LEARNING DYNAMIC GRAPH REPRESENTATION OF BRAIN CONNECTOME WITH SPATIO-TEMPORAL ATTENTION

NeurIPS2021 Paper by:
Byung-Hoon Kim, Jong Chul Ye, Jae-Jin Kim

Presented by
Sajad Abavisani

May 12, 2023

OVERVIEW

- Functional connectivity (FC): degree of temporal correlation using fMRI
- Graph Neural Networks are suitable for this application
- Contribution: STAGIN:
  - learning ***dynamic*** graph representation of the brain connectome with spatio-temporal attention

INTRODUCTION

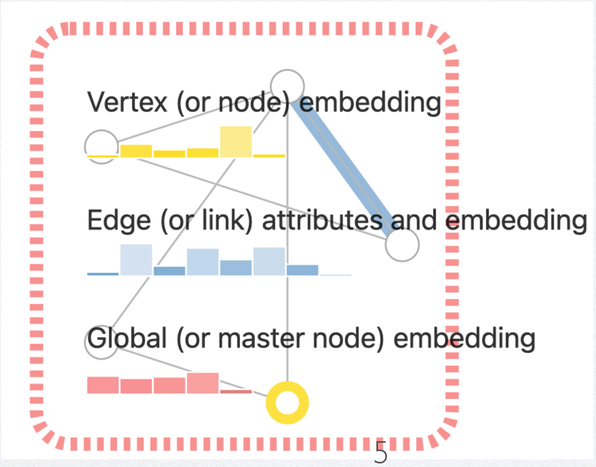
- Graph-based network analysis of brain connectome has been one of the key approaches to understanding how the brain works.
- The graph-structured nature of the brain has led to an increased interest in learning the representation of the brain FC network with the GNNs

INTRODUCTION

- Learning the representation of the brain connectome can be linked to decoding trait or state from human brain signal measurements.
- Accordingly, the current trend in studies attempting to apply GNN to the brain connectome is to input the FC graph from either resting-state or task fMRI data and predict a particular phenotype of the subjects, such as gender or presence of a specific disease.
- One of the most common limitations with previous GNN-based FC network analysis methods is that most of them fail to take advantage of the dynamic properties of the FC network, which fluctuates over time.
- Other methods that tried using FC reported less accuracy than static FC and also lack explainability

GRAPH NEURAL NETWORKS

- A Graph Neural Network (GNN) is a type of deep learning model designed to operate on graph-structured data.
- A graph represents the relations (edges) between a collection of entities (nodes).
- Information in the form of scalars or embeddings can be stored at each graph node or edge

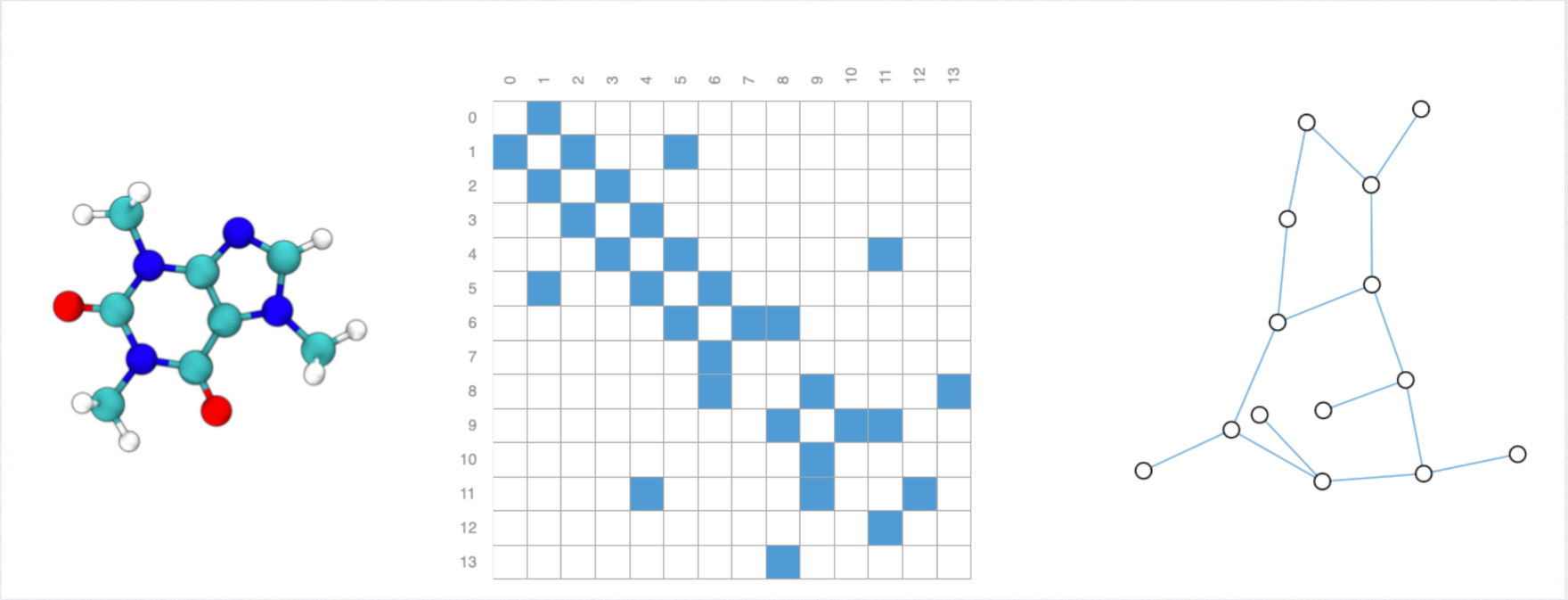


Credit for slides 5-21: https://distill.pub/2021/gnn-intro/

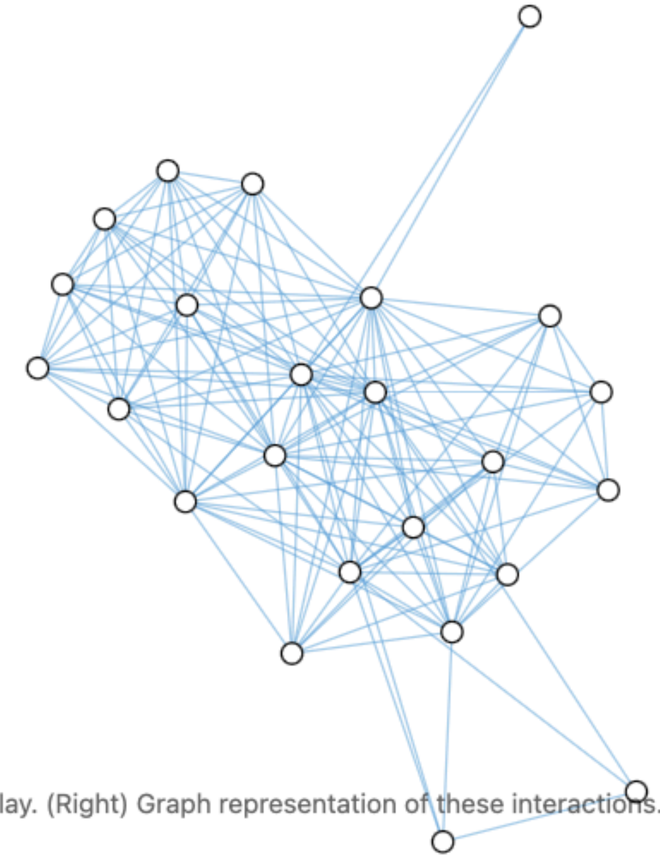
GRAPH NEURAL NETWORKS

- Graphs are a versatile way to represent familiar data, but they become essential when dealing with heterogeneously structured data where the number of neighbors per node varies, making it challenging to express the data in any other format.

