CSCI587 – Lecture 18
Spatiotemporal Forecasting
10/30/2024

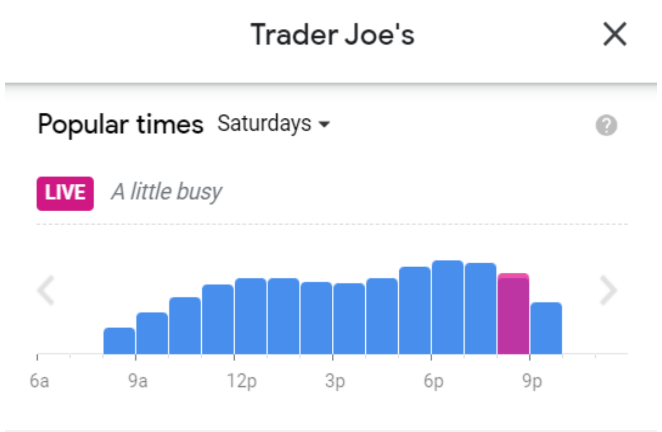
University of Southern California
USC Viterbi School of Engineering
Fall 2024

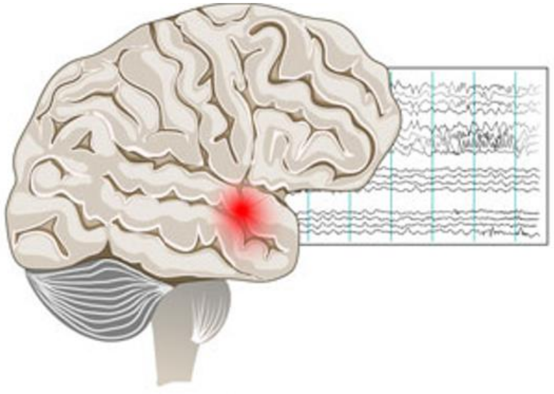
Introduction – Spatiotemporal Data

- **Spatiotemporal Data:** Where + When
- **Unique Characteristics**
  - Massive amounts of high-frequency data from numerous locations
  - Rich underlying patterns across both space and time
- **Why it matters?**
  - Critical for decision making in many domains, e.g., urban planning

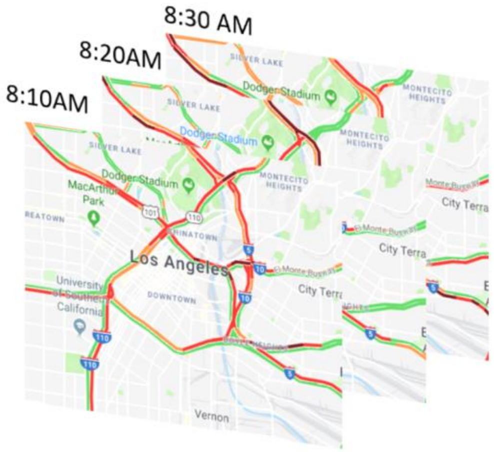
Introduction – Spatiotemporal Forecasting

- **Spatiotemporal Forecasting:** Predict events or conditions in space & time by analyzing spatiotemporal data
- **Examples**
  - **POI Visit Forecasting:** Predicting the # of visits to specific POIs at different times
  - **Seizure Detection:** Predicting the occurrence of seizures in specific brain regions during particular time intervals.
  - **Traffic Forecasting:** Predicting traffic flow on roads during various times of day





Focal seizure



You've already learned about **Spatial** data…

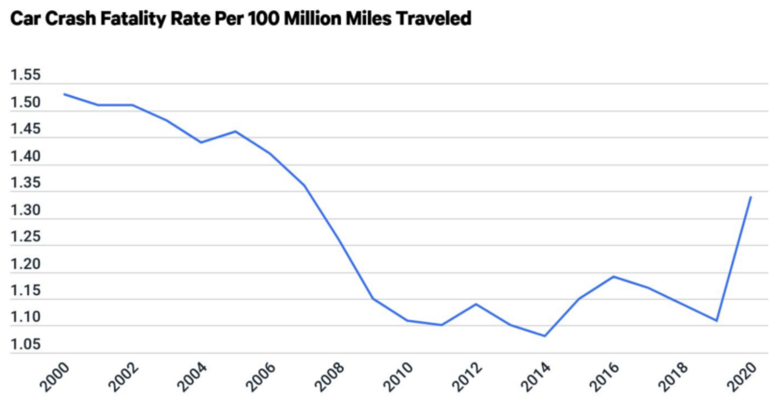
…but what is the Temporal dimension?

Time Series

Time series

A sequence of observations collected over time.

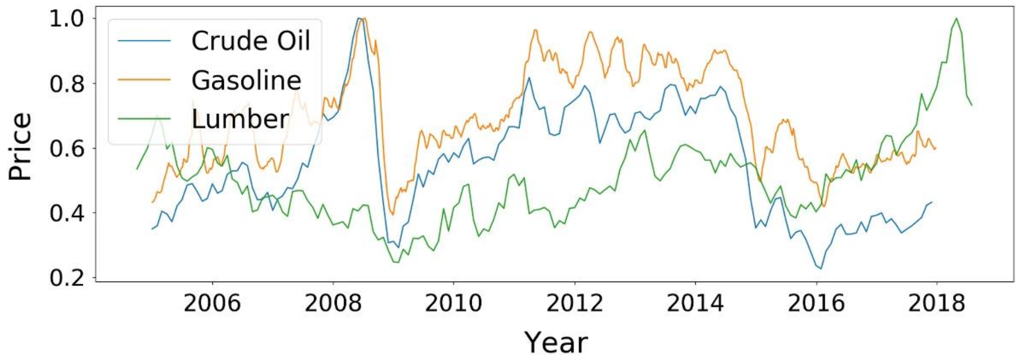
$$x_i = (x_{i1}, x_{i2}, \dots, x_{iT}) \in \mathbb{R}^T$$



Multivariate Time series

A collection of two or more time series observed over time, with each variable being dependent on its own past values as well as the past values of the other series.

$$X = (x_1, x_2, \dots, x_N) \in \mathbb{R}^{N \times T}$$



Time Series Forecasting

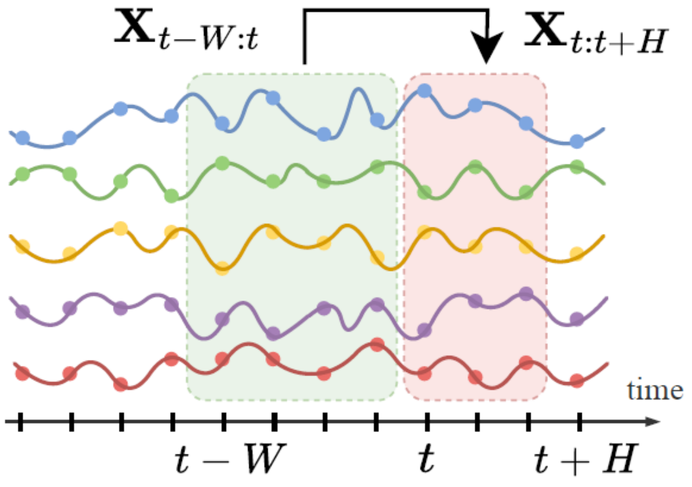
Time Series Forecasting

Given a window of ***W*** $\geq$ **1** of **past** observations:

$$X_{t-W:t} = [X_{t-W}, \dots, X_{t-1}],$$

Predict ***H*** $\geq$ **1 future** observations:

$$X_{t:t+H} = [X_t, \dots, X_{t+H-1}]$$



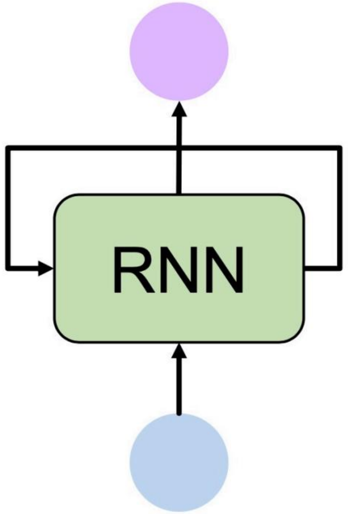
Thus, the goal is to learn a **model** $\boldsymbol{F}$ with parameters $\boldsymbol{\theta}$ that maps past observations to future values:

$$\boldsymbol{F}(\boldsymbol{X}_{t-W:t}, \boldsymbol{\theta}) = \boldsymbol{X}_{t:t+H}$$

Sequence Modeling

To model **sequences**, we need to:

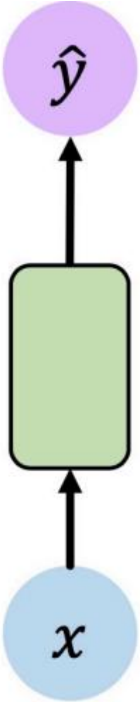
- Handle **variable-length** sequences
- Track **long-term** dependencies
- Maintain information about **order**
- **Share** parameters across the sequence



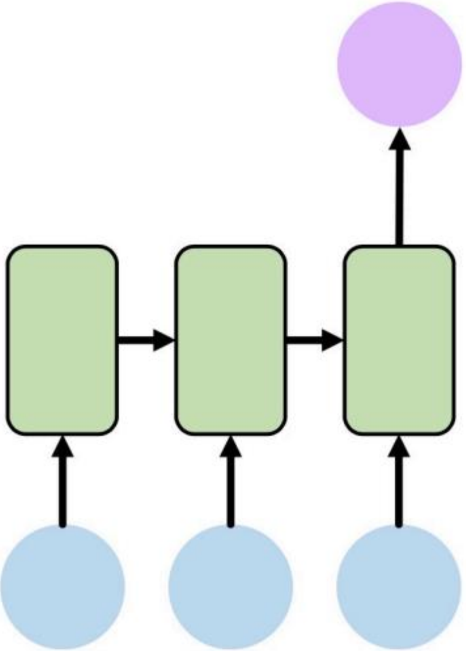
Popular solution:

- Recurrent Neural Networks (**RNNs**)

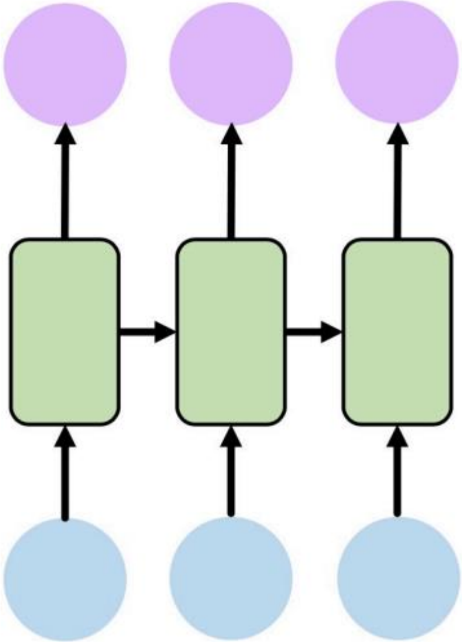
Recurrent Neural Networks



One to One
"Vanilla" Neural Network



Many to One
E.g., Sentiment Classification



Many to Many
E.g., Music Generation

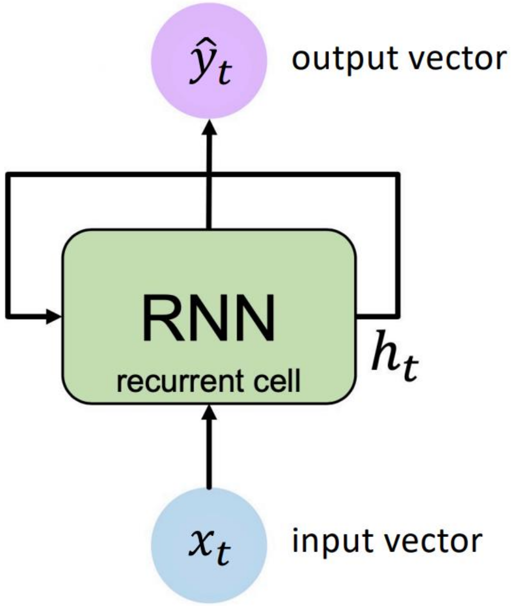
A Recurrent Neural Network (RNN)

- Apply a recurrence relation at every time step to process a sequence:

cell state   old state

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

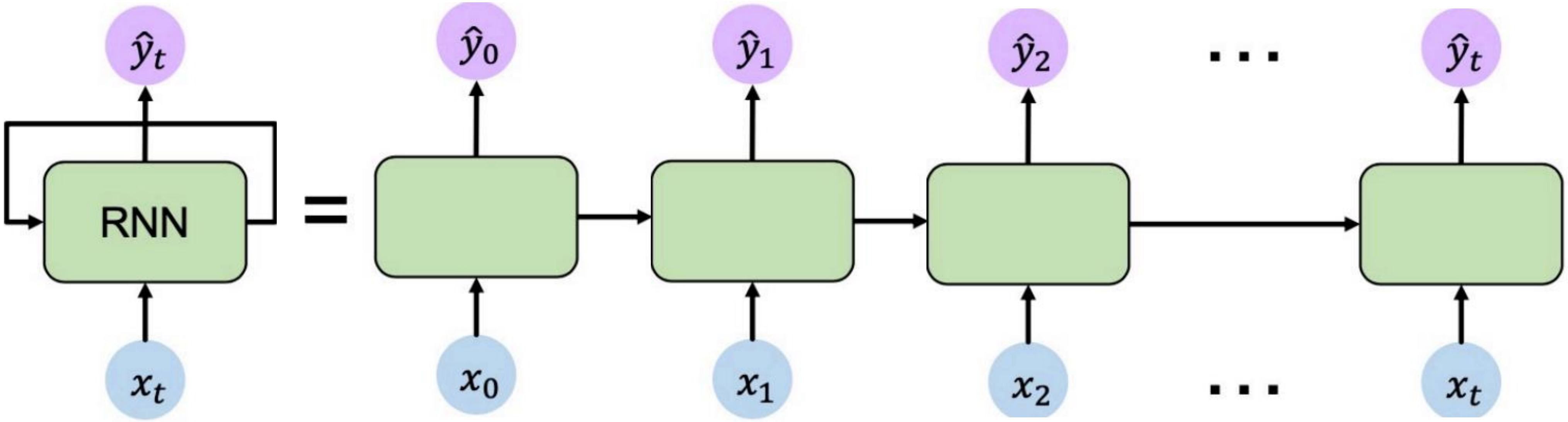
A function parameterized by W   current input



Note: The **same function** and **set of parameters** are used at every time step.

