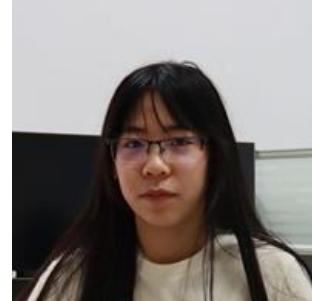


Towards Efficient Temporal Graph Learning: Algorithms, Frameworks, and Tools



Ruijie Wang



Wanyu Zhao



Dachun Sun



Charith Mendis Tarek Abdelzaher



University of Illinois Urbana-Champaign

{ruijiew2, wanyu2, dsun18, charithm, zaher}@illinois.edu

Time: 1:45 PM – 17:30 PM, October 21, 2024

Location: Room 120C, Boise Centre, Boise, ID

Webpage: <https://wjerry5.github.io/cikm2024-tutorial/>

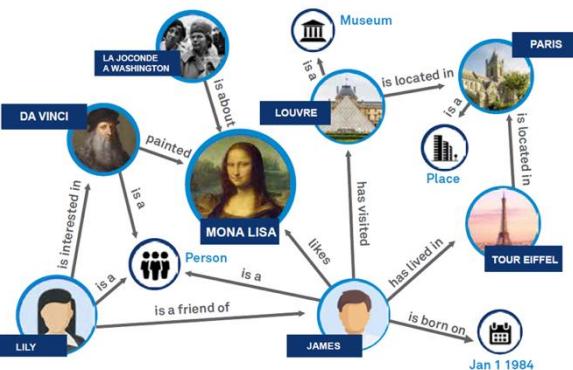
Contents

- **Part I – Introduction**
- **Part II – Data-Efficient Temporal Graph Neural Network**
- **30-min Coffee Break**
- **Part III – Resource-Efficient Temporal Graph Neural Network**
- **Part IV – Discussion and Future Directions**

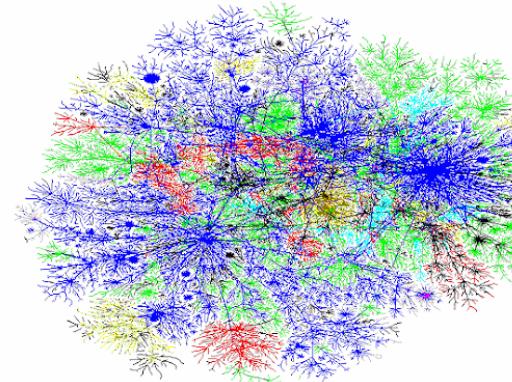
Broad Application Domains of Graph Data



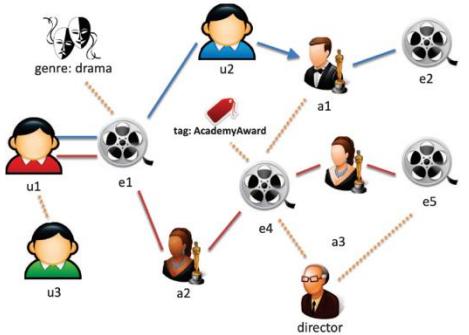
Social Network Analysis



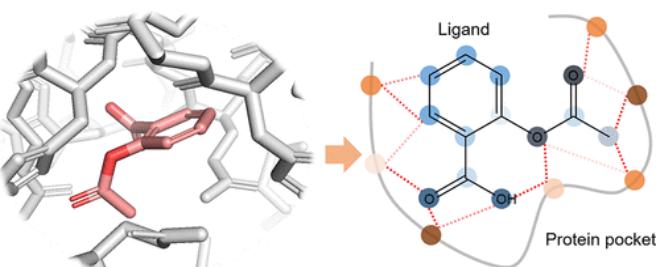
Knowledge Graph Reasoning



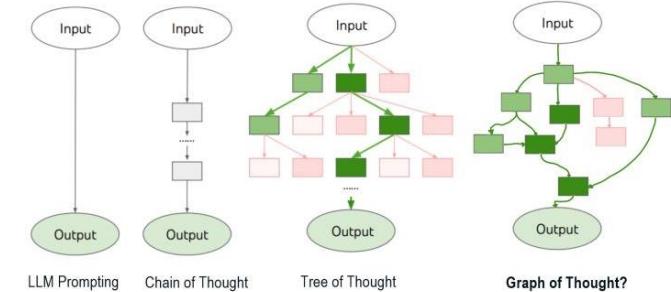
Web Mining



Recommendation



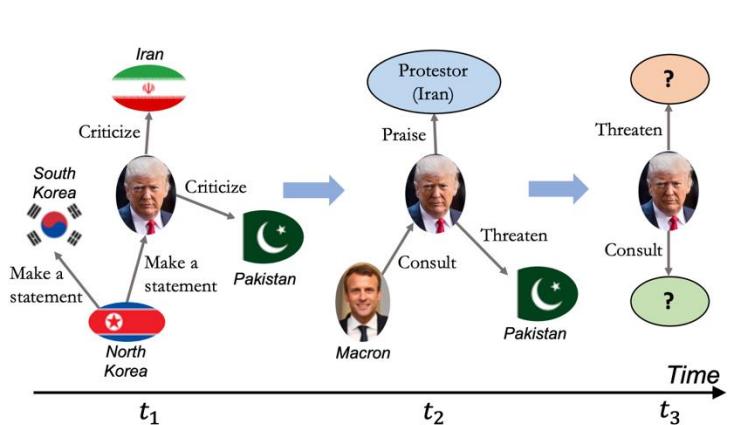
Scientific Discovery



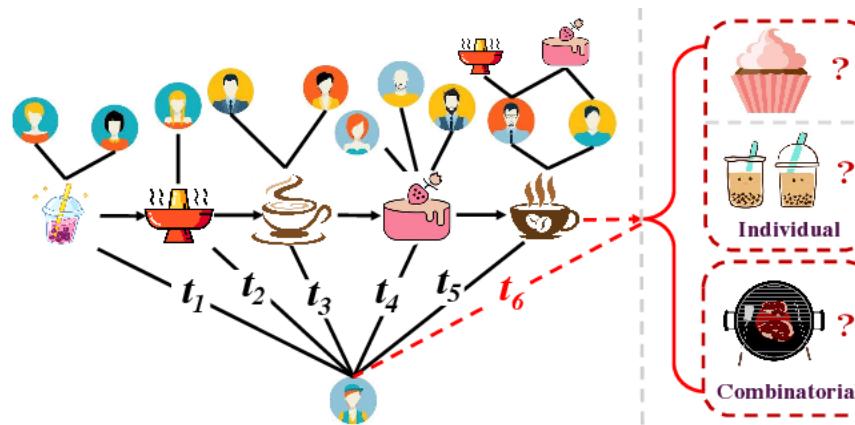
LLM Prompting & Reasoning

Universal language for describing interconnected data!

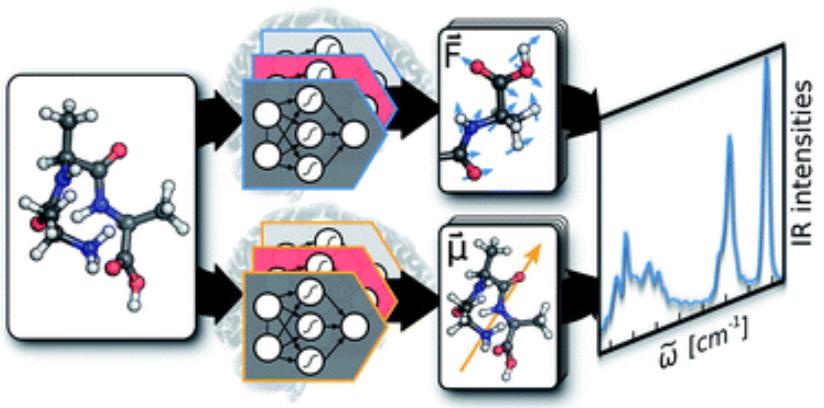
Real-World Graphs are Evolving – Temporal Graphs



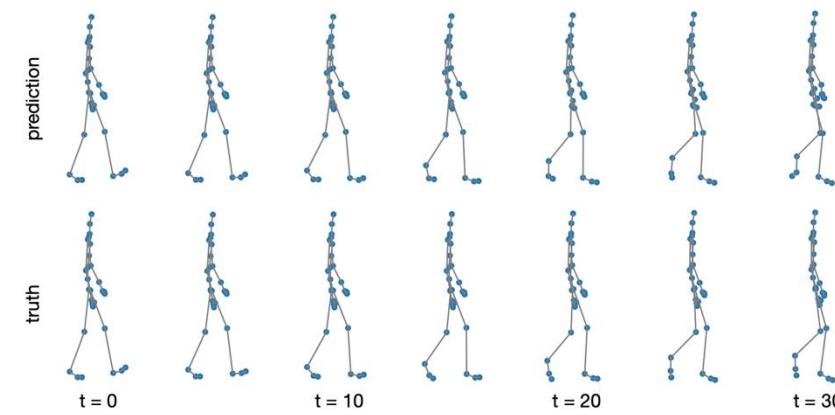
Temporal Facts in KGs



User Online Behaviors



Molecular Dynamics

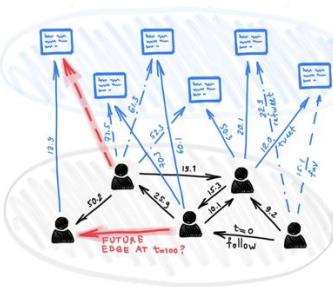


Dynamical Systems

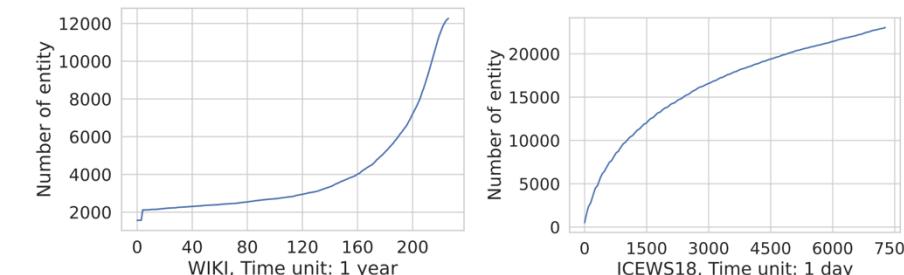
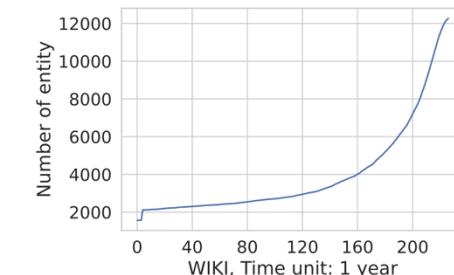
Real-World Graphs are Evolving – Temporal Graphs

- Graphs have time-evolving components, e.g.,

- Topology structures
 - Add/delete nodes
 - Add/delete edges

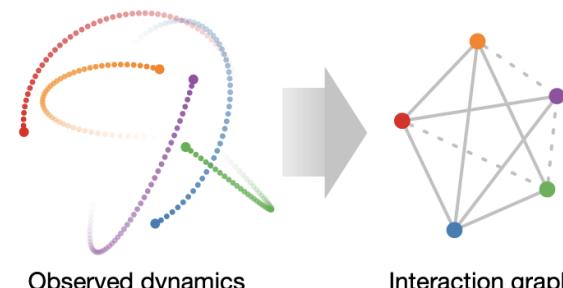


Dynamic edges [1]



Dynamic node set [2]

- Input features
 - Node-level features
 - Edge-level features
 - ...



Dynamic node&edge features [3]

[1] <https://towardsdatascience.com/temporal-graph-networks-ab8f32712efe>.

[2] Wang et. al., Learning to Sample and Aggregate: Few-shot Reasoning over Temporal Knowledge Graphs.

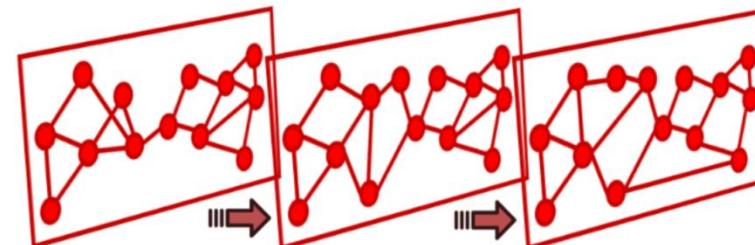
[3] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems.

Temporal Graphs – Definition

- Discrete-time vs. continuous-time temporal graphs

- Discrete-time temporal graphs

- $\mathcal{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_{T-1}, \mathcal{G}_T\}$,
- where $\mathcal{G}_t = (\mathcal{E}_t, \mathcal{V}_t, \mathcal{X}_t)$ denotes one snapshot.

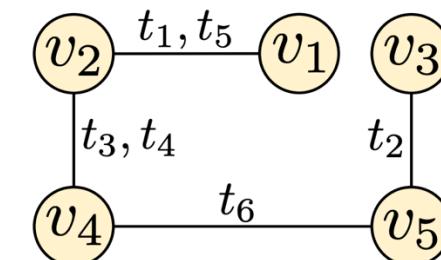


Discrete-time example [1]

- Continuous-time temporal graphs

- $G = \{(e_i, e_j, t, +/ -)\}$,
- where $e_i, e_j \in \mathcal{E}, 0 \leq t < T$

(v_1, v_2) at time t_1, t_5
 (v_3, v_5) at time t_2
 (v_2, v_4) at time t_3, t_4
 (v_4, v_5) at time t_6

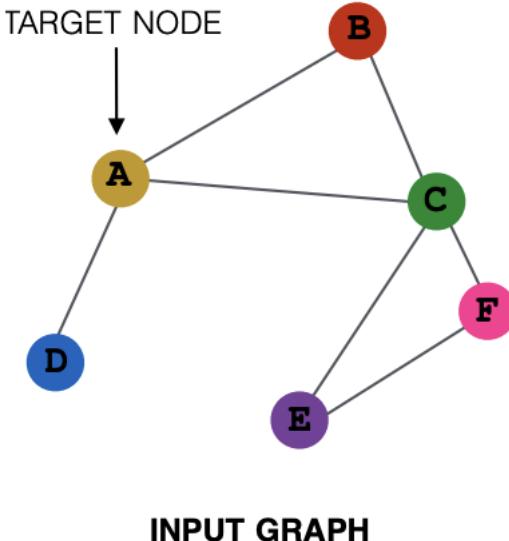


Continuous-time example [2]

