THE 16TH ACM INTERNATIONAL CONFERENCE ON
WEB SEARCH AND DATA MINING
wsdm 2023
SINGAPORE FEBRUARY 27 - MARCH 3

Natural and Artificial Dynamics in GNNs

Dongqi Fu
UIUC

Zhe Xu
UIUC

Hanghang Tong
UIUC

Jingrui He
UIUC

{dongqif2, zhexu3, htong, jingrui}@illinois.edu

https://github.com/DongqiFu/Natural-and-Artificial-Dynamics-in-GNNs-A-Tutorial

Contents

- Part I – Introduction
- Part II – Natural Dynamics in GNNs
- 30 mins Coffee Break
- Part III – Artificial Dynamics in GNNs
- Part IV – Challenges and Future Directions
- Q&A

Basics of graph neural networks (GNNs)

- According to [1], the general formula of GNNs can be expressed as

message-passing: information aggregation among hidden representation vectors of neighbors

$\boldsymbol{h}_v^{(k)}$: is the hidden representation of node v at the k-th layer

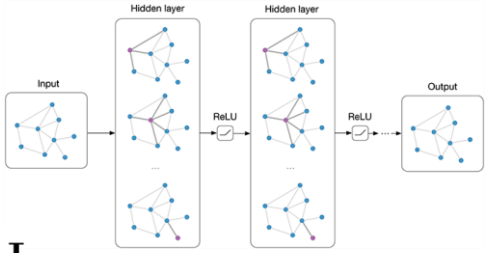
$$\mathbf{a}_v^{(k)} = AGGREGATE^{(k)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}),\ \mathbf{h}_v^{(k)} = COMBINE^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)})$$

- For example, the graph convolutional neural network (GCN) [2] can be instanced as

$$\mathbf{h}_v^{(k)} = ReLU(\mathbf{W}^{(k-1)} \cdot MEAN\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\})$$

with the original formula as

$$\mathbf{H}^{(k)} = ReLU(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}) \qquad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} \qquad \tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$$



[1] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka: How Powerful are Graph Neural Networks? ICLR 2019
[2] Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

GNNs have broad application domains

Computer Vision [1]

Natural Language Processing [2]

Recommender Systems [3]

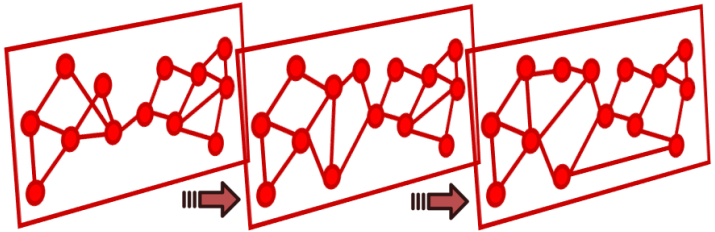
Drug Discovery [4]

image source: https://realpython.com/

[1] Chen et al.: A Survey on Graph Neural Networks and Graph Transformers in Computer Vision: A Task-Oriented Perspective. CoRR 2022
[2] Wu et al.: Graph Neural Networks for Natural Language Processing: A Survey. CoRR 2021
[3] Wang et al.: Graph Learning based Recommender Systems: A Review. IJCAI 2021
[4] Gaudelet et al.: Utilizing Graph Machine Learning within Drug Discovery and Development. Briefings in Bioinformatics 2021

What are natural dynamics?

- Natural dynamics in graphs [1]
  - Input graphs have the time-evolving components, e.g.,
    - Topology Structures
    - Node-level, edge-level, and (sub)graph-level features, etc.
  - Continuous Time
    - $\mathcal{G} = \{A, e = (i, j, t, +/-)\}$
  - Discrete Time
    - $\mathcal{G} = \{A^{(1)}, A^{(1)}, \ldots, A^{(T)}\}$

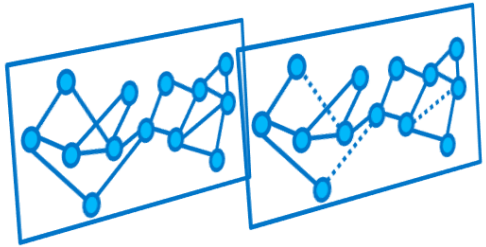


Evolving Graph Structures (Discrete Time Representation)

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022

What are artificial dynamics?

- Artificial dynamics in graphs [1], end-users
  - **change** (e.g., filter, mask, drop, or augment) **the existing** or
  - **construct the non-existing** graph-related elements, e.g.,
    - graph topology
    - node/graph attributes
    - GNN gradients, etc.
  - to realize the certain **performance upgrade**, e.g.,
    - decision accuracy
    - computation efficiency
    - model explanation, etc.



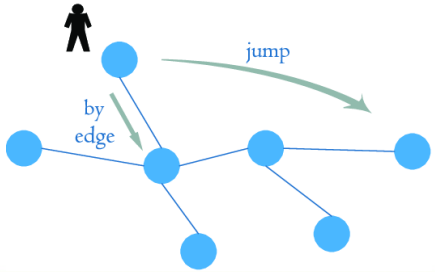
Random Edge Dropping ($A \rightarrow \bar{A}$)

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022

What are artificial dynamics?

- Artificial dynamics in graphs [1], end-users
  - **change** (e.g., filter, mask, drop, or augment) **the existing** or
  - **construct the non-existing** graph-related elements
  - to realize the certain **performance upgrade**

- In 2003, “artificial jump” [2] is proposed to adjust the graph topology for PageRank realizing the personal ranking function on graphs



[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022
[2] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, Gene H. Golub: Extrapolation methods for accelerating PageRank computations. WWW 2003

Relation between natural and artificial dynamics?

- For natural dynamics,
  - The input graph itself is a sequence of observations based on time
  - E.g., daily world wide web like Facebook, Twitter, etc.

- For artificial dynamics,
  - End-users deliberately modify the components for different interests
  - E.g., imperfect or redundant connections, missing features, etc.

- Can they be combined, i.e., **natural + artificial dynamics**?
  - **Yes**, when the input graph is temporal, and the modification is necessary

How natural dynamics contribute GNNs?

- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]

- Running?
- Dancing?
- Or just the static model for photography?

[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)

How natural dynamics contribute GNNs?

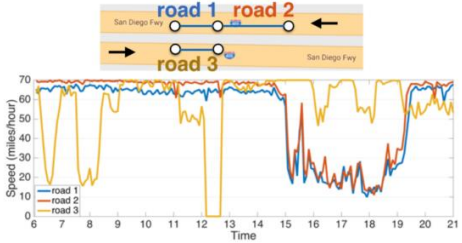
- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]

- Running?
- Dancing?
- Or just the static model for photography?

[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)

How natural dynamics contribute GNNs?

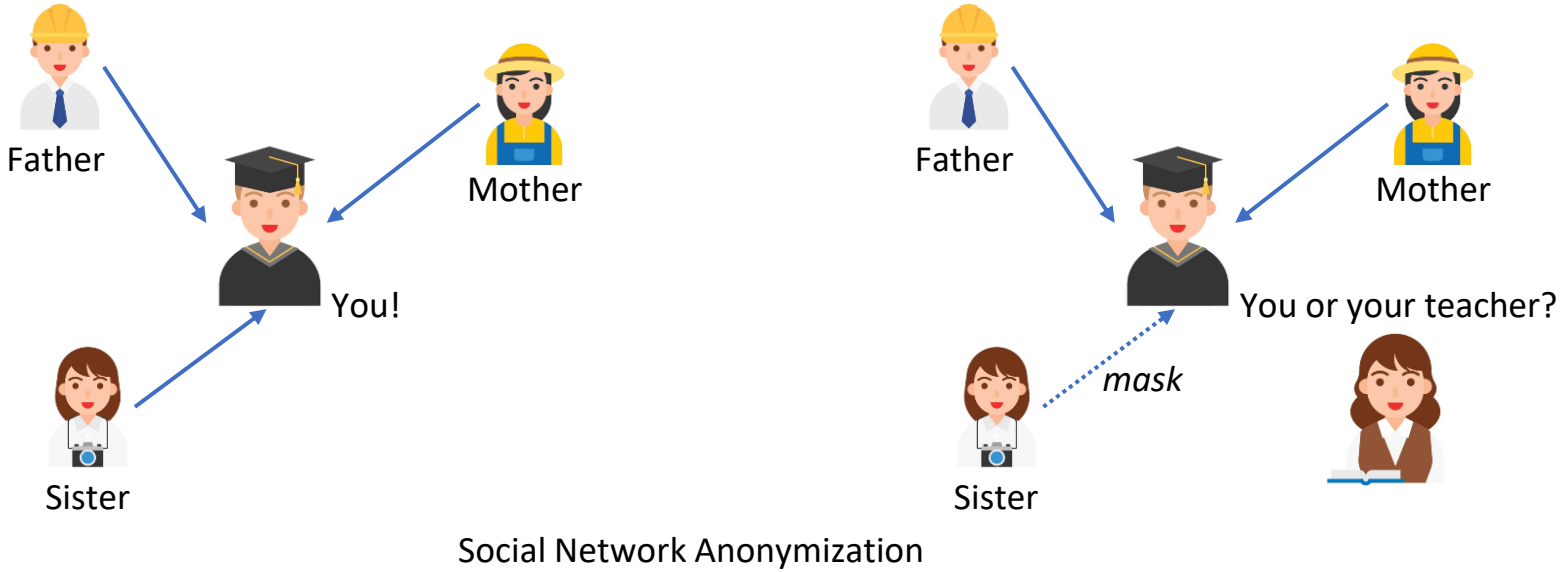
- Considering natural dynamics can help graph machine learning models to capture the temporal correlations among features [1]
  - Motion Recognition [2]
  - Time-Series Forecasting [3]
  - Pandemic Classification [4]
  - Social Network Analysis [5]
  - Many more …



[1] Kazemi et al.: Representation Learning for Dynamic Graphs: A Survey. JMLR (2020)
[2] Yan et al.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[3] Li et al.: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[4] Tsiotas et al.: The Effect of Anti-COVID-19 Policies on the Evolution of the Disease: A Complex Network Analysis of the Successful Case of Greece. Physics (2020)
[5] Aggarwal et al.: Evolutionary Network Analysis: A Survey. ACM Comput. Surv. (2014)

How artificial dynamics contribute GNNs?

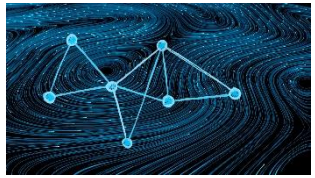
- Considering artificial dynamics can boost graph machine learning performance [1]



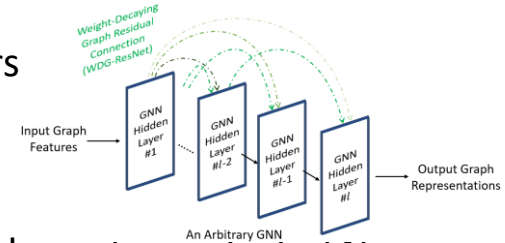
Social Network Anonymization

How artificial dynamics contribute GNNs?

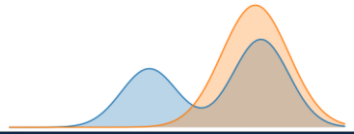
- Considering artificial dynamics can boost graph machine learning performance [1]
  - Privacy-Preserving [2]
    - **Permute the GNN gradients** under the differential privacy constraint



  - Decision Accuracy [3]
    - **Add dependency constraints on weight matrices** of GNN layers



  - Domain Adaption [4]
    - **Graph promoting** for large-scale pre-trained graph models on downstream tasks [4]

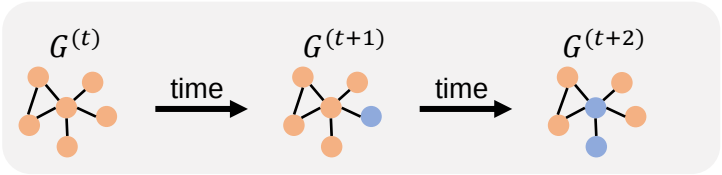


  - Many more …

[1] Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022
[2] Yang et al: Secure Deep Graph Generation with Link Differential Privacy. IJCAI 2021
[3] Zheng et al.: Deeper-GXX: Deepening Arbitrary GNNs. CoRR 2022
[4] Sun et al.:GPPT: Graph Pre-training and Prompt Tuning to Generalize Graph Neural Networks. KDD 2022

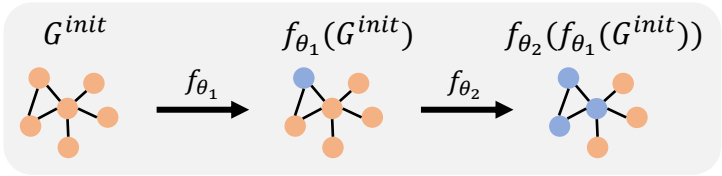
Scope of this tutorial

- Natural dynamics in GNNs
  - We focus on **time-evolving** graph **structures** and node **features**



Naturally Evolving Features

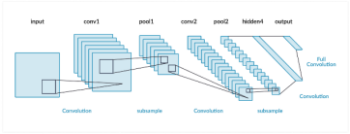
- Artificial dynamics in GNNs
  - We focus the **augmentation strategies** on graph **structures** and node **features**

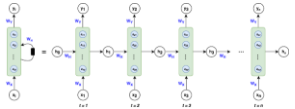


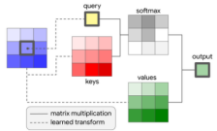
Artificially Evolving Features

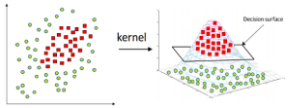
In this tutorial

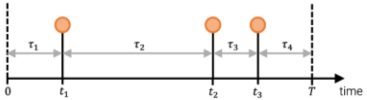
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process





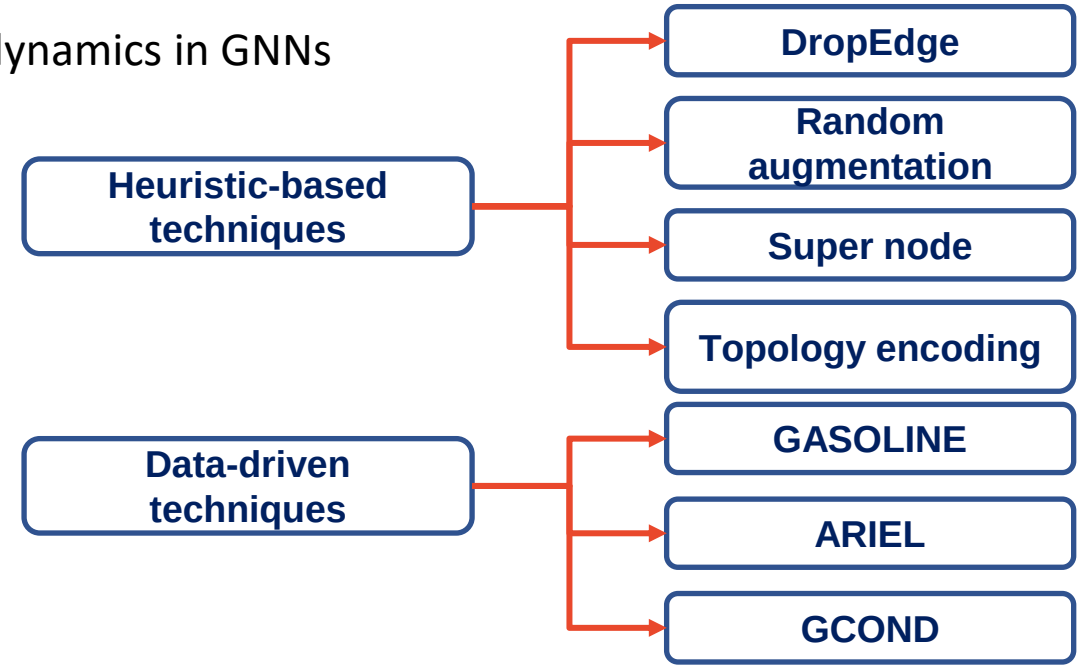






In this tutorial

- Covered works for artificial dynamics in GNNs
  - Heuristic-based Artificial Dynamics
  - Data-driven Artificial Dynamics

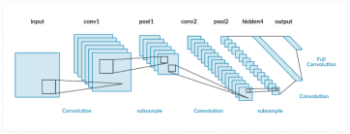


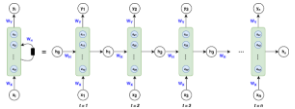
Contents

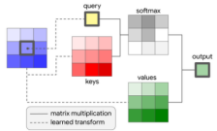
- Part I – Introduction ☑
- Part II – Natural Dynamics in GNNs
- 30 mins Coffee Break
- Part III – Artificial Dynamics in GNNs
- Part IV – Challenges and Future Directions
- Q&A

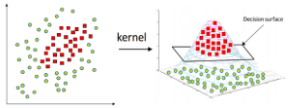
Roadmap for natural dynamics

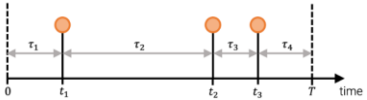
- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process











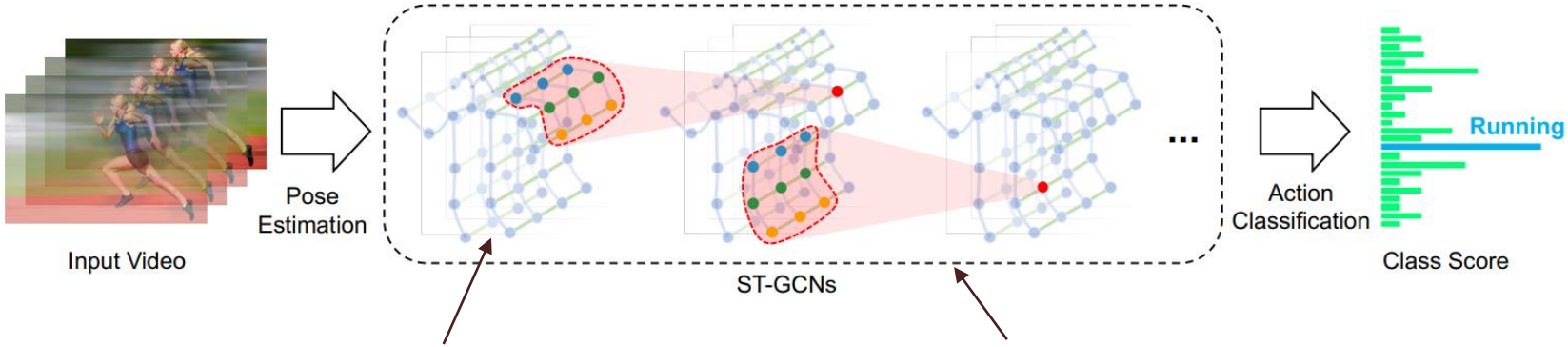
Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- **Task:** graph-level representation learning
- **Natural dynamic:** evolving structures and node features w.r.t time
- **Goal:** graph classification

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Problem Setting
  - Skeleton-based Action Reconstruction
    - or temporal graph classification in the graph research community



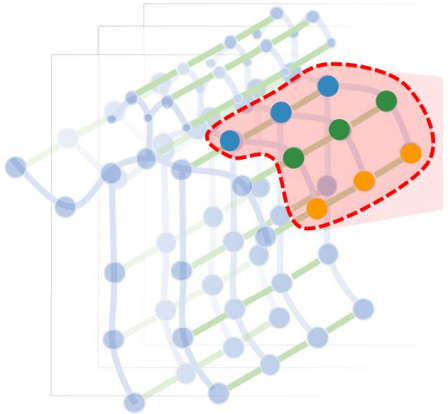
A Skeleton Sequence
(or a Spatial-Temporal Graph)

Going though the Spatial Graph Convolutional Network
to obtain its high-level feature map

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Graph Modeling
  - $G = (V, E)$ on a skeleton sequence with N joints and T timestamps featuring both intra-body and inter-frame connections
  - $V = \{v_{ti} \mid t = 1, \ldots, T, i = 1, \ldots, N\}$
    - the i-th joint at time t
  - $F(v_{ti})$: node feature, containing coordinate vector, estimation confidence, etc.
  - $E_S = \{v_{ti}v_{tj}\}$: human body joints
    - same t
  - $E_F = \{v_{ti}v_{(t+1)i}\}$: a particular joint i's trajectory over time



[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

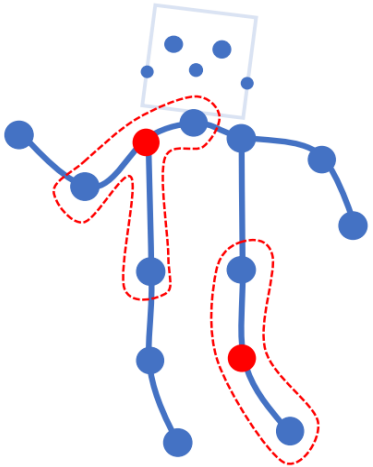
- Let's start from one single frame at t
  - Spatial Graph Convolutional Neural Network

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

the input feature of v_{tj}

normalizing term: how many number of nodes that are equivalent to v_{tj}, towards v_{ti}

neighbors of v_{ti} $B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\}$



  - which can be realized by GCN layer [2]

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- Then, let's consider multiple timestamps
  - Recall the Spatial Graph Convolutional Neural Network

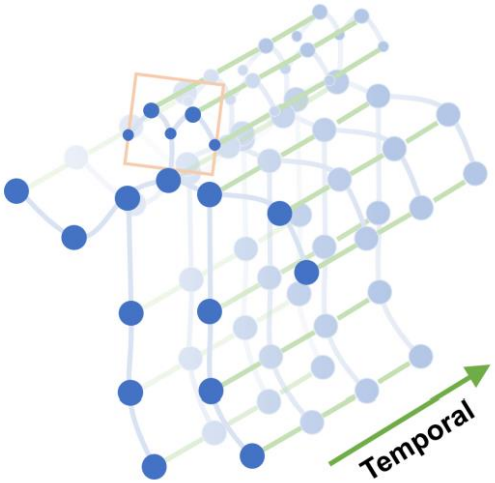
$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

$$B(v_{ti}) = \{v_{tj} | d(v_{tj}, v_{ti}) \leq D\}$$

  - For Spatial Temporal Graph Convolution
    - Spatial Temporal Modeling

$$B(v_{ti}) = \{v_{qj} | d(v_{tj}, v_{ti}) \leq K, |q - t| \leq \lfloor \Gamma/2 \rfloor\}$$

a hyperparameter controlling the time range



[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. ICLR 2017

Spatial Temporal Graph Convolutional Networks (ST-GCN) [1]

- A single frame shares the time, i.e.,

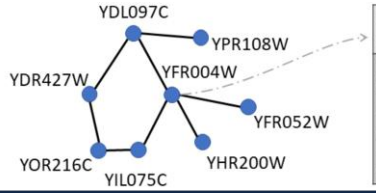
  $E_S = \{v_{ti}v_{tj}\}$: human body joints

  intra-snapshot edges

$$f_{out}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(v_{ti}, v_{tj})$$

spatial graph convolution for a snapshot

- Is that possible that they have different timestamps?
  - In [2], each dynamic protein-protein interaction network has 36 continuous observations (i.e., **36 edge timestamps**)
  - every 12 observations compose a metabolic cycle (i.e., **3 snapshot timestamps**), and each cycle reflects 25 mins in the real world.



Systematic Name: YFR004W
Standard Name: RPN11

Feature Type: ORF, Verified
Description: Metalloprotease subunit of 19S regulatory particle; part of 26S proteasome lid; couples the deubiquitination and degradation of proteasome substrates;

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Dongqi Fu, Jingrui He: DPPIN: A Biological Repository of Dynamic Protein-Protein Interaction Network Data. IEEE Big Data 2022

Given multiple timestamps (e.g., edge timestamps and snapshot timestamps) in a temporal graph

- RQ1: How to integrate the multiple evolution patterns?

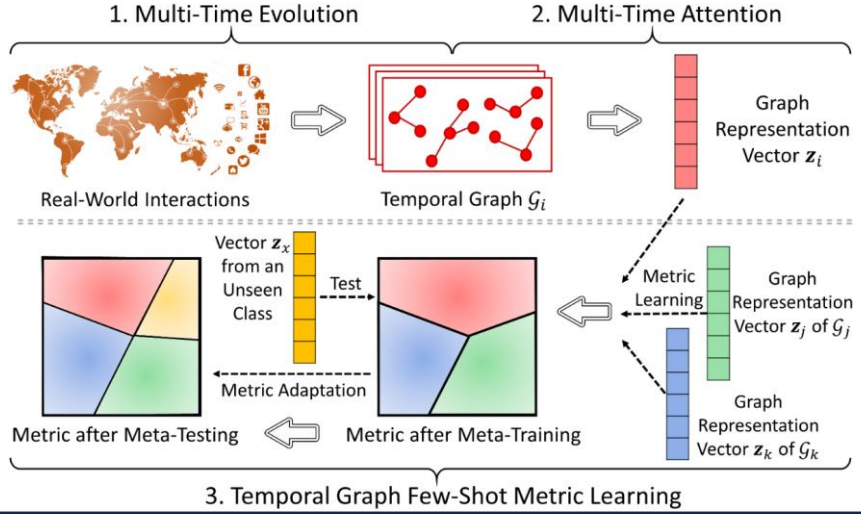
- RQ2: How to encode them for an embedding for temporal graph classification? What evolutions are dominating the graph similarity?

- RQ3: Labeling graph (especially temporal) is costly, how could we leverage fewer labels but effectively?