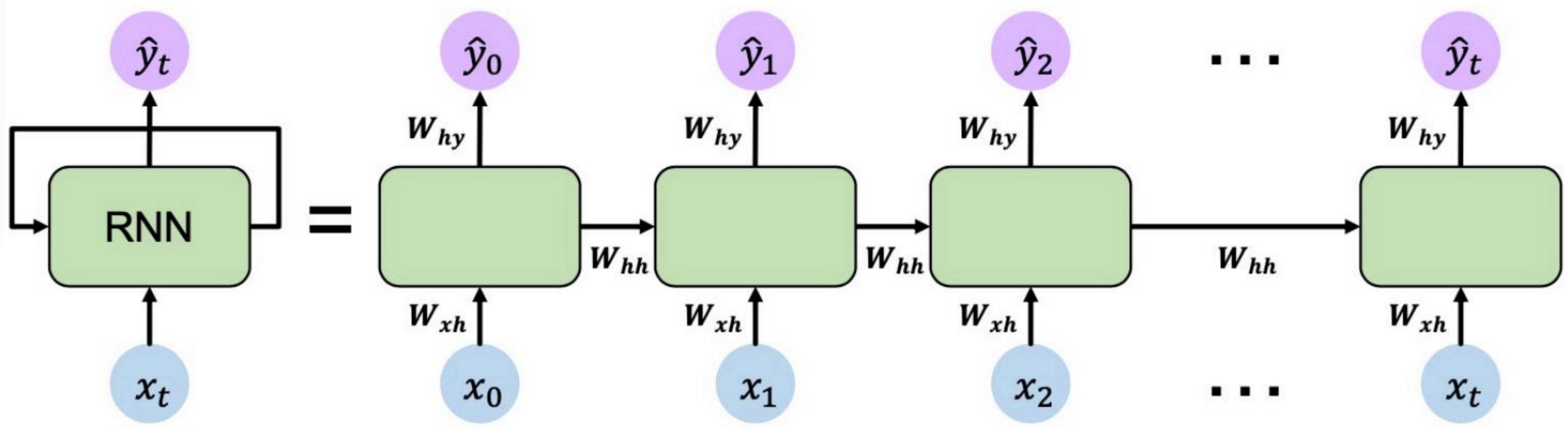
RNN – Computational Graph Across Time

- Represent as computational graph **unrolled** over time



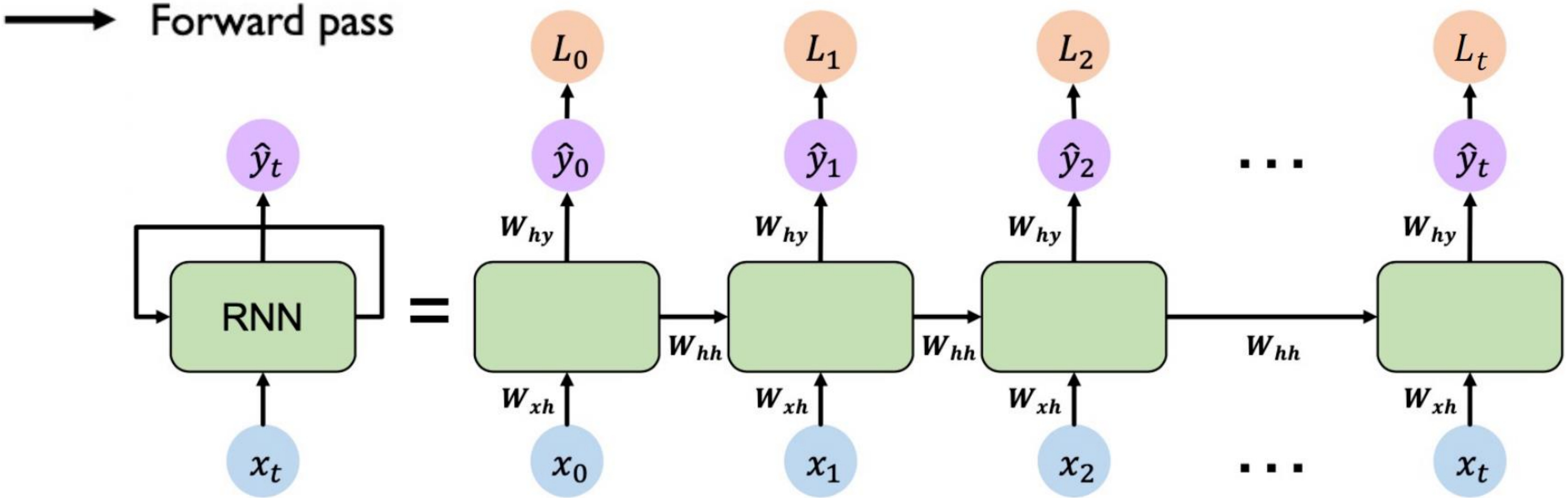
RNN – Computational Graph Across Time

- Re-use the **same weight matrices** at every time step



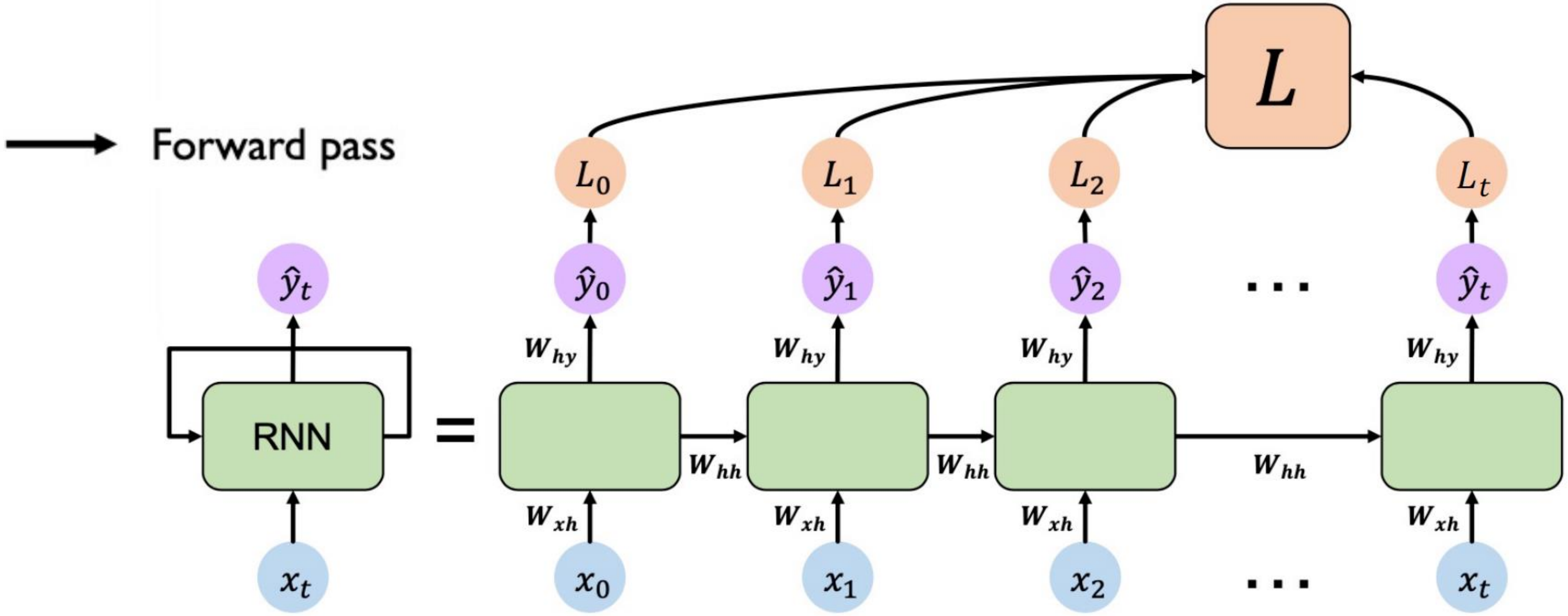
RNN – Computational Graph Across Time

- Compute the loss L_t by comparing $\hat{y_t}$ and y_t (y_t is ground truth)
  - E.g., $L_t = (\hat{y_t} - y_t)^2$



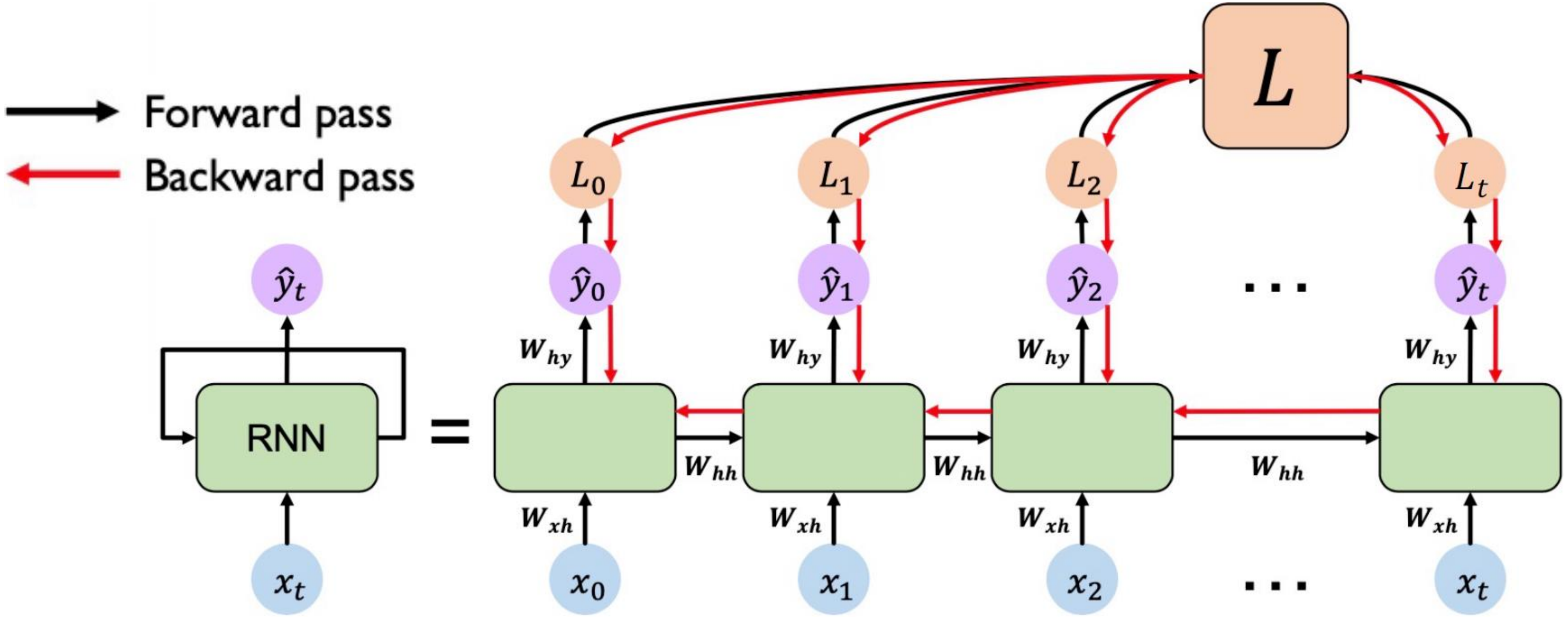
RNN – Computational Graph Across Time

- Total Loss: $L = \sum_{t=1}^{T} L_t$

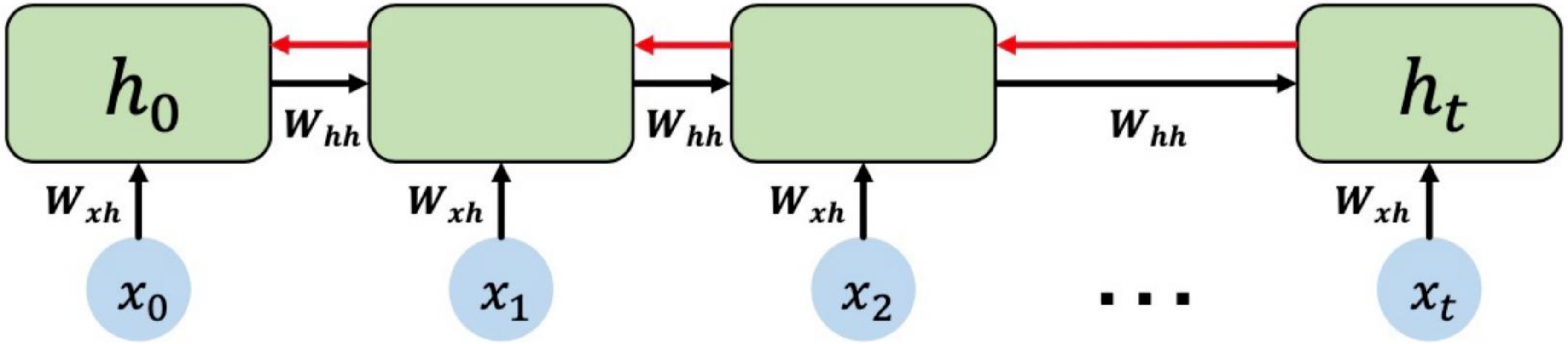


RNN – Backpropagation Through Time

- For backpropagation, we need to compute the gradients w.r.t. $\boldsymbol{W_{hy}}, \boldsymbol{W_{hh}}, \boldsymbol{W_{xh}}$



RNN – Backpropagation Through Time



Computing the gradient involves **many multiplications** (and repeated f')

- When w_{hh} changes (in a small amount), how much would L change?

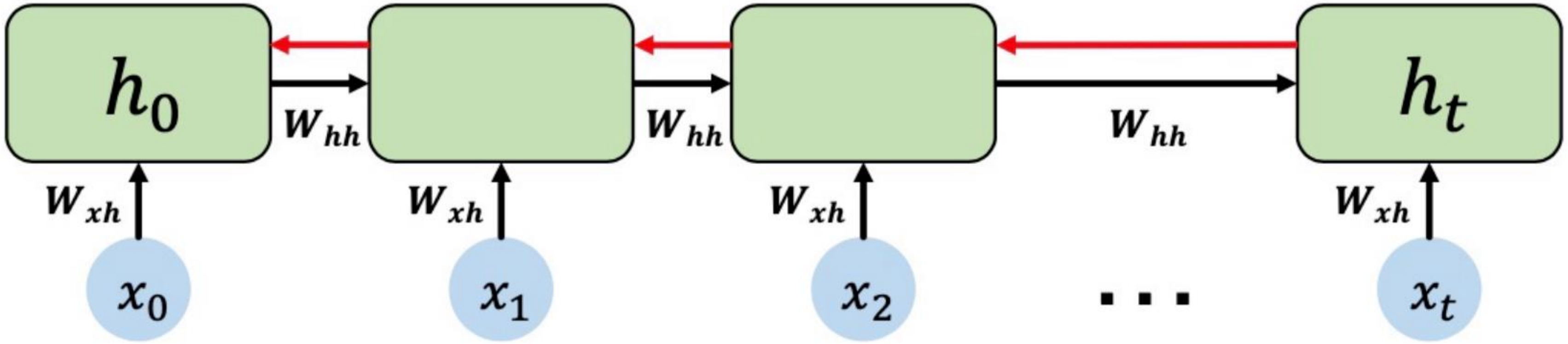
For example, $\frac{\partial L}{\partial w_{hh}} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t,\hat{y}_t)}{\partial w_{hh}}$

$$h_t = \tanh(\boldsymbol{W_{xh}}x_t + \boldsymbol{W_{hh}}h_{t-1})$$

$$= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial l(y_t,\hat{y}_t)}{\partial \hat{y}_t}\frac{\partial g(h_t,w_{hy})}{\partial h_t}\boxed{\frac{\partial h_t}{\partial w_{hh}}}$$

$$\frac{\partial h_t}{\partial w_{hh}} = \frac{\partial f(x_t,h_{t-1},w_{hh})}{\partial w_{hh}} = \frac{\partial f(x_t,h_{t-1},w_{hh})}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial w_{hh}}.$$

RNNs – Gradient Flow Issues



Computing the gradient involves **many multiplications** (and repeated f')

Case 1: Many values are > 1

Exploding gradients

Trick : Gradient clipping to scale big gradients

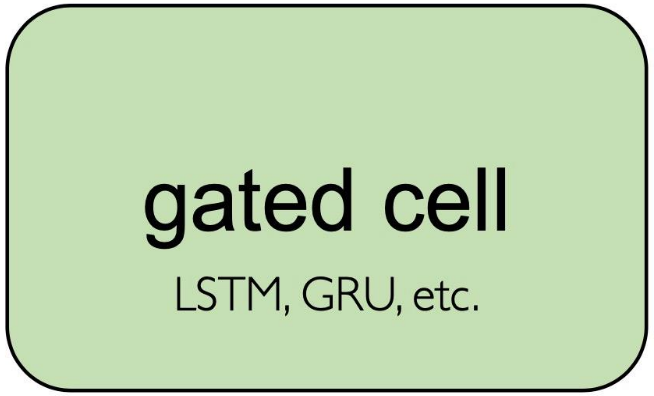
Case 2: Many values are < 1

Vanishing gradients

Trick 1: Activation functions
Trick 2: Network architecture

Remedy to the Gradient Flow Issues

- Use a more complex recurrent unit with **gates** to **control what information is passed through**



- **Long Short-Term Memory (LSTM)** and **Gated Recurrent Unit (GRU)** networks rely on gated cells to track information throughout many time steps.

