Temporal graph neural networks and their applications

Aurora Rossi

Université Côte d'Azur, France

Algorithmic Aspects of Temporal Graphs VIII
Satellite workshop of ICALP 2025

7 July 2025

About me

- **Third-year PhD student** under the supervision of David Coudert in COATI team, Centre Inria d'Université Côte d'Azur

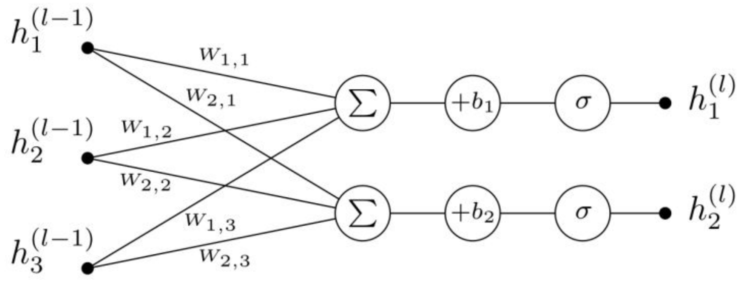
- Starting in October, I will be a **postdoctoral researcher** under the supervision of Petra Mutzel at the University of Bonn

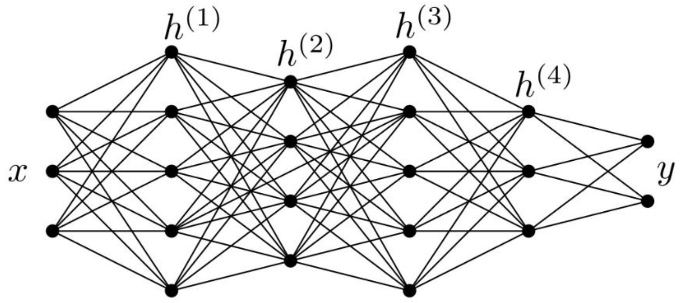
- I contribute to **`GraphNeuralNetworks.jl`**, a Julia package for GNNs that also supports temporal graph learning.
  "GraphNeuralNetworks.jl: Deep Learning on Graphs with Julia", Carlo Lucibello and Aurora Rossi, JMLR 2025

Contents

Feedforward Neural Network

A feedforward neural network is a parametric function composed of multiple layers, where each layer is defined by an affine transformation followed by a non-linear activation function.



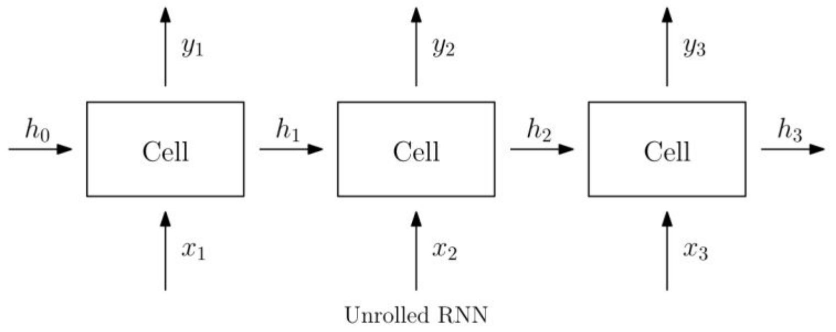


$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)})$$

$h^{(l)}$	hidden representation at layer l
$W^{(l)}$	weight matrix
$b^{(l)}$	bias vector
σ	activation function

Recurrent Neural Network

RNNs have recurrent connections that allow them to maintain a memory of past inputs, making them suitable for tasks such as natural language processing, speech recognition, and time series prediction.



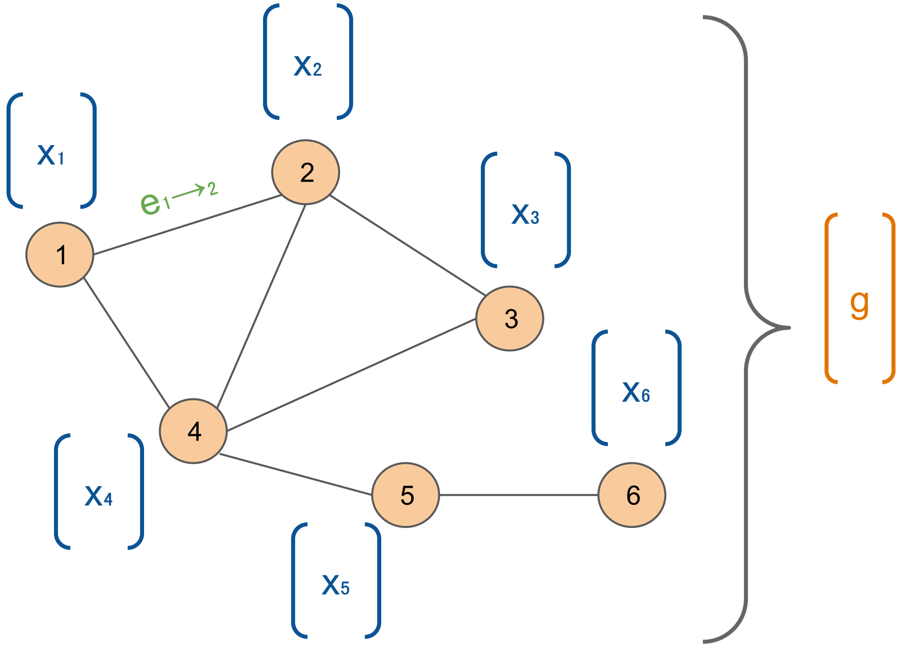
$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

h_t	hidden state at time t
x_t	input at time t
W_x	weight matrix
W_h	weight matrix
b	bias vector

Most popular ones are GRU and LSTM.

Graph

A graph **G(V,E)** is a data structure where **V** is the set of *nodes* and **E** is a set of paired nodes, whose elements are called *edges.*

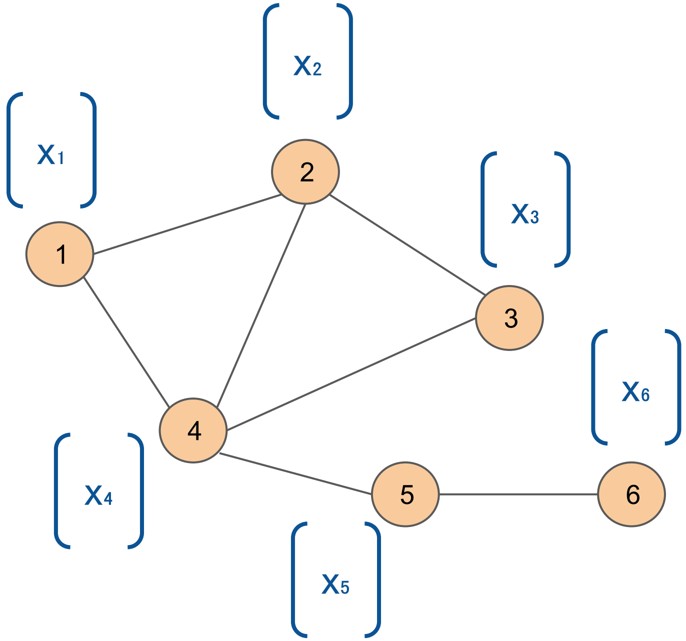


Features can be associated to nodes, edges, and graphs.

In the example on the left, x_i are nodes features of node i and g is the graph feature.

Contents

Message passing



- Every node in the graph computes a **message** for each of its neighbors

$$m_{j \to i} = \phi(x_i, x_j)$$

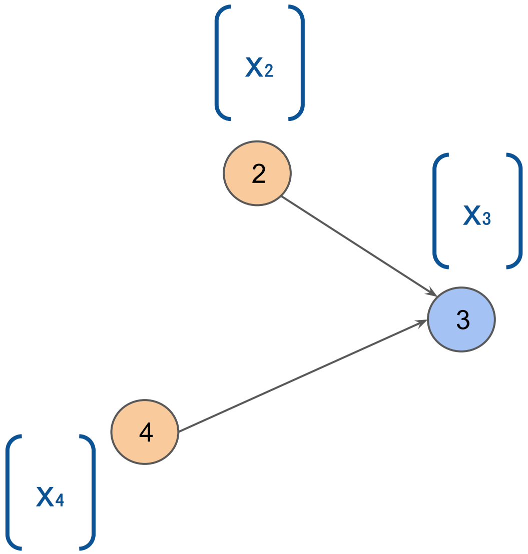
- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j \in N(i)} m_{j \to i}$$