[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018
[2] Dongqi Fu, Jingrui He: DPPIN: A Biological Repository of Dynamic Protein-Protein Interaction Network Data. IEEE Big Data 2022

Facing above research questions, in Temp-GFSM [1]

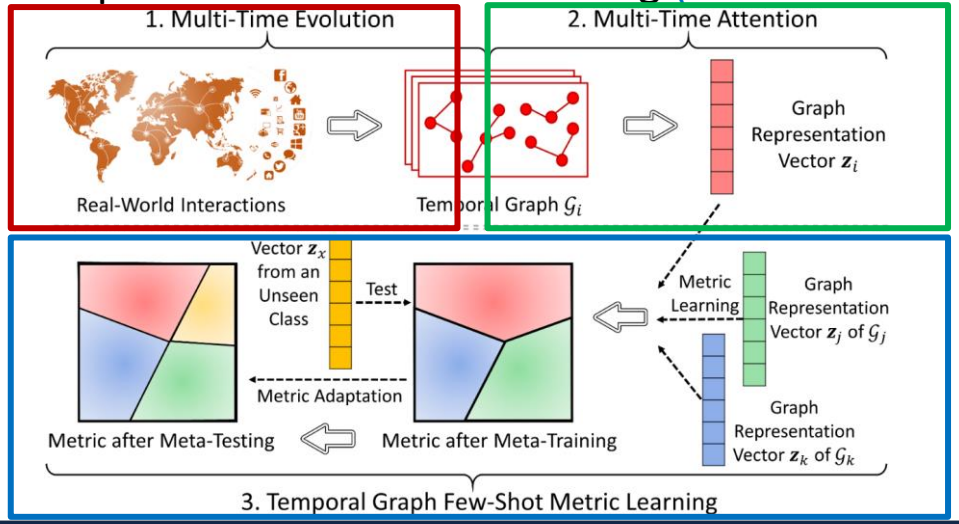
- Multi-Time Evolution
- Multi-Time Attention
- Temporal Graph Few-Shot Metric Learning



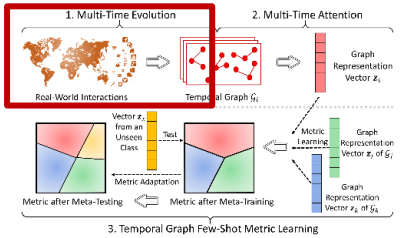
[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022

In Temp-GFSM [1]

- Multi-Time Evolution (Carrying Multiple Dynamics)
- Multi-Time Attention (Weighting Multiple Dynamics)
- Temporal Graph Few-Shot Metric Learning (New Class Adaption)

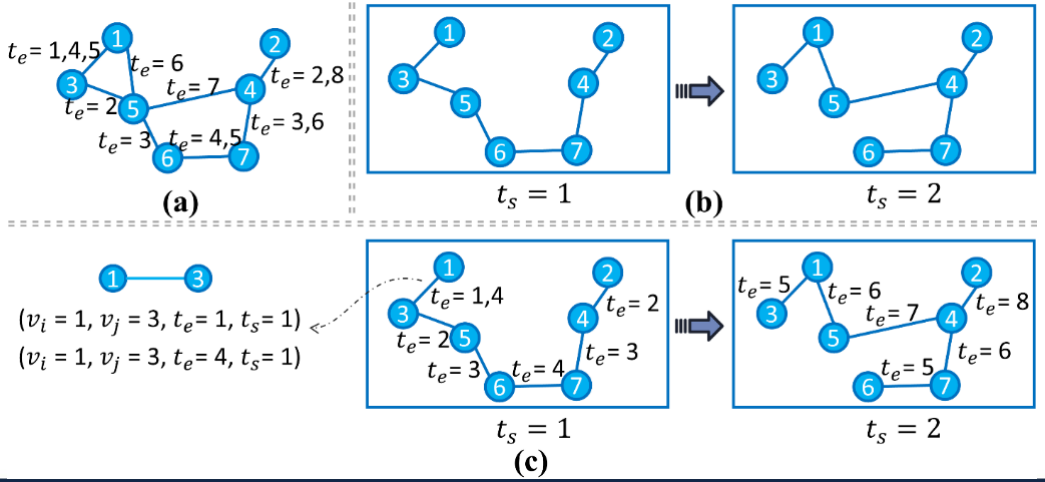


[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022

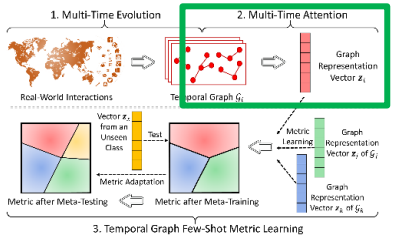


In Multi-Time Evolution of Temp-GFSM [1]

- An edge is marked as quadruplet (v_i, v_j, t_e, t_s), where
  - (v_i, v_j, t_e) means the connection between v_i and v_j exists at time t_e
  - (v_i, v_j, t_e, t_s) means the event (v_i, v_j, t_e) happens in snapshot S^{t_s}

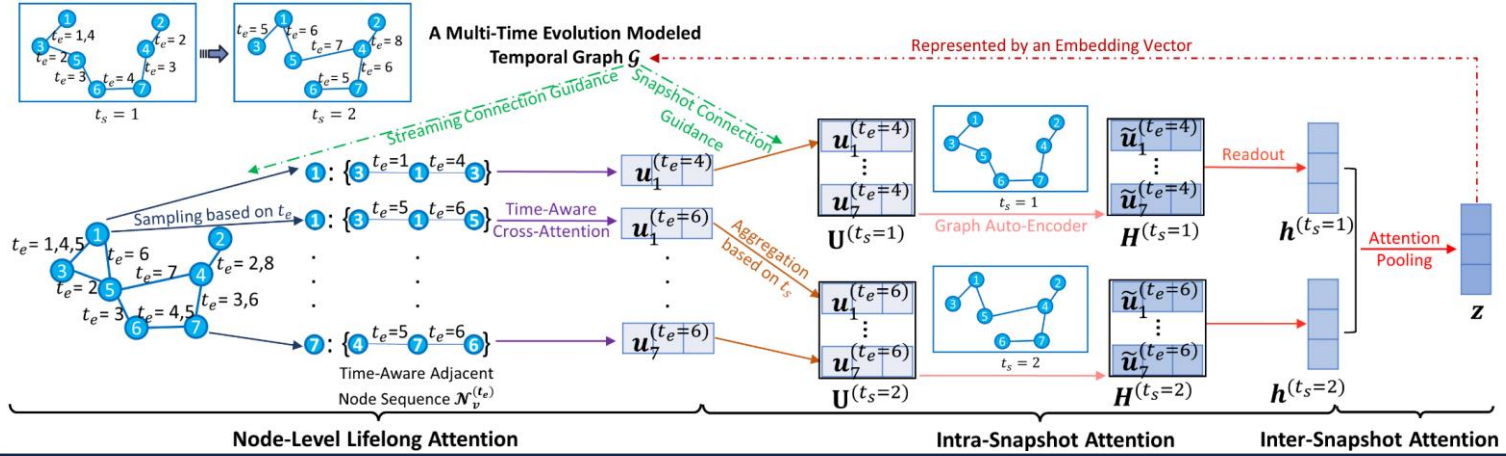


[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022



In Multi-Time Attention of Temp-GFSM [1]

- Temporal graph G -> representation vector Z
  - Node-Level Lifelong Attention (Select Meaningful Words)
  - Intra-Snapshot Attention (Compose Supportive Sentences)
  - Inter-Snapshot Attention (Finish a Fluent Article with Paragraphs)



[1] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, Jingrui He: Meta-Learned Metrics over Multi-Evolution Temporal Graphs. KDD 2022

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving node features w.r.t time
- **Goal:** node feature prediction

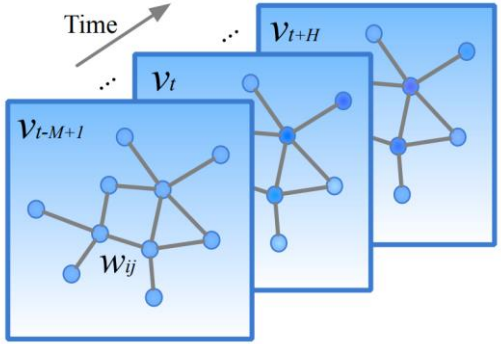
[1] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- A Deep Learning Framework for Traffic Forecasting

- Problem Setting
  - Traffic Flow Prediction

given past volumes,
predict future volumes,
with the latent structure



$$\arg\max \quad \log P(v_{t+1}, \ldots, v_{t+H} | v_{t-M+1}, \ldots, v_t)$$

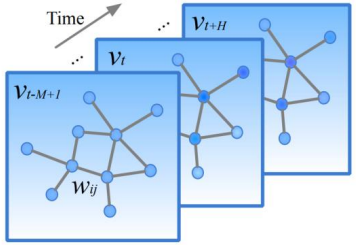
- $v_t \in \mathbb{R}^n$: an observation of n road segments (n nodes in graph), e.g., volume or density

- $w \in \mathbb{R}^{n \times n}$: adjacency matrix of road networks, the shared structure over t

[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- Extract Spatial Features
  - Similar to previous ST-GCN [2], backbone is GCN



- Extract Temporal Features
  - Other than directly calling GCN on the catenation of temporal features, $v^l = \{v_{t-M+1}, \ldots, v_t\}$, involve a time convolution on the time series as below

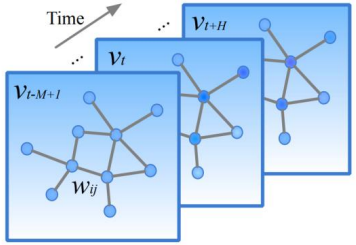
$$v^{l+1} = \Gamma_1^l *_{\mathcal{T}} \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

$v_t \in \mathbb{R}^n$: an observation of n road segments (n nodes in graph), e.g., volume or density

[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018
[2] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- Extract Spatial Features
  - Similar to previous ST-GCN [2], backbone is GCN



- Extract Temporal Features
  - Other than directly calling GCN on the catenation of temporal features, $v^l = \{v_{t-M+1}, \ldots, v_t\}$, involve a time convolution on the time series as below

$$v^{l+1} = \Gamma_1^l *_{\mathcal{T}} \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

l: is the index of unit block (or layer) of STGCN, i.e., $v^l \rightarrow v^{l+1}$

pass a fully-connected output layer to readout v_{t+1}

graph convolution, i.e., GCN

time convolution, (details next page)

v^l: stacking $v_{t-M+1}, \ldots, v_t$

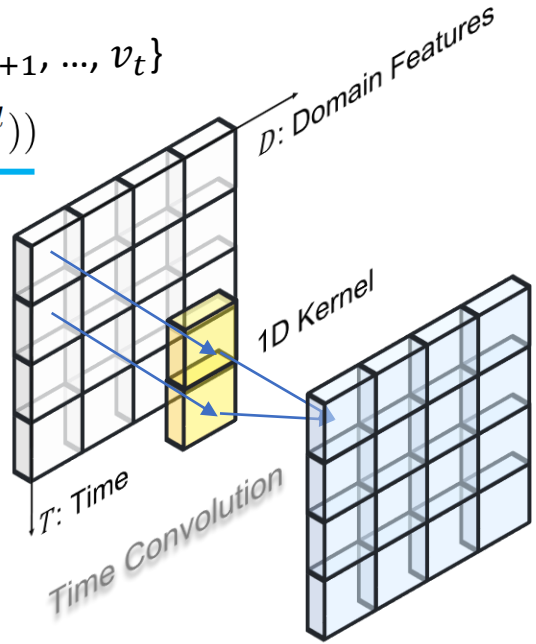
[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018
[2] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

Spatio-Temporal Graph Convolutional Networks (STGCN) [1]

- Extract Temporal Features
  - Other than directly call GCN on the catenation $v^l = \{v_{t-M+1}, \ldots, v_t\}$

$$\mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (v^l)) \quad \text{VS} \quad \mathrm{ReLU}(\Theta^l *_{\mathcal{G}} (\Gamma_0^l *_{\mathcal{T}} v^l))$$

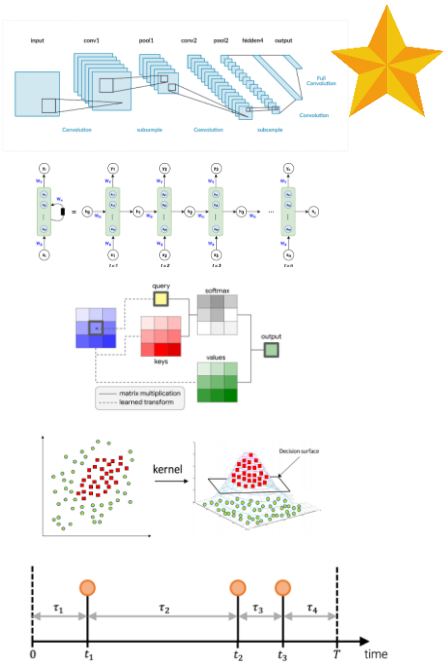
  - A 1-D kernel along the time axis
    - Could aggregate temporal neighbors to **capture temporal behaviors** of features (e.g., traffic flows), especially for long-term time-series



[1] Bing Yu, Haoteng Yin, Zhanxing Zhu: Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. IJCAI 2018
[2] Sijie Yan, Yuanjun Xiong, Dahua Lin: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI 2018

In this tutorial

- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process



Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

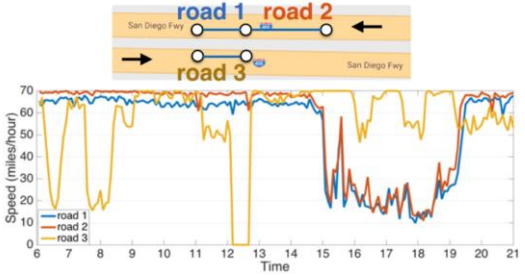
- **Task:** node-level representation learning
- **Natural dynamic:** evolving node features w.r.t time
- **Goal:** node feature prediction

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Problem Definition
  - Let's still focus on Traffic Forecasting

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



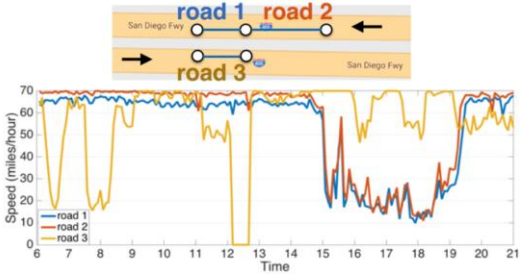
  - But the differences from the previous discussed STGCN are:
    - What if the latent graph structure is directed?
    - How can be deal with time information other than time convolution, e.g., how to take time information recurrently?

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (1) Stationary distribution of the diffusion process

$$\boldsymbol{\mathcal{P}} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$



  - (2) Diffusion Convolution

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1}\left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k + \theta_{k,2}\left(\boldsymbol{D}_I^{-1}\boldsymbol{W}^\intercal\right)^k\right)\boldsymbol{X}_{:,p} \quad \text{for } p \in \{1,\cdots,P\}$$

  - (3) Diffusion Convolution Layer

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \qquad \text{for } q \in \{1,\cdots,Q\}$$

let's step into each one of those

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (1) Stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph

$$\boldsymbol{\mathcal{P}} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (1) Stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph

D_O: out-degree diagonal matrix

W: weighted adjacency matrix

$$\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k$$

k: random walk steps

$\mathcal{P} \in \mathbb{R}^{N\times N}$: whose i-th row represents the likelihood of diffusion (i.e., personalized PageRank vector) from node i

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^\intercal \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

feature matrix, N is the num of node, P is the node feature dimension

learnable parameter, i.e., weight matrices

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^\intercal \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (2) Diffusion Convolution **over** a graph signal $\boldsymbol{X} \in \mathbb{R}^{N \times P}$ and a filter f_θ is defined as

feature matrix, N is the num of node, P is the node feature dimension

learnable parameter, i.e., weight matrices

out-degree based diffusion

in-degree based diffusion

$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1} \boldsymbol{W} \right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1} \boldsymbol{W}^\intercal \right)^k \right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

two sets of parameters from f_θ

feature matrix

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (3) Diffusion Convolution Layer that maps P-dimensional features to Q-dimensional outputs

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \qquad \text{for } q \in \{1, \cdots, Q\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- For directed graph structure
  - (3) Diffusion Convolution Layer that maps P-dimensional features to Q-dimensional outputs

$\boldsymbol{X} \in \mathbb{R}^{N \times P}$: input

$\boldsymbol{H} \in \mathbb{R}^{N \times Q}$: output

$$\boldsymbol{H}_{:,q} = \boldsymbol{a}\left(\sum_{p=1}^{P} \boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\Theta}_{q,p,:,:}}\right) \quad \text{for } q \in \{1, \cdots, Q\}$$

activation function (e.g., ReLU, Sigmoid)

indexing for in-degree or out-degree diffusion

indexing for k

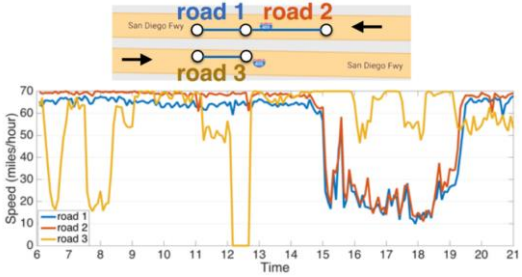
$$\boldsymbol{X}_{:,p} \star_{\mathcal{G}} f_{\boldsymbol{\theta}} = \sum_{k=0}^{K-1} \left(\theta_{k,1} \left(\boldsymbol{D}_O^{-1}\boldsymbol{W}\right)^k + \theta_{k,2} \left(\boldsymbol{D}_I^{-1}\boldsymbol{W}^\intercal\right)^k\right) \boldsymbol{X}_{:,p} \quad \text{for } p \in \{1, \cdots, P\}$$

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- After for directed graph structure,
  - (1) Stationary distribution of the diffusion process
  - (2) Diffusion Convolution
  - (3) Diffusion Convolution Layer

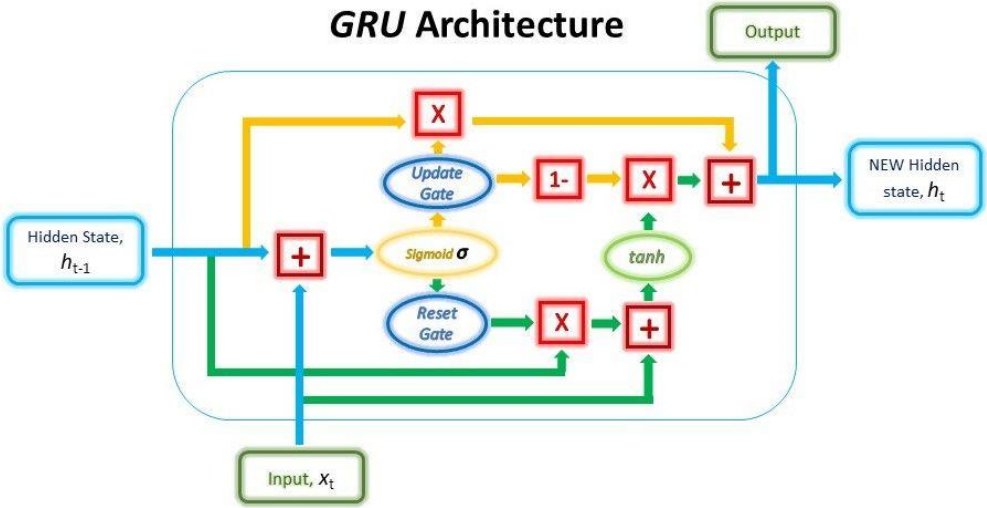
- How to take time information recurrently?
  - Temporal Dynamics Modeling [1]



[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

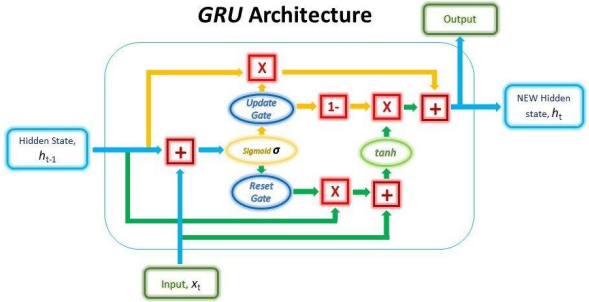
- Temporal Dynamics Modeling
  - Adopt the logic from Gated Recurrent Units (GRU)[2], i.e., make GRU take structured information
  - Input
  - Reset Gate
  - Update Gate
  - Output



[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR (2014)

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Temporal Dynamics Modeling
  - Adopt the logic from Gated Recurrent Units (GRU)[2], i.e., make GRU take structured information



- Input $\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}$
- Reset Gate $\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{\Theta}_r \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}] + \boldsymbol{b}_r)$
- Update Gate $\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{\Theta}_u \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ \boldsymbol{H}^{(t-1)}] + \boldsymbol{b}_u)$
- Output $\boldsymbol{H}^{(t)} = \boldsymbol{u}^{(t)} \odot \boldsymbol{H}^{(t-1)} + (1 - \boldsymbol{u}^{(t)}) \odot \boldsymbol{C}^{(t)}$

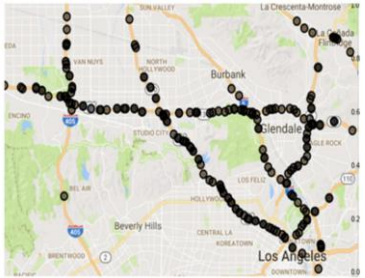
diffusion convolution w. different weight parameters

$$\boldsymbol{C}^{(t)} = \tanh(\boldsymbol{\Theta}_C \star_{\mathcal{G}} [\boldsymbol{X}^{(t)},\ (\boldsymbol{r}^{(t)} \odot \boldsymbol{H}^{(t-1)})] + \boldsymbol{b}_c)$$

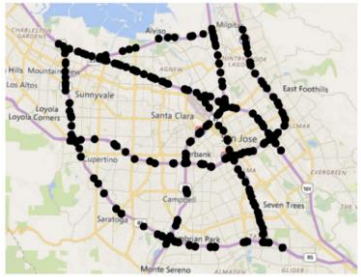
[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR (2014)

Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Experiments
  - Datasets
    - METR-LA: Highway of Los Angeles
      - 207 sensors (traffic speed)
      - May 1st 2012 to Jun 30th 2012
    - PEMS-BAY: Highway in Bay Area of California
      - 325 sensors (traffic speed)
      - Jan 1st 2017 to May 31th 2017



(a) METR-LA



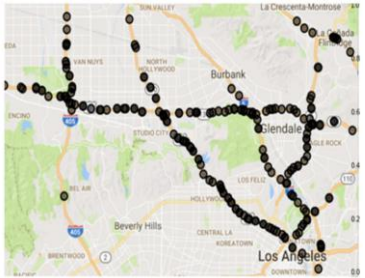
(b) PEMS-BAY

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

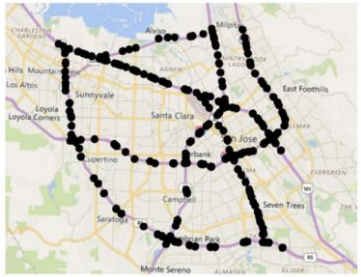
Diffusion Convolutional Recurrent Neural Network (DCRNN) [1]

- Experiments
  - Performance (traffic speed forecasting)

	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%**
PEMS-BAY	15 min	MAE	2.88	1.62	1.74	1.85	2.20	2.05	**1.38**
		RMSE	5.59	3.30	3.16	3.59	4.42	4.19	**2.95**
		MAPE	6.8%	3.5%	3.6%	3.8%	5.19%	4.8%	**2.9%**
	30 min	MAE	2.88	2.33	2.32	2.48	2.30	2.20	**1.74**
		RMSE	5.59	4.76	4.25	5.18	4.63	4.55	**3.97**
		MAPE	6.8%	5.4%	5.0%	5.5%	5.43%	5.2%	**3.9%**
	1 hour	MAE	2.88	3.38	2.93	3.28	2.46	2.37	**2.07**
		RMSE	5.59	6.50	5.44	7.08	4.98	4.96	**4.74**
		MAPE	6.8%	8.3%	6.5%	8.0%	5.89%	5.7%	**4.9%**



(a) METR-LA



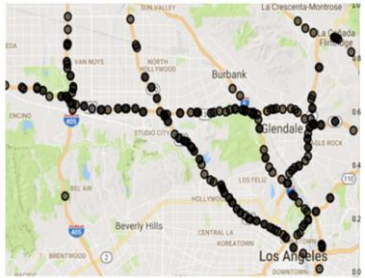
(b) PEMS-BAY

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

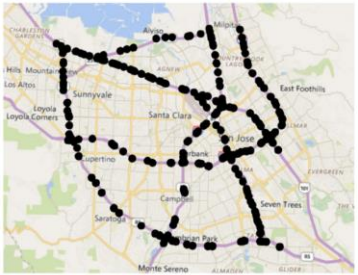
The latent graph structure may be not easy to observe

- In DCRNN[1], the adjacency is hand-crafted

$$W_{ij} = \exp\left(-\frac{\mathrm{dist}(v_i, v_j)^2}{\sigma^2}\right)$$



(a) METR-LA



(b) PEMS-BAY

- Could we find another way to extract that latent structure?

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018

Could we find another way to extract that latent structure?

