Dynamic Graphs

Guest Lecture by Farimah Poursafaei

Oct. 18th, 2022

Outline

- **Introduction to dynamic graphs**
- Encoder-decoder framework
  - Encoders for dynamic graphs
  - Decoders for dynamic graphs
- Dynamic graph learning challenges
- An Application: dynamic link prediction
- Conclusion

Temporal or Dynamic Graphs

- Graphs:
  - Vectors of features + relations that form **graphs** among entities
- **Static** graphs:
  - Nodes and edges are **fixed** and do not change over time

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \mathcal{V} = \left\{ v_1, v_2, \ldots, v_{|V|} \right\} \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$$

- **Dynamic** graphs:
  - A graph where nodes or edges *appear and/or disappear* over time
  - Have both structural and temporal patterns
    - Continuous-time dynamic graph (CTDG)
    - Discrete-time dynamic graphs (DTDG)

Dynamic Graph Storage Model

- Graph storage model ordered by temporal granularity:
  - **Static**
    - Has no temporal information
  - **Edge-weighted**
    - Temporal information included as labels on the edges and/or nodes of a static graph
  - **Discrete**
    - Represented in discrete-time intervals
    - Represented by multiple snapshots of the graph at different time intervals
  - **Continuous**
    - Has no temporal aggregation applied
    - Carries the most information but also the most complex

CTDGs

- Continuous-Time Dynamic Graph (CTDG): $\left(\mathcal{G},\ \mathcal{O}\right)$
  - $\mathcal{G} \rightarrow an\ initial\ state\ of\ a\ dynamic\ graph\ at\ t_0$
  - $\mathcal{O} \rightarrow a\ set\ of\ observation/events\ of\ the\ form\ (event\ type,\ event,\ timestamp)$
- A graph snapshot: $\mathcal{G}^t$ at any point $t \geq t_0$ in time
  - Obtained from a CTDG by updating $\mathcal{G}$ sequentially according to the observations $\mathcal{O}$ th

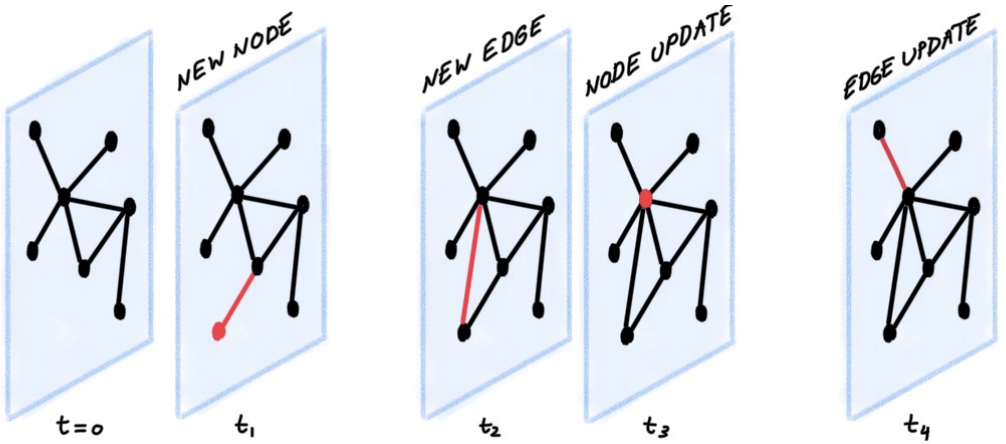


Fig. ref.: Temporal Graph Network

DTDGs

- Discrete-Time Dynamic Graph (DTDG):
  - A sequence of snapshots from a dynamic graph
  - Sampled at regularly-spaced times
  - $\{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$; $\mathcal{G}^t = \{\mathcal{V}^t, \mathcal{E}^t\}$: graph snapshot at t.
- A DTDG may lose information compared to CTDG!

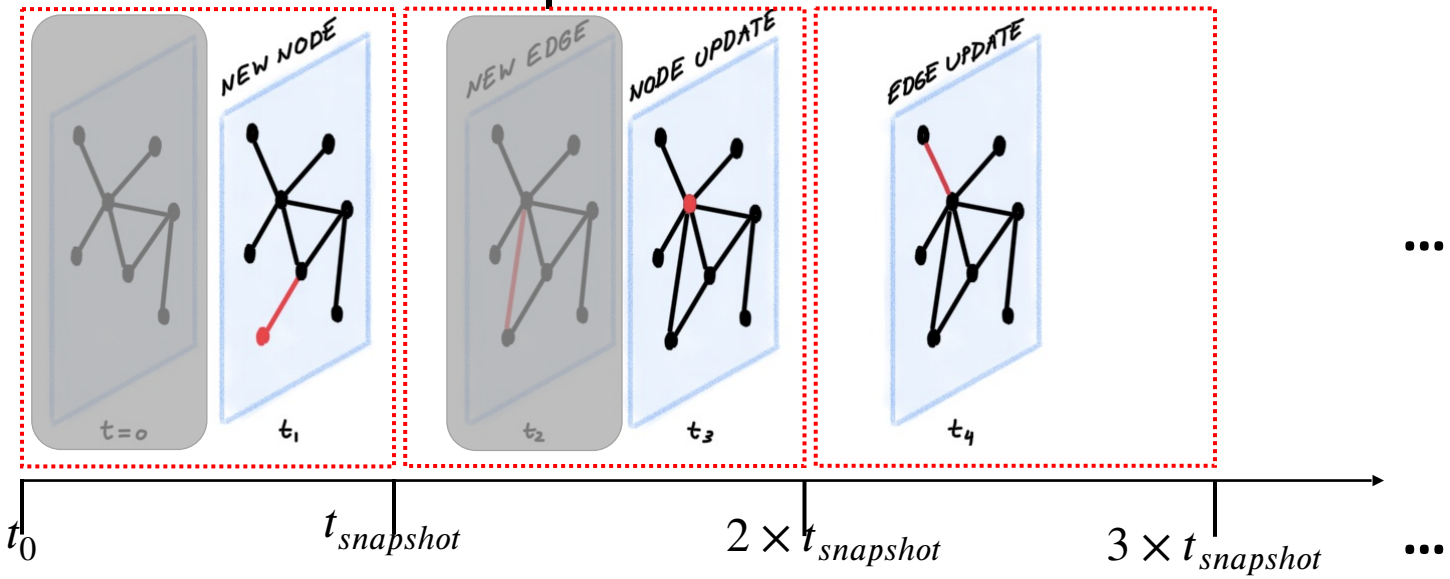
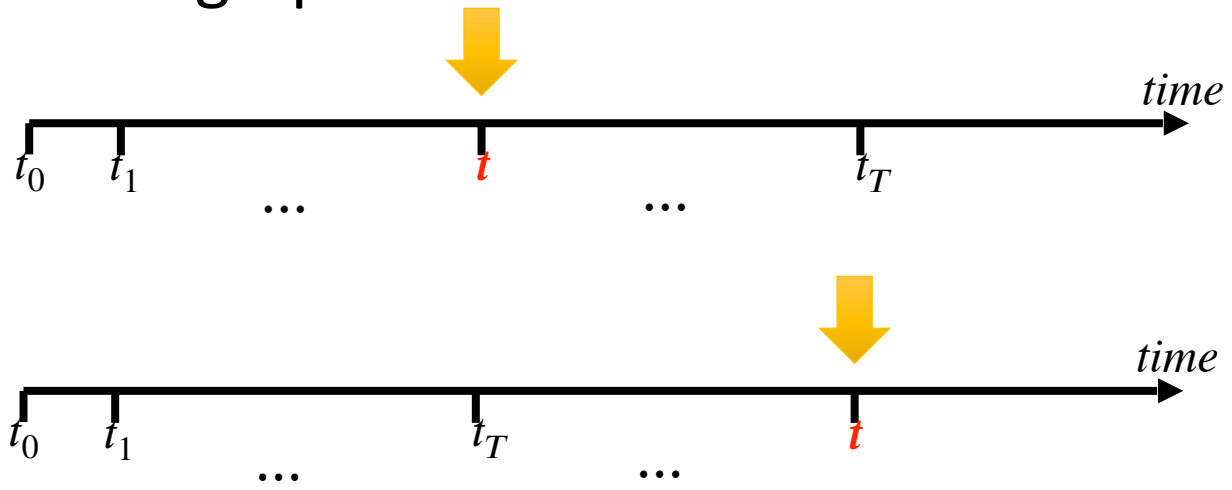