- Node **updates** its attributes as a function of its current feature and the aggregated messages

$$x_i' = \gamma_x(x_i, \bar{m}_i)$$

Message passing



- Every node in the graph computes a **message** for each of its neighbors

$$m_{j\to i} = \phi(x_i, x_j)$$

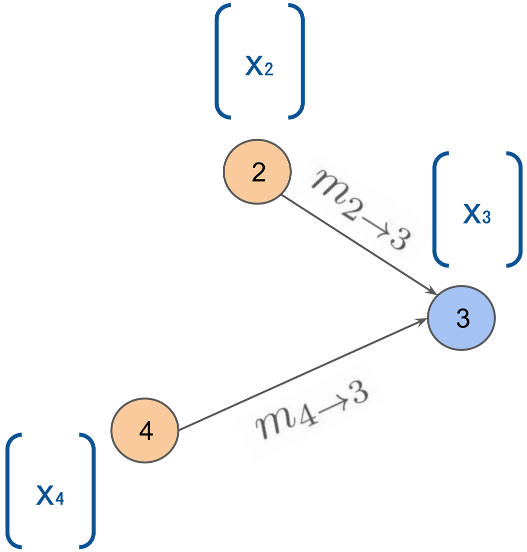
- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j\in N(i)} m_{j\to i}$$

- Node **updates** its attributes as a function of its current feature and the aggregated messages

$$x_i' = \gamma_x(x_i, \bar{m}_i)$$

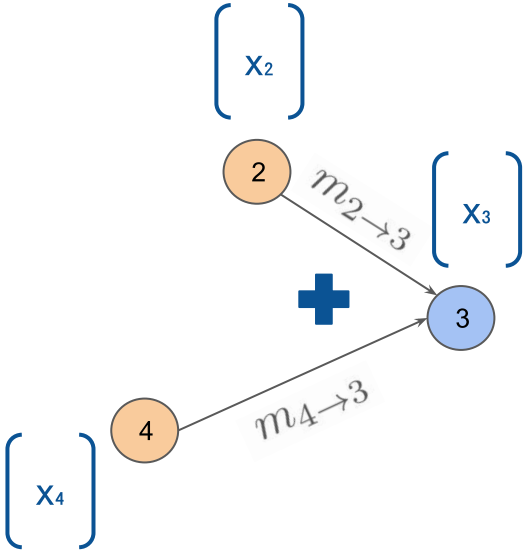
Message passing



- Every node in the graph computes a **message** for each of its neighbors

$$m_{j\to i} = \phi(x_i, x_j)$$

Message passing



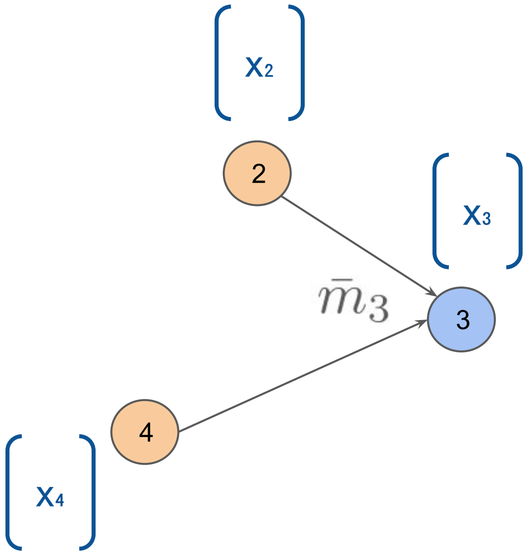
- Every node in the graph computes a **message** for each of its neighbors

$$m_{j\to i} = \phi(x_i, x_j)$$

- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j\in N(i)} m_{j\to i}$$

Message passing



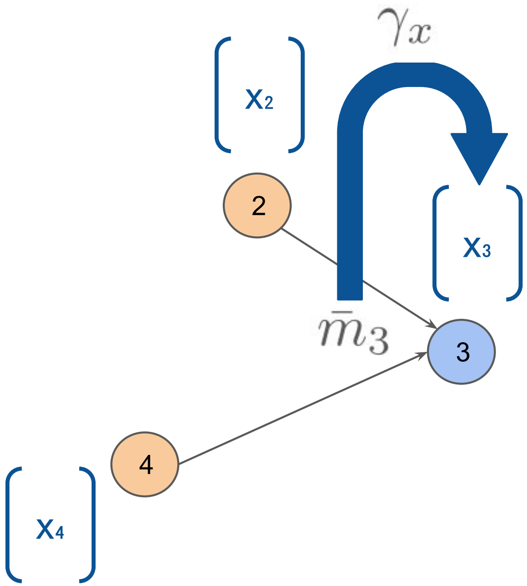
- Every node in the graph computes a **message** for each of its neighbors

$$m_{j\to i} = \phi(x_i, x_j)$$

- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j\in N(i)} m_{j\to i}$$

Message passing



- Every node in the graph computes a **message** for each of its neighbors

$$m_{j \to i} = \phi(x_i, x_j)$$

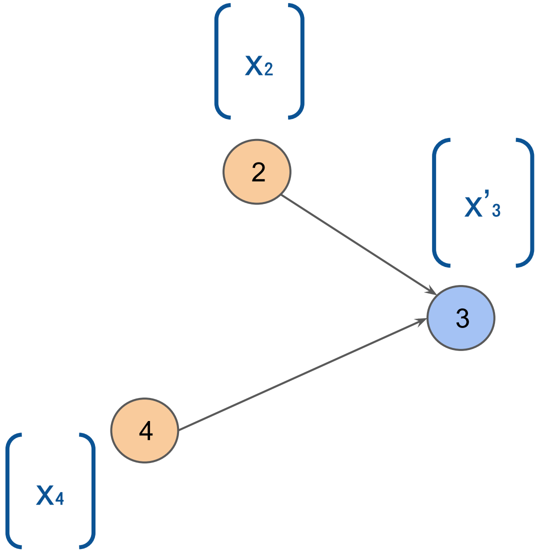
- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j \in N(i)} m_{j \to i}$$

- Node **updates** its attributes as a function of its current feature and the aggregated messages

$$x_i' = \gamma_x(x_i, \bar{m}_i)$$

Message passing



- Every node in the graph computes a **message** for each of its neighbors

$$m_{j \to i} = \phi(x_i, x_j)$$

- Every node **aggregates** the messages it receives, using a permutation-invariant function

$$\bar{m}_i = \square_{j \in N(i)} m_{j \to i}$$

- Node **updates** its attributes as a function of its current feature and the aggregated messages

$$x_i' = \gamma_x(x_i, \bar{m}_i)$$

The message passing procedure is done in parallel for every node.

Contents

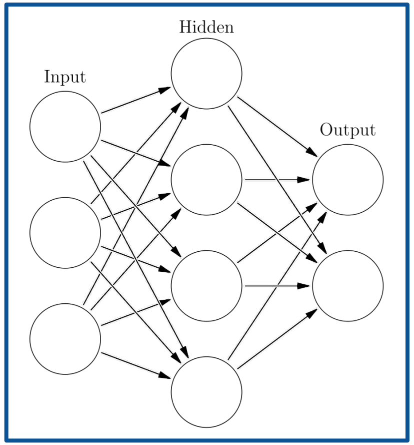
Graph neural network

A graph neural network applies **message passing** using **learnable functions**. Each message function ϕ and update function γ_x is implemented as a **neural network**.

$$m_{j \to i} = \phi(x_i, x_j)$$

$$\bar{m}_i = \square_{j \in N(i)} m_{j \to i}$$

$$x'_i = \gamma_x(x_i, \bar{m}_i)$$

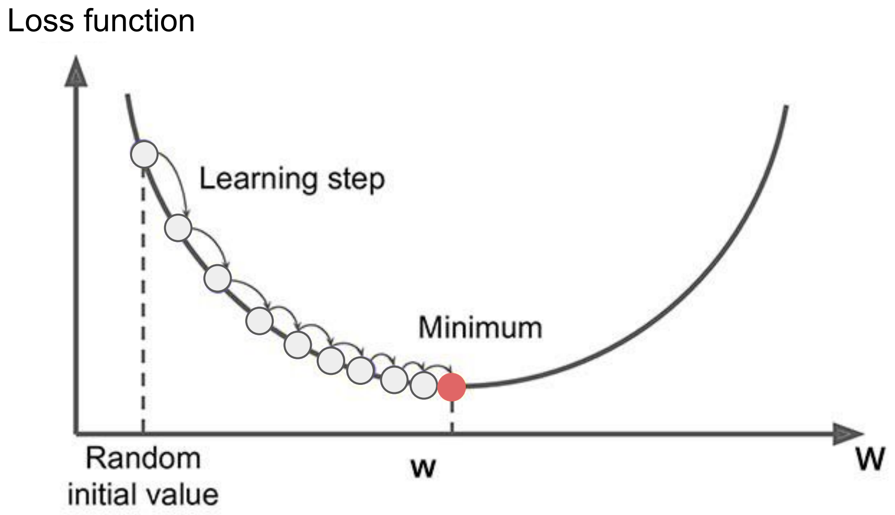


Supervised training

Training is an **optimization** problem:

the model output minimize an objective function called **loss function.**

The model weights W are then updated using
gradient descent to minimize the loss function.