Gated Recurrent Unit (GRU)

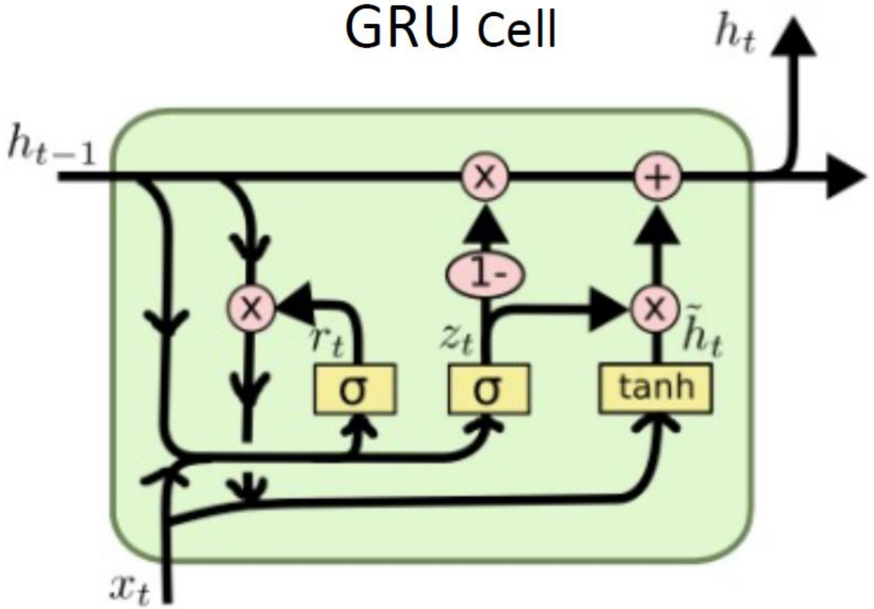
- GRU is an RNN variant with **gating** mechanisms to control **information flow**, helping to prevent **vanishing gradients**.

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

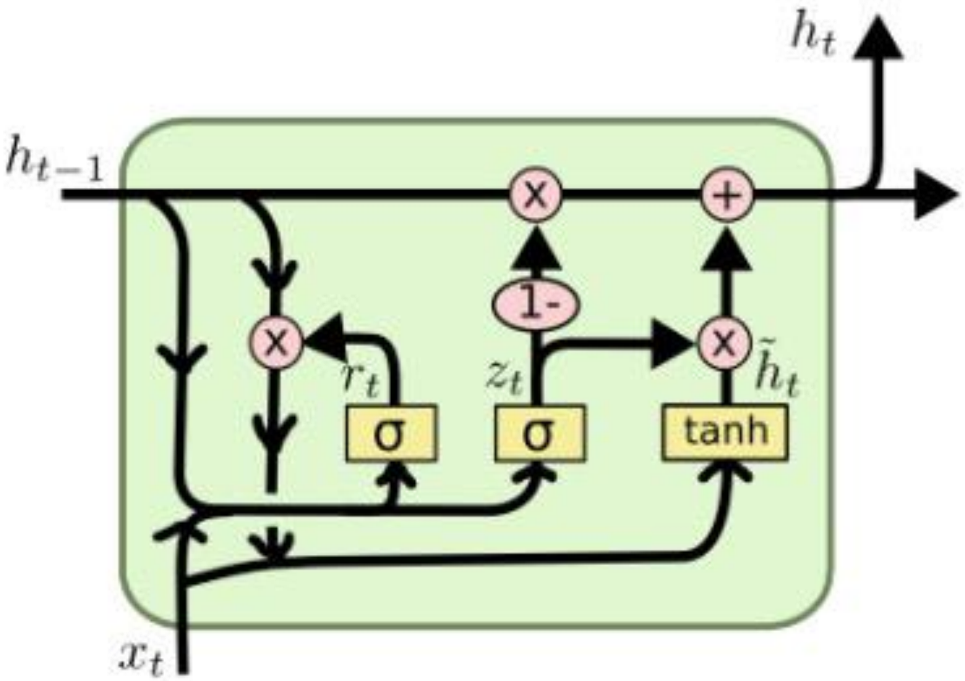
$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$$

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

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



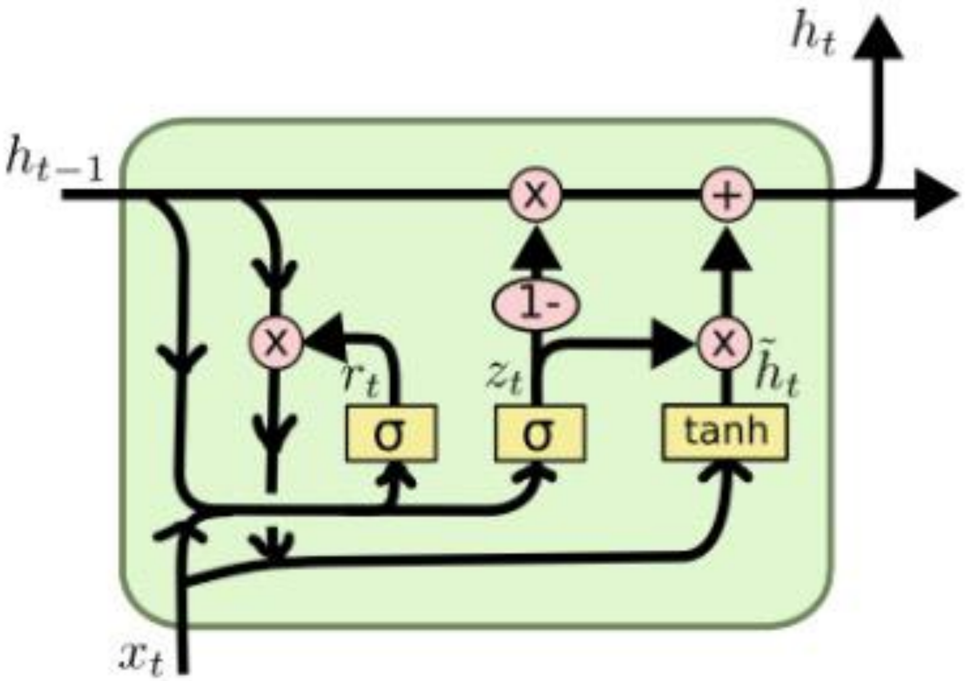
GRU – Update Gate



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

- Concatenate previous hidden state and current input
- Update gate controls what parts of hidden state are **updated** (used as z_t) vs. **preserved** (used as $(1 - z_t)$)

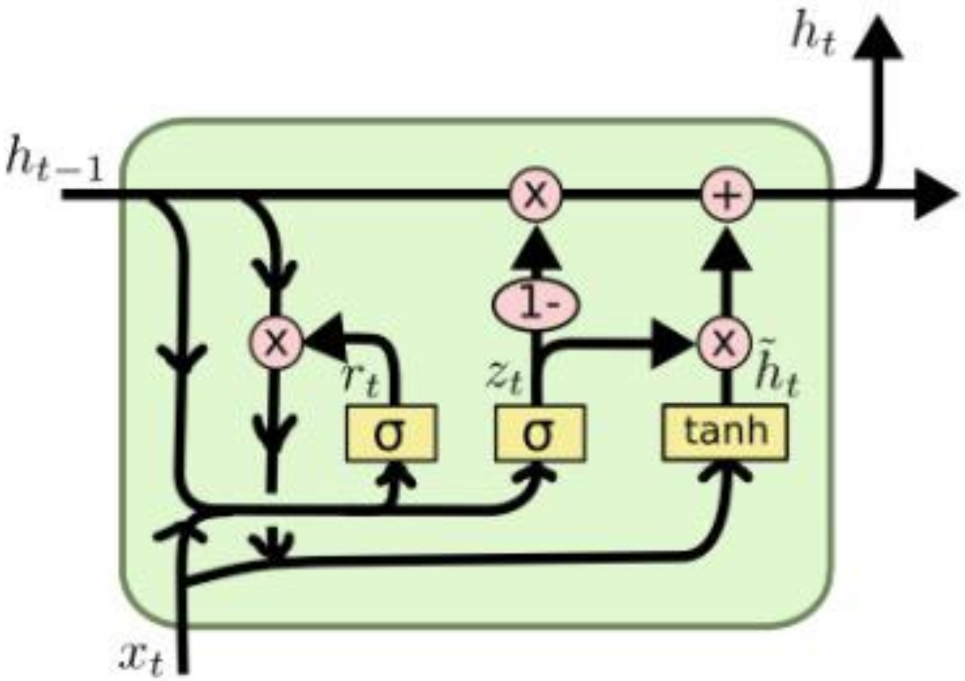
GRU – Reset Gate



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

- Reset gate controls what parts of **previous hidden state** are used to compute new content

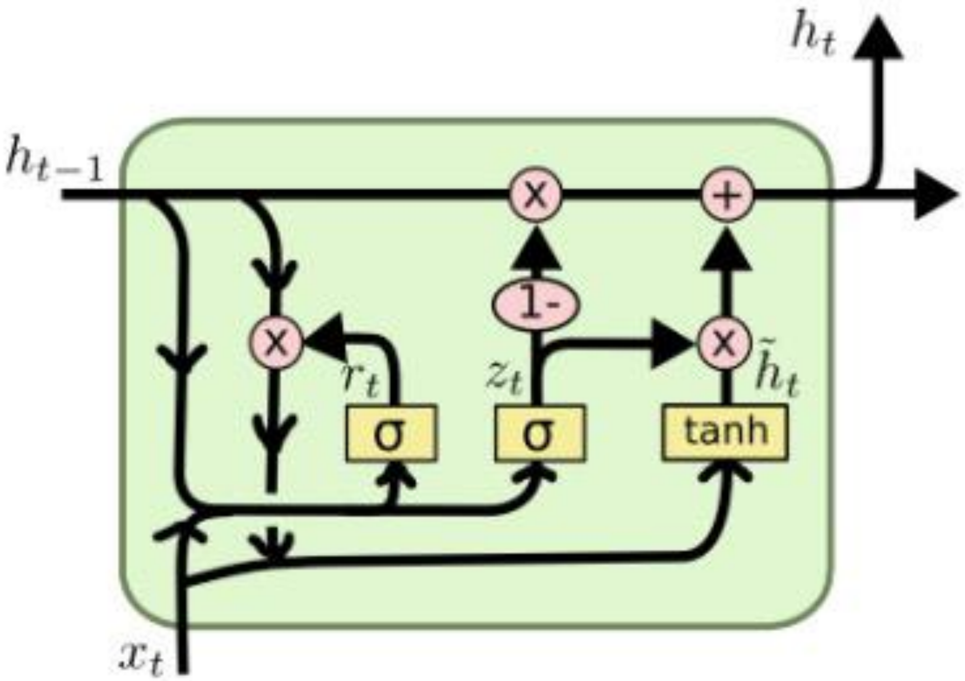
GRU – New Hidden State Content



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

- r_t selects useful parts of **previous hidden state**
- Use $r_t \odot h_{t-1}$ and **current input** to compute new hidden content

GRU – Output Hidden State

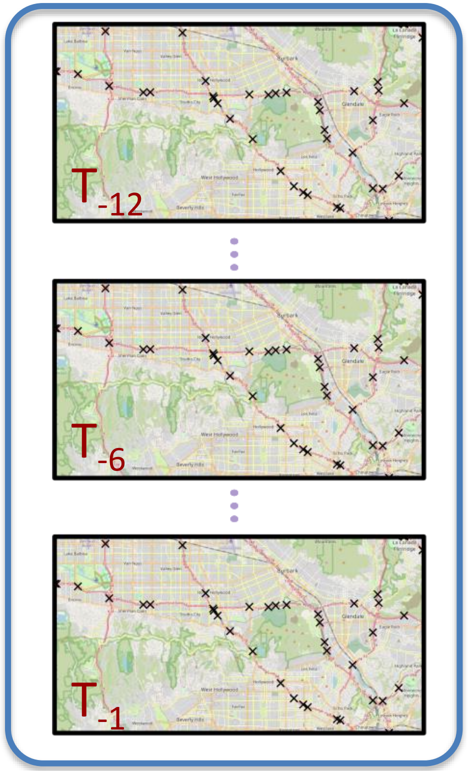


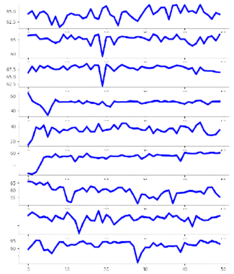
$$h_t = z_t \odot h_{t-1} + (1 - z_t) * \tilde{h}_t$$

- Update gate simultaneously controls what is **kept from previous hidden state**, and what is **updated to new hidden state content**

…Let’s get back into the spatiotemporal tasks

Spatiotemporal Application: Traffic Forecasting Task

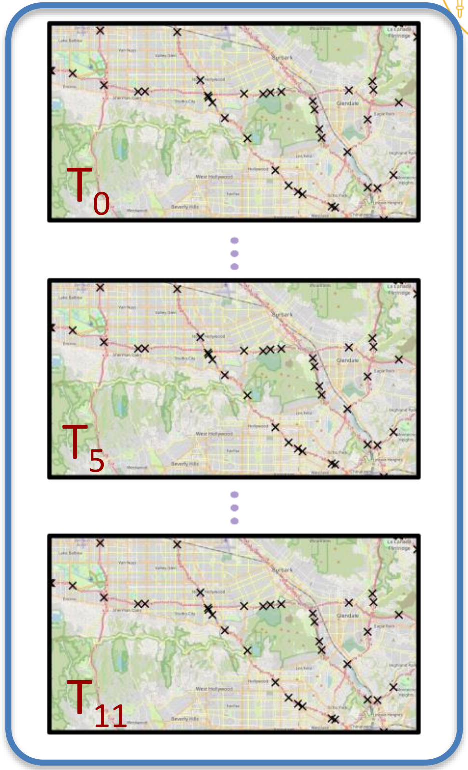


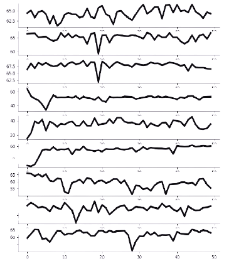


GIVEN: traffic measurements (e.g., avg speed of passing cars) over 12 timesteps of some road segments.