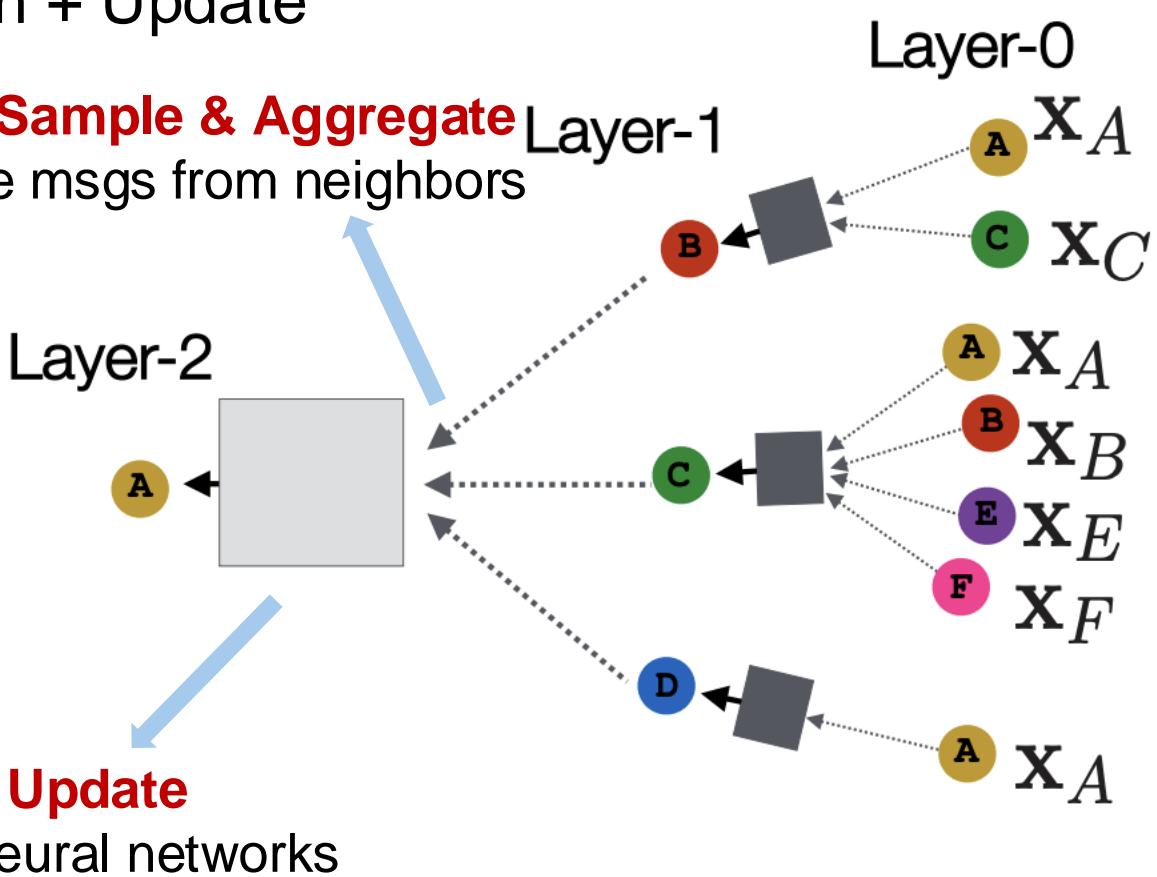
How to enable deep learning on temporal graphs?

Graph Neural Networks (GNNs)

- Nodes have representations at each layer, where layer-0 representations are input features X .
- Basic operations: Sample & Aggregation + Update

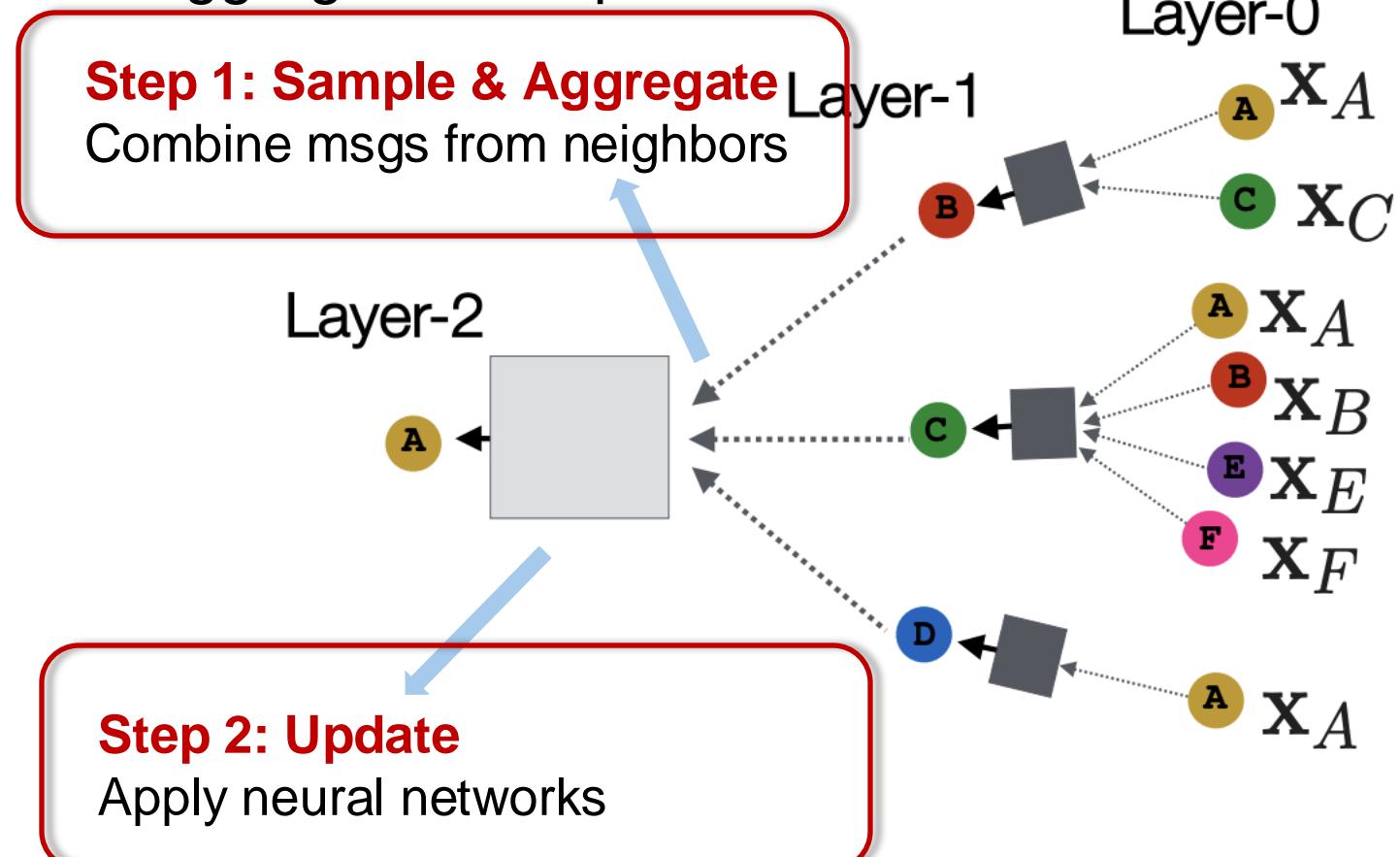
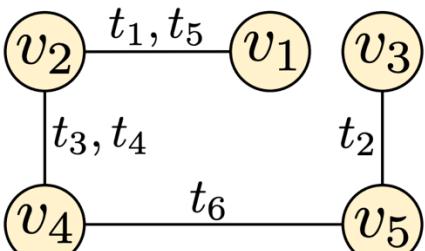
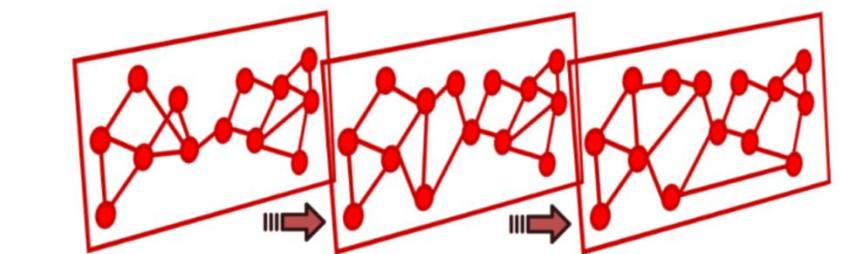


Step 1: Sample & Aggregate
Combine msgs from neighbors



Temporal Graph Neural Networks (TGNNs)

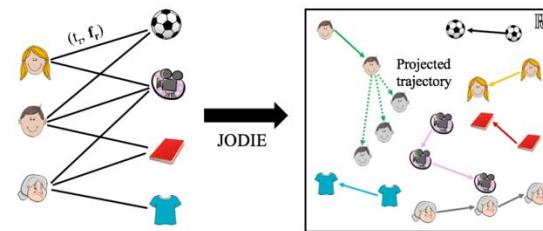
- Nodes have representations at each layer, where layer-0 representations are input features X .
- New operation designs: Sample & Aggregation + Update



Temporal Graph Neural Networks (TGNNs)

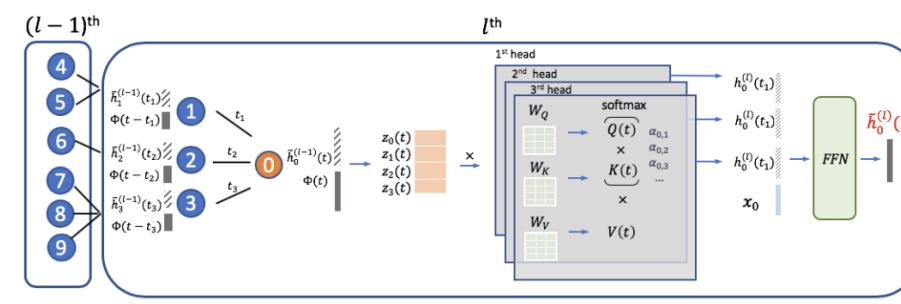
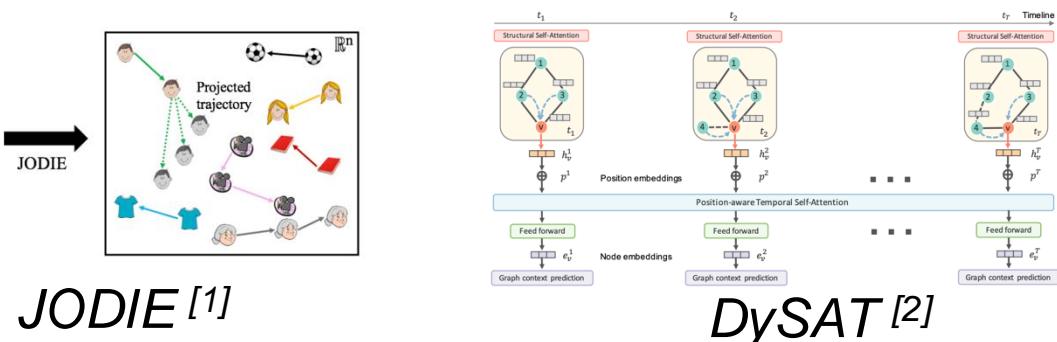
□ Categories of TGNNs

□ TGNN with RNN



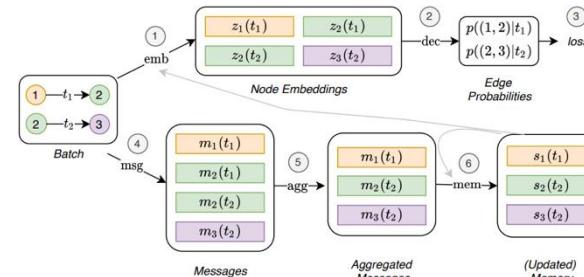
□ TGNN with self attention

□ TGNN with memory



□ TGNN with memory & self attention

□



[1] Kumar et al., JODIE: Predicting Dynamic Emb. Trajectory in Temporal Interaction Networks.

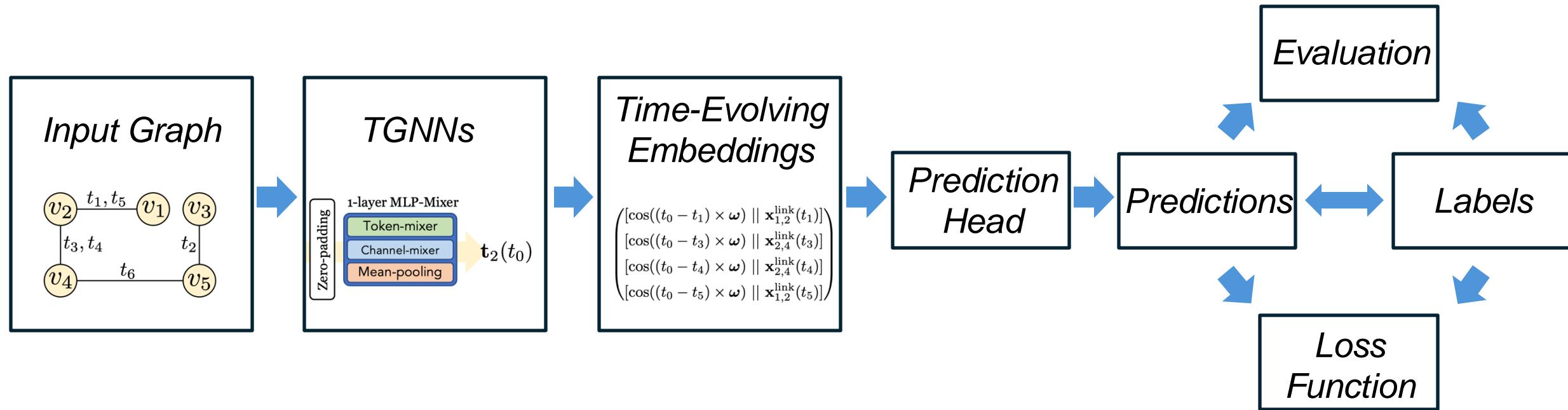
[2] Sankar et al., DySAT: Deep Neural Repr. Learning on Dynamic Graphs via Self-Attention Networks.

[3] Xu et al., Inductive Representation Learning on Temporal Graphs

[4] Rossi et al., Temporal Graph Networks for Deep Learning on Dynamic Graphs

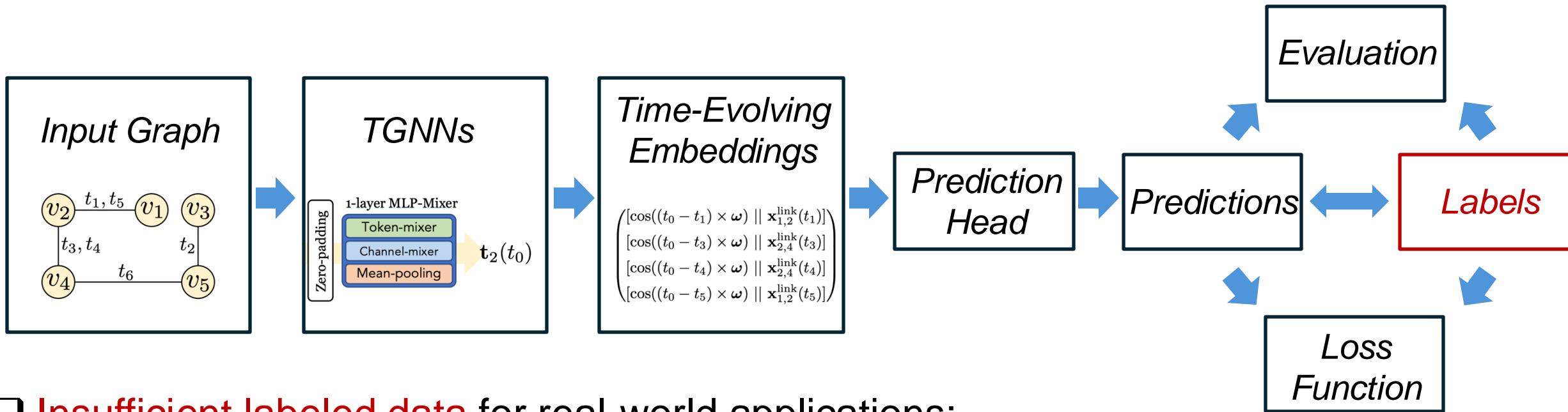
Training and inference pipeline of TGNNs

- Representation learning + task-related optimization.



