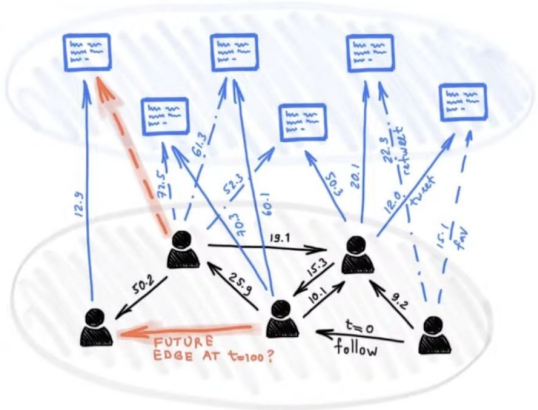


Temporal Graph Networks

But I still haven't found what I'm looking for …

-- Bono, U2

Jay Urbain, PhD - 2/9/2023

References

Temporal Graph Networks:
https://arxiv.org/pdf/2006.10637.pdf

Pytorch Geometric Temporal:
https://arxiv.org/pdf/2104.07788.pdf

Diffusion Convolutional Recurrent Neural Network:
https://arxiv.org/pdf/1707.01926.pdf

MST-GNN: https://arxiv.org/pdf/2108.11244.pdf

Friendly Introduction to Temporal Graph Neural Networks (and some Traffic Forecasting)

https://www.youtube.com/watch?v=WEWq93tioC4&t=14s

Temporal Graph Networks:
https://arxiv.org/pdf/2006.10637.pdf (used for the intro)

Pytorch Geometric Temporal:

TEMPORAL GRAPH NETWORKS FOR DEEP LEARNING ON DYNAMIC GRAPHS

Emanuele Rossi* Twitter
Ben Chamberlain Twitter
Fabrizio Frasca Twitter
Davide Eynard Twitter
Federico Monti Twitter
Michael Bronstein Twitter

ABSTRACT

Graph Neural Networks (GNNs) have recently become increasingly popular due to their ability to learn complex systems of relations or interactions. These arise in a broad spectrum of problems ranging from biology and particle physics to social networks and recommendation systems. Despite the plethora of different models for deep learning on graphs, few approaches have been proposed for dealing with graphs that are dynamic in nature (e.g. evolving features or connectivity over time). We present Temporal Graph Networks (TGNs), a generic, efficient framework for deep learning on dynamic graphs represented as sequences of timed events. Thanks to a novel combination of memory modules and graph-based operators, TGNs significantly outperform previous approaches while being more computationally efficient. We furthermore show that several previous models for learning on dynamic graphs can be cast as specific instances of our framework. We perform a detailed ablation study of different components of our framework and devise the best configuration that achieves state-of-the-art performance on several transductive and inductive prediction tasks for dynamic graphs.

1 INTRODUCTION

In the past few years, graph representation learning (Bronstein et al., 2017; Hamilton et al., 2017b; Battaglia et al., 2018) has produced a sequence of successes, gaining increasing popularity in machine learning. Graphs are ubiquitously used as models for systems of relations and interactions in many fields (Battaglia et al., 2016; Qi et al., 2018; Monti et al., 2016; Choma et al., 2018; Duvenaud et al., 2015; Gilmer et al., 2017; Parisot et al., 2018; Rossi et al., 2019), in particular, social sciences (Ying et al., 2018; Monti et al., 2019; Rossi et al., 2020) and biology (Zitnik et al., 2018; Veselkov et al., 2019; Gainza et al., 2019). Learning on such data is possible using graph neural networks (GNNs) (Hamilton et al., 2017a) that typically operate by a message passing mechanism (Battaglia et al., 2018) aggregating information in a neighborhood of a node and create node embeddings that are then used for node classification (Monti et al., 2016; Velickovic et al., 2018; Kipf & Welling, 2017), graph classification (Gilmer et al., 2017), or edge prediction (Zhang & Chen, 2018) tasks.

The majority of methods for deep learning on graphs assume that the underlying graph is *static*. However, most real-life systems of interactions such as social networks or biological interactomes are *dynamic*. While it is often possible to apply static graph deep learning models (Liben-Nowell & Kleinberg, 2007) to dynamic graphs by ignoring the temporal evolution, this has been shown to be sub-optimal (Xu et al., 2020), and in some cases, it is the dynamic structure that contains crucial insights about the system. Learning on dynamic graphs is relatively recent, and most works are limited to the setting of discrete-time dynamic graphs represented as a sequence of snapshots of the graph (Liben-Nowell & Kleinberg, 2007; Dunlavy et al., 2011; Yu et al., 2019; Sankar et al., 2020; Pareja et al., 2019; Yu et al., 2018). Such approaches are unsuitable for interesting real world settings such as social networks, where dynamic graphs are continuous (i.e. edges can appear at any time)

*erossi@twitter.com

References

Temporal Graph Networks:
https://arxiv.org/pdf/2006.10637.pdf

Pytorch Geometric Temporal:
https://arxiv.org/pdf/2104.07788.pdf

Diffusion Convolutional Recurrent Neural Network:
https://arxiv.org/pdf/1707.01926.pdf

MST-GNN: https://arxiv.org/pdf/2108.11244.pdf

The road vector graphics are from:
https://de.vecteezy.com/vektorkunst/1…

Friendly Introduction to Temporal Graph Neural Networks (and some Traffic Forecasting)

https://www.youtube.com/watch?v=WEWq93tioC4&t=14s

TEMPORAL GRAPH NETWORKS FOR DEEP LEARNING ON DYNAMIC GRAPHS

Emanuele Rossi* Twitter
Ben Chamberlain Twitter
Fabrizio Frasca Twitter
Davide Eynard Twitter
Federico Monti Twitter
Michael Bronstein Twitter

ABSTRACT

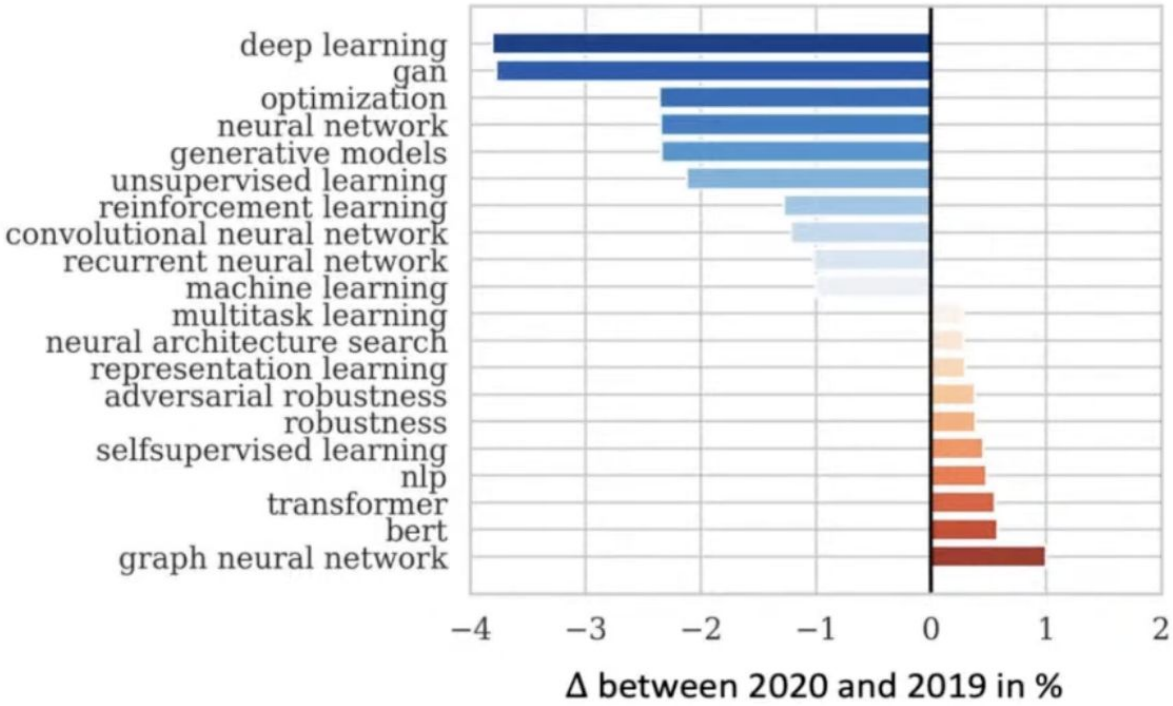
Graph Neural Networks (GNNs) have recently become increasingly popular due to their ability to learn complex systems of relations or interactions. These arise in a broad spectrum of problems ranging from biology and particle physics to social networks and recommendation systems. Despite the plethora of different models for deep learning on graphs, few approaches have been proposed for dealing with graphs that are dynamic in nature (e.g. evolving features or connectivity over time). We present Temporal Graph Networks (TGNs), a generic, efficient framework for deep learning on dynamic graphs represented as sequences of timed events. Thanks to a novel combination of memory modules and graph-based operators, TGNs significantly outperform previous approaches while being more computationally efficient. We furthermore show that several previous models for learning on dynamic graphs can be cast as specific instances of our framework. We perform a detailed ablation study of different components of our framework and devise the best configuration that achieves state-of-the-art performance on several transductive and inductive prediction tasks for dynamic graphs.

1 INTRODUCTION

In the past few years, graph representation learning (Bronstein et al., 2017; Hamilton et al., 2017b; Battaglia et al., 2018) has produced a sequence of successes, gaining increasing popularity in machine learning. Graphs are ubiquitously used as models for systems of relations and interactions in many fields (Battaglia et al., 2016; Qi et al., 2018; Monti et al., 2016; Choma et al., 2018; Duvenaud et al., 2015; Gilmer et al., 2017; Parisot et al., 2018; Rossi et al., 2019), in particular, social sciences (Ying et al., 2018; Monti et al., 2019; Rossi et al., 2020) and biology (Zitnik et al., 2018; Veselkov et al., 2019; Gainza et al., 2019). Learning on such data is possible using graph neural networks (GNNs) (Hamilton et al., 2017a) that typically operate by a message passing mechanism (Battaglia et al., 2018) aggregating information in a neighborhood of a node and create node embeddings that are then used for node classification (Monti et al., 2016; Velickovic et al., 2018; Kipf & Welling, 2017), graph classification (Gilmer et al., 2017), or edge prediction (Zhang & Chen, 2018) tasks.

The majority of methods for deep learning on graphs assume that the underlying graph is *static*. However, most real-life systems of interactions such as social networks or biological interactomes are *dynamic*. While it is often possible to apply static graph deep learning models (Liben-Nowell & Kleinberg, 2007) to dynamic graphs by ignoring the temporal evolution, this has been shown to be sub-optimal (Xu et al., 2020), and in some cases, it is the dynamic structure that contains crucial insights about the system. Learning on dynamic graphs is relatively recent, and most works are limited to the setting of discrete-time dynamic graphs represented as a sequence of snapshots of the graph (Liben-Nowell & Kleinberg, 2007; Dunlavy et al., 2011; Yu et al., 2019; Sankar et al., 2020; Pareja et al., 2019; Yu et al., 2018). Such approaches are unsuitable for interesting real world settings such as social networks, where dynamic graphs are continuous (i.e. edges can appear at any time)

*erossi@twitter.com



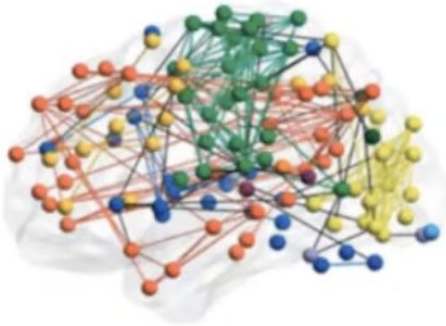
ICLR 2020 submissions keyword statistics