MOLECULES AS GRAPHS



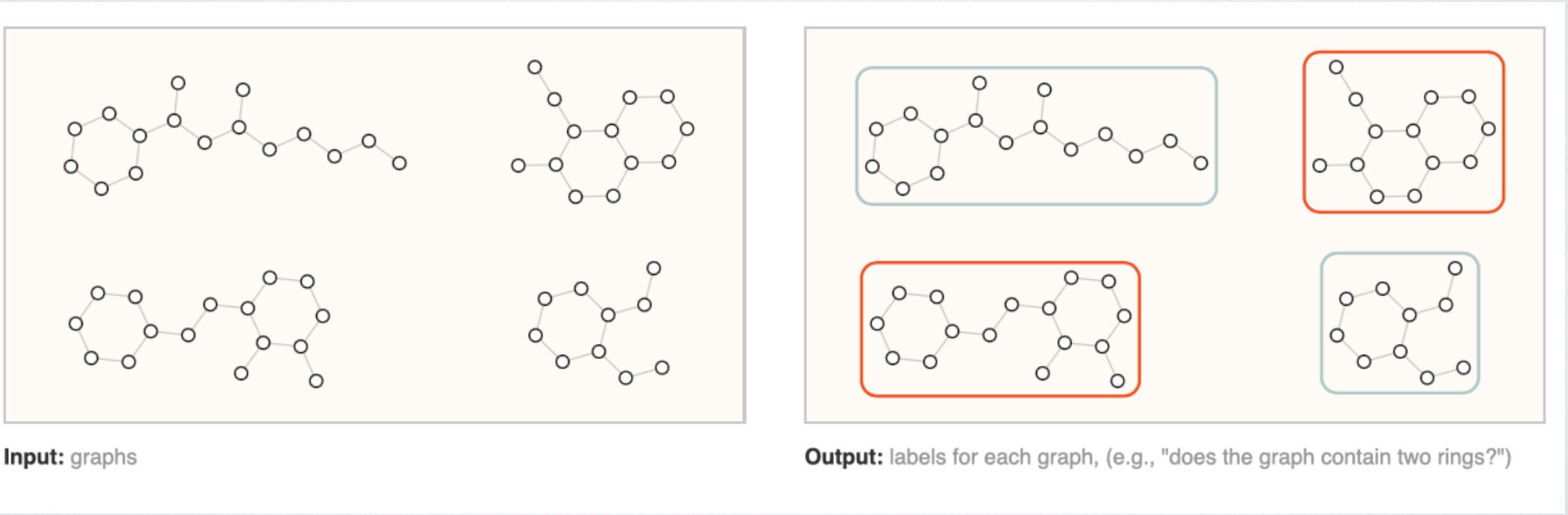
SOCIAL NETWORKS AS GRAPHS.



(Left) Image of a scene from the play "Othello". (Center) Adjacency matrix of the interaction between characters in the play. (Right) Graph representation of these interactions.

GENERAL TYPES OF PREDICTION TASKS ON GRAPHS

- Graph-level task: property of an entire graph.
- Analogy: image classification - associate a label to an entire image



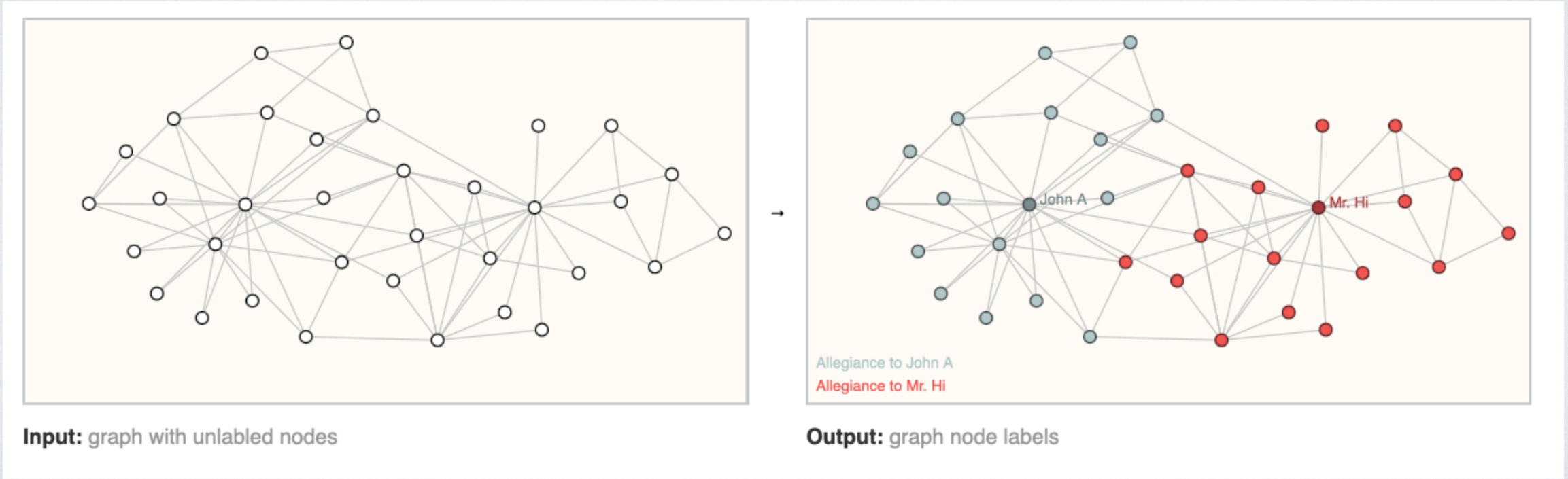
Input: graphs

Output: labels for each graph, (e.g., "does the graph contain two rings?")

GENERAL TYPES OF PREDICTION TASKS ON GRAPHS

- Node-level task: predicting the identity or role of each node within a graph.

