of memory capacity

We propose to compare the memory capacity of the predicted brain graph matrix ($\hat{Y}$) when used as the state of the reservoir, against that of the actual brain graph matrix

(Y). The similarity of the memory capacity would show us the cognitive closeness of the brain graphs. We define the absolute difference in memory capacities as:

$$Diff = |MC(Y) - MC(\hat{Y})|$$

Node strength

The node strength [37] is the sum of the weights of all edges connected to a node in a weighted graph. It measures the importance of node i in a graph. Mathematically, the node strength s_i of a node i is defined as:

$$s_i = \sum_{j \in N(i)} w_{ij}$$

where:

- $N(i)$ is the set of neighbours of node i.
- w_{ij} is the weight of the edge between nodes i and j.

We calculate the average node strength for each graph to allow for an easy comparison between the predicted and actual graphs. This comparison is achieved by computing the absolute difference between the average node strengths of the two graphs.

Participation coefficient

The participation coefficient [38] measures how evenly a node's connections are distributed among different communities within the graph. It indicates the extent to which a node participates in multiple communities, with $P_i = 0$ meaning that the node is fully restricted to its own community. The closer P_i is to 1, the more evenly the node's connections are spread across different communities, indicating a higher level of participation in multiple communities.

Mathematically, the participation coefficient P_i of a node i is defined as:

$$P_i = 1 - \sum_{c=1}^{N_C} \left(\frac{k_{i,c}}{k_i} \right)^2$$

where:

- N_C is the number of communities.
- $k_{i,c}$ is the number of links from node i to nodes in community c.
- k_i is the total degree of node i.

Diversity coefficient

The Shannon entropy-based diversity coefficient [39] H_i measures the diversity of a node's connections in a graph. It reflects how a node's connections are distributed across communities. A high entropy value indicates that connections are evenly spread across multiple communities, signifying high diversity. A low entropy value indicates that connections are concentrated within a few communities, signifying low diversity.

Mathematically, the Shannon entropy-based diversity coefficient H_i of a node i is defined as:

$$H_i = -\sum_{c=1}^{N_C} p_{i,c} \log p_{i,c}$$

where:

- N_C is the number of communities.
- $p_{i,c} = \frac{k_{i,c}}{k_i}$ is the proportion of links from node i to nodes in community c.
- $k_{i,c}$ is the number of links from node i to nodes in community c.
- k_i is the total degree of node i.

Betweenness centrality

Betweenness centrality [40] measures a node's importance in a graph based on the number of shortest paths that pass through it. It quantifies the role of a node as a bridge or connector within the graph, highlighting nodes that significantly influence the flow of information or resources. A node with high betweenness centrality has considerable control over communication between other nodes in the graph.

The betweenness centrality $C_B(v)$ of a node v is defined as:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where:

- σ_{st} is the total number of shortest paths from node s to node t.
- $\sigma_{st}(v)$ is the number of those shortest paths that pass through node v.
- The summation is over all pairs of nodes s and t in the network, excluding node v.

Eigenvector centrality

Eigenvector centrality [41] measures a node's influence within a graph. It extends the concept of degree centrality by considering not only the number of a node's direct connections (its degree) but also the importance of the nodes to which it is connected. A node with high eigenvector centrality is connected to other nodes that themselves have high centrality.

The eigenvector centrality $C_E(v)$ of node v can be defined as:

$$C_E(v) = \frac{1}{\lambda} \sum_{u \in N(v)} A_{uv} C_E(u)$$

where:

- λ is the largest eigenvalue of the adjacency matrix $\mathbf{A}$.
- $N(v)$ is the set of neighbours of node v.
- A_{uv} is the element of the adjacency matrix $\mathbf{A}$ corresponding to the edge between nodes u and v.

Local efficiency

Local efficiency [42] measures how well information is exchanged within the immediate neighbourhood of a node in a graph. It quantifies the robustness of the graph to the failure of individual nodes by evaluating the efficiency of communication among a node's neighbours when the node itself is removed. High local efficiency indicates that the neighbours of a node are well connected, even in the absence of the node, which suggests that the graph has a resilient local structure.

The local efficiency $E_{loc}(i)$ of a node i is defined as the efficiency of the subgraph G_i induced by the neighbours of i. It can be expressed as:

$$E_{loc}(i) = \frac{1}{k_i(k_i - 1)} \sum_{j,h \in N(i)} \frac{1}{d_{jh}}$$

where:

- k_i is the degree of node i, i.e., the number of neighbours of node i.
- $N(i)$ is the set of neighbours of node i.
- d_{jh} is the shortest path distance between nodes j and h in the subgraph G_i.

Global efficiency

Global efficiency [42] is a measure of how efficiently information is exchanged across the entire graph. It quantifies the ease of communication between all pairs of nodes, considering the shortest paths connecting them. High global efficiency indicates that the graph allows for quick information transfer between nodes, reflecting an overall well-connected and integrated structure.

The global efficiency E_{glob} of a graph is defined as the average efficiency of the shortest paths between all pairs of nodes. It can be expressed as:

$$E_{glob} = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d_{ij}}$$

where:

- E_{glob} is the global efficiency of the network.
- N is the total number of nodes in the network.
- d_{ij} is the shortest path distance between nodes i and j.

Modularity

Modularity [43] quantifies the strength of a graph's division into communities. It evaluates how well the graph is partitioned into subgroups, ensuring nodes within the same group are more densely connected to each other than to nodes in different groups. High modularity values indicate a strong community structure, with many connections within modules and relatively few between them. The modularity Q of a graph partition is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where:

- m is the total number of edges in the network.
- A_{ij} is the adjacency matrix, where $A_{ij} = 1$ if there is an edge between nodes i and j, and $A_{ij} = 0$ otherwise.
- k_i and k_j are the degrees of nodes i and j, respectively.
- $\delta(c_i, c_j)$ is the Kronecker delta function, which is 1 if nodes i and j are in the same community (i.e., $c_i = c_j$) and 0 otherwise.

Density

Density is used to quantify the proportion of possible connections that are actual connections within the graph. It reflects how closely knit or interconnected the graph is. In an undirected graph, density is calculated by comparing the number of edges present to the total number of possible edges. A higher density indicates a more interconnected graph, where many nodes are directly connected to each other.

The density D of an undirected graph is defined as:

$$D = \frac{2m}{N(N-1)}$$

where:

- m is the total number of edges in the network.
- N is the total number of nodes in the network.
- $\frac{N(N-1)}{2}$ is the total number of possible edges in an undirected graph with N nodes.

Clustering Coefficient

The clustering coefficient [44] measures how close a node's neighbours are to form a complete graph (i.e., how interconnected a node's neighbours are). In the context of an entire graph, transitivity is a classical version of the clustering coefficient. It measures the overall tendency of nodes to cluster together. Specifically, transitivity is the ratio of the number of triangles (three nodes all connected to each other) to the number of triplets (two edges that share a common node) in the network. This measure provides insight into how likely nodes in the network are to cluster or form tightly-knit groups.

For an undirected weighted graph, the transitivity (global clustering coefficient) is defined as:

$$C = \frac{3 \times \text{number of triangles}}{\text{number of triplets}}$$

In terms of the adjacency matrix A, where A_{ij} represents the weight of the edge between nodes i and j, the equation can be more formally written as:

$$C = \frac{\sum_{i,j,k} (A_{ij} A_{jk} A_{ki})^{1/3}}{\sum_i k_i(k_i - 1)}$$

where:

- A_{ij} is the weight of the edge between nodes i and j.
- k_i is the degree of node i, i.e., the number of edges connected to node i.
- The numerator counts the number of triangles in the graph, considering the weights of the edges.
- The denominator is the number of triplets (three nodes connected by two edges) in the graph.

4.2 Evaluation Datasets

To assess the performance of our model, we will use the OASIS-2 longitudinal dataset [45]. The processed dataset encompasses a cohort of 114 individuals, ranging in age from 60 to 96 years. Each participant underwent magnetic resonance imaging (MRI) during two or more visits, with a minimum interval of one year between each session, adding up to a total of 373 imaging sessions. The dataset includes 3 to 4 individual T1-weighted MRI scans per subject, all acquired in single scan sessions. Before analysis, brain graphs will need to be constructed for each scan, as outlined in Section 2.1, which will facilitate subsequent analyses and model evaluations. Upon visualising the data, we discovered an anomalous brain connectome with a maximum value of 165.648. To ensure the integrity and quality of our dataset, we removed this anomalous data point. After excluding the anomaly, we are left with a dataset of 113 individuals.

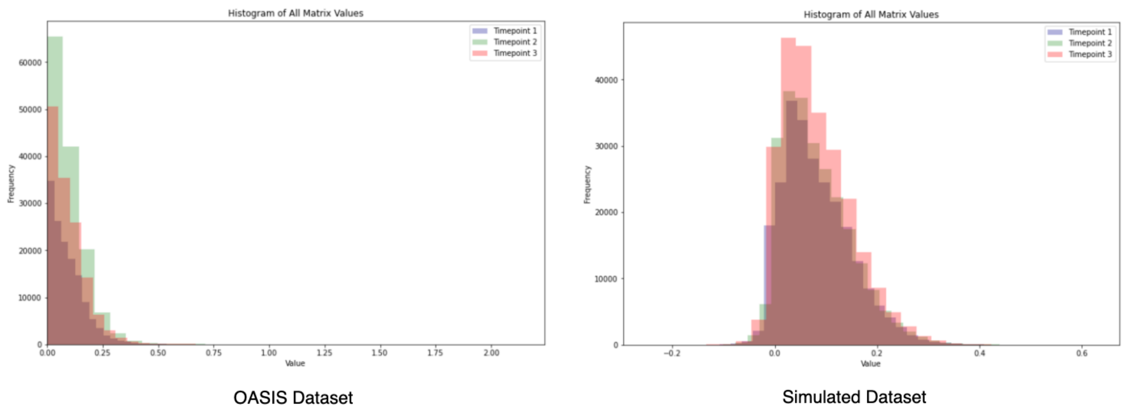


Figure 4.1: The data distribution of all the connectomes in the two different evaluation datasets

We also generated a simulated dataset from the OASIS-2 dataset to further evaluate our model's performance. This process involves calculating the mean connectivity values and the correlation matrices of the dataset and creating samples from a multivariate normal distribution based on these predefined statistics. Each sample is anti-vectorised to convert into an adjacency matrix. It involves reconstructing the upper triangular part of the matrix from a vector and mirrors it to form a symmetric 35x35 matrix. To simulate temporal changes, the matrix is perturbed by adding time-specific noise derived from a tanh function, combined with real-time difference data, ensuring that later time points show greater divergence from the baseline. Overall, we generated 200 samples, each having 3 time points.