Data-Efficiency Issue of TGNNs

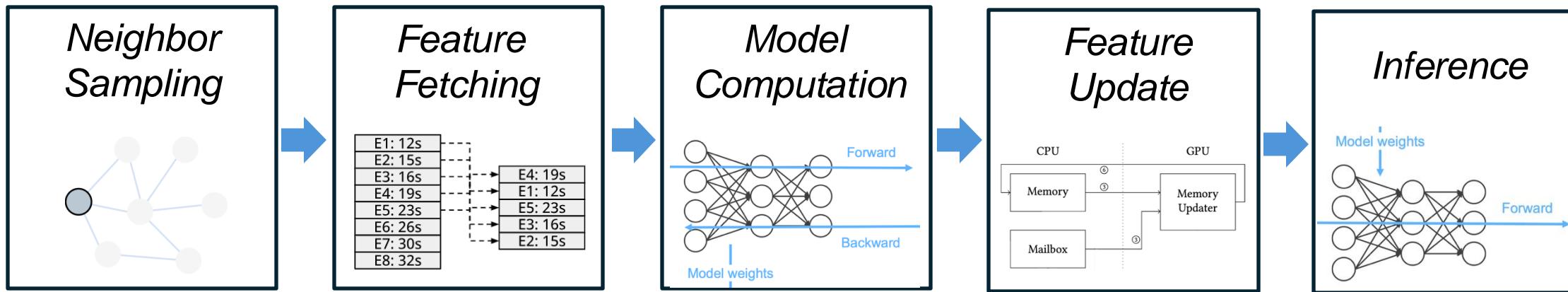
- Training TGNNs requires relatively abundant labeled data.



- Insufficient labeled data for real-world applications:
 - Indirect labels
 - Scarcity of task-specific labels
 - Limited labels for new tasks/distributions

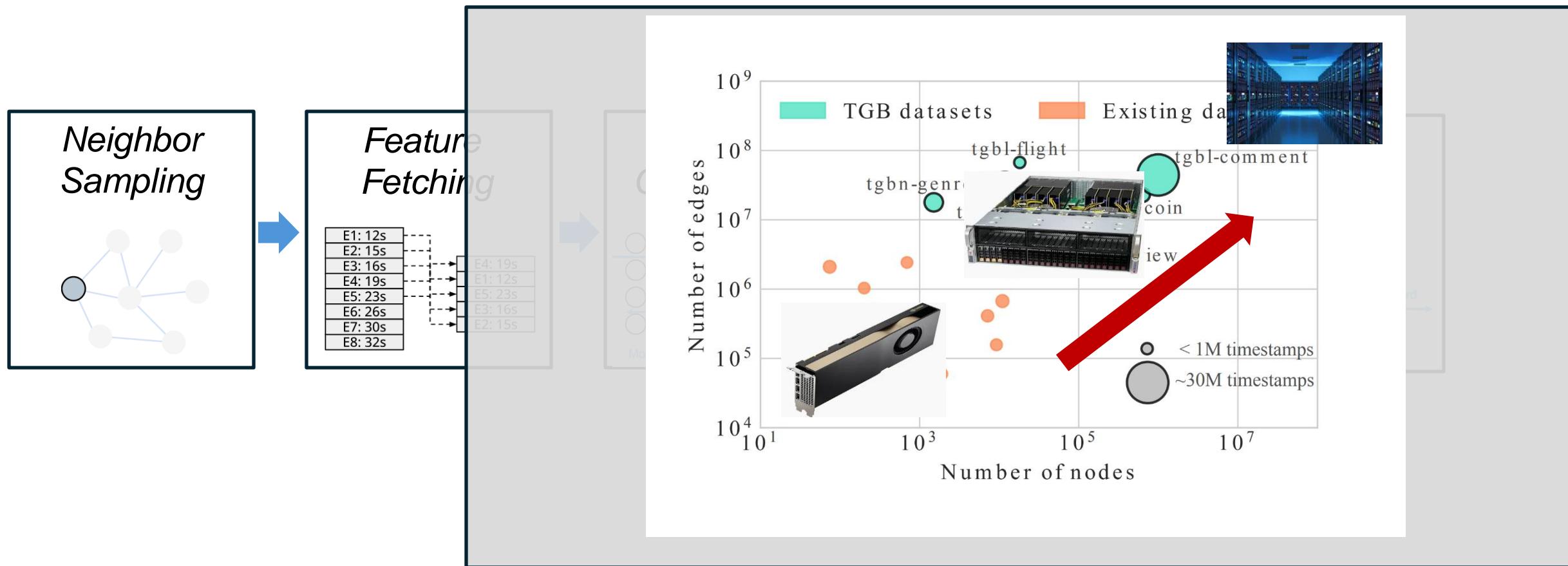
Computation pipeline of TGNNs

- ❑ Training computation
- ❑ Inference computation

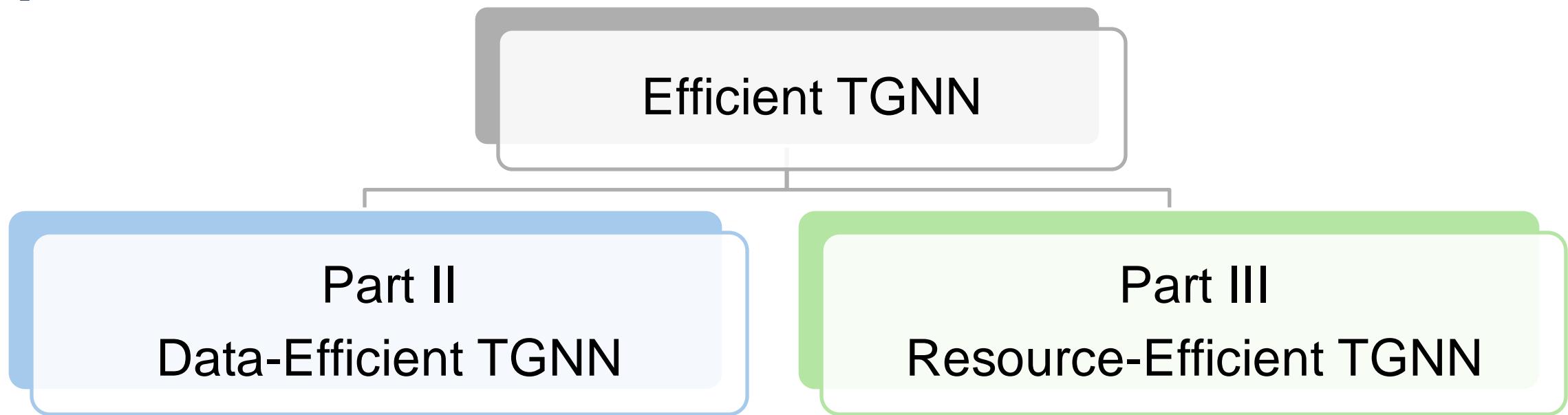


Resource-Efficiency Issue of TGNNs

- Fast growing of temporal graphs v.s. limited resources

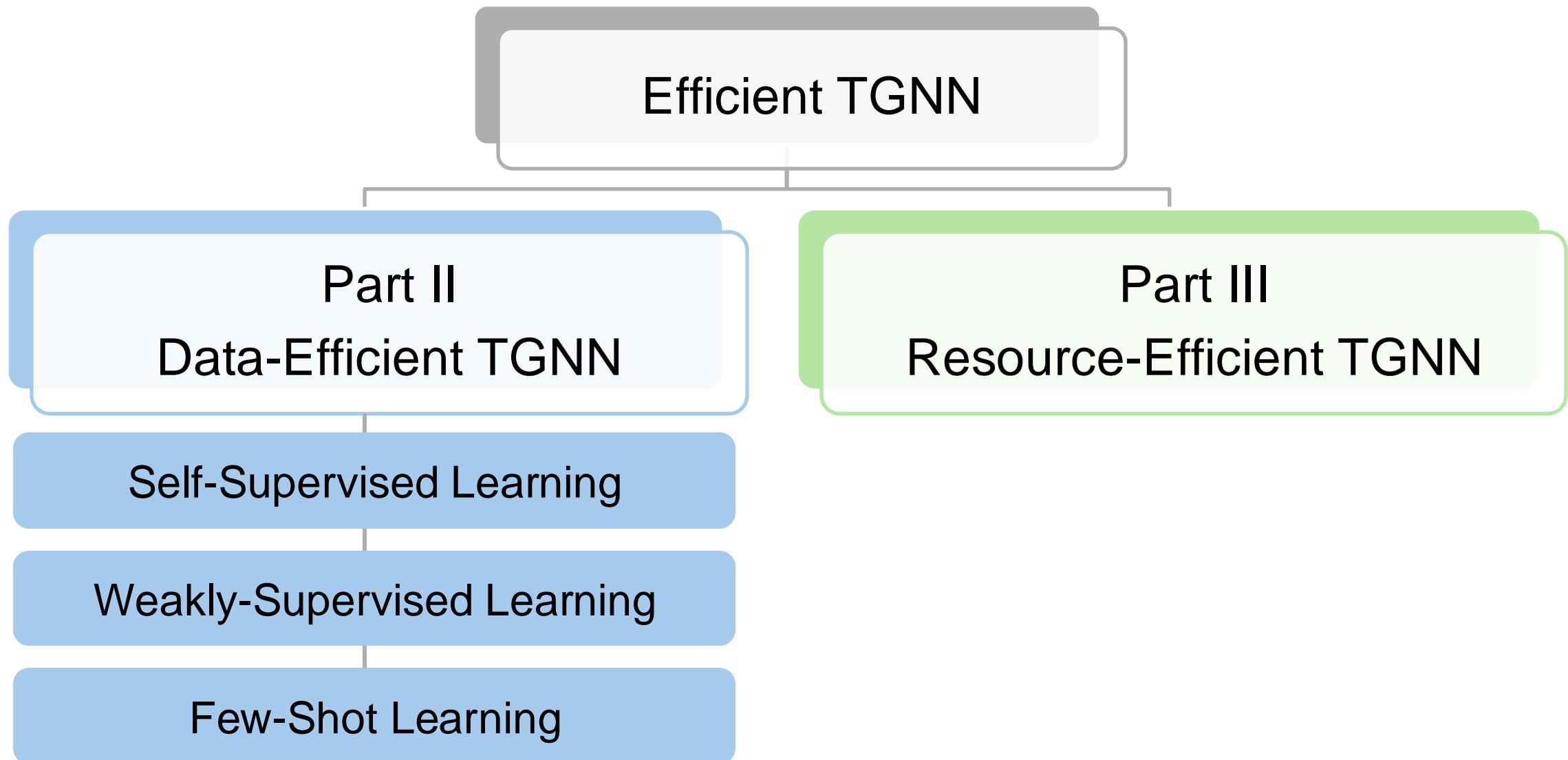


Scope of This Tutorial

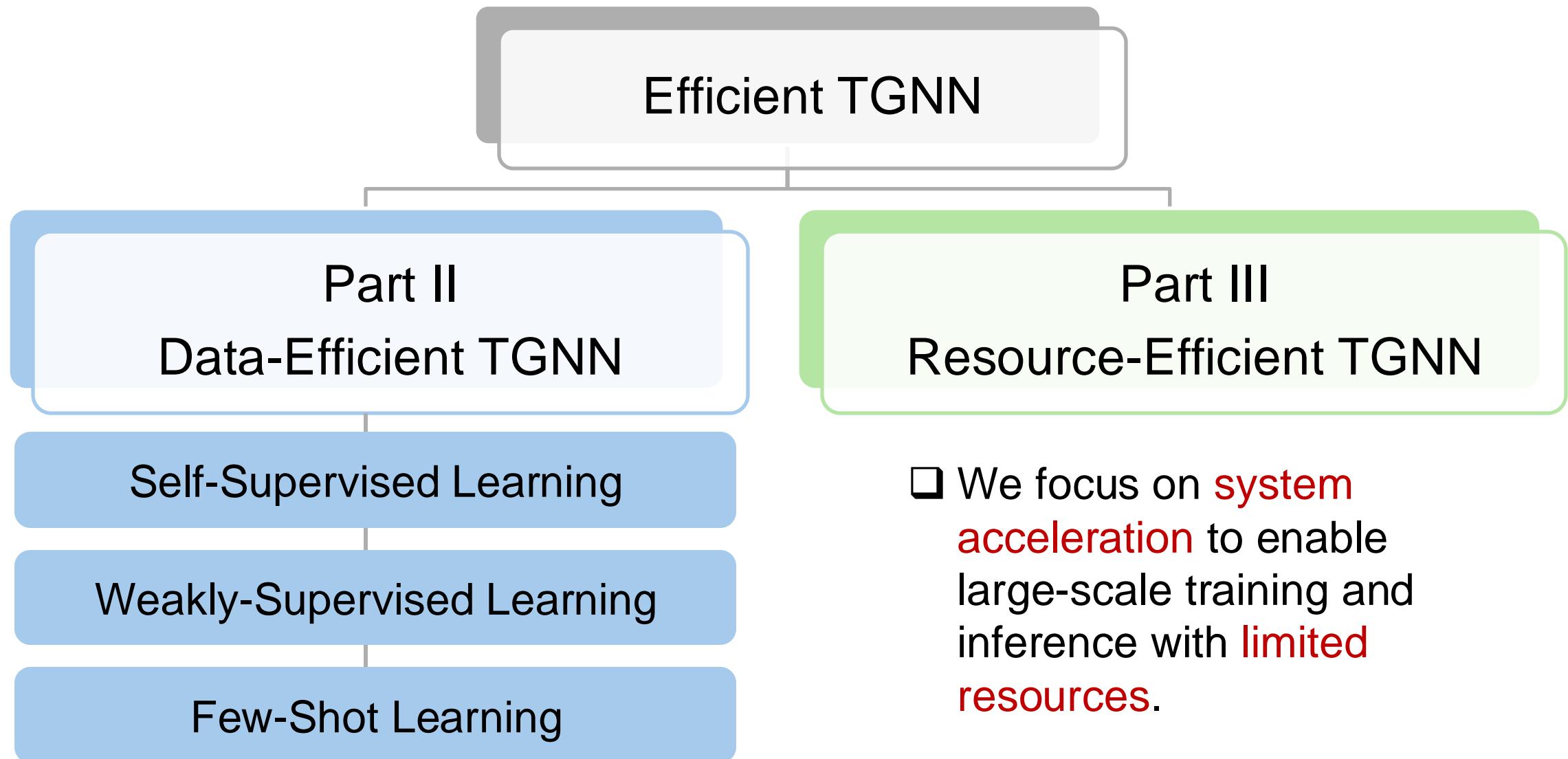


- We focus on **algorithm design** and **optimization techniques** to address the challenges posed by **insufficient labels**.

