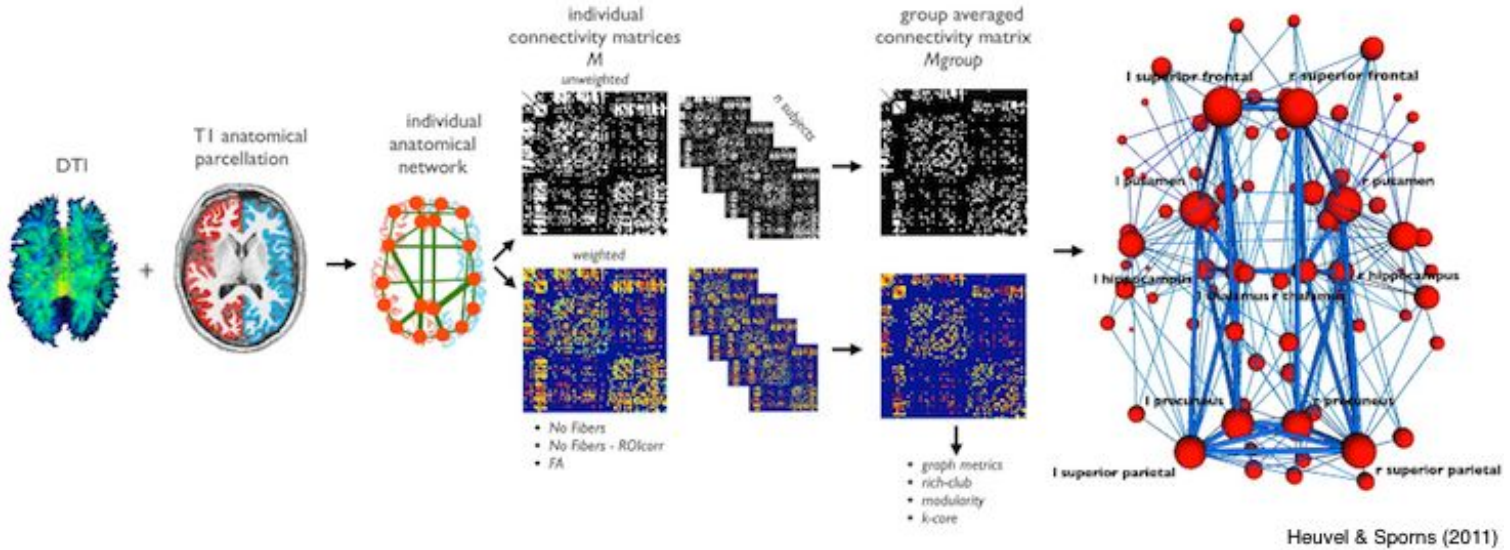
A graph neural network framework for causal inference in brain networks

Presented by: William Ashbee

Understanding Dynamic Brain Interactions

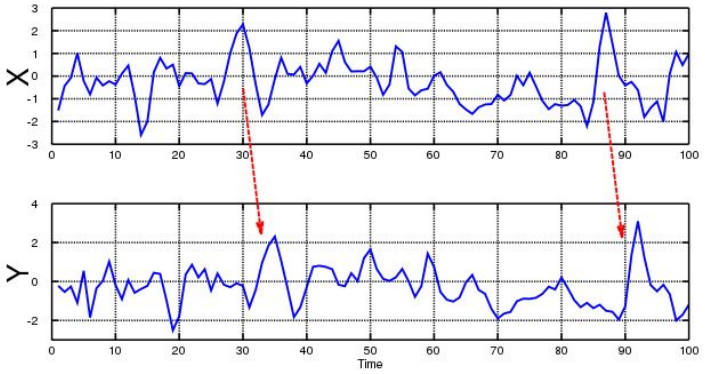


- A challenge in neuroscience:
  - Linking dynamic brain interactions to the static structural backbone.
  - Demonstrating causal relationships between brain regions from fMRI.

https://www.nature.com/scitable/blog/the-artful-brain/the_brain_part_1/

Causality: Granger Causality vs. Dynamic Causal Modeling

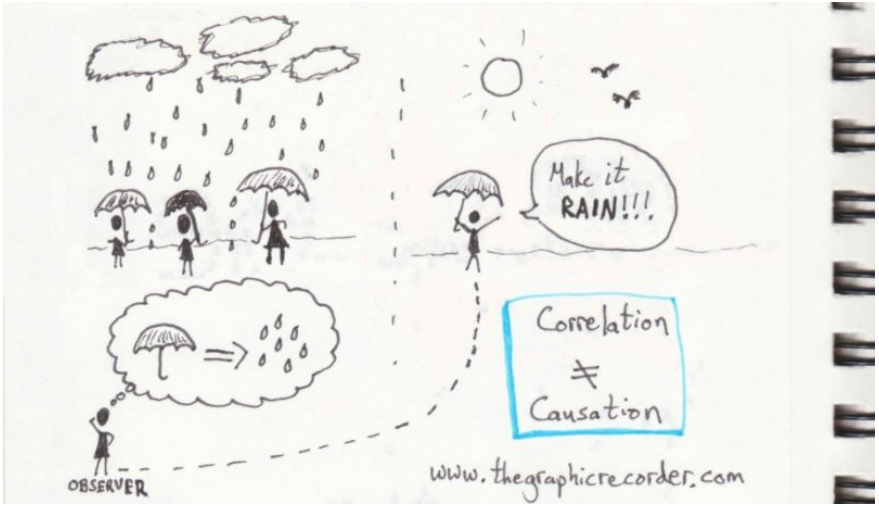
- Origin:
  - Concept by **Clive Granger**, adapted for neuroscience to explore temporal relationships between brain activities.
- Principle:
  - **If event A precedes and helps predict event B, A is considered to cause B.**
    - **The cause happens prior to its effect.**
    - **The cause has unique information about the future values of its effect.**
- Methodology:
  - Utilizes multivariate **vector auto regressive (VAR) models**.
  - **Test if knowledge of activity in one region improves prediction in another.**
  - Provides a measure for directed causal dependencies between brain areas.
- Applications:
  - Ideal for exploring **large-scale brain networks** due to its simplicity and predictive power.



When time series X Granger-causes time series Y, the patterns in X are approximately repeated in Y after some time lag (two examples are indicated with arrows). Thus, past values of X can be used for the prediction of future values of Y.

Weaknesses of Granger causality???

- Granger causality only detects statistical causality, not necessarily true causation. It merely indicates whether one time series helps in predicting another, but it doesn't imply a direct causal relationship between the variables.
- Direction of causality: Granger causality tests can only determine the direction of causality between two variables if there are no feedback loops or simultaneous causality. In many real-world scenarios, causal relationships may be more complex, involving **feedback loops and bidirectional causality**.



Maziarz, Mariusz. "A review of the Granger-causality fallacy." The journal of philosophical economics: Reflections on economic and social issues 8.2 (2015): 86-105.

Causality: Granger Causality vs. Dynamic Causal Modeling

- Concept:
  - Dynamic Causal Modelling (DCM) explores the effective connectivity between brain regions, revealing how these areas communicate and influence each other during different cognitive processes and in various health conditions.
- Methodology:
  - DCM employs a Bayesian framework to infer the directional interactions among neural populations, providing insights into the brain's functional architecture.
- Limitations:
  - The high computational complexity of DCM limits its analysis to predefined brain regions, which may omit critical areas of interest.
- Applications:
  - DCM offers profound insights into the brain’s causal relationships, crucial for targeted research in neurological and psychiatric disorders.
- Real-World Connection:
  - Consider the impact on individuals suffering from Alzheimer's disease as researchers use DCM to pinpoint the neural deteriorations leading to memory loss.

All models in DCM have the following basic form:

$$\dot{z} = f(z, u, \theta^{(n)})$$
$$y = g(z, \theta^{(h)}) + \epsilon$$

Causality: Granger Causality vs. Dynamic Causal Modeling - Model specification

Given:

1. $\dot{z} = f(z; \theta^{(n)}, u)$

- $\dot{z}$: Change in neural activity with respect to time.
- f: Neural function controlling the evolution of neural activity.
- $\theta^{(n)}$: Parameters of the neural function, e.g., connection strengths.
- u: Experimental inputs.