4.3 Graph Neural Network Models

In our experiments, we utilised various models to evaluate the behaviour of memory-aware brain graphs. Below, we provide a brief overview of each model, without delving deeply into the mathematical theory. Interested readers are encouraged to consult the referenced papers for more detailed information.

4.3.1 Chebyshev Graph Convolutional Recurrent Network

Seo et al. introduced the Chebyshev Graph Convolutional Recurrent Network (CGCRN) [13], which leverages a spectral graph convolution approach. The theory behind Chebyshev graph convolution is discussed in detail in Section 2.4.2. By employing the graph filtering function $*_{\mathcal{G}}$ from Equation 2.7 along with the recurrent units discussed in Section 2.2.1, it is possible to create a hybrid model that effectively combines both methodologies.

The CGCRN model integrates these techniques by first applying graph convolutions to the input data at each time step to capture spatial dependencies. The output of the graph convolutional layer is then fed into a recurrent unit to capture temporal dependencies. This integration allows the model to learn complex spatiotemporal patterns in the data, enhancing its ability to predict and understand dynamic systems with graph-structured data.

4.3.2 Topology Adaptive Graph Convolutional Network

The Topology Adaptive Graph Convolutional Network (TAGCN) [14] leverages a spectral graph convolution approach with a notable difference from typical spectral graph convolution: it employs filters that dynamically adapt to the topology of the graph. This dynamic adaptability means that the convolutional filters adjust based on the specific structure and connectivity of the graph at each layer, allowing for more flexible and precise learning of graph features. TAGCN utilises polynomial filter approximations, similar to those used in Chebyshev Graph Convolution, to perform graph convolutions efficiently. This approach enables the model to capture both local and global structural information within the graph. The adaptability of TAGCN makes it particularly effective for complex graph structures where the relationships between nodes can change dynamically over time, such as predicting the evolution of brain graph connectivity.

4.3.3 Recurrent Brain Graph Mapper

The Recurrent Brain Graph Mapper (RBGM) [5], different from all other models, uses a spatial graph convolutional approach. Details on spatial graph convolution are discussed in Section 2.4.2. Specifically, RBGM utilises edge-conditioned filters in its graph convolutional layers, which dynamically adapt to the graph's topology. This dynamic adaptation allows the model to efficiently capture intricate and evolving patterns of brain connectivity.

RBGM enhances these edge-conditioned filters by integrating recurrent neural network (RNN) elements, creating what is termed a graph recurrent filter. Unlike traditional edge-conditioned filters, the graph recurrent filter processes past information by leveraging the hidden state matrix from the previous time point as a memory. This enables the filter to generate messages between nodes (brain regions) that incorporate historical connectivity patterns, thereby improving the model's ability to predict future brain states.

4.3.4 Identity Mapper

We also implemented an identity model as a baseline for all of our models. In this approach, the predicted brain connectome at the next time point $t + 1$ is simply the input brain connectome at time point t. By using the identity model, we establish a reference point to evaluate the effectiveness of our predictive models. If a model cannot outperform this simple baseline, it indicates that the model may not be capturing meaningful temporal dynamics in the brain connectome data.

4.4 Experimental Setup

To evaluate our models, we used the following hardware and software configurations. The system features a 3.70 GHz Intel Xeon E5-1630 v3 processor, 62 GB of dual-channel DDR4 memory, and an NVIDIA TITAN Xp graphics card with 12 GB of GDDR5X video memory. It operates on Ubuntu 22.04.4 LTS and is equipped with NVIDIA driver version 535.171.04 and CUDA toolkit version 10.2.89. Our implementation utilised the PyTorch (version 1.9.1) library, Torch Geometric Temporal (version 0.54.0), and Torch Geometric (version 2.0.0) libraries. We also used the NetworkX and Brain Connectome Toolbox (bctpy) library to help us evaluate the topology of the generated brain graph.

We set up nine different models for our experiments. This set includes the baseline models: CGCRN with GRU unit (termed GConvGRU), CGCRN with LSTM (termed GConvLSTM), TAGCN (termed TAGConv), as well as RBGM. It also features the baseline models integrated with a biologically instantiated reservoir (MemoGNNs), and an identity model. For each model, we hand-tuned individual hyperparameters according to the datasets and model characteristics. The final hyperparameter settings are presented in Table 4.1.

Hyperparameters	**OASIS-2 Dataset**				**Simulated Dataset**			
	GConvGRU	**GConvLSTM**	**TAGConv**	**RBGM**	**GConvGRU**	**GConvLSTM**	**TAGConv**	**RBGM**
# of layers	2	2	2	1	3	3	3	1
Output channels	30, 35	30, 35	30, 35	1225, 35	140, 280, 35	140, 280, 35	140, 280, 35	1225, 35
Dropouts	0.5	0.5	0.5	-	0.3, 0.5	0.3, 0.5	0.3, 0.5	-
Activation functions	relu, relu	relu, relu	elu, relu	relu	relu, relu, relu	relu, relu, relu	elu, elu, relu	relu
λ_1	1	1	1	1	1	1	1	1
λ_2	0.00005	0.001	0.001	0.0001	0.0001	0.001	0.001	0.0005
Optimiser	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001	0.001	0.0001	0.0001	0.0001	0.001
# of epochs	20	30	20	20	20	20	15	20
# of folds	3	3	3	3	5	5	5	5

Table 4.1: Hyperparameters of different models based on the different datasets. The hyperparameter setting is the same for the baseline models and MemoGNN models, where λ_2 only applies to MemoGNN models.

4.5 Regular Memory Capacity Task

In this experiment, we evaluated all of the models using the regular memory capacity task as described in Section 3.2, where each value x_t is randomly chosen from a uniform distribution between -0.5 and 0.5. Using this method, we generated a training set consisting of 1000 time steps and a testing set consisting of 500 time steps, with a time lag of 35. After configuring the models with the hyperparameters specified in Table 4.1, we ran K-Fold cross-validation (80% training 20% testing) to evaluate the impact of MemoGNN on the baseline model's MAE, MAE of memory capacity, time costs, and memory costs. Subsequently, we assessed the errors across various network topologies.

4.5.1 Evaluation Results

First, we analyse the evaluation measures based on the values used in the loss functions. The results for the OASIS-2 dataset are presented in Table 4.2, while the results for the simulated dataset are displayed in Table 4.3. Additionally, graphical representations of the errors across folds are shown in Figure 4.2 for the OASIS-2 dataset and Figure 4.3 for the simulated dataset.

Mean Absolute Error (MAE)

Across both datasets, the baseline models generally exhibit lower MAE compared to their reservoir-enhanced counterparts. For the OASIS-2 dataset, as illustrated in Figure 4.2, only the TAGConv MemoGNN model outperformed the baseline model. A similar trend is observed in the simulated dataset, as seen in Figure 4.3, where only the RBGM MemoGNN model surpassed the baseline model. This indicates that the addition of reservoirs does not significantly improve the MAE in general contexts.

MAE of Memory Capacity

The introduction of reservoirs notably improves the MAE of Memory Capacity for all models across both datasets. From Table 4.2, it is clear that for the OASIS-2 dataset, the RBGM MemoGNN model achieved the best MAE of Memory Capacity performance, with scores of 0.2902 for t1 and 0.3703 for t2. A similar trend is observed in the simulated dataset, where the RBGM MemoGNN model again performed the best in terms of MAE of Memory Capacity, achieving 0.2460 for t1 and 0.6274 for t2. This improvement in memory capacity metrics is consistent across all model types, highlighting the efficacy of reservoirs in enhancing memory retention capabilities. However, for both datasets and across all models, only the RBGM MemoGNN model managed to outperform the identity model's memory capacity at t1, suggesting that there is still room for improvement.

We also analyse the resource utilisation of baseline models compared to MemoGNN models. The results for the OASIS-2 dataset are presented in Table 4.2, while the results for the simulated dataset are displayed in Table 4.3.

Time Cost

Reservoir-enhanced models consistently exhibit higher time costs compared to their baseline counterparts. On the OASIS-2 dataset, the TAGConv MemoGNN model required the most time, with 16.90 minutes per fold. A similar trend is observed on the simulated dataset, where the TAGConv MemoGNN model required 109.62 minutes per fold. For both datasets, we observe an average of around four times increase in time costs with the addition of reservoirs. This increase in computational time is a general characteristic of reservoir-enhanced models, reflecting the additional processing required.

Models	**MAE**		**MAE of Memory Capacity**		**Time Cost (minutes per fold)**	**Memory Cost (MB)**
	t1	t2	t1	t2		
GConvGRU	**0.0390 ± 3.43e-3**	**0.0435 ± 3.76e-3**	0.5570 ± 2.32e-2	0.5817 ± 3.70e-2	**2.99 ± 5.17e-2**	**3316.91 ± 8.67e-1**
GConvGRU + Reservoir	0.0431 ± 3.37e-3	0.0470 ± 3.61e-3	**0.4083 ± 1.21e-1**	**0.4785 ± 1.14e-1**	10.80 ± 9.35e-2	3325.62 ± 2.18
GConvLSTM	**0.0470 ± 3.91e-3**	**0.0513 ± 4.42e-3**	0.5887 ± 1.67e-1	0.7168 ± 1.21e-1	**4.20 ± 4.45e-2**	**2598.42 ± 4.35**
GConvLSTM + Reservoir	0.0487 ± 2.03e-3	0.0528 ± 2.13e-3	**0.5427 ± 1.11e-1**	**0.6820 ± 2.79e-2**	16.81 ± 1.25e-0	2601.19 ± 5.87
TAGConv	0.0400 ± 3.14e-3	0.0444 ± 2.48e-3	0.7914 ± 8.32e-2	0.9983 ± 1.45e-1	**3.28 ± 4.87e-2**	**2120.49 ± 2.32**
TAGConv + Reservoir	**0.0395 ± 2.86e-3**	**0.0437 ± 2.10e-3**	**0.7129 ± 8.01e-2**	**0.8078 ± 5.42e-2**	16.90 ± 2.35e-2	2120.49 ± 67.96
RBGM	0.0443 ± 4.19e-3	**0.0520 ± 4.64e-3**	0.3789 ± 3.11e-2	0.3959 ± 1.18e-1	**3.59 ± 2.32e-2**	**2646.20 ± 5.45**
RBGM + Reservoir	**0.0442 ± 4.23e-3**	0.0545 ± 5.66e-3	**0.2902 ± 2.39e-2**	**0.3703 ± 2.49e-2**	13.12 ± 1.32e-2	2700.56 ± 10.09
Identity	0.0397 ± 6.20e-3	0.0472 ± 6.38e-3	0.2956 ± 3.11e-2	0.2725 ± 6.19e-2	-	-

Table 4.2: Comparison of models with and without reservoir on the **OASIS-2 dataset**.