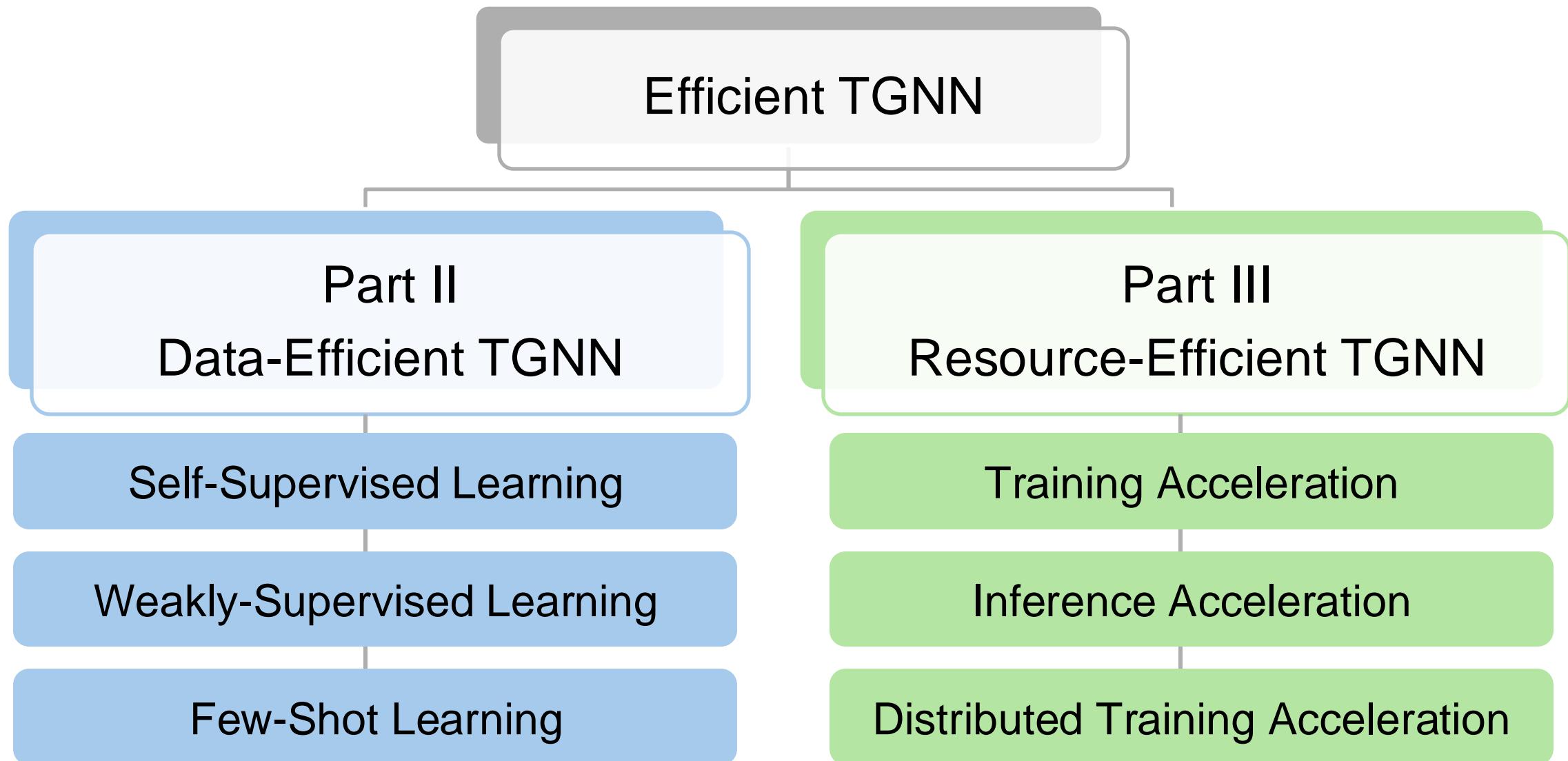
Scope of This Tutorial



Scope of This Tutorial



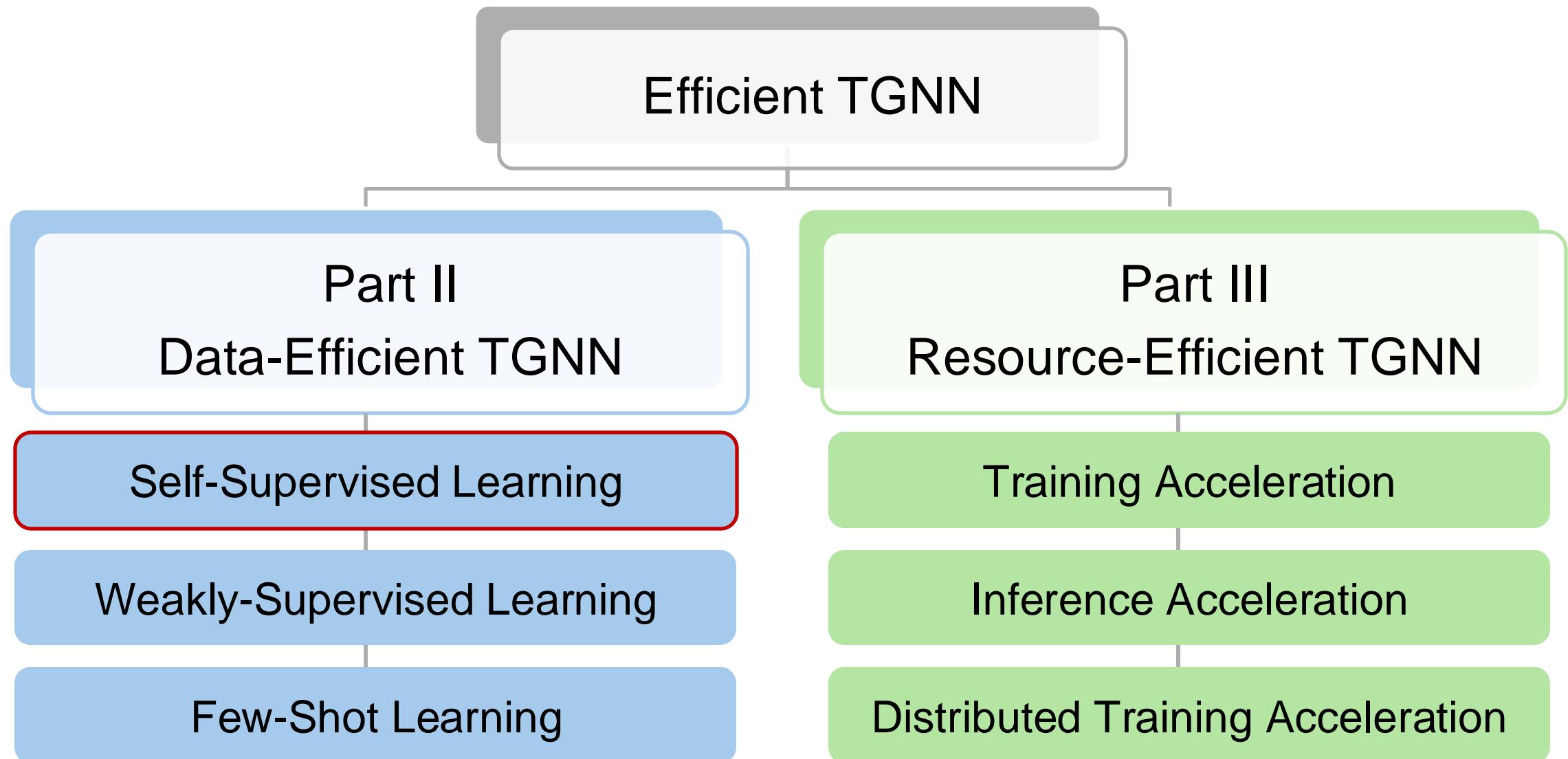
Scope of This Tutorial



Contents

- Part I – Introduction
- Part II – Data-Efficient Temporal Graph Neural Network
- 30-min Coffee Break (15:30 – 16:00)
- Part III – Resource-Efficient Temporal Graph Neural Network
- Part IV – Discussion and Future Directions

Scope of This Tutorial



Self-Supervised Learning on Temporal Graphs

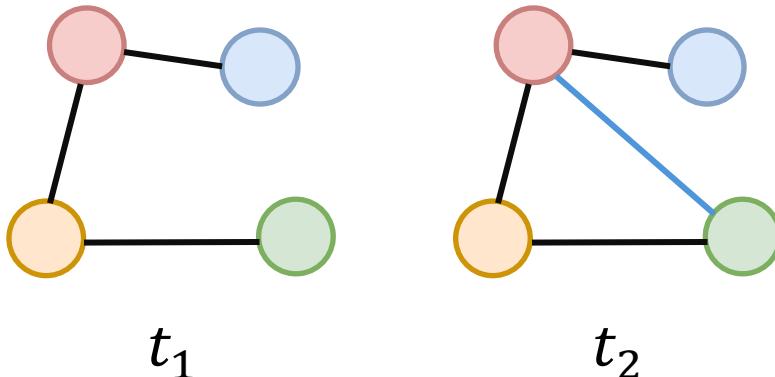
- **Introduction & Background**
- **Self-Supervision by Reconstruction**
- **Self-Supervision by Contrastive Approach**
- **Self-Supervision by Multiview Consistency**

Introduction – Self-Supervised Learning (SSL)

- Learning useful representations **without requiring labeled data**.
- Relies on the **inherent structure** and **temporal dynamics** of the graph itself.

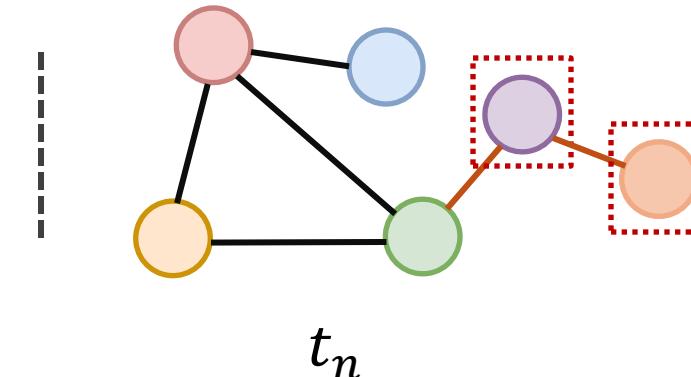
Transductive Task

Examine node embeddings that have been observed in training, via the future link prediction task and the node classification.



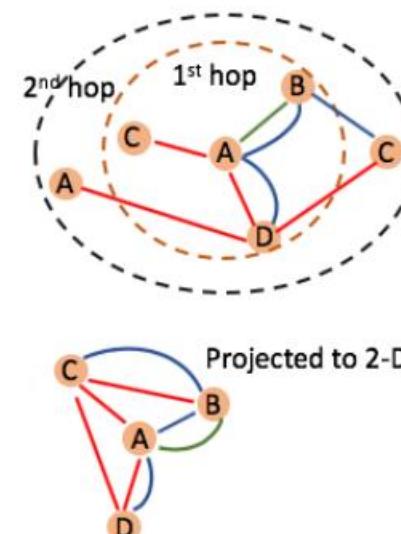
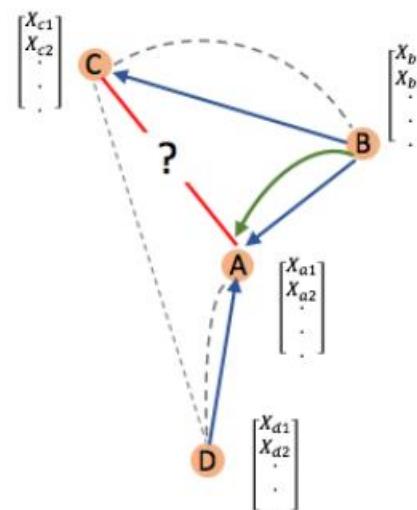
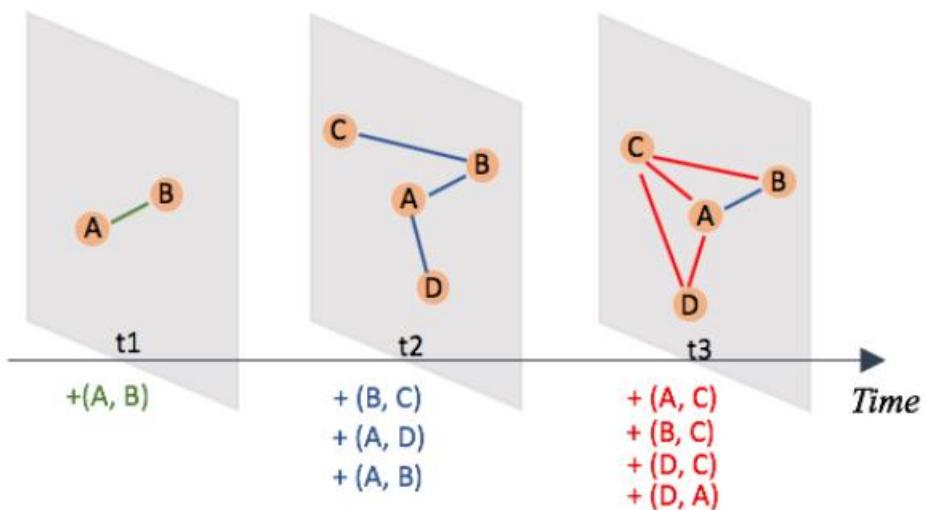
Inductive Task

Examine inferred representations of **unseen** node by predicting the future links between unseen nodes and classify them based on their inferred embedding dynamically



Challenges on Temporal Graphs

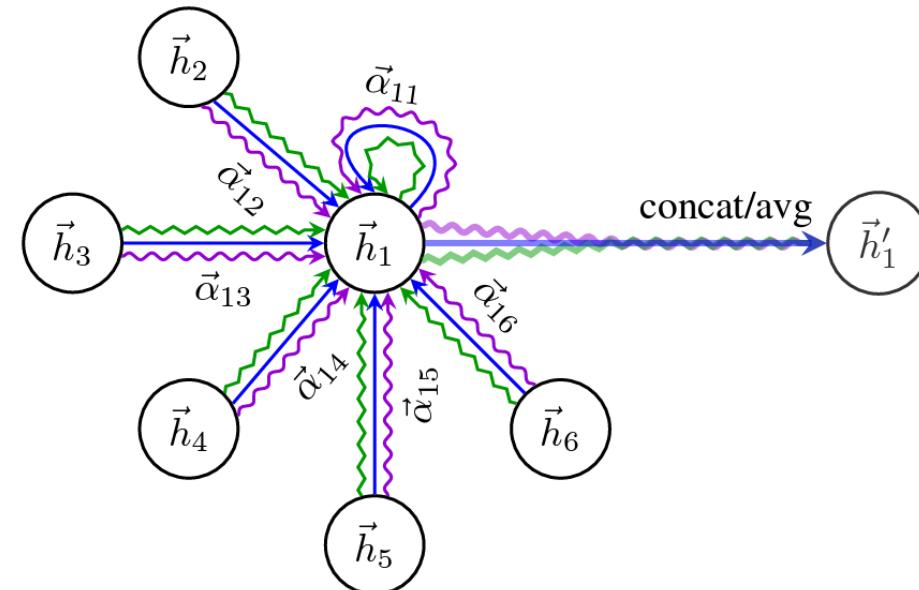
- **Challenge 1:** Node embeddings should also function of time.
- **Challenge 2:** Temporal constraints on neighborhood aggregation methods.
- **Challenge 3:** Possibly multiple node interactions.



Background – Attention Mechanism on Graphs

- Query neighbors by keys derived from their representations, aggregating their value by the attention weight.