2. $y = g(z; \theta^{(h)}) + \varepsilon$

- y: Time series generated by neural activity.
- g: Observation function mapping neural activity to observable signals.
- $\theta^{(h)}$: Parameters of the observation function.
- ε: Additive observation noise.

Objective:
- Main interest in $\theta^{(n)}$, representing connection strengths that may change under different experimental conditions.

Equation 1 represents the underlying neural activity.
Equation 2 represents the time series observed as a function of that neural activity

The equations describe a mathematical model for understanding how neural activity evolves over time and is observed through imaging techniques. They aim to quantify the relationship between experimental inputs, the underlying neural dynamics, and the observable data, emphasizing the significance of neural parameters in interpreting brain connectivity and function under various conditions.

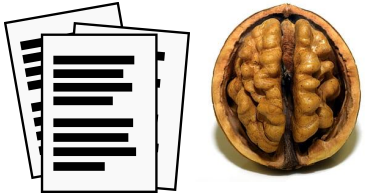
Understanding Causal Implications in Dynamic Causal Modeling (DCM)

- Key Components with Causal Roles
  - External Inputs **U**
    - Represent **experimental stimuli** or conditions.
    - Manipulation allows inference of causal effects on neural activity.
  - Neural Parameters **θ**
    - **Describe connection strengths** between brain regions.
    - **Quantify the causal influence of one region on another**.
- Mechanisms for Causal Inference
  - DCM integrates experimental manipulation, parameter estimation, and model testing to uncover the causal architecture of the brain.

All models in DCM have the following basic form:

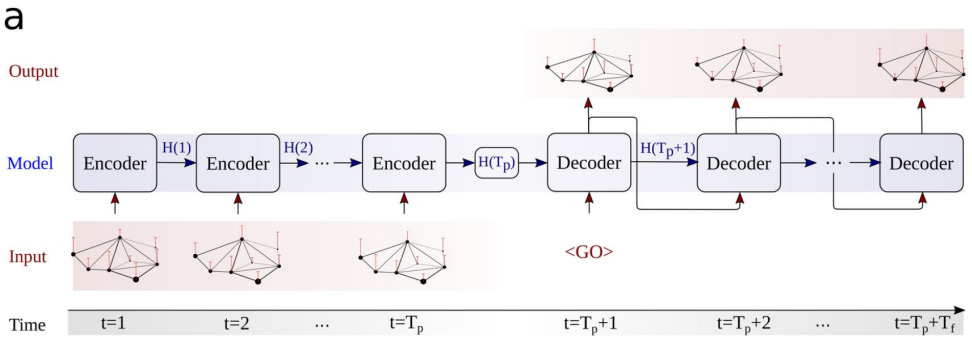
$$\dot{z} = f(z, u, \theta^{(n)})$$
$$y = g(z, \theta^{(h)}) + \epsilon$$

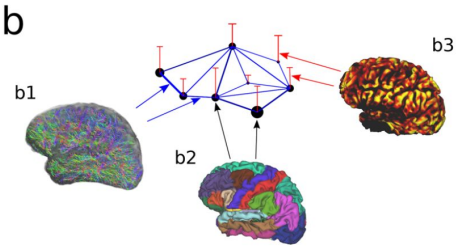
In a nutshell

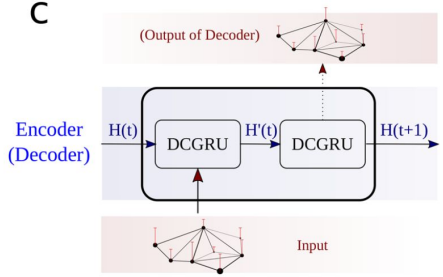


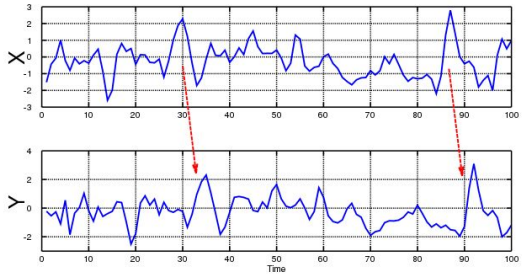
In studying causal relationships, authors use **linear methods e.g. Vector Autoregression** to predict intensities of **Bold Signal** in time from FMRI or we can use **nonlinear methods such as graph neural networks like DCGRU**.

Using **white matter tracts from DTI** provides added context to **improve predictions** of BOLD signals in regions of interest Extracted from Brain parcellation.



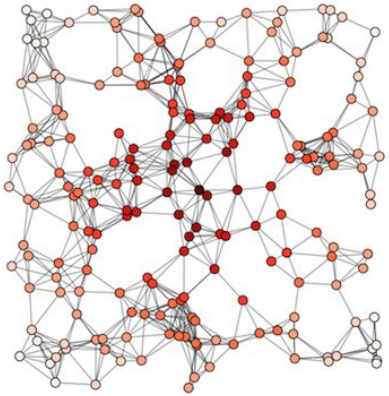


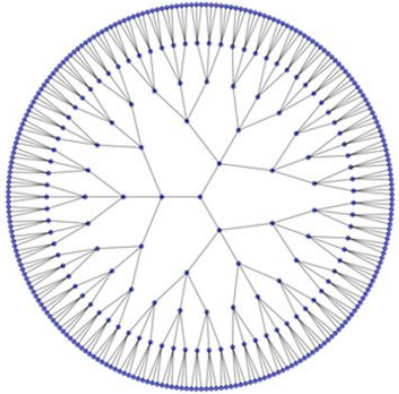


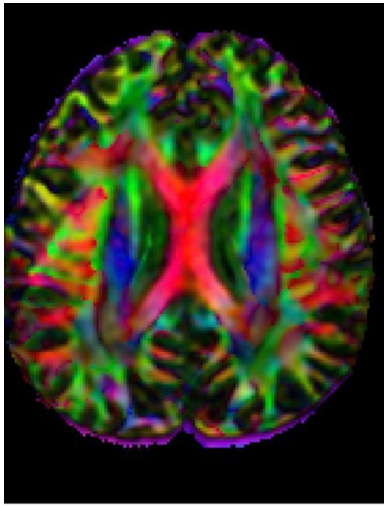


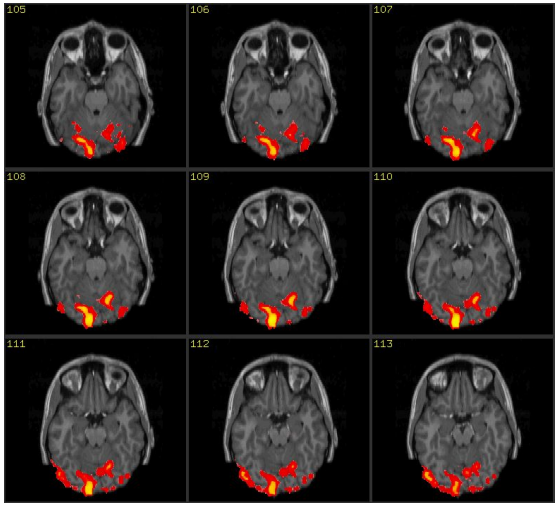
Birds eye view of paper

- Specialized GNNs process graph-structured spatio-temporal signals.
- Parcellated regions' averaged fMRI activity forms the nodes of a graph
- DTI connectivity provides the edges of the graph.
- Graph histories are used to predict future graphs (time series of graphs).