Models	**MAE**		**MAE of Memory Capacity**		**Time Cost (minutes per fold)**	**Memory Cost (MB)**
	t1	t2	t1	t2		
GConvGRU	**0.0322 ± 1.10e-3**	**0.0335 ± 6.07e-4**	0.9942 ± 7.97e-2	1.0809 ± 2.36e-2	**7.70 ± 3.80e-2**	2741.70 ± 15.72
GConvGRU + Reservoir	0.0329 ± 1.05e-3	0.0342 ± 4.57e-4	**0.7091 ± 8.60e-2**	**1.0050 ± 4.90e-2**	29.17 ± 2.06e-1	**2725.43 ± 3.95**
GConvLSTM	**0.0338 ± 1.48e-3**	**0.0350 ± 1.08e-3**	1.0225 ± 5.32e-2	1.1278 ± 2.86e-2	**6.20 ± 4.08e-2**	**2092.54 ± 64.04**
GConvLSTM + Reservoir	0.0357 ± 1.91e-3	0.0371 ± 1.56e-3	**0.8571 ± 1.26e-1**	**1.0260 ± 4.76e-2**	32.99 ± 1.25e-0	2654.06 ± 1.91
TAGConv	**0.03227 ± 1.09e-3**	**0.0335 ± 6.07e-4**	0.9942 ± 7.97e-2	1.0809 ± 2.36e-2	**17.52 ± 6.85e-2**	2092.24 ± 35.19
TAGConv + Reservoir	0.0336 ± 1.06e-3	0.0345 ± 6.49e-4	**0.6049 ± 8.51e-2**	**0.9896 ± 7.20e-2**	109.62 ± 3.95e-2	**1998.17 ± 1.95**
RBGM	0.0358 ± 8.75e-4	0.0375 ± 1.58e-3	0.5410 ± 1.01e-1	0.8663 ± 8.93e-2	**5.59 ± 1.50e-2**	**2588.91 ± 3.149**
RBGM + Reservoir	**0.0356 ± 7.89e-4**	**0.0373 ± 1.35e-3**	**0.2460 ± 3.97e-2**	**0.6274 ± 8.41e-2**	30.48 ± 3.44e-1	2621.08 ± 0
Identity	0.0396 ± 1.22e-3	0.0398 ± 1.34e-3	0.2268 ± 3.14e-2	0.2418 ± 2.94e-2	-	-

Table 4.3: Comparison of models with and without reservoir on the **simulated dataset**.

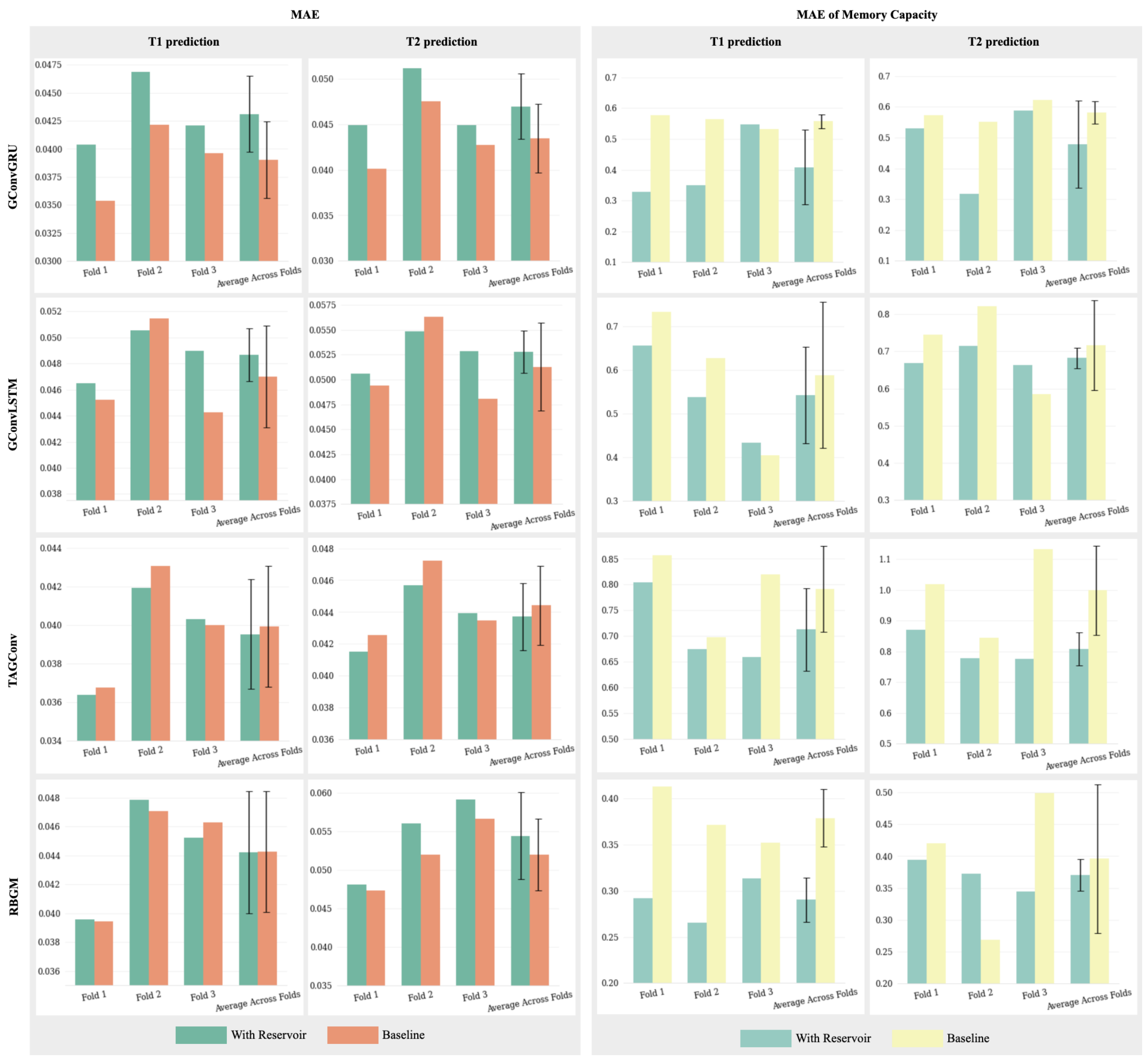


Figure 4.2: Bar chart comparison between baseline and MemoGNNs of the MAE and the MAE of memory capacity across folds on the **OASIS-2 dataset**

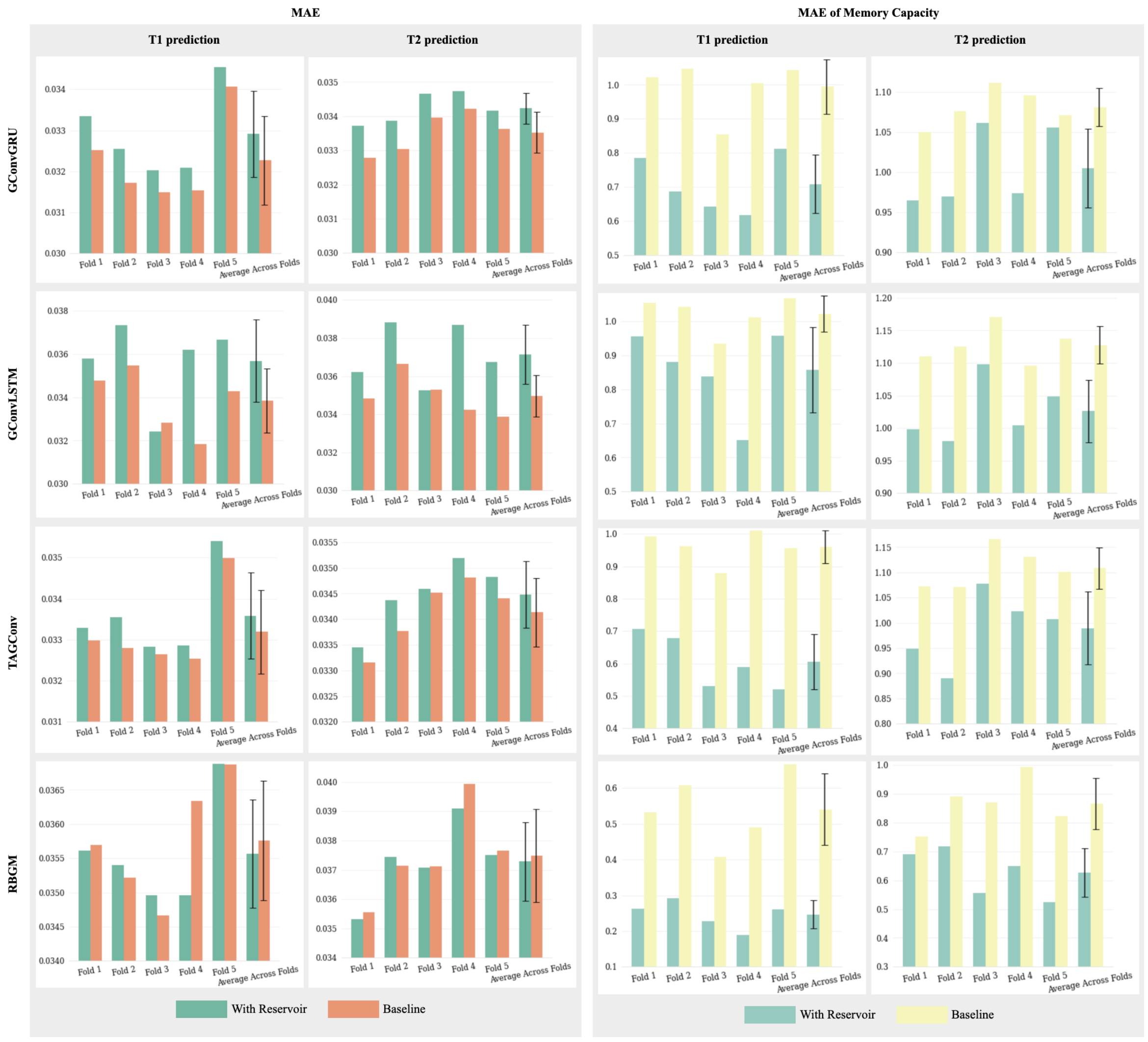


Figure 4.3: Bar chart comparison between baseline and MemoGNNs of the MAE and the MAE of memory capacity across folds on the **simulated dataset**

Memory Cost

Memory costs for models with reservoirs are generally higher, though the increase varies across datasets and models. On the OASIS-2 dataset, all MemoGNN models exhibited a higher memory cost. The results of the simulated dataset are more varied, with the GConvGRU MemoGNN and TAGConv MemoGNN models showing lower memory costs compared to other MemoGNN models. The increase in memory usage is more pronounced in some models, such as GConvLSTM and RBGM, where the reservoir-enhanced versions exhibit a substantial increase in memory cost.