- Discrete Graph Structure Learning for Forecasting Multiple Time Series (GTS) [2]
  - Focusing on the same problem (i.e., traffic forecasting) and the same diffusion convolution structure as DCRNN [1]

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

  - But set the adjacency matrix as a variable to learn

$$A_{ij} = \text{sigmoid}((\log(\theta_{ij}/(1-\theta_{ij})) + (g_{ij}^1 - g_{ij}^2))/s)$$

temperature

samples from a given Gumbel distribution

$$\theta_{ij} = \text{FC}(\text{FC}(z^i \| z^j))$$

$$z^i = \text{FC}(\text{vec}(\text{Conv}(X^i)))$$

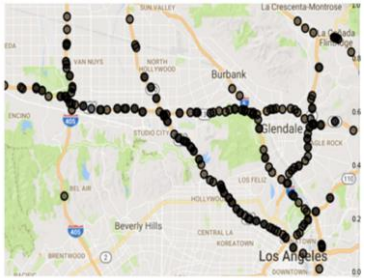
X^i: the i-th node over all features and timestamps

[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021

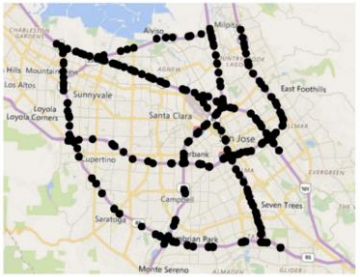
In the same datasets with [1], i.e., METR-LA and PEMS-BAY

- Experiments
  - Performance (traffic speed forecasting)

	Metric	FNN	LSTM	DCRNN	LDS	NRI	GTSv	GTS
15 min	MAE ($\times 10^{-3}$)	1.23	1.02	0.71	0.49	0.66	0.26	**0.24**
	RMSE ($\times 10^{-2}$)	1.28	1.63	1.42	1.26	0.27	0.20	**0.19**
	MAPE	0.20%	0.21%	0.09%	0.07%	0.14%	0.05%	**0.04%**
30 min	MAE ($\times 10^{-3}$)	1.42	1.11	1.08	0.81	0.71	0.31	**0.30**
	RMSE ($\times 10^{-2}$)	1.81	2.06	1.91	1.79	0.30	0.23	**0.22**
	MAPE	0.23%	0.20%	0.15%	0.12%	0.15%	**0.05%**	**0.05%**
60 min	MAE ($\times 10^{-3}$)	1.88	1.79	1.78	1.45	0.83	**0.39**	0.41
	RMSE ($\times 10^{-2}$)	2.58	2.75	2.65	2.54	0.46	0.32	**0.30**
	MAPE	0.29%	0.27%	0.24%	0.22%	0.17%	**0.07%**	**0.07%**



(a) METR-LA

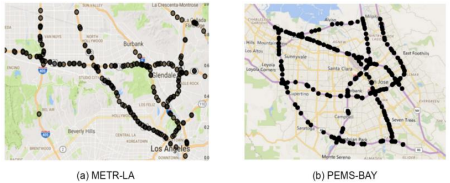


(b) PEMS-BAY

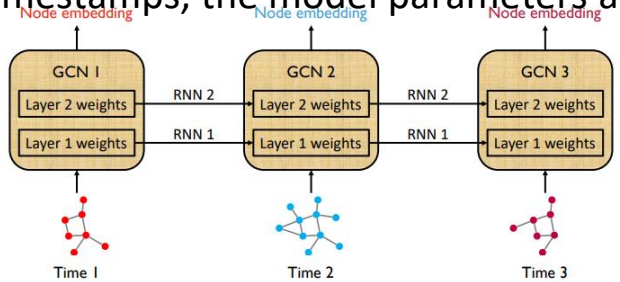
[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021

In DCRNN [1] or GTS [2],

- The adjacency is fixed with evolving node features
- What if the adjacency is also evolving?



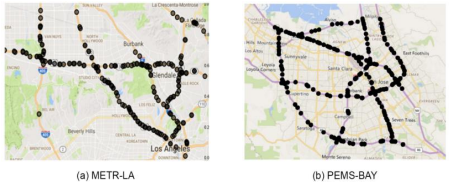
- For evolving structures and features
  - EvolveGCN [3] is proposed to adapt model parameters, i.e.,
    - Each time has its own GCN model with its $A^{(t)}$ and $H^{(t)}$
    - Cross timestamps, the model parameters are dependent, e.g., $W_t^{(l)} = LSTM(W_{t-1}^{(l)})$



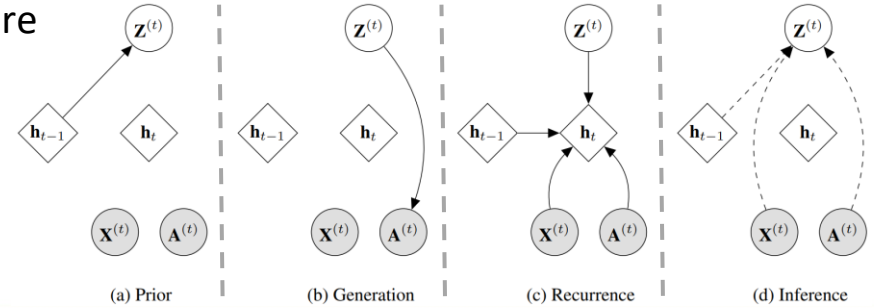
[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021
[3] Pareja et al.: EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. AAAI 2020

In DCRNN [1] or GTS [2],

- The adjacency is fixed with evolving node features
- ~~What if the adjacency is also evolving?~~
- Can we also predict the future adjacency?



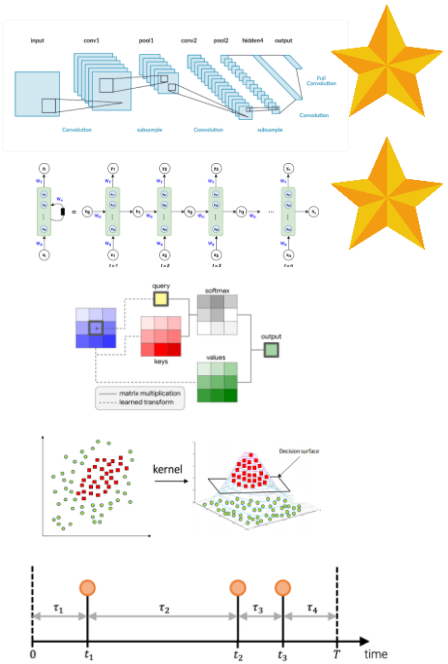
- For evolving structures and features
  - VGRNN [3] is proposed to learn the variational posterior distribution of evolving adjacency structures together, in the RNN structure



[1] Yaguang Li, Rose Yu, Cyrus Shahabi, Yan Liu: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. ICLR 2018
[2] Chao Shang, Jie Chen, Jinbo Bi: Discrete Graph Structure Learning for Forecasting Multiple Time Series. ICLR 2021
[3] Hajiramezanali et al.: Variational Graph Recurrent Neural Networks. NeurIPS 2019

In this tutorial

- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process



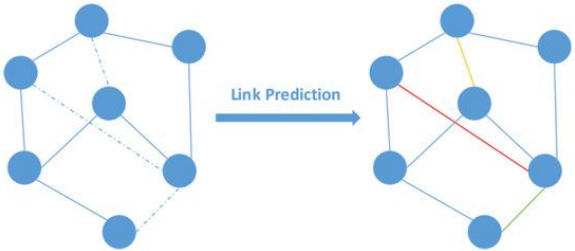
DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

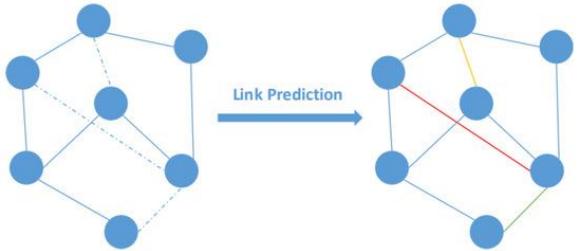
- Problem Definition (Link Prediction)
  - Given a series of graph snapshots $\{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$, and $\mathcal{G}^t = (\boldsymbol{A}^t, \boldsymbol{X}^t)$, DySAT [1] aims to learn the node representation e_v^t for each node v at timestamps $t = \{1, 2, \ldots, T\}$
  - Then, the latest time e_v^T and e_u^T are used to decide if there is an edge links node v and node u at time T+1



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural Self-Attention
  - Apply attention in one single timestamp
  - Topological neighbors
- Temporal Self-Attention
  - Apply attention across timestamps
  - Temporal neighbors



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural Self-Attention
  - At a single timestamp t, *the superscript of t is omitted in the following equation*

$\boldsymbol{z}_v$: hidden representation of node v after structural attention

a_{uv}: attention weight

$$z_v = \sigma\Big(\sum_{u \in \mathcal{N}_v} \alpha_{uv} \boldsymbol{W}^s \boldsymbol{x}_u\Big), \quad \alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{w \in \mathcal{N}_v} \exp(e_{wv})}$$

A_{uv}: adjacency

neighbors in $\boldsymbol{A}^t$

$$e_{uv} = \sigma\Big(A_{uv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_u || \boldsymbol{W}^s \boldsymbol{x}_v]\Big) \quad \forall (u, v) \in \mathcal{E}$$

$\boldsymbol{a}$: learnable weight matrix (e.g., MLP)

$\boldsymbol{W}^s$: projection weight matrix for structural attention

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
  - Who are temporal neighbors for a certain node?
  - Temporal neighbors for node v consist of its **historical behaviors**

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
  - Temporal neighbors for node v consist of its historical behaviors

Node v's time embedding, $\mathbb{R}^{T\times F'}$

$\mathbb{R}^{T\times T}$ $\mathbb{R}^{T\times D'}$ $\mathbb{R}^{D'\times F'}$

$$Z_v = \boldsymbol{\beta_v}(X_v W_v), \qquad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^{T}\exp(e_v^{ik})},$$

attention weights, i and j are two timestamps

X_v: concatenation of $x_v^1, x_v^2, \ldots, x_v^T$

$$e_v^{ij} = \left(\frac{((X_v W_q)(X_v W_k)^T)_{ij}}{\sqrt{F'}} + M_{ij}\right)$$

F': normalization factor

A matrix enforcing auto-regressive manner, details next page

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Temporal Self-Attention
  - Temporal neighbors for node v are its historical behaviors

$$Z_v = \boldsymbol{\beta_v}(X_v W_v), \qquad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^{T} \exp(e_v^{ik})},$$

$$e_v^{ij} = \left(\frac{((X_v W_q)(X_v W_k)^T)_{ij}}{\sqrt{F'}} + M_{ij} \right)$$

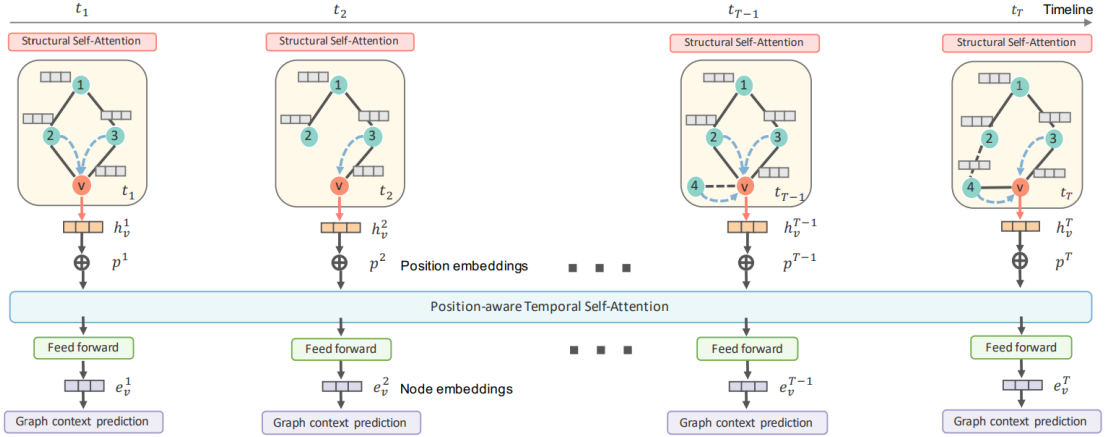
$$M_{ij} = \begin{cases} 0, & i \le j \\ -\infty, & \text{otherwise} \end{cases}$$

when $M_{ij} = -\infty$, $\beta_v^{ij} = 0$, which switches off the attention from timestamp i to j

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

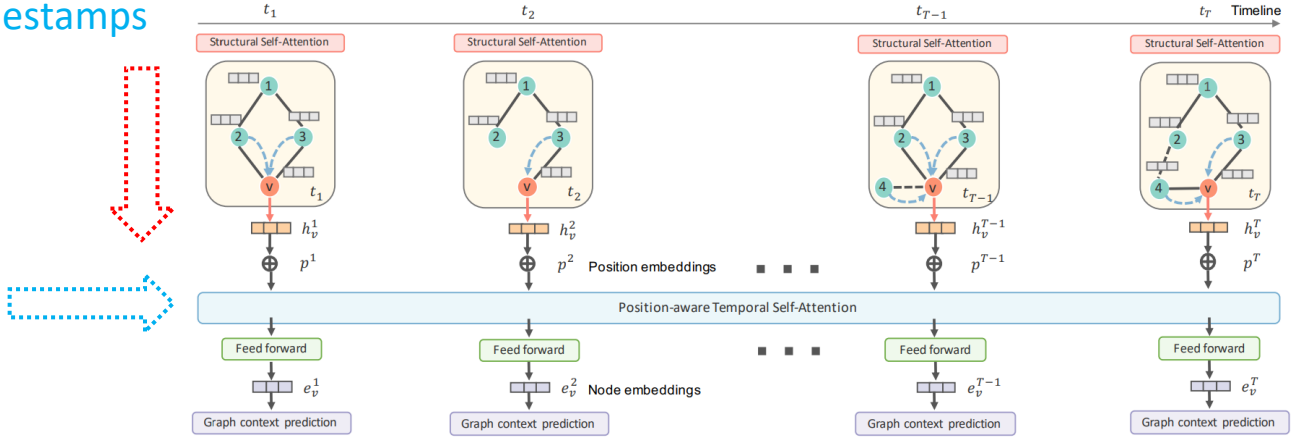
- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp t
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

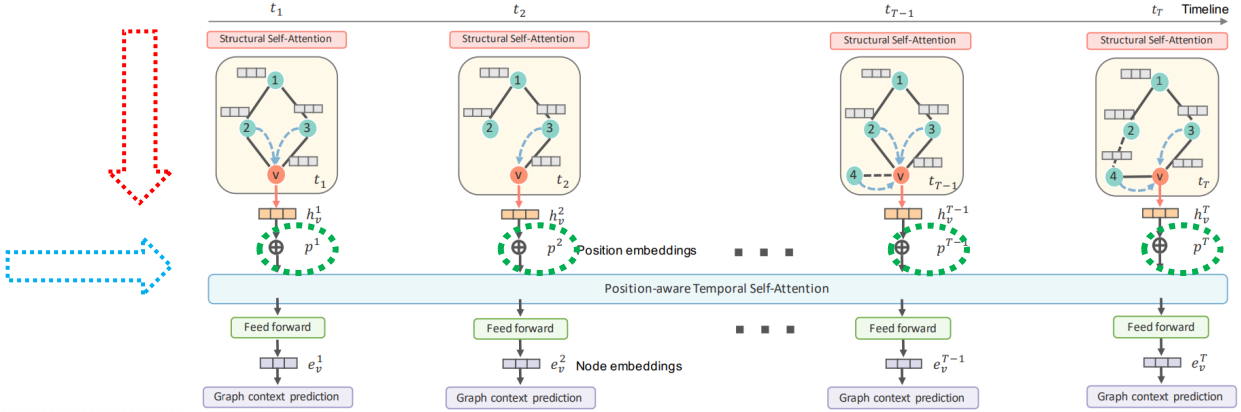
- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp *t*
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural + Temporal Self-Attention
  - Obtain structural encoding independently at each timestamp *t*
  - Then, temporal self-attention take the structural encoding as input to attend over timestamps + positional encoding

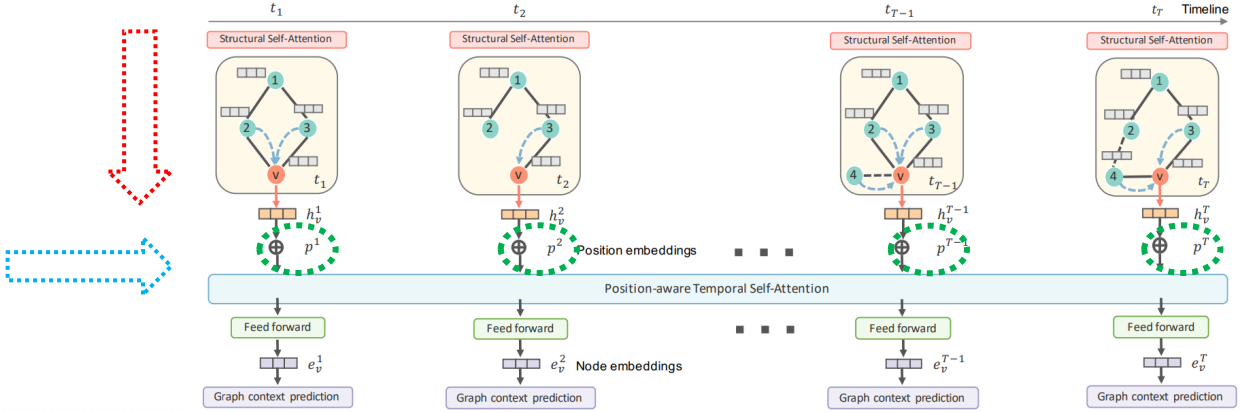


[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Structural + Temporal Self-Attention
  - Add the absolute temporal position of each snapshot

$$\boldsymbol{h}_v^1 + \boldsymbol{p}^1, \boldsymbol{h}_v^2 + \boldsymbol{p}^2, \ldots, \boldsymbol{h}_v^T + \boldsymbol{p}^T$$



[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention [1]

- Positional Encoding

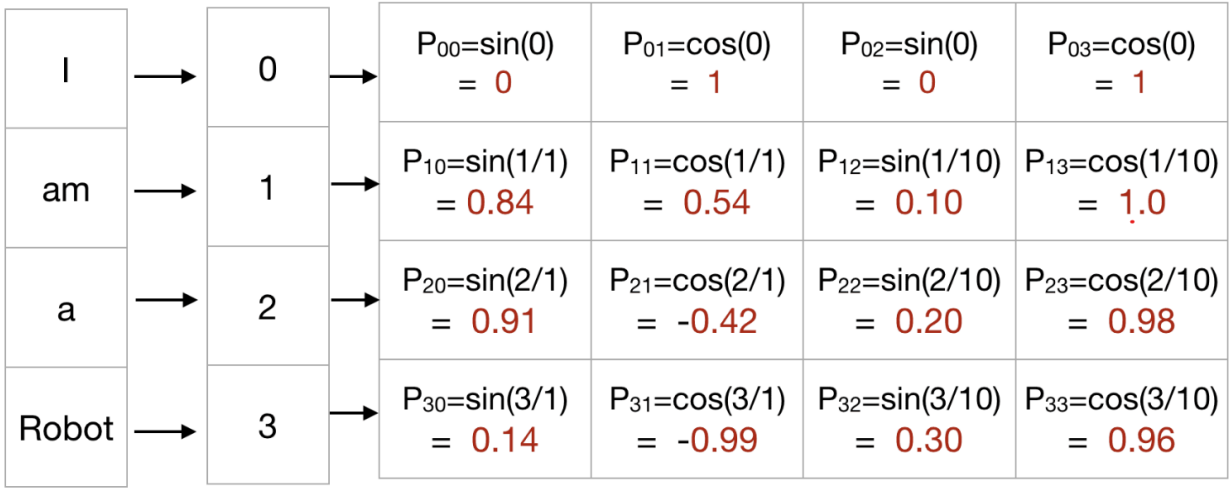
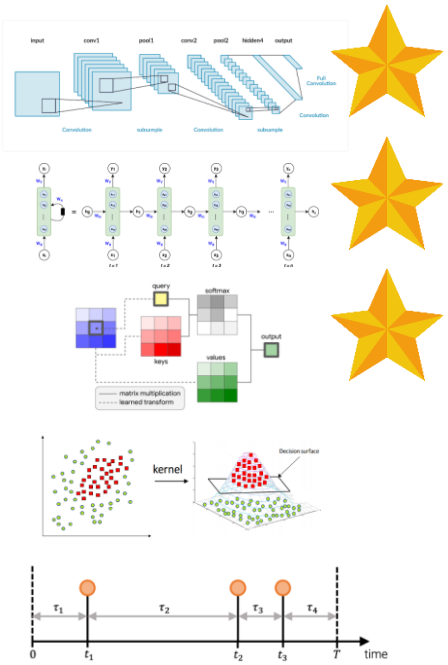


image source: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

[1] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

In this tutorial

- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process



Back to the positional encoding

I	→ 0	$P_{00}=\sin(0)=0$	$P_{01}=\cos(0)=1$	$P_{02}=\sin(0)=0$	$P_{03}=\cos(0)=1$
am	→ 1	$P_{10}=\sin(1/1)=0.84$	$P_{11}=\cos(1/1)=0.54$	$P_{12}=\sin(1/10)=0.10$	$P_{13}=\cos(1/10)=1.0$
a	→ 2	$P_{20}=\sin(2/1)=0.91$	$P_{21}=\cos(2/1)=-0.42$	$P_{22}=\sin(2/10)=0.20$	$P_{23}=\cos(2/10)=0.98$
Robot	→ 3	$P_{30}=\sin(3/1)=0.14$	$P_{31}=\cos(3/1)=-0.99$	$P_{32}=\sin(3/10)=0.30$	$P_{33}=\cos(3/10)=0.96$

- Do we have other options?
  - A concurrent method [1] with DySAT [2] proposes the time kernel function to record the time features

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020
[2] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, Hao Yang: DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. WSDM 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction, node classification

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- In [1], **kernel function** is proposed to map time t to a finite dimensional representation vector

$$t \mapsto \Phi_d(t) := \sqrt{\frac{1}{d}}\left[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)\right]$$

$\omega_1, \omega_2, \ldots, \omega_d$ are learnable parameters

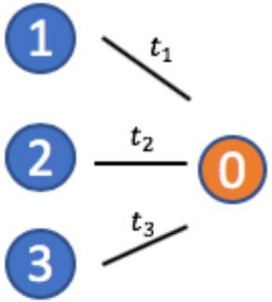
[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- Suppose there is a target node v_0 at time t, which needs to attend over its spatial-temporal neighbors

$$\mathcal{N}(v_0; t) = \{v_1, \ldots, v_N\}$$

- For each node v_i, v_0 connects with it previously at a time t_i, i.e., $t_i < t$



[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- With $\mathcal{N}(v_0; t) = \{v_1, \ldots, v_N\},\ t_i < t$

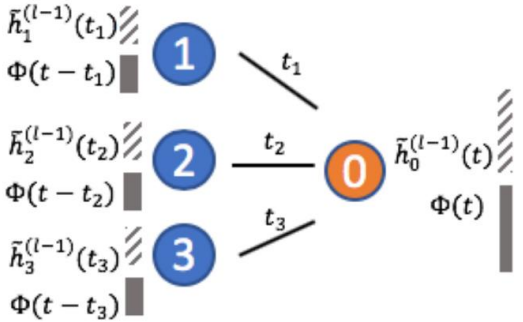
- First, append the position encoding by time kernel functions, to form $\boldsymbol{Z}(t)$

$$\mathbf{Z}(t) = \left[\tilde{\mathbf{h}}_0^{(l-1)}(t)||\Phi_{d_T}(0), \tilde{\mathbf{h}}_1^{(l-1)}(t_1)||\Phi_{d_T}(t-t_1), \ldots, \tilde{\mathbf{h}}_N^{(l-1)}(t_N)||\Phi_{d_T}(t-t_N)\right]^\intercal$$

$\boldsymbol{Z}(t)$ is the intermediate step of getting the node embedding $\tilde{h}_0^{(l)}$ of v_0 at the l-th layer of TGAT

node embedding of v_0 at the (l-1)-th layer of TGAT, the 0-th layer is the initial input feature

time kernel function



[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

Inductive Representation Learning on Temporal Graphs (TGAT) [1]

- With $\mathcal{N}(v_0; t) = \{v_1, \ldots, v_N\},\ t_i < t$

- Append the position encoding by time kernel functions

$$\mathbf{Z}(t) = \left[\tilde{\mathbf{h}}_0^{(l-1)}(t)||\Phi_{d_T}(0), \tilde{\mathbf{h}}_1^{(l-1)}(t_1)||\Phi_{d_T}(t-t_1), \ldots, \tilde{\mathbf{h}}_N^{(l-1)}(t_N)||\Phi_{d_T}(t-t_N)\right]^\intercal$$

- Self-Attention

$$\mathbf{q}(t) = [\mathbf{Z}(t)]_0 \mathbf{W}_Q$$
$$\mathbf{K}(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_K \qquad \mathbf{h}(t) = \mathrm{Attn}\big(\mathbf{q}(t), \mathbf{K}(t), \mathbf{V}(t)\big)$$
$$\mathbf{V}(t) = [\mathbf{Z}(t)]_{1:N} \mathbf{W}_V$$

- Readout

$$\tilde{\mathbf{h}}_0^{(l)}(t) = \mathrm{FFN}\Big(\mathbf{h}(t)||\mathbf{x}_0\Big) \equiv \mathrm{ReLU}\Big([\mathbf{h}(t)||\mathbf{x}_0]\mathbf{W}_0^{(l)} + \mathbf{b}_0^{(l)}\Big)\mathbf{W}_1^{(l)} + \mathbf{b}_1^{(l)}$$

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Inductive representation learning on temporal graphs. ICLR 2020

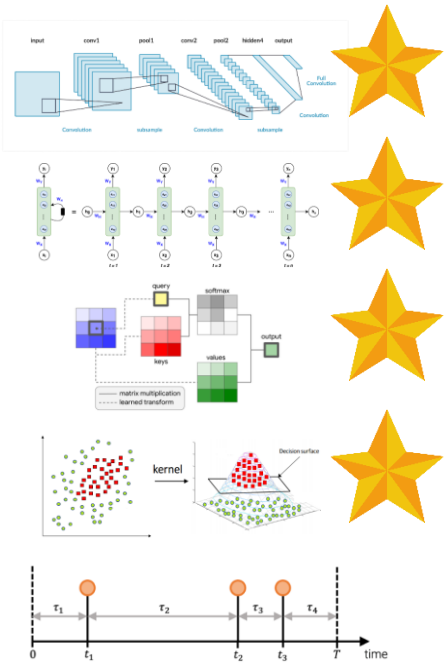
Other Time Kernel Functions [1]

Feature maps specified by $[\phi_{2i}(t), \phi_{2i+1}(t)]$	Origin	Parameters	Interpretations of ω
$\left[\cos\left(\omega_i(\mu)t\right), \sin\left(\omega_i(\mu)t\right)\right]$	Bochner's	μ: location-scale parameters specified for the *reparametrization trick*.	$\omega_i(\mu)$: converts the i^{th} sample (drawn from auxiliary distribution) to target distribution under location-scale parameter μ.
$\left[\cos\left(g_\theta(\omega_i)t\right), \sin\left(g_\theta(\omega_i)t\right)\right]$	Bochner's	θ: parameters for the inverse CDF $F^{-1} = g_\theta$.	ω_i: the i^{th} sample drawn from the auxiliary distribution.
$[\cos(\tilde{\omega}_i t), \sin(\tilde{\omega}_i t)]$	Bochner's	$\{\tilde{\omega}\}_{i=1}^{d}$: transformed samples under non-parametric inverse CDF transformation.	$\tilde{\omega}_i$: the i^{th} sample of the underlying distribution $p(\omega)$ in Bochner's Theorem.
$\left[\sqrt{c_{2i,k}}\cos(\omega_j t), \sqrt{c_{2i+1,k}}\sin(\omega_j t)\right]$	Mercer's	$\{c_{i,k}\}_{i=1}^{2d}$: the Fourier coefficients of corresponding $\mathcal{K}_{\omega_j}$, for $j = 1, \ldots, k$.	ω_j: the frequency for kernel function $\mathcal{K}_{\omega_j}$ (can be parameters).