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

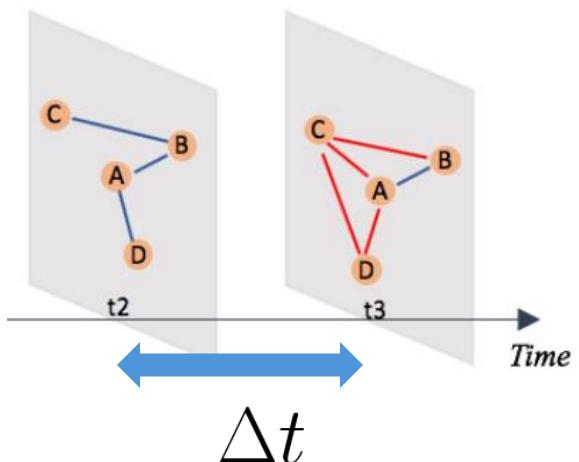


- Question:** How to involve temporal information?

Background – Time Encoding

- A form of “**positional encoding**” concatenated to the node representation.
- Generate a vector encoding given a real number.
- Encoding represents time span rather than absolute value of time
(Translation-invariance).

$$\mathcal{K}(t_1, t_2) := \langle \Phi(t_1), \Phi(t_2) \rangle \quad \mathcal{K}(t_1 + c, t_2 + c) = \mathcal{K}(t_1, t_2)$$



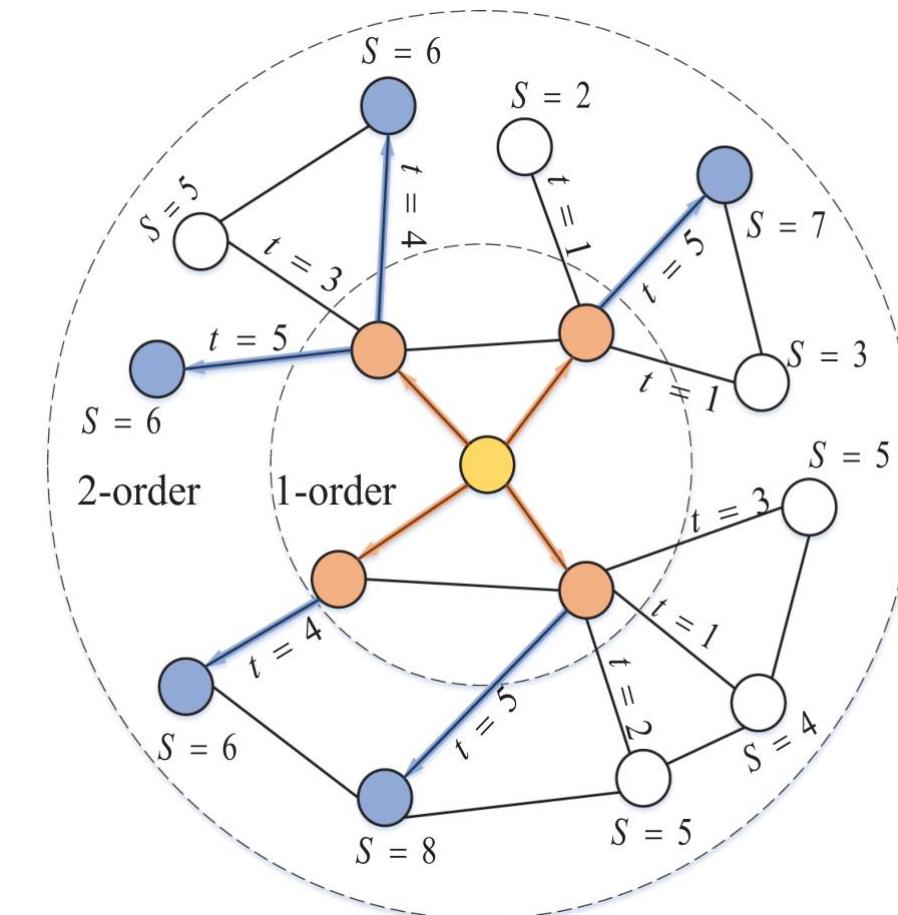
Using Bochner's Theorem and Monte Carlo approximation:

$$\mathcal{K}(t_1, t_2) \approx \frac{1}{d} \sum_{i=1}^d \cos(\omega_i t_1) \cos(\omega_i t_2) + \sin(\omega_i t_1) \sin(\omega_i t_2)$$

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

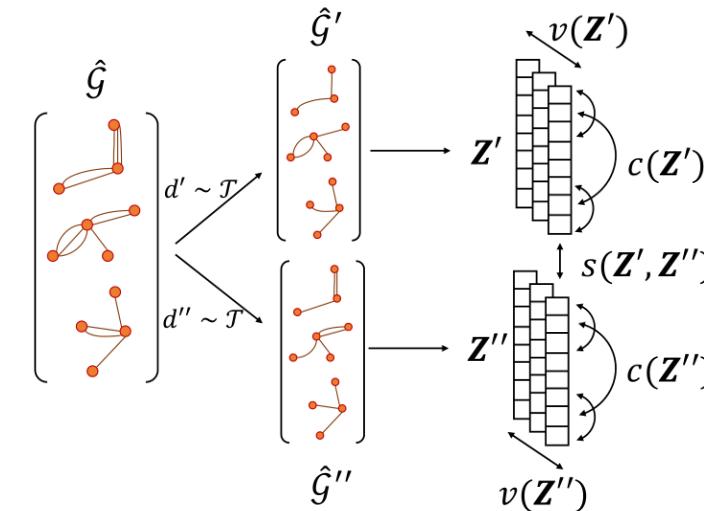
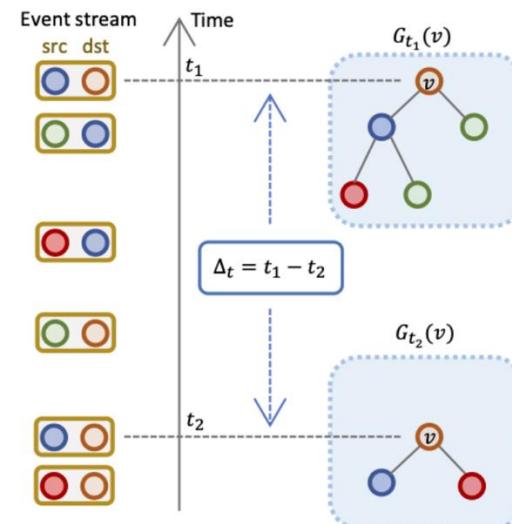
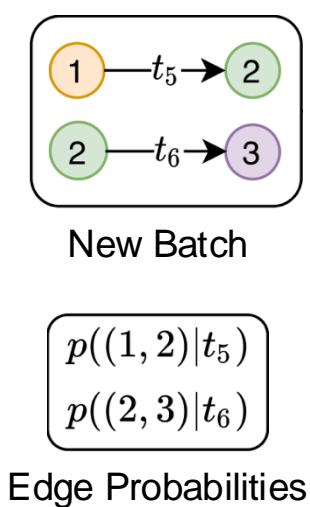
Background – Temporal Subgraph Sampling

- Temporal subgraph sampling is key to batch-wise training and contrastive pair construction.
- Message passing directions must align with the observed chronological orders.
- Given a target number of nodes for subgraph, candidates can be further weighted by structural or temporal importance.
 - Degree, centrality, or PageRank
 - Time elapsed.



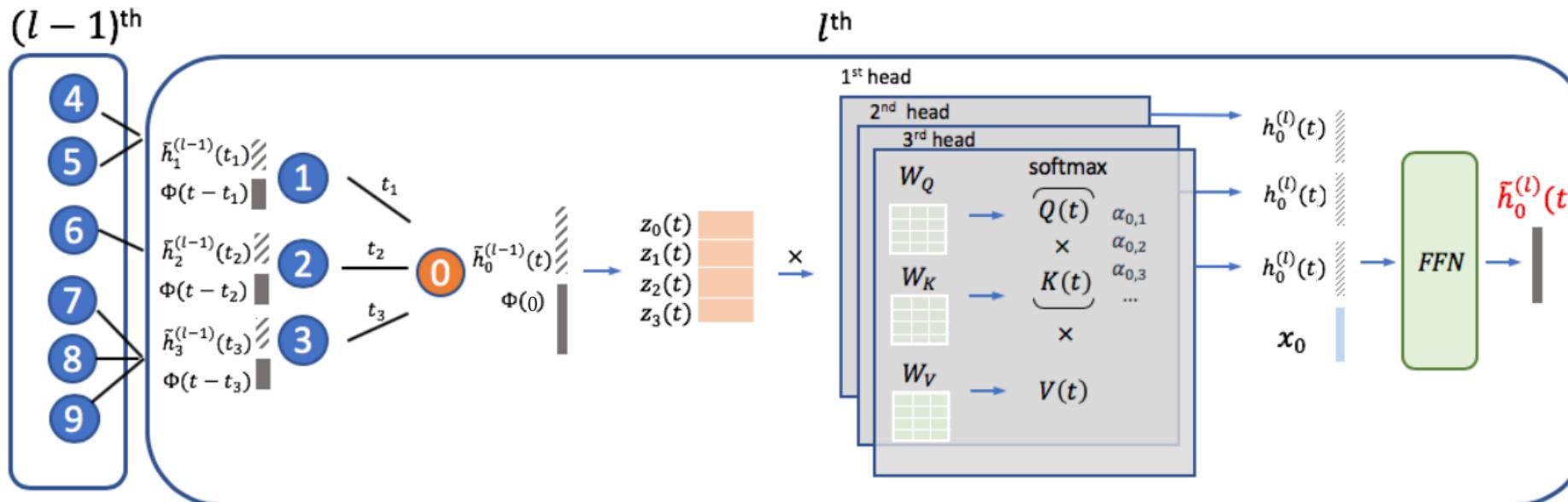
Introduction – Self-Supervised Learning (SSL)

- SSL paradigms typically generate supervision signals through designed tasks:
 - Transductive future link reconstruction. *Loss is based on cross entropy.*
 - Contrastive learning: learning from positive and negative pair of examples. *Loss is based on similarity measure.*
 - Multiview consistency: representations should be robust under perturbations and agree with each other. *Loss is based on regularizations.*



SSL by Reconstruction: TGAT

- **Objective:** Produce time-aware representation for a target node at time point t .
- **Motivation:** Analogous to GraphSAGE or GAT, takes temporal neighborhood with hidden representations and timestamps, and aggregate.
- **Method:** A local aggregation operator, using attention mechanism.



- **Link prediction loss:** $\ell = \sum_{(v_i, v_j, t_{ij}) \in \mathcal{E}} -\log \left(\sigma(-\tilde{\mathbf{h}}_i^l(t_{ij})^\top \tilde{\mathbf{h}}_j^l(t_{ij})) \right) - Q \cdot \mathbb{E}_{v_q \sim P_n(v)} \log \left(\sigma(\tilde{\mathbf{h}}_i^l(t_{ij})^\top \tilde{\mathbf{h}}_q^l(t_{ij})) \right)$

SSL by Reconstruction: TGAT

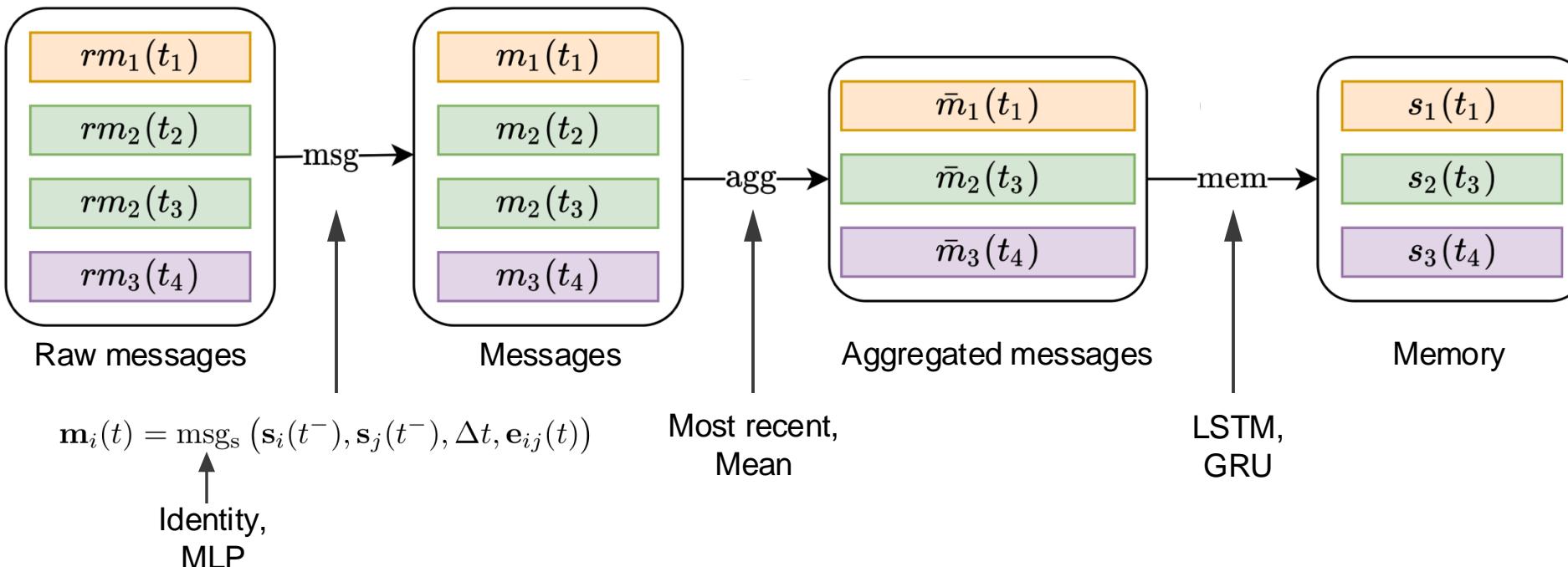
☐ Experiments: Transductive & inductive learning task for future link prediction.