Plot: Pau Rodríguez López

Graphs are Everywhere

Social Networks

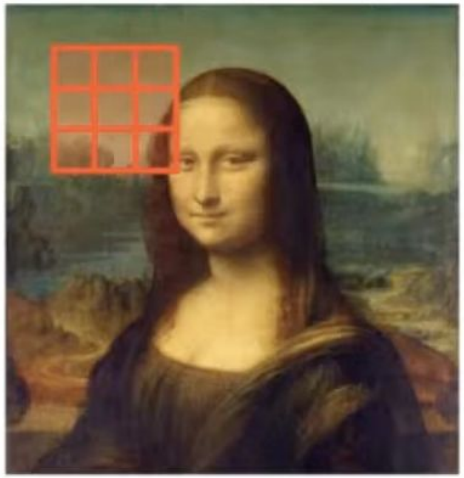
Molecules

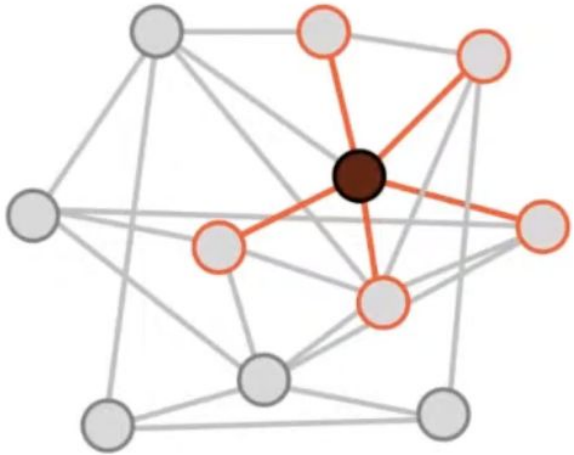


Functional Networks

Interaction Networks

From Images to Graphs



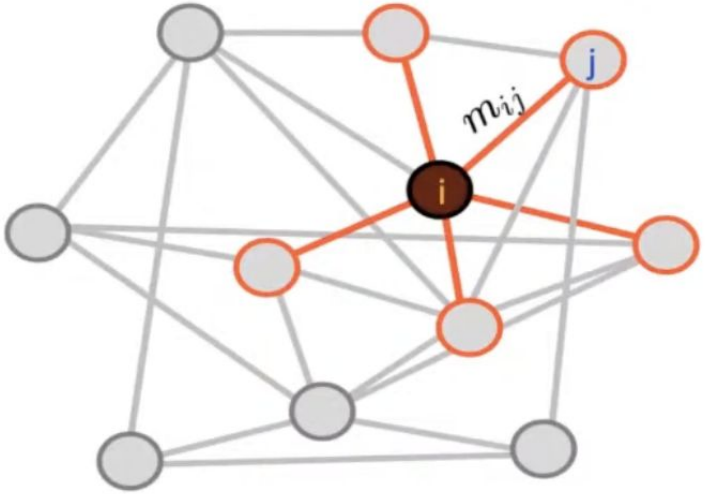


- Constant number of neighbors
- Fixed ordering of neighbors

- Different number of neighbors
- No ordering of neighbors

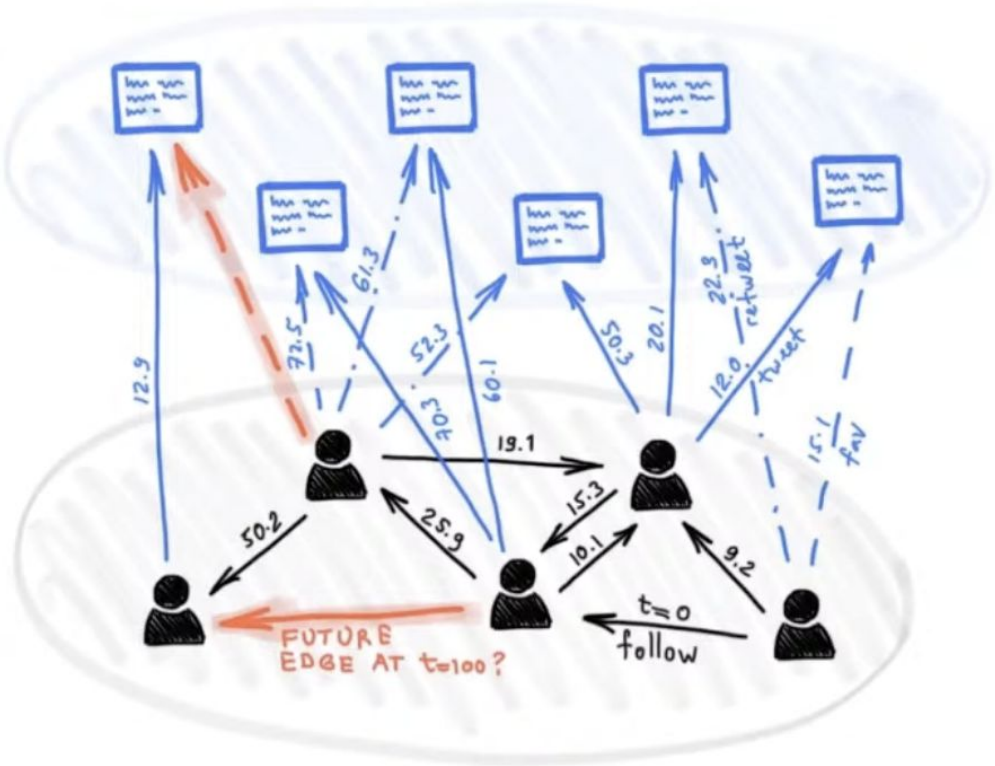
Graph Neural Networks

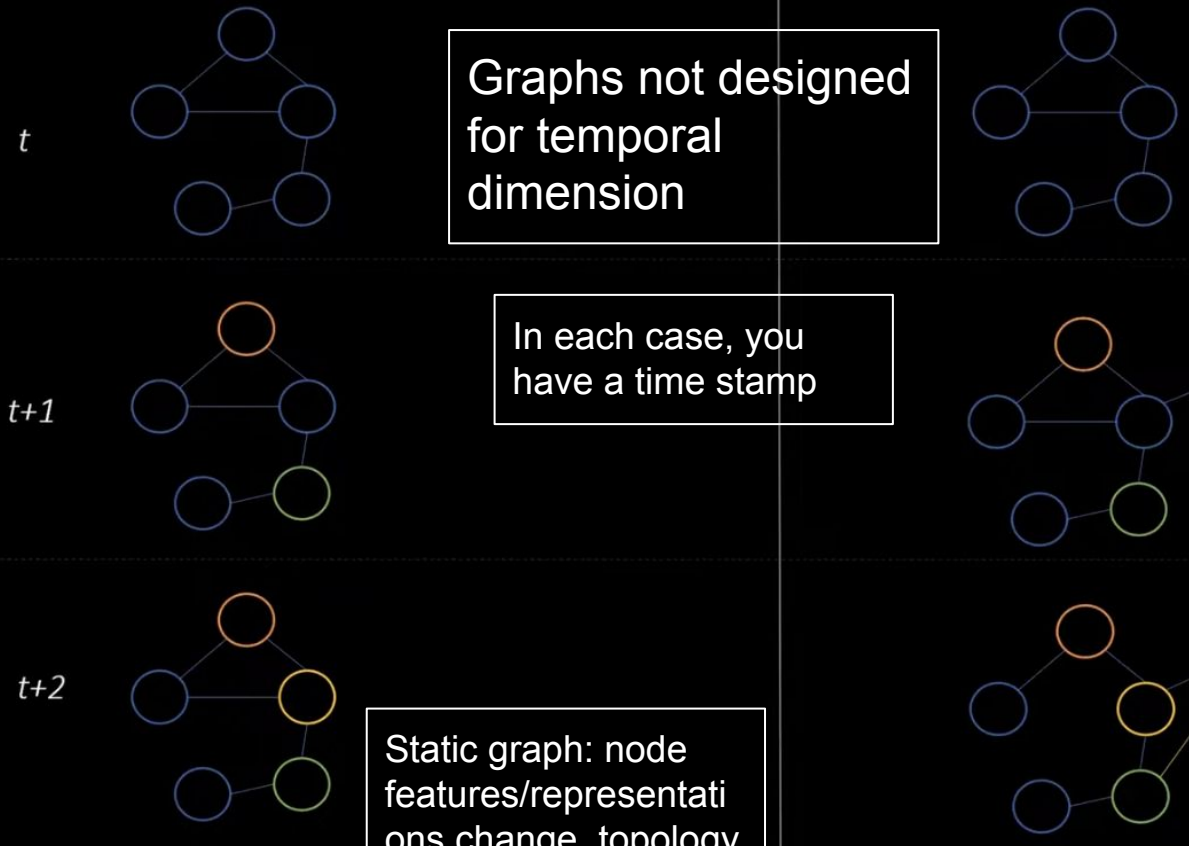
$$\mathbf{m}_{ij} = \text{msg}(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{ij}),$$
$$\mathbf{z}_i = \sum_{j \in \mathcal{N}_i} h(\mathbf{m}_{ij}, \mathbf{v}_i)$$

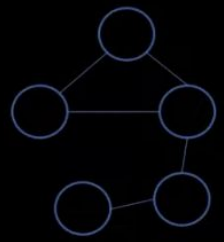


Gilmer et al. 2017

Problem: Many Graphs are Dynamic

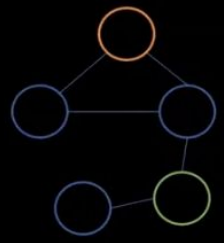



t
t+1
t+2
Graphs not designed for temporal dimension
In each case, you have a time stamp
Static graph: node features/representations change, topology (V,E) is constant
Dynamic Graph: new edges or nodes. (V,E) changes,
Static Graph with Temporal Signals
Dynamic Graph with Temporal Signals



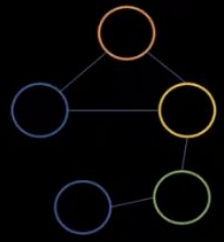
t

$G = (V, E, X_{N,t}, X_{E,t})$



$t+1$

$G = (V, E, X_{N,t+1}, X_{E,t+1})$

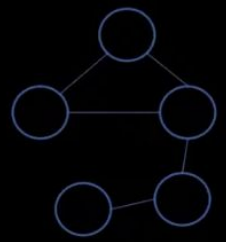


$t+2$

$G = (V, E, X_{N,t+2}, X_{E,t+2})$

Static graph: node features change, (V,E) is constant

Static Graph with Temporal Signals



$G = (V_t, E_t, X_{N,t}, X_{E,t})$

$G = (V_{t+1}, E_{t+1}, X_{N,t+1}, X_{E,t+1})$

$G = (V_{t+2}, E_{t+2}, X_{N,t+2}, X_{E,t+2})$

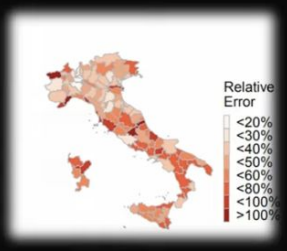
Dynamic Graph: new edges or nodes. (V,E) changes,

Dynamic Graph with Temporal Signals

More applications

Source: DCRNN paper

Traffic Forecasting



Source: Transfer GNN for Pandemic forecasting

Epidemics (Covid Predictions)

Source: mediapipe

Motion Classification

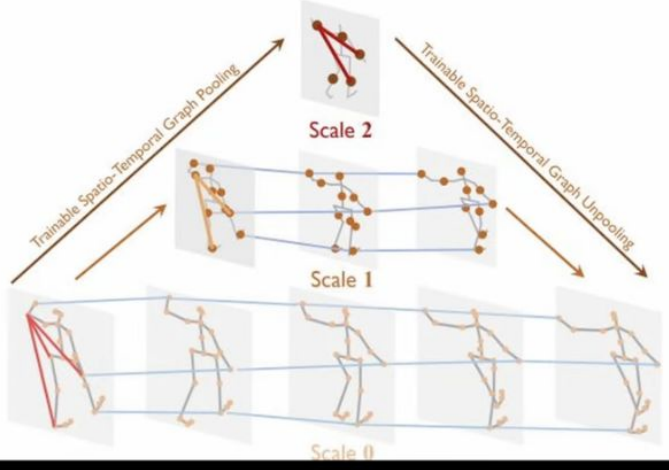
Source: GCLSTM, Simeunovic et al.