Fig. ref.: Temporal Graph Network

Prediction Problem

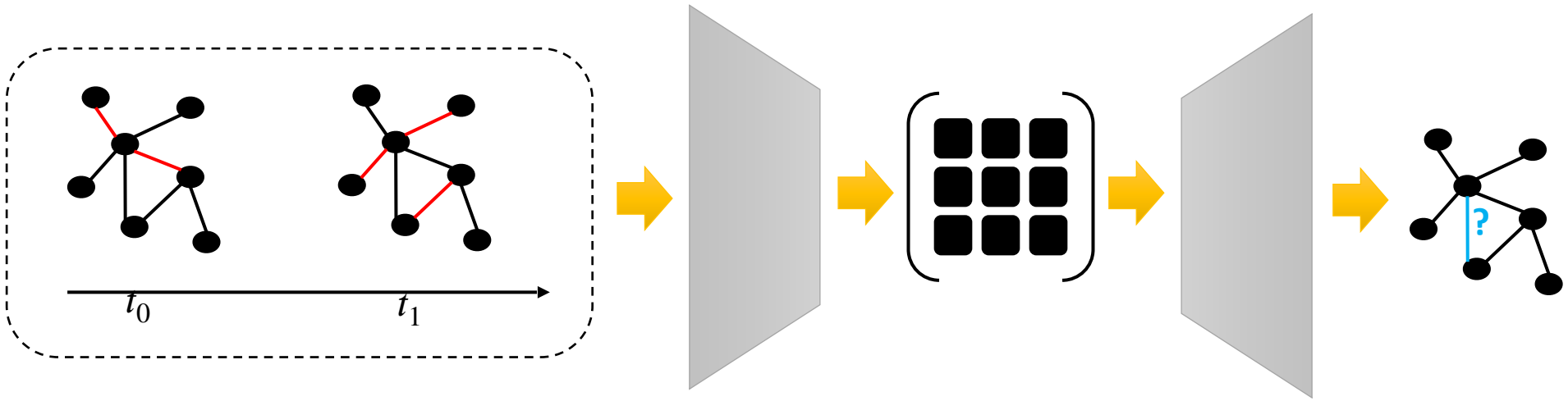
- General problems for dynamic graphs:
  - Node classification
  - Edge prediction
  - Graph classification
- Settings for reasoning over dynamic graphs:
  - Interpolation:
  - Extrapolation:



Outline

- Introduction to dynamic graphs
- **Encoder-decoder framework**
  - Encoders for dynamic graphs
  - Decoders for dynamic graphs
- Dynamic graph learning challenges
- An Application: dynamic link prediction
- Conclusion

Encoder-Decoder Framework (*Hamilton et al. 2017*)



Observations of a dynamic graph | encoder | embeddings | decoder | prediction

Fig. ref.: *Zhu et al. 2022*

Outline

- Introduction to dynamic graphs
- Encoder-decoder framework
  - **Encoders for dynamic graphs**
  - Decoders for dynamic graphs
- Dynamic graph learning challenges
- An Application: dynamic link prediction
- Conclusion

Encoder for Dynamic Graphs

- Aggregating Temporal Observations
- Aggregating Static Features
- Time as a Regularizer
- Random Walk Encoders
- Sequence Model Encoders
- Dynamic Graph Neural Networks (DGNNs)

Aggregating Temporal Observations

- Aggregating **temporal observations** over time
  - Generate a static graph
  - Generate embeddings by a static encoder
- Examples:
  - Ignore the timestamps and take the sum (or union) of the entries of the adjacency matrices across all snapshots:

$$A_{sum}[i][j] = \sum_{t=1}^{T} A^t[i][j]$$

  - To give more weight to snapshots that are more recent:

$$A_{wsum}[i][j] = \sum_{t=1}^{T} \theta^{T-t} A^t[i][j]$$

Aggregating Static Features

- In case of a DTDG $\left\{\mathcal{G}^1,\ \mathcal{G}^2,\ \ldots,\ \mathcal{G}^T\right\}$:
  - Apply a static encoder to each snapshot
  - Aggregate the results over time
- Example:
  - Exponentially decaying older features:

$$z_v = \sum_{t=1}^{T} \exp\left(-\theta(T-t)\right) z_v^t$$

  - Fit a time-series model to the features from previous snapshots

Time as a Regularizer

- Consider a DTDG $\{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$:
  - Embedding of a node v at $(t-1)^{th}$ snapshot: $EMB^{t-1}(v) = \left(z_v^{t-1}\right)$
  - Use a static encoder to learn an embedding function for $\mathcal{G}^t$
  - **Additional constraint**:
    - ***Smoothness*** *constraint*: for any node that has been in the graph in the **previous** and **current** timestamps, $dist(z_v^{t-1}, z_v^t)$ should be small.
    - A common choice for the distance function is the *Euclidean* distance:

$$dist\left(z_v^{t-1}, z_v^t\right) = \| \left| z_v^t - Z_v^{t-1} \right| \|$$

Random Walk Encoders

- Consider a DTDG $\{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$:
- *Dynnode2vec* (Mahdavi et al. 2018)*:*
  - Generate random walks on $\mathcal{G}^1$ and feed them to $\mathcal{M}^1$ to get representations
  - For t^{th} $(t > 1)$ snapshot:
    - Keep the ***valid* random walks** from $(t-1)^{th}$ snapshot
      - *Valid random walk*: all nodes and edges are still in the graph in the t^{th} snapshot
    - ***New random walk*** starts from *affected nodes*:
      - New nodes or nodes involved in added or deleted edges
    - Initialize $\mathcal{M}^t$ with learned parameters from $\mathcal{M}^{t-1}$ and generate representations