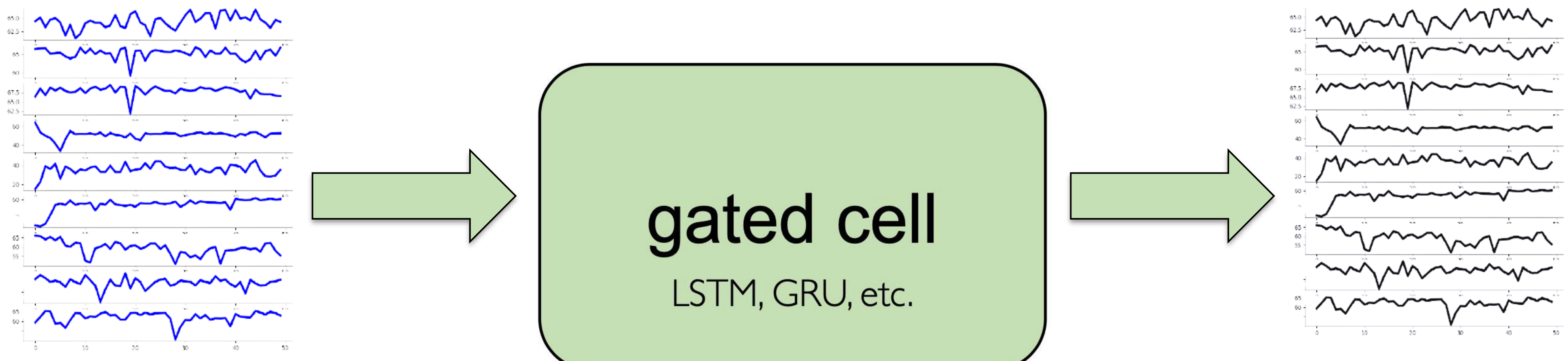
GOAL: Predict traffic measurements for the next 12 timesteps.

Traffic Prediction





RNNs for Traffic Forecasting Task

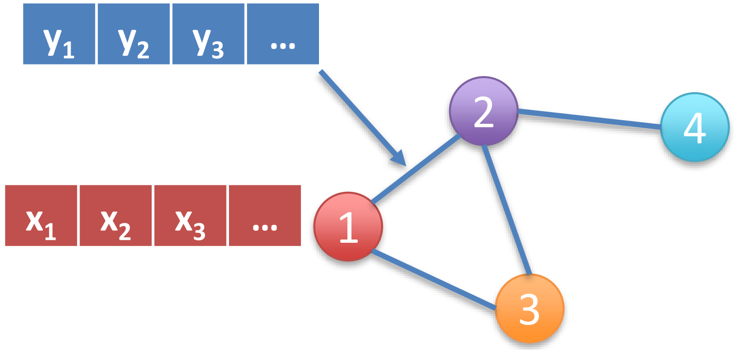


Traffic Observations　　Traffic Predictions

- **Straightforward Approach:** Pass the sequences to an RNN-based model to forecast future values.
- **What's wrong with this approach?**
  - **Missed inductive bias:** This ignores **spatial dependencies**, treating each location **independently** instead of leveraging the **connected road network** structure.

To accurately model spatiotemporal tasks, we need an approach that leverages ***the inductive bias*** *of* ***both spatial and temporal patterns*** *inherent in data, such as in road networks.*

Graphs



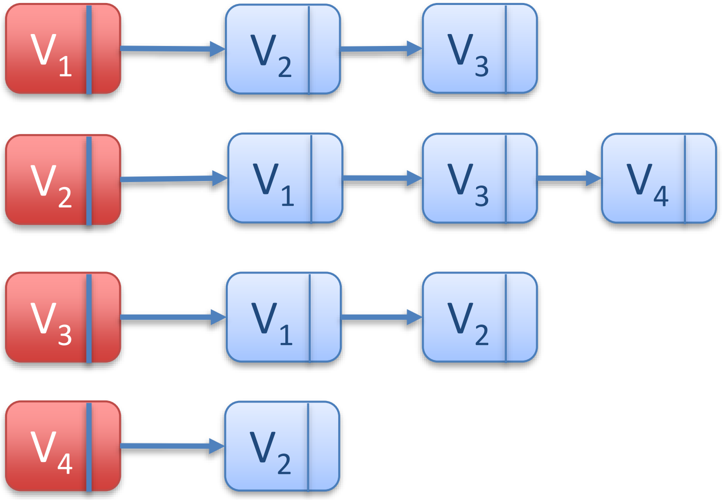
Graph

G = (V,E)

	V_1	V_2	V_3	V_4
V_1	0	1	1	0
V_2	1	0	1	1
V_3	1	1	0	0
V_4	0	1	0	0

Adjacency Matrix

$V \times V$

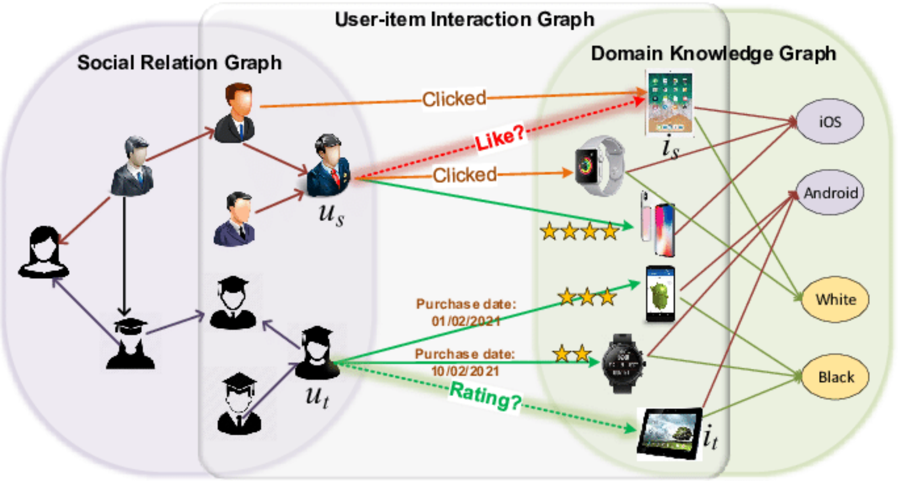


Adjacency List

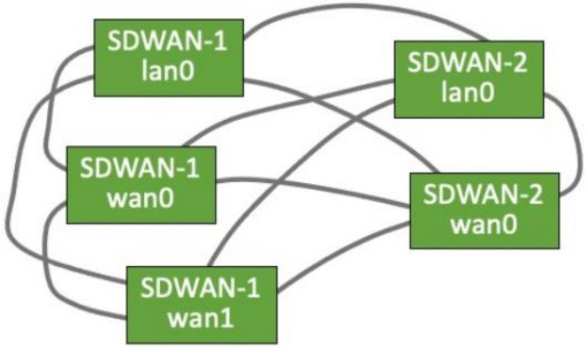
Modeling Real World Problems as Graphs

Road Network
(Veličković, 2021)

Social Networks

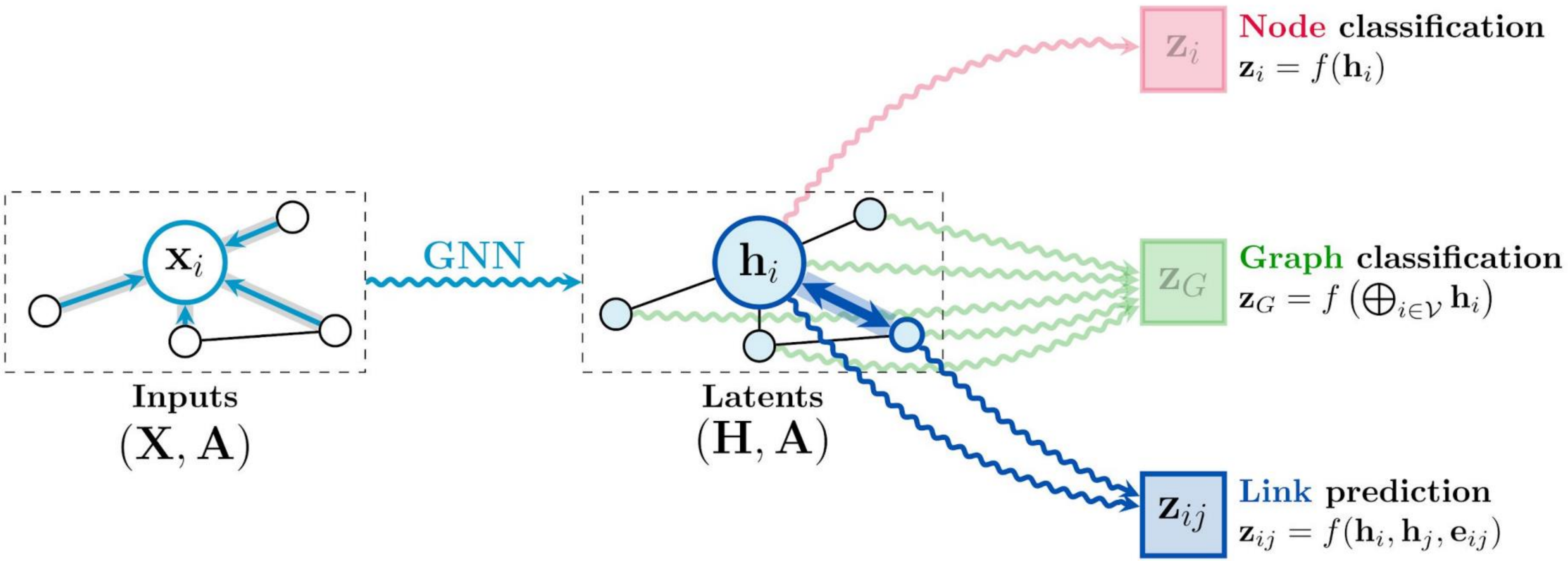


Recommendation Systems
(Wang, 2021)



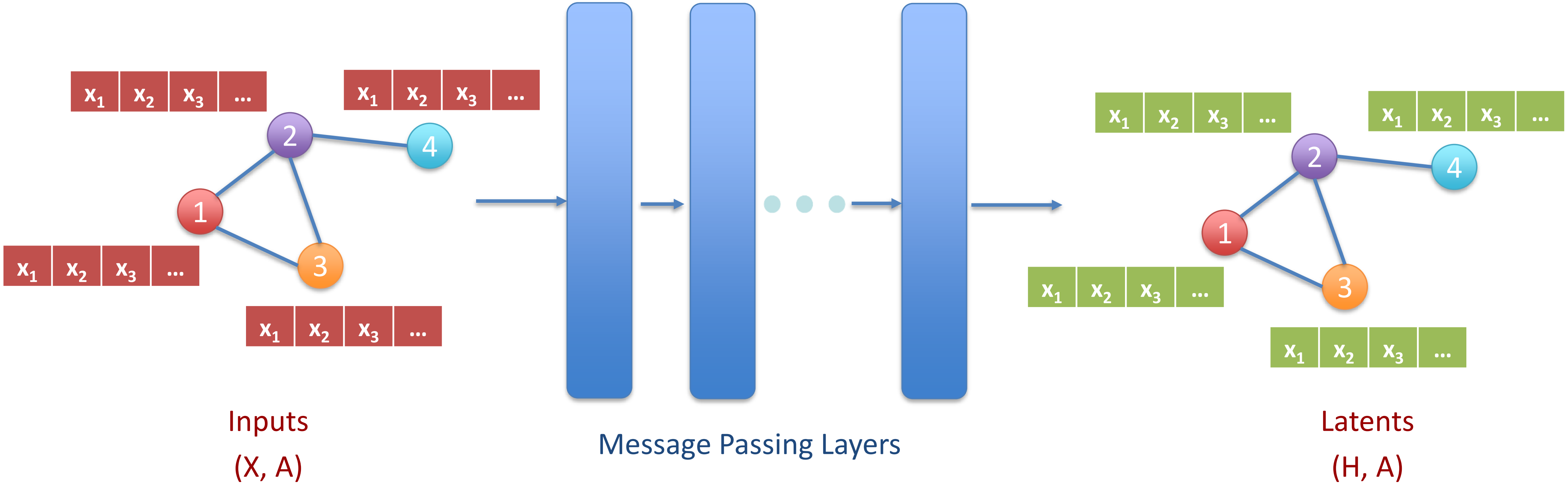
Computer Network Topology
(Lin, 2021)

Graph Neural Networks



Using GNNs to Solve Machine Learning Problems
(Veličković, 2021)

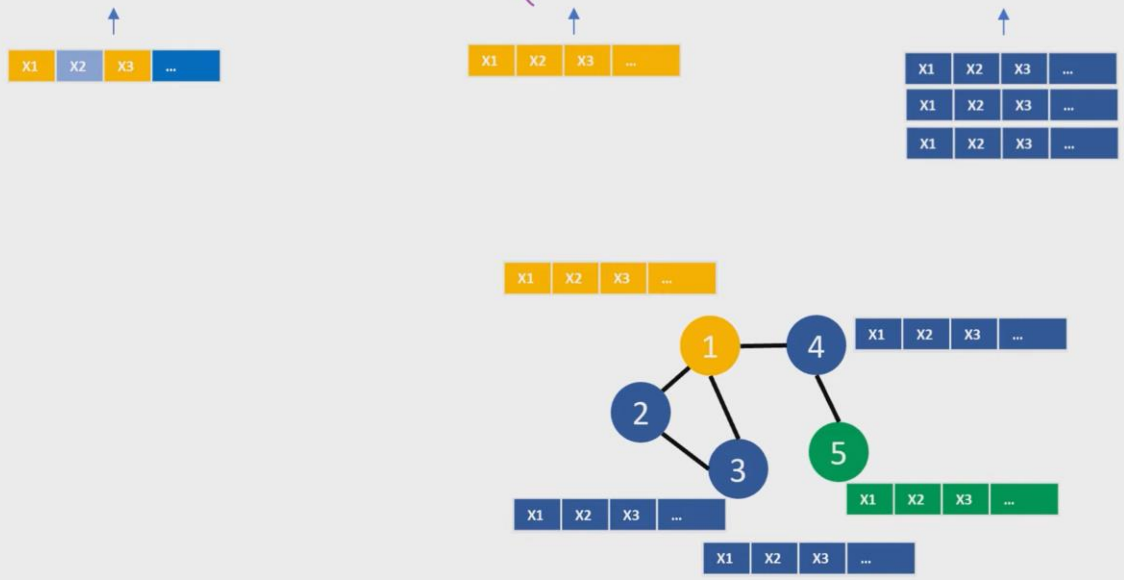
Graph Neural Networks (cont'd)



Message Passing in GNNs

- Message Passing updates and aggregations:

$$h_u^{(k+1)} = UPDATE^{(k)}\left(h_u^{(k)}, AGGREGATE^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$



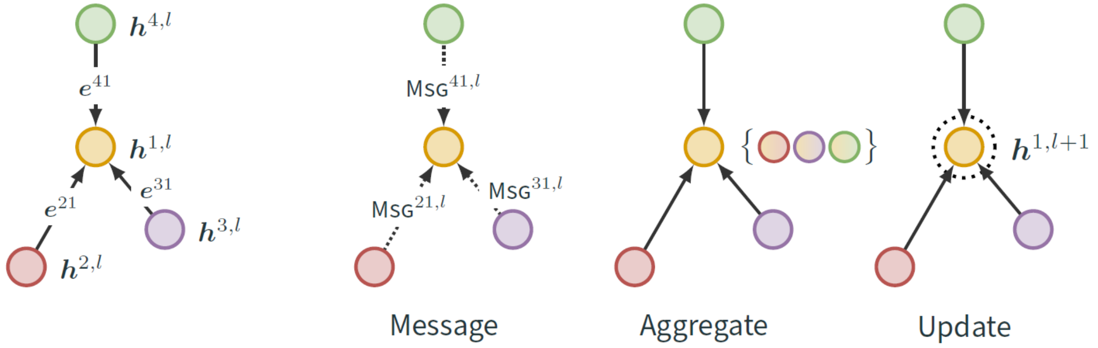
Message Passing in GNNs at a Glance

- Number of message-passing layers is a hyper-parameter
  - Too many layers → over-smoothing

Message Passing in GNNs

To process the spatial dimension, we rely on the **message-passing (MP)** framework

$$\boldsymbol{h}^{i,l+1} = \textsc{Up}^{l}\left(\boldsymbol{h}^{i,l}, \underset{j\in\mathcal{N}(i)}{\textsc{Aggr}}\left\{\textsc{Msg}^{l}\left(\boldsymbol{h}^{i,l}, \boldsymbol{h}^{j,l}, \boldsymbol{e}^{ji}\right)\right\}\right),$$