- Analogy: image segmentation - label the role of each pixel in an image

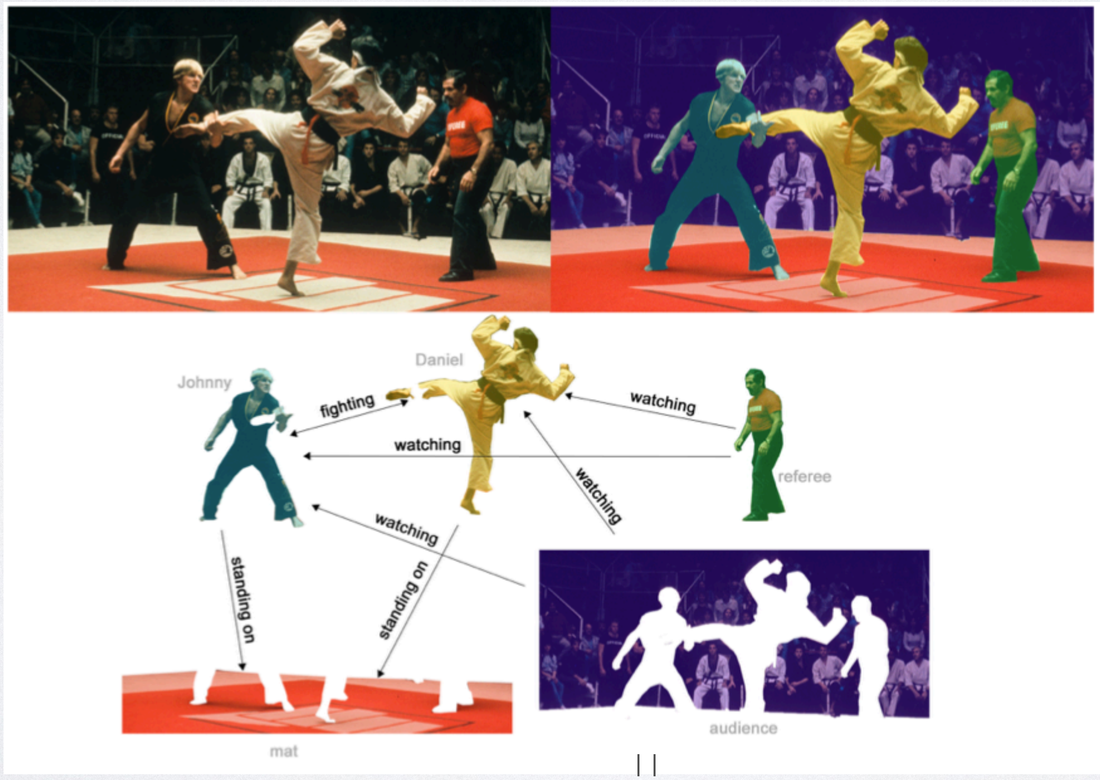


Input: graph with unlabled nodes

Output: graph node labels

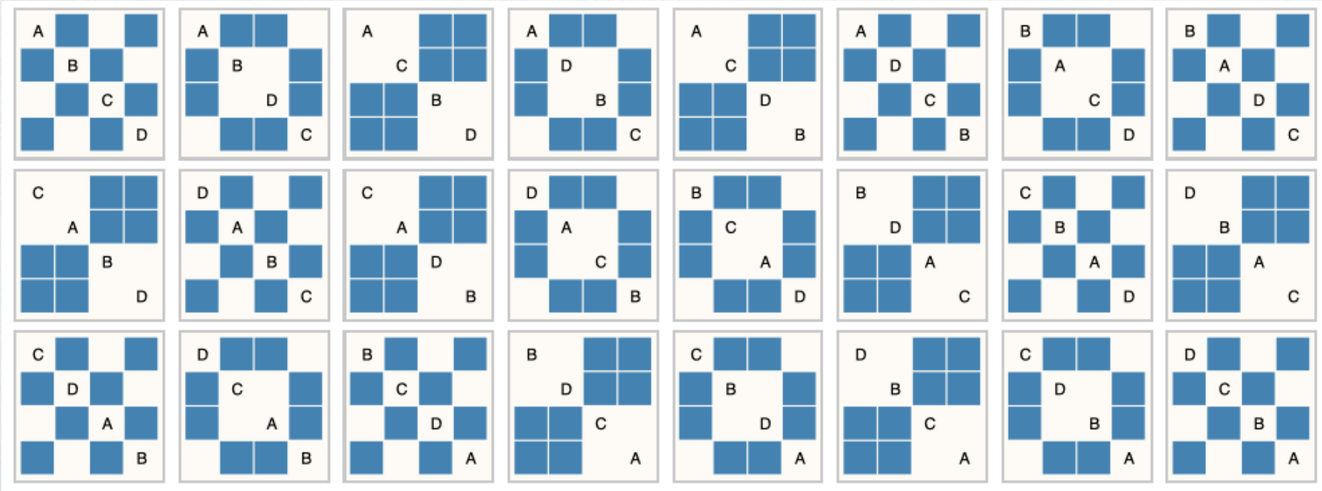
GENERAL TYPES OF PREDICTION TASKS ON GRAPHS

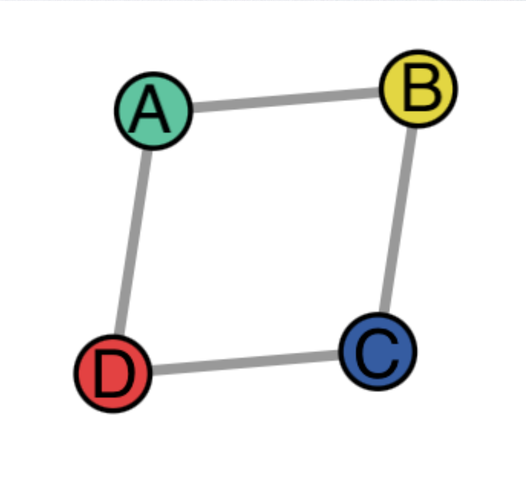
- Edge-level task: predicting the identity or role of each node within a graph.
- Analogy: image scene understanding - label the relationship between objects



THE CHALLENGES OF USING GRAPHS IN MACHINE LEARNING

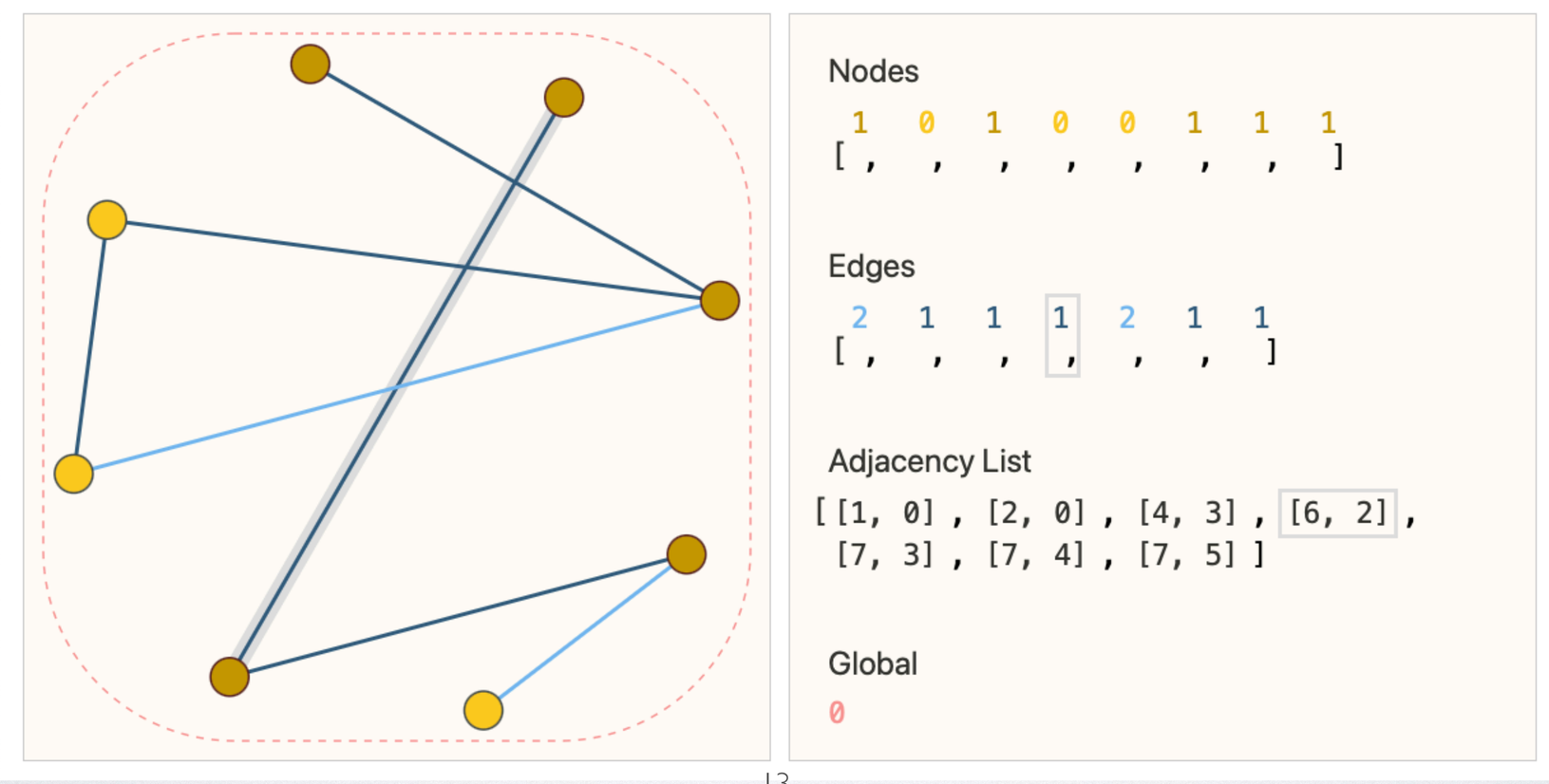
- Adjacency matrix: graphs leads to very sparse adjacency matrices
- Adjacency matrices are not permutation invariant





THE CHALLENGES OF USING GRAPHS IN MACHINE LEARNING

- A permutation invariant representation: Adjacency list instead of matrix



GRAPH NEURAL NETWORKS

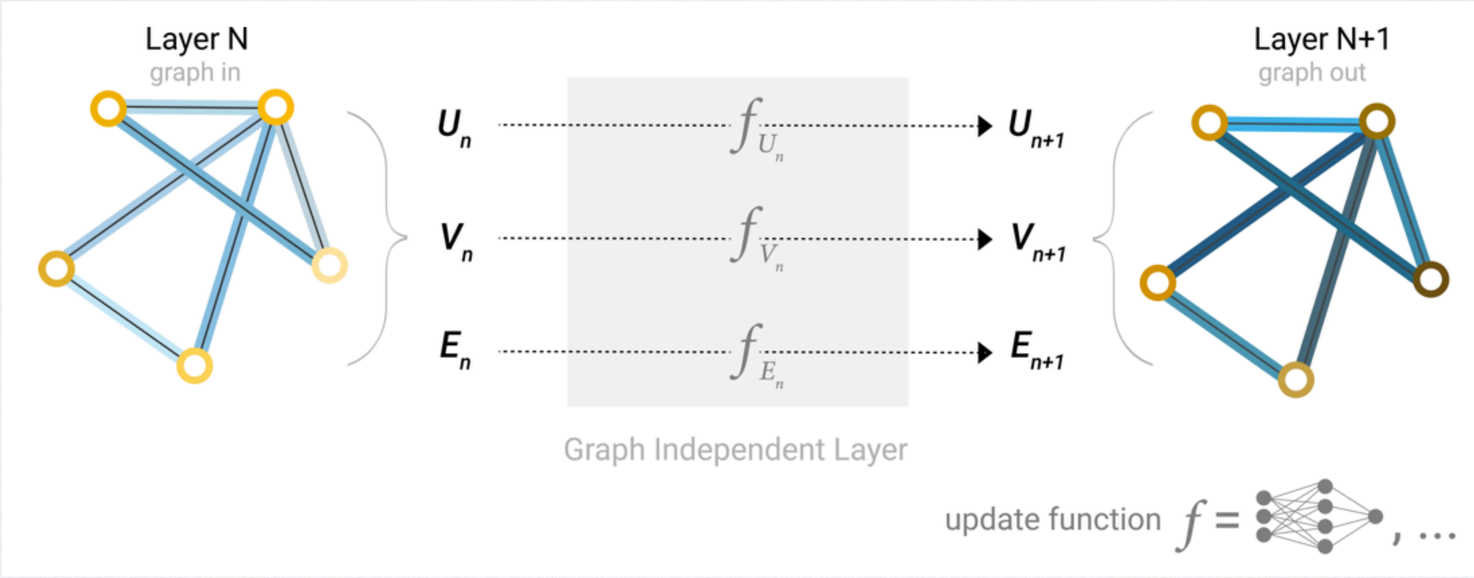
- A GNN is an optimizable transformation on all attributes of the graph (nodes, edges, global-context) that preserves graph symmetries (permutation invariances).
- GNNs using the "message passing neural network" framework proposed by Gilmer et al. [1] using the Graph Nets architecture schematics introduced by Battaglia et al. [2]