Sequence-Model Encoders

- RNN-based Encoders for DTDGs
  - A synchronous sequence modeling problem
  - The duration between any two consecutive items in the sequence is equal
- RNN-based Encoders for CTDGs
  - An asynchronous sequence modeling problem

RNN-based Encoders for DTDGs

- Consider a DTDG $\{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$:
  - An encoder $\mathcal{M}$ that takes a static graph $\mathcal{G}^t$ and returns vector representation for each node.
  - Run $\mathcal{M}$ on each $\mathcal{G}^t$ and obtain a sequence $z_v^1, z_v^2, \ldots, z_v^T$ for node v.
  - Feed the sequence to an RNN to produce a representation z_v containing the information from v's history and evolution.

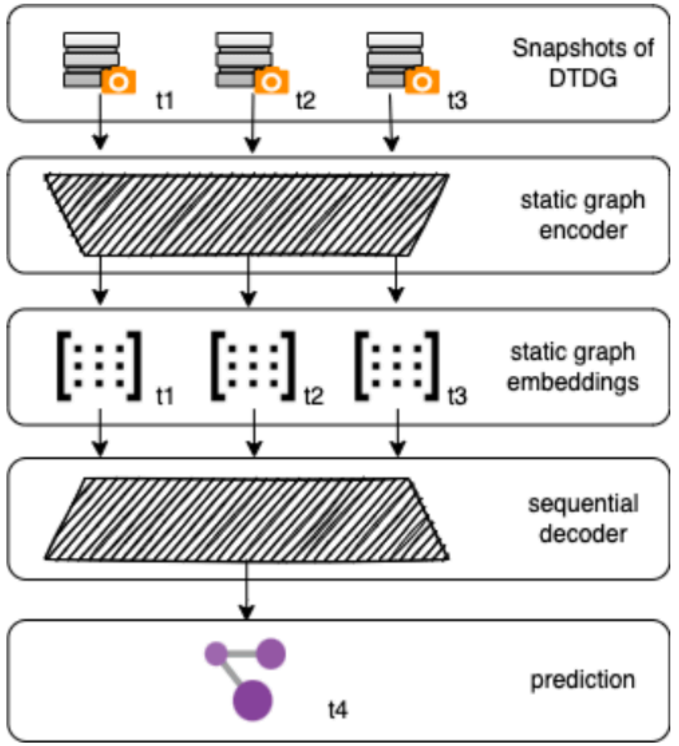


Fig. ref.: *Zhu et al. 2022*

RNN-based Encoders for CTDGs

- So far, mainly CTDGs with only *edge addition events* are considered!
- Consist of **custom RNNs** that update the representations of the source and target nodes that form a new edge upon making a new observation.
- Differences in these approaches:
  - How they define the embedding function
  - How they define the custom RNN
- **Staleness**
  - An encoder that updates embedding whenever *a new observation* is made
  - The last update of embedding of v was at t_v
  - Depending on how long it's passed since t_v $\left(t - t_v\right)$, the v embedding may be staled!

Dynamic Graph Neural Networks (DGNNs)

- A neural network architecture that:
  - The *aggregation of neighboring node features* is part of the NN architecture.
  - Encodes both *structural* and *temporal* patterns in dynamic graphs
  - Structural patterns encoding:
    - ***Graph Neural Network (GNN)***
  - Temporal patterns encoding: ***a time-series module*** like:
    - ***RNN*** or ***positional attention***

Discrete DGNNs

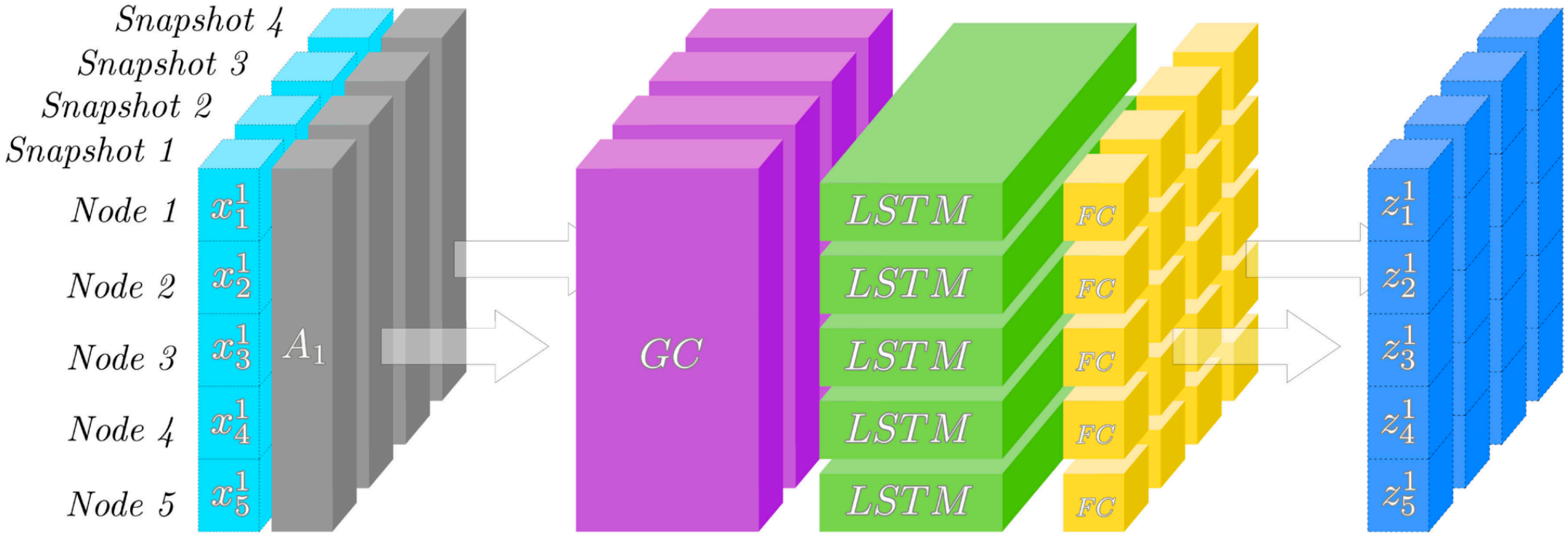
- **Advantage**:
  - Static graph model can be used on each snapshot
- Combines a time-series model with a GNN
  - Consider a DTDG $\{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$:

$$Z^t = GNN(\mathcal{G}^t)$$

$$H^t = f\left(H^{t-1}, Z^t\right) \quad for\ i \in [1, n]$$

  - GNN is used to encode each graph snapshot
  - f (the RNN or self-attention) encodes across the snapshots