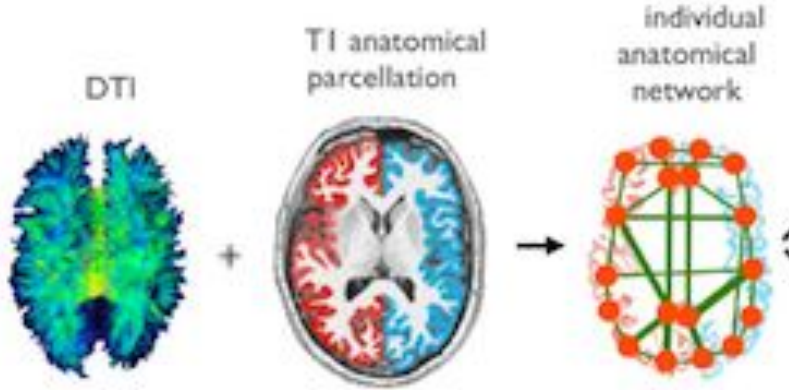




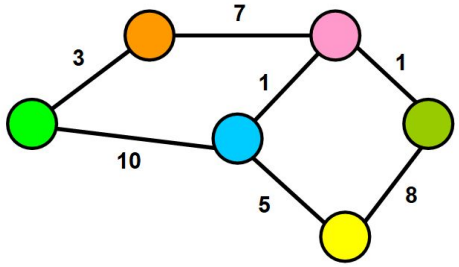


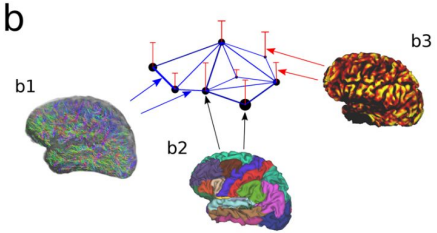


DTI



Graph Representation of Brain Networks





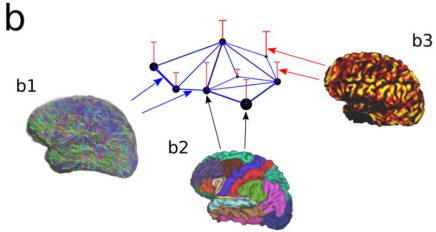
- Brain networks are represented as undirected graphs $G = (V, E, A_w)$, where:
  - V denotes the set of vertices or nodes (ROIs).
  - E represents the edges (axonal connections between ROIs).
  - $A_w \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix, encoding the structural connectivity.
- The weighted adjacency matrix A_w is derived from Diffusion Tensor Imaging (DTI), providing insights into the anatomical connection strengths.
- DTI data inform the edge weights $w_{nn'}$, reflecting the anatomical connectivity between nodes.

Data Matrix and ROIs

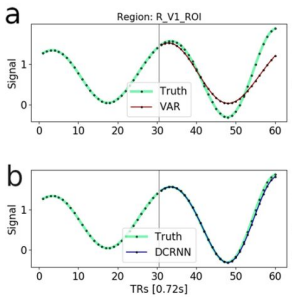
The voxel time series of brain activity maps are collected into data matrix $X = (x(1) \dots x(T))$, where:

- $x(t) \in \mathbb{R}^N$ represents the activation of all ROIs at time t
- N is the number of ROIs, derived from a brain atlas and represented by meta-voxels.
- T is the number of time points, capturing the activation time course of ROIs.

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1T} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NT} \end{pmatrix}$$



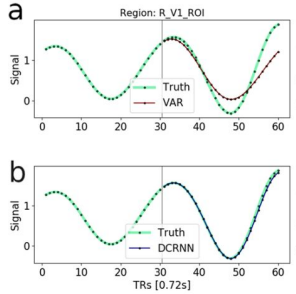
N Rois

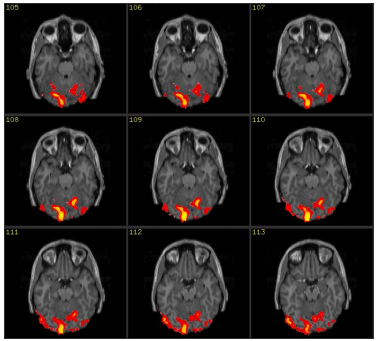


T Time courses

Modeling Activity Flow on the Graph

- The flow of neural activity is represented as a time-dependent graph signal $x(t) \in \mathbb{R}^N$, capturing the BOLD signal amplitude for each ROI.
- Forecasting this activity involves learning a function $h(...)$ that maps a sequence of past graph signals to future signals, effectively predicting the evolution of brain activity.
- This mapping considers both the graph structure and the temporal dynamics inherent in the neuroimaging data.





$$\left[\mathbf{x}\left(t-T_p+1\right),\ldots,\mathbf{x}(t);\mathcal{G}\right] \xrightarrow{h(...)} \left[\mathbf{x}(t+1),\ldots,\mathbf{x}\left(t+T_f\right)\right]. \quad (4)$$

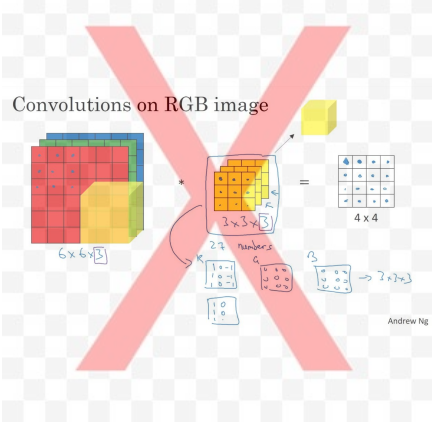
Stochastic Modeling of Information Flow

- Information flow within the brain graph G is modeled as a stochastic random walk, characterized by:
  - A re-start probability $\alpha \in [0, 1]$.
  - A state transition matrix $T = D^{-1}A_w$, where $D = \text{diag}(A_w\mathbf{1})$ is the degree matrix.
- The transition matrix T, reflecting normalized edge strengths, facilitates diffusion across the graph, modeling the spread of neural activity.
- Due to DTI's inability to determine directionality, the graph and its diffusion process are considered symmetric ($T = T^T$).

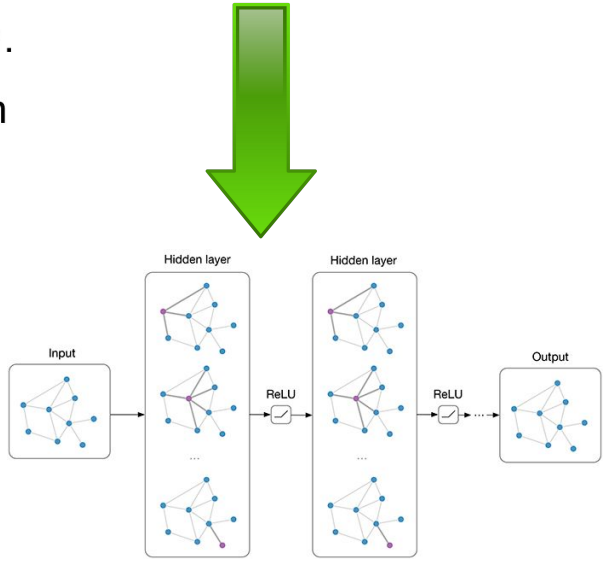
The Graph Convolution Operator

Graph Convolution in the Spectral Domain:

$$y_t = x_t *_G f_\theta = U(\theta_\omega \odot U^T x_t)$$

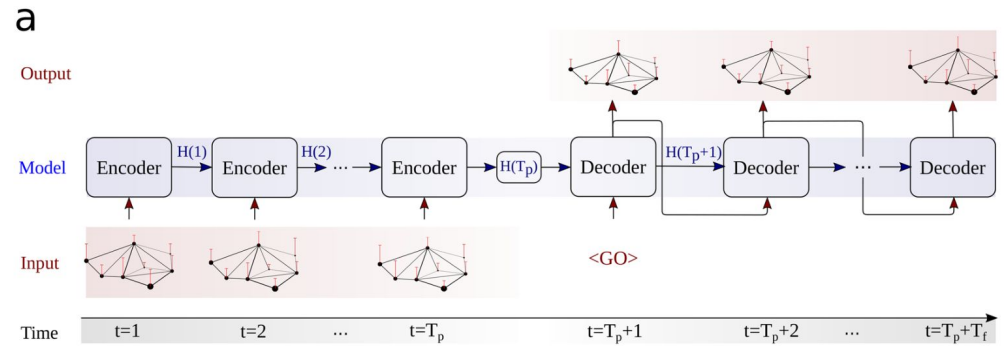


- x_t: Input signal on the graph nodes.
- f_θ: Filter parametrized by θ, applied in the spectral domain.
- U: Eigenvectors of the graph Laplacian, enabling the graph Fourier transform.
- θ_ω: Filter parameters in the spectral domain.
- $\odot$: Hadamard (element-wise) product, applied after transforming x_t and f_θ into the spectral domain.
- This operation allows filtering signals on graphs, capturing node interactions based on the graph structure.