[1] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, Kannan Achan: Self-attention with Functional Time Representation Learning. NeurIPS 2019

In this tutorial

- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process



DyRep: Learning Representations over Dynamic Graphs [1]

- **Task:** node-level representation learning
- **Natural dynamic:** evolving graph structures and node features w.r.t time
- **Goal:** link prediction

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - A user is tweeting, they tweeted at time t_1= 8:00 am, t_2=10:00 am, t_3= 11:00 am, what is t_4 = ?

  - TPP is a model that could fit the process of t_1, t_2, and t_3 to predict t_4

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

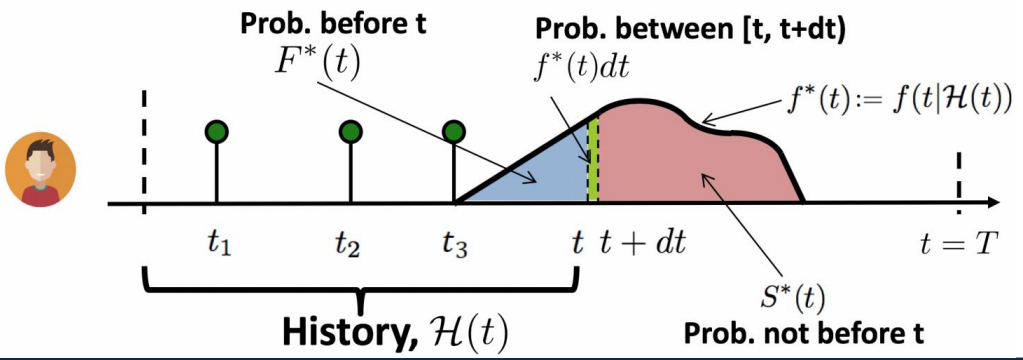
DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Given the history of events $\mathcal{H}(t) = \{t_1, \ldots, t_{i-1}\}$, we need to model,
    - A conditional probability density function $f^* = f(t|\mathcal{H}(t))$, which is the conditional probability that the next event t will occur during the interval $[t, t+dt)$
    - A cumulative distribution function $F^*(t) = F(t|\mathcal{H}(t)) = \int_{t_{i-1}}^{t} f^*(\tau)d\tau$, which is the conditional probability that the next event will occur before t
    - A complementary of $F^*(t)$, $S^*(t) = S(t|\mathcal{H}(t)) = 1 - F^*(t)$, the conditional probability that the next event will not occur before time t

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Given the history $\mathcal{H}(t) = \{t_1, \ldots, t_{i-1}\}$, we need to model,
    - $f^*(t) = f(t|\mathcal{H}(t))$: next event t will occur during the interval $[t, t+dt)$
    - $F^*(t) = F(t|\mathcal{H}(t)) = \int_{t_{i-1}}^{t} f^*(\tau)d\tau$: next event will occur before t
    - $S^*(t) = S(t|\mathcal{H}(t)) = 1 - F^*(t)$: next event will not occur before time t

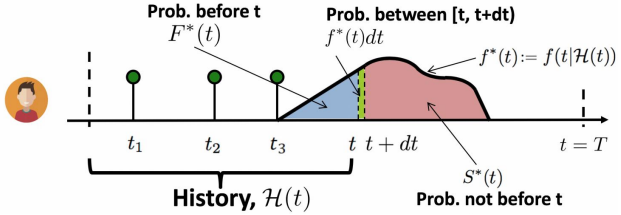


[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - The conditional intensity function $\lambda^*(t) = \lambda(t|\mathcal{H}(t))$, i.e., the conditional probability that the next event will happened during $[t, t+dt)$, is defined as follows

$$\lambda^*(t)dt = \frac{f^*(t)dt}{S^*(t)}$$



  - $\lambda^*(t)$ can be also understood as the instantaneous rate of events per time of unit, e.g., $\lambda^*(t)$ = 10 tweets/minute

  - Using the form of $\lambda^*(t)$ also contributes to TPP model parameterization and model reusability [2]

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Temporal Point Process (TPP) [2]
  - Different forms of functions model the intensity function $\lambda^*(t)$, e.g.,
    - Homogeneous Poisson process
      - $\lambda^*(t) = \mu \geq 0$ 10 tweets/minute
    - Inhomogeneous Poisson process
      - $\lambda^*(t) = g_\theta(t) \geq 0$ 2 tweets/@8:35am, 25 tweets/@2:58pm, …
    - Hawkes process
      - $\lambda^*(t) = \mu + \alpha \sum_{t_i \in \mathcal{H}(t)} \kappa_\omega(t - t_i),\ \kappa_\omega(t) = \exp(-\omega t)$
  - The parameters are obtained by fitting the model with the observation and maximizing the log-likelihood

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019
[2] Upadhyay et al., Temporal Point Processes: https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Each **edge connection** is considering as an **event** (u, v, t)
  - We want to predict whether a node u and a node v will connect at time t, given node u's history and node v's history before t

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Intensity functions for graphs, i.e., an edge connection between nodes u and v

$$\lambda^{u,v}(t) = f(g^{u,v}(\bar{t}))$$

$\bar{t}$ means the timestamp just before the current event

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - Intensity functions for graphs, i.e., an edge connection between nodes u and v

$$\lambda^{u,v}(t) = f(g^{u,v}(\bar{t}))$$

the outer function $f()$ is a softplus function with trainable a parameter to make sure positive output

$$f(x) = \psi \log(1 + \exp(x/\psi))$$

the inner function $g()$ computes the compatibility of the catenation

$$g^{u,v}(\bar{t}) = \boldsymbol{\omega}\,[z^u(\bar{t}); z^v(\bar{t})]$$

  - Now, the question is how to get the node embeddings, e.g., $z^v(\bar{t})$?

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Model Temporal Point Process (TPP) for Graphs
  - How to get node embeddings, e.g., $z^v(\bar{t})$?
    - Self-Propagation: w.r.t its historical behavior
    - Exogeneous Drive: for the smooth update of the current
    - Localized Embedding Propagation: message passing within second-order proximity
  - Suppose node u and node v participating in any type of event at time t
    - E.g., for the p-th event of node v at time t

$$\mathbf{z}^v(t_p) = \sigma\Big(\underbrace{\mathbf{W}^{struct}\mathbf{h}^u_{struct}(\bar{t_p})}_{\text{Localized Embedding Propagation}} + \underbrace{\mathbf{W}^{rec}\mathbf{z}^v(\bar{t_p^v})}_{\text{Self-Propagation}} + \underbrace{\mathbf{W}^t(t_p - \bar{t_p^v})}_{\text{Exogenous Drive}} \Big)$$

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

DyRep: Learning Representations over Dynamic Graphs [1]

- Suppose node u and node v participating in any type of event at time t
  - For the p-th event of node v at time t

$$\mathbf{z}^v(t_p) = \sigma(\underbrace{\mathbf{W}^{struct}\mathbf{h}^u_{struct}(\bar{t_p})}_{\text{Localized Embedding Propagation}} + \underbrace{\mathbf{W}^{rec}\mathbf{z}^v(\bar{t^v_p})}_{\text{Self-Propagation}} + \underbrace{\mathbf{W}^t(t_p - \bar{t^v_p})}_{\text{Exogenous Drive}})$$

h^u_{struc} is the aggregation on node u's neighbors

$$h^u_{struc}(\bar{t}) = \max(\{\sigma\left(q_{ui}(t) \cdot h^i(\bar{t})\right),\ \forall i \in N_u(\bar{t})\})$$

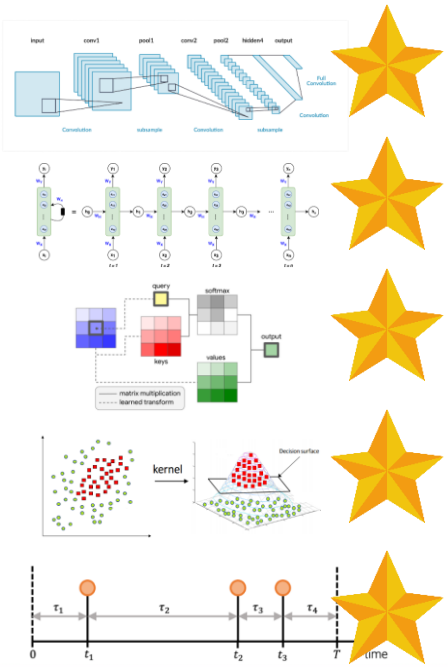
$q_{ui}(t)$ can be understood as the weight of the connection

$$h^i(\bar{t}) = \mathbf{W}z^i(\bar{t}) + \boldsymbol{b}$$

[1] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, Hongyuan Zha: DyRep: Learning Representations over Dynamic Graphs. ICLR 2019

In this tutorial

- Covered works for natural dynamics in GNNs
  - Temporal GNNs with Convolutional Operations
  - Temporal GNNs with Recurrent Units
  - Temporal GNNs with Time Attention
  - Temporal GNNs with Time Kernel
  - Temporal GNNs with Temporal Point Process



There are also many great research works on natural dynamics

- Ma et al. **Streaming Graph Neural Networks**. SIGIR 2020
- Rossi et al. **Temporal Graph Networks for Deep Learning on Dynamic Graphs**. CoRR (2020)
- Tian et al. **Self-supervised Representation Learning on Dynamic Graphs**. CIKM 2021
- Fu et al. **SDG: A Simplified and Dynamic Graph Neural Network**. SIGIR 2021
- You et al. **ROLAND: Graph Learning Framework for Dynamic Graphs**. KDD 2022
- Cong et al. **Do We Really Need Complicated Model Architectures For Temporal Networks?** ICLR 2023
- Many more……

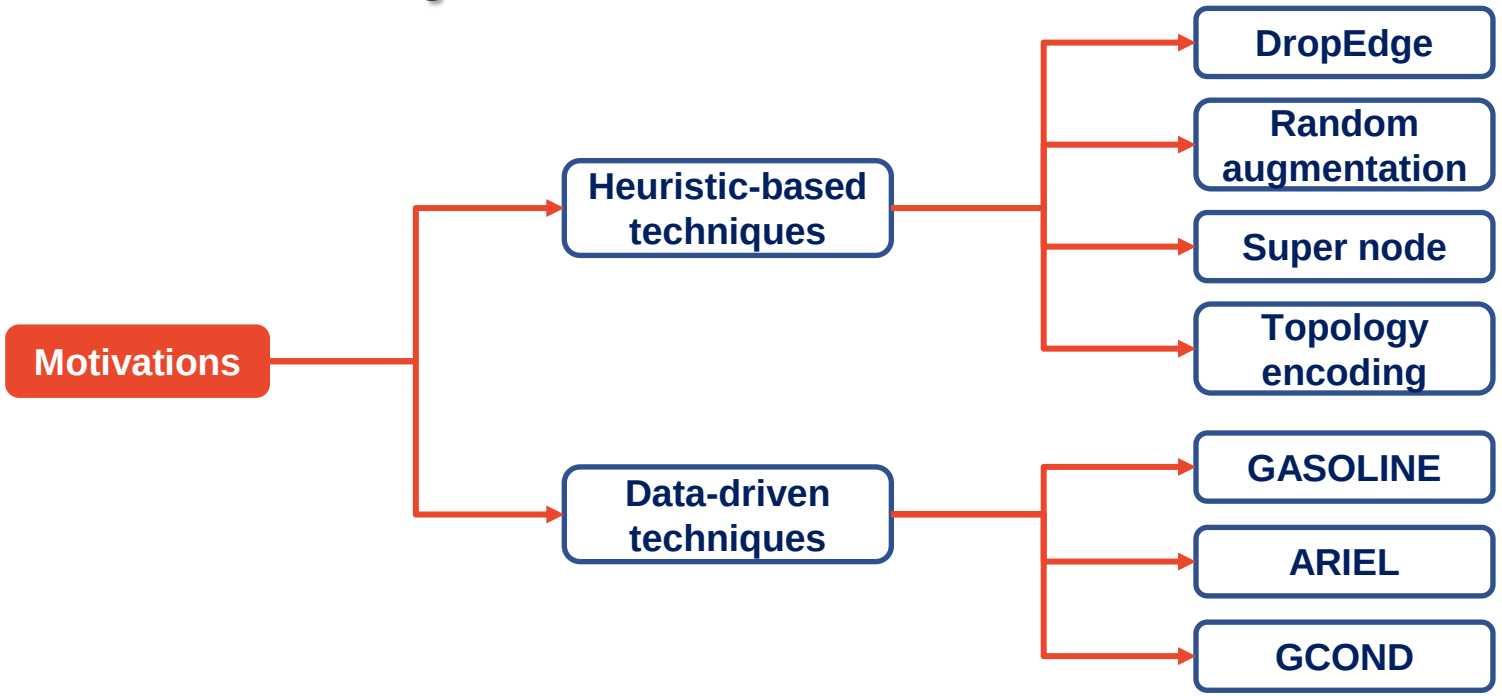
Contents

- Part I – Introduction ☑
- Part II – Natural Dynamics in GNNs ☑
- 30 mins Coffee Break
- Part III – Artificial Dynamics in GNNs
- Part IV – Challenges and Future Directions
- Q&A

Contents

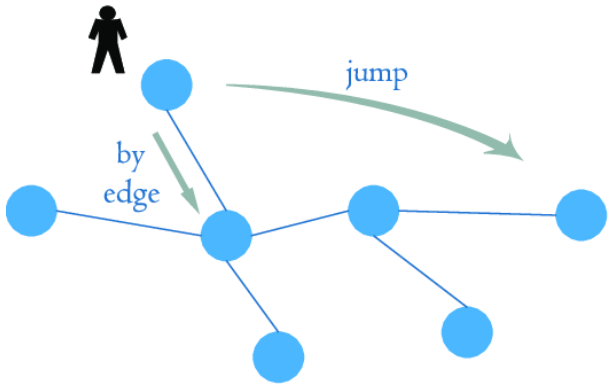
- Part I – Introduction ☑
- Part II – Natural Dynamics in GNNs ☑
- 30 mins Coffee Break ☑
- Part III – Artificial Dynamics in GNNs
- Part IV – Challenges and Future Directions
- Q&A

Artificial dynamics in GNNs



Artificial dynamics in GNNs – definition

- **"Artificial jump"** [1] is proposed to adjust the graph topology for PageRank realizing the personal ranking function on graphs.
- Artificial dynamics in the **GNN** context: modifying the given graph(s) for specific functionality.
- A.k.a. graph data augmentation (for GNNs) [2,3].



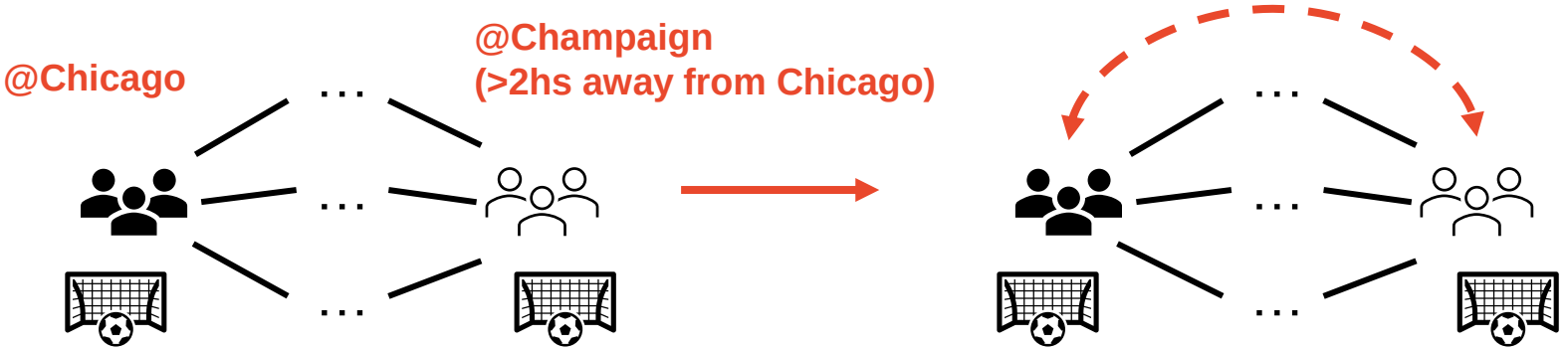
[1] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, Gene H. Golub: Extrapolation methods for accelerating PageRank computations. WWW 2003
[2] Ding, Kaize, et al. "Data augmentation for deep graph learning: A survey." ACM SIGKDD Explorations Newsletter 24.2 (2022): 61-77.
[3] Zhao, Tong, et al. "Graph data augmentation for graph machine learning: A survey." arXiv preprint arXiv:2202.08871 (2022).

Artificial dynamics in GNNs – motivation (1)

The given graph(s) could be **noisy and poisoned.**
Example 1: An academic literature classification system

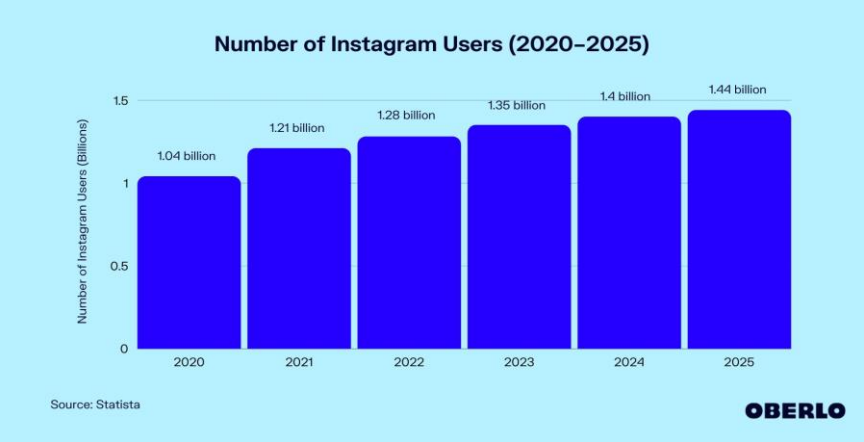
Artificial dynamics in GNNs – motivation (2)

The given graph(s) could be **not informative.**
Example 2: Friend recommendation on social networks



Artificial dynamics in GNNs – motivation (3)

The given graph(s) could be **huge.**
Example 3: representation learning on real-world networks



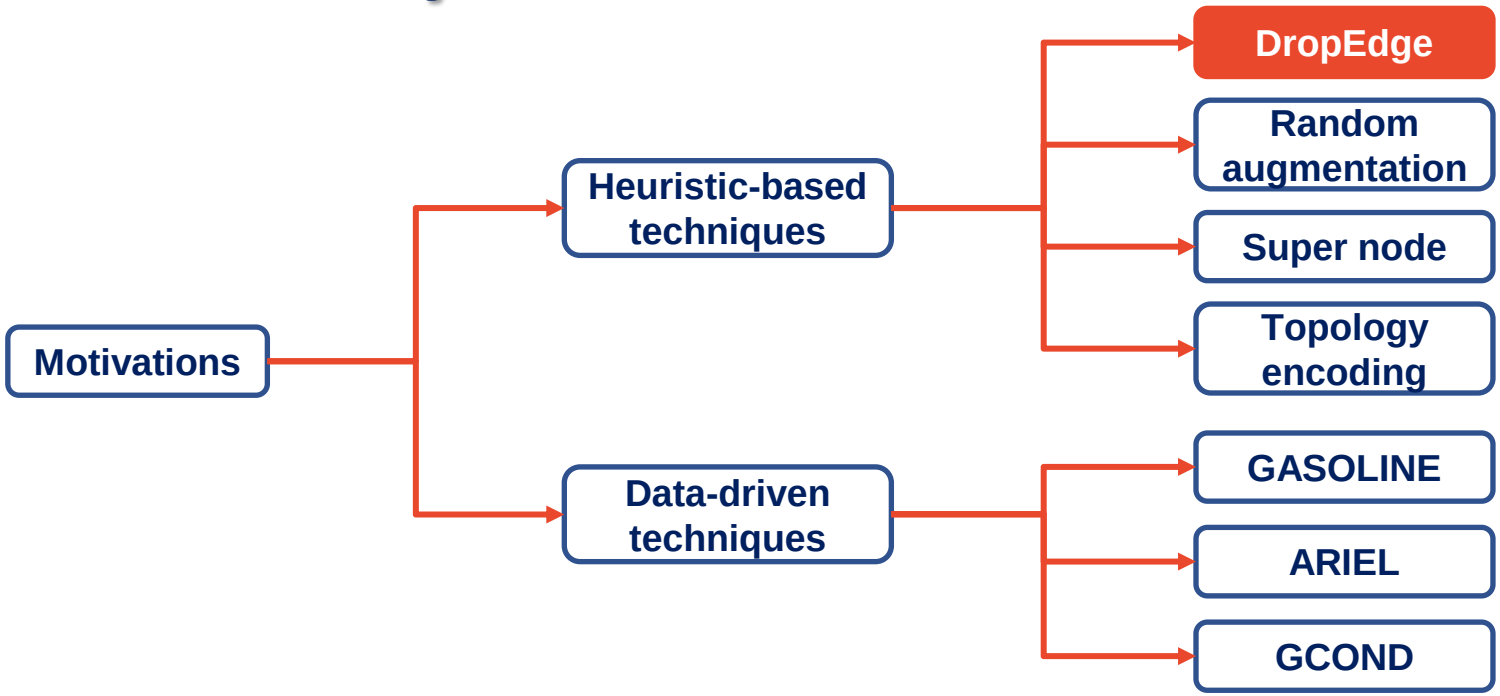
Task category	Name	Package	#Graphs	#Total nodes	#Total edges
Node-level	MAG240M	>=1.3.2	1	244,160,499	1,728,364,232
Link-level	WikiKG90Mv2	>=1.3.3	1	91,230,610	601,062,811
Graph-level	PCQM4Mv2	>=1.3.2	3,746,619	52,970,652	54,546,813

Figure credit: [1] [2]

And many more …

[1] https://www.oberlo.com/statistics
[2] https://ogb.stanford.edu/

Artificial dynamics in GNNs



DropEdge: Towards deep graph convolutional networks on node classification

Task: node classification
Artificial dynamic: randomly removes a certain number of edges from the input graph at each training epoch
Goal: reduce the message passing of GNN to alleviate over-smoothing

$$\mathbf{A}_{drop} = \mathbf{A} - \mathbf{A}'$$

DropEdge: alleviate over-smoothing

- **PageRank's convergence,** $\widetilde{\mathbf{A}}$ is a normalized adjacency matrix, $\boldsymbol{\pi}$ is only determined by $\widetilde{\mathbf{A}}$ for all the $\mathbf{x}$ except 0:

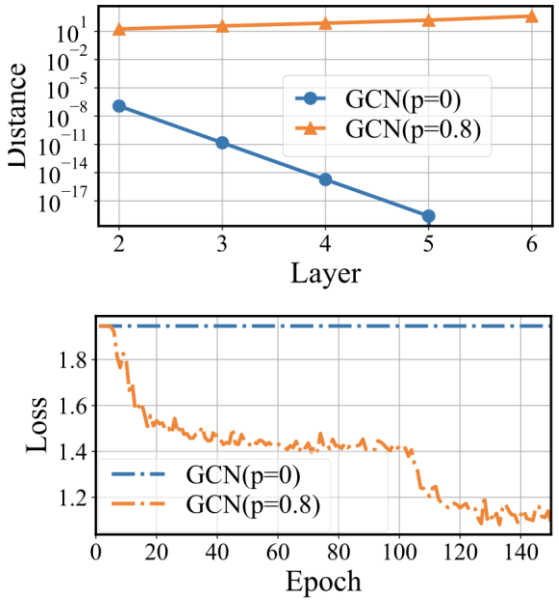
$$\lim_{k\to\infty} \widetilde{\mathbf{A}}^{k}\mathbf{x} = \boldsymbol{\pi}$$

- The following two GNNs share **similar spectral characteristics** and **empirical performance**:
  - **GCN:** $\sigma_2(\widetilde{\mathbf{A}}\sigma_1(\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2)$
  - **SGC:** $\sigma_2\left(\widetilde{\mathbf{A}}\widetilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1\mathbf{W}_2\right) = \sigma\left(\widetilde{\mathbf{A}}^{2}\mathbf{X}\mathbf{W}\right)$
- **SGC with many layers:** $\lim_{k\to\infty} \sigma\left(\widetilde{\mathbf{A}}^{k}\mathbf{X}\mathbf{W}\right)$, information from initial node feature $\mathbf{X}$ is missing ($\widetilde{\mathbf{A}}^{k}\mathbf{X}\mathbf{W}$ will converge to a matrix independent of $\mathbf{X}\mathbf{W}$).
- **Dropedge:** $\mathbf{A}_{drop} = \mathbf{A} - \mathbf{A}'$

Rong, Yu, et al. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." *International Conference on Learning Representations*.