Lastly, the topological closeness to the actual brain graph was analysed by categorising the topologies into three types. First, node-wise network topology includes metrics like node strength, participation coefficient, diversity coefficient, betweenness centrality, and eigenvector centrality. These metrics help us understand the roles and influence of individual nodes within the network. Second, efficiency-wise network topology includes global and local efficiency, reflecting how efficiently information is exchanged across the network, both overall and within local neighbourhoods. Lastly, global-level network topology comprises modularity, density, and clustering, providing a broad overview of the network's structure, such as the degree of modular organisation, overall connectivity, and local interconnectedness. These topological analyses comprehensively understand how closely the models replicate the brain's neural network structure. The results tables can be found in the appendix for both datasets.

Node-wise Network Topology

For node-wise network topology, we calculated the average values to enable comparison between the predicted and actual brain graphs. As shown in Table A.1, there isn't a general trend of MemoGNN improving across all these metrics for the OASIS-2 dataset. However, in the simulated dataset (Table A.4), MemoGNN models exhibit lower MAE in terms of participation coefficient and diversity coefficient across all models.

Efficiency-wise Network Topology

For efficiency-wise network topology, as observed in Table A.2, there is no consistent trend of improvement with MemoGNN models in the OASIS-2 dataset. In contrast, the simulated dataset (Table A.5) shows that MemoGNN models have a lower MAE in terms of both local and global efficiency across all models.

Global-level Network Topology

For global-level network topology, Table A.3 shows that, in the OASIS-2 dataset, the addition of reservoirs in MemoGNNs generally results in improved MAE for density compared to baseline models. However, there is no specific trend for modularity and clustering. In the simulated dataset (Table A.6), density and clustering metrics have both improved with the addition of reservoirs in MemoGNN models across all models.

4.5.2 Discussion

Overall, the results indicate that while reservoirs do not significantly enhance the general MAE performance, they substantially improve the memory capacity of the models. The RBGM MemoGNN models' superior performance in specific contexts suggests that certain GNN architectures initially generate brain graphs that are more biologically representative due to their inherent design and structure. This biological representativeness may lead to more accurate memory retention and processing. The inconsistent improvements in general MAE suggest that the reservoirs might introduce additional complexity, which could lead to overfitting or inefficient learning in certain scenarios. This complexity may arise from the increased number of parameters and interactions within the network, which might not always align with the underlying data patterns.

The resource utilisation analysis reveals that while reservoir-enhanced models improve memory capacity metrics, they incur significant costs in terms of both time and memory resources. The higher time costs are primarily due to the added computational complexity introduced by the reservoir layers. These layers involve extra processing steps to calculate each biologically instantiated reservoir's memory capacity, inherently requiring more computation per epoch and leading to increased training times. Similarly, the elevated memory costs are attributed to the additional parameters and storage necessary for the reservoirs.

In terms of the network topologies, the findings suggest that MemoGNN models have the potential to enhance certain aspects of them, which was theorised in our preliminary experiment as the topologies have a strong correlation to the regular memory capacity. However, while MemoGNN models demonstrate improvements in specific topological metrics for the simulated dataset, these improvements are not consistently observed in the OASIS-2 dataset. The only consistent improvement across both datasets is the MAE of density.

One potential reason for the consistent improvement in the MAE of density with MemoGNN across both datasets might be the memory capacity's strong correlation with this metric. This was shown in the preliminary experiment in Figure 1.6 that the density has the strongest correlation to the memory capacity. Thus, the models might find manipulating the graph's density an easier shortcut due to its direct impact on memory capacity.

The discrepancy across the datasets could be due to the inherent differences between the real-world OASIS-2 dataset and the controlled simulated dataset. To understand this discrepancy further, we analysed the data distributions of various network topology metrics from both datasets. We measured the network topologies across subjects and time points for each dataset. The box plots for each network topology, shown in Figure A.1, provide a visual comparison of the distributions between the two datasets.

A key factor explaining the performance difference is the spectral radius (largest absolute value of a matrix's eigenvalues) of the two datasets. In the real OASIS dataset, the MemoGNN model only achieved better performance in the MAE of density. This limited improvement could be due to the distribution of the spectral radius, which in the OASIS dataset shows greater variability and values further from 1. According to the study by Suarez et al. [36], optimal performance in neuromorphic networks, especially in tasks requiring high memory capacity, is achieved when the spectral radius is closer to 1. The variability in the spectral radius in the OASIS dataset suggests a less stable and optimal dynamical regime, reducing the effectiveness of the MemoGNN's memory capacity improvements in closely replicating the brain's topological features.

In contrast, the simulated dataset displayed a more centralised spectral radius distribution closer to 1, aligning more closely with the conditions described for optimal memory capacity. This more centralised spectral radius could have enhanced the MemoGNN's ability to replicate various topological features effectively. The simulated dataset's superior performance in participation coefficient, diversity coefficient, local efficiency, global efficiency, density, and clustering metrics underscores this point. While the simulated dataset exhibits greater variability in diversity and participation coefficients, its more centralised spectral radius allows for a better alignment with the neural network topologies, resulting in improved MAE in these metrics.

Further analysis was conducted on the local efficiency, global efficiency, modularity and density distribution of the two datasets, where the MemoGNN models consistently improved in the simulated dataset. All of these measurements have a more centralised and less variable distribution in the simulated data, which could have allowed for a more consistent and predictable network structure, thus supporting the models' ability to achieve lower MAE in these metrics.

4.6 Language-Infused Memory Task

In this experiment, we evaluated the models using the language-infused memory capacity task. This is different from the regular memory capacity task as it generates input signals from vectors representing words in a text and obtains output signals as lagged versions of these inputs. We convert words into word embeddings to capture the semantic meaning in a continuous vector space. For word selection, we used the Gutenberg corpus, a collection of classic English literature texts that is available through the NLTK library. This corpus includes works by various authors, providing a diverse set of textual data. We trained a Word2Vec model on this corpus to obtain word embeddings, setting the vector size to one dimension as the reservoir training is already computationally expensive.

Similar to the first experiment, we generated a training set with 1000 time steps and a testing set with 500 time steps, applying a time lag of 35. We selected the GConvGRU model, the most time-efficient GNN model as shown in the first experiment, as our baseline. This model was tested on both evaluation datasets and trained under a similar hyperparameter setting as described in Table 4.1. Except, in the case of the λ_2, which directly impacts the dynamics of the memory capacity loss. We tuned it carefully, setting $\lambda_2 = 0.00008$ for the OASIS-2 dataset and $\lambda_2 = 0.00005$ for the simulated dataset.

4.6.1 Evaluation Results

To begin with, we examine the evaluation measurements derived from the values used in the loss functions. Figure 4.4 provides a graphical representation of the errors across folds for the OASIS-2 and simulated datasets and Table 4.4 lists the average values obtained across folds with their standard deviation.

Mean Absolute Error (MAE)

Across both datasets, we observe that in the language-infused memory capacity task, the reservoir-enhanced MemoGNN consistently displays a lower MAE compared to the baseline GConvGRU model. While the OASIS-2 dataset shows less clear outperformance, the average across folds indicates that the MemoGNN trained with the language-infused memory capacity task still surpasses the baseline model. In the simulated dataset, the reservoir-enhanced MemoGNN outperformed the baseline across all folds. This result contrasts the results we obtained when training the MemoGNN models on the regular memory task.

Models	Dataset	MAE (T1)	MAE (T2)	MAE of Mem Cap (T1)	MAE of Mem Cap (T2)
GConvGRU	OASIS-2	0.0430 ± 3.43e-3	0.0468 ± 3.42e-4	0.7149 ± 1.59e-1	0.6769 ± 1.68e-1
GConvGRU + Reservoir	OASIS-2	**0.0426 ± 3.98e-3**	**0.0464 ± 3.90e-4**	**0.5619 ± 8.68e-2**	**0.61698 ± 1.56e-1**
GConvGRU	Simulated	0.0323 ± 1.09e-3	0.0335 ± 6.10e-4	0.9297 ± 8.76e-2	1.0041 ± 2.16e-1
GConvGRU + Reservoir	Simulated	**0.0321 ± 1.14e-3**	**0.0333 ± 5.40e-4**	**0.7326 ± 5.48e-2**	**0.8003 ± 1.54e-1**

Table 4.4: *Comparison of MAE and MAE of Memory Capacity for GConvGRU and GConvGRU with Reservoir (MemoGNN) on OASIS-2 and Simulated Datasets*. The value for the model that performs the best is highlighted for the different datasets.

MAE of Memory Capacity

By using the MemoGNN models, we have improved the MAE of Memory Capacity across both datasets. From Figure 4.4, it is clear that across all folds and for both datasets, the MemoGNN model consistently achieves a better MAE of Memory Capacity performance, for both time points 1 and 2. This is in common with our results in the regular memory capacity task.