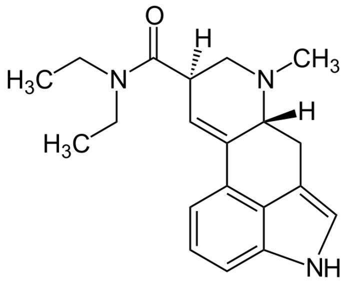


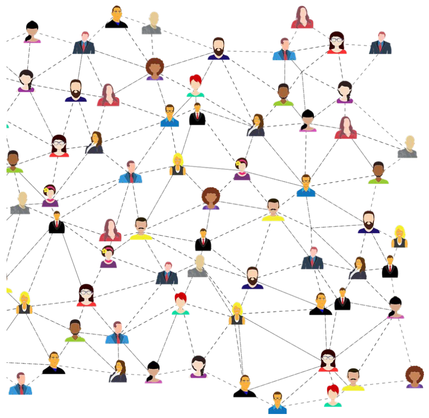
$$W_{i+1} = W_i - \alpha \nabla_W L(\mathrm{model(W)},\ \mathrm{trainset})$$

What tasks can GNNs perform?

After performing some layers, depending on how you treat the resulting features GNN can perform:

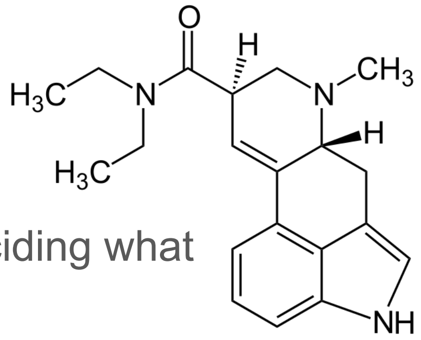
- **Graph classification**
- **Link prediction**
- **Node classification**

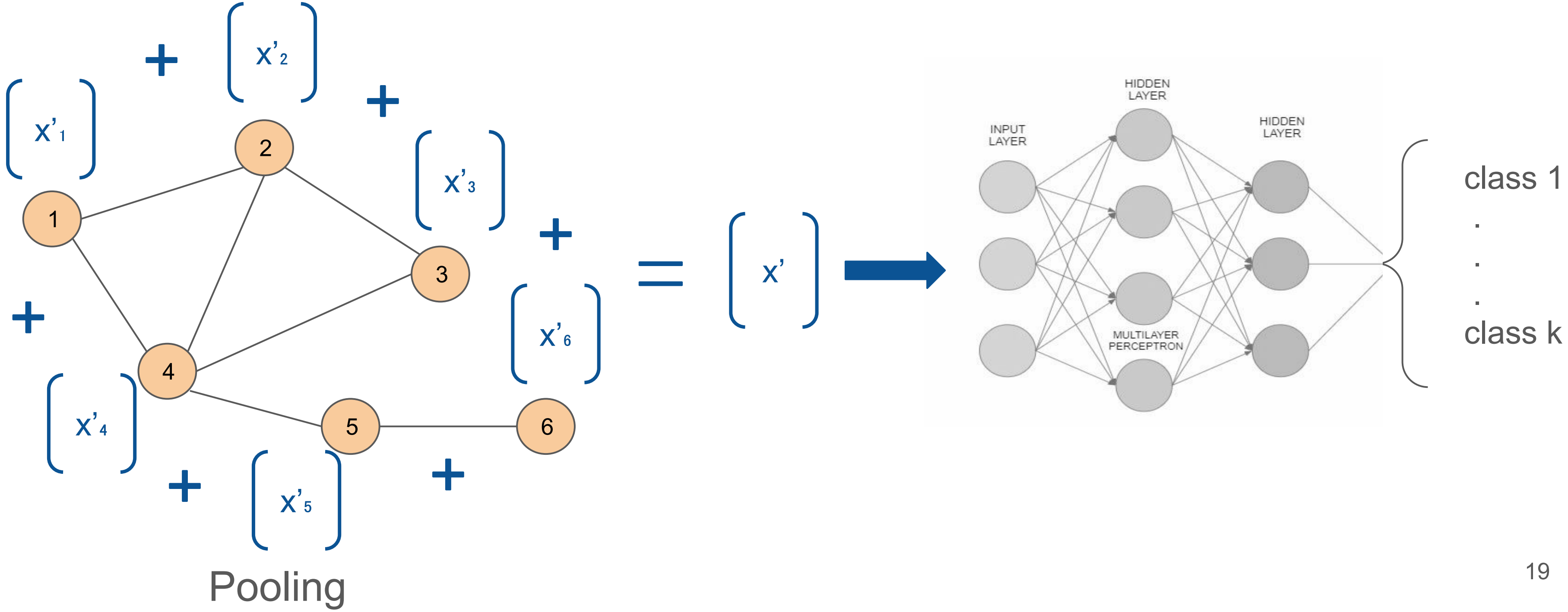




Graph classification

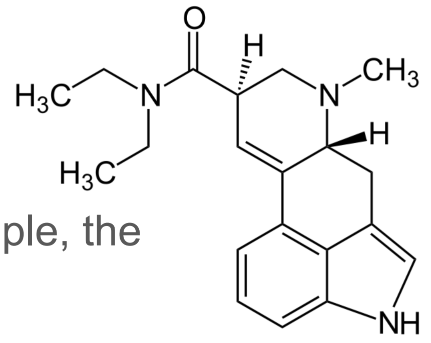
The problem of determining the category of the graph is, for example, deciding what kind of molecule

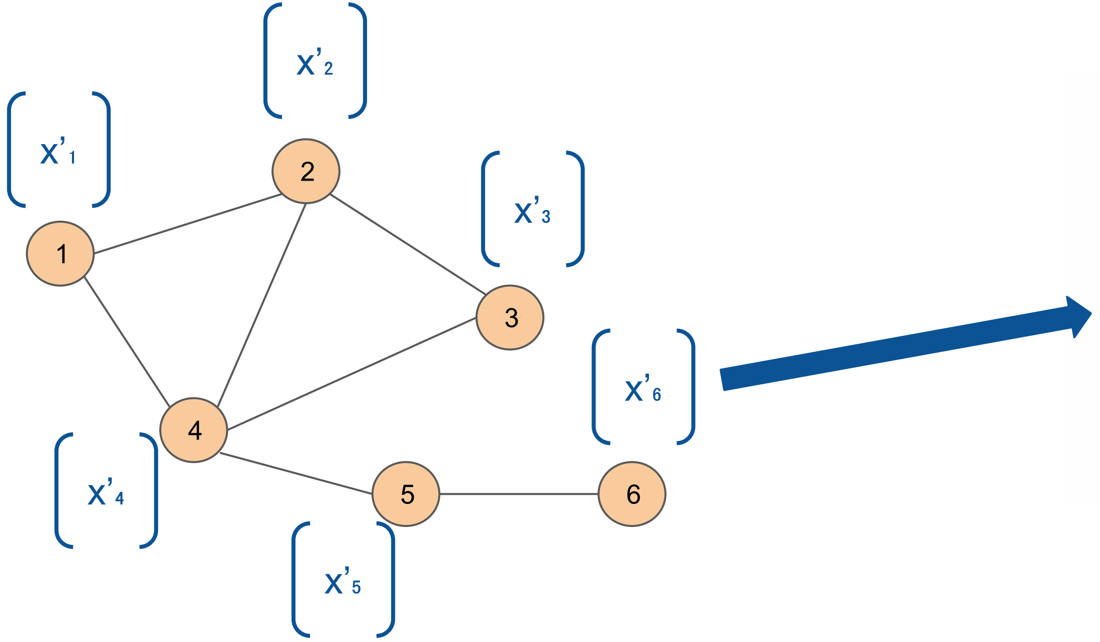


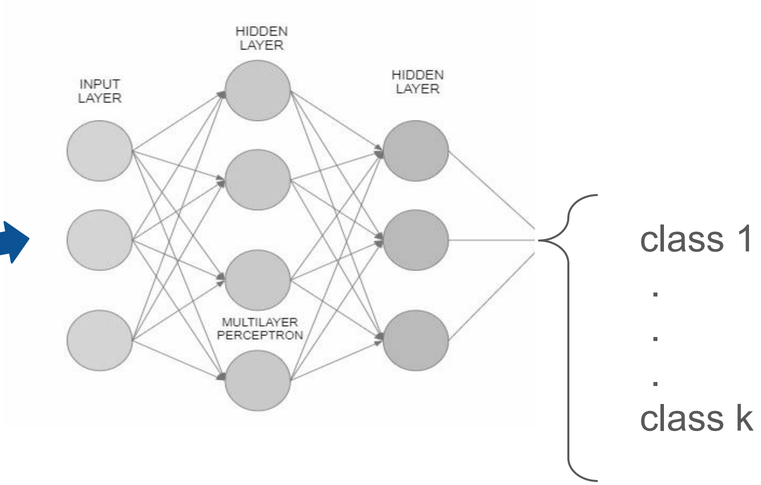


Node classification

The problem of determining the category of the graph nodes is, for example, the decision of what kind of atom