[1] Neural Message Passing for Quantum Chemistry - J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl.

[2] Relational inductive biases, deep learning, and graph networks- P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, R. Pascanu.

THE SIMPLEST GNN

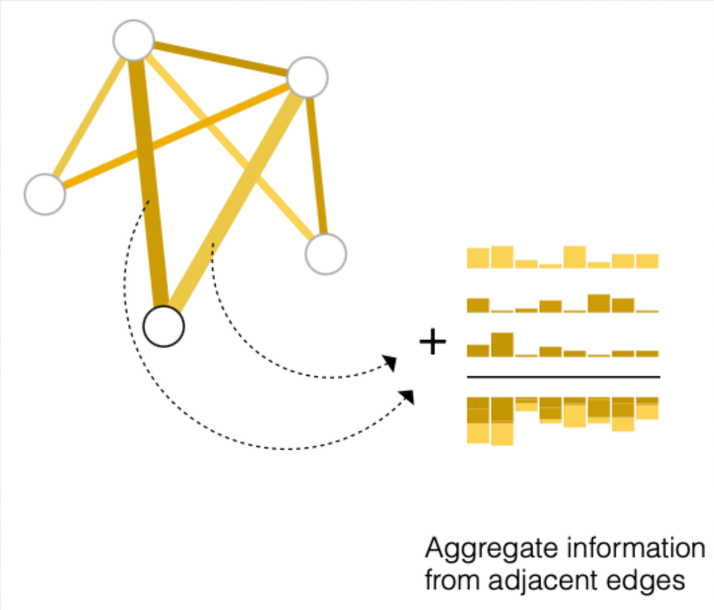
- GNN uses a separate MLP (or any differentiable model) on each component of a graph (GNN layer)
- Does not yet use the connectivity of the graph
- Stack multiple layers of GNN to get more complex representation



A single layer of a simple GNN. A graph is the input, and each component (V,E,U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n^{th} layer of a GNN mode

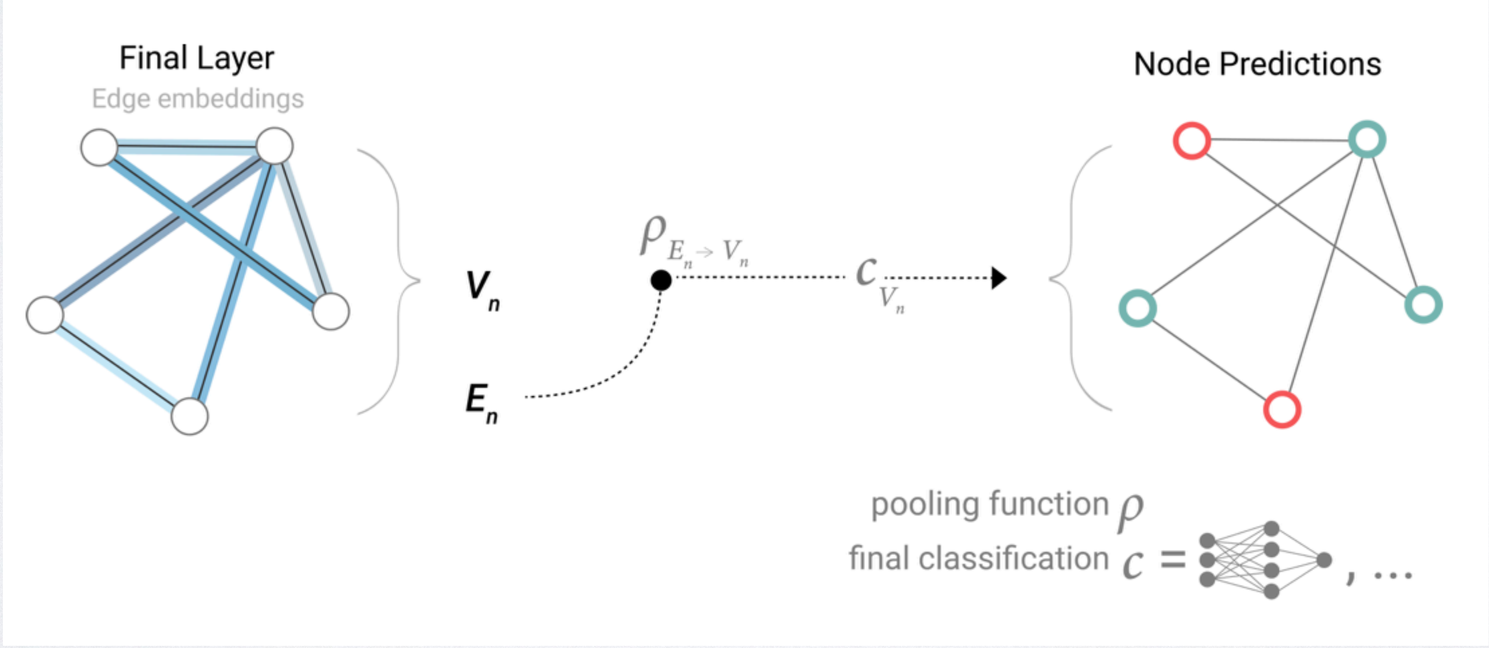
GNN POOLING

- Gather each of their embeddings and concatenate them into a matrix.
- The gathered embeddings are then **aggregated**, usually via a sum operation.
- Pooling operation ρ denote that we are gathering information from edges to nodes as $\rho_{En \to Vn}$