MAE of Network Topologies

In addition, we also analysed the network topology performances. Aside from a slight but not significant improvement in node strength for the MemoGNN across both datasets for both time points, no other consistent patterns were observed. In

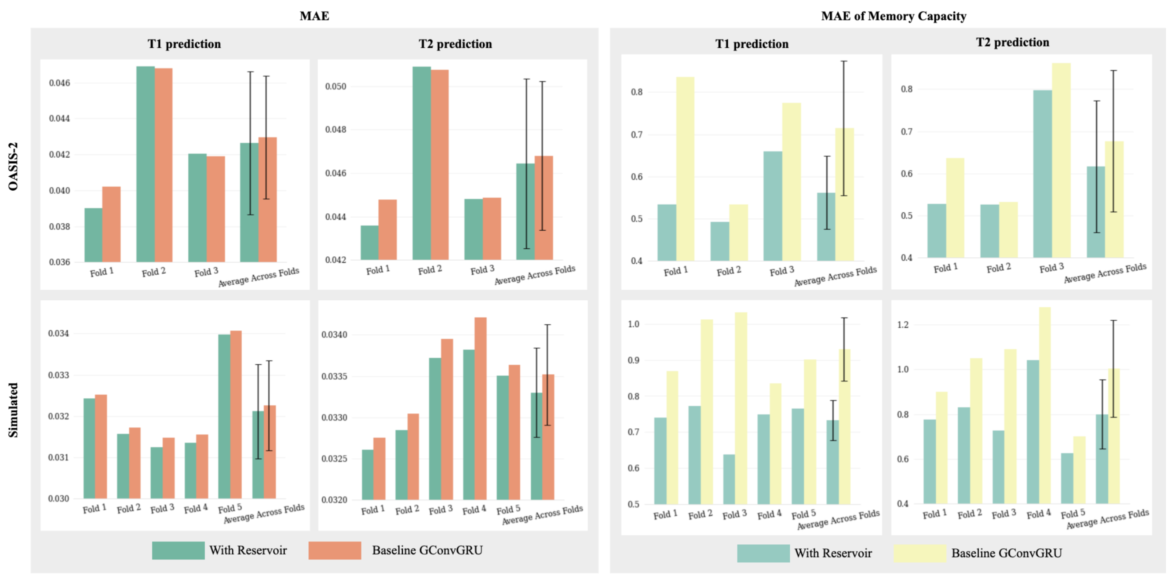


Figure 4.4: Comparison of MAE and MAE of memory capacity across folds for the language-infused memory capacity task. The graphs compare the performance of with and without reservoirs in predicting T1 and T2 across different folds for the OASIS-2 and simulated datasets.

particular, for the OASIS dataset, we saw an increase in the performance of node strength, betweenness centrality, global efficiency, and clustering with the MemoGNN architecture at both time points. The comparison plot of the performance across folds for the OASIS dataset is shown in Figure A.2. For the simulated dataset, across both time points, there was an increase in the performance of node strength, diversity coefficient, eigenvector centrality, modularity, and density with the MemoGNN architecture. The comparison plot of the performance across folds for the simulated dataset is found in Figure A.3.

4.6.2 Discussion

We have shown in our results that, a MemoGNN model, trained with the language-infused task, can outperform the baseline model in producing more accurate brain graphs, as well as a closer cognitive functionality to the real brain connectomes. One of the reasons could be that the word embeddings as inputs are more aligned with the way the human brain processes and stores information, compared to the randomly generated inputs. Another reason could be that the λ_2 is more well-chosen compared to the first experiment, allowing us to balance the L1 and the memory capacity loss better. Thus, in the future, a more rigorous selection of hyperparameters should be carried out. However, due to the huge amount of training time required, it is not feasible under the scope of this project.

Furthermore, unlike the regular memory capacity task, which showed a significant reduction in the error of the density of the generated graphs for both datasets, the network topology improvements in the language-infused memory task were not consistent across datasets. This inconsistency might be due to the language-infused memory capacity task having less impact on network topology. In contrast, the regular memory capacity task demonstrated a high correlation between several network topologies and memory capacity, as observed in our preliminary experiment in Section 1.3.2. The language-infused memory capacity might have revealed new insights about network topology in terms of biological representativeness compared to existing network measurements, thereby allowing us to improve the MAE of the generated graphs. It would be interesting to further explore the nature of language-infused memory by conducting similar experiments as those in Section 1.3.2, along with additional experiments that could help us understand its impact on network topology and biological representativeness.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In conclusion, this research introduces MemoGNN, the first memory-aware graph neural network model designed to forecast 4D brain connectivity. By addressing the gap in the literature, our work focuses on generating brain connectomes that are cognitively aligned with actual brain networks. We evaluated the performance of MemoGNN using two distinct memory capacity tasks: the regular memory capacity task and a newly proposed language-infused memory capacity task. Experimental results from various datasets, including the OASIS-2 and simulated datasets, demonstrate MemoGNN's effectiveness in predicting brain connectivities that closely align with actual cognitive functions.

Our findings show that MemoGNN is capable of significantly improving the MAE of memory capacity, indicating its proficiency in capturing and replicating the cognitive functionalities of brain graphs. In the regular memory capacity task, while the MAE of the generated graphs did not improve beyond the baseline, there was an enhancement in network topology measurements, particularly in density. This improvement is likely due to the strong correlation between density and regular memory capacity, shown in our preliminary experiments. On the other hand, MemoGNN models trained with the language-infused memory capacity task exhibited a notable reduction in MAE across both datasets. This suggests that language-based inputs might provide a more biologically representative method for predicting brain activity patterns, offering new insights into brain graphs beyond existing network topology measures.

Despite these promising results, integrating biologically instantiated reservoirs results in increased computational and memory costs. This means that further optimisation of the MemoGNN model should be carried out to enhance its feasibility for practical applications. Future research should focus on fine-tuning hyperparameters, exploring alternative cognitive functionality tasks, and expanding MemoGNN to the usage of hypergraphs.

5.2 Future Works

While this study has demonstrated the potential of memory-aware GNNs in predicting brain graph evolution and improving the closeness of memory capacity to real brain graphs, several avenues remain for further exploration and improvement.

Improving the training time of MemoGNN The memory capacity loss function can significantly increase training time, impacting performance. Thus, thorough hyperparameter tuning could not be carried out. For future work, we propose leveraging the high correlation between network topology metrics and the regular memory capacity. By training a differentiable model to predict memory capacity based on these network metrics, we can generate a synthetic memory capacity that approximates the true value but is computationally cheaper. Maybe with this new model and shorter training time, we would be able to fine-tune our model better to produce improved results.

Exploration of different cognitive functions and the impact of subdivisions of the brain The main focus of this study was on memory capacity. Thus, future research could explore other cognitive functions, such as attention or decision-making. Also, the impact of different subdivisions of the brain on cognitive task performance can be explored using the reservoir. Perhaps the integration of these approaches with the GNN will allow for deeper insights into brain connectivity.

Extension to hypergraphs using multiview brain connectomes Our research can be extended by exploring biologically inspired reservoir computing that incorporates high-order relationships, namely hyperconnectomes. Standard graph models, despite their effectiveness, often oversimplify the brain's network by limiting connections to one-to-one relationships, with each edge linking only two nodes. While this method is more straightforward, it may not fully encompass the brain's extensive multi-region interactions, thus restricting the depth of the model's learning capabilities. By including hyperconnectomes, perhaps a more holistic depiction of the brain's network can be achieved.

Bibliography

[1] Martijn P. van den Heuvel and Olaf Sporns. A cross-disorder connectome landscape of brain dysconnectivity. *Nature Reviews Neuroscience*, 20:435–446, 07 2019. ISSN 1471-0048. doi: 10.1038/s41583-019-0177-6. URL `https://doi.org/10.1038/s41583-019-0177-6`. pages 2

[2] Alaa Bessadok, Mohamed Ali Mahjoub, and Islem Rekik. Graph neural networks in network neuroscience, 2022. pages 2

[3] A Jon Stoessl. Neuroimaging in the early diagnosis of neurodegenerative disease. *Translational Neurodegeneration*, 1(1), 01 2012. ISSN 2047-9158. doi: 10.1186/2047-9158-1-5. URL `http://dx.doi.org/10.1186/2047-9158-1-5`. pages 2

[4] Ahmed Nebli, Ugur Ali Kaplan, and Islem Rekik. Deep evographnet architecture for time-dependent brain graph data synthesis from a single timepoint, 2020. pages 2, 21

[5] Alpay Tekin, Ahmed Nebli, and Islem Rekik. Recurrent brain graph mapper for predicting time-dependent brain graph evaluation trajectory, 2021. pages 2, 3, 21, 37

[6] Oytun Demirbilek and Islem Rekik. Predicting the evolution trajectory of population-driven connectional brain templates using recurrent multigraph neural networks. *Medical Image Analysis*, 83:102649, 2023. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2022.102649. URL `https://www.sciencedirect.com/science/article/pii/S1361841522002778`. pages 2, 21

[7] Zeynep Gürler and Islem Rekik. Federated brain graph evolution prediction using decentralized connectivity datasets with temporally-varying acquisitions. *IEEE Trans. Medical Imaging*, 42(7):2022–2031, 2023. URL `http://dblp.uni-trier.de/db/journals/tmi/tmi42.html#GurlerR23`. pages 2, 21

[8] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.03.005. URL `https://www.sciencedirect.com/science/article/pii/S0893608019300784`. pages 3, 16, 22

[9] Manuel Molano-Mazon, Joao Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jérémy Forest, Jorge Lerida, Li Ji-An, Christopher Cueva, Jaime Rocha, Devika Narain, and Guangyu Yang. Neurogym: An open resource for developing and sharing neuroscience tasks. 02 2022. doi: 10.31234/osf.io/aqc9n. pages 3

[10] Fabrizio Damicelli, Claus C. Hilgetag, and Alexandros Goulas. Brain connectivity meets reservoir computing. *bioRxiv*, 2021. doi: 10.1101/2021.01.22.427750. URL `https://www.biorxiv.org/content/early/2021/01/23/2021.01.22.427750`. pages 3, 22

[11] Laura E. Suárez, Agoston Mihalik, Filip Milisav, Kenji Marshall, Mingze Li, Petra E. Vértes, Guillaume Lajoie, and Bratislav Misic. conn2res: A toolbox for connectome-based reservoir computing. *bioRxiv*, 2023. doi: 10.1101/2023.05.31.543092. URL `https://www.biorxiv.org/content/early/2023/06/04/2023.05.31.543092`. pages 3, 23

[12] Herbert Jaeger. Short term memory in echo state networks. 01 2002. pages 3, 27

[13] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks, 2016. pages 3, 36

[14] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks, 2018. pages 3, 37

[15] Markus D. Schirmer, Archana Venkataraman, Islem Rekik, Minjeong Kim, Stewart H. Mostofsky, Mary Beth Nebel, Keri Rosch, Karen Seymour, Deana Crocetti, Hassna Irzan, Michael Hütel, Sebastien Ourselin, Neil Marlow, Andrew Melbourne, Egor Levchenko, Shuo Zhou, Mwiza Kunda, Haiping Lu, Nicha C. Dvornek, Juntang Zhuang, Gideon Pinto, Sandip Samal, Jennings Zhang, Jorge L. Bernal-Rusiel, Rudolph Pienaar, and Ai Wern Chung. Neuropsychiatric disease classification using functional connectomics - results of the connectomics in neuroimaging transfer learning challenge. *Medical Image Analysis*, 70:101972, 2021. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2021.101972. URL `https://www.sciencedirect.com/science/article/pii/S1361841521000189`. pages 5

[16] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10 (3):186–198, 02 2009. ISSN 1471-0048. doi: 10.1038/nrn2575. URL `http://dx.doi.org/10.1038/nrn2575`. pages 12