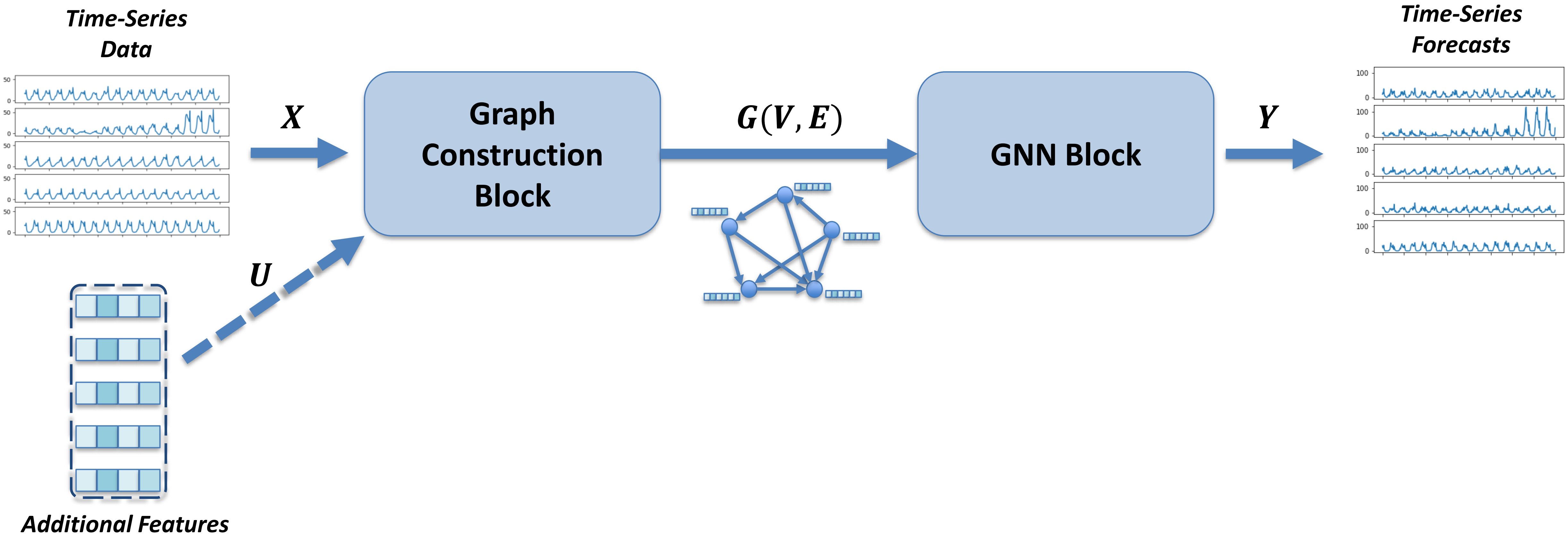
Where:

- $\textsc{Msg}^{l}(\,\cdot\,)$ is the **message function**, e.g., implemented by an MLP.
- $\textsc{Aggr}\{\,\cdot\,\}$ is the permutation invariant **aggregation function**.
- $\textsc{Up}^{l}(\,\cdot\,)$ is the **update function**, e.g., implemented by an MLP.

Aggregation is performed over $\mathcal{N}(i)$, i.e., the set of neighbors of node i.

GNN-based Time-Series Forecasting

General Framework



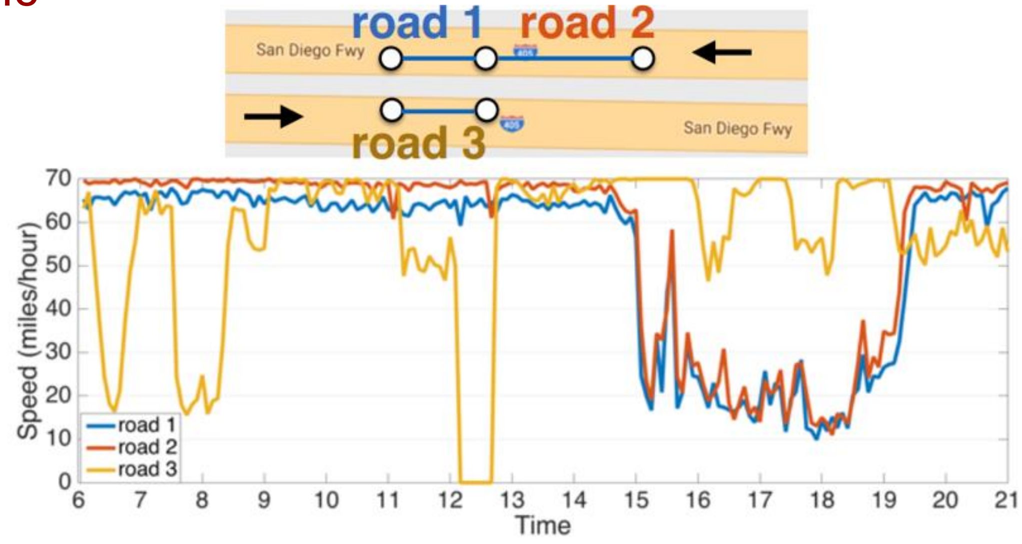
DCRNN for Traffic Forecasting

DCRNN for Traffic Forecasting

- **Forecasting Problem:**

$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

What are some interesting observations from the traffic observations in this figure?



DCRNN for Traffic Forecasting

- **Forecasting Problem:**

$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

What are some interesting observations from the traffic observations in this figure?

1. **Complex spatial dependencies:** Traffic patterns show **non-Euclidean** dependencies

DCRNN for Traffic Forecasting

- **Forecasting Problem:**

$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

What are some interesting observations from the traffic observations in this figure?

2. **Non-linear temporal dynamics:** Rush hours and traffic incidents causes **non-stationarity**

DCRNN for Traffic Forecasting

- **Forecasting Problem:**

$$[\boldsymbol{X}^{(t-T'+1)}, \cdots, \boldsymbol{X}^{(t)}; \mathcal{G}] \xrightarrow{h(\cdot)} [\boldsymbol{X}^{(t+1)}, \cdots, \boldsymbol{X}^{(t+T)}]$$

What are some interesting observations from the traffic observations in this figure?

3. **Difficulty of long-term forecasting:** Traffic measurements **fluctuate** heavily during a long window

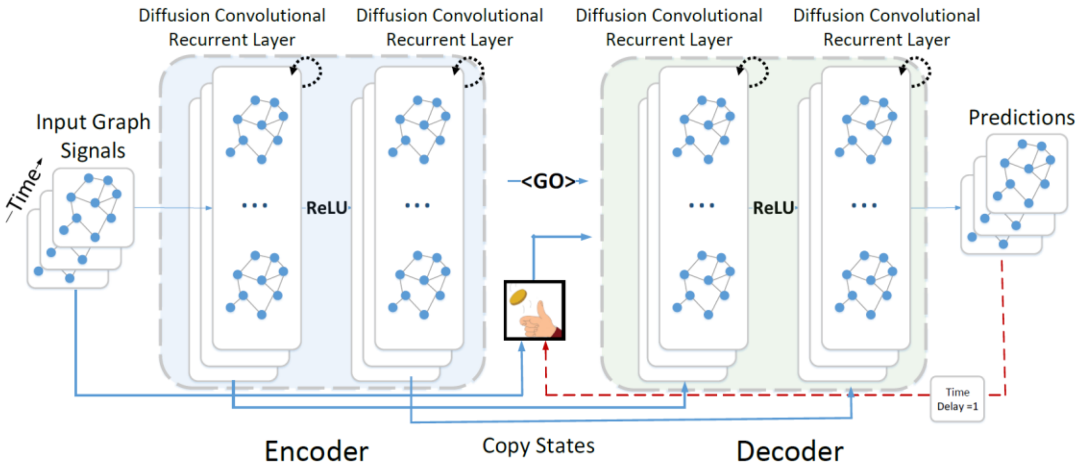
DCRNN for Traffic Forecasting – Overall Framework

Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting

Goal: Vehicle traffic forecasting

Graph Construction Block: Building a **static graph** of traffic sensors based on their road-network distances

GNN Block: Applying **Graph Diffusion Convolution** with a **seq-to-seq** architecture on the previous graph utilizing the following techniques:



DCRNN Overall Architecture

Diffusional Convolution: To capture complex spatial dependencies

Recurrent Neural Networks: To capture non-linear temporal dynamics

Encoder-Decoder Architecture: To capture better long-term dependencies

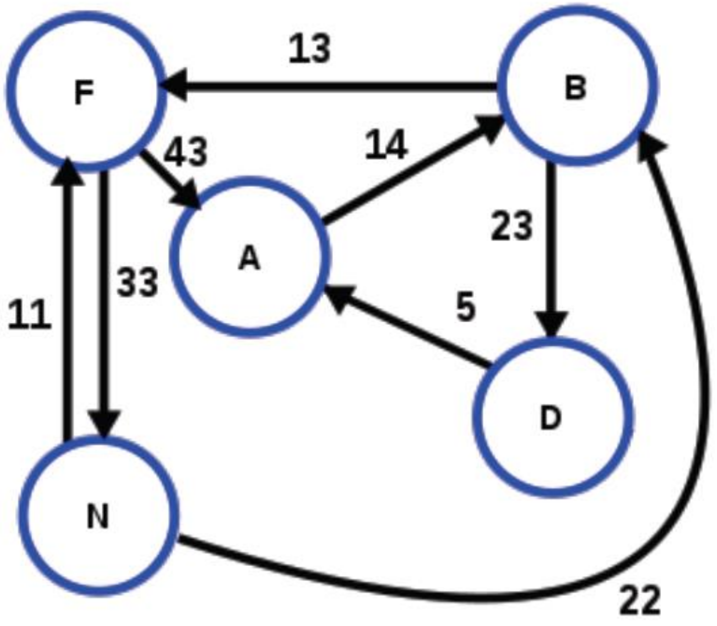
DCRNN – Graph Construction

Graph Construction: Based on the **road-network distance** between traffic sensors

- Transportation network as graph
  - V = Vertices (sensors)
  - E = Edges (roads)
  - A = Weighted adjacency matrix
    (A function of the road network distance)

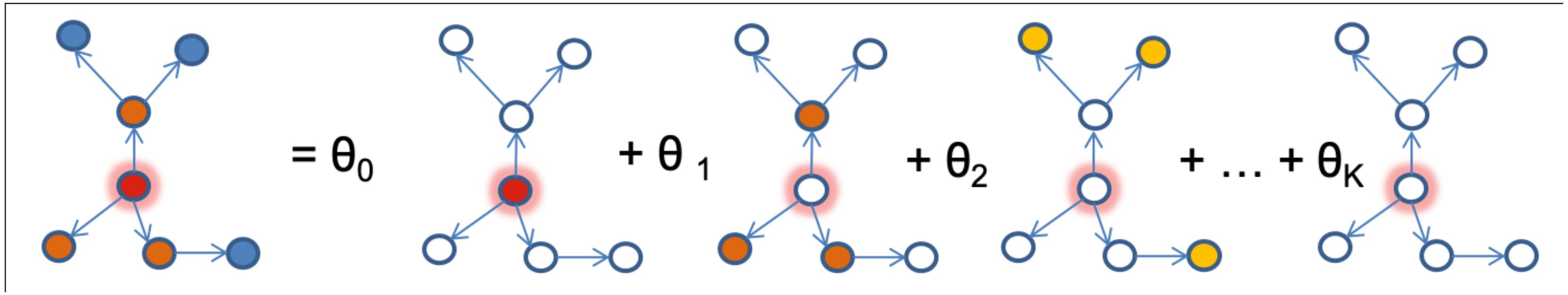
$$A_{ij} = \exp\left(-\frac{\mathrm{dist}_{\mathrm{net}}(v_i, v_j)^2}{\sigma^2}\right) \text{ if } \mathrm{dist}_{\mathrm{net}}(v_i, v_j) \le \kappa$$

$\mathrm{dist}_{\mathrm{net}}(v_i, v_j)$: road network distance from v_i to v_j,
κ: threshold to ensure sparsity, σ^2 variance of all pairwise road network distances



DCRNN – GNN Block

Diffusion Convolution



Out-degree In-degree

$$X_{:,p} \star_{\mathcal{G}} f_\theta = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D_O^{-1} A} \right)^k + \theta_{k,2} \left(\boldsymbol{D_I^{-1} A^{\intercal}} \right)^k \right) \boldsymbol{X}_{:,p}$$

- $\boldsymbol{\theta} \in \mathbb{R}^{K \times 2}$ are the parameters to train
- $\boldsymbol{A}$ is the adjacency matrix (previous page)
- $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ is the input with $\boldsymbol{N}$ as the # of nodes, $\boldsymbol{P}$ as the feature dimension of each node
- $\boldsymbol{D_O}$: Out-degree matrix for outgoing flow / $\boldsymbol{D_I}$: In-degree matrix for incoming flow

DCRNN – GNN Block

Diffusion Convolution

- Take $\boldsymbol{K} = \mathbf{3}$ for example
- Note – Change of notation: $\boldsymbol{W}$ now represents the adjacency matrix (previously denoted as A in earlier slides).

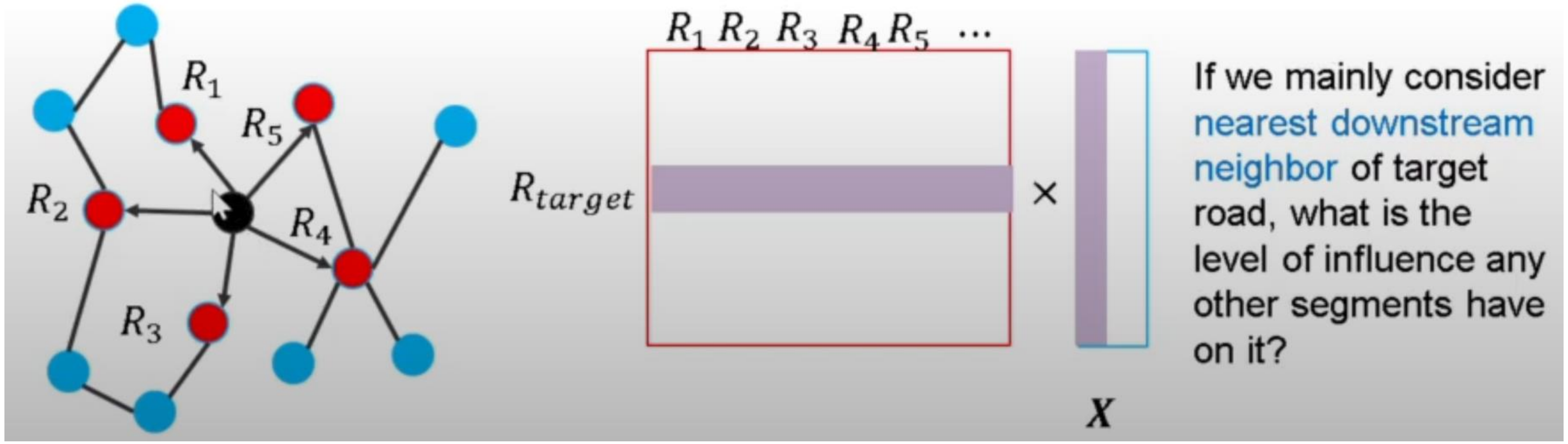
$$\sum_{k=0}^{K-1} \theta_{k,1} \left(D_O^{-1}\boldsymbol{W}\right)^k \boldsymbol{X}_{:,p} = \boxed{\theta_{0,1}\boldsymbol{X}_{:,p}} + \theta_{1,1}\left(D_O^{-1}\boldsymbol{W}\right)^1 \boldsymbol{X}_{:,p} + \theta_{1,2}\left(D_O^{-1}\boldsymbol{W}\right)^2 \boldsymbol{X}_{:,p}$$