DCGRU/DCRNN (Overview)

- The DCRNN integrates diffusion convolution within a recurrent neural network framework, tailored for graph-structured spatio-temporal data.
- It leverages a sequence-to-sequence architecture and scheduled sampling technique to accurately model and predict neural activity dynamics.
- Key components:
  - Diffusion convolution layers capture spatial dependencies across the ROIs.
  - Recurrent layers model the temporal evolution of neural activity.
  - The sequence-to-sequence model maps sequences of past activity to future activity, enabling accurate predictions of brain dynamics.
- This architecture is particularly suited for neuroimaging data, where understanding the flow of activity is crucial.



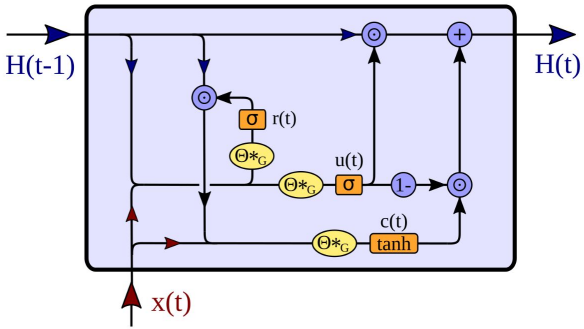


Figure 6. Overview of the processing steps of the DCGRU cell. The input $\mathbf{x}(t)$, as well as the previous hidden state $\mathbf{H}(t-1)$ are concatenated and passed to the reset gate $\mathbf{r}(t)$, as well as to the update gate $\mathbf{u}(t)$. The reset gate $\mathbf{r}(t)$ controls the proportion of $\mathbf{H}(t-1)$ which enters $\mathbf{c}(t)$, together with input $\mathbf{x}(t)$. Then the hidden state $\mathbf{H}(t-1)$ is updated by $\mathbf{c}(t)$, whereby the amount of new information is controlled by $\mathbf{u}(t)$.

DCGRU (Reset Gate)

The reset gate determines the amount of past information to forget. It is computed as follows:

$$r(t) = \sigma(\Theta_r *_G [x(t), H(t-1)] + b_r)$$

where:

- σ: Sigmoid activation function.
- Θ_r: Weight matrix for the reset gate.
- $*_G$: Diffusion convolution operation.
- $[x(t), H(t-1)]$: Concatenation of the current input and the previous hidden state.
- b_r: Bias term for the reset gate.

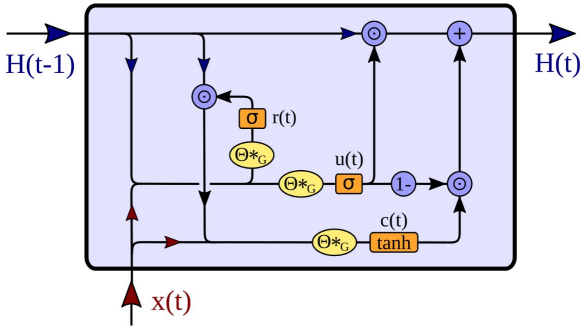


Figure 6. Overview of the processing steps of the DCGRU cell. The input $\mathbf{x}(t)$, as well as the previous hidden state $\mathbf{H}(t-1)$ are concatenated and passed to the reset gate $\mathbf{r}(t)$, as well as to the update gate $\mathbf{u}(t)$. The reset gate $\mathbf{r}(t)$ controls the proportion of $\mathbf{H}(t-1)$ which enters $\mathbf{c}(t)$, together with input $\mathbf{x}(t)$. Then the hidden state $\mathbf{H}(t-1)$ is updated by $\mathbf{c}(t)$, whereby the amount of new information is controlled by $\mathbf{u}(t)$.

$$\mathbf{r}(t) = \sigma(\boldsymbol{\Theta}_r *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_r)$$

$$\mathbf{u}(t) = \sigma(\boldsymbol{\Theta}_u *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_u)$$

$$\mathbf{c}(t) = \tanh(\boldsymbol{\Theta}_c *_G [\mathbf{x}(t), (\mathbf{r}(t) \odot \mathbf{H}(t-1))] + \mathbf{b}_c)$$

$$\mathbf{H}(t) = \mathbf{u}(t) \odot \mathbf{H}(t-1) + (1 - \mathbf{u}(t)) \odot \mathbf{c}(t),$$

DCGRU (Update Gate)

The update gate decides how much of the past information to keep and how much new information to add. It is given by:

$$u(t) = \sigma(\Theta_u *_G [x(t), H(t-1)] + b_u)$$

where:

- Θ_u: Weight matrix for the update gate.
- The rest of the symbols follow the same pattern as in the reset gate equation.

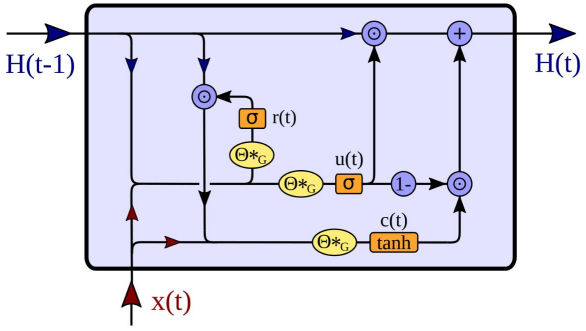


Figure 6. Overview of the processing steps of the DCGRU cell. The input $\mathbf{x}(t)$, as well as the previous hidden state $\mathbf{H}(t-1)$ are concatenated and passed to the reset gate $\mathbf{r}(t)$, as well as to the update gate $\mathbf{u}(t)$. The reset gate $\mathbf{r}(t)$ controls the proportion of $\mathbf{H}(t-1)$ which enters $\mathbf{c}(t)$, together with input $\mathbf{x}(t)$. Then the hidden state $\mathbf{H}(t-1)$ is updated by $\mathbf{c}(t)$, whereby the amount of new information is controlled by $\mathbf{u}(t)$.

$$\mathbf{r}(t) = \sigma(\boldsymbol{\Theta}_r *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_r)$$

$$\mathbf{u}(t) = \sigma(\boldsymbol{\Theta}_u *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_u)$$

$$\mathbf{c}(t) = \tanh(\boldsymbol{\Theta}_c *_G [\mathbf{x}(t), (\mathbf{r}(t) \odot \mathbf{H}(t-1))] + \mathbf{b}_c)$$

$$\mathbf{H}(t) = \mathbf{u}(t) \odot \mathbf{H}(t-1) + (1 - \mathbf{u}(t)) \odot \mathbf{c}(t),$$

DCGRU (Candidate Activation (c(t))

This computes the candidate hidden state which can be added to the memory, influenced by the reset gate:

$$c(t) = \tanh(\Theta_c *_G [x(t), (r(t) \odot H(t-1))] + b_c)$$

where:

- tanh: Hyperbolic tangent activation function.
- Θ_c: Weight matrix for the candidate state.
- $\odot$: Element-wise multiplication.