Dataset Metric	Reddit		Wikipedia		Industrial	
	Accuracy	AP	Accuracy	AP	Accuracy	AP
GAE	74.31 (0.5)	93.23 (0.3)	72.85 (0.7)	91.44 (0.1)	68.92 (0.3)	81.15 (0.2)
VAGE	74.19 (0.4)	92.92 (0.2)	78.01 (0.3)	91.34 (0.3)	67.81 (0.4)	80.87 (0.3)
DeepWalk	71.43 (0.6)	83.10 (0.5)	76.67 (0.5)	90.71 (0.6)	65.87 (0.3)	80.93 (0.2)
Node2vec	72.53 (0.4)	84.58 (0.5)	78.09 (0.4)	91.48 (0.3)	66.64 (0.3)	81.39 (0.3)
CTDNE	73.76 (0.5)	91.41 (0.3)	79.42 (0.4)	92.17 (0.5)	67.81 (0.3)	80.95 (0.5)
GAT	92.14 (0.2)	97.33 (0.2)	87.34 (0.3)	94.73 (0.2)	69.58 (0.4)	81.51 (0.2)
GAT+T	92.47 (0.2)	97.62 (0.2)	87.57 (0.2)	95.14 (0.4)	70.15 (0.3)	82.66 (0.4)
GraphSAGE	92.31(0.2)	97.65 (0.2)	85.93 (0.3)	93.56 (0.3)	70.19 (0.2)	83.27 (0.3)
GraphSAGE+T	92.58 (0.2)	97.89 (0.3)	86.31 (0.3)	93.72 (0.3)	71.84 (0.3)	84.95 (0.)
Const-TGAT	91.39 (0.2)	97.86 (0.2)	86.03 (0.4)	93.50 (0.3)	68.52 (0.2)	81.91 (0.3)
TGAT	92.92 (0.3)	98.12 (0.2)	88.14 (0.2)	95.34 (0.1)	73.28 (0.2)	86.32 (0.1)

Dataset Metric	Reddit		Wikipedia		Industrial	
	Accuracy	AP	Accuracy	AP	Accuracy	AP
GAT	89.86 (0.2)	95.37 (0.3)	82.36 (0.3)	91.27 (0.4)	68.28 (0.2)	79.93 (0.3)
GAT+T	90.44 (0.3)	96.31 (0.3)	84.82 (0.3)	93.57 (0.3)	69.51 (0.3)	81.68 (0.3)
GraphSAGE	89.43 (0.1)	96.27 (0.2)	82.43 (0.3)	91.09 (0.3)	67.49 (0.2)	80.54 (0.3)
GraphSAGE+T	90.07 (0.2)	95.83 (0.2)	84.03 (0.4)	92.37 (0.5)	69.66 (0.3)	82.74 (0.3)
Const-TGAT	88.28 (0.3)	94.12 (0.2)	83.60 (0.4)	91.93 (0.3)	65.87 (0.3)	77.03 (0.4)
TGAT	90.73 (0.2)	96.62 (0.3)	85.35 (0.2)	93.99 (0.3)	72.08 (0.3)	84.99 (0.2)

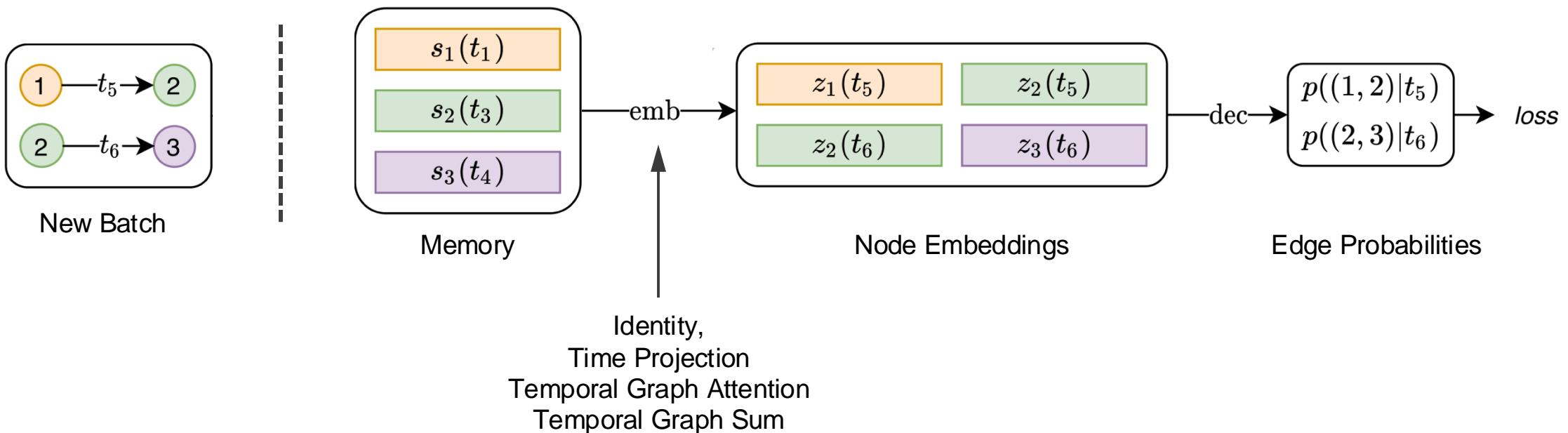
SSL by Reconstruction: Temporal Graph Networks (TGN)

- **Motivation:** Viewing dynamic graphs as sequences of timed events.
- **Method:** Five modules that process dynamic graphs as a series of node-wise event, interaction event, or deletion event, and save the node states to memory.



SSL by Reconstruction: Temporal Graph Networks (TGN)

- **Motivation:** Viewing dynamic graphs as sequences of timed events.
- **Method:** Five modules that process dynamic graphs as a series of node-wise event, interaction event, or deletion event, and save the node states to memory.



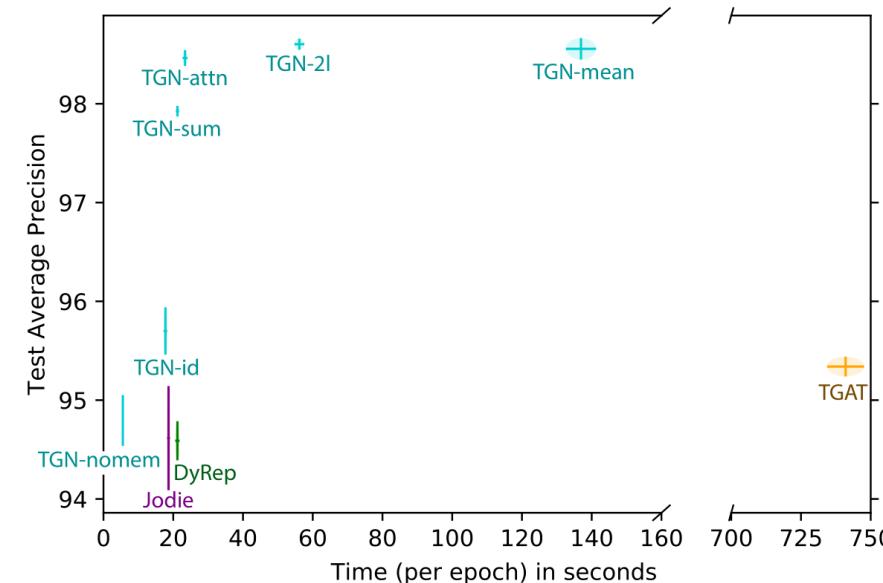
SSL by Reconstruction: Temporal Graph Networks (TGN)

☐ Experiments: Transductive & inductive learning task for future link prediction.

Table 2: Average Precision (%) for future edge prediction task in transductive and inductive settings.

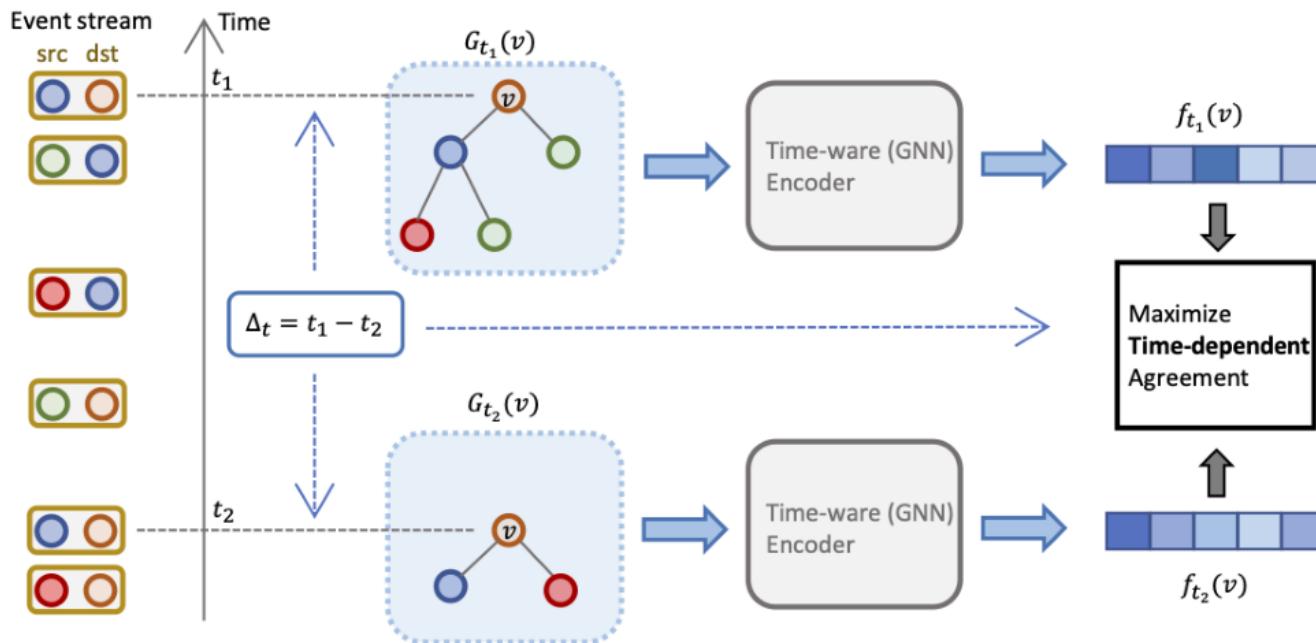
First, **Second**, **Third** best performing method. *Static graph method. †Does not support inductive.

	Wikipedia		Reddit		Twitter	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
GAE*	91.44 ± 0.1	†	93.23 ± 0.3	†	—	†
VAGE*	91.34 ± 0.3	†	92.92 ± 0.2	†	—	†
DeepWalk*	90.71 ± 0.6	†	83.10 ± 0.5	†	—	†
Node2Vec*	91.48 ± 0.3	†	84.58 ± 0.5	†	—	†
GAT*	94.73 ± 0.2	91.27 ± 0.4	97.33 ± 0.2	95.37 ± 0.3	67.57 ± 0.4	62.32 ± 0.5
GraphSAGE*	93.56 ± 0.3	91.09 ± 0.3	97.65 ± 0.2	96.27 ± 0.2	65.79 ± 0.6	60.13 ± 0.6
CTDNE	92.17 ± 0.5	†	91.41 ± 0.3	†	—	†
Jodie	94.62 ± 0.5	93.11 ± 0.4	97.11 ± 0.3	94.36 ± 1.1	85.20 ± 2.4	79.83 ± 2.5
TGAT	95.34 ± 0.1	93.99 ± 0.3	98.12 ± 0.2	96.62 ± 0.3	70.02 ± 0.6	66.35 ± 0.8
DyRep	94.59 ± 0.2	92.05 ± 0.3	97.98 ± 0.1	95.68 ± 0.2	83.52 ± 3.0	78.38 ± 4.0
TGN-attn	98.46 ± 0.1	97.81 ± 0.1	98.70 ± 0.1	97.55 ± 0.1	94.52 ± 0.5	91.37 ± 1.1



SSL by Contrastive Learning: TGAT-CL

- **Motivation:** Node representation process is in general “smooth”.
- **Method:** Contrast the same node representation over time.



$$\text{sim}(x, y) = x^T \Omega y \quad \Omega(t_x, t_y) = \Omega(|t_x - t_y|)$$

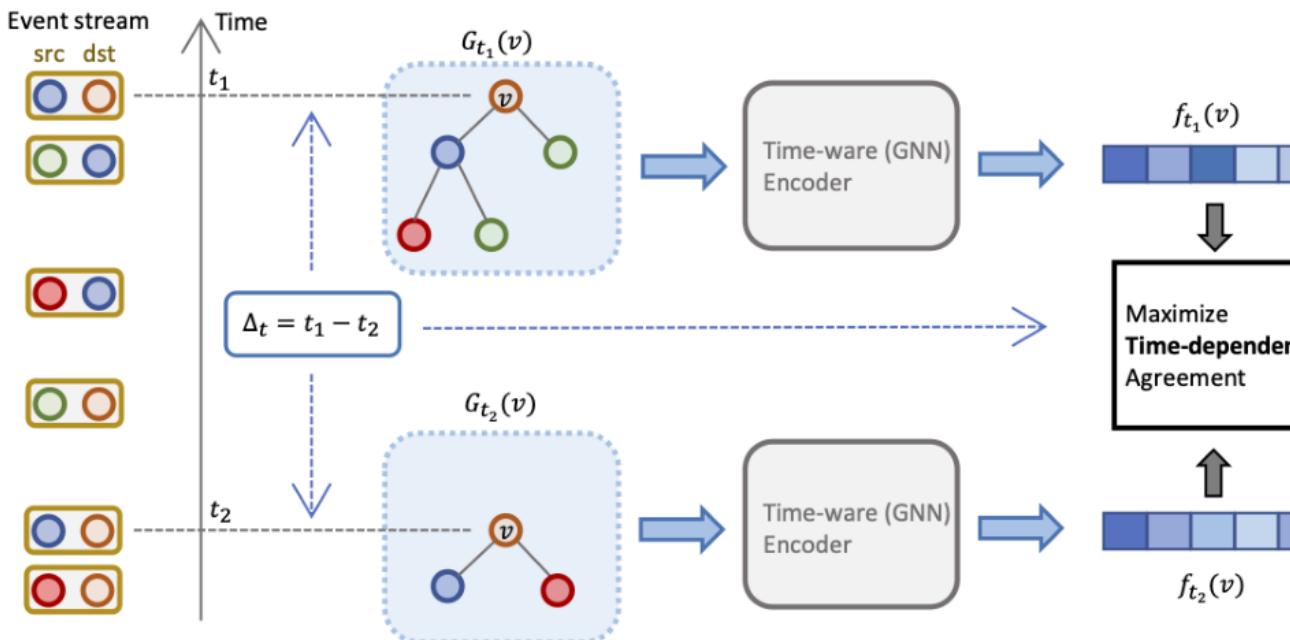
Algorithm 1 Temporal view construction with a given anchor

Require: Anchor $x = (v, t, G_t^k(v))$, desired number of hops k , sampling duration proportion γ , event stream \mathcal{E} .