DCRNN – GNN Block

Diffusion Convolution

- Take $\boldsymbol{K} = \mathbf{3}$ for example
- Note – Change of notation: $\boldsymbol{W}$ now represents the adjacency matrix (previously denoted as A in earlier slides).

$$\sum_{k=0}^{K-1} \theta_{k,1} \left(D_O^{-1}\boldsymbol{W}\right)^k \boldsymbol{X}_{:,p} = \theta_{0,1}\boldsymbol{X}_{:,p} + \boxed{\theta_{1,1}\left(D_O^{-1}\boldsymbol{W}\right)^1\boldsymbol{X}_{:,p}} + \theta_{1,2}\left(D_O^{-1}\boldsymbol{W}\right)^2\boldsymbol{X}_{:,p}$$

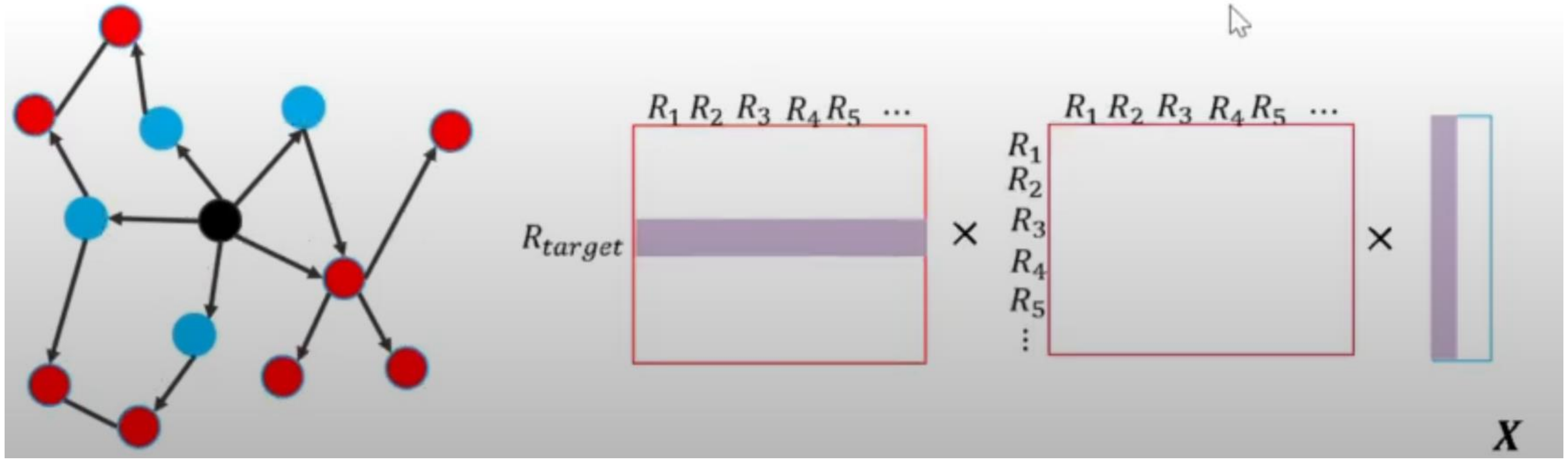


DCRNN – GNN Block

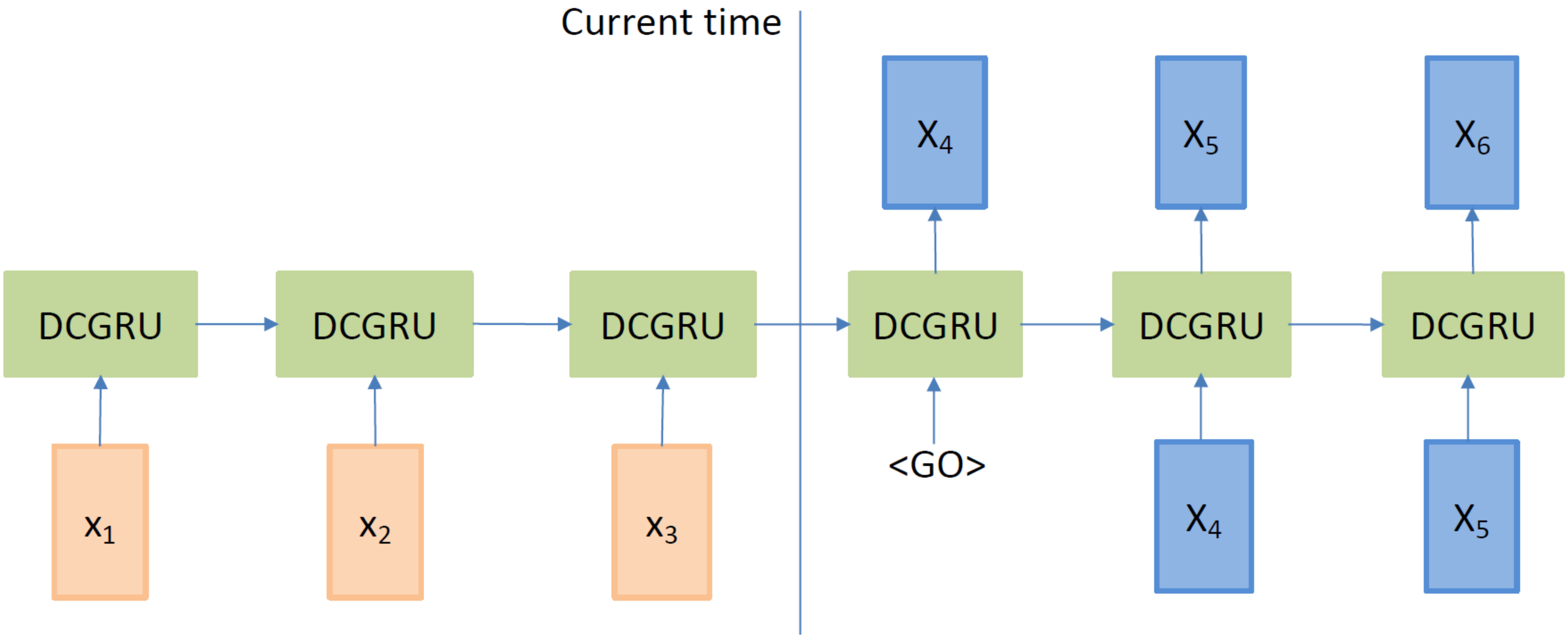
Diffusion Convolution

- Take $\boldsymbol{K} = \mathbf{3}$ for example
- Note – Change of notation: $\boldsymbol{W}$ now represents the adjacency matrix (previously denoted as A in earlier slides).

$$\sum_{k=0}^{K-1} \theta_{k,1} (D_O^{-1}\boldsymbol{W})^k \boldsymbol{X}_{:,p} = \theta_{0,1}\boldsymbol{X}_{:,p} + \theta_{1,1}(D_O^{-1}\boldsymbol{W})^1\boldsymbol{X}_{:,p} + \boxed{\theta_{1,2}(D_O^{-1}\boldsymbol{W})^2\boldsymbol{X}_{:,p}}$$



Encoder-Decoder Framework of DCRNN



Encoder-Decoder Framework of DCRNN

- In DCRNN, the standard **matrix multiplication** in **GRU** is replaced with a **diffusion convolution** operation to capture spatial dependencies in the graph.

GRU Cell

$$u_t = \sigma(w_u[h_{t-1}, x_t])$$

$$r_t = \sigma(w_r[h_{t-1}, x_t])$$

$$c_t = tanh(w_c[r_t * h_{t-1}, x_t])$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * c_t$$

DCRNN Cell

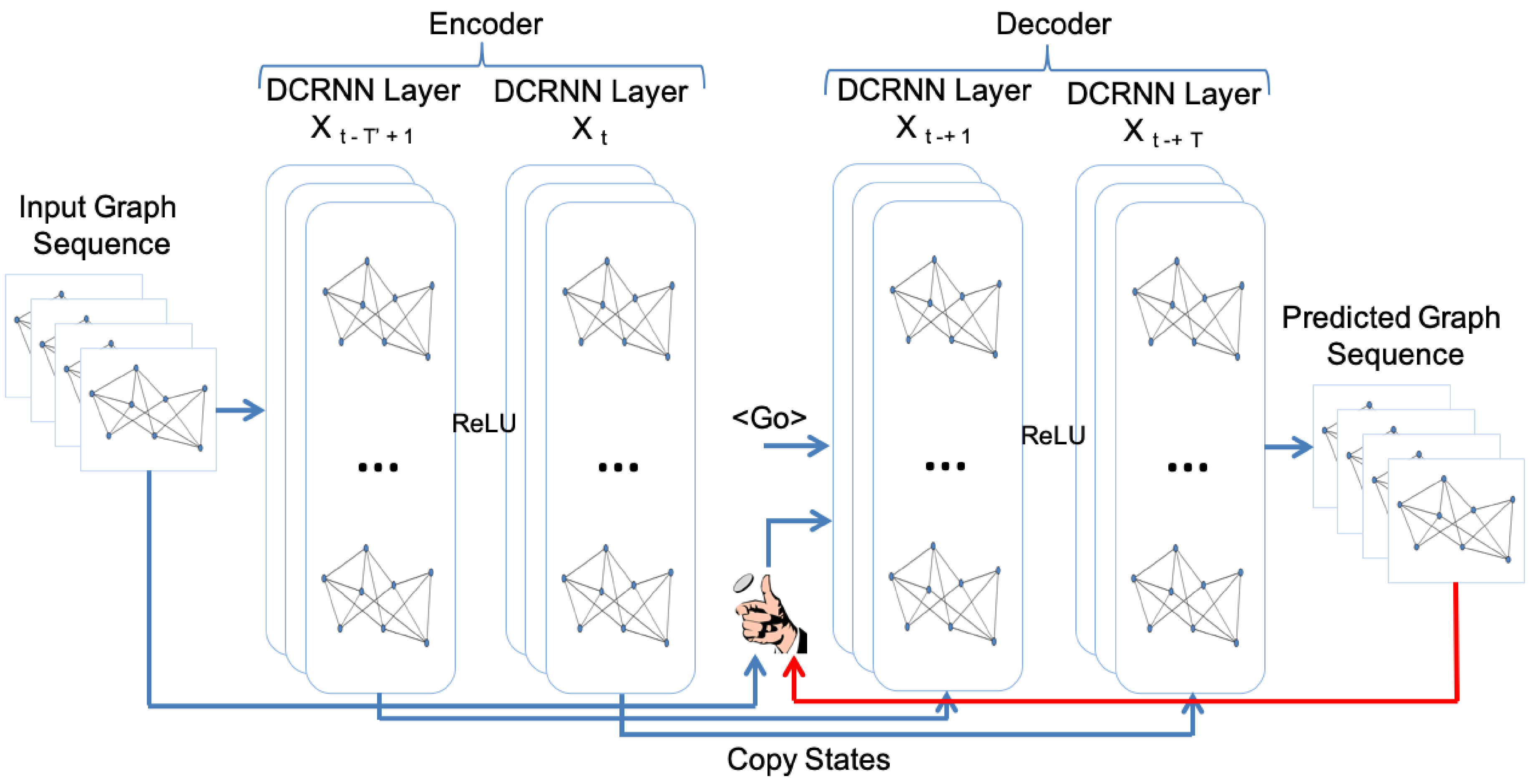
$$u_t = \sigma(w_{u*G}[h_{t-1}, x_t])$$

$$r_t = \sigma(w_{r*G}[h_{t-1}, x_t])$$

$$c_t = tanh(w_{c*G}[r_t * h_{t-1}, x_t])$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * c_t$$

DCRNN End-to-End Framework



BysGNN for Traffic Forecasting

Busyness Graph Neural Network – Motivation

- **Limitations of Static Models (e.g., DCRNN): Fixed graphs** miss the **changing nature** of traffic and **context**-based correlations (e.g., road type).
- **Need for Dynamic Adaptation:** Effective forecasting requires capturing both **static** and **dynamic** relationships between sensors.
- **BysGNN's Goal:** Create a model that **dynamically** learns these relationships to improve traffic prediction accuracy.

BysGNN – Definitions

Definition 1: Multi-Context Correlations

Latent relationships among traffic sensors that are influenced by various contextual factors, including:

- **Spatial correlations:** closeness of **geographical** sensor locations.
- **Temporal dependencies:** the changes in visit patterns of individual sensor observations over time **(intra-series)** and the dependencies between traffic patterns of different sensors **(inter-series).**
- **Semantic similarities:** Similarity of sensor **attributes**, such as their road types (e.g., highway/arterial).
- **Taxonomic correlations:** Similarities in traffic patterns of different **groups of sensors**, e.g., similarities between **high-level** traffic observed in two different neighborhoods.

BysGNN – Definitions

Definition 2: Busyness Graph

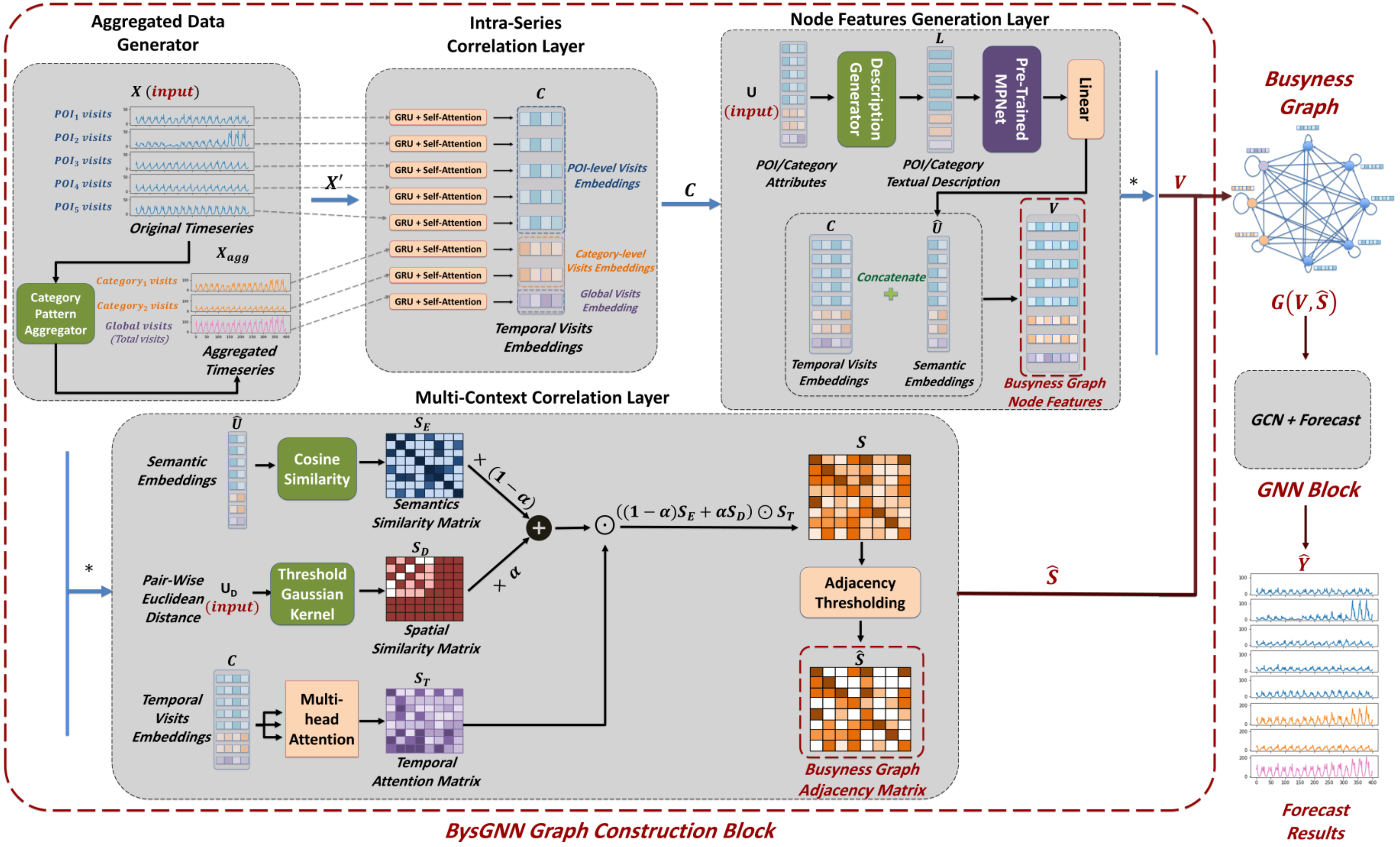
- A graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{A})$ where $\boldsymbol{V}$ is a set of $|V| = N$ **nodes**, and each node corresponds to **a specific traffic sensor** or **a group of traffic sensors** (e.g., all sensors in West Hollywood).
- We denote $A \in \mathbb{R}^{N \times N}$ as the **adjacency matrix** in which a_{ij} indicates the amount of **influence** that node v_i has on the forecasts of v_j.
- Adjacency matrix A is **dynamically** updated based on the **Multi-Context Correlations** and captures the most **recent knowledge** about **interaction** between traffic sensors.