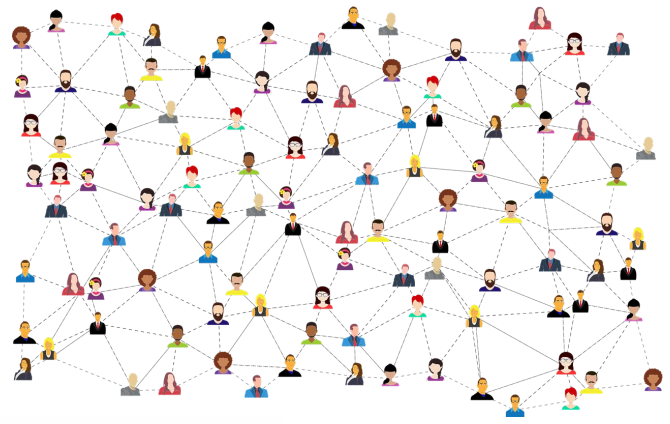


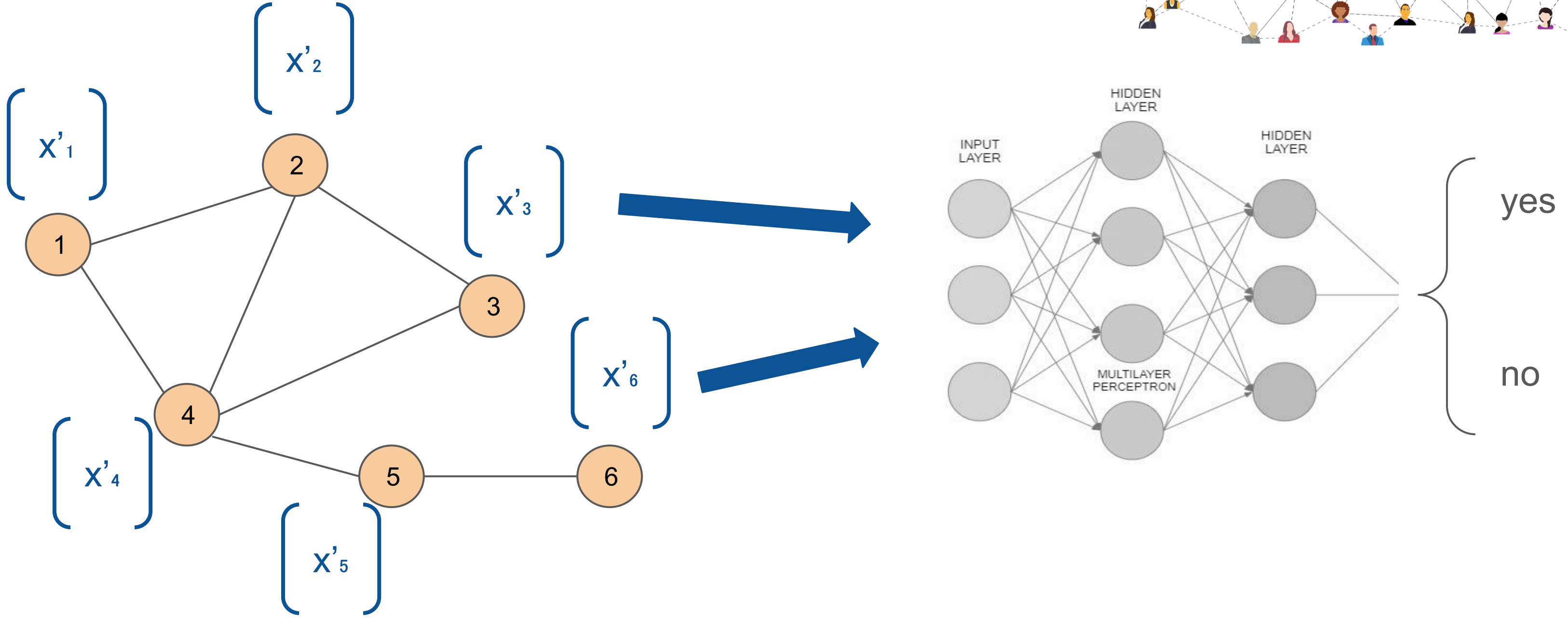




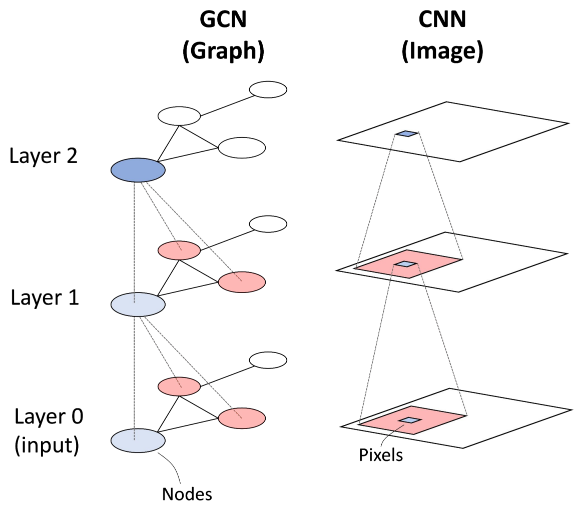
Link prediction

The problem of determining whether or not there will be an edge between two nodes in the future





Popular GNN layers



- Graph Convolutional Network **GCN**

  "Semi-Supervised Classification with Graph Convolutional Network", Kipf & Welling, 2016

$$x'_i = \sigma\Big(\textstyle\sum_{j \in N(i) \cup \{i\}} a_{ij} W x_j\Big) \quad \text{where} \quad a_{ij} = \frac{1}{\sqrt{|N(i)||N(j)|}}$$

- Graph Isomorphism Network **GIN** *"How Powerful are Graph Neural Networks?",* Xu et al., 2018

$$x'_i = f_\theta((1+\epsilon)x_i + \textstyle\sum_{j \in N(i)} x_j)$$ where f_θ is a learnable function

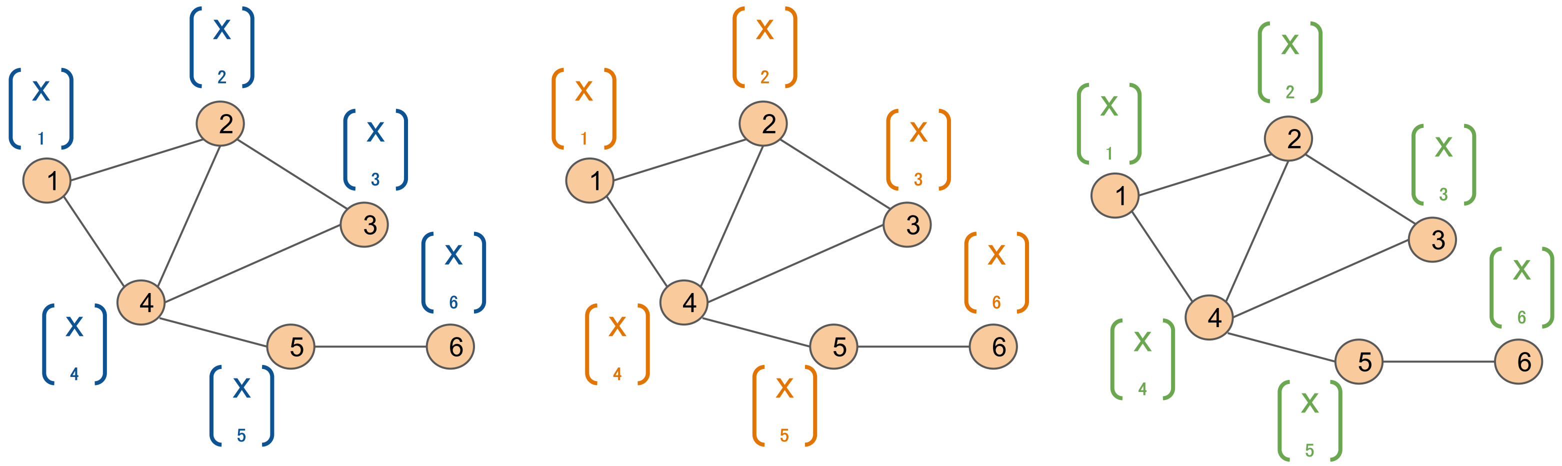
High expressive power (provably as strong as the 1-Weisfeiler Leman test)

Contents

Temporal Graph

Real world network are usually dynamic such as social networks, transportation networks and brain activity.