Discrete DGNNs: Stacked DGNNs

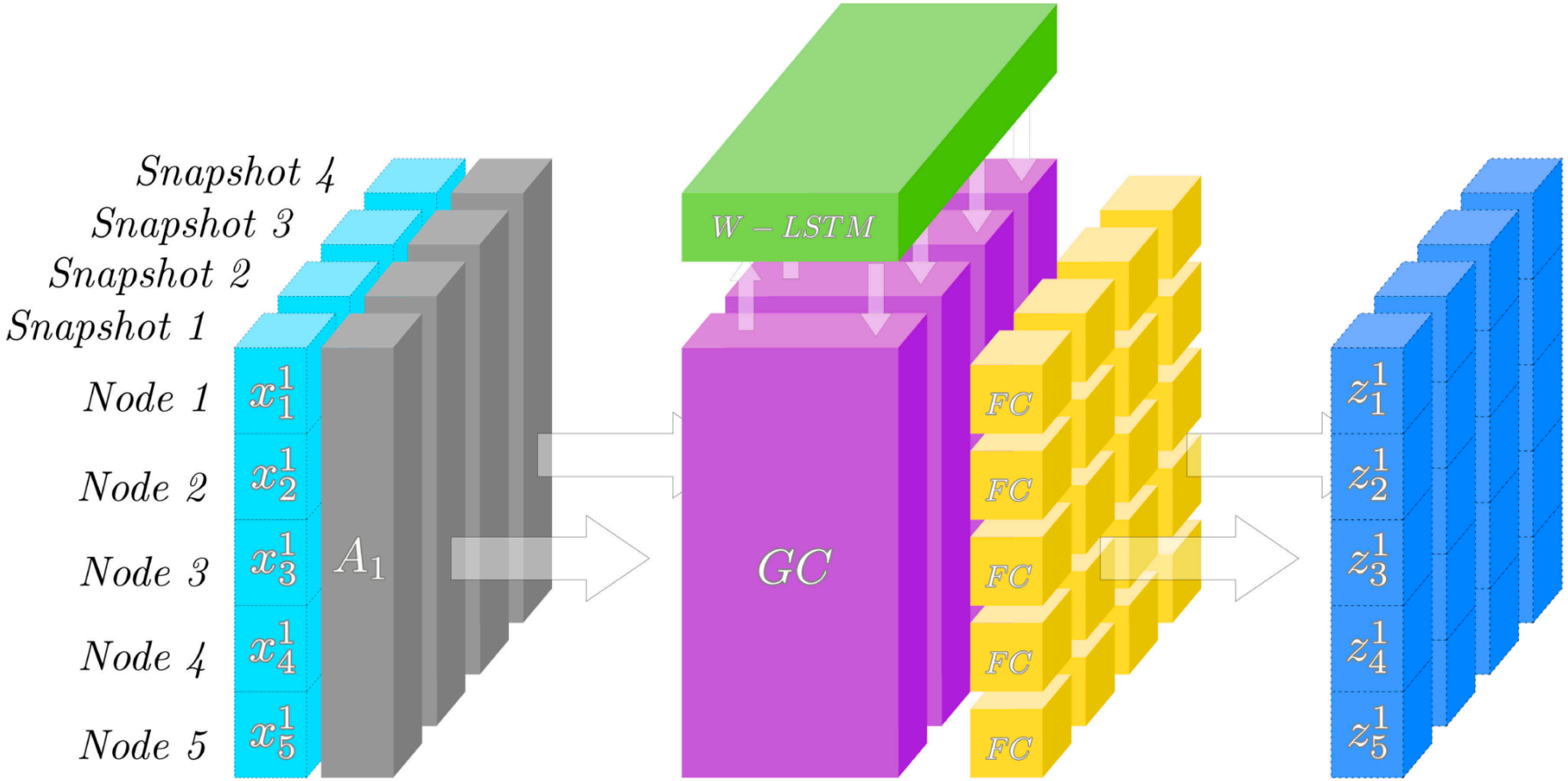


Stacked DGNN structure from *Manessi et al. 2020.*

The graph convolution layer (GC) encodes the graph structure in each snapshot while the LSTMs encode temporal patterns.

Fig. ref.: *Skarding et al. 2021*

Discrete DGNNs: Integrated DGNNs



Integrated DGNN structure of *EvolveGCN* from *Pareja et al. 2020*.
The EGCU-O layer constitutes the GC and the W-LSTM which is used to initialize the weights of the GC.

Fig. ref.: *Skarding et al. 2021*

Continuous DGNNs

- There are three DGNN approaches to CTDGs:
  - **1. RNN-based Models**
    - Use RNNs to maintain node embeddings in a continuous fashion
    - The embeddings of the affected nodes are updated *as soon as there is an event*
      - → The embeddings stay up to date continuously

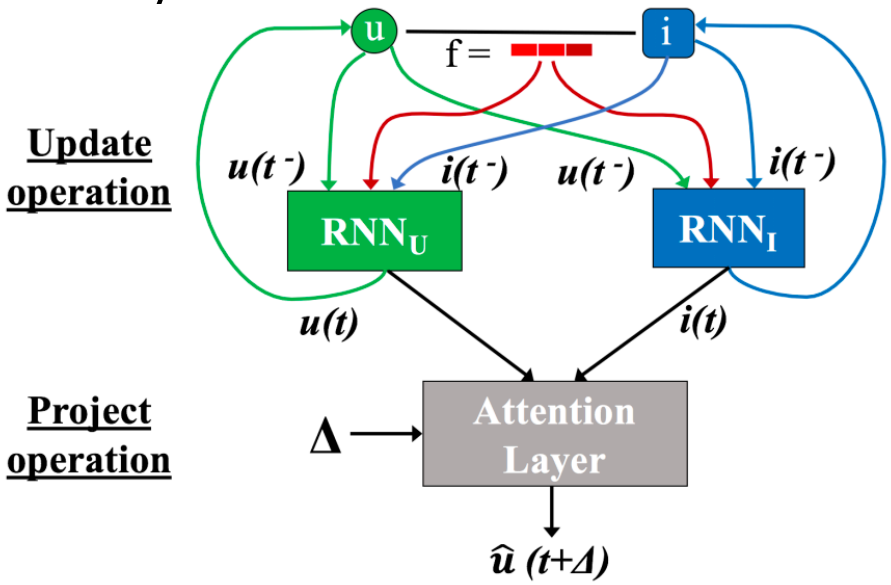


Fig. ref.: Kumar, Srijan, Xikun Zhang, and Jure Leskovec. "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks." In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1269-1278. 2019.