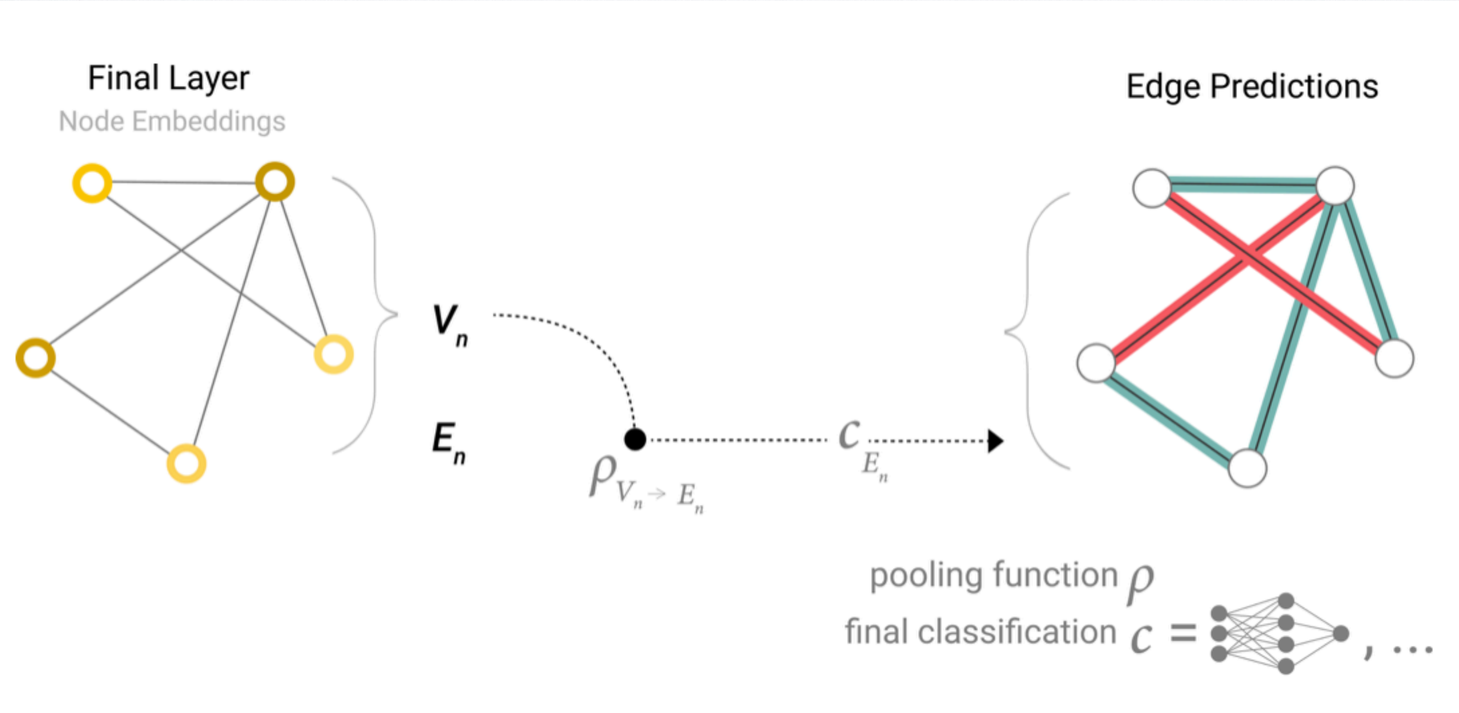
GNN POOLING

- If we only have edge-level features, and are trying to predict binary node information, we can use pooling to route (or pass) information to where it needs to go



GNN POOLING

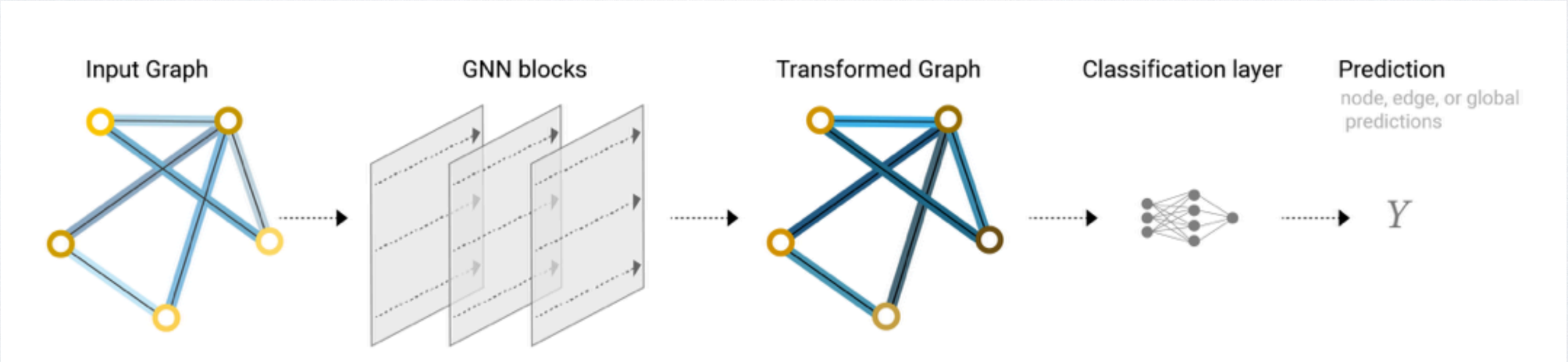
- If we only have node-level features, and are trying to predict binary edge-level information, the model looks like this.



END-TO-END PREDICTION TASK WITH A GNN MODEL.

- The pooling technique will serve as a building block for constructing more sophisticated GNN models

- In the simplest GNN, We only use connectivity when pooling information for prediction.



PASSING MESSAGES BETWEEN PARTS OF THE GRAPH

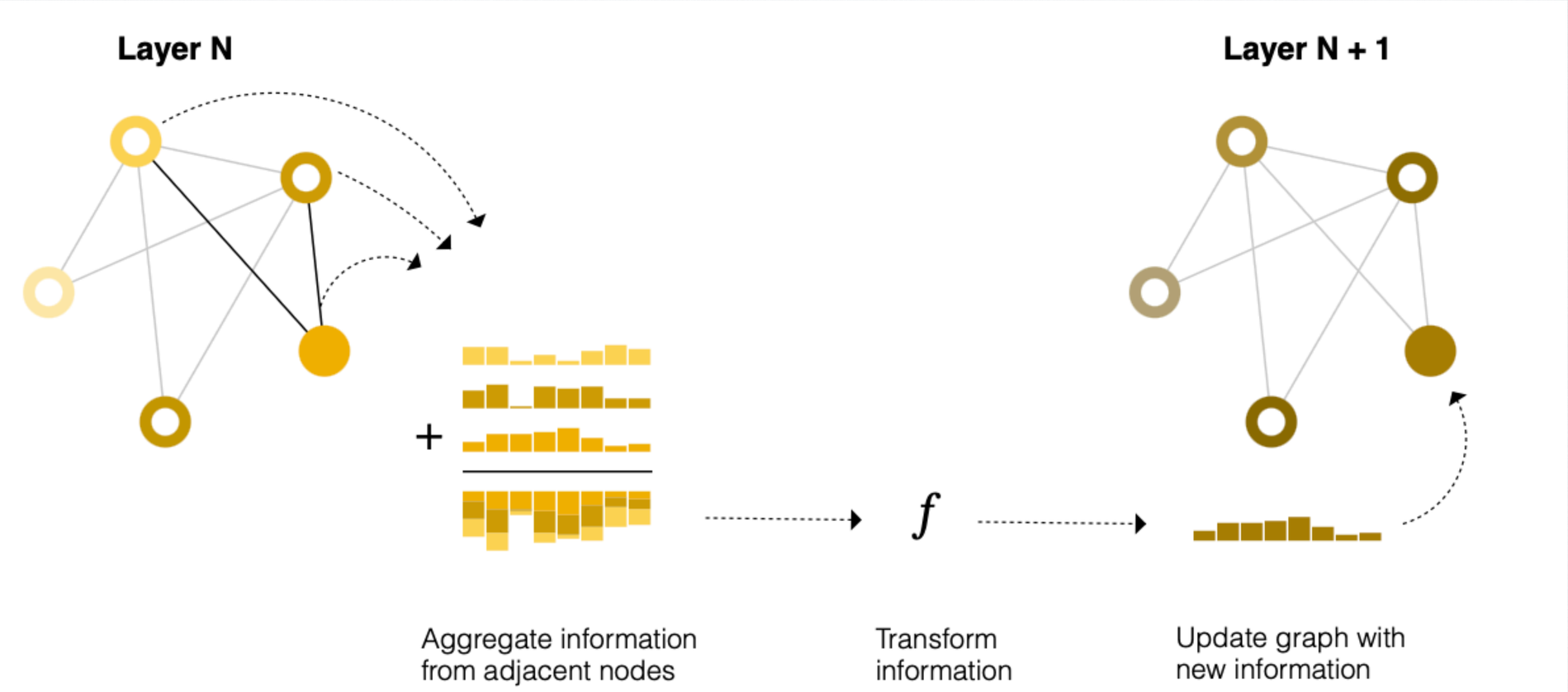
- Message passing: neighboring nodes or edges exchange information and influence each other's updated embeddings

- Message passing works in three steps:

    1. ***Gather*** all the neighboring node embeddings

    2. ***Aggregate*** all messages via an aggregate function (like sum)

    3. All pooled messages are passed through an update function(***Combine***), usually a learned neural network



REMINISCENT OF STANDARD CONVOLUTION

In essence, message passing and convolution are operations to aggregate and process the information of an element's neighbors in order to update the element's value.

GRAPH ISOMORPHISM NETWORK

- As mentioned before, The GNNs are generally composed of functions that
  - (i) integrate the node features from its neighbors, and
  - (ii) embed the integrated information with a nonlinear transformation to obtain the next layer node features.
  - The choice of these functions define many variants of the GNN

$$\boldsymbol{a}_v^{(k)} = \texttt{AGGREGATE}^{(k)}\left(\left\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right),$$
$$\boldsymbol{h}_v^{(k)} = \texttt{COMBINE}^{(k)}\left(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k)}\right),$$

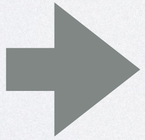
where $\boldsymbol{h}_v^{(k)}$ denotes the feature vector of node v at layer k and $\boldsymbol{h}_v^{(0)} := \boldsymbol{x}_v$.