Power Systems Forecasting

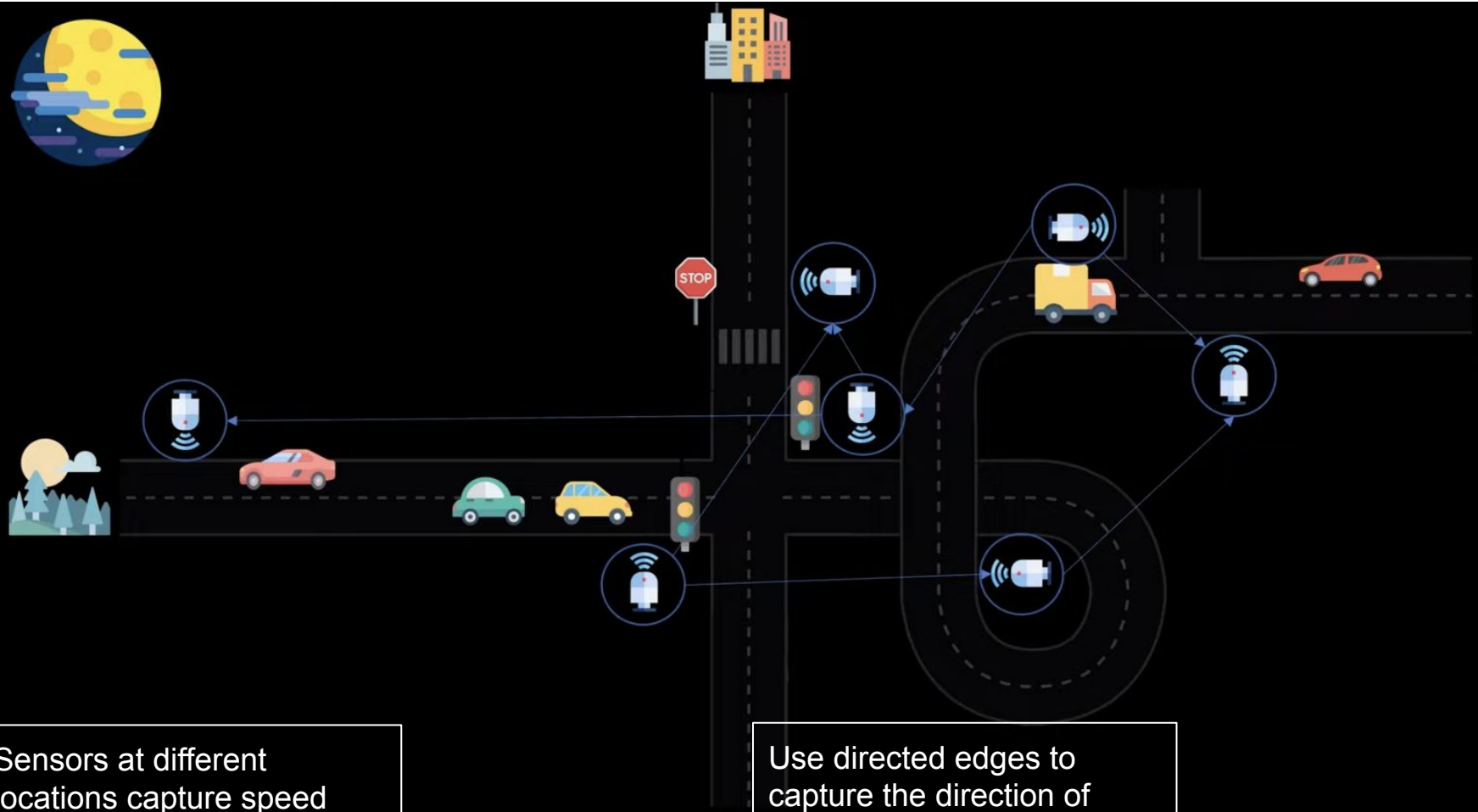
Multiscale Spatio-Temporal Graph Neural Networks for 3D Skeleton-Based Motion Prediction

Maosen Li, *Student Member, IEEE,* Siheng Chen, *Member, IEEE,* Yangheng Zhao, Ya Zhang, *Member, IEEE,* Yanfeng Wang, and Qi Tian, *Fellow, IEEE*

***Abstract*—We propose a multiscale spatio-temporal graph neural network (MST-GNN) to predict the future 3D skeleton-based human poses in an action-category-agnostic manner. The core of MST-GNN is a multiscale spatio-temporal graph that explicitly models the relations in motions at various spatial and temporal scales. Different from many previous hierarchical structures, our multiscale spatio-temporal graph is built in a *data-adaptive fashion*, which captures nonphysical, yet motion-based relations. The key module of MST-GNN is a multiscale spatio-temporal graph computational unit (MST-GCU) based on the trainable graph structure. MST-GCU embeds underlying features at individual scales and then fuses features across scales to obtain a comprehensive representation. The overall architecture of MST-GNN follows an encoder-decoder framework, where the encoder consists of a sequence of MST-GCUs to learn the spatial and**

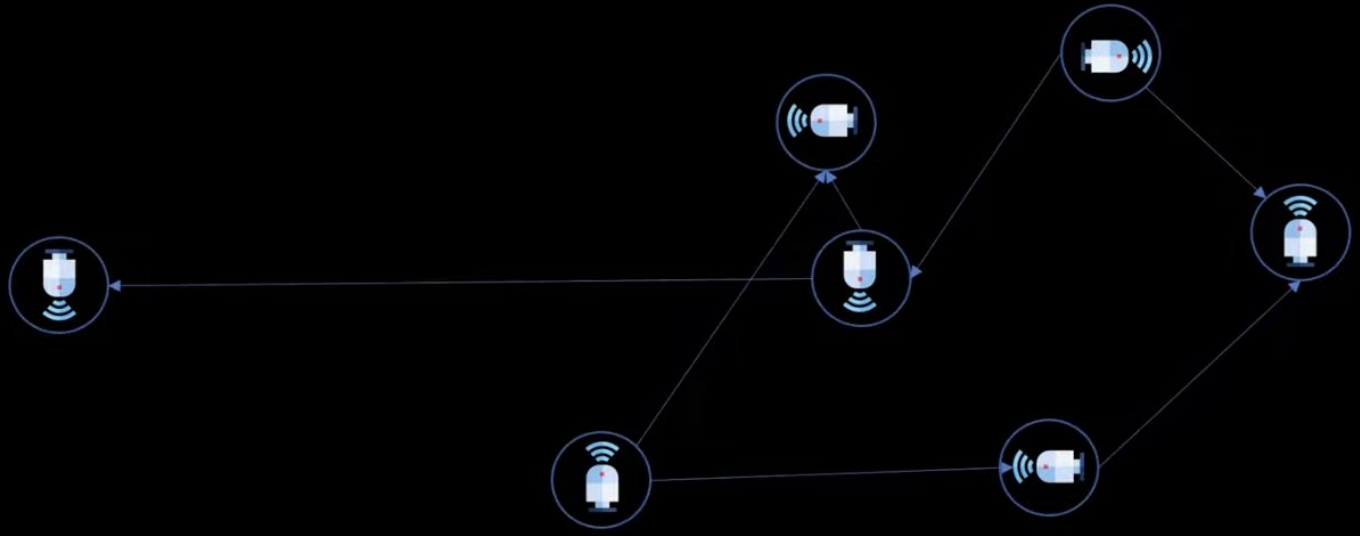


The road vector graphics are from: https://de.vecteezy.com/vektorkunst/1…

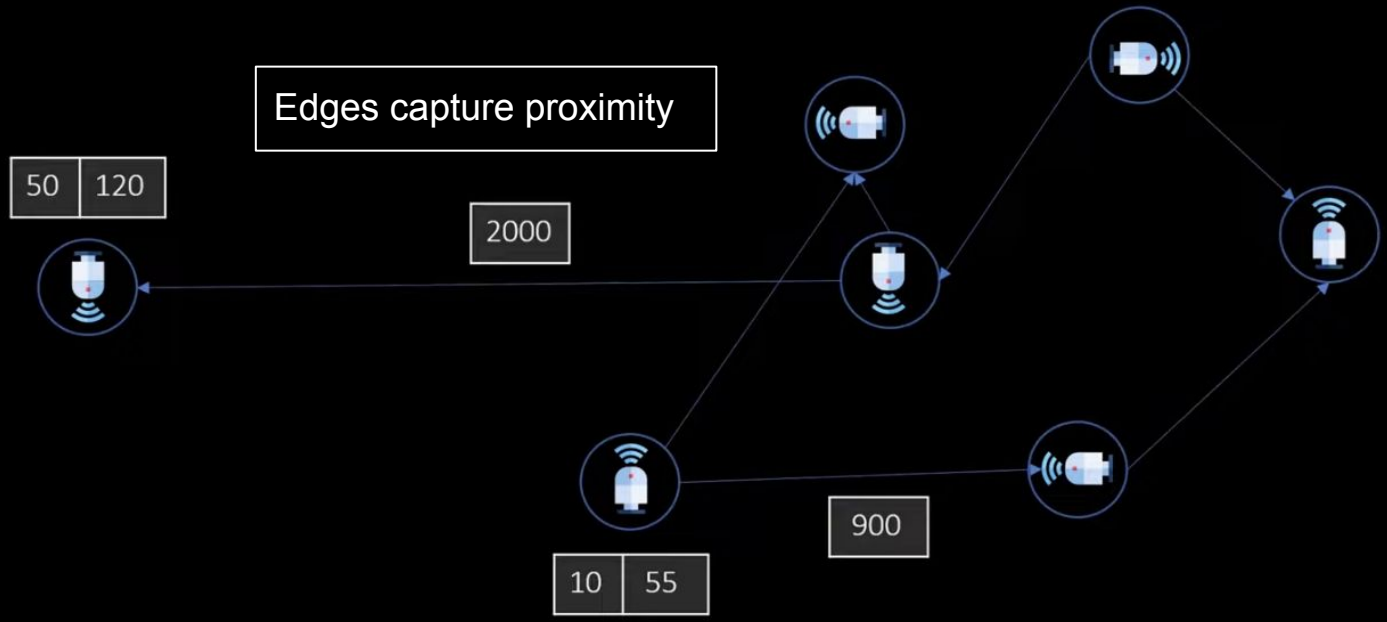


Sensors at different locations capture speed and volume

Use directed edges to capture the direction of traffic flow

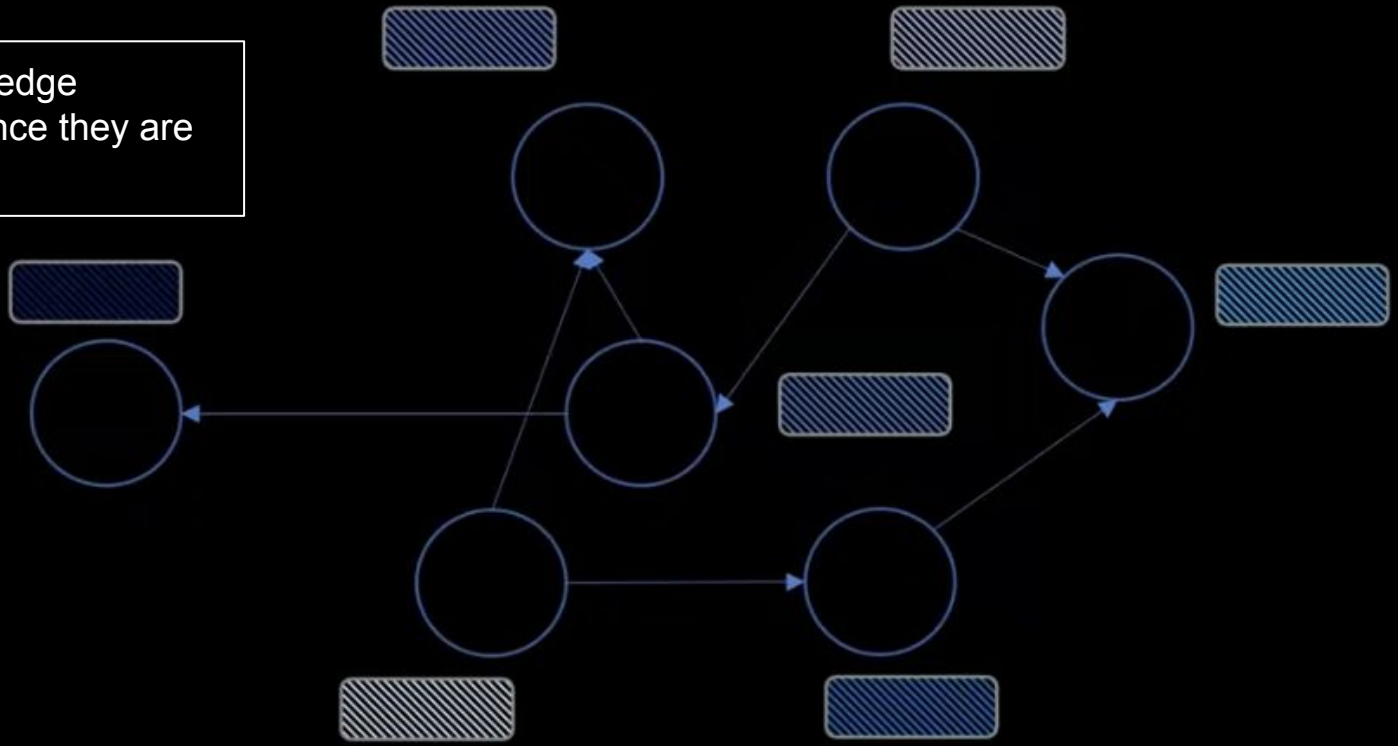


Now we just have nodes



Node features include speed and volume. These are signals for our graph.