Continuous DGNNs

- There are three DGNN approaches to CTDGs:

2. Temporal Point Process (TPP)-based Models

- *Know-Evolve (Trivedi et al. 2017)*
  - Models an interaction network by parametrizing a TPP by a modified RNN
- *DyRep (Trivedi et al. 2019)*
  - Uses a TPP model which is parameterized by a recurrent architecture

3. Time Embedding Approaches

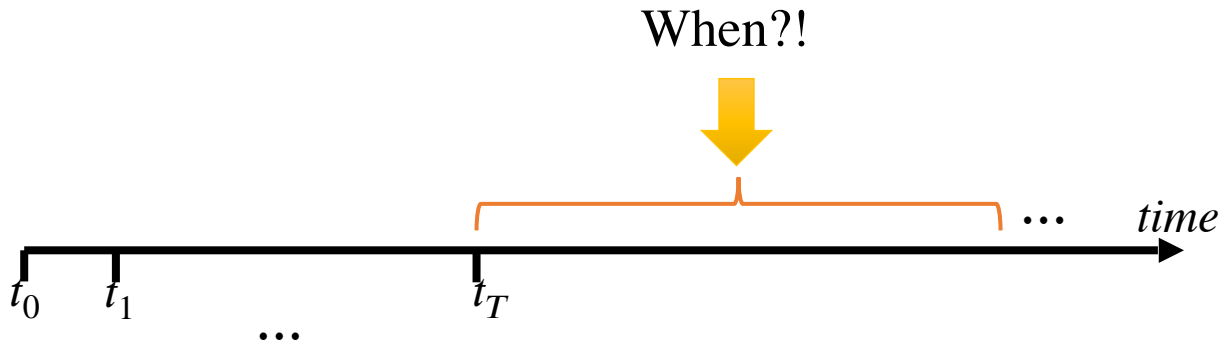
- Rely on **time embedding** methods; e.g., *Time2vec (Kazemi et al. 2019)*
- Time embedding is a positional encoding focused on encoding temporal patterns
- Aim at capturing temporal time difference $t_i - t_j$ → **capture inter-event time**

Outline

- Introduction to dynamic graphs
- Encoder-decoder framework
  - Encoders for dynamic graphs
  - **Decoders for dynamic graphs**
- Dynamic graph learning challenges
- An Application: dynamic link prediction
- Conclusion

Decoders for Dynamic Graphs

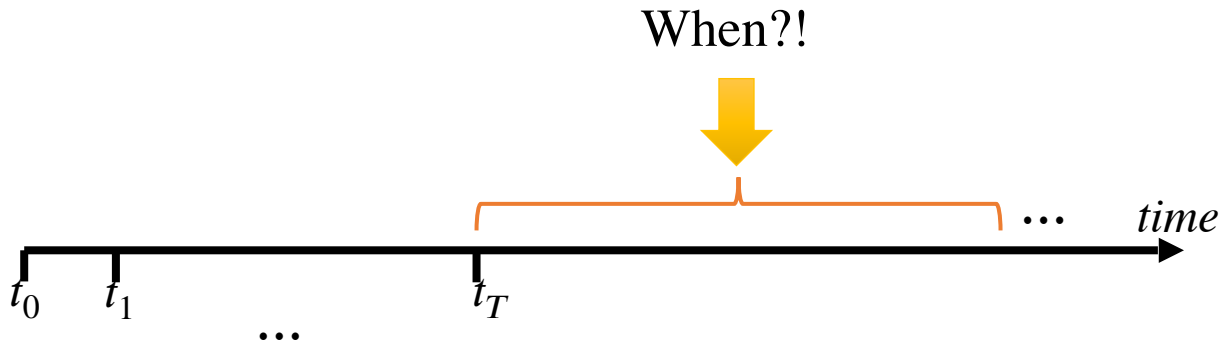
- Time-Predicting Decoders:



- Time-Conditioned Decoders:

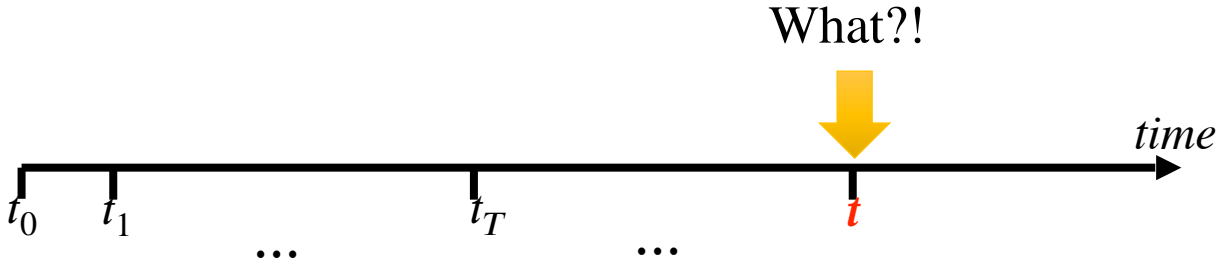
Time-Predicting Decoders

- For extrapolation or interpolation
- In extrapolation:
  - Predict when an event will happen
- In interpolation:
  - Predict a missing timestamp



Time-Conditioned Decoders

- **Goal**:
  - To make predictions for specific timestamps
- For extrapolation or interpolation
- In extrapolation:
  - E.g., *who will be the CEO of Apple two years from now?*
- In interpolation:
  - E.g., *who was the CEO of Apple at a specific time in the past?*



Outline

- Introduction to dynamic graphs
- Encoder-decoder framework
  - Encoders for dynamic graphs
  - Decoders for dynamic graphs
- **Dynamic graph learning challenges**
- An Application: dynamic link prediction
- Conclusion

Dynamic Graph Learning Challenges (You et al. 2022)

- Limitation of ***model design***
  - *Skip-connections*, *batch normalization*, and *edge embedding* are beneficial for *GNN message passing* **but** unexplored for DGNNs
  - Start from a mature static GNN and adapt it to dynamic graphs
- Limitation of **evaluation setting**
  - Existing literature ignores the evolving nature of data and models
    - Overestimate the model performance given the presence of long-term pattern changes
    - Models do not update within the time span of evaluation → and get ***stale*** over time!
- Limitation of **training strategies**
  - Mostly require keeping the entire graph in GPU memory!
  - Thus, DGNNs are often evaluated on small networks

Outline

- Introduction to dynamic graphs
- Encoder-decoder framework
  - Encoders for dynamic graphs
  - Decoders for dynamic graphs
- Dynamic graph learning challenges
- **An Application: dynamic link prediction**
- Conclusion

An Application: Dynamic Link Prediction

- Learning on dynamic graphs is ***challenging***:
  - Several evolving elements: node/edge attributes, graph structure, …
- One important task: dynamic link prediction
  - Given a timestamped stream of edges:

$$G = \{(s_1, d_1, t_1), (s_2, d_2, t_2), \ldots\}; \qquad 0 \leq t_1 \leq t_2 \leq \ldots \leq T$$

  - Objective:
    - Predicting the existence of an edge between a pair of nodes in the future
- **Remarkable observation**:
  - SOTA methods have **near-perfect performance** for dynamic link prediction!