[17] Jinhui Wang, Xi-Nian Zuo, and Yong He. Graph-based network analysis of resting-state functional mri. *Frontiers in systems neuroscience*, 4:16, 06 2010. doi: 10.3389/fnsys.2010.00016. pages 12, 13

[18] Gary H. Glover. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics of North America*, 22(2):133–139, 04 2011. ISSN 1042-3680. doi: 10.1016/j.nec.2010.11.001. URL `http://dx.doi.org/10.1016/j.nec.2010.11.001`. pages 12

[19] M. Anatürk, N. Demnitz, K.P. Ebmeier, and C.E. Sexton. A systematic review and meta-analysis of structural magnetic resonance imaging studies investigating cognitive and social activity levels in older adults. *Neuroscience and Biobehavioral Reviews*, 93:71–84, 10 2018. ISSN 0149-7634. doi: 10.1016/j.neubiorev.2018.06.012. URL `http://dx.doi.org/10.1016/j.neubiorev.2018.06.012`. pages 12

[20] Chun-Yi Zac Lo, Yong He, and Ching-Po Lin. Graph theoretical analysis of human brain structural networks. *Reviews in the Neurosciences*, 22(5):551–563, 2011. doi: doi:10.1515/RNS.2011.039. URL `https://doi.org/10.1515/RNS.2011.039`. pages 12

[21] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404: 132306, 2020. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2019.132306. URL `https://www.sciencedirect.com/science/article/pii/S0167278919305974`. pages 13

[22] Timothy P Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current Opinion in Neurobiology*, 55:82–89, 2019. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2019.01.011. URL `https://www.sciencedirect.com/science/article/pii/S0959438818302009`. Machine Learning, Big Data, and Neuroscience. pages 14

[23] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. *GMD-Forschungszentrum Informationstechnik, 2002.*, 5, 01 2002. pages 14

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. pages 15

[25] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. pages 16

[26] Herbert Jaeger. Short term memory in echo state networks. 01 2002. pages 16

[27] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3): 127–149, 2009. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2009.03.005. URL `https://www.sciencedirect.com/science/article/pii/S1574013709000173`. pages 16

[28] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001. URL `https://www.sciencedirect.com/science/article/pii/S2666651021000012`. pages 18

[29] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016. URL `https://arxiv.org/abs/1609.02907`. pages 19

[30] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017. pages 20

[31] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1), 09 2019. ISSN 2197-4314. doi: 10.1186/s40649-019-0069-y. URL `http://dx.doi.org/10.1186/s40649-019-0069-y`. pages 20

[32] Peter F. Dominey. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics*, 73(3):265–274, 08 1995. ISSN 1432-0770. doi: 10.1007/bf00201428. URL `http://dx.doi.org/10.1007/BF00201428`. pages 22

[33] Peter Ford Dominey and Franck Ramus. Neural network processing of natural language: I. sensitivity to serial, temporal and abstract structure of language in the infant. *Language and Cognitive Processes*, 15(1):87–127, 02 2000. ISSN 1464-0732. doi: 10.1080/016909600386129. URL `http://dx.doi.org/10.1080/016909600386129`. pages 22

[34] Peter F. Dominey and Toshio Inui. Cortico-striatal function in sentence comprehension: Insights from neurophysiology and modeling. *Cortex*, 45(8): 1012–1018, 09 2009. ISSN 0010-9452. doi: 10.1016/j.cortex.2009.03.007. URL `http://dx.doi.org/10.1016/j.cortex.2009.03.007`. pages 22

[35] Xavier Hinaut and Peter Ford Dominey. Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing. *PLOS ONE*, 8(2):1–18, 02 2013. doi: 10.1371/journal.pone.0052946. URL `https://doi.org/10.1371/journal.pone.0052946`. pages 22

[36] Laura E. Suárez, Blake A. Richards, Guillaume Lajoie, and Bratislav Misic. Learning function from structure in neuromorphic networks. *bioRxiv*, 2020. doi: 10.1101/2020.11.10.350876. URL `https://www.biorxiv.org/content/early/2020/11/11/2020.11.10.350876`. pages 23, 45

[37] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 2004. pages 31

[38] Roger Guimerà and Luís A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005. doi: 10.1038/nature03288. URL `https://doi.org/10.1038/nature03288`. pages 31

[39] Mikail Rubinov and Olaf Sporns. Weight-conserving characterization of complex functional brain networks. *NeuroImage*, 56(4):2068–2079, 2011. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2011.03.069. URL `https://www.sciencedirect.com/science/article/pii/S105381191100348X`. pages 32

[40] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163 – 177, 2001. doi: 10.1080/0022250X.2001.9990249. pages 32

[41] Mark Newman. *Networks: An Introduction*. Oxford University Press, 03 2010. ISBN 9780199206650. doi: 10.1093/acprof:oso/9780199206650.001.0001. URL `https://doi.org/10.1093/acprof:oso/9780199206650.001.0001`. pages 32

[42] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Phys. Rev. Lett.*, 87:198701, Oct 2001. doi: 10.1103/PhysRevLett.87.198701. URL `https://link.aps.org/doi/10.1103/PhysRevLett.87.198701`. pages 33

[43] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, Mar 2008. doi: 10.1103/PhysRevLett.100.118703. URL `https://link.aps.org/doi/10.1103/PhysRevLett.100.118703`. pages 34

[44] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52(3):1059–1069, 2010. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2009.10.003. URL `https://www.sciencedirect.com/science/article/pii/S105381190901074X`. Computational Models of the Brain. pages 35

[45] D. S. Marcus, A. F. Fotenos, J. G. Csernansky, J. C. Morris, and R. L. Buckner. Open access series of imaging studies: longitudinal MRI data in nondemented and demented older adults. *J Cogn Neurosci*, 22(12):2677–2684, Dec 2010. pages 35

Appendix A

Additional Results

Models	**MAE of Node Strength**		**MAE of Participation Coef**		**MAE of Diversity Coef**		**MAE of Betweenness C**		**MAE of Eigenvector C**	
	t1	t2	t1	t2	t1	t2	t1	t2	t1	t2
GConvGRU	**0.4525 ± 1.81e-1**	0.7425 ± 2.28e-1	0.0070 ± 1.15e-3	0.0073 ± 1.15e-3	0.0352 ± 2.07e-3	0.0347 ± 4.16e-3	**72.00 ± 1.60**	**76.70 ± 1.77**	**0.0025 ± 5.21e-4**	**0.0032 ± 2.83e-4**
GConvGRU + Reservoir	0.5774 ± 1.92e-1	**0.6781 ± 2.56e-1**	**0.0057 ± 2.14e-3**	**0.0064 ± 7.92e-4**	**0.0237 ± 8.25e-3**	**0.0256 ± 5.73e-3**	77.40 ± 1.72	82.40 ± 2.90	0.0031 ± 5.69e-4	0.0043 ± 5.14e-4
GConvLSTM	**0.7571 ± 1.80e-1**	**0.9738 ± 2.72e-1**	0.0035 ± 1.57e-3	0.0039 ± 5.13e-4	0.0140 ± 7.35e-3	0.0139 ± 7.82e-3	78.24 ± 2.44	**83.20 ± 3.22**	0.0033 ± 7.55e-4	0.0051 ± 4.31e-4
GConvLSTM + Reservoir	0.8518 ± 8.04e-2	1.0367 ± 2.08e-1	**0.0025 ± 1.06e-3**	**0.0030 ± 4.53e-4**	**0.0104 ± 2.22e-3**	**0.0097 ± 3.16e-3**	**77.86 ± 1.69**	83.37 ± 3.59	**0.0032 ± 6.61e-4**	**0.0049 ± 4.60e-4**
TAGConv	0.4298 ± 1.22e-1	0.6094 ± 1.85e-1	**0.0076 ± 1.57e-2**	**0.0091 ± 1.29e-3**	**0.0380 ± 8.37e-3**	**0.0435 ± 1.00e-2**	75.56 ± 2.43	81.25 ± 4.89	**0.0032 ± 6.64e-4**	**0.0051 ± 5.75e-4**
TAGConv + Reservoir	**0.4068 ± 9.60e-2**	**0.5294 ± 1.18e-1**	0.0079 ± 2.44e-4	0.0096 ± 1.69e-3	0.0405 ± 2.17e-3	0.0463 ± 6.45e-3	**74.94 ± 2.58**	**80.79 ± 3.22**	0.0034 ± 4.97e-4	0.0053 ± 4.01e-4
RBGM	**0.5720 ± 1.56e-1**	**1.0142 ± 3.41e-1**	0.0036 ± 6.00e-4	0.0078 ± 5.05e-3	0.0158 ± 2.80e-3	**0.0355 ± 2.56e-2**	**59.22 ± 1.43**	**55.02 ± 2.73**	**0.0040 ± 1.87e-4**	0.0046 ± 4.81e-4
RBGM + Reservoir	0.5968 ± 1.31e-1	1.1558 ± 4.41e-1	**0.0030 ± 7.68e-4**	**0.0075 ± 3.61e-3**	**0.0119 ± 2.37e-3**	0.0374 ± 2.22e-2	59.37 ± 1.77	55.77 ± 5.69e-1	0.0041 ± 4.37e-4	**0.0044 ± 4.24e-4**

Table A.1: Comparison on node-wise network topology on **OASIS-2** dataset

Models	**MAE of Global Efficiency**		**MAE of Local Efficiency**	
	t1	t2	t1	t2
GConvGRU	**0.0413** ± **9.73e-3**	**0.0581** ± **1.64e-2**	**0.0105** ± **3.43e-3**	**0.0151** ± **5.54e-3**
GConvGRU + Reservoir	0.0515 ± 8.85e-3	0.0664 ± 1.40e-2	0.0116 ± 3.33e-3	0.0149 ± 4.73e-3
GConvLSTM	**0.0587** ± **7.21e-3**	**0.0741** ± **1.58e-2**	**0.0120** ± **2.12e-3**	**0.0168** ± **4.70e-3**
GConvLSTM + Reservoir	0.0622 ± 1.03e-2	0.0767 ± 1.31e-2	0.0132 ± 1.74e-3	0.0175 ± 4.38e-3
TAGConv	0.0414 ± 1.12e-2	0.0572 ± 1.65e-2	0.0101 ± 1.57e-3	0.0143 ± 2.10e-3
TAGConv + Reservoir	**0.0399** ± **1.21e-2**	**0.0553** ± **1.06e-2**	**0.0096** ± **1.60e-3**	**0.0128** ± **1.54e-3**
RBGM	**0.0399** ± **1.21e-2**	**0.0644** ± **1.17e-2**	**0.0134** ± **2.56e-3**	**0.0245** ± **9.57e-3**
RBGM + Reservoir	0.0444 ± 1.37e-2	0.0704 ± 1.59e-2	0.0132 ± 2.34e-3	0.0228 ± 5.79e-3