GRAPH ISOMORPHISM NETWORK

- A variant of the GNN suitable for graph classification tasks
  - The GIN typically defines sum as the AGGREGATE and a multi-layer perceptron (MLP) with two layers as the COMBINE updating

$$\boldsymbol{h}_v^{(k)} = \texttt{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot \boldsymbol{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} \boldsymbol{h}_u^{(k-1)}\right),$$

Matrix form of the above



$$\boldsymbol{H}^{(k)} = \sigma\left((\epsilon^{(k)}\cdot \boldsymbol{I} + \boldsymbol{A})\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right),$$

$$\boldsymbol{H}^{(k)} = \left[\boldsymbol{h}_1^{(k)}, \cdots, \boldsymbol{h}_N^{(k)}\right] \in \mathbb{R}^{D\times N}$$

I = identity matrix,
A = adjacency matrix between the node features,
W = MLP weights,
σ = nonlinearity function.

READOUT FUNCTION

- The READOUT function takes the updated node features $h^{(k)}_v$ to compute the representation of the whole graph
- In general, the READOUT function is defined simply as computing the sum or average of the input node features

$$\boldsymbol{h}_G^{(k)} = \texttt{READOUT}\Big(\{\boldsymbol{h}_v^{(k)} \mid v \in G\}\Big).$$

$$\boldsymbol{h}_G^{(k)} = \boldsymbol{H}^{(k)}\boldsymbol{\phi}_{\text{mean}}.$$

$$\boldsymbol{\phi}_{\text{mean}}^{\top} = [1/N, ..., 1/N]$$

ENCODER-DECODER UNDERSTANDING OF GNNS

- Matrix formulation of the GIN operation can be thought of a CNN layer with shift operation of the convolution as the adjacency matrix A.

$$\mathrm{Vec}\left(\boldsymbol{H}^{(k)}\right) = \boldsymbol{\Sigma}^{(k)}\boldsymbol{E}^{(k)\top}\cdots\boldsymbol{\Sigma}^{(1)}\boldsymbol{E}^{(1)\top}\boldsymbol{x}, \quad \text{where} \quad \boldsymbol{x} := \mathrm{Vec}\left([\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]\right) \tag{9}$$

where $\mathrm{Vec}(\cdot)$ denotes the vectorization operation, and the k-th layer encoder matrix $\boldsymbol{E}^{(k)}$ is defined as

$$\boldsymbol{E}^{(k)} = \boldsymbol{W}^{(k)} \otimes (\epsilon^{(k)} \cdot \boldsymbol{I} + \boldsymbol{A}^T)$$

$\boldsymbol{\Sigma}^{(k)}$ is the diagonal matrix with values 1 or 0 depending on the activation pattern of the nonlinearity.

- ϕ_{mean} of equation can be thought as the decoder at the k-th layer which yields the whole graph feature vector from the encoded node feature vectors.

$$\phi_{\text{mean}}^{\top} = [1/N, ..., 1/N]$$

Proposition 1. *The READOUT function ϕ_{mean} in (8) generates a decoder with fixed constant bases.*

Proof. From the READOUT function (8), we have

$$\begin{aligned} \boldsymbol{h}_G^{(k)} =& \text{Vec}\left(\boldsymbol{h}_G^{(k)}\right) = \text{Vec}\left(\boldsymbol{H}^{(k)}\boldsymbol{\phi}_{\text{mean}}\right) = \left(\boldsymbol{\phi}_{\text{mean}}^T \otimes \boldsymbol{I}\right) \text{Vec}\left(\boldsymbol{H}^{(k)}\right) \\ =& \left(\boldsymbol{\phi}_{\text{mean}}^T \otimes \boldsymbol{I}\right) \boldsymbol{\Sigma}^{(k)} \boldsymbol{E}^{(k)\top} \cdots \boldsymbol{\Sigma}^{(1)} \boldsymbol{E}^{(1)\top} \boldsymbol{x} \end{aligned}$$

Now let $\boldsymbol{b}_i$ and $\tilde{\boldsymbol{b}}_i$ denote the i-th column of the encoder matrix $\boldsymbol{E}^{(1)}\boldsymbol{\Sigma}^{(1)} \cdots \boldsymbol{E}^{(k)}\boldsymbol{\Sigma}^{(k)}$ and the decoder matrix $\left(\boldsymbol{\phi}_{\text{mean}}^T \otimes \boldsymbol{I}\right)$, respectively. Then, it is straight to obtain the following representation:

$$\boldsymbol{h}_G^{(k)} = \sum_i \langle \boldsymbol{b}_i, \boldsymbol{x} \rangle \tilde{\boldsymbol{b}}_i$$

Therefore, we can see that although the encoder basis $\boldsymbol{b}_i$ is a function of $\boldsymbol{x}$, the decoder basis $\tilde{\boldsymbol{b}}_i$ is a constant. □

READOUT FUNCTION

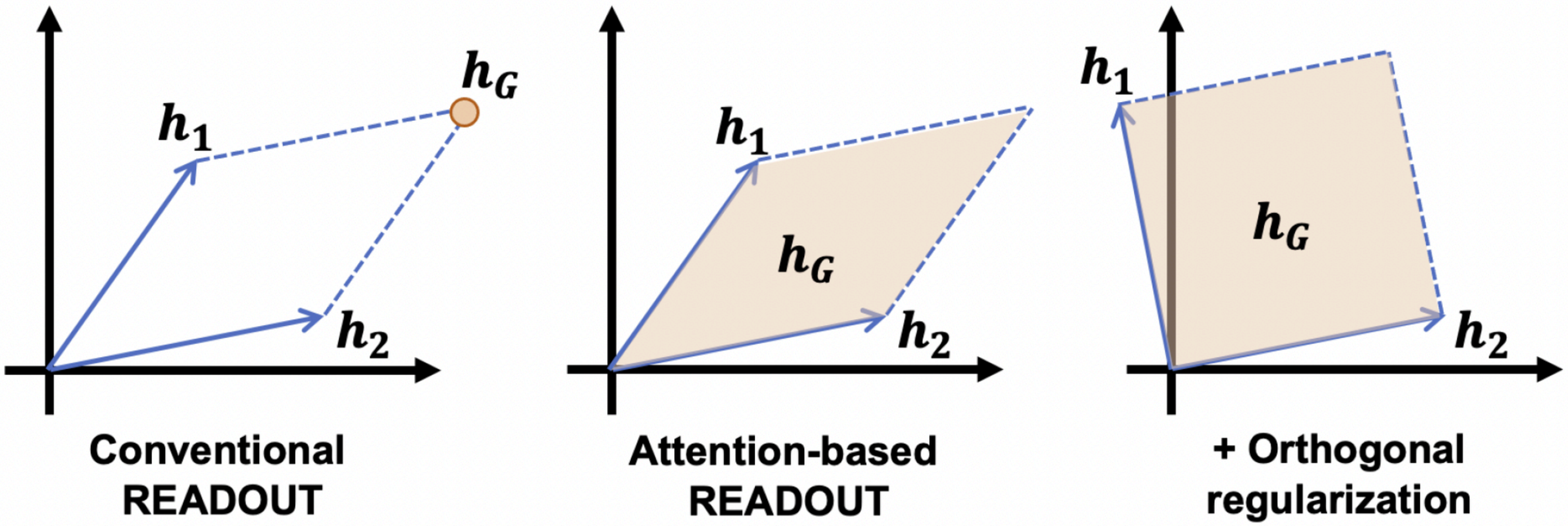


Figure 5: Geometric interpretation of the orthogonal regularization. Blue arrows indicate node feature vectors $\boldsymbol{h}_v$ of the latent space, and the orange area/point indicate possible range of graph feature vector $\boldsymbol{h}_G$ obtained by applying READOUT to $\boldsymbol{h}_v$.