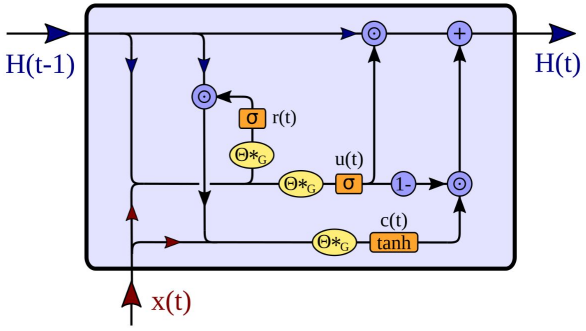


Figure 6. Overview of the processing steps of the DCGRU cell. The input $\mathbf{x}(t)$, as well as the previous hidden state $\mathbf{H}(t-1)$ are concatenated and passed to the reset gate $\mathbf{r}(t)$, as well as to the update gate $\mathbf{u}(t)$. The reset gate $\mathbf{r}(t)$ controls the proportion of $\mathbf{H}(t-1)$ which enters $\mathbf{c}(t)$, together with input $\mathbf{x}(t)$. Then the hidden state $\mathbf{H}(t-1)$ is updated by $\mathbf{c}(t)$, whereby the amount of new information is controlled by $\mathbf{u}(t)$.

$$\mathbf{r}(t) = \sigma(\mathbf{\Theta}_r *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_r)$$

$$\mathbf{u}(t) = \sigma(\mathbf{\Theta}_u *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_u)$$

$$\mathbf{c}(t) = \tanh(\mathbf{\Theta}_c *_G [\mathbf{x}(t), (\mathbf{r}(t) \odot \mathbf{H}(t-1))] + \mathbf{b}_c)$$

$$\mathbf{H}(t) = \mathbf{u}(t) \odot \mathbf{H}(t-1) + (1 - \mathbf{u}(t)) \odot \mathbf{c}(t),$$

DCGRU (Hidden State (H(t))

The new state of the unit, a combination of the old state and the new candidate activation, controlled by the update gate:

$$H(t) = u(t) \odot H(t-1) + (1 - u(t)) \odot c(t)$$

This equation ensures that the hidden state is updated by a combination of the previous state and the new candidate activation.

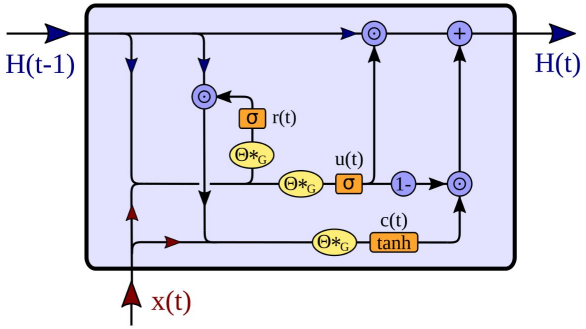


Figure 6. Overview of the processing steps of the DCGRU cell. The input $\mathbf{x}(t)$, as well as the previous hidden state $\mathbf{H}(t-1)$ are concatenated and passed to the reset gate $\mathbf{r}(t)$, as well as to the update gate $\mathbf{u}(t)$. The reset gate $\mathbf{r}(t)$ controls the proportion of $\mathbf{H}(t-1)$ which enters $\mathbf{c}(t)$, together with input $\mathbf{x}(t)$. Then the hidden state $\mathbf{H}(t-1)$ is updated by $\mathbf{c}(t)$, whereby the amount of new information is controlled by $\mathbf{u}(t)$.

$$\mathbf{r}(t) = \sigma(\mathbf{\Theta}_r *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_r)$$

$$\mathbf{u}(t) = \sigma(\mathbf{\Theta}_u *_G [\mathbf{x}(t), \mathbf{H}(t-1)] + \mathbf{b}_u)$$

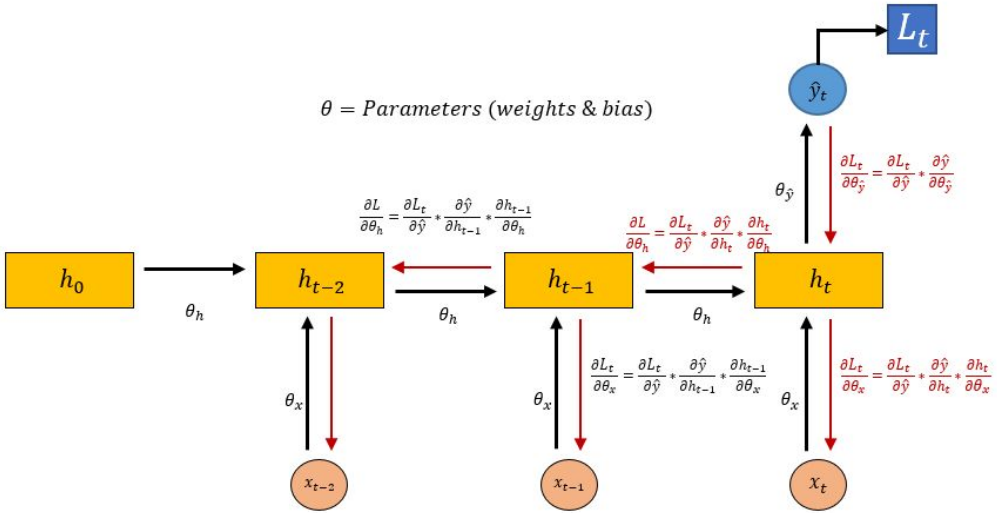
$$\mathbf{c}(t) = \tanh(\mathbf{\Theta}_c *_G [\mathbf{x}(t), (\mathbf{r}(t) \odot \mathbf{H}(t-1))] + \mathbf{b}_c)$$

$$\mathbf{H}(t) = \mathbf{u}(t) \odot \mathbf{H}(t-1) + (1 - \mathbf{u}(t)) \odot \mathbf{c}(t),$$

Temporal Dynamics and DCRNN Training

- Gated recurrent units (GRUs) are adapted for graph signals, forming DCGRUs.

- The **training process involves backpropagation through time (BPTT)**, maximizing likelihood for accurate predictions.



Vector Autoregression

$$Y_t = A_1 Y_{t-1} + A_2 Y_{t-2} + \ldots + A_p Y_{t-p} + \varepsilon_t$$

where:

- Y_t: Vector of signal values at time t for all regions.
- A_i: Influence matrix at lag i.
- p: Number of lags.
- ε_t: Error term vector.

- Y_t is a $n \times 1$ column vector.
- $A_1, A_2, \ldots, A_p$ are $n \times n$ matrices, where each element represents the coefficient of the corresponding lagged endogenous variable.
- ε_t is also a $n \times 1$ column vector, representing the error terms for each endogenous variable.

Vector Autoregression (in the paper)

Autoregressive models. As Granger causality[21] is typically based on linear vector autoregressive (VAR) models for stochastic time series data, we evaluated a VAR as one baseline method. The idea of an autoregressive process (AR) is that a time series $x(t)$ can be described by a linear function of the first T_p of its lagged values[84]

$$x(t) = \beta + \alpha_1 x(t-1) + \alpha_2 x(t-2) + \cdots + \alpha_p x\left(t - T_p\right) + u(t) \tag{21}$$

with coefficients $\alpha_1 \ldots \alpha_p$, intercept β and an error term $u(t)$. This expression can be extended to a multivariate VAR model with N time series $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]$ as[84]

$$\mathbf{x}(t) = \mathbf{b} + \mathbf{A}_1 \mathbf{x}(t-1) + \mathbf{A}_2 \mathbf{x}(t-2) + \cdots + \mathbf{A}_p \mathbf{x}\left(t - T_p\right) + \mathbf{u}(t), \tag{22}$$

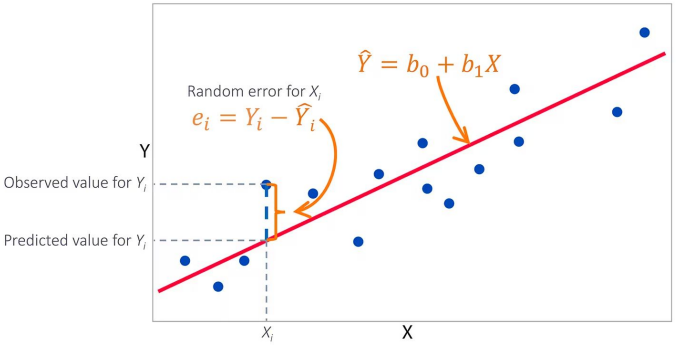
where coefficients are stored in matrices $\mathbf{A} \in \mathbb{R}^{N \times N}$, and intercepts and errors are described by vectors $\mathbf{b} \in \mathbb{R}^N$ and $\mathbf{u}(t) \in \mathbb{R}^N$. In the context of this study, time series $\mathbf{x}(t)$ reflect the BOLD signal of N brain regions, measured at different times t.