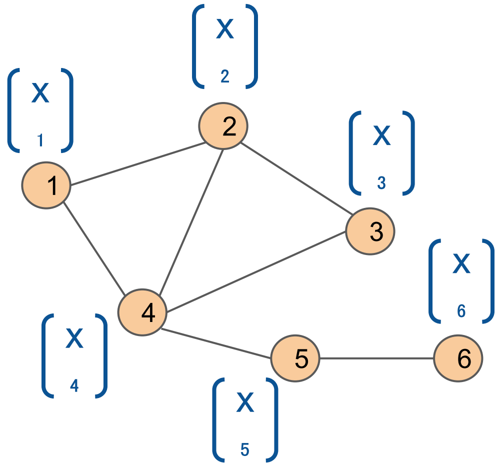
- **Static Temporal Graphs**: the graph structure is fixed, but the node and edge features change over time.

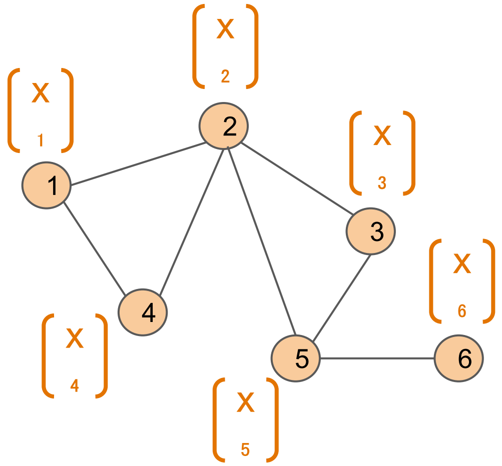


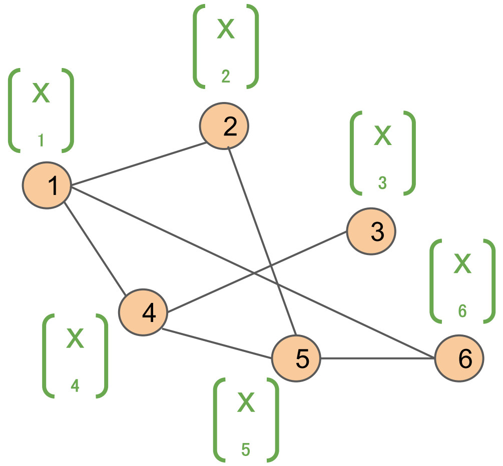
Temporal Graph

- **Dynamic Temporal Graphs**: the structure of the graph (edges) and the features of the nodes and edges change over time.

  G = {**G**^{1}, **G**^{2}, …, **G**^{T}} where each snapshot **G**^{t} = (**V**, **E**^{t}, **X**^{t}) shares the same node set **V**, but allows for time-varying edge sets and features.







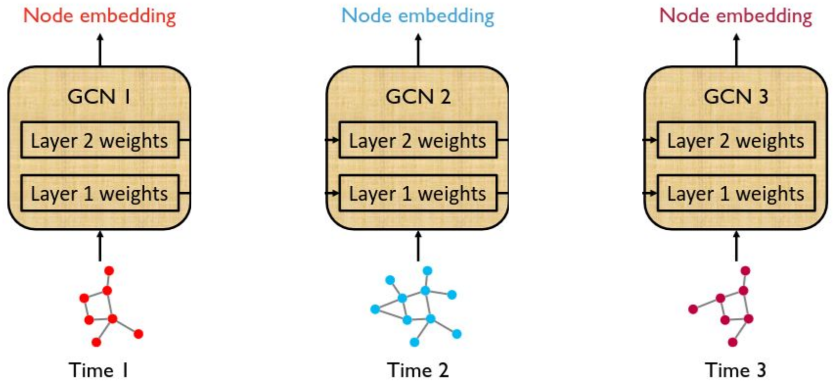
Temporal Models

Static Temporal Graphs
Spatio-temporal GNN

GNN over a fixed graph to model **spatial dependence** combined with **recurrent neural network** (GRU, LSTM) to model **temporal dependence**

Dynamic Temporal Graphs
Temporal GNN

Apply a GNN to each snapshot, or, as in **EvolveGCN**, use a recurrent neural network to evolve the GNN's weights over time



Pareja et al, 2020 AAI

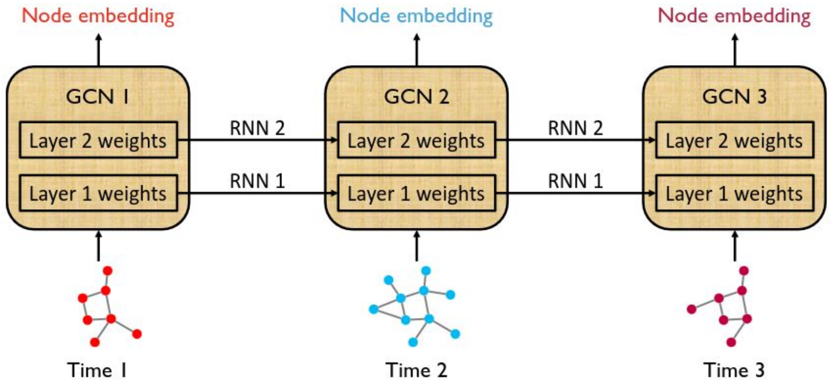
Temporal Models

Static Temporal Graphs
Spatio-temporal GNN

GNN over a fixed graph to model **spatial dependence** combined with **recurrent neural network** (GRU, LSTM) to model **temporal dependence**

Dynamic Temporal Graphs
Temporal GNN

Apply a GNN to each snapshot, or, as in **EvolveGCN**, use a recurrent neural network to evolve the GNN's weights over time



Pareja et al, 2020 AAI

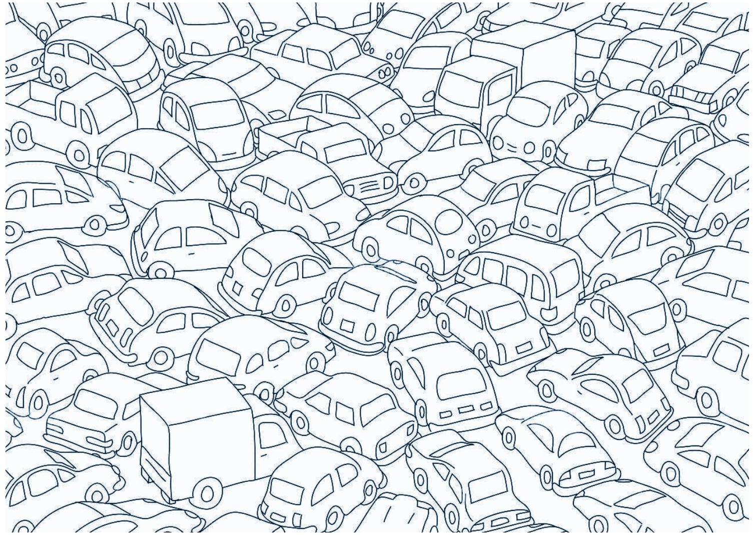
Contents

Traffic prediction

Traffic forecasting is the problem of predicting future **traffic trends** on a road network given historical traffic data such as traffic speed and time of day.

- Dataset: METR-LA

- Model from the paper *"Diffusion Convolutional Recurrent Neural Network: Data-driven Traffic Forecasting"* *Li et al. 2018,* **NeurIPS**



METR-LA Dataset

It contains traffic data from 207 sensors in highways of Los Angeles County.