Can ignore edge proximity since they are constant

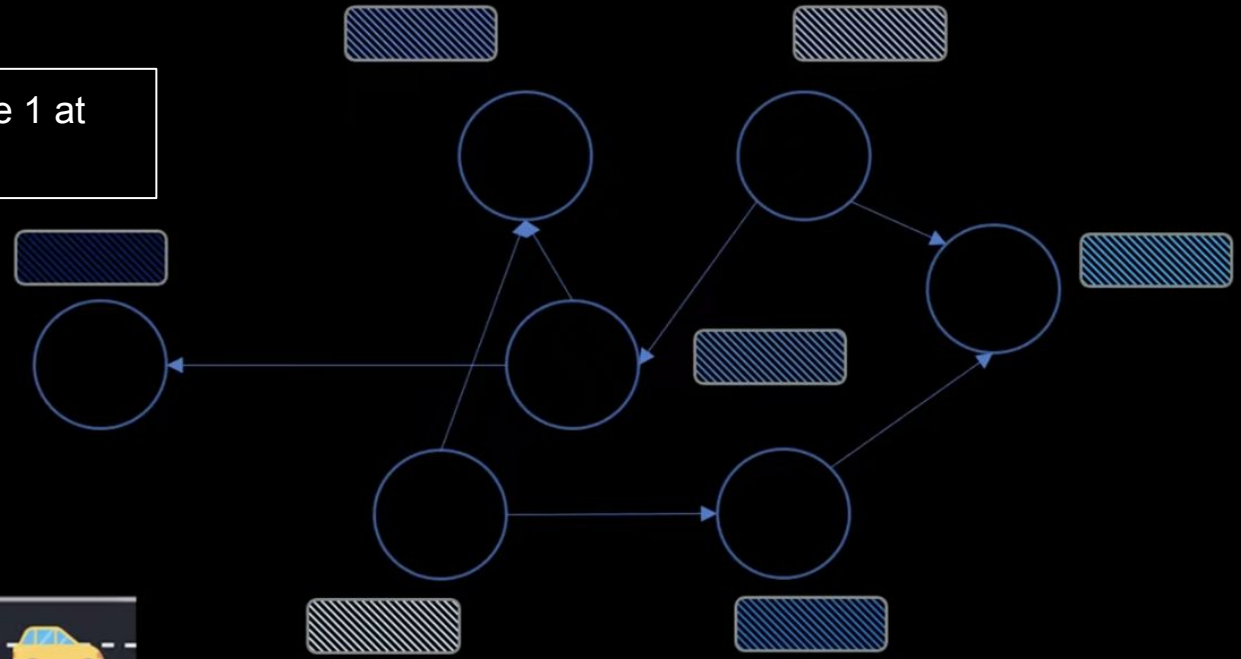


Nodes with features and directed edges.

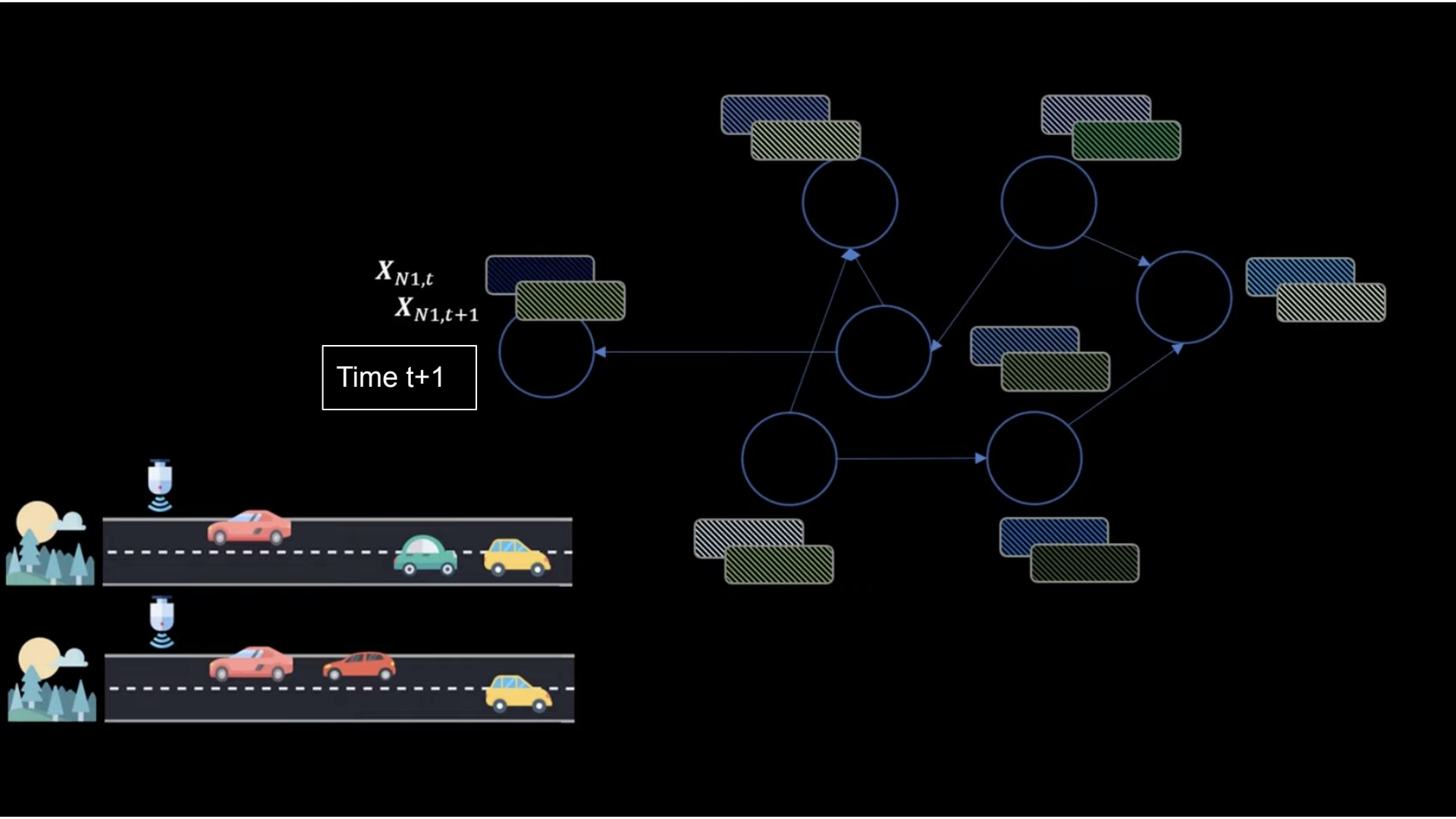
$$G = (V, E, X_{N,t}, X_E)$$

X indexed and Node 1 at time t

$X_{N1,t}$

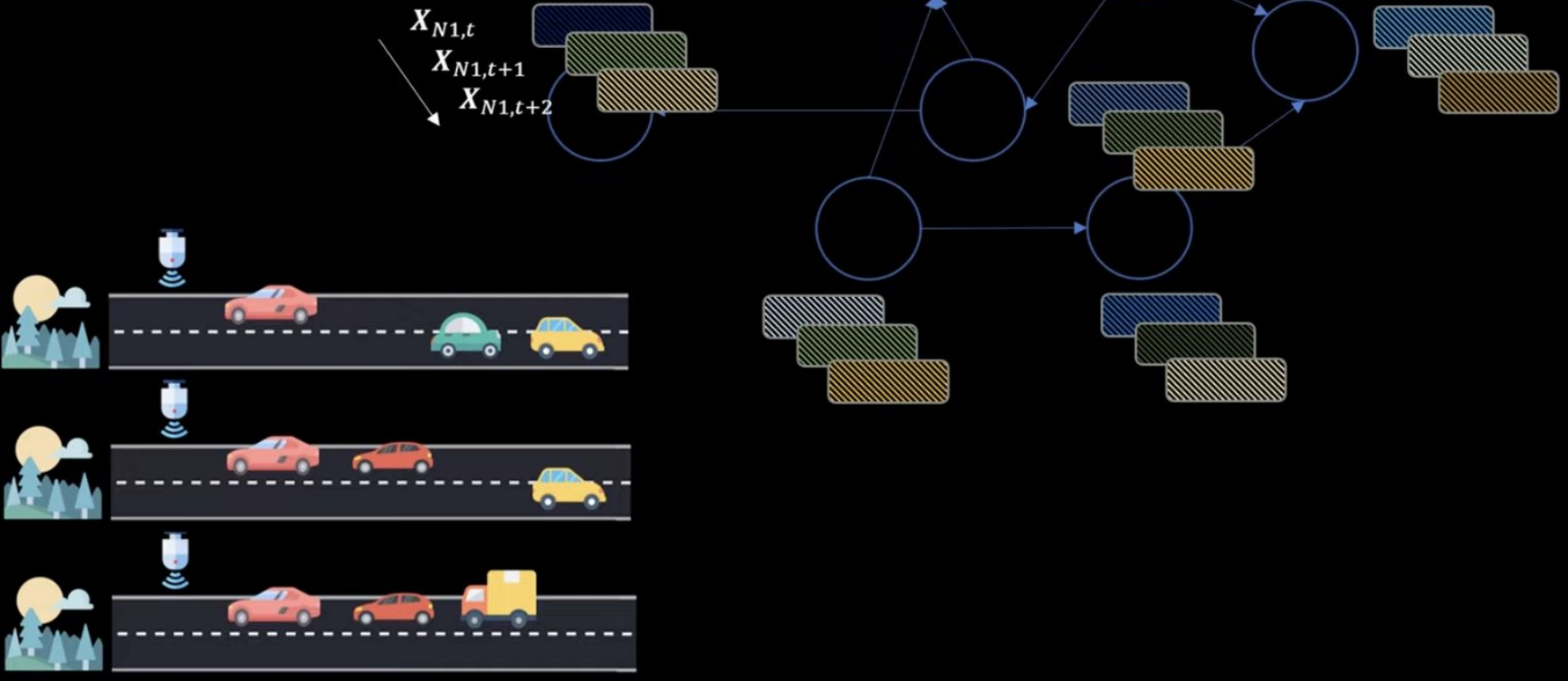


Nodes with features and directed edges.

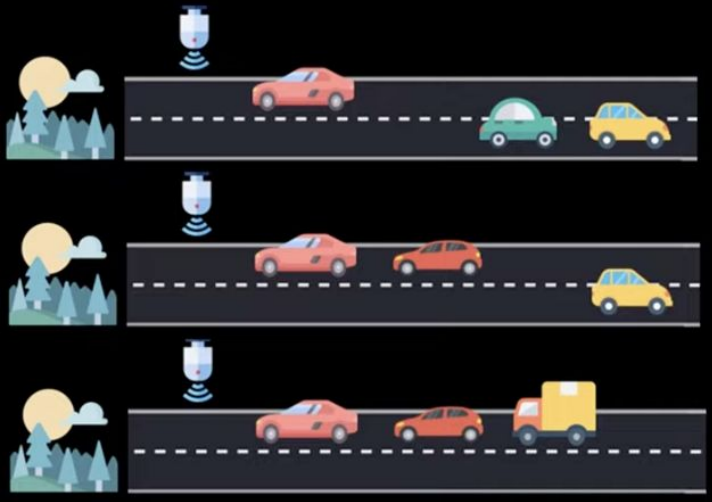

$X_{N1,t}$
$X_{N1,t+1}$
Time t+1

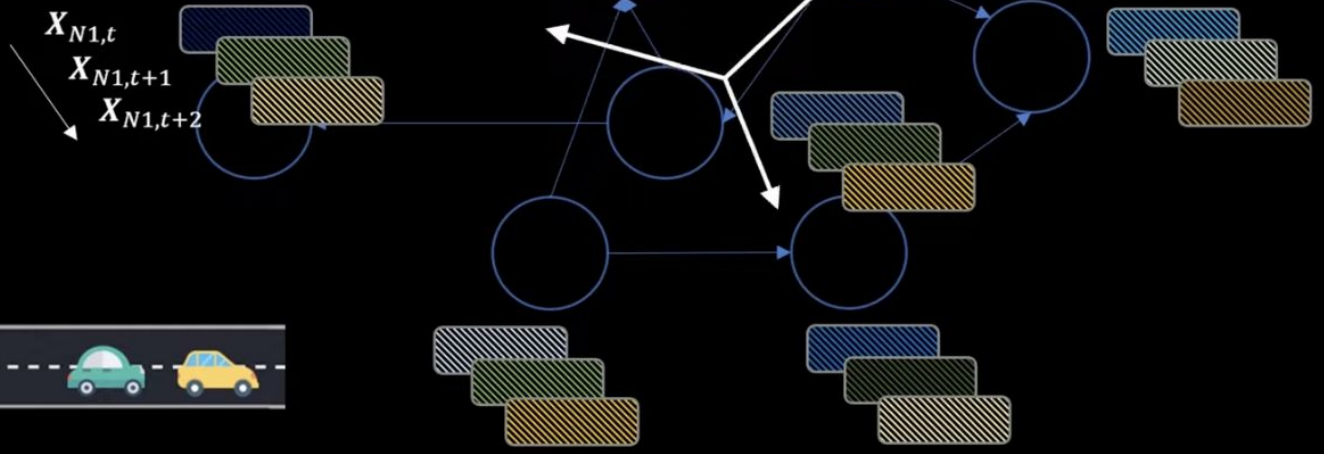
Find information in 2 dimensions:

- Spatial dimension
- Time dimension

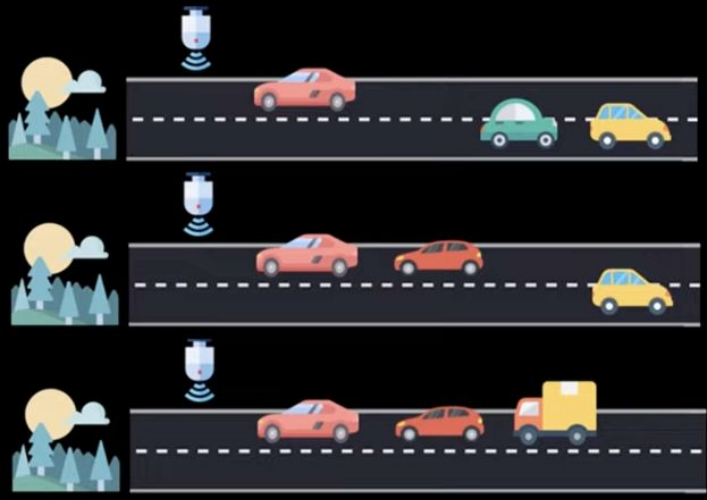


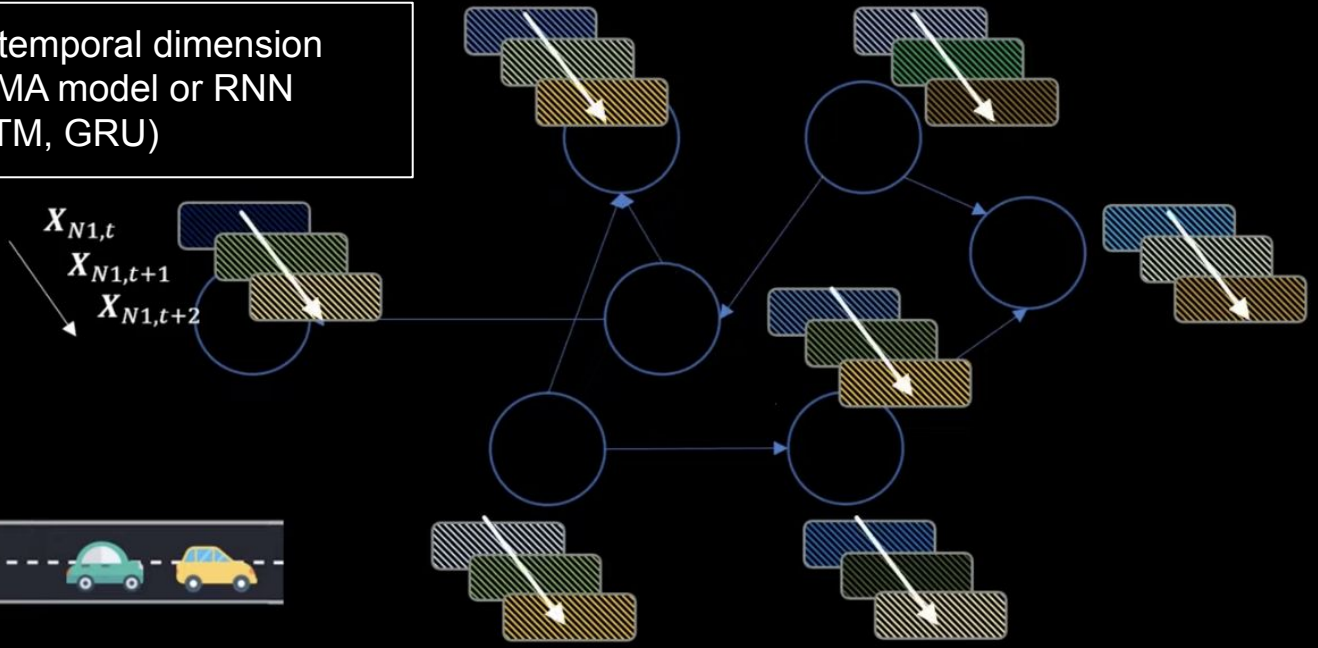
For spatial dimension using anyone of the models we've learned: Graph NN, Graph CNN, GraphSage

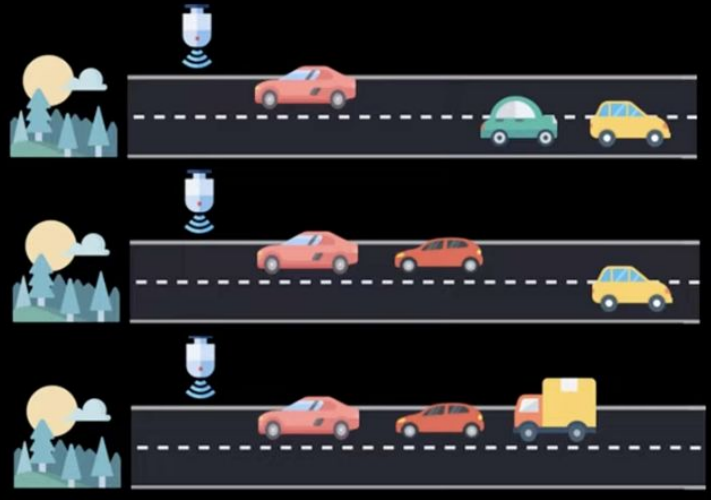


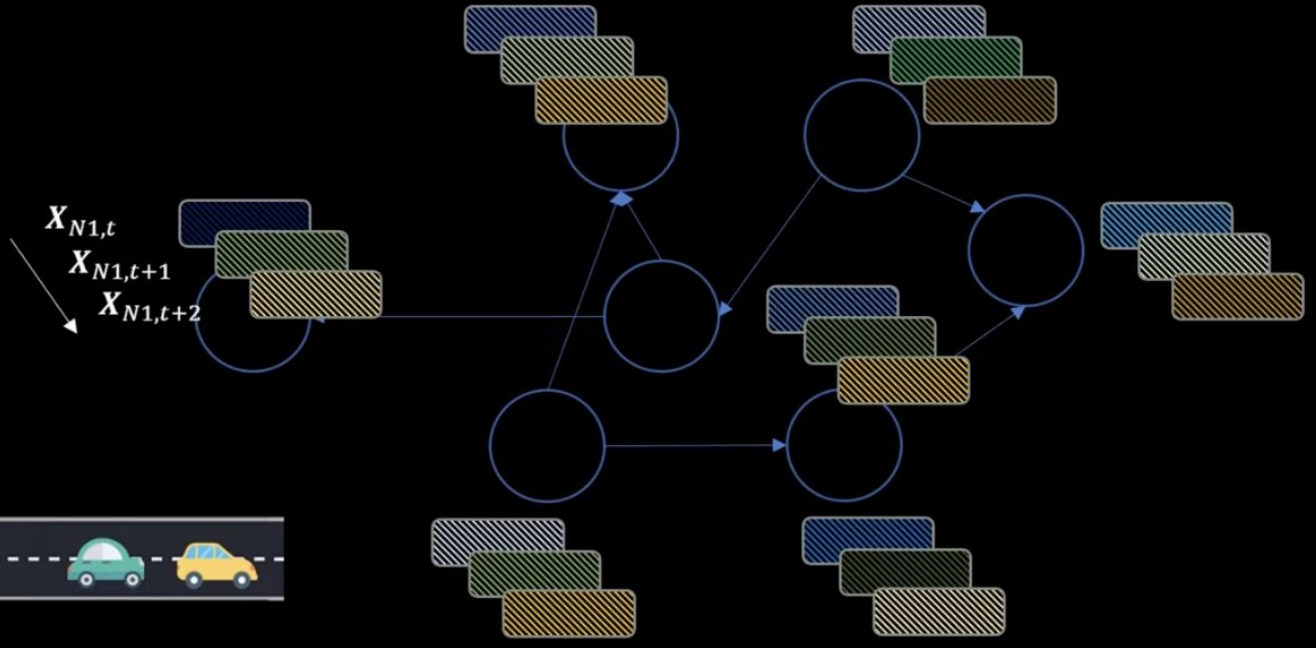


For temporal dimension
ARIMA model or RNN
(LSTM, GRU)