$$Y_t = A_1 Y_{t-1} + A_2 Y_{t-2} + \ldots + A_p Y_{t-p} + \varepsilon_t$$

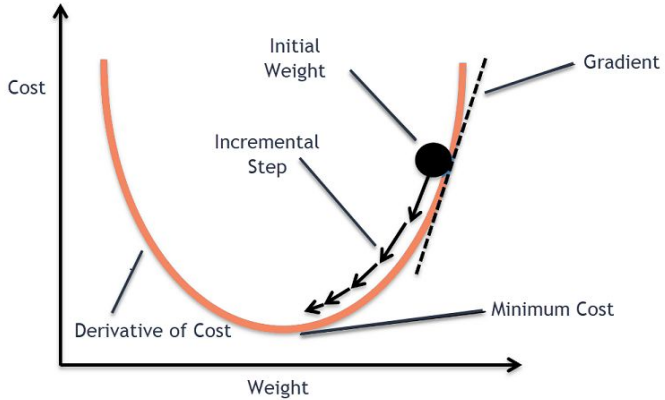
Optimization Strategies in DCRNN and VAR Models

- Two strategies for optimization: **Ordinary Least Squares (OLS) and gradient descent.**
  - VAR can use Ordinary Least Squares (OLS)
  - DCGRU uses gradient descent
- Optimization employed stochastic gradient descent with specific learning rate adjustments.

$$SSR = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

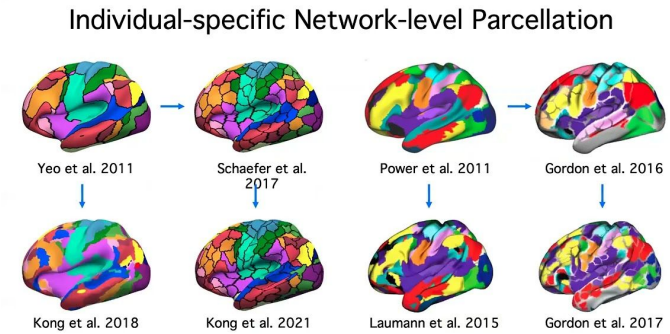


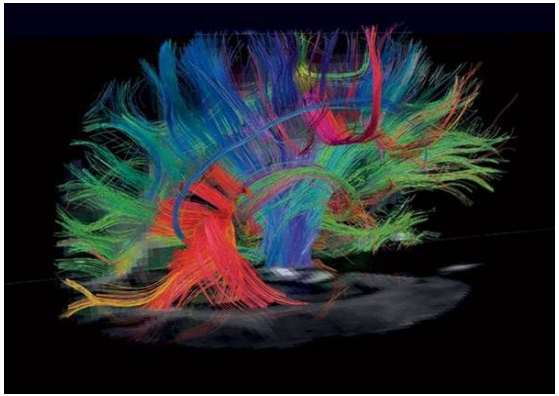
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$



Integrating Structural and Functional Data

- Structural connections from DTI data and BOLD signals from fMRI are converted into edges.
- The BOLD signal in brain regions (nodes) is average for each Region of Interest (ROI) at each time point.



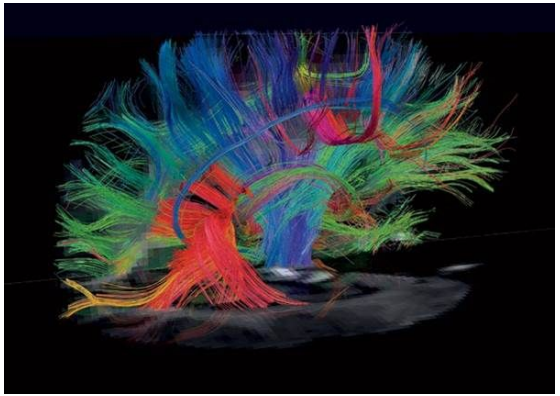


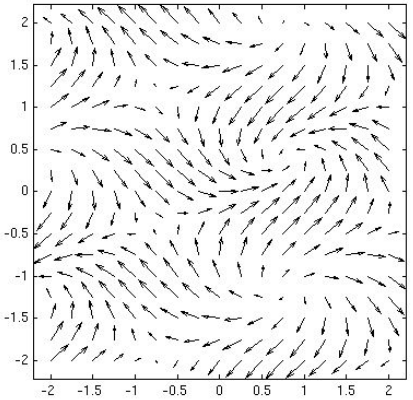
DTI is the only imaging method that shows the actual nerve tract, instead of the fat surrounding nerve linings. DTI is useful for analysis of white matter, especially with detecting diffuse axonal injury and traumatic brain injury (TBI) pathologies not visible on standard MRI scans.

https://www.umimri.com/dti/

Creation of the Adjacency Matrix from DTI Data

- Overview:
  - The adjacency matrix is derived from Diffusion Tensor Imaging (DTI) data, which captures the diffusion of water molecules in brain tissue, indicating the directionality and strength of white matter tracts.
- Processing Steps:
  - DTI data is processed using multi-shell multi-tissue constrained spherical deconvolution, implemented in MRtrix3 software, to estimate fiber orientation distributions.
  - **White matter tractography** is performed to estimate whole-brain structural connectivity, linking the **N = 360 regions** defined in the multi-modal parcellation atlas.
  - The number of streamlines connecting two brain regions is used to define the edge strengths in the adjacency matrix $\mathbf{A_w} \in \mathbf{R^{N \times N}}$.





https://www.umimri.com/dti/

Comparative Analysis of VAR and DCRNN Models

- Figure 2 showcases the prediction accuracy of the VAR model and the DCRNN.

- Demonstrates how the DCRNN maintains stable predictions over longer forecasting horizons compared to VAR.

- Highlights the Mean Absolute Error (MAE) as a key metric, showing DCRNN's lower error rate.

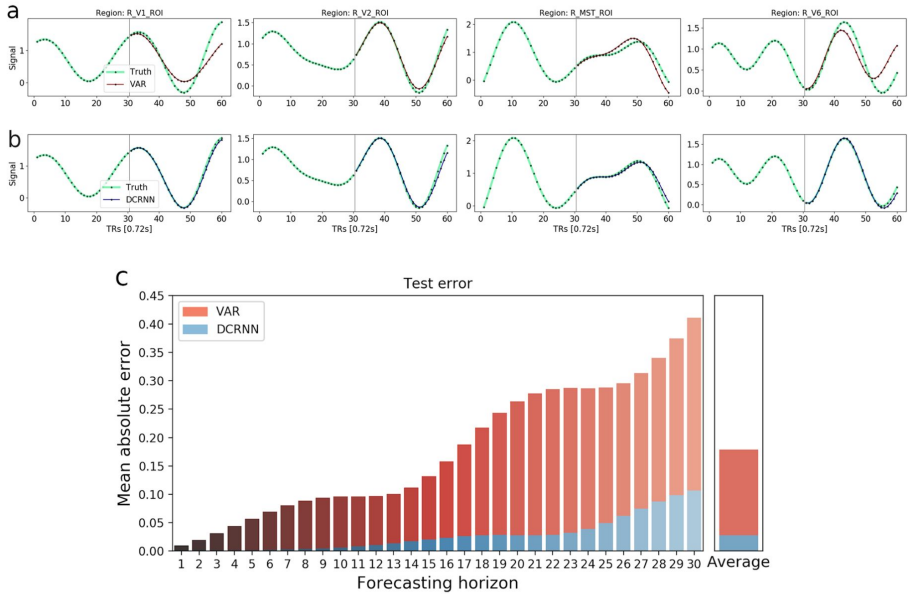


Figure 2. The figure illustrates the prediction accuracy of a VAR model (**a**) in comparison to the DCRNN (**b**). The true BOLD signal in these 4 ROIs is marked green, while predictions of the VAR are highlighted in red, and for the DCRNN in blue. The first 30 *TRs* of BOLD signal were used as the model inputs, and the goal was to predict the subsequent 30 *TRs*. This illustrative example was chosen to represent the whole test set, the prediction error of the VAR model on this sample is 0.169, and as such slightly below average, while the error of the DCRNN is with 0.037 higher than its average. Below in (**c**) the overall test MAE is illustrated in dependence of the forecasting horizon, computed as the average over all subjects, sessions, brain regions and test samples. On the right side in (**c**) the average of all horizons is shown.