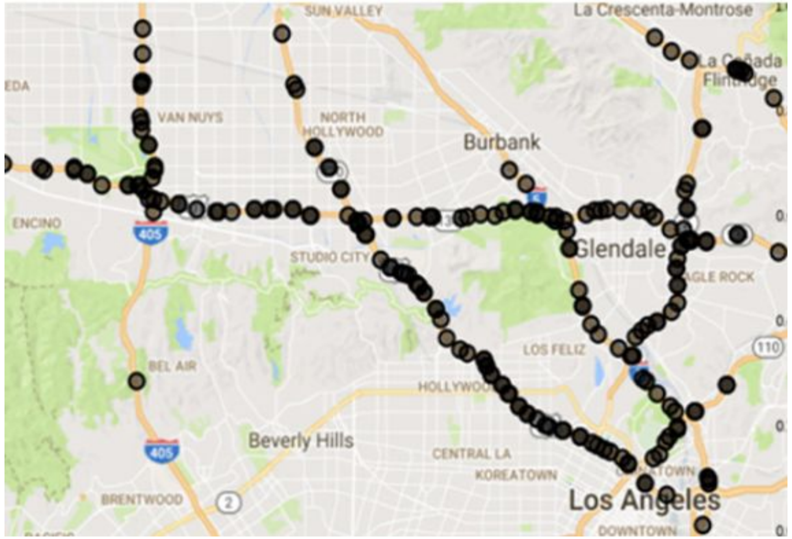
Graph nodes: sensors

Edge weights: distances between the sensors.

Node features:

- traffic speed
- time of the day

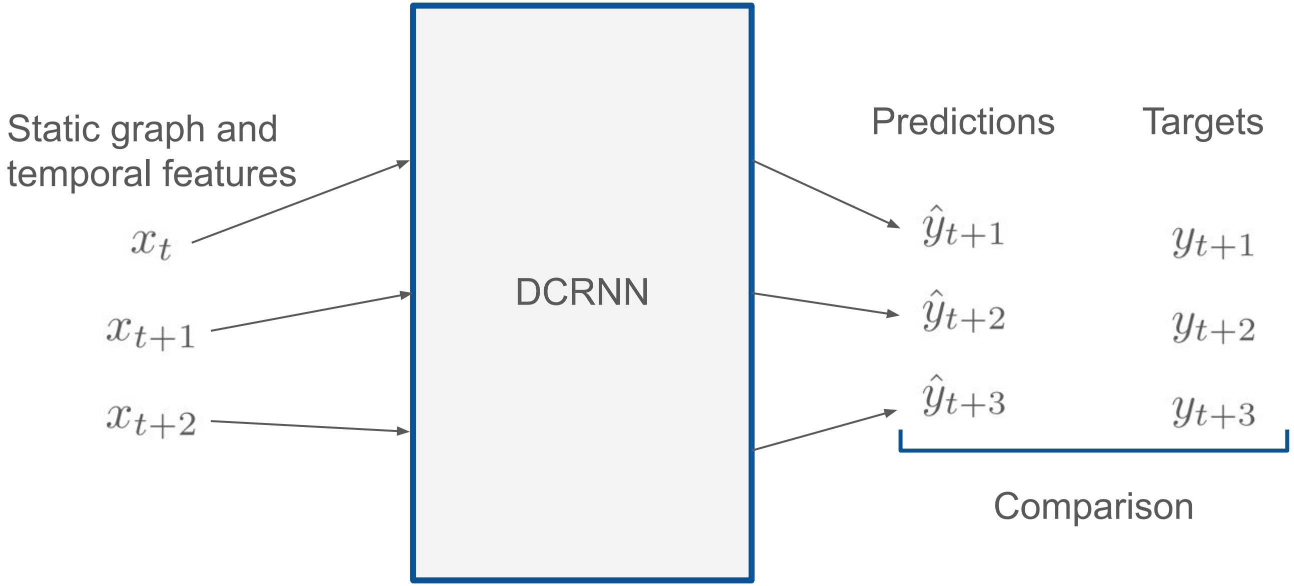
collected from March 1, 2012 to June 30, 2012 every 5 minutes.



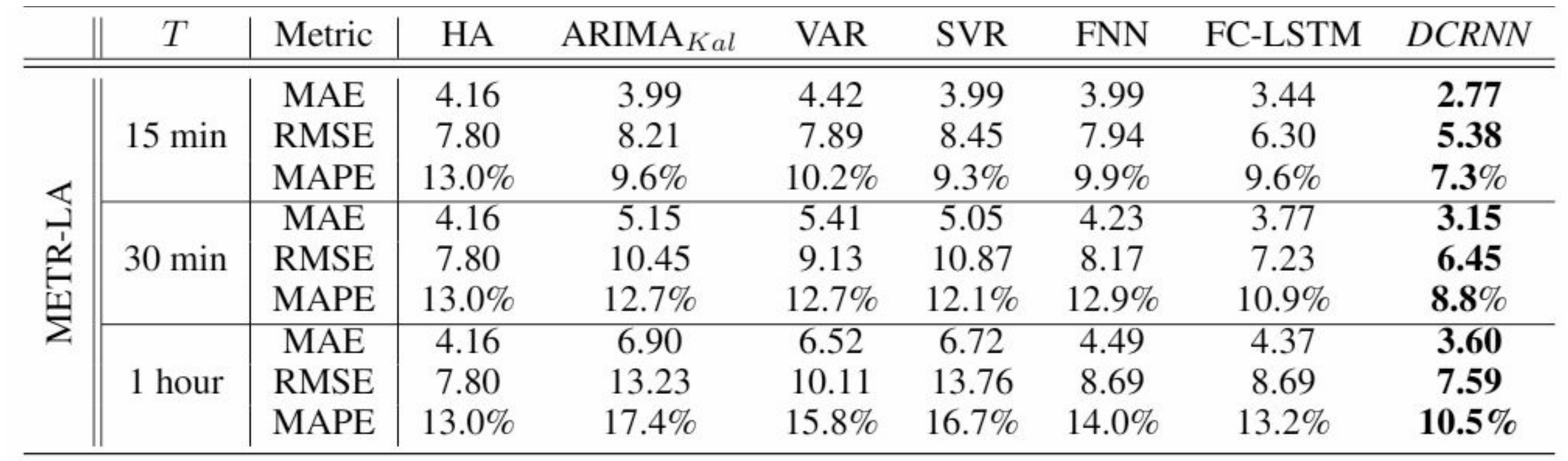
DCRNN Model

It is a recurrent model and one cell is composed by:

- Diffusion convolutional layer DConv to model **spatial dependence**
- Gated Recurrent Unit GRU to model **temporal dependence**