BysGNN – Traffic Forecasting Problem

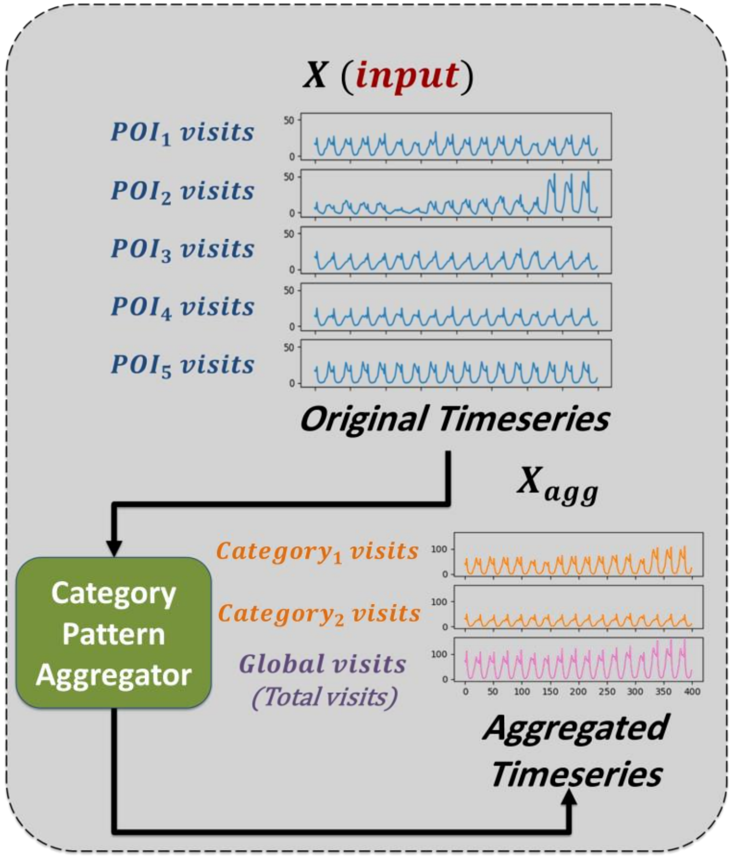
Traffic Forecasting Problem:

Given $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}) \in \mathbb{R}^{\boldsymbol{N \times T}}$ as the sequence of traffic measurements for the past $\boldsymbol{T}$ hours to $\boldsymbol{N}$ sensors, and $\boldsymbol{U} = (\boldsymbol{u_1}, \ldots, \boldsymbol{u_N}) \in \mathbb{R}^{\boldsymbol{N \times J}}$ as the set of $\boldsymbol{J}$ attributes (e.g., road type, number of lanes, etc.) of each traffic sensor, generate Busyness Graph G to find $Y = (y_1, \ldots, y_N) \in \mathbb{R}^{N \times H}$, the future traffic measurements for the next H time steps for each sensor.

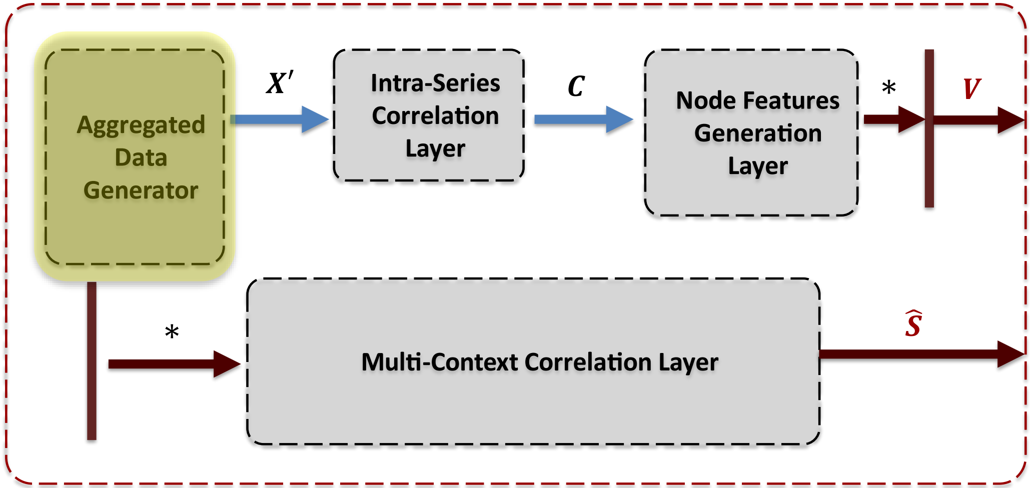
Busyness Graph Neural Network (BysGNN) Framework



BysGNN Framework – Aggregated Data Generator

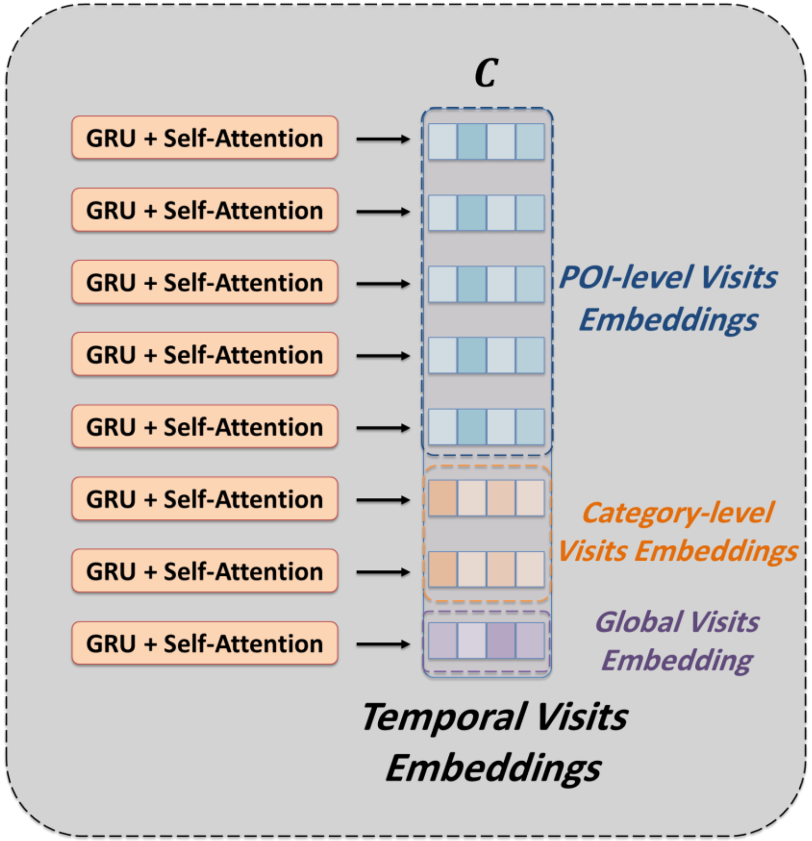


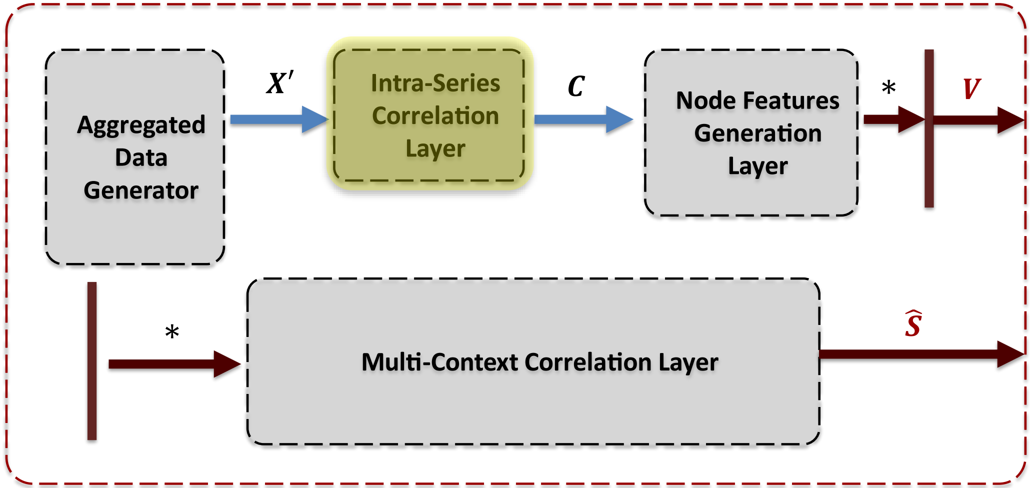
To adapt this framework for traffic forecasting, aggregated series are generated by **aggregating sensors from each geographical neighborhood**



Graph Construction Block

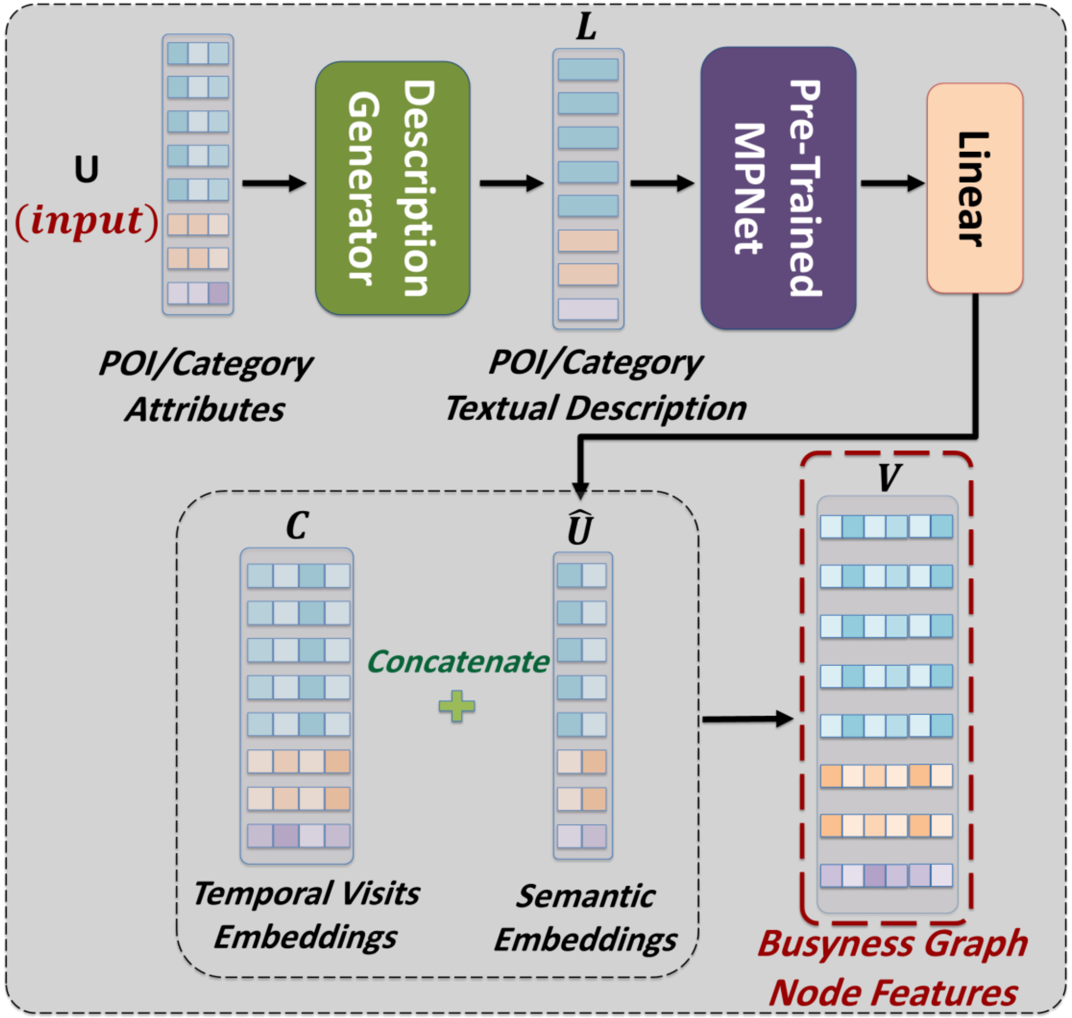
BysGNN Framework – Intra-Series Correlation Layer

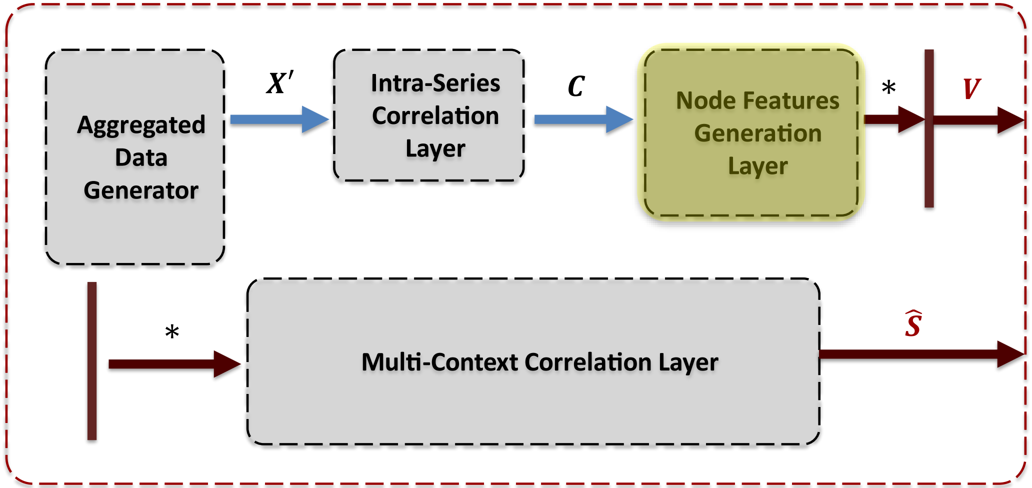




Graph Construction Block

BysGNN Framework – Node Features Generation Layer



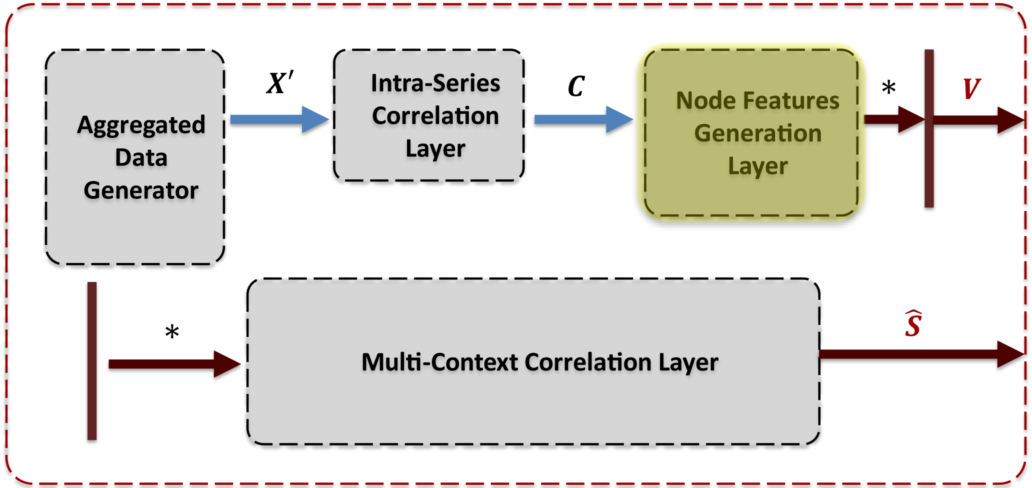


Graph Construction Block

BysGNN Framework – Node Features Generation Layer

Sample Generated Textual Description for a traffic sensor:

This arterial road on **Main St., Los Angeles, CA**, has **3 lanes** with a **max speed limit of 45 mph**. Peak traffic occurs **weekdays from 7:00 - 9:00 AM and 4:00 - 6:00 PM**. It primarily serves **commuter traffic**, classified under **Urban Roads** and **High-Traffic Zones**.