Empirical results of DropEdge

(1) Embeddings are not over-smoothed and training moves forward:



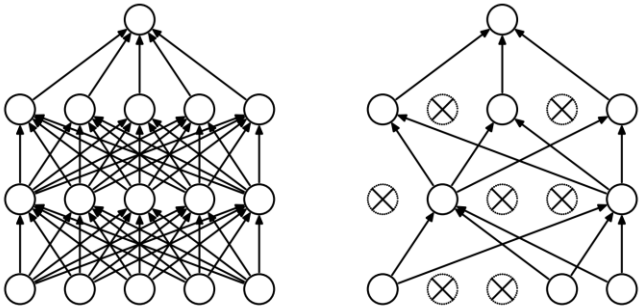
(2) Better performance, especially for the deep GNN cases

Dataset	Backbone	2 layers		8 layers		32 layers	
		Orignal	DropEdge	Orignal	DropEdge	Orignal	DropEdge
Cora	GCN	86.10	**86.50**	78.70	**85.80**	71.60	**74.60**
	ResGCN	-	-	85.40	**86.90**	85.10	**86.80**
	JKNet	-	-	86.70	**87.80**	87.10	**87.60**
	IncepGCN	-	-	86.70	**88.20**	87.40	**87.70**
	GraphSAGE	87.80	**88.10**	84.30	**87.10**	31.90	**32.20**
Citeseer	GCN	75.90	**78.70**	74.60	**77.20**	59.20	**61.40**
	ResGCN	-	-	77.80	**78.80**	74.40	**77.90**
	JKNet	-	-	79.20	**80.20**	71.70	**80.00**
	IncepGCN	-	-	79.60	**80.50**	72.60	**80.30**
	GraphSAGE	78.40	**80.00**	74.10	**77.10**	37.00	**53.60**
Pubmed	GCN	90.20	**91.20**	90.10	**90.90**	84.60	**86.20**
	ResGCN	-	-	89.60	**90.50**	90.20	**91.10**
	JKNet	-	-	90.60	**91.20**	89.20	**91.30**
	IncepGCN	-	-	90.20	**91.50**	OOM	**90.50**
	GraphSAGE	90.10	**90.70**	90.20	**91.70**	41.30	**47.90**
Reddit	GCN	96.11	**96.13**	96.17	**96.48**	45.55	**50.51**
	ResGCN	-	-	96.37	**96.46**	93.93	**94.27**
	JKNet	-	-	96.82	**97.02**	OOM	OOM
	IncepGCN	-	-	96.43	**96.87**	OOM	OOM
	GraphSAGE	96.22	**96.28**	96.38	**96.42**	96.43	**96.47**

Rong, Yu, et al. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." *International Conference on Learning Representations*.

Related works of DropEdge

- **Dropout [1]:** randomly zeroing a part of the feature transformation matrices



(a) Standard Neural Net (b) After applying dropout.

$$\sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W})$$

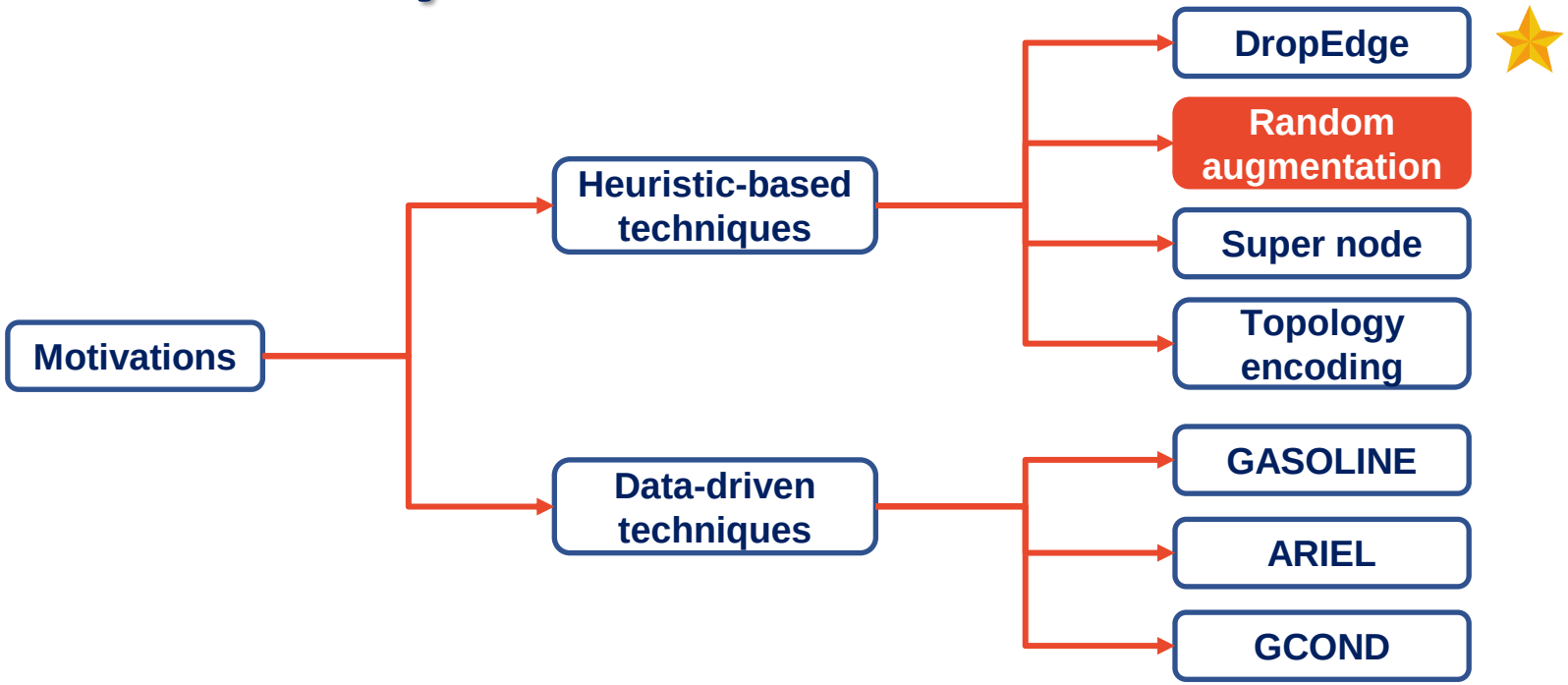
Target of DropEdge Target of Dropout

- **Sampling-based methods:** GraphSAGE [2], FastGCN [3], GraphSAINT[4], and many more

- **Other applications:** (1) improving GNN scalability, (2) alleviating over-fitting, (3) generating different views of the given graphs, and many more

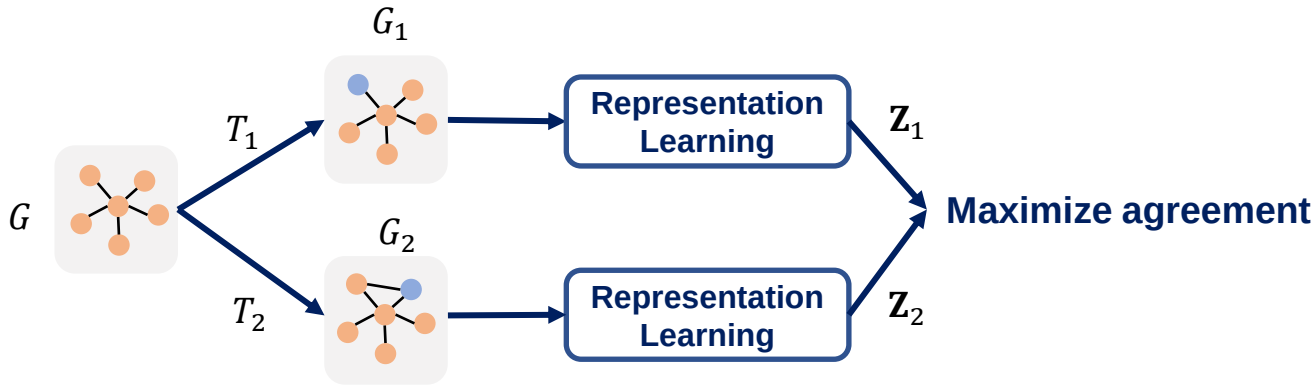
[1] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, pp. 1025–1035, 2017.
[2] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In Proceedings of the 6th International Conference on Learning Representation, 2018.
[3] Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method." International Conference on Learning Representations.

Artificial dynamics in GNNs



Graph contrastive learning (GraphCL) with augmentations

Task: unsupervised **graph-level** representation learning
Artificial dynamic: using random augmentations to generate views
Goal: contrastive learning on graph data

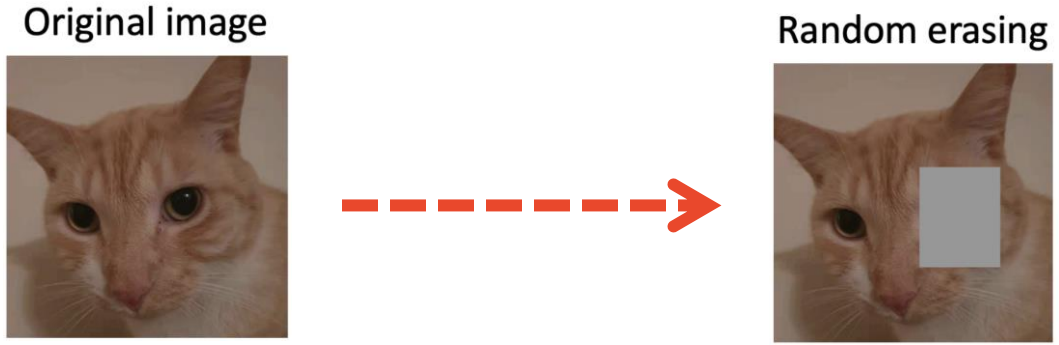


T_1, T_2 : Heuristic-based augmentations

You, Yuning, et al. "Graph contrastive learning with augmentations." Advances in neural information processing systems 33 (2020): 5812-5823.

Heuristic-based augmentations from GraphCL

Image data [1]:



Graph data [2]:

Table 1: Overview of data augmentations for graphs.

Data augmentation	Type	Underlying Prior
Node dropping	Nodes, edges	Vertex missing does not alter semantics.
Edge perturbation	Edges	Semantic robustness against connectivity variations.
Attribute masking	Nodes	Semantic robustness against losing partial attributes per node.
Subgraph	Nodes, edges	Local structure can hint the full semantics.

[1] Han, Jiawei, Jian Pei, and Hanghang Tong. Data mining: concepts and techniques. Morgan kaufmann, 2022.
[2] You, Yuning, et al. "Graph contrastive learning with augmentations." Advances in neural information processing systems 33 (2020): 5812-5823.

Empirical results of GraphCL in terms of augmentations

- The **selection** of augmentation methods is critical
- The augmentations **cannot generalize** across different datasets

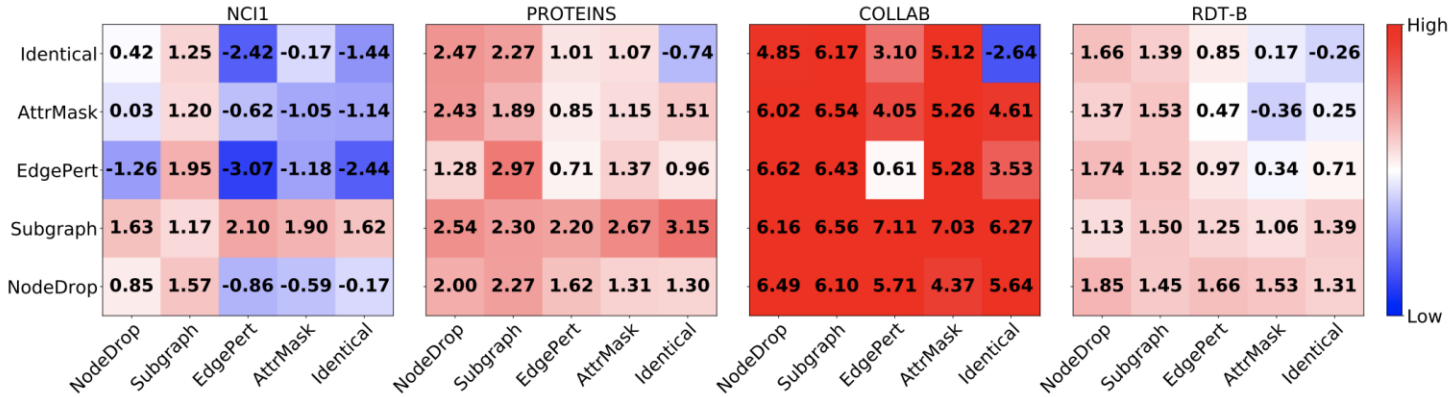


Figure 2: Semi-supervised learning accuracy gain (%) when contrasting different augmentation pairs, compared to training from scratch, under four datasets: NCI1, PROTEINS, COLLAB, and RDT-B. Pairing "Identical" stands for a no-augmentation baseline for contrastive learning, where the positive pair diminishes and the negative pair consists of two non-augmented graphs. Warmer colors indicate better performance gains. The baseline training-from-scratch accuracies are 60.72%, 70.40%, 57.46%, 86.63% for the four datasets respectively.

You, Yuning, et al. "Graph contrastive learning with augmentations." Advances in neural information processing systems 33 (2020): 5812-5823.

Trials and errors of heuristic-based augmentations

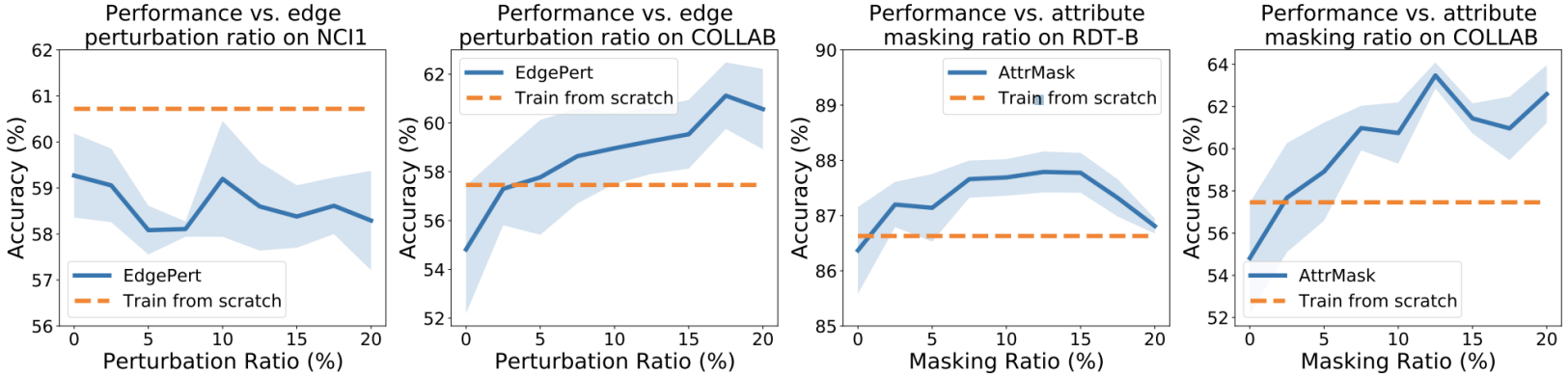


Figure 4: Performance versus augmentation strength. Left two figures implemented edge perturbation with different ratios. The right two figures apply attribute masking with different masking ratios.

The selection of augmentation tools and hyperparameters is **critical** but **labor-intensive**.

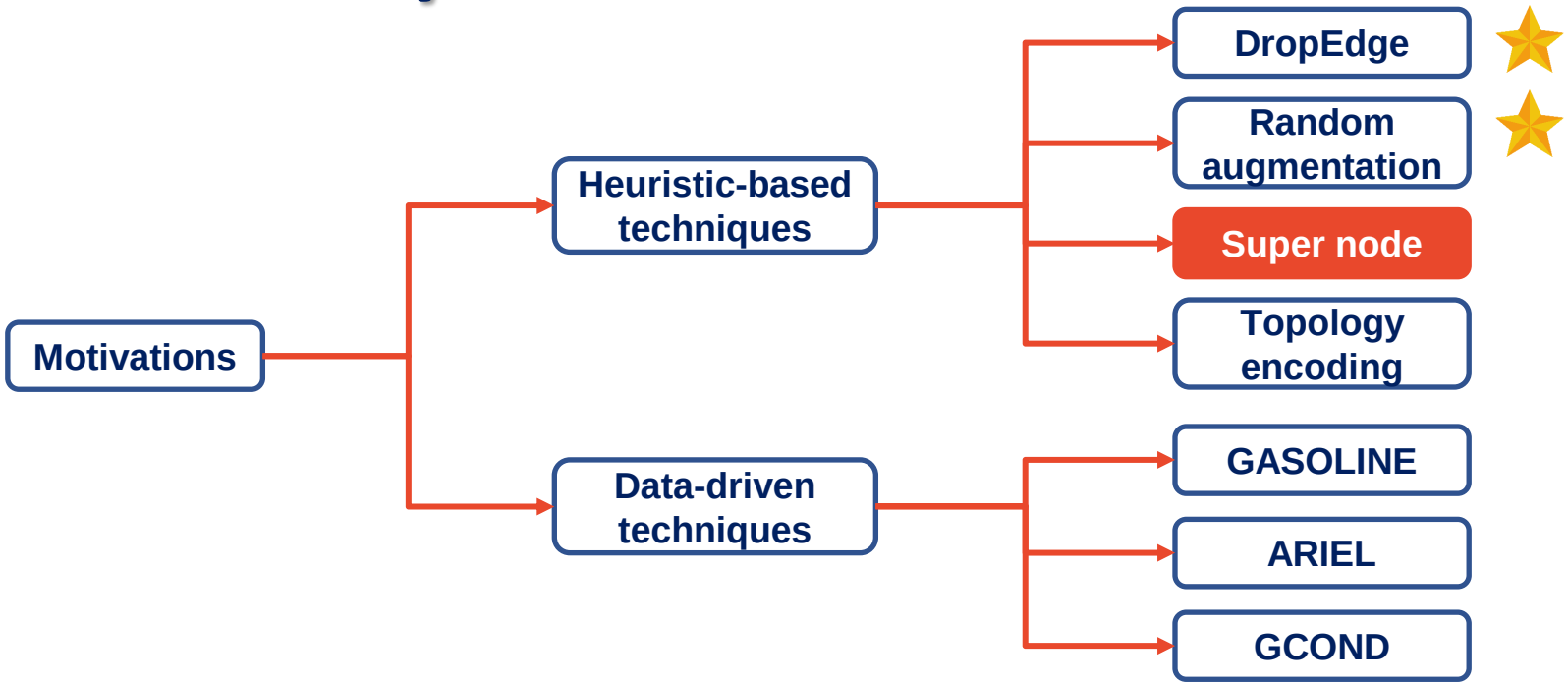
You, Yuning, et al. "Graph contrastive learning with augmentations." Advances in neural information processing systems 33 (2020): 5812-5823.

Related works of GraphCL

- Most of the augmentations adopted by GraphCL [1] are **heuristic**, e.g., uniformly random dropping nodes.
- **Learn to select** the heuristic-based augmentation combinations [2].
- **Adapt the probability of heuristic-based augmentation** according to some metrics [3]. E.g., adapt the node dropping probability according to the node centrality.
- Many more …

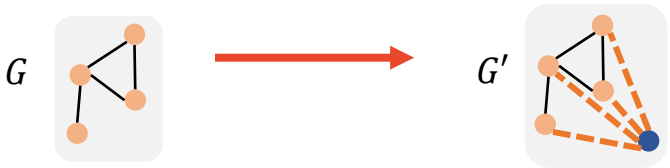
[1] You, Yuning, et al. "Graph contrastive learning with augmentations." Advances in neural information processing systems 33 (2020): 5812-5823.
[2] You, Yuning, et al. "Graph contrastive learning automated." International Conference on Machine Learning. PMLR, 2021.
[3] Zhu, Yanqiao, et al. "Graph contrastive learning with adaptive augmentation." Proceedings of the Web Conference 2021. 2021.

Artificial dynamics in GNNs



Inserting super node

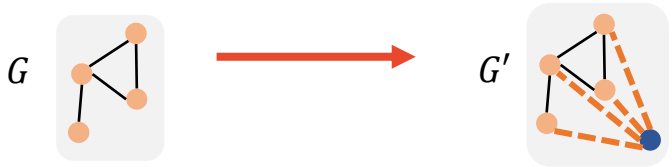
Representation works: [1-4]
Task: node/graph property prediction
Artificial dynamic: inserting a super node (a.k.a., virtual/master/dummy node) into the given graph
Goal: (1) to improve the message passing of GNN, and (2) can be used as a readout node for graph-level tasks.



[1] Gilmer, Justin, et al. "Neural message passing for quantum chemistry." International conference on machine learning. PMLR, 2017.
[2] J. Li, D. Cai, and X. He. Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741, 2017
[3] K. Ishiguro, S.-i. Maeda, and M. Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. arXiv preprint arXiv:1902.01020, 2019.
[4] T. Pham, T. Tran, H. Dam, and S. Venkatesh. Graph classification via deep learning with virtual nodes. arXiv preprint arXiv:1708.04357, 2017

Details comparison of inserting super node

	[1]	[2]	[3]	[4]
Weighted?	No	No	Yes. Learnable gates.	No
Directed?	No	Yes	No	No
Virtual node feature	Extra node features	Zeros at initialization	Graph features. E.g., graph diameter	Graph features
As readout?	No	Yes	No	Yes



[1] Gilmer, Justin, et al. "Neural message passing for quantum chemistry." International conference on machine learning. PMLR, 2017.
[2] J. Li, D. Cai, and X. He. Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741, 2017
[3] K. Ishiguro, S.-i. Maeda, and M. Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. arXiv preprint arXiv:1902.01020, 2019.
[4] T. Pham, T. Tran, H. Dam, and S. Venkatesh. Graph classification via deep learning with virtual nodes. arXiv preprint arXiv:1708.04357, 2017

Effectiveness of inserting super node on OGB dataset

Statistics from the leaderboard of the OGB datasets [1,2].