Evaluating DCRNN Against VAR

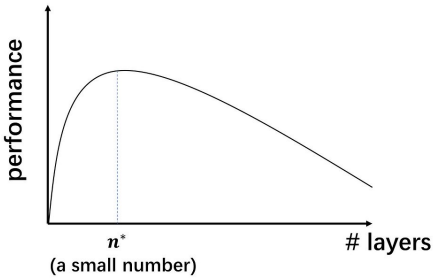
Model	Forcasting horizon (TRs)					
	5	10	15	20	25	30
VAR	0.0321	0.0589	0.0751	0.1099	0.1449	0.1786
DCRNN	0.0018	0.0028	0.0065	0.0115	0.0163	0.0279

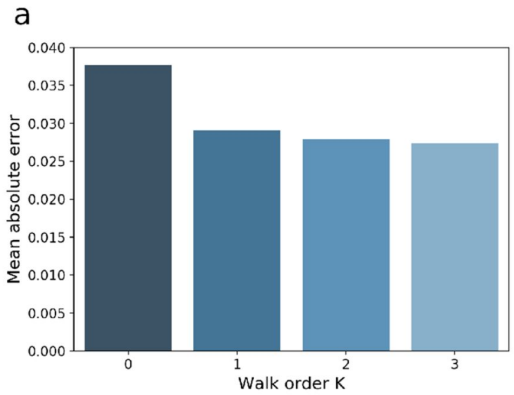
Table 1. The overall test MAE of the VAR and DCRNN model, in dependence of different forecasting horizons.

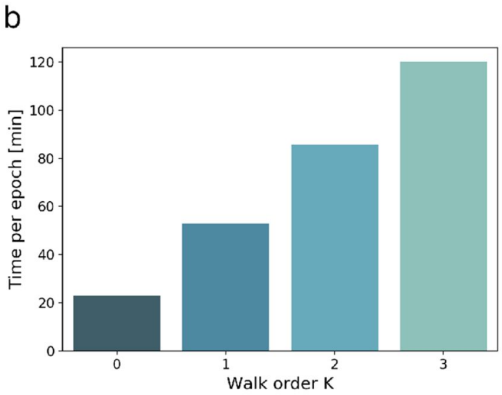
- Table 1 presents the Mean Absolute Error (MAE) comparison between the DCRNN and VAR models across different forecasting horizons.
- The data illustrates the DCRNN model's consistently lower MAE, showcasing its superior predictive accuracy.
- Emphasizes the DCRNN's effectiveness in long-term forecasting in brain network analysis.

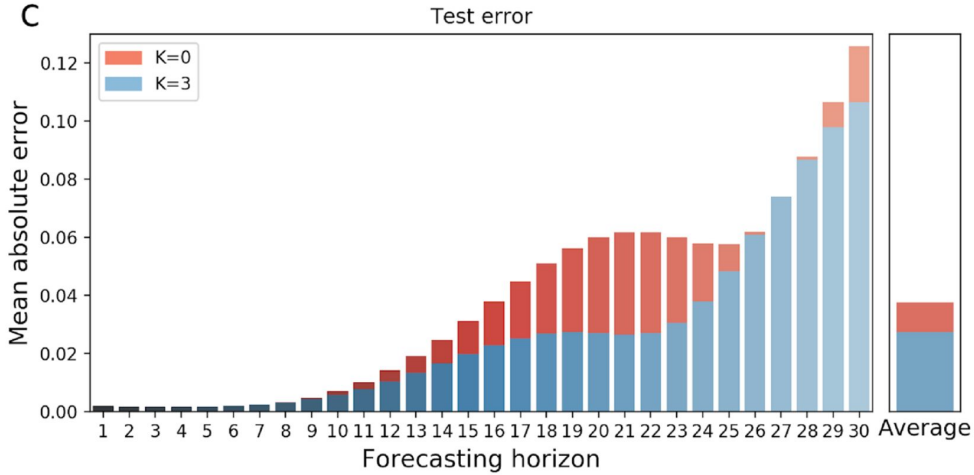
Depth of GNN (Walk order) on prediction accuracy

- **Depth of GNN (K)**:
  - Represents the number of layers or steps for information propagation.
  - Determines how far (in terms of steps away) the network integrates information from the graph structure.
- **Implications of Depth**:
  - *Depth* $K = 0$: Only considers the node itself (self-couplings).
  - *Increasing* K: Integrates information from connected nodes, enhancing model's capacity to capture complex spatial relationships.
- **Trade-offs**:
  - Improved accuracy by leveraging comprehensive spatial information.
  - Increases computational complexity and the risk of over-smoothing.









Impact of PIVC Activity on Brain Connectivity

- Figure 4 visualizes the influence of the Parieto-Insular Vestibular Cortex (PIVC) on other brain regions.
- The left and right hemispheres are shown, highlighting the PIVC in the right hemisphere in blue.
- Influence measures, normalized between 0 and 100, are encoded in red, indicating the PIVC's impact across the brain.
- The figure demonstrates a strong causal relationship primarily in the ipsilateral hemisphere, emphasizing the significant role of PIVC in brain connectivity.

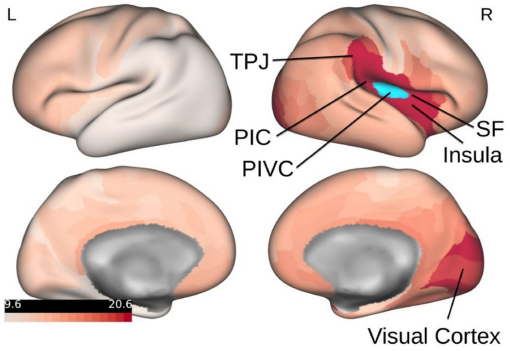


Figure 4. The figure illustrates the influence of activity in PIVC on all other brain regions. The left side depicts the left hemisphere, while on the right side the right hemisphere is shown. The target region PIVC in the right hemisphere is marked in blue. The values of the influence measure $\mathbf{I}(n')$ were normalized between 0 and 100 and are encoded in red in this illustration. *PIC* posterior insular cortex, *PIVC* parieto-insular vestibular cortex, *SF* Sylvian fissure and surrounding perisylvian cortex, *TPJ* temporo-parietal junction. Note that causal relationships from right PIVC were primarily found in the ipsilateral hemisphere. The figure was created with the Connectome Workbench software (version 1.4.2): https://www.humanconnectome.org/software/connectome-workbench.

$$I_n(n') = \frac{1}{S}\sum_{s=0}^{S}\frac{1}{T_f}\sum_{t=0}^{T_f}|\hat{\mathbf{x}}_n^{(s)}(t) - \hat{\mathbf{x}}_n'^{(s)}(t)|$$

Impact of PIVC Activity on Brain Connectivity

- $I(n')$: Influence measure of region of interest (ROI) n' on other ROIs.
- N: Total number of ROIs in the study.
- $x_n(t)$: Original prediction for ROI n at time t.
- $x_{n'}(t)$: Prediction for the perturbed ROI n' at time t, replaced with the mean value (0) to simulate perturbation.
- T_f: Number of future time steps over which predictions are made.
- S: Total number of test samples used to evaluate the model.