- 1: **if** No events in \mathcal{E} with target v happens during $((1 - \gamma)t, t)$ **then return**
 - 2: Sample $t^+ \sim \text{Unif}((1 - \gamma)t, t)$
 - 3: Construct k -hop temporal graph $G_{t^+}^k(v)$ of v at t^+ .
 - 4: **return** $x^+ = (v, t^+, G_{t^+}^k(v))$
-

SSL by Contrastive Learning: TGAT-CL

- **Motivation:** Node representation process is in general “smooth”.
- **Method:** Contrast the same node representation over time.
- **Challenge:** Bias in negative example sampling.



$$\text{sim}(x, y) = x^T \Omega y$$

$$\Omega(t_x, t_y) = \Omega(|t_x - t_y|)$$

- **Contrastive Loss:** $-\mathbb{E}_x \left[\mathbb{E}_{x^+ \sim p^+} \log \frac{1}{1 + e^{-\text{sim}(f(x^+), f(x))}} + \mathbb{E}_{x^- \sim p^-} \log \frac{1}{1 + e^{\text{sim}(f(x^-), f(x))}} \right]$

SSL by Contrastive Learning: TGAT-CL

- **Motivation:** Node representation process is in general “smooth”.
- **Method:** Contrast the same node representation over time.
- **Challenge:** Bias in negative example sampling.
- **Debiased Contrastive Loss:**

$$\tau^+ = p(c(x') = c(x))$$

$$\mathcal{L}_{\text{DDGCL}} =$$

$$\begin{aligned} & - \frac{1}{N} \sum_{x \in X} \sum_{l=1}^{N_{\text{pos}}} \frac{1}{N_{\text{pos}}} \log \frac{1}{1 + e^{-\text{sim}(f(x_l^+), f(x))}} \\ & - \frac{1}{N(1 - \tau^+)} \sum_{x \in X} \sum_{i=1}^{N_{\text{neg}}} \left(\frac{e^{\beta \text{sim}(f(x_i^-), f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^-), f(x))}} \right) \log \frac{1}{1 + e^{\text{sim}(f(x_i^-), f(x))}} \\ & + \frac{\tau^+}{N(1 - \tau^+)} \sum_{x \in X} \sum_{i=1}^{N_{\text{pos}}} \left(\frac{e^{\beta \text{sim}(f(x_i^+), f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^+), f(x))}} \right) \log \frac{1}{1 + e^{\text{sim}(f(x_i^+), f(x))}} \end{aligned}$$

SSL by Contrastive Learning: TGAT-CL

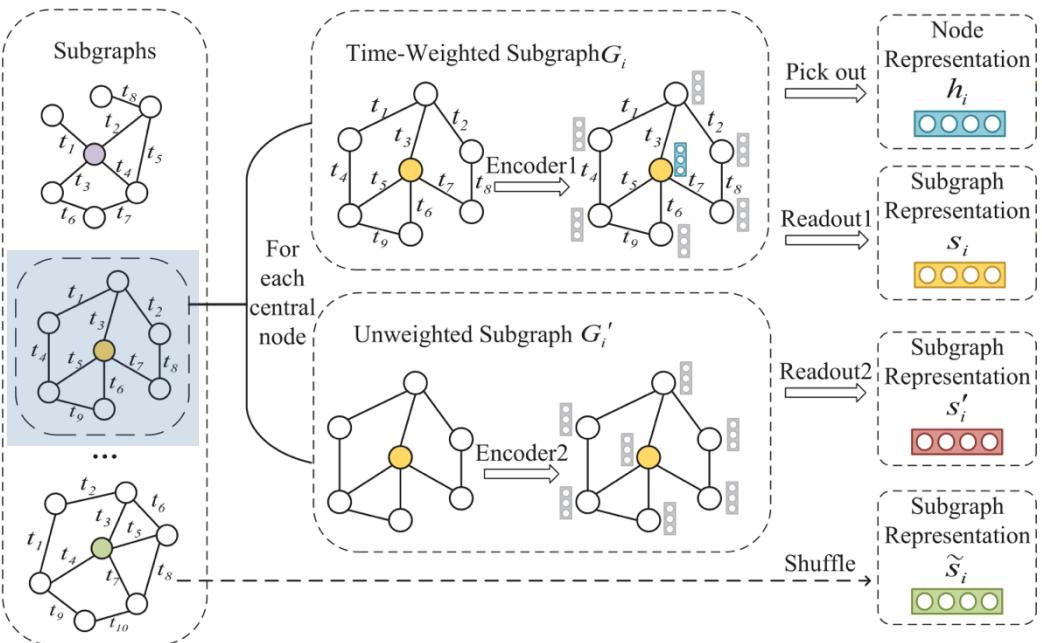
- **Motivation:** Dynamic node classification performance in average AUC and dynamic link prediction performance in average precision.

	Wikipedia	Reddit	MOOC
GAT [41]	82.34 ± 0.8	64.52 ± 0.5	66.21 ± 0.4
GraphSAGE [15]	82.42 ± 0.7	61.24 ± 0.6	65.17 ± 0.5
JODIE [23]	84.84 ± 1.2	61.83 ± 2.7	66.69 ± 0.9
CAW-N [43]	86.77 ± 0.3	67.46 ± 0.8	68.77 ± 0.4
TGN (S) [35]	86.80 ± 1.1	64.03 ± 1.0	69.89 ± 0.6
TGN [35]	88.56 ± 0.3	68.63 ± 0.7	71.64 ± 0.3
TGAT(S) [46]	81.28 ± 0.7	64.80 ± 0.8	68.28 ± 0.4
TGAT [46]	83.69 ± 0.7	65.56 ± 0.7	69.46 ± 0.4
TGAT-CL(PF)	87.48 ± 0.5	68.71 ± 0.7	73.41 ± 0.3
TGAT-CL(MTL)	89.32 ± 0.5	71.13 ± 0.8	74.54 ± 0.2

	Wikipeida		Reddit	
	Transductive	Inductive	Transductive	Inductive
GAT [41]	94.73 ± 0.2	91.27 ± 0.4	97.33 ± 0.2	95.37 ± 0.3
GraphSAGE [15]	93.56 ± 0.3	91.09 ± 0.3	97.65 ± 0.2	96.27 ± 0.2
JODIE [23]	94.62 ± 0.4	93.11 ± 0.4	97.11 ± 0.3	94.36 ± 1.1
CAW-N [43]	99.51 ± 0.1	99.74 ± 0.1	99.72 ± 0.1	99.37 ± 0.1
TGN [35]	98.46 ± 0.3	97.81 ± 0.1	98.70 ± 0.1	97.55 ± 0.1
TGAT [46]	95.34 ± 0.1	93.99 ± 0.3	98.12 ± 0.2	96.62 ± 0.3
TGAT-CL(PF)	98.54 ± 0.1	98.22 ± 0.1	98.63 ± 0.1	98.19 ± 0.1
TGAT-CL(MTL)	98.79 ± 0.1	99.06 ± 0.1	99.13 ± 0.1	98.45 ± 0.1

SSL by Contrastive Learning: DySubC

- **Motivation:** Node vs. subgraph representations; temporal vs. static representations.
- **Method:** Contrast between a positive subgraph sample, a temporal negative sample, and a structural negative sample.



Readout:

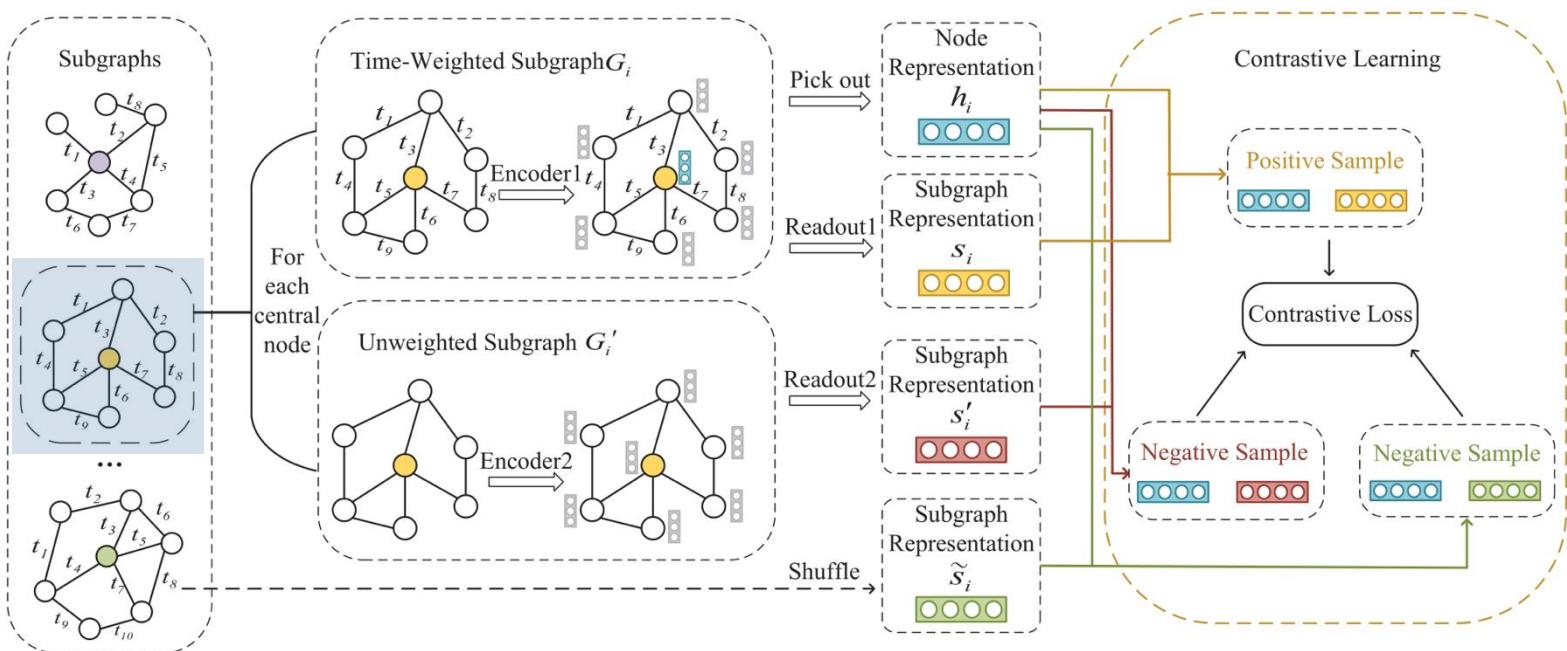
$$Inf_j = \tau(i, j) + \beta \frac{1}{Dist(i, j)}$$

$$s_i = \frac{1}{\sum_{j=1}^k Inf_j} \sum_{j=1}^k Inf_j h_j$$

$$s'_i = \frac{1}{k} \sum_{j=1}^k h'_j$$

SSL by Contrastive Learning: DySubC

- **Motivation:** Node vs. subgraph representations; temporal vs. static representations.
- **Method:** Contrast between a positive subgraph sample, a temporal negative sample, and a structural negative sample.



Training Loss:

$$\mathcal{L}_1 = \frac{1}{|V|} \sum_{i=1}^{|V|} \mathbb{E}_{(X,A)}(-\max(\sigma(h_i s_i) - \sigma(h_i \tilde{s}_i) + \phi, 0))$$

$$\mathcal{L}_2 = \frac{1}{|V|} \sum_{i=1}^{|V|} \mathbb{E}_{(X,A)}(-\max(\sigma(h_i s_i) - \sigma(h_i s'_i) + \varphi, 0))$$

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2$$

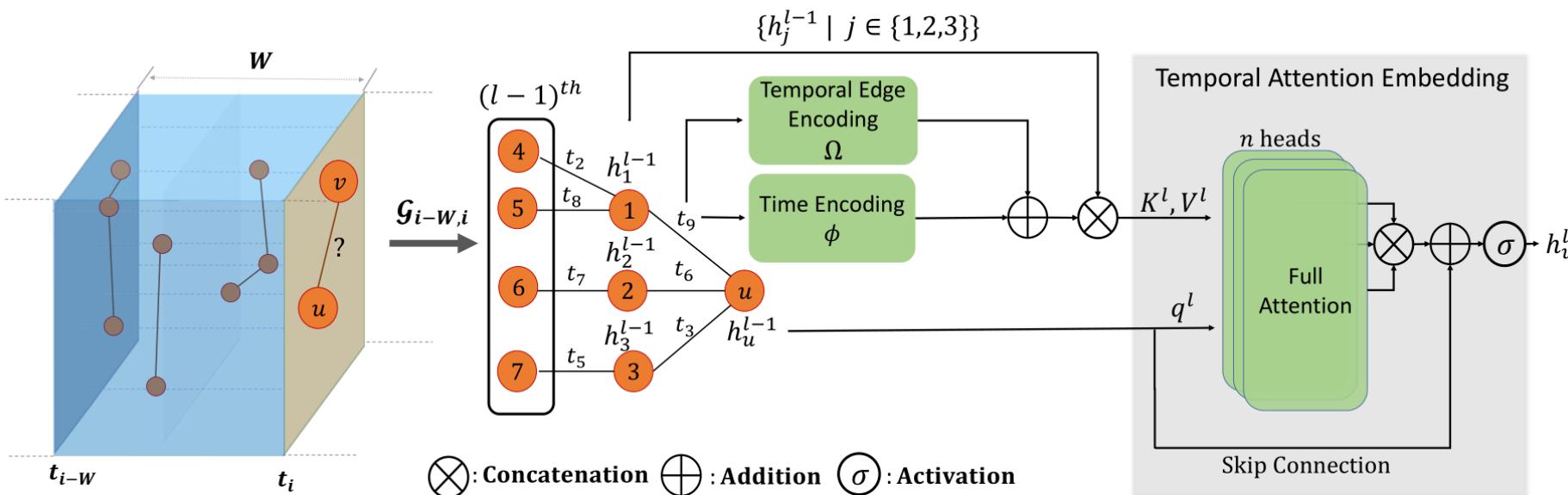
SSL by Contrastive Learning: DySubC

☐ Experiments: Link prediction in terms of average AUC score and Accuracy.

	Forum		Bitcoin Alpha		Wikipedia		Movielens		Math Overflow	
	AUC	Accuracy								
node2vec	73.51 ± 0.5	67.03 ± 0.6	70.01 ± 0.4	65.14 ± 0.6	68.77 ± 0.5	64.39 ± 0.2	71.18 ± 0.3	64.77 ± 0.2	75.21 ± 0.5	69.32 ± 0.8
graphSAGE	74.65 ± 0.3	68.15 ± 0.4	73.89 ± 0.4	68.87 ± 0.3	74.54 ± 0.5	70.14 ± 0.2	76.48 ± 0.5	69.94 ± 0.7	78.63 ± 0.4	72.58 ± 0.6
SEAL	74.81 ± 0.3	68.27 ± 0.3	74.22 ± 0.4	69.37 ± 0.3	72.68 ± 0.4	69.04 ± 0.5	78.60 ± 0.3	72.42 ± 0.5	80.91 ± 0.5	74.88 ± 0.6
DGI	81.18 ± 0.3	75.89 ± 0.1	79.48 ± 0.9	74.28 ± 1.1	88.28 ± 0.5	81.27 ± 0.4	89.75 ± 0.3	84.84 ± 0.5	83.92 ± 0.2	78.03 ± 0.5
Sub-Con	85.06 ± 0.6	78.02 ± 0.4	87.35 ± 0.4	80.92 ± 0.2	87.84 ± 0.1	80.84 ± 0.3	80.52 ± 0.6	73.21 ± 0.4	82.81 ± 0.7	77.25 ± 0.5
GCA	86.74 ± 0.3	77.97 ± 0.6	89.11 ± 0.1	81.23 ± 0.3	89.92 ± 0.2	81.92 ± 0.6	91.97 ± 0.7	85.37 ± 1.1	89.82 ± 0.5	84.21 ± 0.6
CTDNE	75.22 ± 0.5	68.97 ± 0.2	73.34 ± 0.3	68.91 ± 0.6	72.25 ± 0.3	68.54 ± 0.9	78.55 ± 1.4	72.34 ± 1.3	85.07 ± 1.2	79.87 ± 0.8
HTNE	77.56 ± 0.2	70.23 ± 0.3	73.65 ± 0.7	68.72 ± 0.3	82.53 ± 0.5	76.73 ± 0.8	82.71 ± 0.4	75.62 ± 0.3	84.29 ± 0.7	79.04 ± 0.3
DyRep	75.92 ± 0.5	68.14 ± 0.3	82.74 ± 0.4	76.84 ± 0.5	83.81 ± 0.3	77.94 ± 0.8	83.95 ± 0.5	78.83 ± 0.4	87.66 ± 0.3	82.37 ± 0.7
TGN	86.78 ± 0.3	78.95 ± 0.4	83.75 ± 0.2	77.76 ± 0.5	87.94 ± 0.1	81.16 ± 0.4	86.59 ± 0.3	80.17 ± 0.3	93.71 ± 0.3	88.21 ± 0.4
MetaDyGNN	83.94 ± 0.2	82.86 ± 0.3	85.62 ± 0.4	81.80 ± 0.4	89.00 ± 0.3	83.20 ± 0.5	85.20 ± 0.4	80.01 ± 0.5	88.56 ± 0.3	85.48 ± 0.3
DySubC	88.61 ± 0.3	80.82 ± 0.2	92.21 ± 0.4	84.57 ± 0.6	92.29 ± 0.5	85.58 ± 0.4	95.24 ± 0.1	89.02 ± 0.3	93.02 ± 0.3	87.91 ± 0.2