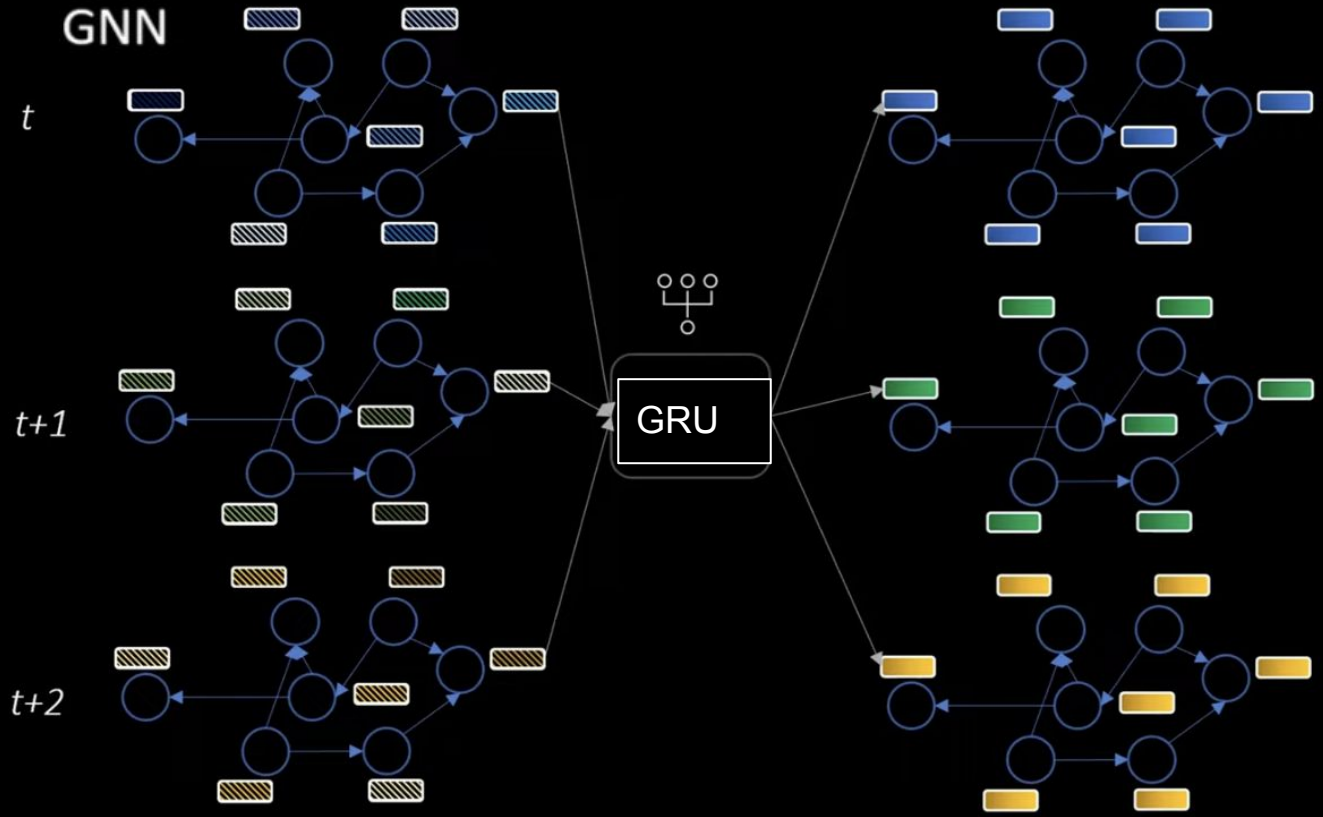








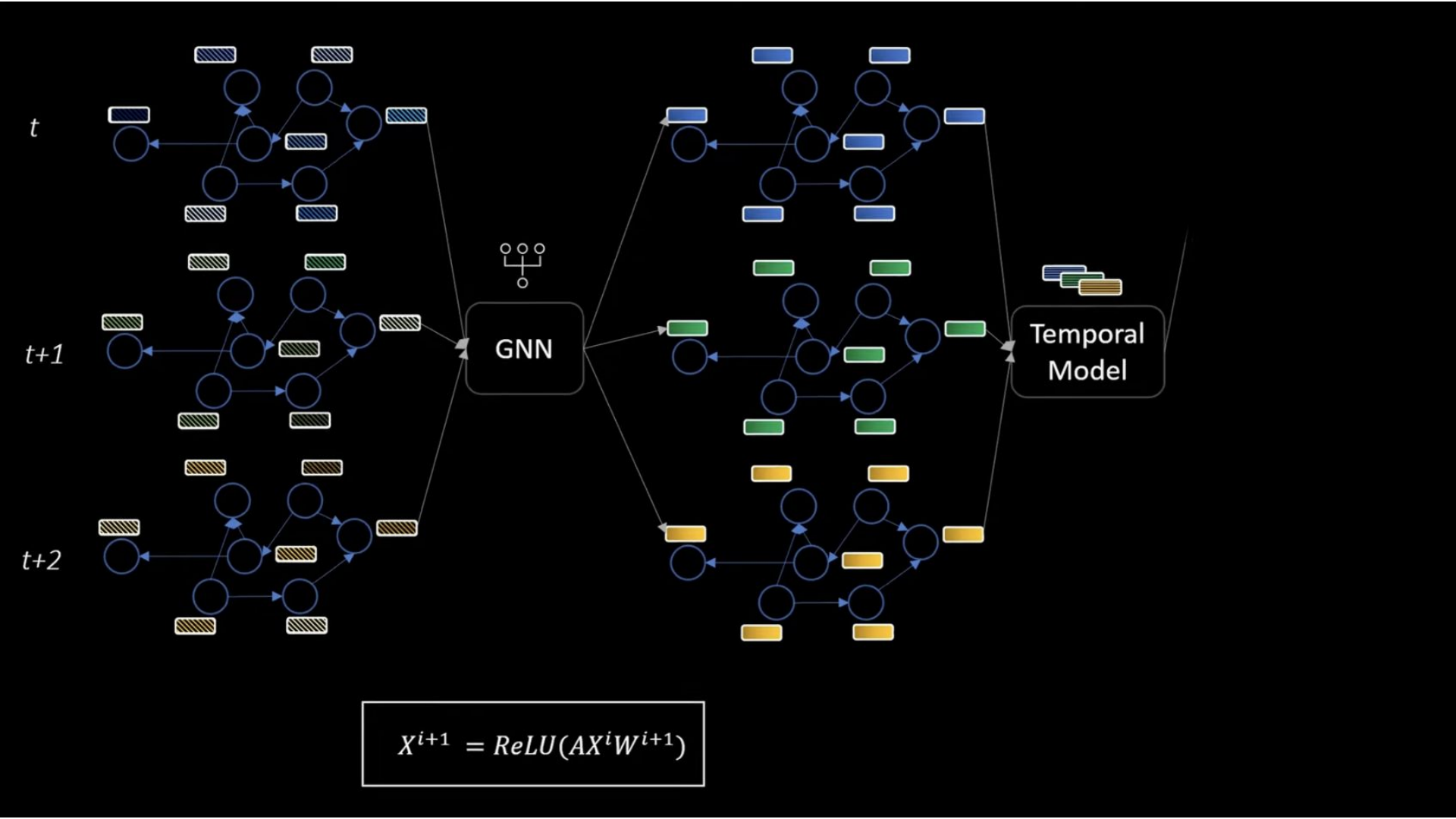
A Spatial-Temporal Graph learning integrates spatial + temporal models



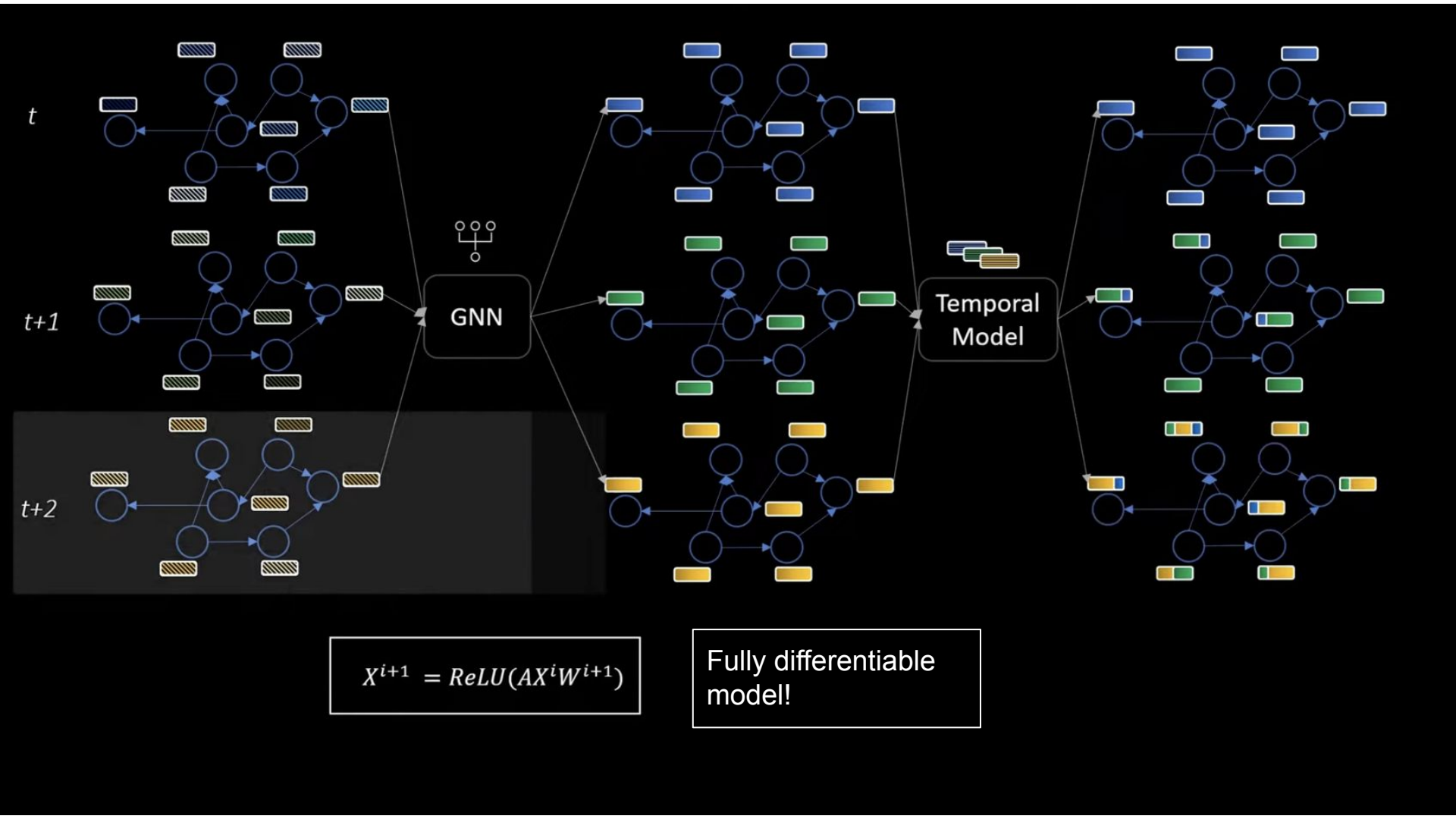
Graph attention Networks are popular.

$$X^{i+1} = ReLU(AX^iW^{i+1})$$

Need LSTM or GRU to store previous information

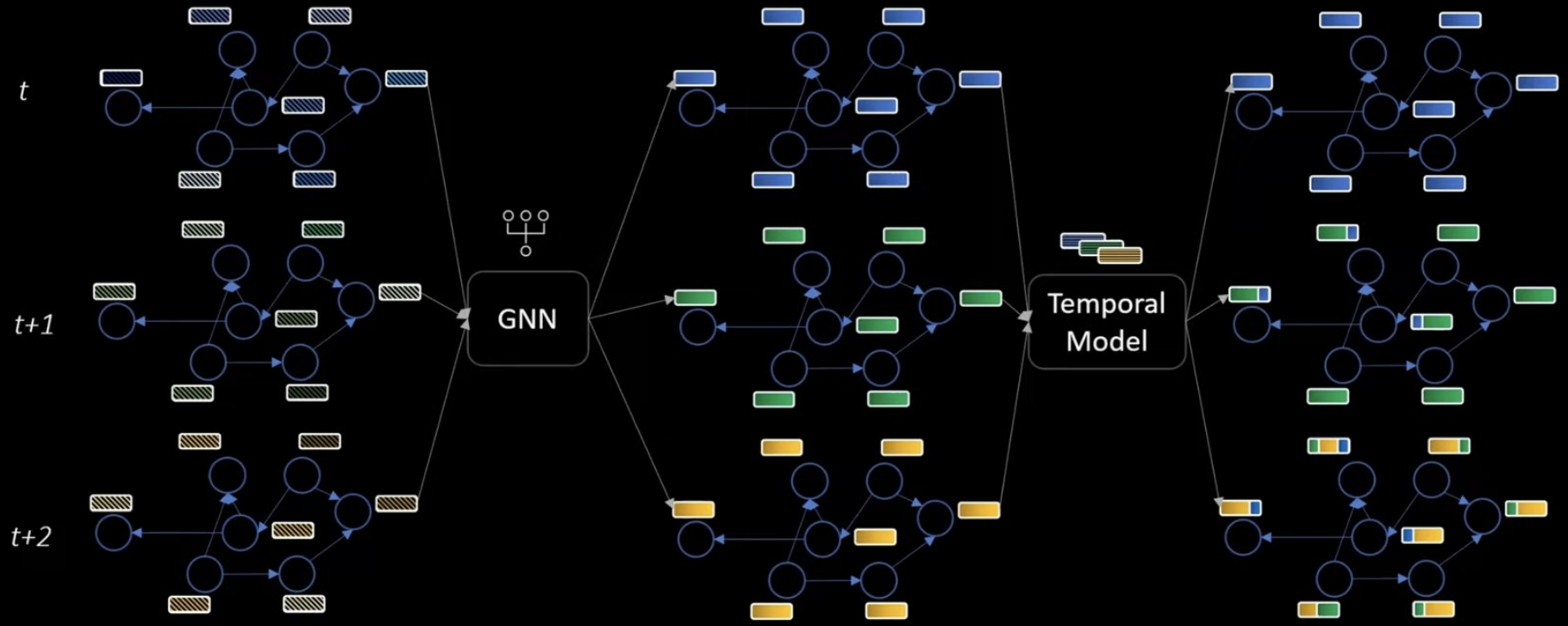


$$X^{i+1} = ReLU(AX^iW^{i+1})$$



$$X^{i+1} = ReLU(AX^iW^{i+1})$$

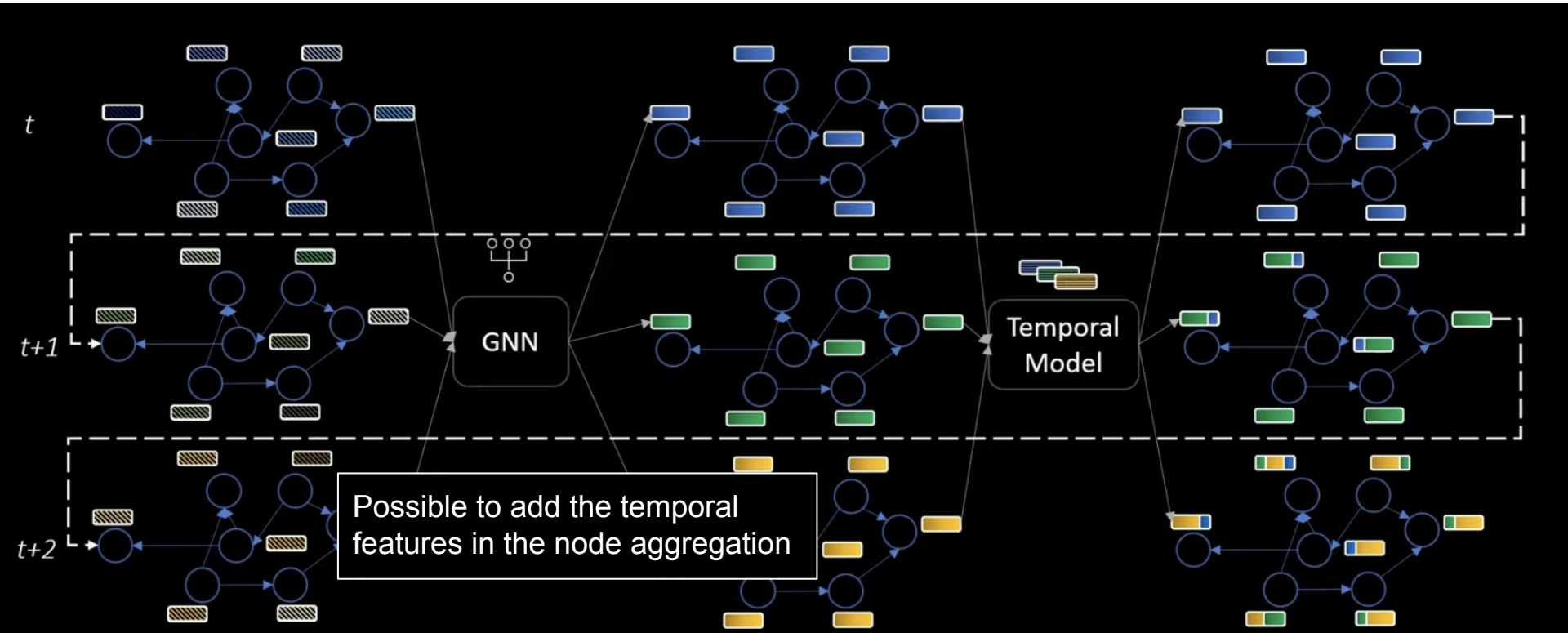
Fully differentiable model!