Dynamic Link Prediction Challenges (Poursafaei et al. 2022)

- **Limited domain diversity**
  - Existing datasets are mostly social interaction networks.
  - Networks across different domain exhibit a diverse set of properties.
  - It is necessary to test dynamic link prediction in various domains.
- **Easy negative edges**
  - Positive examples correspond to the actual links.
  - Edges that have never been seen previously are ***easy*** negative edges.
  - It's important to evaluate methods on different sets of negative samples!
- **Memorization works well!**
  - Making predictions based on seen edges achieves high performance!

Understanding Dynamic Graph Datasets

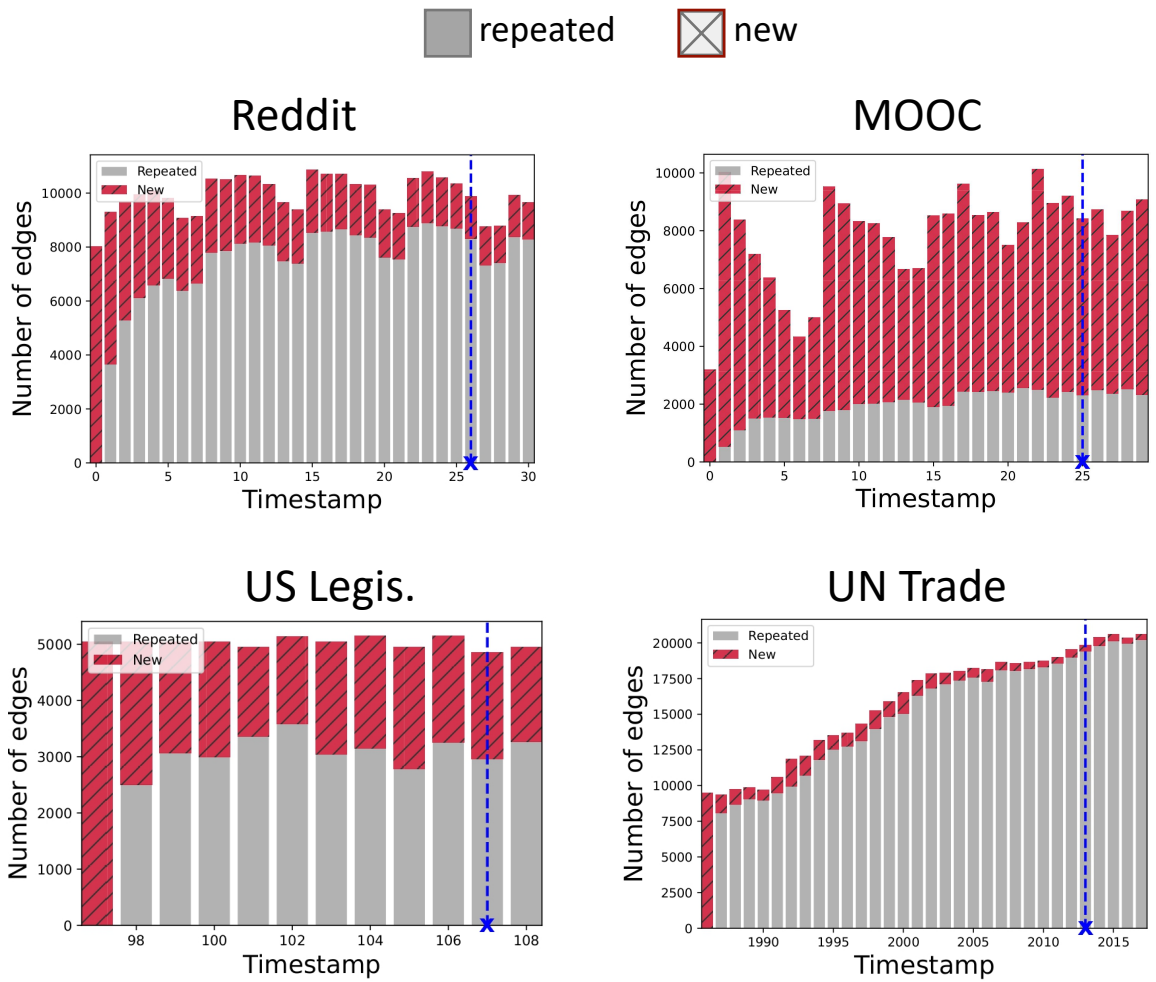
- How to characterize dynamic graph datasets?
  - By statistics presented in a table?!

Table 1: Dataset statistics.

Dataset	Domain	# Nodes	Total Edges	Unique Edges	Unique Steps	Time Granularity	Duration
Wikipedia	Social	9,227	157,474	18,257	152,757	Unix timestamp	1 month
Reddit	Social	10,984	672,447	78,516	669,065	Unix timestamp	1 month
MOOC	Interaction	7,144	411,749	178,443	345,600	Unix timestamp	17 month
LastFM	Interaction	1,980	1,293,103	154,993	1,283,614	Unix timestamp	1 month
Enron	Social	184	125,235	3,125	22,632	Unix timestamp	3 years
Social Evo.	Proximity	74	2,099,519	4,486	565,932	Unix timestamp	8 months
UCI	Social	1,899	59,835	20,296	58,911	Unix timestamp	196 days
Flights (new)	Transport	13,169	1,927,145	395,072	122	days	4 months
Can. Parl. (new)	Politics	734	74,478	51,331	14	years	14 years
US Legis. (new)	Politics	225	60,396	26,423	12	congresses	12 congresses
UN Trade (new)	Economics	255	507,497	36,182	32	years	32 years
UN Vote (new)	Politics	201	1,035,742	31,516	72	years	72 years
Contact (new)	Proximity	694	2,426,280	79,531	8,065	5 minutes	1 month

- But…
  "a picture is worth a thousand words"!