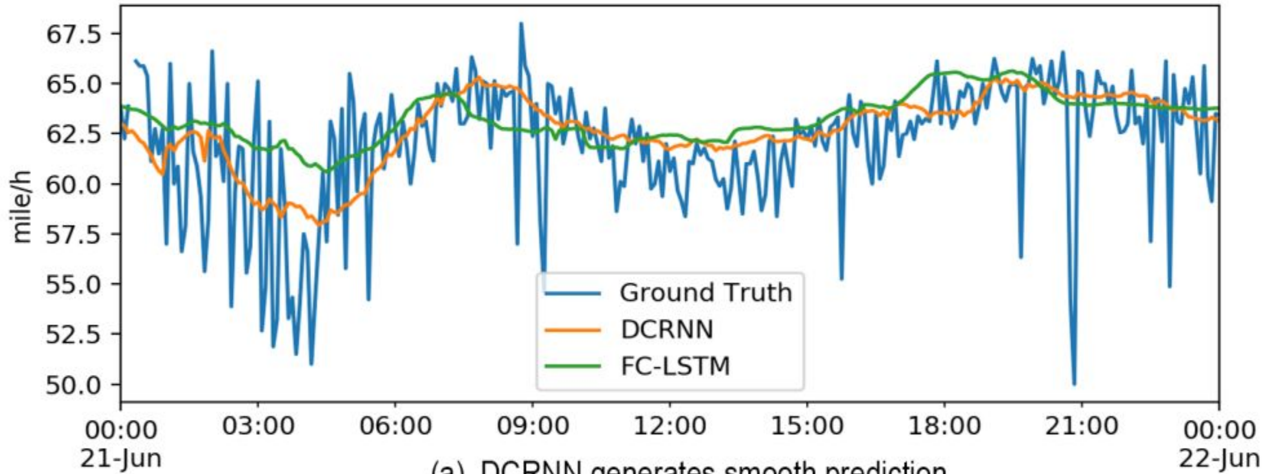
Results



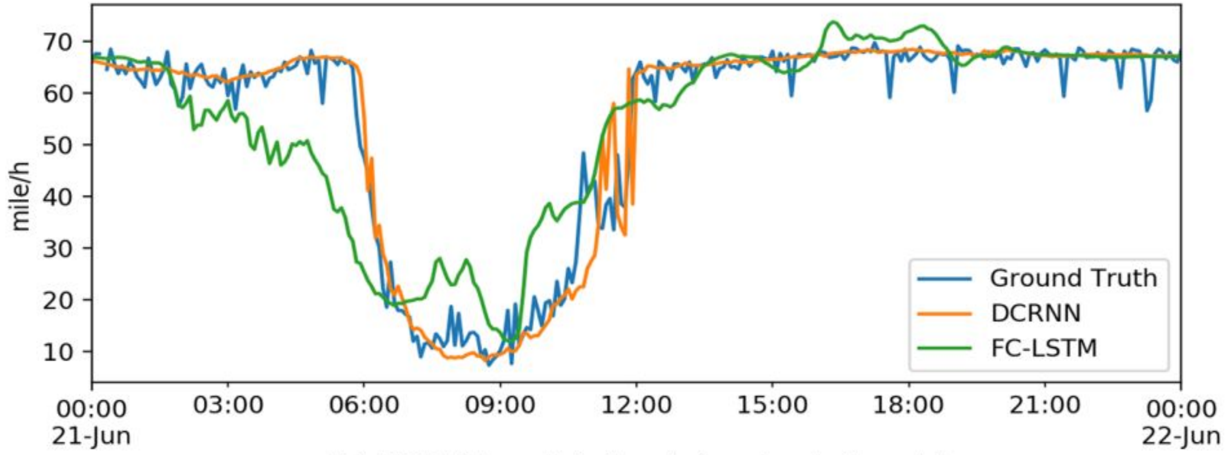
	T	Metric	HA	ARIMA_{Kal}	VAR	SVR	FNN	FC-LSTM	*DCRNN*
METR-LA	15 min	MAE	4.16	3.99	4.42	3.99	3.99	3.44	**2.77**
		RMSE	7.80	8.21	7.89	8.45	7.94	6.30	**5.38**
		MAPE	13.0%	9.6%	10.2%	9.3%	9.9%	9.6%	**7.3%**
	30 min	MAE	4.16	5.15	5.41	5.05	4.23	3.77	**3.15**
		RMSE	7.80	10.45	9.13	10.87	8.17	7.23	**6.45**
		MAPE	13.0%	12.7%	12.7%	12.1%	12.9%	10.9%	**8.8%**
	1 hour	MAE	4.16	6.90	6.52	6.72	4.49	4.37	**3.60**
		RMSE	7.80	13.23	10.11	13.76	8.69	8.69	**7.59**
		MAPE	13.0%	17.4%	15.8%	16.7%	14.0%	13.2%	**10.5%**

HA: Historical Average
ARIMAkal: Auto-Regressive Integrated Moving Average
VAR: Vector Auto-Regression
SVR: Support Vector Regression
FNN:Neural network with two hidden layers and L2 regularization
FC-LSTM: Recurrent Neural Network with fully connected LSTM hidden units

Results



(a). DCRNN generates smooth prediction.

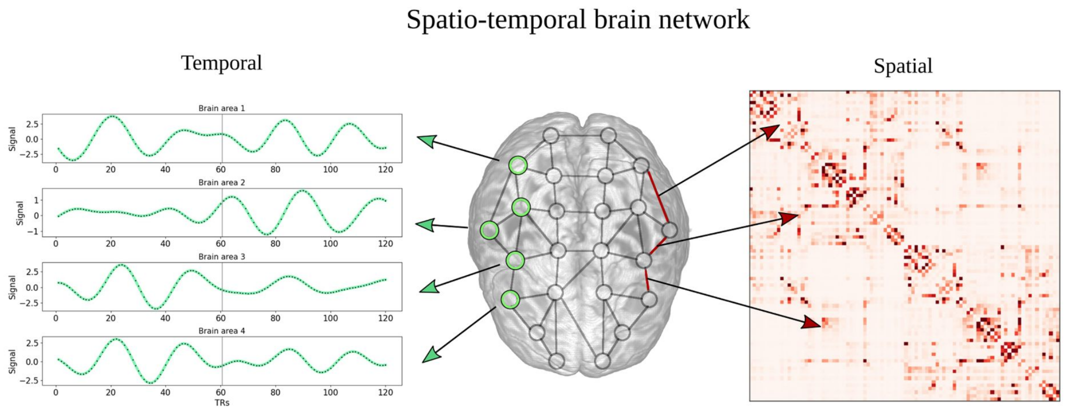


(b). DCRNN predicts the start and end of peak hours.

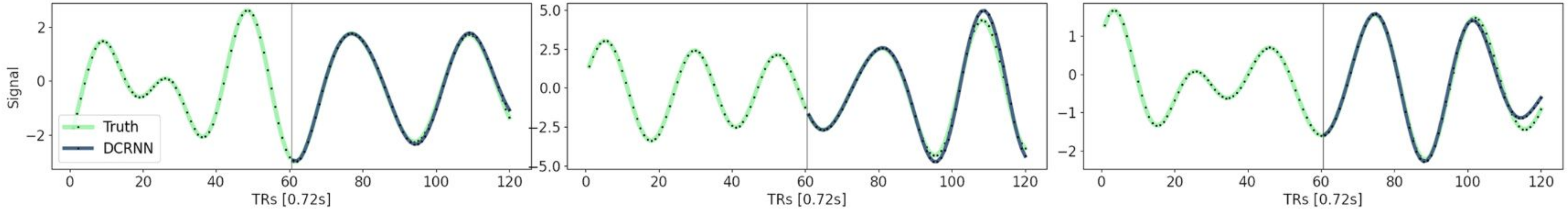
Brain activity prediction

DCRNN model applied to the networks obtained from **resting-state fMRI** data from the Human Connectome Project (HCP).
"*Forecasting brain activity based on models of spatiotemporal brain dynamics: A comparison of graph neural network architectures*" Wein, 2022 Network Neuroscience

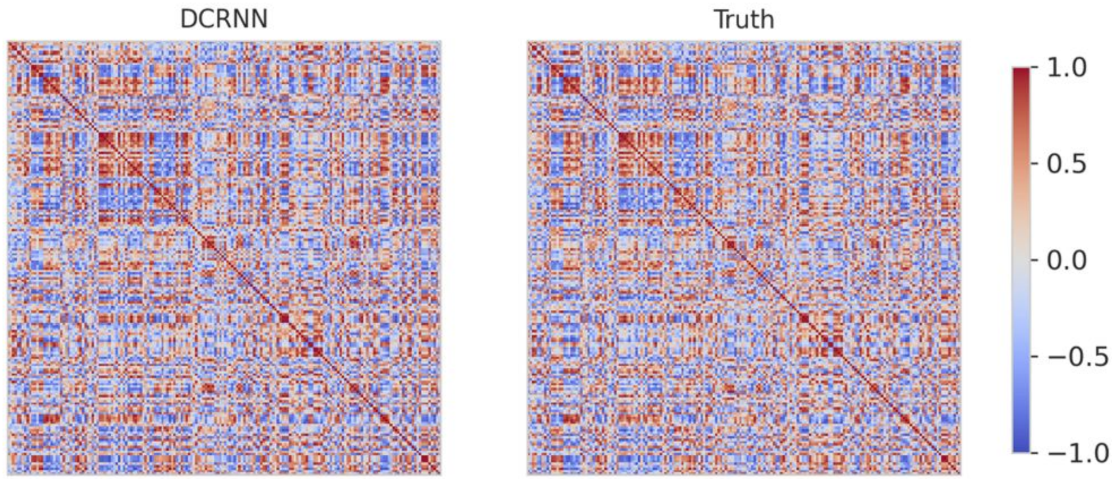
Graph nodes: brain regions 180 Glasser atlas
Edge weights: number of fibers connecting two brain regions
Node features: time series of brain activity of the region



Brain activity prediction



MAE = 0.1158



Temporal Katz centrality

A **temporal graph** defined as $G = (V, E, T)$, where V, E and T are the sets of the vertices, the temporal edges, and the timestamps. A temporal edge $e = (u, v, t)$ from node u to v at time t in T.

A **temporal walk** z from node u_0 to $u_{\square}$ is an ordered series of nodes and temporal edges in a temporal network, represented by $z = (u_0, u_1, t_1), (u_1, u_2, t_2), \ldots, (u_{\square-1}, u_{\square}, t_{\square})$, such that $\forall\ 1 \leq i < n, t_{\square} < t_{\square}+1$.