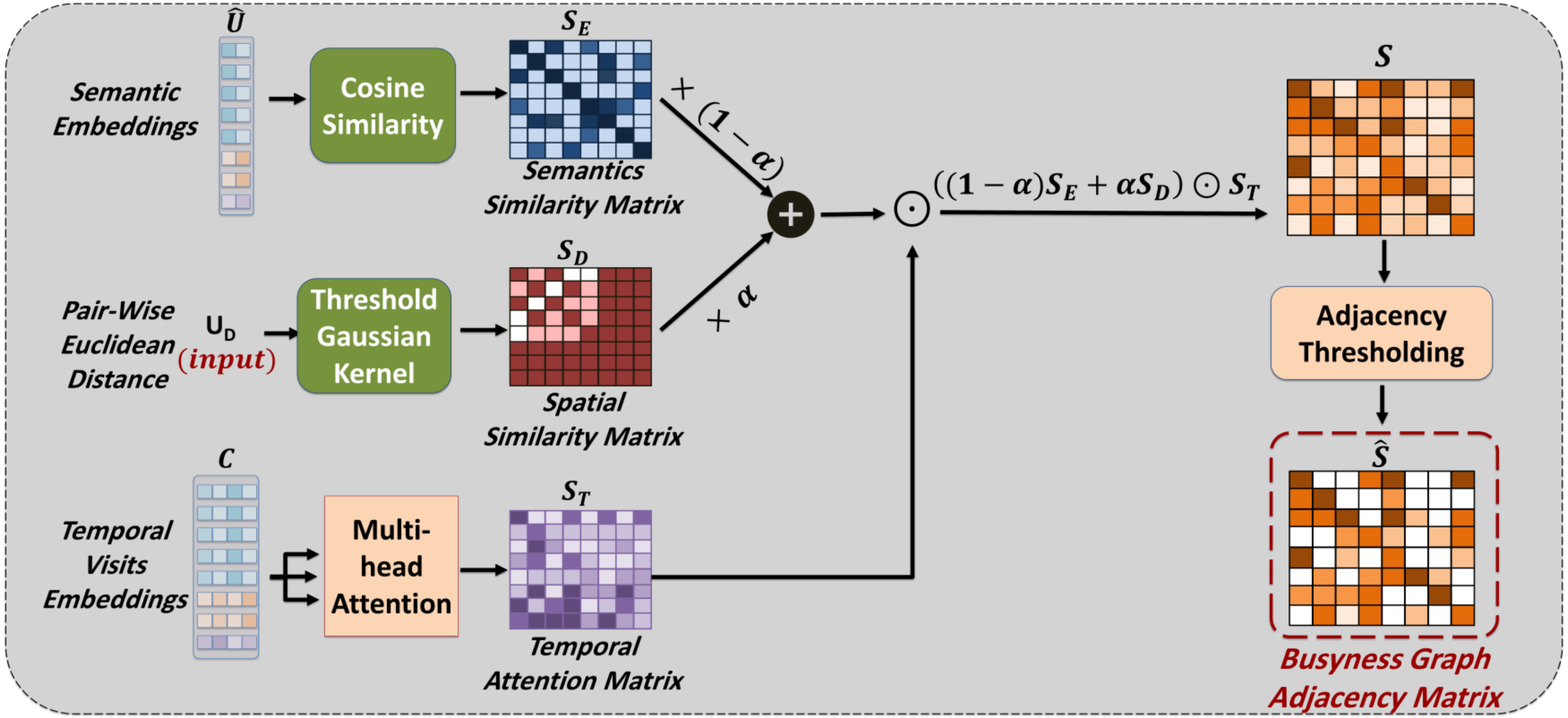


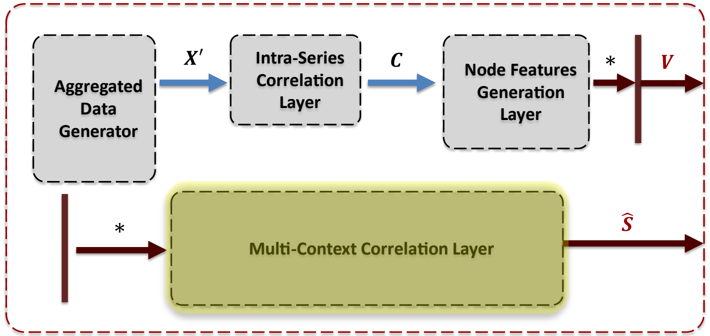
Graph Construction Block

Sample Generated Textual Description for a Category:

This neighborhood is a **commercial district**, experiencing high traffic volumes due to **commuter and shopping** traffic. Primary access routes include **Broadway and Main St**.

BysGNN Framework – Multi-Context Correlation Layer



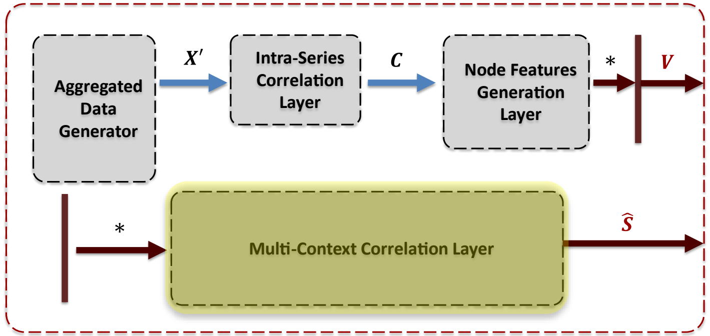


Graph Construction Block

BysGNN Framework – Multi-Context Correlation Layer (cont'd)

Gating Mechanism:

Helps to preserve strong long-term relationships and penalize noisy relationships between distant or semantically dissimilar nodes



Graph Construction Block

Adjacency Thresholding:

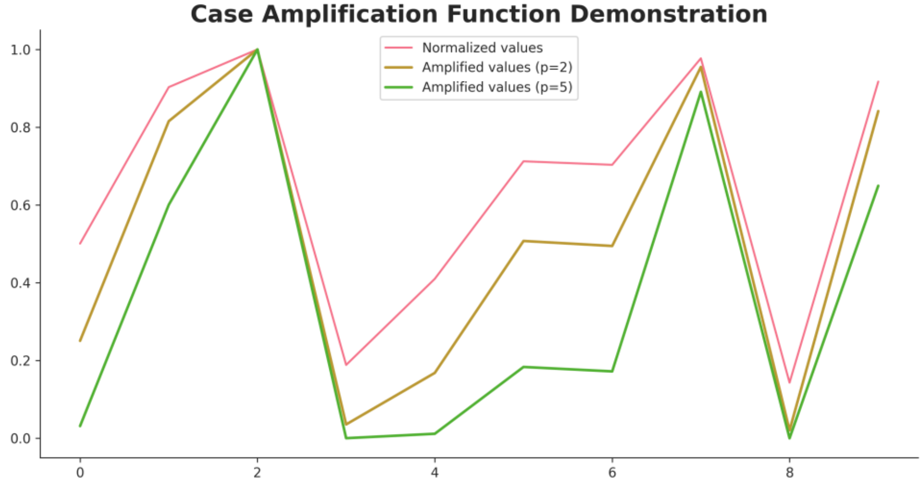
Filters out the previous noisy relationships.

BysGNN uses a case amplification function to differentiate between small and large values in adjacency matrix. This reduces the impact of of small values more significantly than the larger values. This is done as follows:

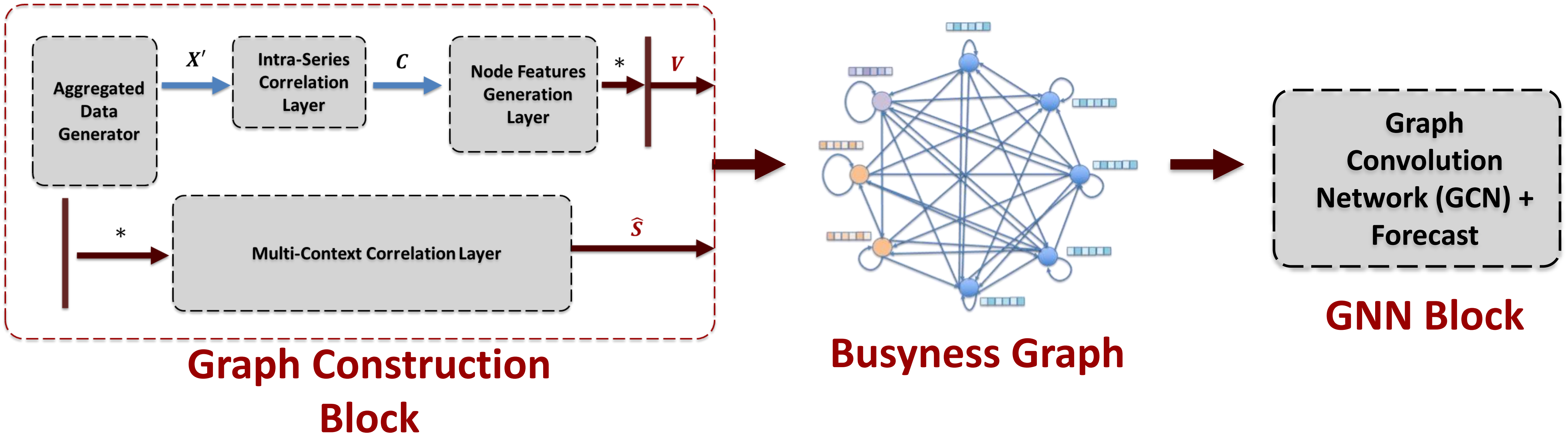
S: Original Adjacency Matrix

$\hat{S}$: Thresholded Adjacency Matrix

$$\hat{S}ij = \begin{cases} Sij, & \text{if } \left(\frac{S_{ij}}{\max(S_i)}\right)^p > \eta \\ 0, & \text{otherwise} \end{cases}$$



BysGNN – Busyness Graph and GNN Block



Traffic Forecasting Evaluation

Task: Forecast the # of visits to each POI for the **next 6 hours** from the **past 24 hours** of visitation data

Evaluation Metrics

MAE: Average of the difference between the ground truth and the predicted values

$$MAE = \frac{1}{N}\sum_{j=1}^{N} |y_j - \hat{y}_j|$$

MAPE: the percentage equivalent of MAE

$$MAPE = \frac{100\%}{N}\sum_{j=1}^{N} \left|\frac{y_j - \hat{y}_j}{y_j}\right|$$

RMSE: Square root of the average of the squared difference between the target value and the value predicted

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N} (y_j - \hat{y}_j)^2}$$

Experiments Results – Large Data Regime

Evaluation

- BySGNN significantly outperforms DCRNN
- …demonstrating that **dynamically capturing relationships** yields **better** results than relying on a **fixed**, **predefined** structure.

Forecasting Results for High # of POIs Data Regime

Dataset	Houston			Los Angeles			New York		
Evaluation Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
Naïve Seasonal	4.746	*0.664*	18.166	2.934	0.752	10.005	3.681	0.699	9.216
Historical Average	8.860	0.783	26.911	4.388	0.786	11.729	6.555	0.770	22.018
ConvGRU	6.415	2.539	20.179	3.781	3.139	10.905	4.605	1.851	17.028
ConvLSTM	8.076	4.270	23.127	4.362	4.397	11.879	5.448	2.697	18.959
DCRNN	5.683	1.990	18.941	3.389	2.693	9.879	4.139	1.605	15.504
A3T-GCN	8.380	3.377	23.9	4.604	4.129	12.601	5.824	2.579	20.171
StemGNN	*4.390*	0.735	*14.604*	*2.485*	**0.671**	*6.951*	*3.261*	*0.652*	*8.074*
BysGNN	**4.095**	**0.658**	**12.904**	**2.377**	*0.676*	**6.091**	**3.113**	**0.598**	**7.351**
RelError	-6.71%	-0.90%	-11.64%	-4.34%	+0.74%	-12.37%	-4.53%	-8.28%	-8.95%

Dataset	Chicago			San Antonio		
Evaluation Metric	MAE	MAPE	RMSE	MAE	MAPE	RMSE
Naïve Seasonal	3.237	0.754	*9.216*	3.85	0.689	10.573
Historical Average	4.624	0.791	14.011	6.494	0.78	15.776
ConvGRU	4.18	2.675	10.753	4.622	2.838	10.218
ConvLSTM	5.019	3.853	11.972	5.745	5.202	11.608
DCRNN	3.756	2.327	9.857	4.167	2.269	9.513
A3TGCN	5.343	3.654	12.821	6.086	4.517	12.731
StemGNN	*2.776*	*0.724*	9.857	*3.371*	*0.686*	*8.923*
BysGNN	**2.750**	**0.718**	**8.218**	**3.278**	**0.629**	**8.418**
RelError	-0.93%	-0.82%	-10.82%	-2.75%	-8.30%	-5.65%

References

[1] *Chiang, Yao-Yi Introduction to Spatial Artificial Intelligence. Available from https://yaoyichi.github.io/spatial-ai.html*.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” *ICLR*, 2017.

[3] Hajisafi, A., Lin, H., Shaham, S., Hu, H., Siampou, M. D., Chiang, Y. Y., & Shahabi, C. (2023, November). Learning dynamic graphs from all contextual information for accurate point-of-interest visit forecasting. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems* (pp. 1-12).

[4] Alippi, C., Zambon, D., Cini, A., & Marisca, I. (2023, September 22). *Graph Deep Learning for Spatiotemporal Time Series Forecasting, Reconstruction, and Analysis*. Presented at ECML/PKDD, Turin. Graph Machine Learning Group (gmlg.ch), The Swiss AI Lab IDSIA, Università della Svizzera italiana.

[5] Mallick, T., Balaprakash, P., Rask, E., & Macfarlane, J. (2020). Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting. *Transportation Research Record*, *2674*(9), 473-488.