On ogbg-molhiv dataset

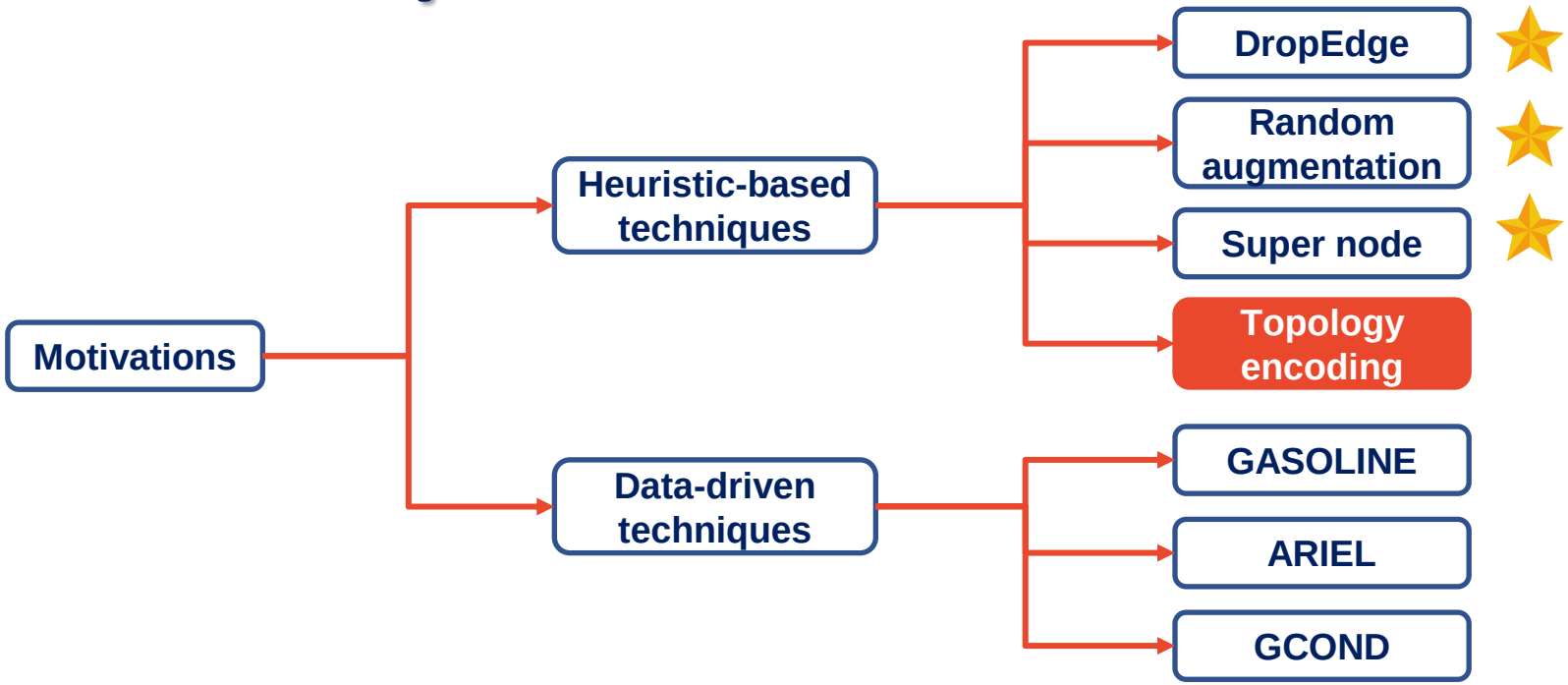
Rank	Method	Test ROC-AUC	Validation ROC-AUC
25	GIN+virtual node+FLAG	0.7748 ± 0.0096	0.8438 ± 0.0128
27	GIN+virtual node	0.7707 ± 0.0149	0.8479 ± 0.0068
31	GCN+virtual node	0.7599 ± 0.0119	0.8384 ± 0.0091

On ogbg-molhiv dataset

Rank	Method	Test AP	Validation AP
4	NestedGIN+virtual node	0.3007 ± 0.0037	0.3059 ± 0.0056
10	GINE+virtual nodes	0.2917 ± 0.0015	0.3065 ± 0.0030
13	DeeperGCN+virtual node+FLAG	0.2842 ± 0.0043	0.2952 ± 0.0029

[1] Hu, Weihua, et al. "Open graph benchmark: Datasets for machine learning on graphs." Advances in neural information processing systems 33 (2020): 22118-22133.
[2] https://ogb.stanford.edu/docs/leader_graphprop/

Artificial dynamics in GNNs



Distance encoding: Design provably more powerful neural networks for graph representation learning

Task: representation learning of a target (sub)graph S
Artificial dynamic: generating extra node features by encoding the topology information (distance in this paper) which is hard to be caught by 1-WL GNNs*.
Goal: to enrich the informativeness of node features

$$\mathbf{x}_u \leftarrow \mathbf{x}_u \oplus \mathrm{DE}(u, S)$$

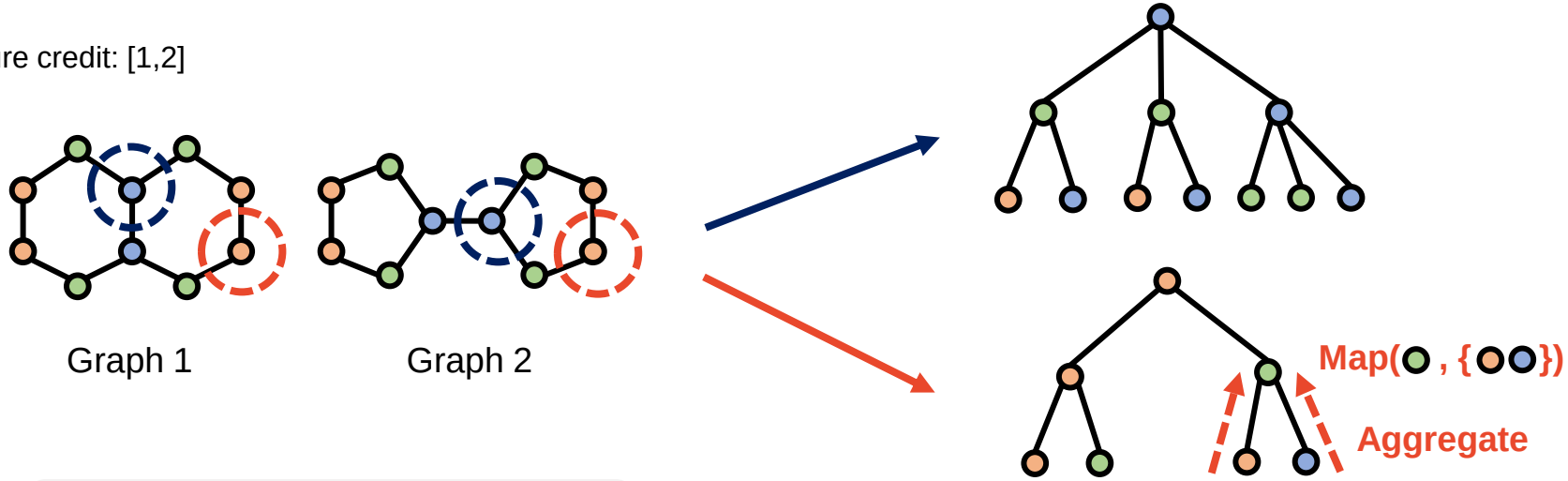
Raw feature of node u concatenate

1-WL GNNs refer to GNNs whose expressiveness is limited by the 1-WL test.

Li, Pan, et al. "Distance encoding: Design provably more powerful neural networks for graph representation learning." Advances in Neural Information Processing Systems 33 (2020): 4465-4478.

Expressiveness limitation of 1-WL GNNs

Figure credit: [1,2]



GNNs are not substructure-aware [3,4] and position-aware [5]

Node pairs from two graphs share the **same rooted subtrees**, which are used by Weisfeiler-Lehman (WL) graph isomorphism test and (most) GNN operations.

[1] Ding, Kaize, et al. "Data augmentation for deep graph learning: A survey." ACM SIGKDD Explorations Newsletter 24.2 (2022): 61-77.
[2] Sato, Ryoma. "A survey on the expressive power of graph neural networks." arXiv preprint arXiv:2003.04078 (2020).
[3] Chen, Zhengdao, et al. "Can graph neural networks count substructures?." Advances in neural information processing systems 33 (2020): 10383-10395.
[4] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec. Identity-aware graph neural networks. In AAAI, 2021
[5] You, Jiaxuan, Rex Ying, and Jure Leskovec. "Position-aware graph neural networks." International conference on machine learning. PMLR, 2019.

Details of the distance encoding

$$\mathbf{x}_u = \mathbf{x}_u \oplus \mathrm{DE}(u, S)$$

$$\mathrm{DE}(u, S) = \mathrm{AGG}(\{\mathrm{DE}(u, v) | v \in S\})$$

Set aggregation, e.g., SUM

DE Option 1: (generalized) PageRank score

$$\mathrm{DE}(u, v) = \sum_{k \geq 1} \gamma_k \tilde{\mathbf{A}}^k [u, v]$$

DE Option 2: shortest path distance

$$\mathrm{DE}(u, v) = \mathrm{spd}(u, v)$$

Li, Pan, et al. "Distance encoding: Design provably more powerful neural networks for graph representation learning." Advances in Neural Information Processing Systems 33 (2020): 4465-4478.

Empirical results of the distance encoding

- The size of the target subgraph S determines whether the task is node-level, link-level, or triangle-level.
- Their experiments focus on structure representation, so all the datasets are non-attributed.

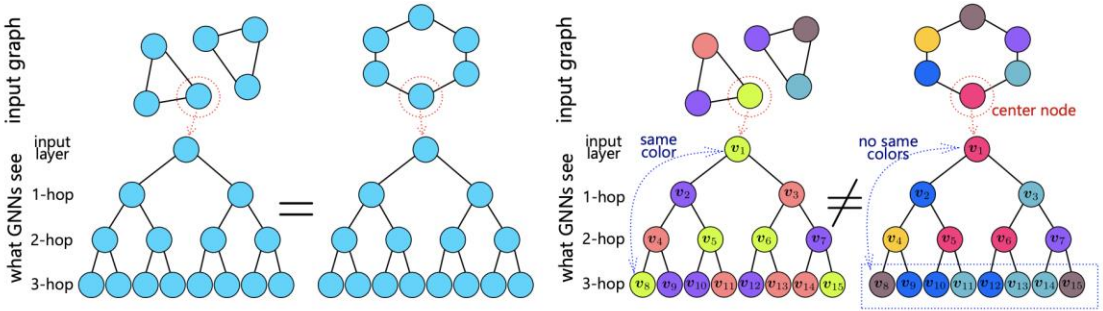
Data / Method	Nodes (Task 1): Average Accuracy			Node-pairs (Task 2): AUC			Node-triads (Task 3): AUC	
	Bra.-Airports	Eur.-Airports	USA-Airports	C.elegans	NS	PB	C.elegans	NS
GCN [20]	64.55±4.18	54.83±2.69	56.58±1.11	74.03±0.99	74.21±1.72	89.78±0.99	80.94±0.51	81.72±1.50
SAGE [21]	70.65±5.33	56.29±3.21	50.85±2.83	73.91±0.32	79.96±1.40	90.23±0.74	84.72±0.40	84.06±1.14
GIN [16]	71.89±3.60†	57.05±4.08	58.87±2.12	75.58±0.59	87.75±0.56	91.11±0.52	86.42±1.12†	94.59±0.66†
Struc2vec [5]	70.88±4.26	57.94±4.01†	61.92±2.61†	72.11±0.31	82.76±0.59	90.47±0.60	77.72±0.58	81.93±0.61
PGNN [10]	N/A	N/A	N/A	78.20±0.33	94.88±0.77	89.72±0.32	86.36±0.74	79.36±1.49
SEAL [9]	N/A	N/A	N/A	88.26±0.56†	98.55±0.32†	94.18±0.57†	N/A	N/A
DE-GNN-SPD	**73.28±2.47**	56.98±2.79	**63.10±0.68***	**89.37±0.17***	**99.09±0.79**	**94.95±0.37***	**92.17±0.72***	**99.65±0.40***
DE-GNN-LP	**75.10±3.80***	58.41±3.20*	**64.16±1.70***	86.27±0.33	98.01±0.55	91.45±0.41	86.24±0.18	**99.31±0.12***
DEA-GNN-SPD	**75.37±3.25***	57.99±2.39*	**63.28±1.59**	**90.05±0.26***	**99.43±0.63***	**94.49±0.24***	**93.35±0.65***	**99.84±0.14***

Table 1: Performance in Average Accuracy and Area Under the ROC Curve (AUC) (mean in percentage ± 95% confidence level). † highlights the best baselines. *, **bold font**, **bold font*** respectively highlights the case where our models' performance exceeds the best baseline on average, by 70% confidence, by 95% confidence.

Li, Pan, et al. "Distance encoding: Design provably more powerful neural networks for graph representation learning." Advances in Neural Information Processing Systems 33 (2020): 4465-4478.

Related works of distance encoding

- Distance [1] is not the only information that is hard to be caught by 1-WL GNNs
- GSN [2] and ID-GNN [3] augment the node feature with the counts of various motifs (e.g., cycles).
- Adding random features [4] into the existing node features can help GNN discriminate triangle structure



(a) Identical Features. (b) Random Features.

Figure credit: [4]

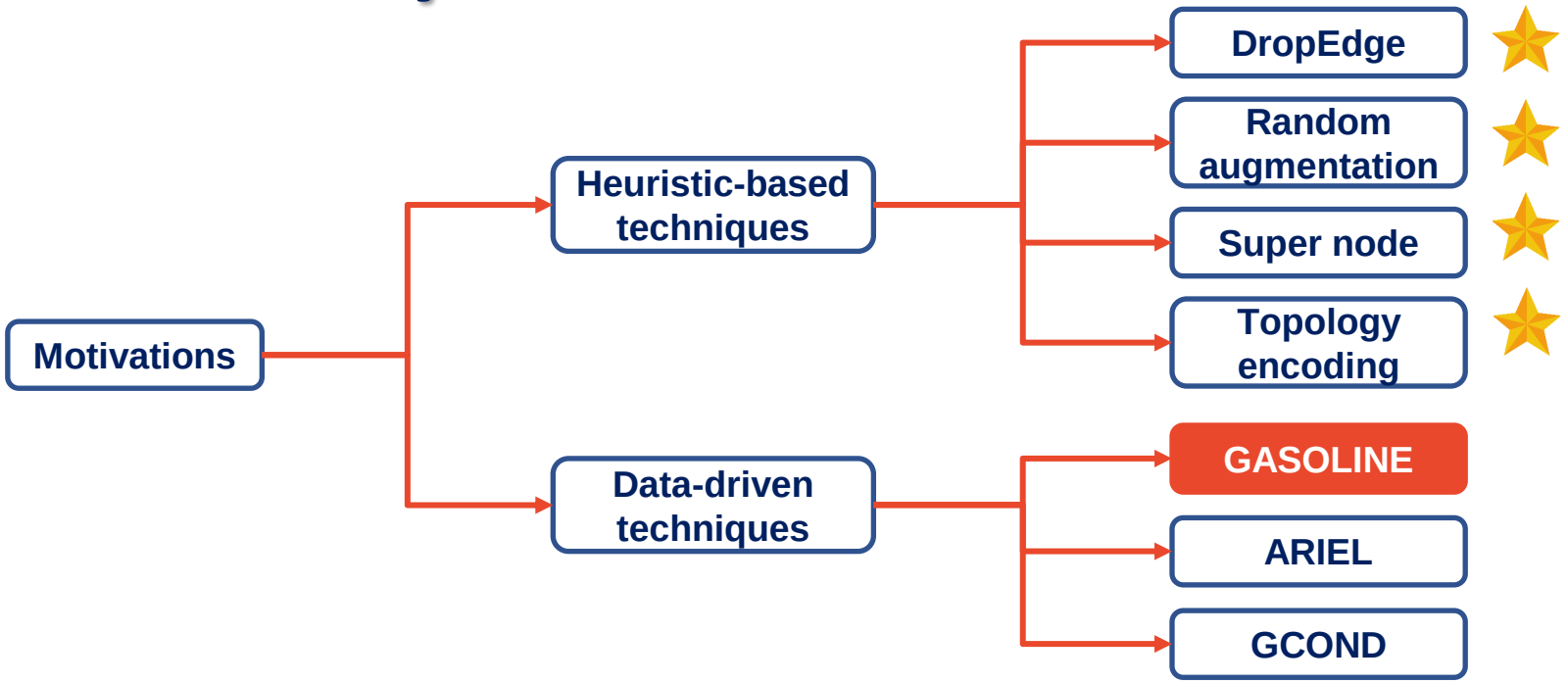
[1] Li, Pan, et al. "Distance encoding: Design provably more powerful neural networks for graph representation learning." Advances in Neural Information Processing Systems 33 (2020): 4465-4478.
[2] G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. TPAMI, 2022.
[3] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec. Identity-aware graph neural networks. In AAAI, 2021
[4] R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. In SDM, 2021

Other heuristic-based artificial dynamics

- Graph diffusions [1]
- Low-rank approximated adjacency matrix [2]
- Recover missing features based on homophily assumption [3]
- Many more [4] …

[1] Gasteiger, Johannes, Stefan Weißenberger, and Stephan Günnemann. "Diffusion improves graph learning." Advances in neural information processing systems 32 (2019).
[2] Entezari, Negin, et al. "All you need is low (rank) defending against adversarial attacks on graphs." Proceedings of the 13th International Conference on Web Search and Data Mining. 2020.
[3] Rossi, Emanuele, et al. "On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features." Learning on Graphs Conference. PMLR, 2022.
[4] Ding, Kaize, et al. "Data augmentation for deep graph learning: A survey." ACM SIGKDD Explorations Newsletter 24.2 (2022): 61-77.

Artificial dynamics in GNNs

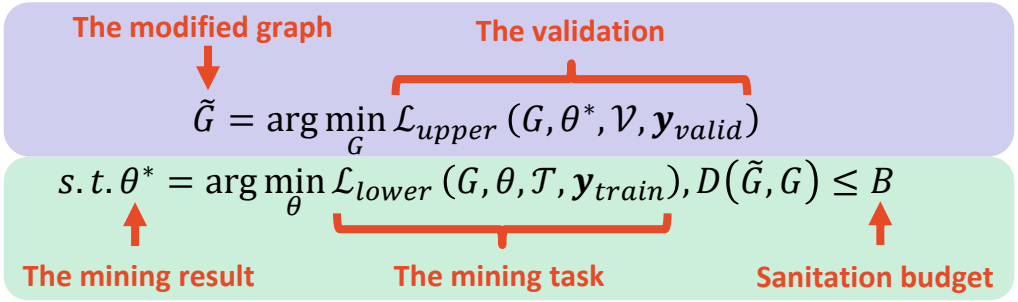


Graph sanitation with application to node classification (GASOLINE)

Task: Node classification

Artificial dynamic: View the graph data itself as a hyper-parameter and optimize it by a meta-learning fashioned bilevel optimization objective

Goal: improve the node classification performance



Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.

Instantiation of GASOLINE on node classification tasks

$$\tilde{G} = \arg\min_{G} \mathcal{L}_{upper}(G, \theta^*, \mathcal{V}, \boldsymbol{y}_{valid})$$
$$s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{lower}(G, \theta, \mathcal{T}, \boldsymbol{y}_{train})$$

$$\tilde{G} = \arg\min_{G} -\sum_{i \in \mathcal{V}} \sum_{j=1}^{c} y_{ij} \ln f(G, \theta^*)[i,j]$$

Performance of **the trained classifier** (i.e., $f(\theta^*)$) on **the validation set ($\mathcal{V}$)**

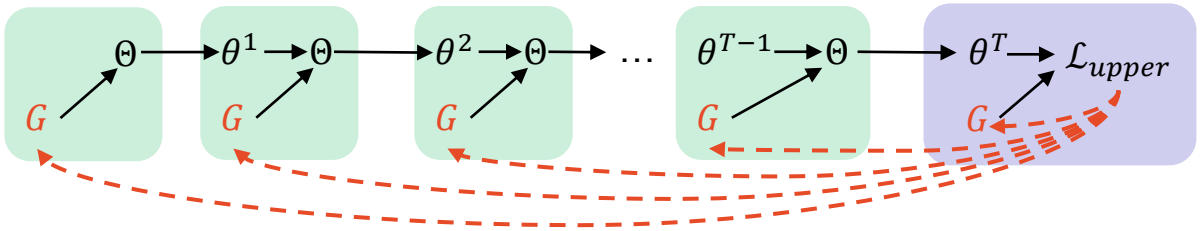
$$s.t.\ \theta^* = \arg\min_{\theta} -\sum_{i \in \mathcal{T}} \sum_{j=1}^{c} y_{ij} \ln f(G, \theta)[i,j]$$

Performance of the classifier (i.e., $f(\theta)$) on **the training set ($\mathcal{T}$)**

Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.

Hyper-gradient computation of GASOLINE

Iterative differentiation [1]: view the updating of lower-level problem as a dynamic system in T iterations (i.e., $\theta^* = \theta^T$):



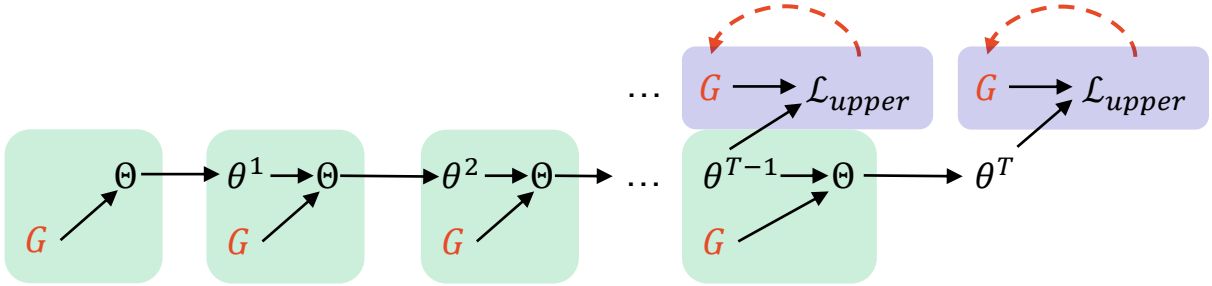
$$\tilde{G} = \arg\min_{G} \mathcal{L}_{upper}(G, \theta^*, \mathcal{V}, \boldsymbol{y}_{valid})$$
$$s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{lower}(G, \theta, \mathcal{T}, \boldsymbol{y}_{train})$$

Θ: iterative optimizer over $\mathcal{L}_{lower}$ (e.g., GD)

- - ▸: gradient for updating G

Sum all the gradient (i.e., - - ▸) together as the hyper-gradient ∇_G for updating G

First order approximation [2]:



[1] Franceschi, Luca, et al. "Forward and reverse gradient-based hyperparameter optimization." International Conference on Machine Learning. PMLR, 2017.
[2] Nichol, Alex, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms." arXiv preprint arXiv:1803.02999 (2018).

Hyper-gradient-guided modification of GASOLINE

- Update G based on $\nabla_G = \{\nabla_{\mathbf{A}}, \nabla_{\mathbf{X}}\}$(update $\mathbf{A}$ as an example)
  - **(continuous update)** gradient descent:
    - $\mathbf{A} \leftarrow \mathbf{A} - \underbrace{\frac{B}{\sum_{i,j}|\nabla_{\mathbf{A}}|[i,j]}}_{\text{Budget constrained learning rate}} \cdot \nabla_{\mathbf{A}}$. Or
  - **(discrete update)** flip B entries in $\mathbf{A}$ whose indices obtain largest score in score matrix [1]:
    - $\mathbf{S} = \underbrace{(-\nabla_{\mathbf{A}})}_{\text{preference}} \circ \underbrace{(\mathbf{1} - 2\mathbf{A})}_{\text{availability}}$

Explanation of $\mathbf{S}$

- Node i, j are connected, $\mathbf{A}[\mathrm{i},\mathrm{j}] = 1$
- $(\mathbf{1} - \mathbf{2A})[\mathrm{i},\mathrm{j}] = -1$
- $-\nabla_{\mathbf{A}}[\mathrm{i},\mathrm{j}] > 0, \mathbf{S}[\mathrm{i},\mathrm{j}] < 0$ 
- $-\nabla_{\mathbf{A}}[\mathrm{i},\mathrm{j}] < 0, \mathbf{S}[\mathrm{i},\mathrm{j}] > 0$ 
- Dropping edge will get high score
- Reverse scenario for unconnected node pairs

[1] Zügner, Daniel, and Stephan Günnemann. "Adversarial Attacks on Graph Neural Networks via Meta Learning." International Conference on Learning Representations. 2018.

Low-rank speedup of GASOLINE (1)

- For updating $\mathbf{A}$, $\nabla_{\mathbf{A}}$ tends to be **dense**
- $\nabla_{\mathbf{A}}\mathcal{L}$: time complexity $O(n^2 d)$, space complexity $O(n^2)$
- Recap the goal: $\widetilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$
- Key idea: the modification should be implemented over **a small set of nodes**, specifically:
  - decompose the incremental matrix: $\Delta\mathbf{A} = \mathbf{U}\mathbf{V}'$

Low-rank speedup of GASOLINE (2)

- Change optimization variable from $\mathbf{A}$ to $\mathbf{U}$ and $\mathbf{V}$. I.e., represent $G = \{\mathbf{A} + \mathbf{UV}', \mathbf{X}\}$ in the formula:

$$\tilde{G} = \arg\min_{G} \mathcal{L}_{upper}(G, \theta^*, \mathcal{V}, \boldsymbol{y}_{valid})$$
$$s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{lower}(G, \theta, \mathcal{T}, \boldsymbol{y}_{train})$$

- Initialize $\mathbf{U}$ and $\mathbf{V}$ as zero-mean gaussian
- Compute $\nabla_{\mathbf{U}}$ and $\nabla_{\mathbf{V}}$ in the same paradigm as the full-version Gasoline
- Update $\mathbf{U}$ and $\mathbf{V}$ in a continuous way
- $\nabla_{\mathbf{U}}\mathcal{L}$ and $\nabla_{\mathbf{V}}\mathcal{L}$: time complexity $\boldsymbol{O(nd^2 + md)}$, space complexity $\boldsymbol{O(m + nd)}$
  - $\nabla_{\mathbf{A}}\mathcal{L}$: time complexity $O(n^2 d)$, space complexity $O(n^2)$

Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.

Empirical performance of GASOLINE

Attack	Data	APPNP	GAT	Jaccard	SVD	RGCN	DE	LDS	G-DT	G-CF	G-DTCF
metattack	Cora	47.0±0.7	48.8±0.2	65.4±0.9	60.3±0.8	50.6±0.8	48.7±0.9	58.7±1.3	67.3±0.7	57.0±0.9	**68.8±0.9**
	Citeseer	49.4±2.2	62.4±0.7	57.1±1.0	49.5±0.8	55.5±1.4	50.1±2.3	58.2±2.3	**63.5±1.5**	58.4±1.5	62.2±1.0
	Polblogs	58.4±3.6	48.2±6.6	N/A	**79.1±2.4**	50.8±0.9	56.4±6.3	63.7±5.7	65.0±0.7	55.0±4.1	64.7±1.4
Nettack	Cora	60.7±1.2	54.2±2.3	63.7±1.4	52.9±2.8	56.5±1.1	60.8±1.0	64.5±2.4	64.5±2.2	63.9±2.4	**66.1±1.9**
	Citeseer	68.3±6.8	61.9±4.4	72.5±3.3	50.2±6.6	56.4±1.5	63.3±4.7	71.0±3.3	71.6±3.9	69.4±4.8	**74.3±1.6**
	Polblogs	90.5±1.0	91.1±0.7	N/A	**93.6±1.2**	93.1±0.2	89.1±2.4	91.1±1.8	92.3±1.6	90.3±0.7	92.4±1.7
random attack	Cora	74.3±0.4	58.1±1.0	75.1±0.5	72.6±0.3	68.9±0.4	73.9±0.6	76.6±0.4	77.1±0.3	**78.3±0.5**	77.8±0.2
	Citeseer	69.8±0.6	60.8±1.6	69.7±0.5	66.7±0.4	65.7±0.2	69.4±0.5	72.3±0.4	**73.8±0.2**	72.3±0.4	73.4±0.5
	Polblogs	74.7±2.8	**84.5±1.0**	N/A	83.3±2.8	81.7±0.9	75.9±1.4	73.2±2.8	73.4±4.1	77.1±1.6	77.6±2.9

- Variants of Gasoline (short as **G**) perform best on Cora and CiteSeer
- Incorporating with established methods
- More experiments in our paper

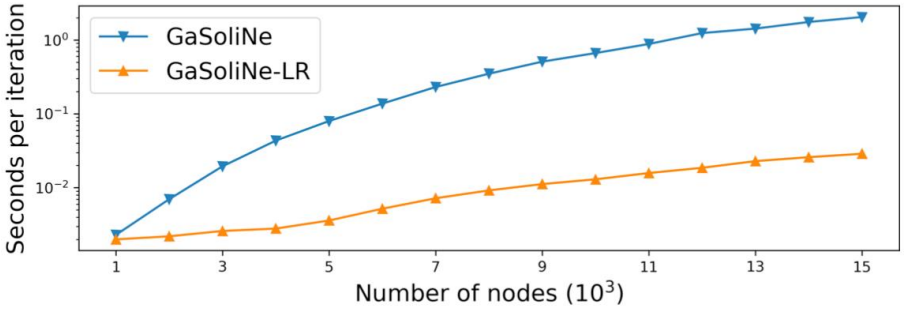
Data	With GASOLINE?	GAT	SVD	RGCN
Cora	N	48.8±0.2	60.3±0.8	50.6±0.8
	Y	63.7±0.6	79.7±0.6	62.6±0.6 ✓
Citeseer	N	62.4±0.7	49.5±0.8	55.5±1.4
	Y	69.7±0.2	76.5±0.6	66.1±0.8 ✓
Polblogs	N	48.2±6.6	79.1±2.4	50.8±0.9
	Y	70.8±0.6	89.2±0.7	67.7±0.3 ✓

Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.