Only one graph feature vector h_G is possible from the combination of two node features with conventional READOUT

DYNAMIC GRAPH DEFINITION

The sequence of input dynamic FC graphs is constructed from 4D fMRI data with 3D voxels across time. The ROI-timeseries matrix $\boldsymbol{P} \in \mathbb{R}^{N \times T_{\max}}$ is extracted by taking the mean values within a pre-defined 3D atlas which consists of N ROIs at each timepoint. Values of each ROI are standardized across time. Constructing dynamic FC matrix follows the sliding-window approach, where the temporal window of length Γ is shifted across time with stride S to generate $T = \lfloor T_{\max} - \Gamma / S \rfloor$ windowed matrices $\bar{\boldsymbol{P}}(t) \in \mathbb{R}^{N \times \Gamma}$ (Figure 2 (a)). The FC at time t is defined as the correlation coefficient matrix $\boldsymbol{R}(t)$ of the windowed timeseries between $\bar{\boldsymbol{p}}_i(t)$ and $\bar{\boldsymbol{p}}_j(t)$:

$$R_{ij}(t) = \frac{\mathrm{Cov}(\bar{\boldsymbol{p}}_i(t), \bar{\boldsymbol{p}}_j(t))}{\sigma_{\bar{\boldsymbol{p}}_i}(t)\sigma_{\bar{\boldsymbol{p}}_j}(t)} \in \mathbb{R}^{N \times N},$$

where the subscript i and j are the row and column indices of $\bar{\boldsymbol{P}}(t)$, Cov denotes the cross covariance, and $\sigma_{\boldsymbol{p}}$ denotes the standard deviation of $\boldsymbol{p}$. The final binary adjacency matrix $\boldsymbol{A}(t) \in \{0, 1\}^{N \times N}$ is obtained from the FC matrix $\boldsymbol{R}(t)$ by thresholding the top 30-percentile values of the correlation matrix as connected, and otherwise unconnected following [23].

DYNAMIC GRAPH DEFINITION

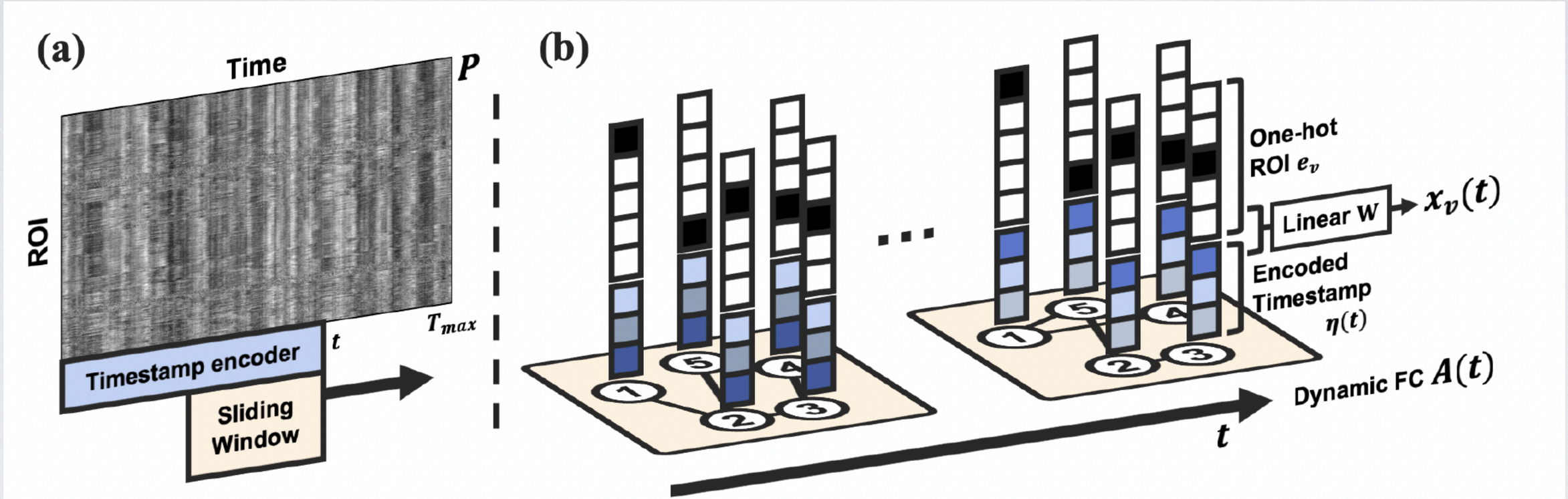
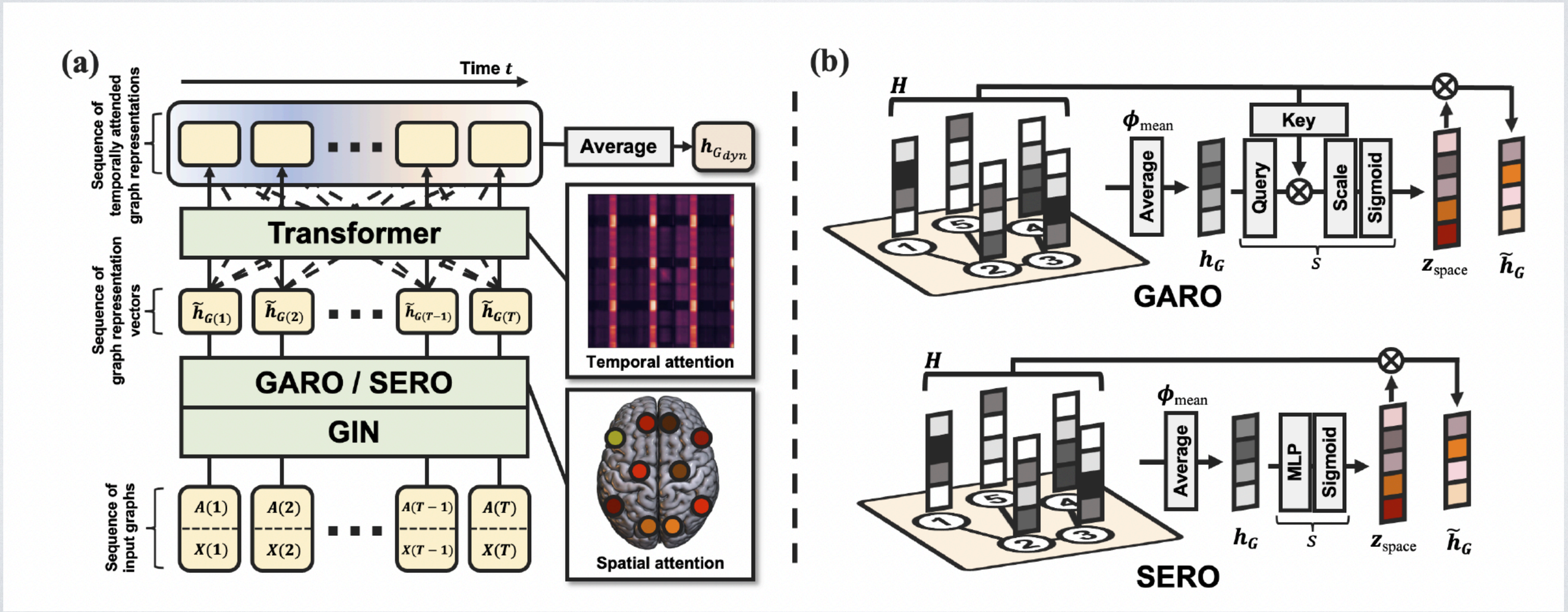


Figure 2: Defining the dynamic graph. (a) Scheme of extracting dynamic graph from ROI-timeseries matrix $\boldsymbol{P}$. (b) Example of a constructed dynamic graph.

STAGIN: SPATIO-TEMPORAL ATTENTION GRAPH ISOMORPHISM NETWORK



SPATIAL ATTENTION WITH ATTENTION-BASED READOUT

As suggested from Proposition 1, conventional READOUT function of GNN can be thought of as a fixed decoder that decodes whole-graph feature from the node features with no learnable parameters. We address this issue by incorporating attention to the READOUT function, which the attention here refers to the scaling coefficient across the nodes learned by the model. Specifically, the spatial attention vector $\boldsymbol{z}_{\text{space}}(t) \in [0,1]^N$ is computed by taking the $\boldsymbol{H}$ as a prior:

$$\boldsymbol{z}_{\text{space}} = s(\boldsymbol{H}), \tag{11}$$

$$\tilde{\boldsymbol{h}}_G = \boldsymbol{H}\boldsymbol{z}_{\text{space}}, \tag{12}$$

where $s : \mathbb{R}^{D\times N} \rightarrow [0,1]^N$ is the attention function and $\tilde{\boldsymbol{h}}_G$ denotes spatially attended graph

GARO: GRAPH-ATTENTION READOUT

The GARO follows key-query embedding based attention of the Transformer [46]. However, the key embedding is computed from the matrix of node features $\boldsymbol{H}$, while the query embedding is computed from the vector of unattended graph representation $\boldsymbol{H}\boldsymbol{\phi}_{\text{mean}}$:

$$\begin{aligned} \boldsymbol{K} &= \boldsymbol{W}_{\text{key}}\boldsymbol{H}, \\ \boldsymbol{q} &= \boldsymbol{W}_{\text{query}}\boldsymbol{H}\boldsymbol{\phi}_{\text{mean}}, \\ \boldsymbol{z}_{\text{space}} &= \text{sigmoid}\left(\frac{\boldsymbol{q}^{\top}\boldsymbol{K}}{\sqrt{D}}\right), \end{aligned} \tag{13}$$

where $\boldsymbol{W}_{\text{key}} \in \mathbb{R}^{D\times D}$, $\boldsymbol{W}_{\text{query}} \in \mathbb{R}^{D\times D}$ are learnable key-query parameter matrices, $\boldsymbol{K} \in \mathbb{R}^{D\times N}$ is the embedded key matrix, and $\boldsymbol{q} \in \mathbb{R}^{D}$ is the embedded query vector.