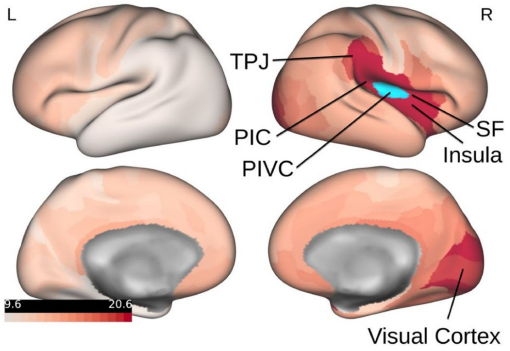


Figure 4. The figure illustrates the influence of activity in PIVC on all other brain regions. The left side depicts the left hemisphere, while on the right side the right hemisphere is shown. The target region PIVC in the right hemisphere is marked in blue. The values of the influence measure $\mathbf{I}(n')$ were normalized between 0 and 100 and are encoded in red in this illustration. *PIC* posterior insular cortex, *PIVC* parieto-insular vestibular cortex, *SF* Sylvian fissure and surrounding perisylvian cortex, *TPJ* temporo-parietal junction. Note that causal relationships from right PIVC were primarily found in the ipsilateral hemisphere. The figure was created with the Connectome Workbench software (version 1.4.2): https://www.humanconnectome.org/software/connectome-workbench.

$$I_n(n') = \frac{1}{S} \sum_{s=0}^{S} \frac{1}{T_f} \sum_{t=0}^{T_f} |\hat{\mathbf{x}}_n^{(s)}(t) - \hat{\mathbf{x}}_n'^{(s)}(t)|$$

Transfer Learning: Boosting DCRNN's Efficacy

Transfer learning significantly improves DCRNN performance, especially with limited data.

Figure 5a: Shows a decrease in training and validation MAE from epoch 70, highlighting how pretraining (blue) reduces error compared to standard training (red).

Figure 5b: Demonstrates the reduced test MAE over different forecasting horizons, affirming transfer learning's efficacy.

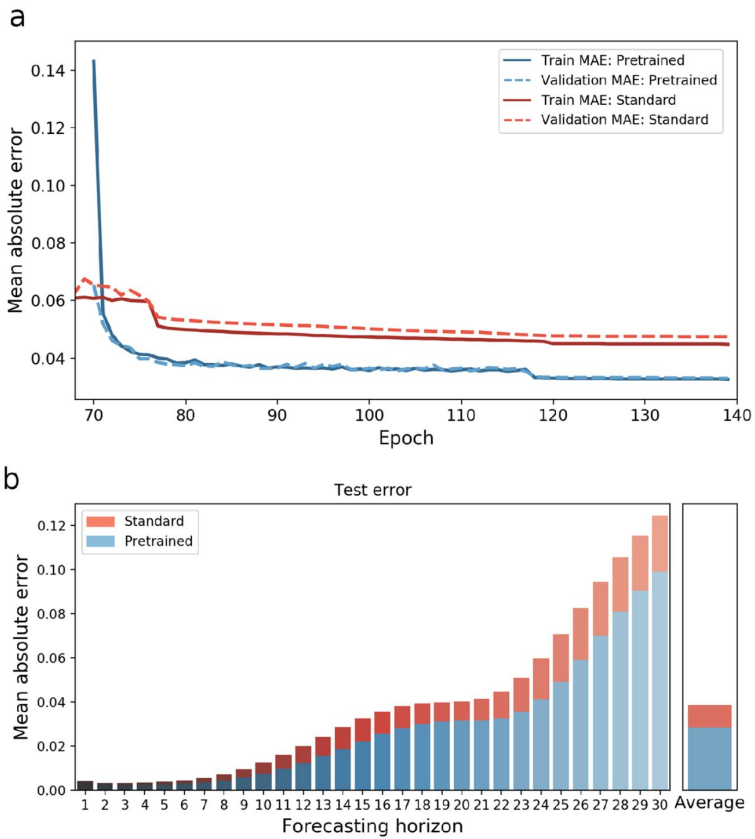


Figure 5. This figure illustrates the performance difference between standard training and encompassing transfer learning. Figure (**a**) shows the validation and training MAE during learning from epoch 70 onwards, and the errors with and without pretraining are depicted in blue and red respectively. The MAE values were computed as the average over all subjects, sessions, brain regions and test samples. At the very beginning of fine tuning, the error of the pretrained model is relatively high, but decreases after the model adapts to the UR dataset. In figure (**b**) the final test MAE of both models is shown in dependence on the forecasting horizon.

Training - Schedule sampling

1. **Scheduled Sampling**: This is a training strategy where, instead of always feeding the true previous outputs as inputs to the model for the next prediction step, the model gradually starts to use its own predictions from previous steps.
2. **Probability** $1 - \epsilon_i$: At the i-th iteration of training, the model uses the true previous output with a probability of $1 - \epsilon_i$. Early in training, the model is more likely to receive the actual previous output, making it easier for it to learn the correct sequences.

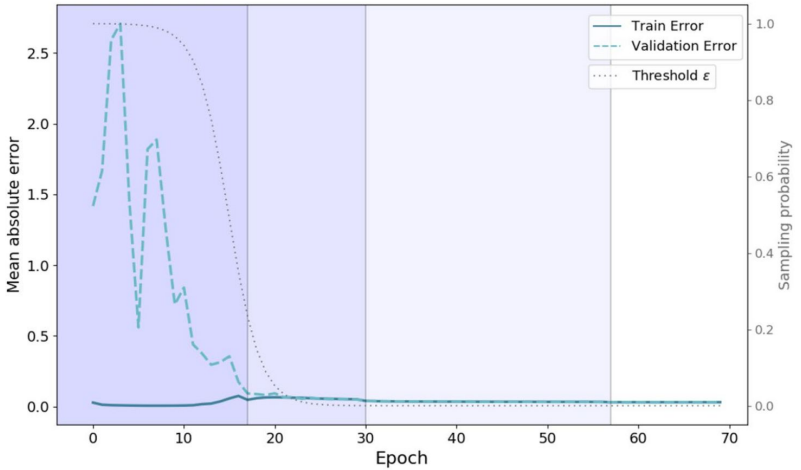


Figure 7. Illustration of the model performance during training. The figure shows the MAE during the training (solid blue line) and validation data set (dashed light blue line) in dependence of the number of epochs. The gray line illustrates the scheduled sampling probability ϵ_i over time. Vertical lines indicate when the learning rate was lowered by a factor of 0.1. In the first few epochs the training error, due to the high schedule sampling probability ϵ_i, is already quite low. During testing and validation the inputs for the decoder are always the models own predictions, what reflects the large discrepancy between training and validation error within the first epochs. When the sampling probability is subsequently decreased, the model also learns to successfully make long term forecasting.

Conclusions

- Non linear (DCGRU) methods using memory designed with temporal recurrence in mind outperform linear methods (VAR).
- Causal relationships are demonstrated between PIVC and other brain regions.
- However none of their figures show predictions with and without dti data as both the VAR and DCGRU use same adjacency matrix, so no claims about structure determining function or vice versa are supported by the figures.

GNN Depth (Walk order)

Kernel Expansion:

- The convolution kernel Θ_ω can be expanded into a power series regarding the eigenvalues (Λ) of the graph Laplacian.
- This expansion allows for an approximation of the graph convolution by truncating the series to keep terms up to a certain order K.

Approximation:

- Truncating the series to order K captures local graph structure within K steps from each node.
- The approximation makes the convolution operation computationally feasible and interpretable in terms of local neighborhood aggregation.

$$\begin{aligned}\mathbf{y}_t &= \mathbf{U}\left[\left(\sum_{k=0}^{K}\theta_k(\omega)\mathbf{\Lambda}^k\right)\mathbf{U}^T\mathbf{x}_t\right] \\ &= \sum_{k=0}^{K}\theta_k(\omega)\mathbf{T}^k\mathbf{x}_t.\end{aligned}$$

Implementing Diffusion Convolution in Neural Networks

- Diffusion Convolution Layers (DCL) integrate the graph convolution process within neural network architectures, specifically designed for graph-structured data.
- Each layer's output, $h_{q,t}$, is computed as:

$$h_{q,t} = \sigma\left(\sum_{k=0}^{K} \theta_{k,q} T^k x_t\right)$$

where:

- σ: Non-linear activation function.
- T: Transition matrix, guiding the diffusion process.
- $\theta_{k,q}$: Parameters of the q-th convolutional kernel at order k.
- x_t: Input signal at time t.

- This framework allows for the learning of graph-structured data representations, capturing both local and global structures.