Efficacy of the low-rank Gasoline

Data	Ptb Rate (%)	APPNP	GaSoliNe	GaSoliNe-LR
Cora	0	84.0±0.4	85.2±0.2	84.4±0.3
	5	74.1±0.7	77.4±0.5	75.0±0.3
	10	65.2±0.4	70.8±0.5	67.9±0.9
	15	58.2±1.1	67.1±0.8	65.3±0.8
	20	51.7±0.7	62.5±0.5	60.2±1.2
	25	47.0±0.7	57.3±0.6	57.1±0.5
Citeseer	0	71.8±0.4	74.7±0.2	73.4±0.2
	5	67.6±0.9	69.6±0.7	68.2±0.8
	10	61.8±0.8	66.3±1.0	63.9±0.4
	15	54.1±0.8	59.3±1.1	56.8±1.1
	20	51.0±1.2	56.5±0.9	55.3±0.9
	25	49.4±2.2	57.7±1.8	56.5±0.8
Polblogs	0	94.1±0.6	95.3±0.6	95.7±0.3
	5	70.1±0.6	73.8±0.9	93.4±0.3
	10	69.8±0.8	72.8±0.4	90.8±0.2
	15	67.5±0.5	70.1±1.2	88.7±0.3
	20	64.1±0.9	68.5±1.0	88.0±0.3
	25	57.0±3.6	64.8±2.1	89.9±0.5

- **Comparable performance** with Gasoline
- Strong performance on **Polblogs**
- **Much more efficient** compared with Gasoline

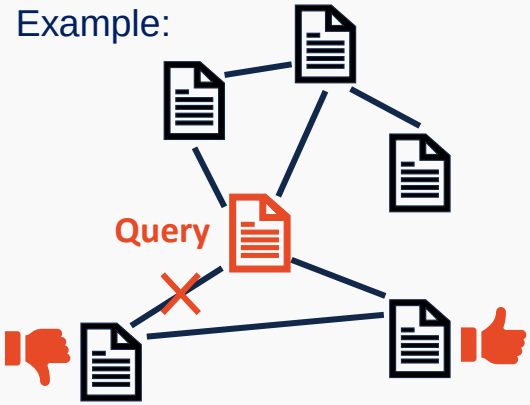


Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.

Related works of GASOLINE

- GASOLINE [1] can work beyond node classification, E.g., supervised random walk [2,3]

$$\tilde{G} = \arg\min_{G} \mathcal{L}_{upper}(G, \theta^*, \mathcal{V}, \boldsymbol{y}_{valid})$$
$$s.t.\ \theta^* = \arg\min_{\theta} \mathcal{L}_{lower}(G, \theta, \mathcal{T}, \boldsymbol{y}_{train})$$



$\mathcal{L}_{lower}$: **personalized PageRank**

$\mathcal{L}_{upper}$: **node pair-wise ranking performance constraints**

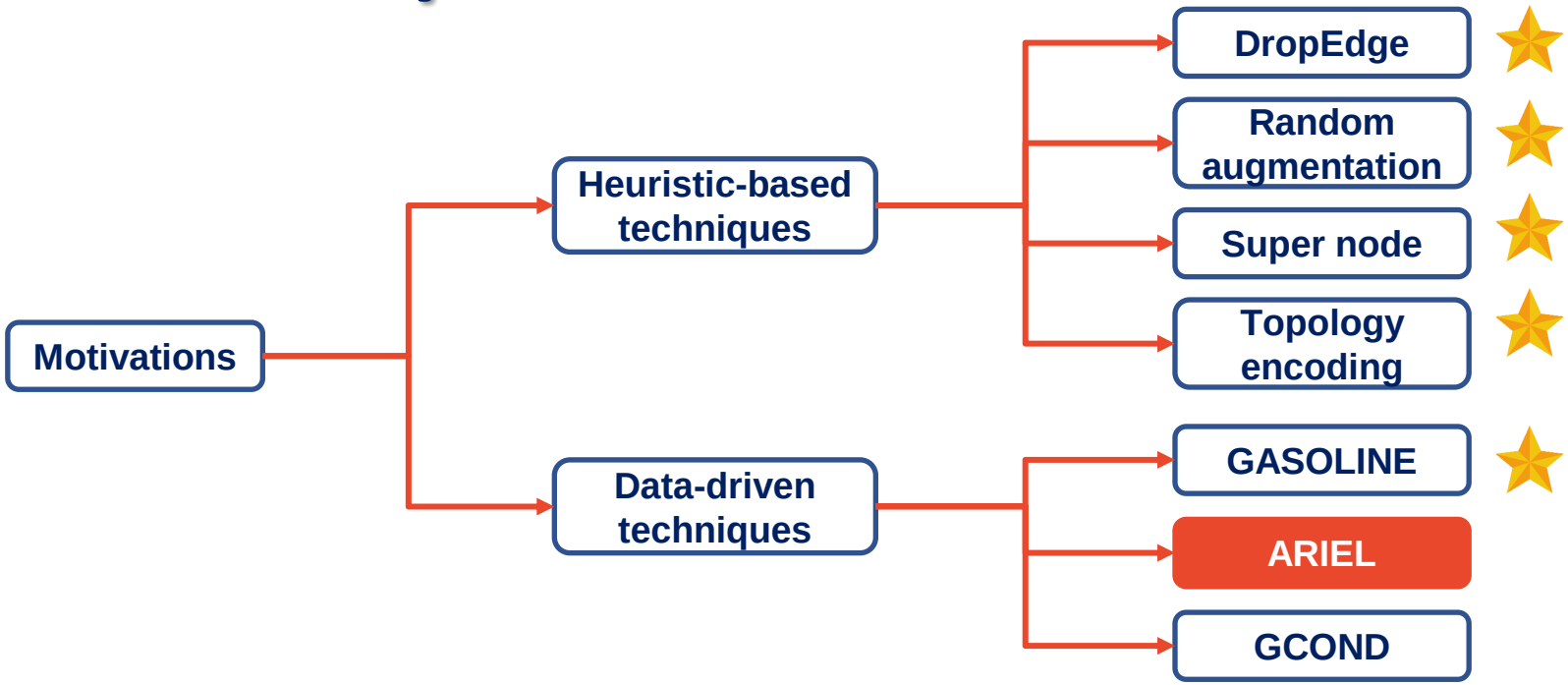
$\boldsymbol{D}(\tilde{\boldsymbol{G}}, \boldsymbol{G}) \leq \boldsymbol{B}$: **remove 1 edge**

More examples in our paper.

- GASOLINE learns the adjacency matrix determinedly, and LDS [4] learns the graph generation matrix instead.

[1] Xu, Zhe, Boxin Du, and Hanghang Tong. "Graph sanitation with application to node classification." Proceedings of the ACM Web Conference 2022.
[2] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In WSDM. 635–644.
[3] Liangyue Li, Yuan Yao, Jie Tang, Wei Fan, and Hanghang Tong. 2016. QUINT: on query-specific optimal networks. In SIGKDD. 985–994.
[4] Franceschi, Luca, et al. "Learning discrete structures for graph neural networks." International conference on machine learning. PMLR, 2019.

Artificial dynamics in GNNs

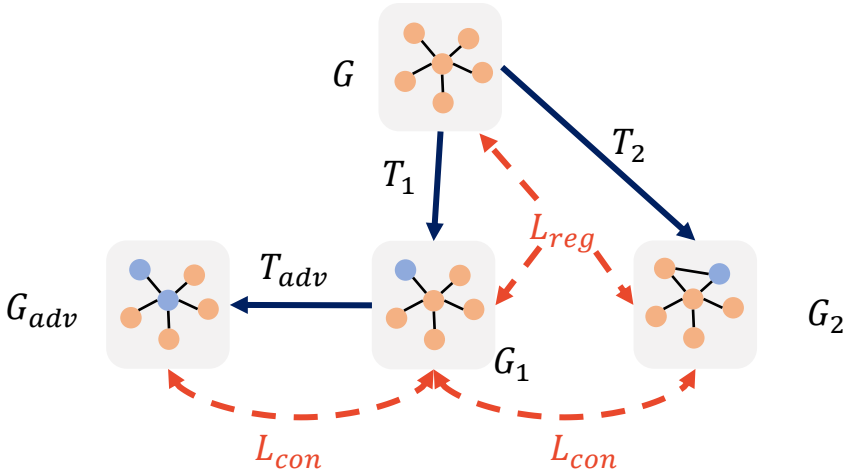


Adversarial graph contrastive learning with information regularization (ARIEL)

Task: self-supervised node representation learning

Artificial dynamic: using adversarial attacks to generate an extra view of the given graph

Goal: improve graph contrastive learning performance



T_1, T_2 : sampled from **(1) edge dropping, (2) feature masking**

Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

Attack details of the ARIEL

$$G_{adv} = \arg\max_{G'} L_{con}(G_1, G')$$

$$\text{s.t.} \sum_{i,j} |\mathbf{A}'[i,j] - \mathbf{A}[i,j]| \leq \Delta_{\mathbf{A}}, \sum_{i,j} |\mathbf{X}'[i,j] - \mathbf{X}[i,j]| \leq \Delta_{\mathbf{X}}$$

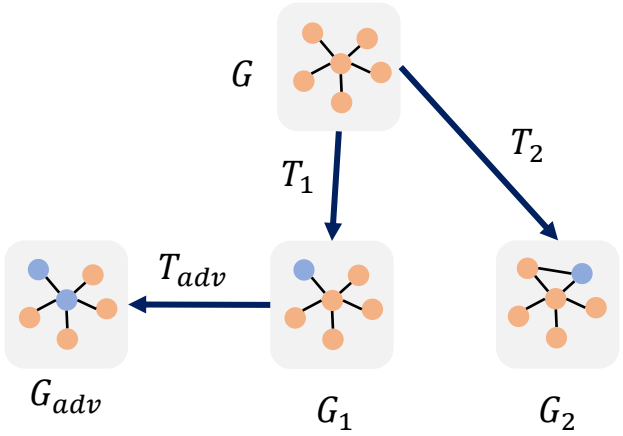
$\mathbf{L_A}$: perturbations on edges; $\mathbf{L_X}$: perturbations on features

$$\mathbf{A}_{adv} = \mathbf{A} + (\mathbf{1}_{n\times n} - \mathbf{I}_n - 2\mathbf{A}) \circ \mathbf{L_A}$$

$$\mathbf{X}_{adv} = \mathbf{X} + \mathbf{L_X}$$

$\mathbf{L_A} \in \{0,1\}^{n\times n}$ is symmetric and discrete,

its convex hull $\tilde{\mathbf{L}}_{\mathbf{A}} \in [0,1]^{n\times n}$



Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

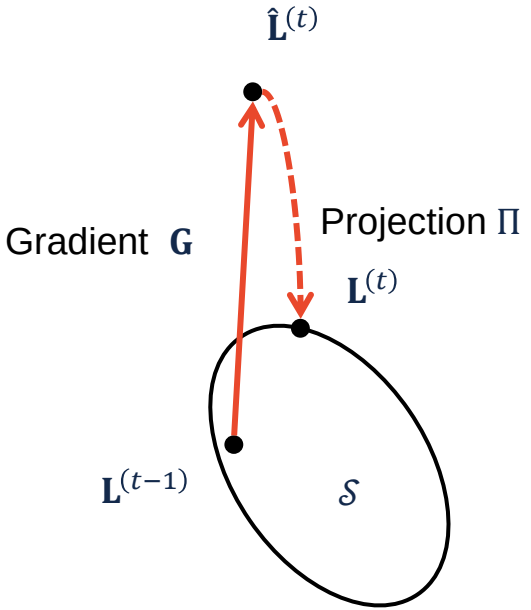
Project gradient attack (PGD) details (2)

PGD attack updates:

$$\tilde{\mathbf{L}}_{\mathbf{A}}^{(t)} = \Pi_{\mathcal{S}_A}(\tilde{\mathbf{L}}_{\mathbf{A}}^{(t-1)} + \alpha \mathbf{G}_{\mathbf{A}}^{(t)})$$

$$\mathbf{L}_{\mathbf{X}}^{(t)} = \Pi_{\mathcal{S}_X}(\mathbf{L}_{\mathbf{X}}^{(t-1)} + \beta\, \mathrm{sgn}(\mathbf{G}_{\mathbf{X}}^{(t)}))$$

$\mathbf{G}_{\mathbf{A}}^{(t)}$ and $\mathbf{G}_{\mathbf{X}}^{(t)}$: gradients of contrastive loss with respect to $\tilde{\mathbf{L}}_{\mathbf{A}}^{(t-1)}$ and $\mathbf{L}_{\mathbf{X}}^{(t-1)}$



[1] Xu et al. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. IJCAI 2019.
[2] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.

Project gradient attack (PGD) details (2)

PGD attack updates:

$$\tilde{\mathbf{L}}_{\mathbf{A}}^{(t)} = \Pi_{\mathcal{S}_{\mathbf{A}}}\left(\tilde{\mathbf{L}}_{\mathbf{A}}^{(t-1)} + \alpha \mathbf{G}_{\mathbf{A}}^{(t)}\right), \mathbf{L}_{\mathbf{X}}^{(t)} = \Pi_{\mathcal{S}_{\mathbf{X}}}(\mathbf{L}_{\mathbf{X}}^{(t-1)} + \beta\, \mathrm{sgn}(\mathbf{G}_{\mathbf{X}}^{(t)}))$$

- $\Pi_{\mathcal{S}_{\mathbf{A}}}(\boldsymbol{Z}) = \begin{cases} P(\mathbf{Z} - \mu \mathbf{1}_{n\times n}), & \sum_{i,j} P(\mathbf{Z} - \mu \mathbf{1}_{n\times n})[i,j] = \Delta_{\mathbf{A}} \text{ for } \mu > 0 \\ P(\mathbf{Z}), & \sum_{i,j} P(\mathbf{Z})[i,j] \leq \Delta_{\mathbf{A}} \end{cases}$

($P(\mathbf{Z})$ elementwisely clips $\mathbf{Z}$ into $[0,1]$)

- $\Pi_{\mathcal{S}_{\mathbf{X}}}$ clips $\mathbf{L}_{\mathbf{X}}$ into the range $[-\delta_{\mathbf{X}}, \delta_{\mathbf{X}}]$ elementwisely

[1] Xu et al. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. IJCAI 2019.
[2] Madry et al. Towards Deep Learning Models Resistant to Adversarial Attacks. ICLR 2018.

Regularization of the ARIEL

Information regularization: to constrain the views from the random augmentation

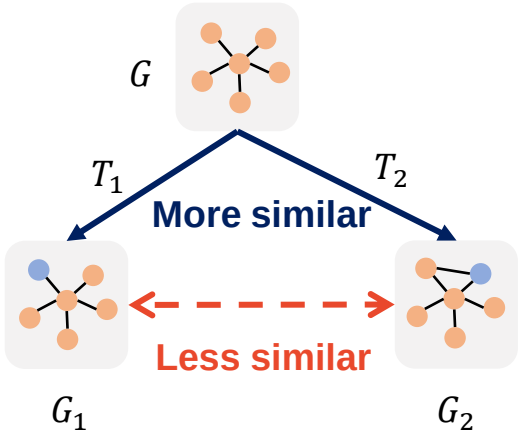
$$I_{i,1} = similarity(\mathbf{H}_1[i,:], \mathbf{H}[i,:])$$

$$I_{i,2} = similarity(\mathbf{H}_2[i,:], \mathbf{H}[i,:])$$

$$I_{i,12} = similarity(\mathbf{H}_1[i,:], \mathbf{H}_2[i,:])$$

$$d_i = 2I_{i,12} - (I_{i,1} + I_{i,2})$$

$$L_{reg}(G_1, G_2, G) = \frac{1}{n}\sum_{i=1}^{n} \max(d_i, 0)$$



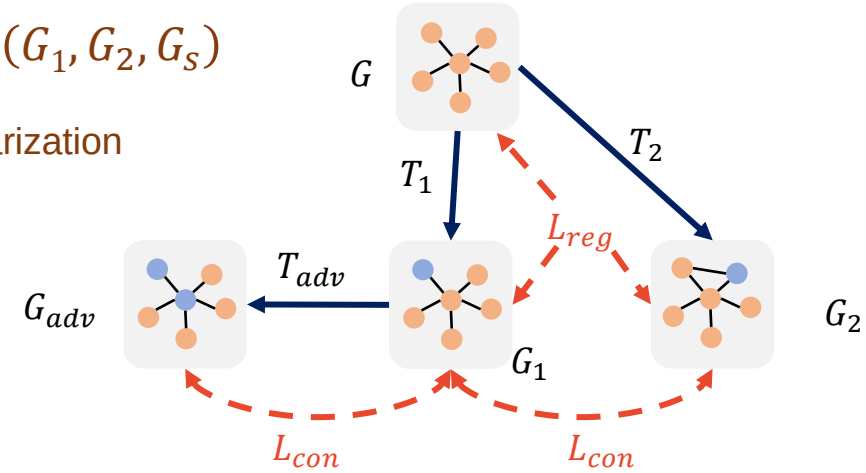
Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

Objective function of the ARIEL

$$L = L_{con}(G_1, G_2) + \varepsilon_1 L_{con}(G_1, G_{adv}) + \varepsilon_2 L_{reg}(G_1, G_2, G_s)$$

Contrastive loss Adversarial contrastive loss regularization

$$\text{s.t. } G_{adv} = \arg\max_{G'} L_{con}(G_1, G'),$$
$$\sum_{i,j} |\mathbf{A}'[i,j] - \mathbf{A}[i,j]| \leq \Delta_{\mathbf{A}},$$
$$\sum_{i,j} |\mathbf{X}'[i,j] - \mathbf{X}[i,j]| \leq \Delta_{\mathbf{X}}$$



Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

Empirical performance (node classification accuracy) of the ARIEL

Method	Cora	CiteSeer	Amazon-Computers	Amazon-Photo	Coauthor-CS	Coauthor-Physics
GCN	84.14 ± 0.68	69.02 ± 0.94	88.03 ± 1.41	92.65 ± 0.71	92.77 ± 0.19	95.76 ± 0.11
GAT	83.18 ± 1.17	69.48 ± 1.04	85.52 ± 2.05	91.35 ± 1.70	90.47 ± 0.35	94.82 ± 0.21
DeepWalk+features	79.82 ± 0.85	67.14 ± 0.81	86.23 ± 0.37	90.45 ± 0.45	85.02 ± 0.44	94.57 ± 0.20
DGI	84.24 ± 0.75	69.12 ± 1.29	88.78 ± 0.43	92.57 ± 0.23	92.26 ± 0.12	95.38 ± 0.07
GMI	82.43 ± 0.90	70.14 ± 1.00	83.57 ± 0.40	88.04 ± 0.59	OOM	OOM
MVGRL	**84.39 ± 0.77**	69.85 ± 1.54	89.02 ± 0.21	92.92 ± 0.25	92.22 ± 0.22	95.49 ± 0.17
GRACE	83.40 ± 1.08	69.47 ± 1.36	87.77 ± 0.34	92.62 ± 0.25	93.06 ± 0.08	95.64 ± 0.08
GCA-DE	82.57 ± 0.87	72.11 ± 0.98	88.10 ± 0.33	92.87 ± 0.27	93.08 ± 0.18	95.62 ± 0.13
GCA-PR	82.54 ± 0.87	72.16 ± 0.73	88.18 ± 0.39	92.85 ± 0.34	93.09 ± 0.15	95.58 ± 0.12
GCA-EV	81.80 ± 0.92	67.07 ± 0.79	87.95 ± 0.43	92.63 ± 0.33	93.06 ± 0.14	95.64 ± 0.08
ARIEL	84.28 ± 0.96	**72.74 ± 1.10**	**91.13 ± 0.34**	**94.01 ± 0.23**	**93.83 ± 0.14**	**95.98 ± 0.05**

- More consistent improvement with adversarial training
- Better augmentation qualities than heuristic-based methods

Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

Robustness of the ARIEL

Topology attack with Metattack [1] and random mask attributes

Method	Cora	CiteSeer	Amazon-Computers	Amazon-Photos	Coauthor-CS	Coauthor-Physics
GCN	80.03 ± 0.91	62.98 ± 1.20	84.10 ± 1.05	91.72 ± 0.94	80.32 ± 0.59	87.47 ± 0.38
GAT	79.49 ± 1.29	63.30 ± 1.11	81.60 ± 1.59	90.66 ± 1.62	77.75 ± 0.80	86.65 ± 0.41
DeepWalk+features	74.12 ± 1.02	63.20 ± 0.80	79.08 ± 0.67	88.06 ± 0.41	49.30 ± 1.23	79.26 ± 1.38
DGI	80.84 ± 0.82	64.25 ± 0.96	83.36 ± 0.55	91.27 ± 0.29	78.73 ± 0.50	85.88 ± 0.37
GMI	79.17 ± 0.76	65.37 ± 1.03	77.42 ± 0.59	89.44 ± 0.47	80.92 ± 0.64	87.72 ± 0.45
MVGRL	**80.90 ± 0.75**	64.81 ± 1.53	83.76 ± 0.69	91.76 ± 0.44	79.49 ± 0.75	86.98 ± 0.61
GRACE	78.55 ± 0.81	63.17 ± 1.81	84.74 ± 1.13	91.26 ± 0.37	80.61 ± 0.63	85.71 ± 0.38
GCA	76.79 ± 0.97	64.89 ± 1.33	85.05 ± 0.51	91.71 ± 0.34	82.72 ± 0.58	89.00 ± 0.31
ARIEL	80.33 ± 1.25	**69.13 ± 0.94**	**88.61 ± 0.46**	**92.99 ± 0.21**	**84.43 ± 0.59**	**89.09 ± 0.31**

ARIEL is consistently robust on attacked graphs

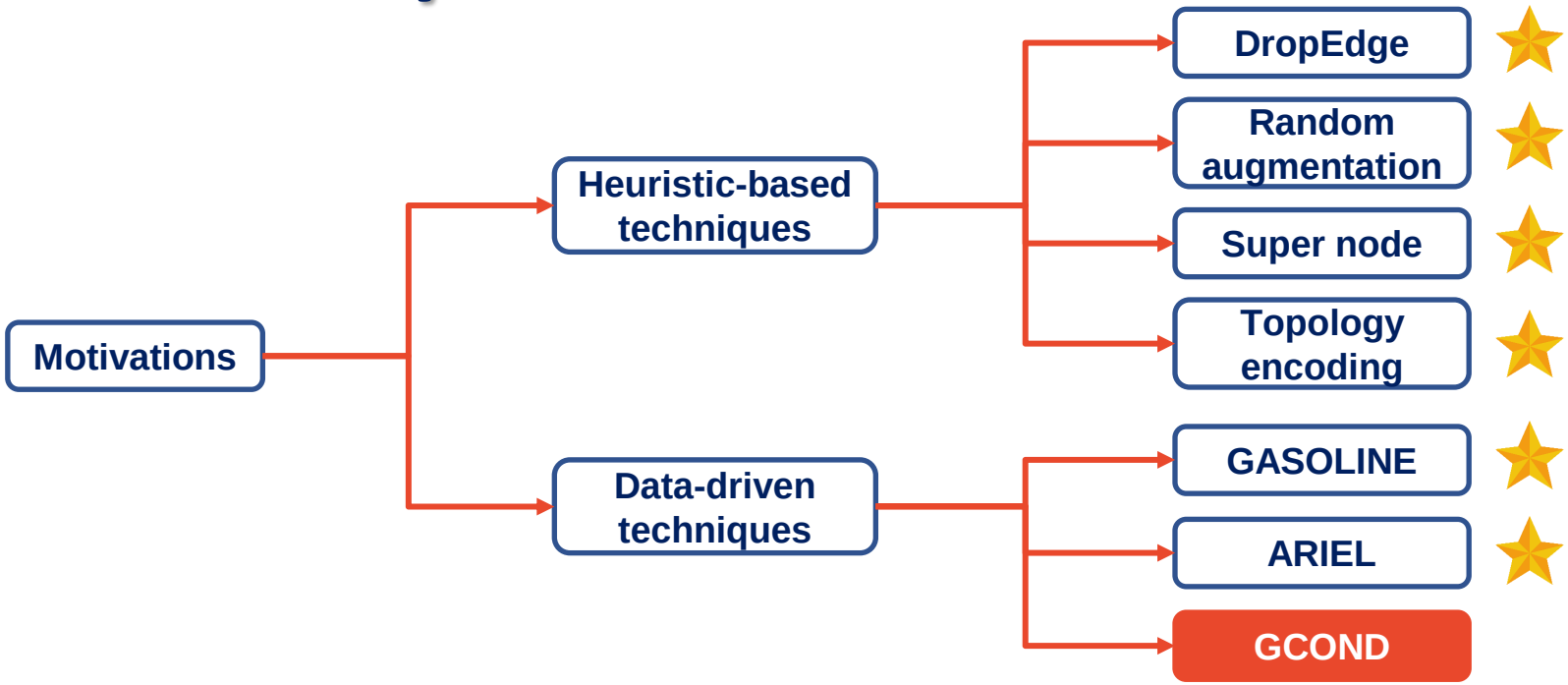
[1] Zügner et al. Adversarial Attacks on Graph Neural Networks via Meta Learning. ICLR 2019.
[2] Feng, Shengyu, et al. "Adversarial graph contrastive learning with information regularization." Proceedings of the ACM Web Conference 2022. 2022.