Dynamic Graphs: Temporal Evolutionary Pattern



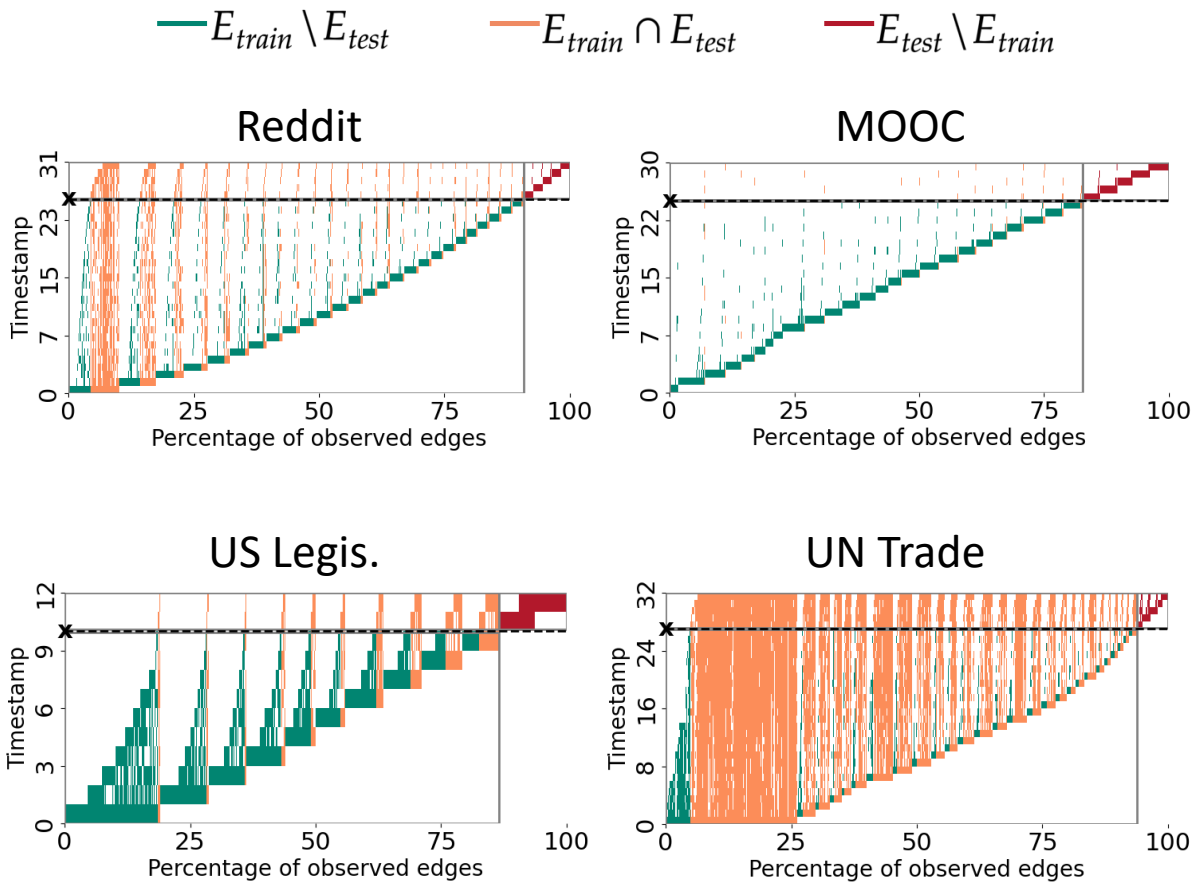
TEA Plots

- Temporal Edge Appearance (TEA)
  - Repeated vs. new edges
- High variance in temporal evolutionary patterns.
- A simple memorization approach:
  - Good for repeated edges.
  - Bad for new edges.

Dynamic Graphs: Consistency of Edge Repeats

TET Plots

- Temporal Edge Traffic (TET)
- Recurrence pattern of edges over time
- A simple memorization approach:
  - Positive transductive edges
    - ✓ Consistent recurrence
  - × Positive inductive edges



EdgeBank: A Baseline for Dynamic Link Prediction

- A **pure memorization-based** approach: A ***bank*** of observed edges
- Two memory update strategies:
  - $EdgeBank_{\infty}$ → stores ***all*** observed edges
  - $EdgeBank_{tw}$ → only remembers edges from ***a fixed time window***

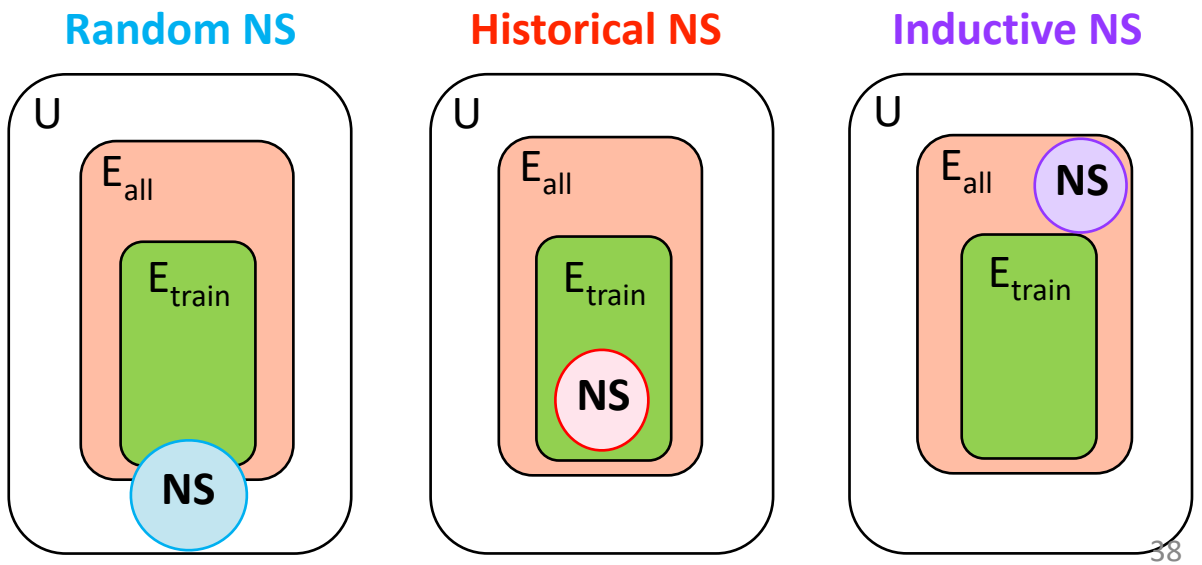
✅ **Good** for edges with *frequent reoccurrence* pattern

❌ **Bad** for :

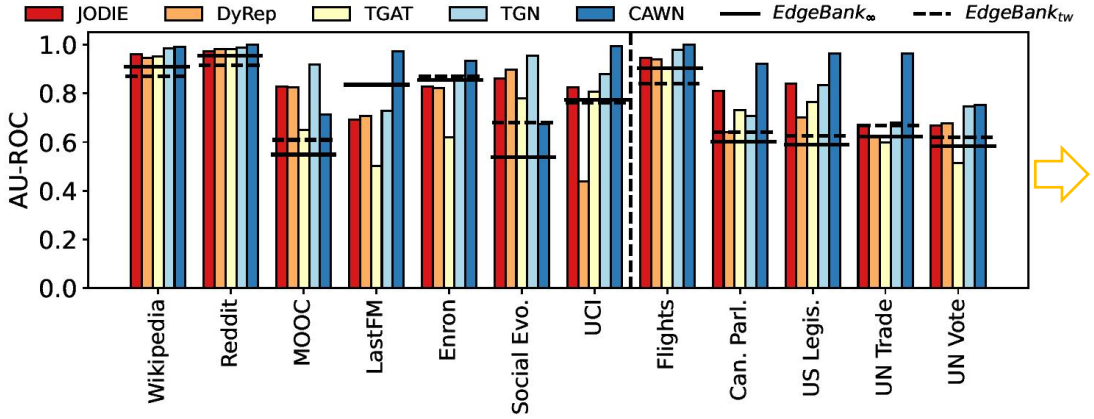
- An unseen positive edge
- A previously seen negative edge