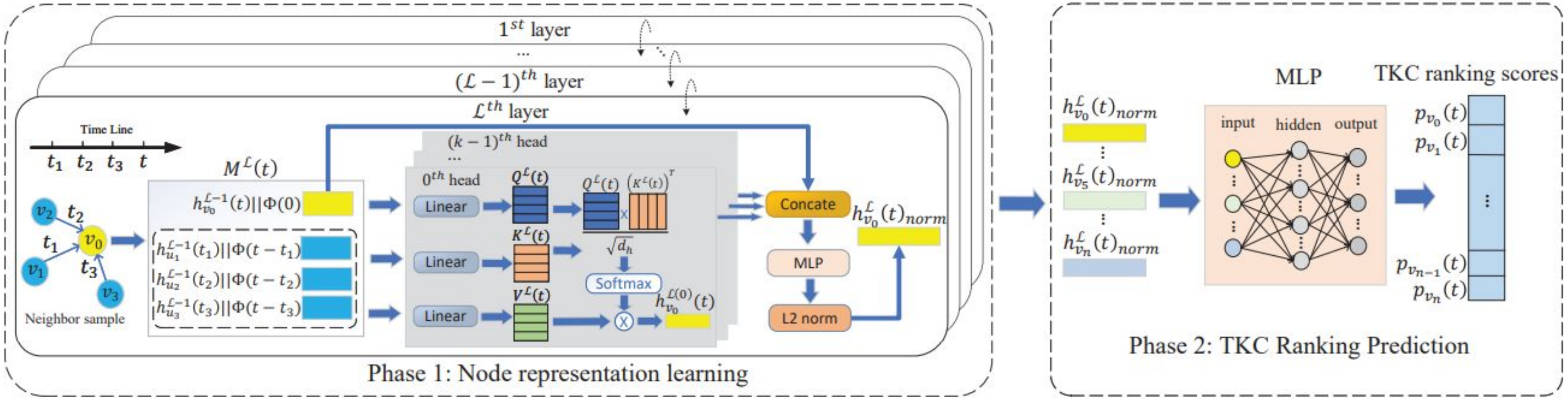
The **temporal Katz centrality** of a node $u \in V$ at time t is the weighted sum of all temporal walks that end in node u, denoted by:

$$r_u(t) := \sum_{v} \sum_{temporal\ walks\ z\ from\ v\ to\ u} (\beta)^n \cdot exp(-c(t - t_1))$$

TATKC Model

“TATKC: A Temporal Graph Neural Network for Fast Approximate Temporal Katz Centrality Ranking”, Zhang 2024, WWW24

At each timestamp t, a subset of neighbors N□(v) is sampled for every node v, prioritizing those with higher out-degree; their messages are then combined through attention-based weighted aggregation to form a temporal embedding for v, which a lightweight MLP converts directly into Katz-inspired ranking scores.


1st layer
...
$(\mathcal{L}-1)^{th}$ layer
$\mathcal{L}^{th}$ layer
Time Line
t_1 t_2 t_3 t
Neighbor sample
$M^{\mathcal{L}}(t)$
$h_{v_0}^{\mathcal{L}-1}(t)||\Phi(0)$
$h_{u_1}^{\mathcal{L}-1}(t_1)||\Phi(t-t_1)$
$h_{u_2}^{\mathcal{L}-1}(t_2)||\Phi(t-t_2)$
$h_{u_3}^{\mathcal{L}-1}(t_3)||\Phi(t-t_3)$
$(k-1)^{th}$ head
0^{th} head
Linear
Linear
Linear
$Q^{\mathcal{L}}(t)$
$K^{\mathcal{L}}(t)$
$V^{\mathcal{L}}(t)$
$Q^{\mathcal{L}}(t)$ $(K^{\mathcal{L}}(t))^T$
$\sqrt{d_h}$
Softmax
$h_{v_0}^{\mathcal{L}(0)}(t)$
Concate
MLP
L2 norm
$h_{v_0}^{\mathcal{L}}(t)_{norm}$
Phase 1: Node representation learning
$h_{v_0}^{\mathcal{L}}(t)_{norm}$
$h_{v_5}^{\mathcal{L}}(t)_{norm}$
$h_{v_n}^{\mathcal{L}}(t)_{norm}$
MLP
input
hidden
output
TKC ranking scores
$p_{v_0}(t)$
$p_{v_1}(t)$
$p_{v_{n-1}}(t)$
$p_{v_n}(t)$
Phase 2: TKC Ranking Prediction

Results

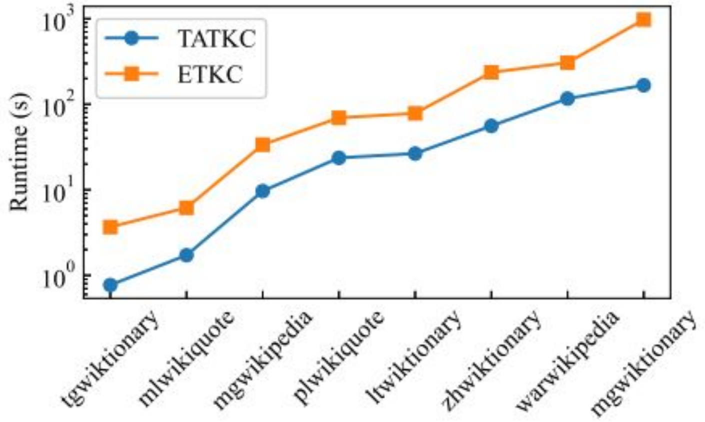
Performance evaluation:

$$\text{Top-N\%} = \frac{|\text{predicted top-N\% nodes} \cap \text{true top-N\% nodes}|}{\lceil |V| \times \text{N\%} \rceil}$$

Dataset	**|V|**	**|E|**	**|T|**
tgwiktionary	33,968	81,516	67,065
mlwikiquote	43,889	142,340	137,389
mgwikipedia	220,064	750,811	736,680
plwikiquote	581,646	1,472,273	1,452,278
ltwiktionary	689,678	1,693,277	1,633,334
zhwiktionary	1,347,094	5,276,371	4,448,306
warwikipedia	2,877,072	6,145,080	5,918,117
mgwiktionary	4,064,239	22,720,139	19,759,219

ETKC algorithm for exact Katz centrality computation

KT kendall tau correlation



Dataset	**Top-1%**			**Top-5%**			**Top-10%**			**Top-20%**			**KT**		
	TATKC	**TATKC***	**TGAT**	**TATKC**	**TATKC***	**TGAT**	**TATKC**	**TATKC***	**TGAT**	**TATKC**	**TATKC***	**TGAT**	**TATKC**	**TATKC***	**TGAT**
tgwiktionary	**90.68±0.96**	75.39±1.98	64.87±1.90	**93.77±0.17**	85.44±0.59	80.44±1.19	**88.41±0.54**	84.12±0.31	81.07±0.65	**92.53±0.21**	88.42±0.40	86.07±0.36	**80.02±0.42**	78.87±0.21	77.67±0.76
mlwikiquote	**85.27±0.88**	60.86±1.43	61.19±1.67	**89.37±0.82**	73.56±1.72	69.02±1.23	**89.12±0.61**	76.35±1.17	73.73±0.70	**91.82±0.29**	81.85±0.26	80.49±0.42	**86.60±0.23**	82.70±0.37	82.04±0.38
mgwikipedia	**80.01±0.46**	65.96±0.51	49.41±0.48	**86.67±0.14**	76.88±1.02	67.67±0.33	**97.47±0.05**	94.92±0.19	91.29±0.14	**94.85±0.03**	94.55±0.08	94.12±0.08	**76.98±0.08**	76.58±0.12	76.29±0.14
plwikiquote	**85.34±0.18**	73.33±0.93	66.98±0.58	**85.08±0.02**	74.37±0.38	70.95±0.24	**84.76±0.27**	76.02±0.18	73.69±0.23	**85.97±0.18**	78.74±0.22	77.47±0.12	**82.49±0.17**	79.60±0.03	79.03±0.04
ltwiktionary	**88.14±0.82**	60.45±0.13	56.48±0.21	**94.32±0.08**	89.51±0.08	83.99±0.13	**95.29±0.05**	94.03±0.04	92.67±0.11	**94.98±0.03**	94.48±0.05	94.38±0.02	**70.68±0.04**	70.34±0.06	70.17±0.04
zhwiktionary	**72.20±0.41**	57.48±0.11	55.40±0.18	**91.48±0.31**	82.31±0.68	73.42±0.31	**88.89±0.36**	83.13±0.29	79.13±0.31	**91.88±0.03**	89.93±0.14	87.90±0.17	**84.90±0.03**	83.13±0.29	82.14±0.11
warwikipedia	**91.35±0.08**	72.02±0.28	58.46±0.21	**95.74±0.03**	94.01±0.05	93.48±0.14	**76.32±0.11**	75.08±0.27	75.63±0.09	**78.13±0.03**	78.04±0.04	78.07±0.09	**71.28±0.02**	71.13±0.04	71.10±0.02
mgwiktionary	**90.25±0.21**	60.33±0.87	46.59±1.43	**89.81±0.26**	76.84±0.40	62.69±0.98	**90.10±0.14**	79.21±0.19	69.29±0.73	**97.61±0.16**	84.41±0.09	78.17±0.48	**84.65±0.06**	79.42±0.10	75.12±0.33

Thank you

Questions?