Related works of the ARIEL

- Graph (virtual) adversarial training, e.g., [1,2]
- Adversarial graph augmentation to improve graph contrastive learning [3]: on graph-level representation task.
- Augmenting hyper-graph for contrastive learning in a data-driven fashion [4]

[1] Kong, Kezhi, et al. "Flag: Adversarial data augmentation for graph neural networks." arXiv preprint arXiv:2010.09891 (2020).
[2] F. Feng, X. He, J. Tang, and T.-S. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. TKDE, 2019.
[3] Suresh, Susheel, et al. "Adversarial graph augmentation to improve graph contrastive learning." Advances in Neural Information Processing Systems 34 (2021): 15920-15933.
[4] Wei, Tianxin, et al. "Augmentations in Hypergraph Contrastive Learning: Fabricated and Generative." Advances in Neural Information Processing Systems, 2022.

Artificial dynamics in GNNs



Graph condensation for graph neural networks (GCOND)

Task: node classification
Artificial dynamic: synthesizing a small graph on which GNN's training gradient matches the gradient on the original graph
Goal: Save storage space and training time for graph learning

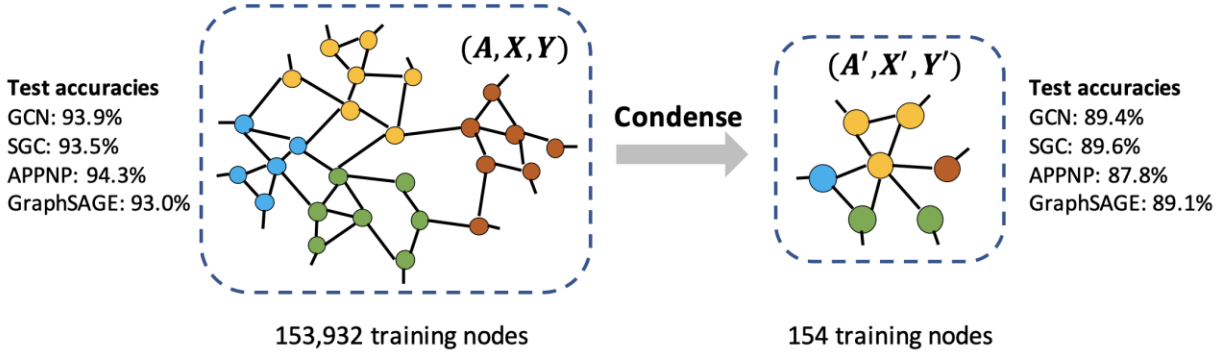


Figure credit: [1]

[1] Jin, Wei, et al. "Graph Condensation for Graph Neural Networks". International Conference on Learning Representations (ICLR). 2022

Objective function of the GCOND

Given a graph $\mathcal{T} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$, find the optimized synthetic graph $\mathcal{S} = \{\mathbf{A}', \mathbf{X}', \mathbf{Y}'\}$:

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t} Distance\left(\nabla_\theta \mathcal{L}\left(\mathrm{GNN}_{\theta_t}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right),\ \nabla_\theta \mathcal{L}(\mathrm{GNN}_{\theta_t}(\mathbf{A}, \mathbf{X}), \mathbf{Y})\right) \right]$$

GNN initializations

Training steps

Gradients on $\mathcal{S}$

Gradients on $\mathcal{T}$

The choice of Distance function is flexible. [1] choices the sum of the cosine distance of every GNN layer.

[1] Jin, Wei, et al. "Graph Condensation for Graph Neural Networks". International Conference on Learning Representations (ICLR). 2022

Optimization details of the GCOND

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_t Distance\left(\nabla_\theta \mathcal{L}\left(\mathrm{GNN}_{\theta_t}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right),\ \nabla_\theta \mathcal{L}(\mathrm{GNN}_{\theta_t}(\mathbf{A}, \mathbf{X}), \mathbf{Y})\right) \right]$$

- Mini-batch: The given graph $\mathcal{T} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$ might be huge. In practice, $\mathcal{T} = \{\mathbf{A}, \mathbf{X}, \mathbf{Y}\}$ can be replaced with a **sampled** graph $\mathcal{T}_s = \{\mathbf{A}_s, \mathbf{X}_s, \mathbf{Y}_s\}$

- The search space of $\mathbf{A}'$ is **quadratic** w.r.t. the number of nodes. To decrease the optimization variable, [1] proposes to encode $\mathbf{A}'$ as $\mathrm{MLP}_\varphi(\mathbf{X}')$:

$$\mathbf{A}'[i,j] = \mathrm{Sigmoid}\left(\frac{\mathrm{MLP}_\varphi(\mathbf{X}'[i,:], \mathbf{X}'[j:]) + \mathrm{MLP}_\varphi(\mathbf{X}'[j,:], \mathbf{X}'[i:])}{2}\right)$$

Node i/j's features

[1] Jin, Wei, et al. "Graph Condensation for Graph Neural Networks". International Conference on Learning Representations (ICLR). 2022

A graph-less version: GCOND-X

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[\sum_{t} Distance\left(\nabla_\theta \mathcal{L}\left(\mathrm{GNN}_{\theta_t}(\mathbf{A}', \mathbf{X}'), \mathbf{Y}'\right),\ \nabla_\theta \mathcal{L}(\mathrm{GNN}_{\theta_t}(\mathbf{A}, \mathbf{X}), \mathbf{Y})\right) \right]$$

- Explicitly set $\mathbf{A}'$ as $\mathbf{I}$
- A “special” representation learning method. Not only the topology information is encoded, but the # of nodes is also condensed.

[1] Jin, Wei, et al. "Graph Condensation for Graph Neural Networks". International Conference on Learning Representations (ICLR). 2022

Empirical performance of the GCOND

Table 2: GCOND and GCOND-X achieves promising performance in comparison to baselines even with extremely large reduction rates. We report transductive performance on Citeseer, Cora, Ogbn-arxiv; inductive performance on Flickr, Reddit. Performance is reported as test accuracy (%).

		Baselines					Proposed		
Dataset	Ratio (r)	Random ($\mathbf{A}', \mathbf{X}'$)	Herding ($\mathbf{A}', \mathbf{X}'$)	K-Center ($\mathbf{A}', \mathbf{X}'$)	Coarsening ($\mathbf{A}', \mathbf{X}'$)	DC-Graph ($\mathbf{X}'$)	GCOND-X ($\mathbf{X}'$)	GCOND ($\mathbf{A}', \mathbf{X}'$)	Whole Dataset
Citeseer	0.9%	54.4±4.4	57.1±1.5	52.4±2.8	52.2±0.4	66.8±1.5	**71.4**±0.8	70.5±1.2	71.7±0.1
	1.8%	64.2±1.7	66.7±1.0	64.3±1.0	59.0±0.5	66.9±0.9	69.8±1.1	**70.6**±0.9	
	3.6%	69.1±0.1	69.0±0.1	69.1±0.1	65.3±0.5	66.3±1.5	69.4±1.4	**69.8**±1.4	
Cora	1.3%	63.6±3.7	67.0±1.3	64.0±2.3	31.2±0.2	67.3±1.9	75.9±1.2	**79.8**±1.3	81.2±0.2
	2.6%	72.8±1.1	73.4±1.0	73.2±1.2	65.2±0.6	67.6±3.5	75.7±0.9	**80.1**±0.6	
	5.2%	76.8±0.1	76.8±0.1	76.7±0.1	70.6±0.1	67.7±2.2	76.0±0.9	**79.3**±0.3	
Ogbn-arxiv	0.05%	47.1±3.9	52.4±1.8	47.2±3.0	35.4±0.3	58.6±0.4	**61.3**±0.5	59.2±1.1	71.4±0.1
	0.25%	57.3±1.1	58.6±1.2	56.8±0.8	43.5±0.2	59.9±0.3	**64.2**±0.4	63.2±0.3	
	0.5%	60.0±0.9	60.4±0.8	60.3±0.4	50.4±0.1	59.5±0.3	63.1±0.5	**64.0**±0.4	
Flickr	0.1%	41.8±2.0	42.5±1.8	42.0±0.7	41.9±0.2	46.3±0.2	45.9±0.1	**46.5**±0.4	47.2±0.1
	0.5%	44.0±0.4	43.9±0.9	43.2±0.1	44.5±0.1	45.9±0.1	45.0±0.2	**47.1**±0.1	
	1%	44.6±0.2	44.4±0.6	44.1±0.4	44.6±0.1	45.8±0.1	45.0±0.1	**47.1**±0.1	
Reddit	0.05%	46.1±4.4	53.1±2.5	46.6±2.3	40.9±0.5	88.2±0.2	**88.4**±0.4	88.0±1.8	93.9±0.0
	0.1%	58.0±2.2	62.7±1.0	53.0±3.3	42.8±0.8	89.5±0.1	89.3±0.1	**89.6**±0.7	
	0.2%	66.3±1.9	71.0±1.6	58.5±2.1	47.4±0.9	**90.5**±1.2	88.8±0.4	90.1±0.5	

- Empirically, GCOND is the state-of-the-art method for condensing a given graph.
- Even the structure-free version GCOND-X can get decent performance.

Jin, Wei, et al. "Graph Condensation for Graph Neural Networks". International Conference on Learning Representations (ICLR). 2022

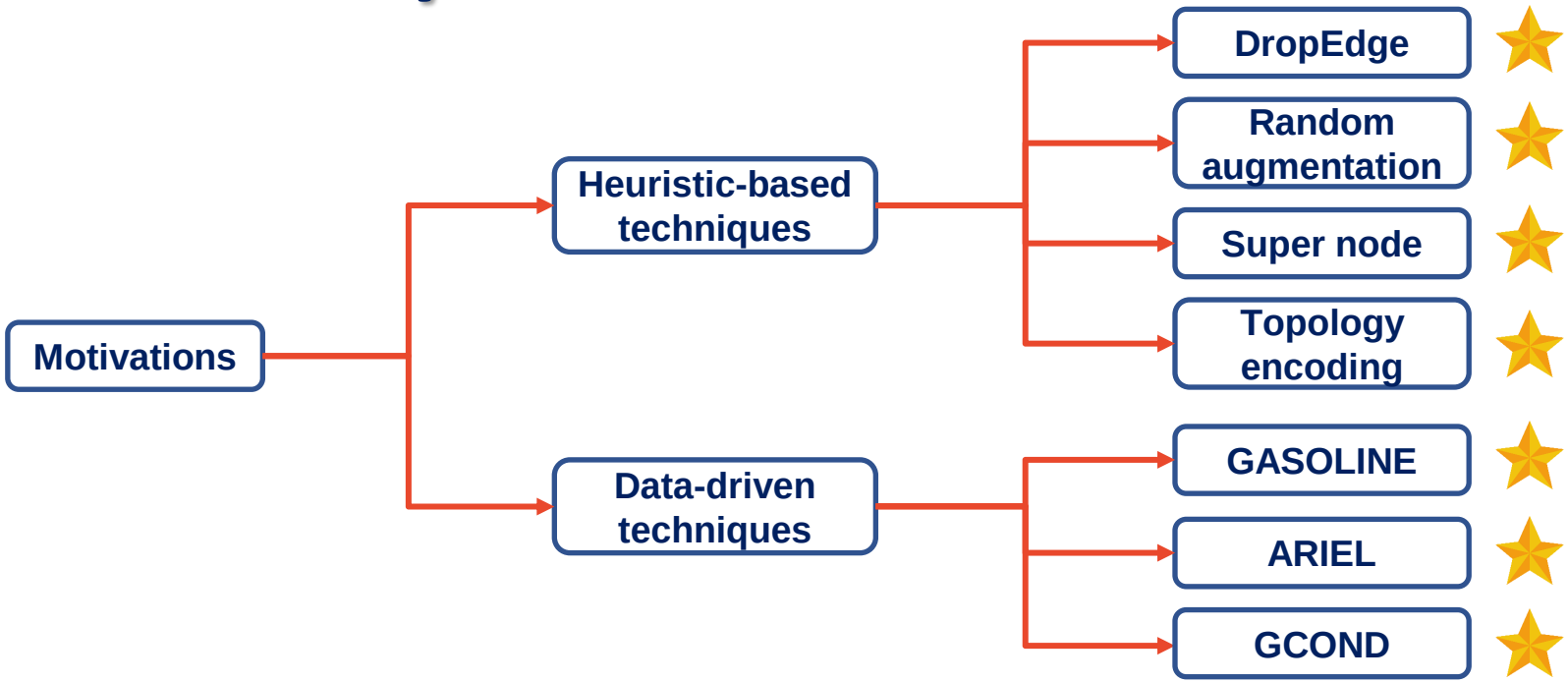
Applicability of the GCOND

Table 3: Graph condensation can work well with different architectures. Avg. stands for the average test accuracy of APPNP, Cheby, GCN, GraphSAGE and SGC. SAGE stands for GraphSAGE.

	Methods	Data	MLP	GAT	APPNP	Cheby	GCN	SAGE	SGC	Avg.
Citeseer $r = 1.8\%$	DC-Graph	$\mathbf{X}'$	66.2	-	66.4	64.9	66.2	65.9	69.6	66.6
	GCOND-X	$\mathbf{X}'$	69.6	-	69.7	70.6	69.7	69.2	71.6	70.2
	GCOND	$\mathbf{A}', \mathbf{X}'$	63.9	55.4	69.6	68.3	70.5	66.2	70.3	69.0
Cora $r = 2.6\%$	DC-Graph	$\mathbf{X}'$	67.2	-	67.1	67.7	67.9	66.2	72.8	68.3
	GCOND-X	$\mathbf{X}'$	76.0	-	77.0	74.1	75.3	76.0	76.1	75.7
	GCOND	$\mathbf{A}', \mathbf{X}'$	73.1	66.2	78.5	76.0	80.1	78.2	79.3	78.4
Ogbn-arxiv $r = 0.25\%$	DC-Graph	$\mathbf{X}'$	59.9	-	60.0	55.7	59.8	60.0	60.4	59.2
	GCOND-X	$\mathbf{X}'$	64.1	-	61.5	59.5	64.2	64.4	64.7	62.9
	GCOND	$\mathbf{A}', \mathbf{X}'$	62.2	60.0	63.4	54.9	63.2	62.6	63.7	61.6
Flickr $r = 0.5\%$	DC-Graph	$\mathbf{X}'$	43.1	-	45.7	43.8	45.9	45.8	45.6	45.4
	GCOND-X	$\mathbf{X}'$	42.1	-	44.6	42.3	45.0	44.7	44.4	44.2
	GCOND	$\mathbf{A}', \mathbf{X}'$	44.8	40.1	45.9	42.8	47.1	46.2	46.1	45.6
Reddit $r = 0.1\%$	DC-Graph	$\mathbf{X}'$	50.3	-	81.2	77.5	89.5	89.7	90.5	85.7
	GCOND-X	$\mathbf{X}'$	40.1	-	78.7	74.0	89.3	89.3	91.0	84.5
	GCOND	$\mathbf{A}', \mathbf{X}'$	42.5	60.2	87.8	75.5	89.4	89.1	89.6	86.3

- The condensed graph from GCOND and GCOND-X can be applied to various downstream models.

Artificial dynamics in GNNs



Contents

- Part I – Introduction ☑
- Part II – Natural Dynamics in GNNs ☑
- 30 mins Coffee Break ☑
- Part III – Artificial Dynamics in GNNs ☑
- Part IV – Challenges and Future Directions
- Q&A

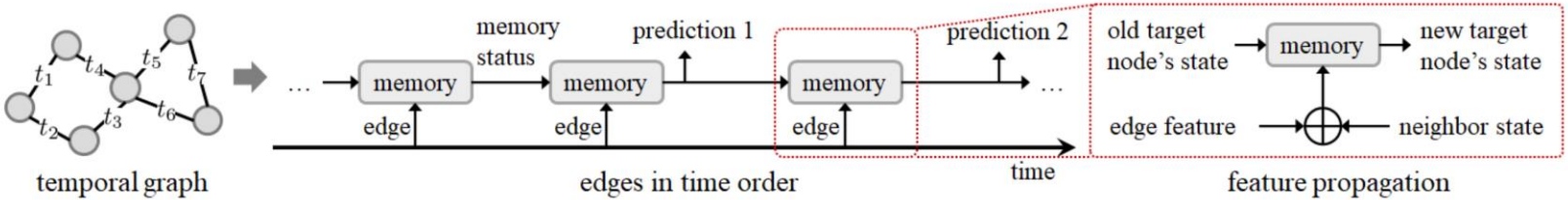
Open questions (1)

Question 1: how to augment the temporal graphs, i.e., the temporal or the topological features?

Recap: Encoding temporal graphs [1]

$$\mathbf{s}_i(t) = \mathrm{mem}\big(\mathbf{s}_i(t^-), \{\mathbf{e}_{ij}(t) \oplus \mathbf{s}_j(t^-) \mid j \in N(i)\}\big)$$

Node i's state at time t | Update function | Edge attribute at time t | Node j's state before time t



Wang, Yiwei, et al. "Adaptive data augmentation on temporal graphs." Advances in Neural Information Processing Systems 34 (2021): 1440-1452.

Adaptive data augmentation on temporal graphs (MeTA)

Task: Edge prediction and node classification
(Heuristic) artificial dynamics: (1) perturbing time: $t_{new} = t + \tau$, $\tau \sim N(0, \sigma^2)$, (2) edge dropping, (3) edge adding with noisy time
Goal: improving temporal GNNs' performance

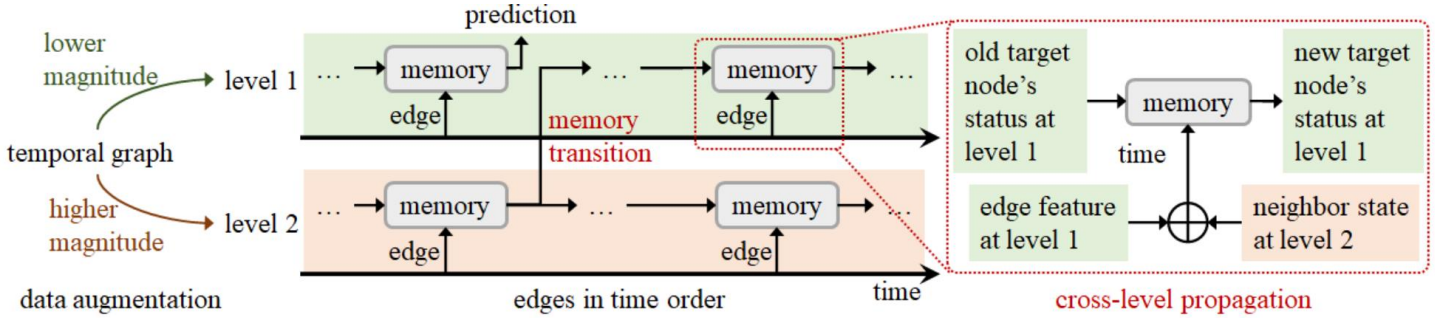
Key idea: (1) augmenting both the topological component and the time features; (2) merging representations from graphs with different augmentation magnitudes

Wang, Yiwei, et al. "Adaptive data augmentation on temporal graphs." Advances in Neural Information Processing Systems 34 (2021): 1440-1452.

Details of MeTA

E.g., perturb 10% edges (less random)

E.g., perturb 80% edges (more random)



$$\mathbf{s}_i^{(l)}(t) = \mathsf{mem}\left(\mathbf{s}_i^{(l)}(t^-), \mathbf{e}_{ij}^{(l)}(t) \| \mathbf{s}_j^{(\min(l+1,L))}(t^-)\right),$$

Node embedding from the less random graph

"adaptive"

Node embedding from the more random graph

Wang, Yiwei, et al. "Adaptive data augmentation on temporal graphs." Advances in Neural Information Processing Systems 34 (2021): 1440-1452.

Applicability of MeTA

Table 2: Test accuracy and average precision (AP) of transductive edge prediction. We conduct 100 trials with random weight initialization. Mean (%) and standard deviations are reported. The best results in each column are highlighted in **bold** font.

Method	MOOC		Reddit		Wikipedia	
	Accuracy	AP	Accuracy	AP	Accuracy	AP
CTDNE [21]	65.34 ± 0.7	74.29 ± 0.6	73.76 ± 0.5	91.41 ± 0.3	79.42 ± 0.4	92.17 ± 0.5
JODIE [16]	76.45 ± 0.6	83.87 ± 0.4	90.91 ± 0.3	97.11 ± 0.3	87.04 ± 0.4	94.62 ± 0.5
TGAT [39]	75.20 ± 0.5	82.66 ± 0.4	92.92 ± 0.3	98.12 ± 0.2	88.14 ± 0.2	95.34 ± 0.1
DyRep [28]	73.36 ± 0.4	81.75 ± 0.3	92.11 ± 0.2	97.98 ± 0.1	87.77 ± 0.2	94.59 ± 0.2
TGN [25]	81.38 ± 0.6	89.79 ± 0.5	92.56 ± 0.2	98.70 ± 0.1	89.51 ± 0.4	98.46 ± 0.1
DyRep + MeTA (Ours)	76.21 ± 0.4	84.18 ± 0.3	93.04 ± 0.3	98.62 ± 0.1	88.92 ± 0.2	95.63 ± 0.2
TGN + MeTA (Ours)	**83.84 ± 0.5**	**92.03 ± 0.3**	**94.19 ± 0.2**	**99.08 ± 0.1**	**91.34 ± 0.3**	**98.87 ± 0.1**

- The method can work with different backbone models (e.g., DyRep and TGN)
- It can work on both the edge-level tasks and the node-level tasks (omitted in this slides for brevity)

Wang, Yiwei, et al. "Adaptive data augmentation on temporal graphs." Advances in Neural Information Processing Systems 34 (2021): 1440-1452.

Open questions (2)

Question 2: how to augment the temporal graph in a data-driven way?
Recap: [1] perturb the edge time heuristically: $t_{new} = t + \tau$, $\tau \sim N(0, \sigma^2)$.
Challenge: If we aim to augment (i.e., optimize) the edge time in a data-driven way (e.g., gradient-based method), a given optimization objective must be differentiable w.r.t. the "time" parameters.

$$\mathbf{s}_i(t) = \mathrm{mem}\left(\mathbf{s}_i(t^-), \{\mathbf{e}_{ij}(t) \oplus \mathbf{s}_j(t^-) \,\middle|\, j \in N(i)\}\right)$$

Node i's state at time t | Update function | Edge attribute at time t | Node i's state before time t

Open questions (3)

Question 3: How to ensure augmentation efficiency?

Challenges:

- Even if we do not augment the time parameter and view the temporal graph as a sequence of static graph snapshots, the method running time is linear w.r.t. # of time steps.
- The ”optimal” augmented graph snapshots should be **related**. How can we **incrementally** get the optimal graph snapshots from previous time steps?

Open questions (4)

Question 4: How to generate new graph snapshots given the previously observed ones?

Challenges:

- Highly open and there are many unknown hyper-parameters. E.g., How many graphs are generated? What is the size of generated graphs?
- How to ensure the performance of downstream tasks is improved given the generated graphs?

Takeaways

- **Introduction**
- **Natural Dynamics in GNNs**
- **Artificial Dynamics in GNNs**
  - Heuristic-based techniques
  - Data-driven techniques
- **Open Questions and Challenges**
  - Augmentation of both the topological and temporal components
  - Augmentation the temporal modality in a data-driven way
  - How to maintain efficiency

Resources

- **Surveys**
  - Dongqi Fu and Jingrui He: Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future. Frontiers in Big Data 2022
  - Ding, Kaize, et al. "Data augmentation for deep graph learning: A survey." ACM SIGKDD Explorations Newsletter 24.2 (2022): 61-77.
  - Zhao, Tong, et al. "Graph data augmentation for graph machine learning: A survey." arXiv preprint arXiv:2202.08871 (2022).
- **Related Tutorials**
  - Rozenshtein, Polina, and Aristides Gionis. "Mining temporal networks." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.
  - Zhang, Chuxu, Jundong Li, and Meng Jiang. "Data Efficient Learning on Graphs." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.
  - Jin, Wei, et al. "Graph representation learning: foundations, methods, applications and systems." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.

Contents

- Part I – Introduction ☑
- Part II – Natural Dynamics in GNNs ☑
- 30 mins Coffee Break ☑
- Part III – Artificial Dynamics in GNNs ☑
- Part IV – Challenges and Future Directions ☑
- Q&A

THE 16TH ACM INTERNATIONAL CONFERENCE ON
WEB SEARCH AND DATA MINING
wsdm 2023
SINGAPORE FEBRUARY 27 - MARCH 3

Thanks!




Dongqi Fu
UIUC

Zhe Xu
UIUC

Hanghang Tong
UIUC

Jingrui He
UIUC

{dongqif2, zhexu3, htong, jingrui}@illinois.edu

https://github.com/DongqiFu/Natural-and-Artificial-Dynamics-in-GNNs-A-Tutorial

NSF
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
IDEA