Revisiting Negative Sampling in Dynamic Graphs

- Link prediction: modeled as a binary classification problem
  - We only have positive samples!
  - Negative Sampling (NS) is needed to train/evaluate the models.
- **Random negative sampling**: The ***standard*** negative sampling strategy
  - × No collision checking
  - × No reoccurring edges
- Historical negative sampling
- Inductive negative sampling

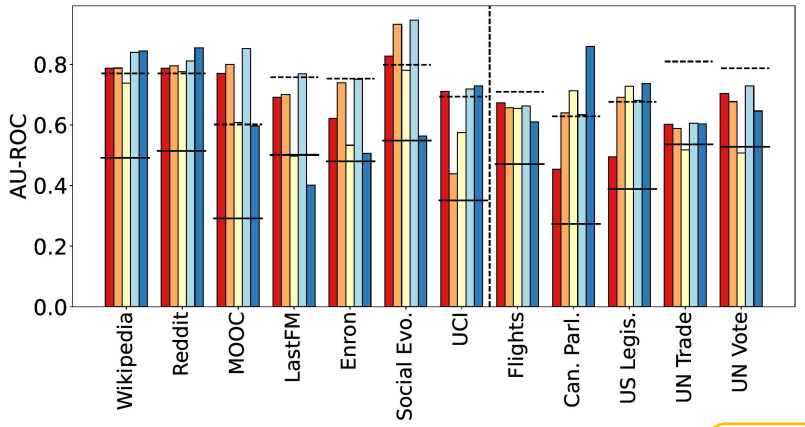


Performance Evaluation

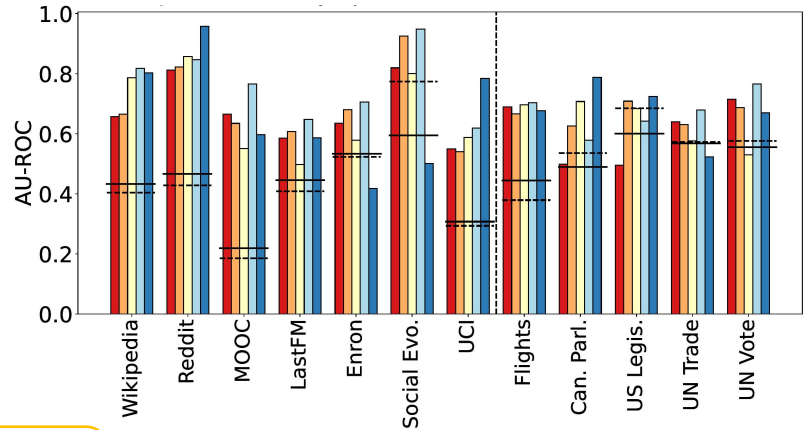


Standard **random** NS.

- ***Inconsistency*** in ranking among methods across datasets.
- Memorization baselines are ***on par*** with SOTAs.



Historical NS.



Inductive NS.

- A clear ***gap*** between SOTAs & EdgeBank.
- The ranking of the models changes.

Reproducibility

- A link to the paper:
  - https://arxiv.org/pdf/2207.10128.pdf
- Code repository:
  - https://github.com/fpour/DGB
- All datasets are hosted online:
  - https://zenodo.org/record/7008205#.Y03Qqi8r1hC

- Please feel free to reach out for any question or discussion… 😊
  - ✉️ → farimah.poursafaei@mila.quebec
  - Or on slack!

Outline

- Introduction to dynamic graphs
- Encoder-decoder framework
  - Encoders for dynamic graphs
  - Decoders for dynamic graphs
- Dynamic graph learning challenges
- An Application: dynamic link prediction
- **Conclusion**

Conclusion

- Preliminaries of dynamic/temporal graphs
- Dynamic graph storage model
- Continuous-time vs. discrete-time dynamic graphs
- Prediction problem: interpolation vs. extrapolation
- Encoder-decoder framework
- Encoders for dynamic graphs:
  - Aggregating Temporal Observations
  - Aggregating Static Features
  - Time as a Regularizer
  - Random Walk Encoders
  - Sequence Model Encoders
  - Dynamic Graph Neural Networks (DGNNs)

Conclusion

- Decoders for dynamic graphs
  - Time-predicting decoders
  - Time-conditioned decoders
- Dynamic graph learning challenges
- An application: dynamic link prediction
  - Towards better evaluation for dynamic link prediction
    - Limited domain diversity
    - Easy negatives
    - Memorization works well

Thank you!

Any question?

Reference

- Kazemi, Seyed Mehran, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. "Representation Learning for Dynamic Graphs: A Survey." *J. Mach. Learn. Res.* 21, no. 70 (2020): 1-73.
- Skarding, Joakim, Bogdan Gabrys, and Katarzyna Musial. "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey." *IEEE Access* 9 (2021): 79143-79168.
- Zhu, Yuecai, Fuyuan Lyu, Chengming Hu, Xi Chen, and Xue Liu. "Encoder-Decoder Architecture for Supervised Dynamic Graph Learning: A Survey." CoRR (2022).
- Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." arXiv preprint arXiv:1709.05584 (2017).
- Manessi, Franco, Alessandro Rozza, and Mario Manzo. "Dynamic Graph Convolutional Networks." Pattern Recognition97 (2020): 107000.
- Pareja, Aldo, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 5363-5370. 2020.
- Trivedi, Rakshit, Hanjun Dai, Yichen Wang, and Le Song. "Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs." In international conference on machine learning, pp. 3462-3471. PMLR, 2017.
- Kazemi, Seyed Mehran, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. "Time2vec: Learning a Vector Representation of Time." arXiv preprint arXiv:1907.05321 (2019).
- You, Jiaxuan, Tianyu Du, and Jure Leskovec. "ROLAND: Graph Learning Framework for Dynamic Graphs." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2358-2366. 2022.
- Poursafaei, Farimah, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. "Towards Better Evaluation for Dynamic Link Prediction." arXiv preprint arXiv:2207.10128 (2022).
- Mahdavi, Sedigheh, Shima Khoshraftar, and Aijun An. "dynnode2vec: Scalable Dynamic Network Embedding." In 2018 IEEE international conference on big data (Big Data), pp. 3762-3765. IEEE, 2018.
- Trivedi, Rakshit, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. "DyRep: Learning Representations over Dynamic Graphs." In International conference on learning representations. 2019.