$$X^{i+1} = ReLU(AX^iW^{i+1})$$

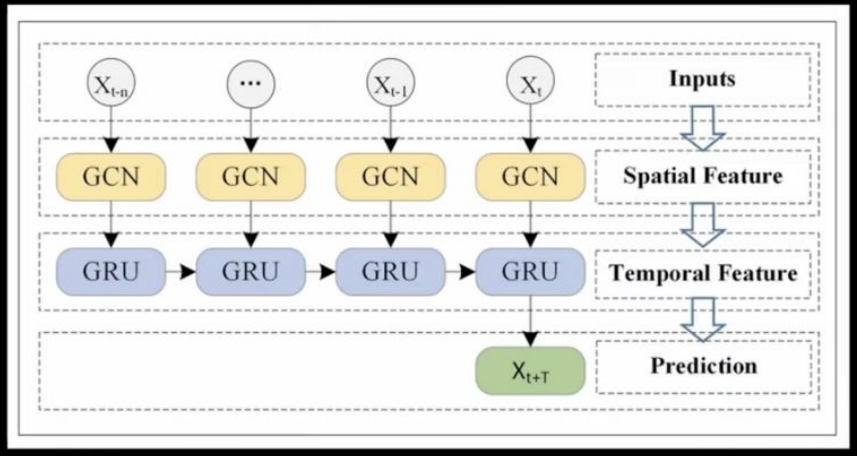
Learned embeddings combine spatial and temporal information

$$u_t = \sigma(W_u GNN(A, X_t) + U_u h_{t-1}] + b_u)$$
$$r_t = \sigma(W_r GNN(A, X_t) + U_r h_{t-1}] + b_r)$$
$$\hat{h}_t = tanh(W_h GNN(A, X_t) + U_h r_t * h_{t-1}] + b_h)$$
$$\boldsymbol{h_t = (1 - u_t) * h_{t-1} + u_t * \hat{h}_t}$$



$$X^{i+1} = ReLU(AX^iW^{i+1})$$

$$u_t = \sigma(W_u GNN(A, X_t) + U_u h_{t-1}] + b_u)$$
$$r_t = \sigma(W_r GNN(A, X_t) + U_r h_{t-1}] + b_r)$$
$$\hat{h}_t = tanh(W_h GNN(A, X_t) + U_h r_t * h_{t-1}] + b_h)$$
$$\boldsymbol{h_t = (1 - u_t) * h_{t-1} + u_t * \hat{h}_t}$$

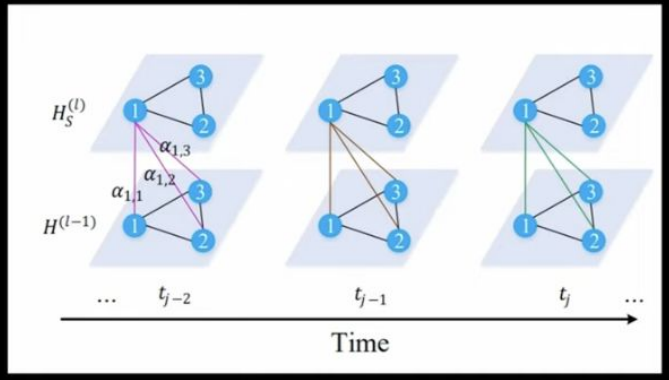


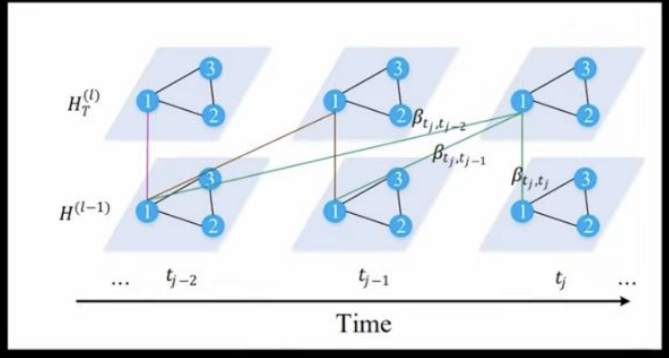
T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction, Zhao et al.

- GNN and GRU can have multiple layers

Model	Temporal Layer	GNN Layer	Proximity Order	Multi Type
DCRNN [32]	GRU	DiffConv	Higher	False
GConvGRU [51]	GRU	Chebyshev	Lower	False
GConvLSTM [51]	LSTM	Chebyshev	Lower	False
GC-LSTM [10]	LSTM	Chebyshev	Lower	True
DyGrAE [54, 55]	LSTM	GGCN	Higher	False
LRGCN [31]	LSTM	RGCN	Lower	False
EGCN-H [39]	GRU	GCN	Lower	False
EGCN-O [39]	LSTM	GCN	Lower	False
T-GCN [65]	GRU	GCN	Lower	False
A3T-GCN [68]	GRU	GCN	Lower	False
AGCRN [4]	GRU	Chebyshev	Higher	False
MPNN LSTM [38]	LSTM	GCN	Lower	False
STGCN [63]	Attention	Chebyshev	Higher	False
ASTGCN [22]	Attention	Chebyshev	Higher	False
MSTGCN [22]	Attention	Chebyshev	Higher	False
GMAN [66]	Attention	Custom	Lower	False
MTGNN [61]	Attention	Custom	Higher	False
AAGCN [52]	Attention	Custom	Higher	False

PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models, Rozemberczki et al.

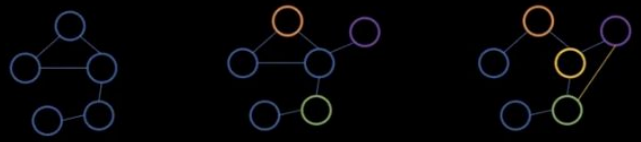




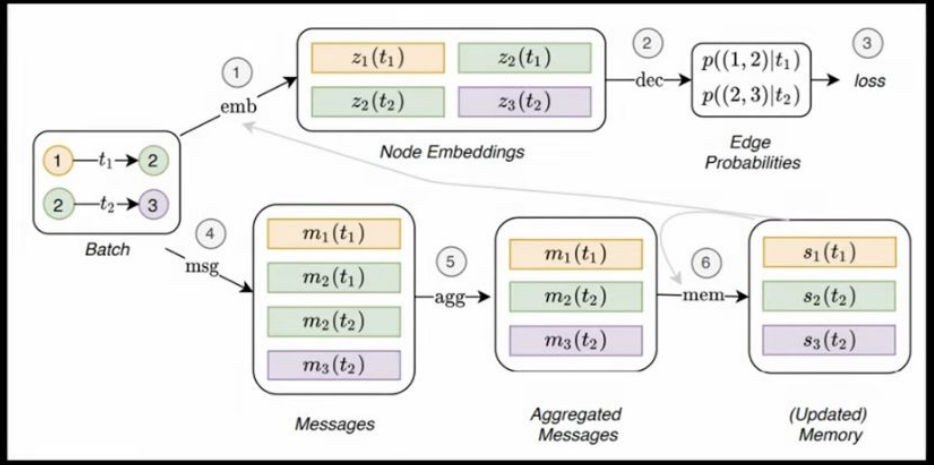
Graph Multl-Attention Network for Traffic Prediction, Zheng et al.

Model	Temporal Layer	GNN Layer	Proximity Order	Multi Type
DCRNN [32]	GRU	DiffConv	Higher	False
GConvGRU [51]	GRU	Chebyshev	Lower	False
GConvLSTM [51]	LSTM	Chebyshev	Lower	False
GC-LSTM [10]	LSTM	Chebyshev	Lower	True
DyGrAE [54, 55]	LSTM	GGCN	Higher	False
LRGCN [31]	LSTM	RGCN	Lower	False
EGCN-H [39]	GRU	GCN	Lower	False
EGCN-O [39]	LSTM	GCN	Lower	False
T-GCN [65]	GRU	GCN	Lower	False
A3T-GCN [68]	GRU	GCN	Lower	False
AGCRN [4]	GRU	Chebyshev	Higher	False
MPNN LSTM [38]	LSTM	GCN	Lower	False
STGCN [63]	Attention	Chebyshev	Higher	False
ASTGCN [22]	Attention	Chebyshev	Higher	False
MSTGCN [22]	Attention	Chebyshev	Higher	False
GMAN [66]	Attention	Custom	Lower	False
MTGNN [61]	Attention	Custom	Higher	False
AAGCN [52]	Attention	Custom	Higher	False

Spatial attention pays attention to the most import roads. Temporal attention pays attention to previous time steps



Nodes and edges can be added or removed at each time step



Maintain memory module over time

Temporal Graph Networks for Deep Learning on Dynamic Graphs

Emanuele Rossi[*] Twitter
Ben Chamberlain Twitter
Fabrizio Frasca Twitter
Davide Eynard Twitter
Federico Monti Twitter
Michael Bronstein Twitter

Abstract

Graph Neural Networks (GNNs) have recently become increasingly popular due to their ability to learn complex systems of relations or interactions. These arise in a broad spectrum of problems ranging from biology and particle physics to social networks and recommendation systems. Despite the plethora of different models for deep learning on graphs, few approaches have been proposed for dealing with graphs that are dynamic in nature (e.g. evolving features or connectivity over time). We present Temporal Graph Networks (TGNs), a generic, efficient framework for deep learning on dynamic graphs represented as sequences of timed events. Thanks to a novel combination of memory modules and graph-based operators, TGNs significantly outperform previous approaches while being more computationally efficient. We furthermore show that several previous models for learning on dynamic graphs can be cast as specific instances of our framework. We perform a detailed ablation study of different components of our framework and devise the best configuration that achieves state-of-the-art performance on several transductive and inductive prediction tasks for dynamic graphs.

1 Introduction

In the past few years, graph representation learning (Bronstein et al., 2017; Hamilton et al., 2017b; Battaglia et al., 2018) has produced a sequence of successes, gaining increasing popularity in machine learning. Graphs are ubiquitously used as models for systems of relations and interactions in many fields (Battaglia et al., 2016; Qi et al., 2018; Monti et al., 2016; Choma et al., 2018; Duvenaud et al., 2015; Gilmer et al., 2017; Parisot et al., 2018; Rossi et al., 2019), in particular, social sciences (Ying et al., 2018; Monti et al., 2019; Rossi et al., 2020) and biology (Zitnik et al., 2018; Veselkov et al., 2019; Gainza et al., 2019). Learning on such data is possible using graph neural networks (GNNs) (Hamilton et al., 2017a) that typically operate by a message passing mechanism (Battaglia et al., 2018) aggregating information in a neighborhood of a node and create node embeddings that are then used for node classification (Monti et al., 2016; Velickovic et al., 2018; Kipf & Welling, 2017), graph classification (Gilmer et al., 2017), or edge prediction (Zhang & Chen, 2018) tasks.