Table A.2: Comparison of efficiency-wise network topology on **OASIS-2 dataset**

Models	**MAE of Modularity**		**MAE of Density**		**MAE of Clustering**	
	t1	t2	t1	t2	t1	t2
GConvGRU	0.0040 ± 5.60e-4	**0.0030** ± **3.71e-4**	0.0548 ± 1.08e-3	0.0533 ± 1.81e-3	**0.0098** ± **2.27e-3**	**0.0129** ± **4.20e-3**
GConvGRU + Reservoir	**0.0028** ± **2.74e-4**	0.0035 ± 3.71e-4	**0.0491** ± **1.52e-3**	**0.0491** ± **1.55e-3**	0.0111 ± 3.38e-3	0.0141 ± 5.12e-3
GConvLSTM	**0.0031** ± **3.52e-4**	**0.0033** ± **9.82e-4**	0.0288 ± 1.87e-2	0.0323 ± 2.01e-2	**0.0132** ± **3.81e-3**	**0.0174** ± **5.68e-3**
GConvLSTM + Reservoir	0.0034 ± 6.77e-4	0.0035 ± 7.61e-4	**0.0252** ± **4.01e-3**	**0.0299** ± **7.22e-3**	0.0153 ± 1.35e-3	0.0187 ± 3.89e-3
TAGConv	0.0033 ± 9.65e-4	**0.0033** ± **6.62e-4**	0.0617 ± 7.26e-3	0.0622 ± 7.65e-3	0.0098 ± 1.05e-3	0.0137 ± 2.60e-3
TAGConv + Reservoir	**0.0032** ± **1.10e-4**	0.0039 ± 9.97e-4	**0.0600** ± **5.53e-3**	**0.0601** ± **6.69e-3**	**0.0097** ± **1.49e-3**	**0.0126** ± **8.36e-4**
RBGM	0.0094 ± 1.51e-3	**0.0064** ± **1.39e-3**	0.0509 ± 9.65e-3	0.0391 ± 1.30e-2	**0.0131** ± **2.59e-3**	**0.0236** ± **7.78e-3**
RBGM + Reservoir	**0.0089** ± **3.77e-4**	0.0074 ± 1.25e-3	**0.0471** ± **8.42e-3**	**0.0336** ± **7.06e-3**	0.0145 ± 5.19e-3	0.0274 ± 1.42e-2

Table A.3: Comparison of global-level network topology on **OASIS-2 dataset**

Models	**MAE of Node Strength**		**MAE of Participation Coef**		**MAE of Diversity Coef**		**MAE of Betweenness C**		**MAE of Eigenvector C**	
	t1	t2	t1	t2	t1	t2	t1	t2	t1	t2
GConvGRU	**0.2819** ± **2.78e-2**	**0.3622** ± **4.79e-2**	0.2112 ± 1.61e-2	0.2268 ± 3.78e-3	0.1412 ± 1.14e-2	0.1485 ± 1.35e-3	253.34 ± 12.22	**264.48** ± **7.50**	**0.0034** ± **3.40e-4**	**0.0045** ± **3.80e-4**
GConvGRU + Reservoir	0.3527 ± 3.77e-2	0.4360 ± 3.50e-2	**0.2106** ± **1.60e-2**	**0.2264** ± **3.92e-3**	**0.1315** ± **2.29e-4**	**0.1458** ± **2.09e-3**	**252.20** ± **12.12**	264.52 ± 7.55	0.0039 ± 2.96e-4	0.0046 ± 3.21e-4
GConvLSTM	**0.3105** ± **5.00e-2**	**0.3884** ± **5.42e-2**	0.2103 ± 1.66e-2	0.2261 ± 4.03e-3	0.1349 ± 1.65e-2	0.1437 ± 6.11e-3	255.18 ± 11.35	**267.94** ± **7.50**	**0.0037** ± **4.79e-4**	**0.0049** ± **5.40e-4**
GConvLSTM + Reservoir	0.4635 ± 7.30e-2	0.5457 ± 6.53e-2	**0.2097** ± **1.59e-2**	**0.2261** ± **5.04e-3**	**0.1301** ± **1.40e-2**	**0.1419** ± **1.01e-2**	**254.53** ± **11.89**	268.18 ± 7.04	0.0039 ± 6.38e-4	0.0050 ± 9.45e-4
TAGConv	**0.2897** ± **3.20e-2**	**0.3020** ± **3.79e-2**	0.2114 ± 1.60e-2	0.2289 ± 3.75e-3	0.1419 ± 1.15e-2	0.1588 ± 3.72e-3	251.73 ± 12.97	267.06 ± 7.71	**0.0036** ± **2.90e-4**	**0.0052** ± **1.76e-4**
TAGConv + Reservoir	0.3463 ± 3.89e-2	0.3681 ± 3.59e-2	**0.2108** ± **1.59e-2**	**0.2283** ± **3.75e-3**	**0.1385** ± **1.12e-2**	**0.1549** ± **2.15e-2**	**248.63** ± **11.92**	**264.69** ± **8.23**	0.0038 ± 1.94e-4	0.0053 ± 1.48e-4
RBGM	**0.3357** ± **2.21e-2**	**0.4456** ± **7.52e-2**	0.2102 ± 1.61e-2	0.2271 ± 4.01e-3	0.1330 ± 1.13e-2	0.1457 ± 4.51e-3	241.01 ± 11.35	255.19 ± 10.31	**0.0042** ± **2.37e-4**	0.0051 ± 2.15e-4
RBGM + Reservoir	0.3533 ± 2.27e-2	0.4770 ± 1.11e-1	**0.2101** ± **1.59e-2**	**0.2268** ± **4.01e-3**	**0.1320** ± **1.03e-2**	**0.1439** ± **4.25e-3**	**238.05** ± **12.54**	**250.32** ± **6.49**	0.0043 ± 2.72e-4	**0.0049** ± **3.90e-4**

Table A.4: Comparison on node-wise network topology on **simulated dataset**

Models	**MAE of Global Efficiency**		**MAE of Local Efficiency**	
	t1	t2	t1	t2
GConvGRU	0.1004 ± 1.86e-3	0.0960 ± 9.43e-4	0.0696 ± 2.22e-3	0.0728 ± 1.64e-2
GConvGRU + Reservoir	**0.0965** ± **1.83e-3**	**0.0926** ± **1.43e-3**	**0.0669** ± **2.01e-3**	**0.0703** ± **1.69e-2**
GConvLSTM	0.0984 ± 3.46e-3	0.0937 ± 4.06e-3	0.0698 ± 1.57e-3	0.0733 ± 1.72e-2
GConvLSTM + Reservoir	**0.0890** ± **5.04e-3**	**0.0836** ± **5.16e-3**	**0.0649** ± **2.39e-3**	**0.0689** ± **1.78e-2**
TAGConv	0.0984 ± 3.17e-3	0.0946 ± 2.45e-3	0.0702 ± 2.30e-3	0.0774 ± 1.90e-2
TAGConv + Reservoir	**0.0951** ± **2.80e-3**	**0.0907** ± **2.44e-3**	**0.0671** ± **1.92e-3**	**0.0737** ± **1.72e-2**
RBGM	0.0989 ± 2.53e-3	0.0942 ± 3.04e-3	0.0719 ± 2.46e-3	0.0764 ± 1.78e-2
RBGM + Reservoir	**0.0968** ± **1.74e-2**	**0.0898** ± **2.75e-3**	**0.0698** ± **1.93e-3**	**0.0720** ± **1.52e-2**

Table A.5: Comparison of efficiency-wise network topology on **simulated dataset**

Models	**MAE of Modularity**		**MAE of Density**		**MAE of Clustering**	
	t1	t2	t1	t2	t1	t2
GConvGRU	0.0036 ± 1.05e-3	0.0023 ± 8.41e-4	0.0477 ± 3.42e-3	0.0442 ± 1.59e-3	0.0106 ± 1.14e-3	0.0094 ± 7.00e-4
GConvGRU + Reservoir	**0.0031** ± **2.29e-4**	**0.0015** ± **2.30e-4**	**0.0119** ± **5.71e-3**	**0.0164** ± **1.43e-2**	**0.0088** ± **9.73e-4**	**0.0084** ± **1.14e-3**
GConvLSTM	0.0034 ± 8.91e-4	0.0032 ± 9.94e-4	0.0471 ± 3.08e-3	0.0469 ± 2.48e-3	0.0102 ± 1.62e-3	0.0094 ± 8.96e-4
GConvLSTM + Reservoir	**0.0030** ± **1.60e-3**	**0.0027** ± **1.63e-3**	**0.0394** ± **4.45e-3**	**0.0404** ± **4.75e-3**	**0.0083** ± **8.44e-4**	**0.0084** ± **1.22e-3**
TAGConv	**0.0039** ± **9.90e-4**	**0.0030** ± **1.19e-3**	0.0505 ± 2.79e-3	0.0541 ± 2.27e-3	0.0113 ± 1.93e-3	0.0126 ± 1.69e-3
TAGConv + Reservoir	0.0053 ± 9.71e-4	0.0040 ± 2.06e-3	**0.0478** ± **2.66e-3**	**0.0499** ± **2.73e-3**	**0.0097** ± **1.43e-3**	**0.0099** ± **3.26e-4**
RBGM	**0.0076** ± **2.15e-3**	0.0041 ± 1.14e-3	0.0460 ± 2.98e-3	0.0422 ± 4.07e-3	0.0123 ± 1.17e-3	0.0146 ± 9.66e-4
RBGM + Reservoir	0.0081 ± 3.13e-3	**0.0041** ± **7.38e-4**	**0.0439** ± **1.43e-3**	**0.0381** ± **2.85e-3**	**0.0119** ± **9.60e-4**	**0.0127** ± **1.17e-3**