SERO: SQUEEZE-EXCITATION READOUT

The SERO follows MLP based attention of the Squeeze-and-Excitation Networks [19]. However, attention from the *squeezed* graph representation does not scale the channel dimension, but the node dimension in SERO:

$$\boldsymbol{z}_{\text{space}} = \text{sigmoid}\Big(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{H}\boldsymbol{\phi}_{\text{mean}})\Big), \tag{14}$$

where σ is the nonlinearity function and $\boldsymbol{W}_1 \in \mathbb{R}^{D\times D}$, $\boldsymbol{W}_2 \in \mathbb{R}^{N\times D}$ are learnable parameter matrices. This type of spatial dimension squeeze-excitation module has been shown to improve performance of the CNN models [41], but was not easily applicable to general graphs which may vary in number of nodes for each graph. We exploit the fact that the brain graphs have fixed number of nodes N across participants based on the chosen atlas.

ORTHOGONAL REGULARIZATION

If we take a closer look at (8) and (12), computation of graph feature vector $\boldsymbol{h}_G$ from the node feature matrix $\boldsymbol{H}$ can also be viewed as reconstructing signal $\boldsymbol{h}_G$ from the basis frames $\boldsymbol{H}$ with vectors $\boldsymbol{\phi}_{\text{mean}}$ and $\boldsymbol{z}_{\text{space}}$, respectively. While $\boldsymbol{z}_{\text{space}}$ provides further expressivity of the model with adaptive coefficients when compared to $\boldsymbol{\phi}_{\text{mean}}$, we find it desirable to encourage the orthogonality of $\boldsymbol{H}$ as elaborated in the Appendix Section A. The orthogonal regularization $\mathcal{L}_{\text{ortho}}$ is defined as:

$$\mathcal{L}_{\text{ortho}} = \left\| 1/m \cdot \boldsymbol{H}^\top \boldsymbol{H} - \boldsymbol{I} \right\|_2 , \tag{15}$$

where $m = \max(\boldsymbol{H}^\top \boldsymbol{H})$. The scaling term $1/m$ ensures the columns of the matrix $\boldsymbol{H}$ become orthogonal to each other with the same length, while not restricting the specific length that the column vectors should follow.

READOUT FUNCTION

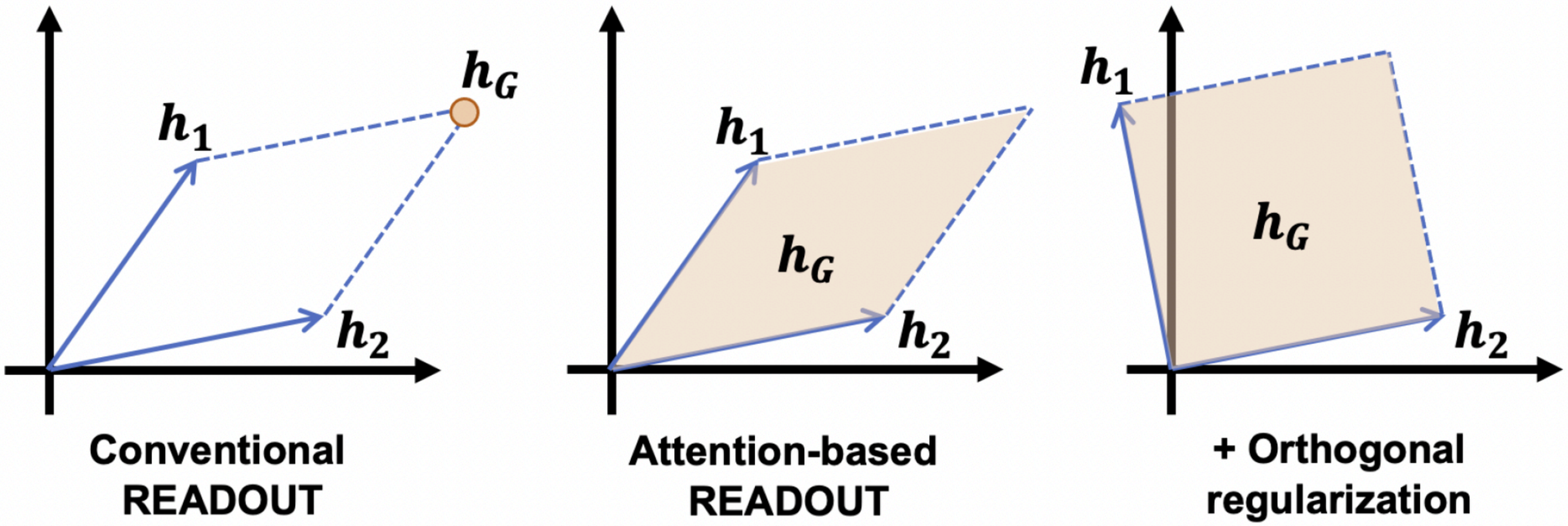


Figure 5: Geometric interpretation of the orthogonal regularization. Blue arrows indicate node feature vectors $\boldsymbol{h}_v$ of the latent space, and the orange area/point indicate possible range of graph feature vector $\boldsymbol{h}_G$ obtained by applying READOUT to $\boldsymbol{h}_v$.

Only one graph feature vector h_G is possible from the combination of two node features with conventional READOUT

RESULTS

Table 1: Comparative study on HCP-Rest and HCP-Task dataset.

Model	HCP-Rest		HCP-Task	Type of FC	# Params
	Accuracy (%)	AUROC	Accuracy (%)		
STAGIN-SERO	**88.20** ± 1.33	**0.9296** ± 0.0187	**99.19** ± 0.20	Dynamic	1,209k
STAGIN-GARO	87.01 ± 3.00	0.9151 ± 0.0258	**99.02** ± 0.17	Dynamic	1,068k
ST-GCN [15]	76.95 ± 3.00	0.8545 ± 0.0316	98.92 ± 0.27	Dynamic	355k
MS-G3D [10]	79.16 ± 2.53	0.8912 ± 0.0329	-	Dynamic	3,045k
BAnD++ [36]	-	-	97.20 ± 0.57	None	2,010k
BAnD [36]	-	-	95.10 ± 0.62	None	2,010k
r-BAnD	-	-	98.90 ± 0.27	Dynamic	664k
GIN [23]	81.34 ± 2.40	0.8955 ± 0.0237	93.87 ± 0.66	Static	169k
GCN [24]	80.79 ± 2.00	0.8741 ± 0.0174	45.07 ± 1.63	Static	101k
GraphSAGE [31]	75.48 ± 1.97	0.8237 ± 0.0228	54.52 ± 0.97	Static	202k
ChebGCN [2]	77.76 ± 2.09	0.8582 ± 0.0233	73.06 ± 0.68	Static	704k

DATASET

Table 2: Description of the experiment datasets.

Dataset	Task type	Subtasks	No. images	$T_{\max}$	C
HCP-Rest	Resting-state	Rest	1093	1200	2
HCP-Task	Working Memory	Task, Rest	1087	405	7
	Social	Mental, Random, Rest	1053	274	
	Relational	Task, Rest	1043	232	
	Motor	(L,R).(Hand,Foot), Tongue, Rest	1085	284	
	Language	Story, Math, Response	1051	316	
	Gambling	Task, Rest	1082	253	
	Emotion	Shape, Face, Rest	1049	176	

- STAGIN learns to temporally attend to the subtasks regardless of the task type, without any subtask timing information provided during training