The majority of methods for deep learning on graphs assume that the underlying graph is *static*. However, most real-life systems of interactions such as social networks or biological interactomes are *dynamic*. While it is often possible to apply static graph deep learning models (Liben-Nowell & Kleinberg, 2007) to dynamic graphs by ignoring the temporal evolution, this has been shown to

Traffic Forecasting with Pytorch Geometric Temporal

Installation

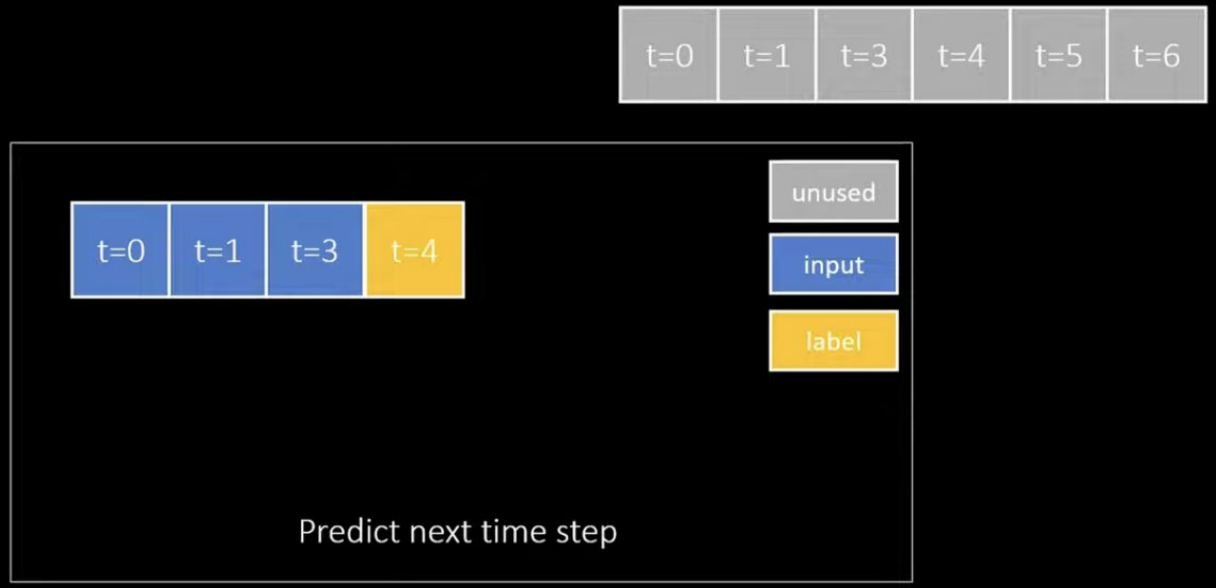
```
import torch
from IPython.display import clear_output
pt_version = torch.__version__
print(pt_version)
```

```
1.10.0+cu111
```

This took some time for me, so be patient :)

```
!pip install torch-scatter -f https://pytorch-geometric.com/whl/torch-${pt_version}.html
!pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-${pt_version}.html
!pip install torch-cluster -f https://pytorch-geometric.com/whl/torch-${pt_version}.html
!pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-${pt_version}.html
!pip install torch-geometric
!pip install torch-geometric-temporal
clear_output()
```

```
Looking in links: https://pytorch-geometric.com/whl/torch-.10.0+cu111.html
Collecting torch-scatter
  Downloading torch_scatter-2.0.9.tar.gz (21 kB)
Building wheels for collected packages: torch-scatter
  Building wheel for torch-scatter (setup.py) ... done
```


t=0
t=1
t=3
t=4
t=5
t=6
t=0
t=1
t=3
t=4
unused
input
label
Predict next time step

t=0	t=1	t=3	t=4	t=5	t=6

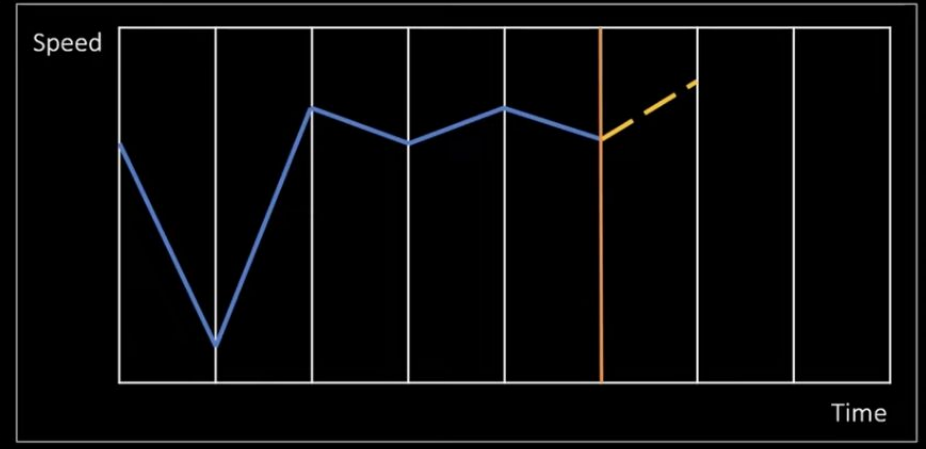
t=0	t=1	t=3	t=4			
t=0	t=1	t=3	t=4	t=5		
t=0	t=1	t=3	t=4	t=5	t=6	

unused

input

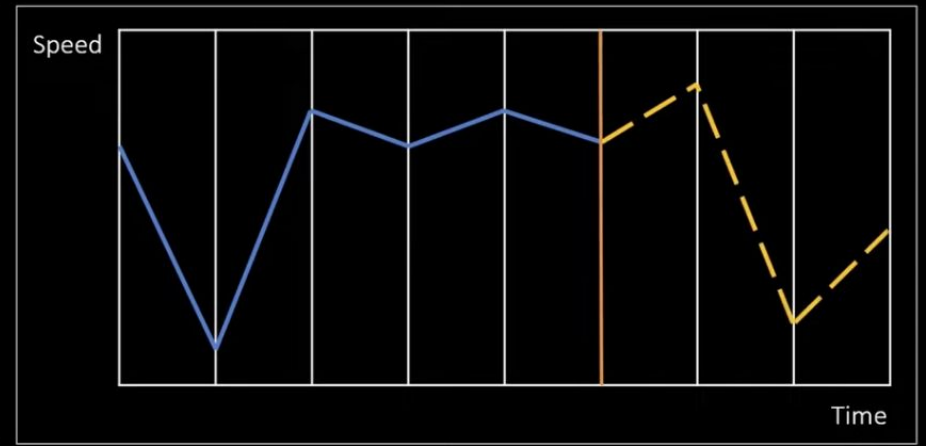
label

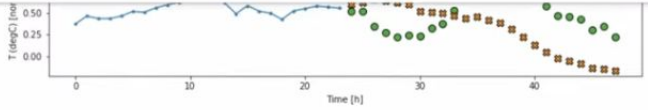
Predict next time step



t=0	t=1	t=3	t=4	t=5	t=6	
t=0	t=1	t=3	t=4	t=5	t=6	t=7
t=0	t=1	t=3	t=4	t=5	t=6	t=7	t=8

Predict multiple time steps

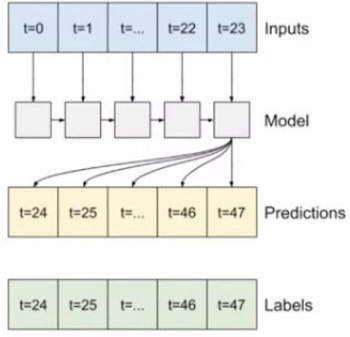




RNN

A recurrent model can learn to use a long history of inputs, if it's relevant to the predictions the model is making. Here the model will accumulate internal state for 24 hours, before making a single prediction for the next 24 hours.

In this single-shot format, the LSTM only needs to produce an output at the last time step, so set `return_sequences=False` in `tf.keras.layers.LSTM`.



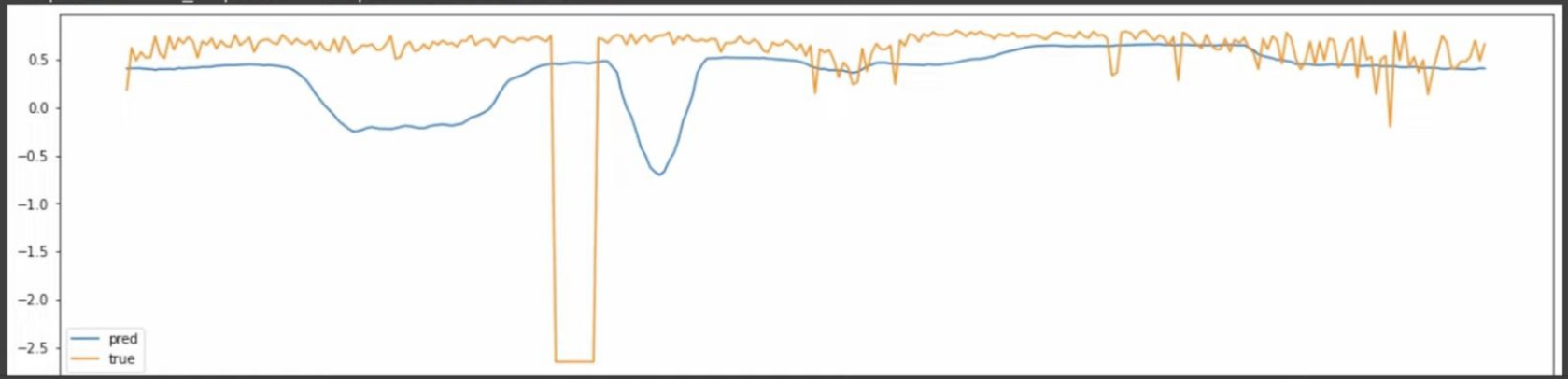

```
multi_lstm_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, lstm_units].
    # Adding more `lstm_units` just overfits more quickly.
```

Results - needs work!

```
Data points:, (289,)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,5))
sns.lineplot(data=preds, label="pred")
sns.lineplot(data=labs, label="true")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7b14858950>
```



Executing (2m 26s) Cell > _call_impl() > forward() > _call_impl() > forward() > _call_impl() > forward() > _calculate_candidate_state() > _call_impl() > forward() > propagate() > aggregate() > scatter() > scatter_sum()

A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting

Jiawei Zhu, Yujiao Song, Lin Zhao and Haifeng Li*

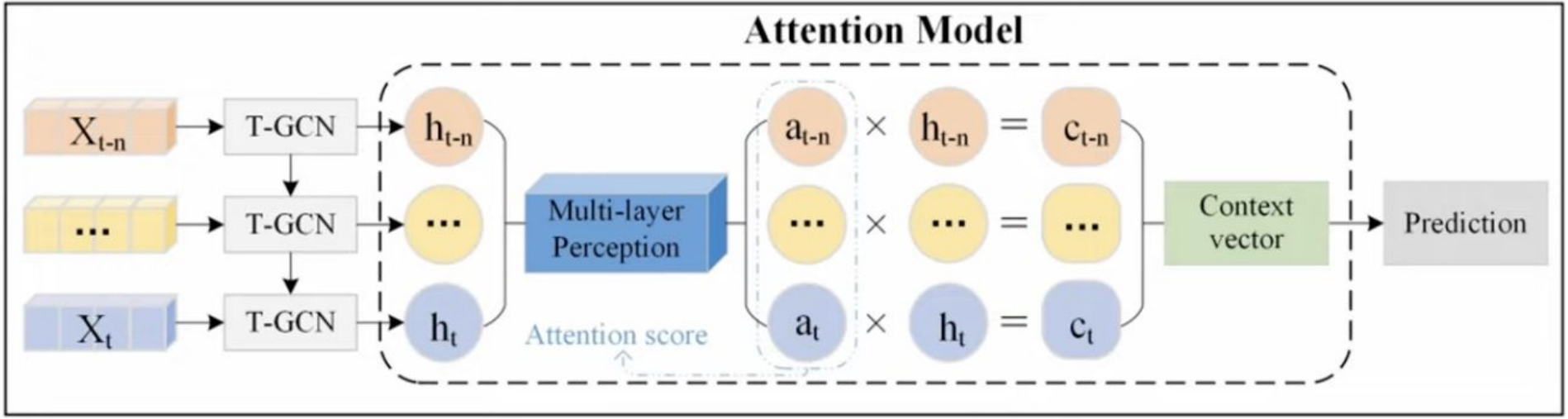
Abstract—Accurate real-time traffic forecasting is a core technological problem against the implementation of the intelligent transportation system. However, it remains challenging considering the complex spatial and temporal dependencies among traffic flows. In the spatial dimension, due to the connectivity of the road network, the traffic flows between linked roads are closely related. In terms of the temporal factor, although there exists a tendency among adjacent time points in general, the importance of distant past points is not necessarily smaller than that of recent past points since traffic flows are also affected by external factors. In this study, an attention temporal graph convolutional network (A3T-GCN) traffic forecasting method was proposed to simultaneously capture global temporal dynamics and spatial correlations. The A3T-GCN model learns the short-time trend in time series by using the gated recurrent units and learns the spatial dependence based on the topology of the road network through the graph convolutional network. Moreover, the attention mechanism was introduced to adjust the importance of different time points and assemble global temporal information to improve prediction accuracy. Experimental results in real-world datasets demonstrate the effectiveness and robustness of proposed A3T-GCN. The source code can be visited at https://github.com/lehaifeng/T-GCN/A3T.

Index Terms—traffic forecasting, attention temporal graph convolutional network, spatial dependence, temporal dependence

1 Introduction

Traffic forecasting is an important component of intelligent transportation systems and a vital part of transportation planning and management and traffic control [12, 15–17]. Accurate real-time traffic forecasting has been a great challenge because of complex spatiotemporal dependencies. Temporal dependence means that traffic state changes with time, which is manifested by periodicity and

(RNNs), long short-term memory (LSTM) [13], and gated recurrent units (GRUs)[7] have been successfully utilized in traffic forecasting because they can use self-circulation mechanism and model temporal dependence [20, 25]. However, these models only consider the temporal variation of traffic state and neglect spatial dependence. Many scholars have introduced convolutional neural networks (CNNs) in


Attention Model
X_{t-n}
...
X_t
T-GCN
T-GCN
T-GCN
h_{t-n}
...
h_t
Multi-layer Perception
Attention score
a_{t-n}
...
a_t
×
h_{t-n}
...
h_t
=
c_{t-n}
...
c_t
Context vector
Prediction