Table A.6: Comparison of global-level network topology on **simulated dataset**

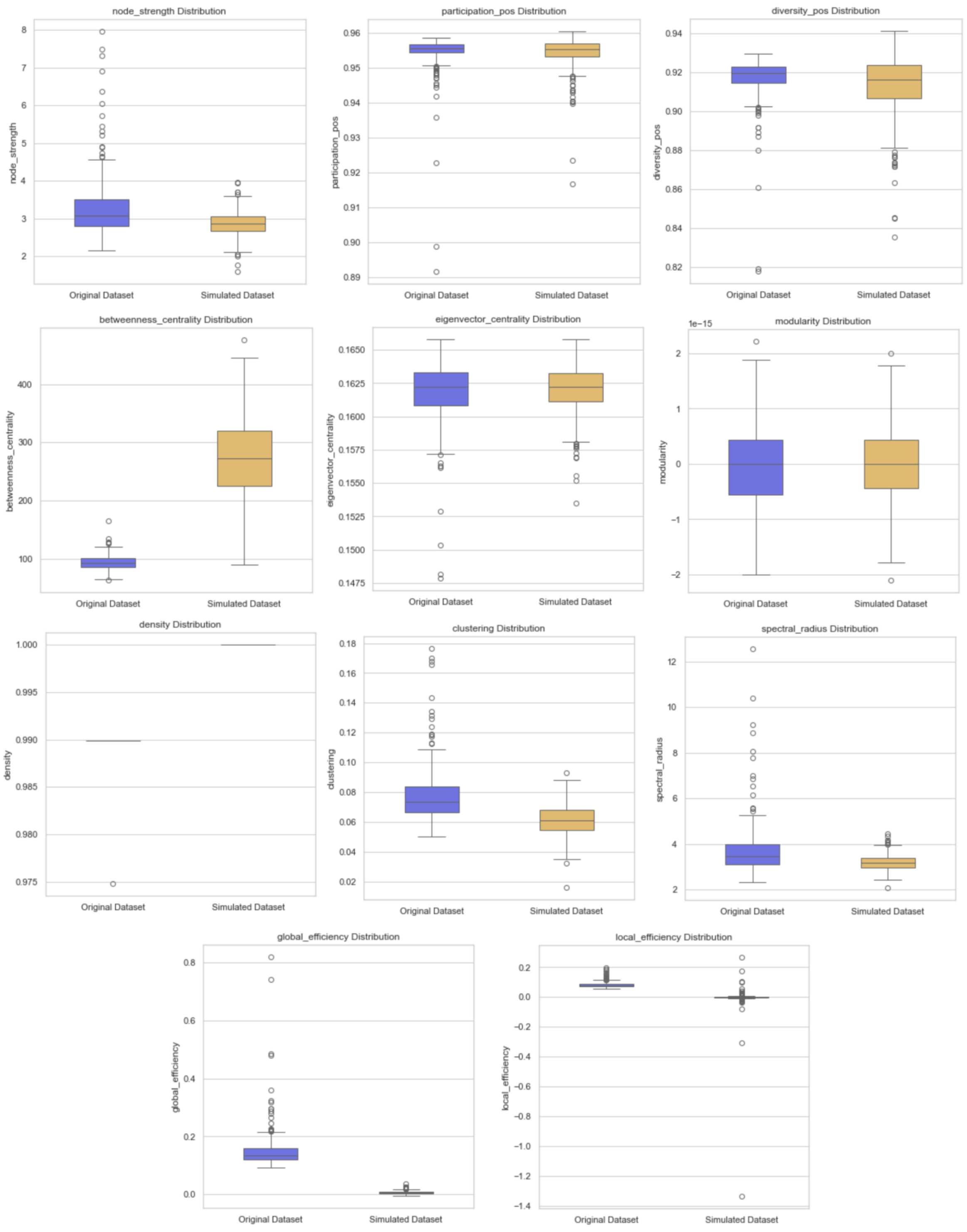


Figure A.1: Comparison of various topological distributions of the two datasets, original and simulated.

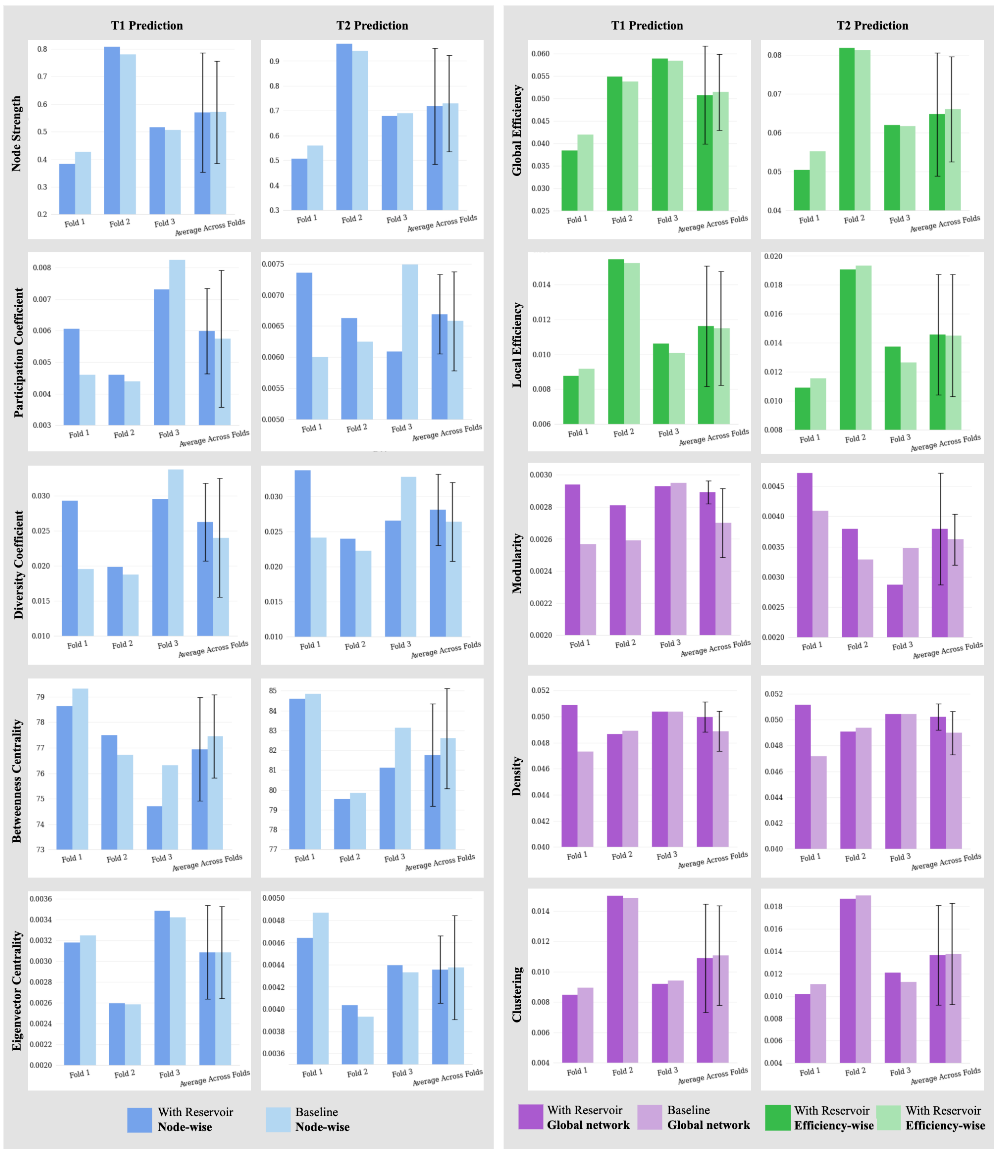


Figure A.2: Comparison of various topological measurements across folds for the language-infused memory capacity task. The graphs compare the performance of with and without reservoirs in predicting T1 and T2 across different folds for the **OASIS-2** dataset.

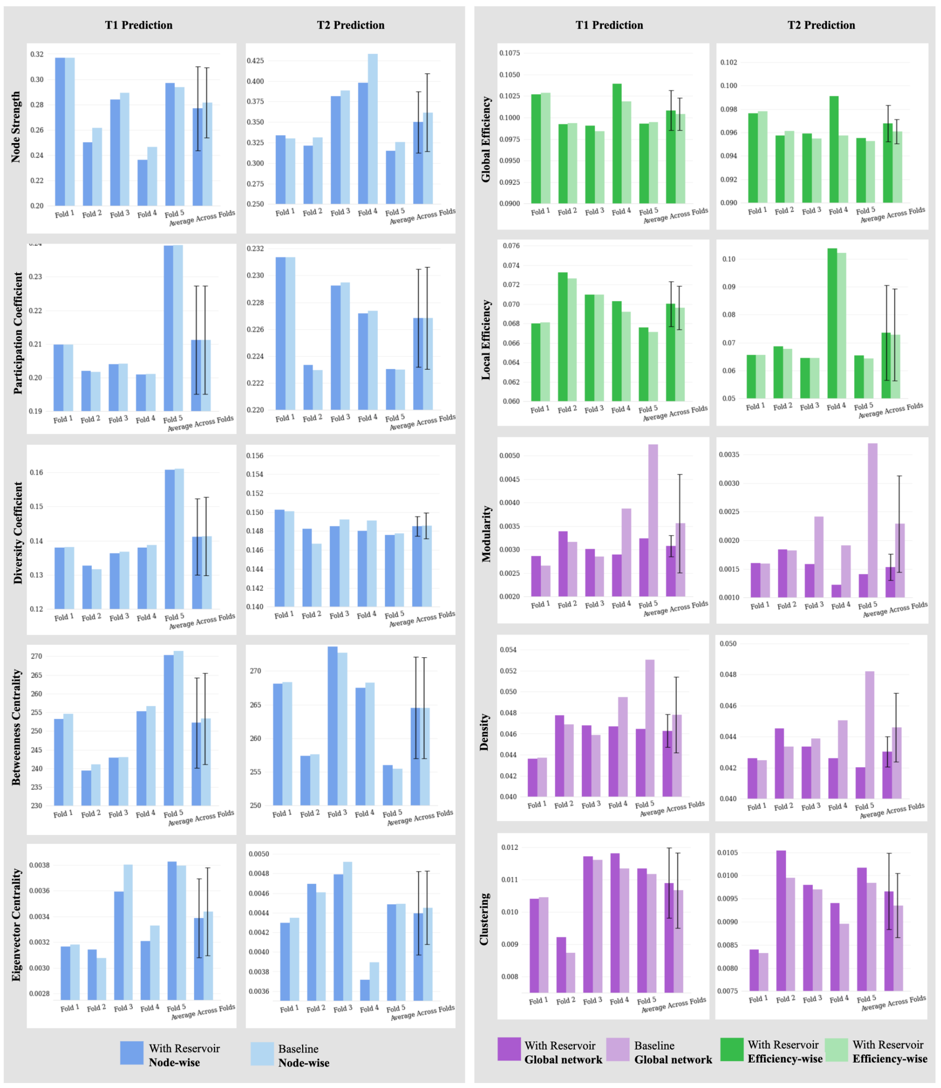


Figure A.3: Comparison of various topological measurements across folds for the language-infused memory capacity task. The graphs compare the performance of with and without reservoirs in predicting T1 and T2 across different folds for the **simulated** dataset.