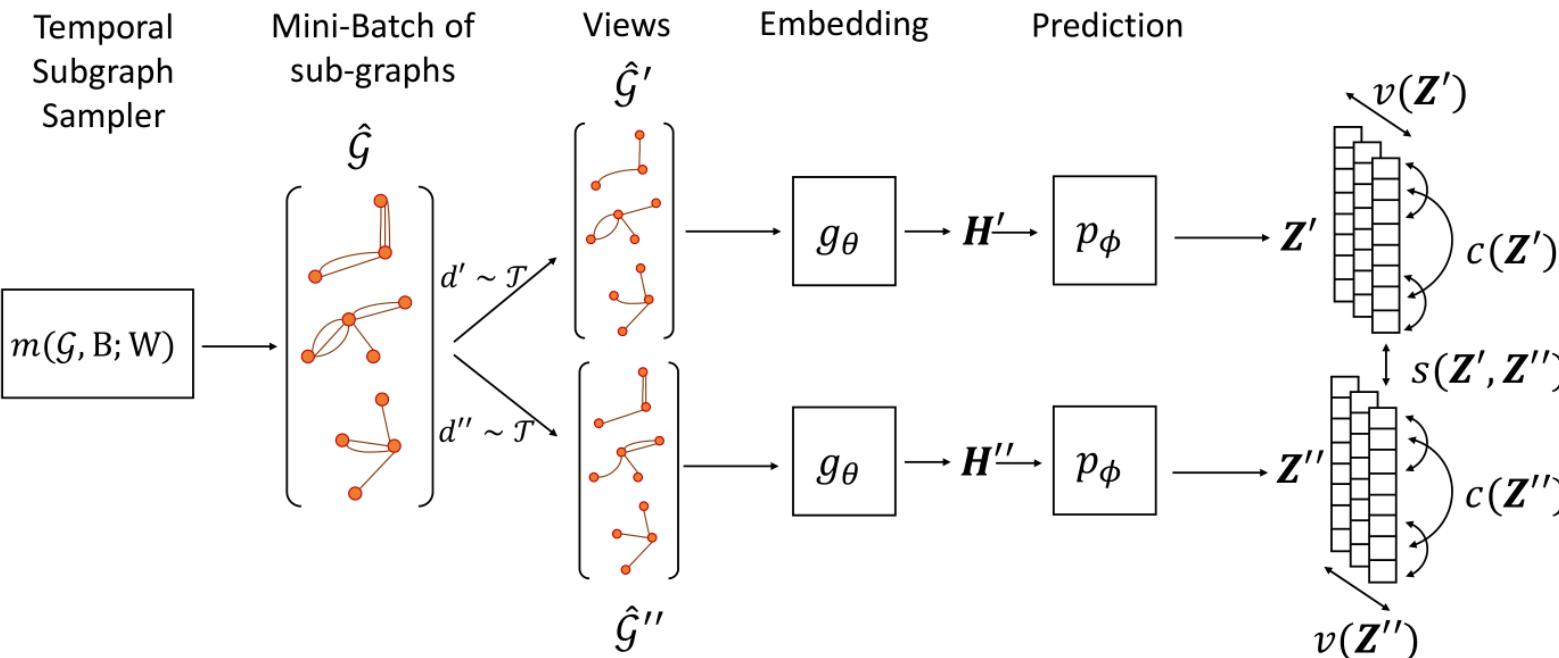
SSL by Multiview Consistency: DyG2Vec

- **Motivation:** Embeddings should be consistent under graph perturbations.
- **New Graph Encoder:** According to ablation study, the subgraph encoder is undirected and non-causal.



SSL by Multiview Consistency: DyG2Vec

- **Motivation:** Embeddings should be consistent under graph augmentations.
- **Method:** Use *edge dropout* and *edge feature masking* to produce different “views”, and use regularization-based SSL loss function



v : maintain variance
 c : bring covariance to zero
 s : minimize distance

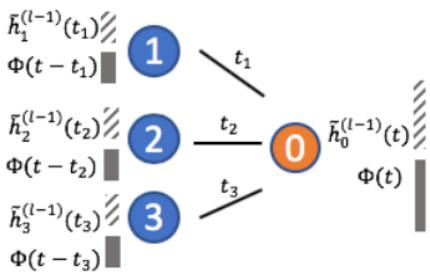
$$\mathcal{L}^{SSL} = \lambda s(\mathbf{Z}', \mathbf{Z}'') + \mu [v(\mathbf{Z}') + v(\mathbf{Z}'')] + \nu [c(\mathbf{Z}') + c(\mathbf{Z}'')]$$

SSL by Multiview Consistency: DyG2Vec

□ Experiments: Link prediction in terms of average AUC score and Accuracy.

Setting	Model	Wikipedia	Reddit	MOOC	LastFM	Enron	UCI	SocialEvol.
Transductive	JODIE	0.956 ± 0.002	0.979 ± 0.001	0.797 ± 0.01	0.691 ± 0.010	0.785 ± 0.020	0.869 ± 0.010	0.847 ± 0.014
	DyRep	0.955 ± 0.004	$0.981 \pm 1e-4$	0.840 ± 0.004	0.683 ± 0.033	0.795 ± 0.042	0.524 ± 0.076	0.885 ± 0.004
	TGAT	0.968 ± 0.001	$0.986 \pm 3e-4$	0.793 ± 0.006	0.633 ± 0.002	0.637 ± 0.002	0.835 ± 0.003	0.631 ± 0.001
	TGN	0.986 ± 0.001	0.985 ± 0.001	0.911 ± 0.010	0.743 ± 0.030	0.866 ± 0.006	0.843 ± 0.090	0.966 ± 0.001
	CaW	0.976 ± 0.007	$0.988 \pm 2e-4$	0.940 ± 0.014	$0.903 \pm 1e-4$	0.970 ± 0.001	0.939 ± 0.008	$0.947 \pm 1e-4$
	NAT	0.987 ± 0.001	0.991 ± 0.001	0.874 ± 0.004	$0.859 \pm 1e-4$	0.924 ± 0.001	0.944 ± 0.002	0.944 ± 0.010
	DyG2Vec	0.995 ± 0.003	$0.996 \pm 2e-4$	0.980 ± 0.002	$0.960 \pm 1e-4$	0.991 ± 0.001	0.988 ± 0.007	$0.987 \pm 2e-4$
Inductive	JODIE	0.891 ± 0.014	0.865 ± 0.021	0.707 ± 0.029	0.865 ± 0.03	0.747 ± 0.041	0.753 ± 0.011	0.791 ± 0.031
	DyRep	0.890 ± 0.002	0.921 ± 0.003	0.723 ± 0.009	0.869 ± 0.015	0.666 ± 0.059	0.437 ± 0.021	$0.904 \pm 3e-4$
	TGAT	0.954 ± 0.001	0.979 ± 0.001	0.805 ± 0.006	0.644 ± 0.002	0.693 ± 0.004	0.820 ± 0.005	0.632 ± 0.005
	TGN	0.974 ± 0.001	0.954 ± 0.002	0.855 ± 0.014	0.789 ± 0.050	0.746 ± 0.013	0.791 ± 0.057	0.904 ± 0.023
	CaW	0.977 ± 0.006	$0.984 \pm 2e-4$	0.933 ± 0.014	0.890 ± 0.001	0.962 ± 0.001	0.931 ± 0.002	$0.950 \pm 1e-4$
	NAT	0.986 ± 0.001	0.986 ± 0.002	$0.832 \pm 1e-4$	0.878 ± 0.003	0.949 ± 0.010	0.926 ± 0.010	0.952 ± 0.006
	DyG2Vec	0.992 ± 0.001	0.991 ± 0.002	0.938 ± 0.010	0.979 ± 0.006	0.987 ± 0.004	0.976 ± 0.002	0.978 ± 0.010

Data-Efficient TGNN Checkpoint

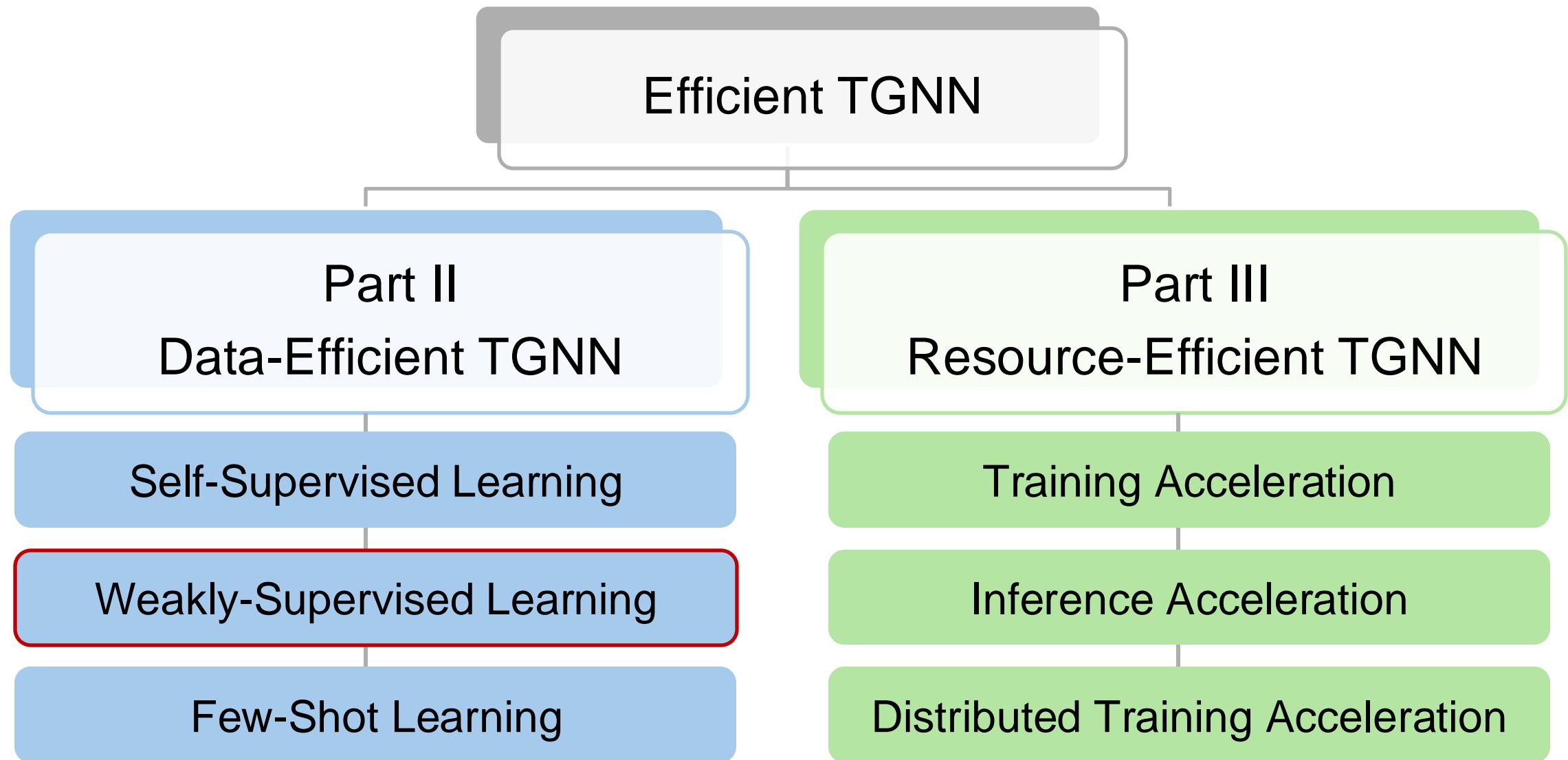


Self-Supervised Learning

- Introduction & Background
- Reconstruction
- Contrastive Learning
- Multiview Approach

Q & A

Scope of This Tutorial

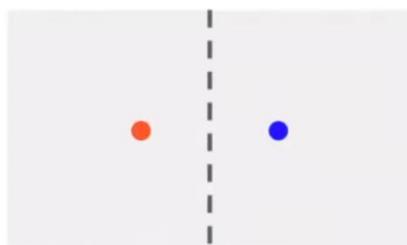


Weakly-Supervised Learning on Temporal Graphs

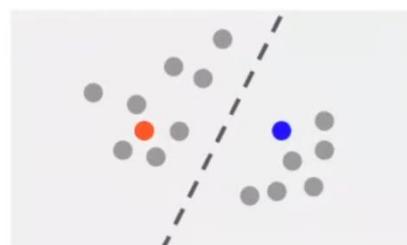
- Introduction & Background
- Weak-Supervision using Limited Labeled Data
- Weak-Supervision on Sparse Temporal Graph

Introduction – Weakly-Supervised Learning

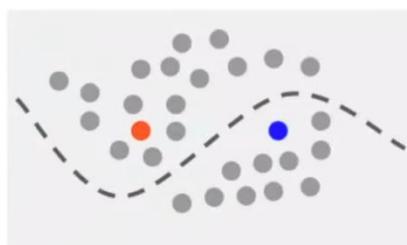
- Learning useful (or better) representations **using limited labeled or noisy data.**
- Inherent structure and temporal dynamics of the graph itself are still important.
- **Challenge 1:** Effectively incorporate labeled data in the training process.
- **Challenge 2:** Learning representations on dynamic and noisy graphs.



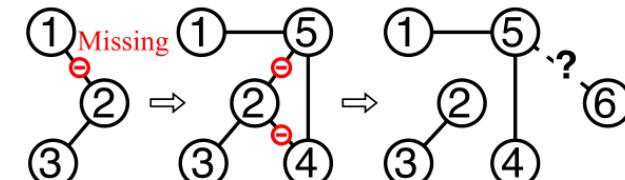
Some labeled data



Some more unlabeled data

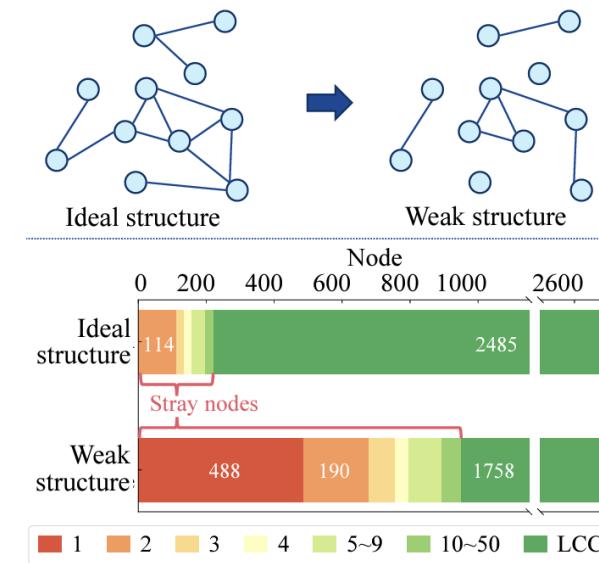
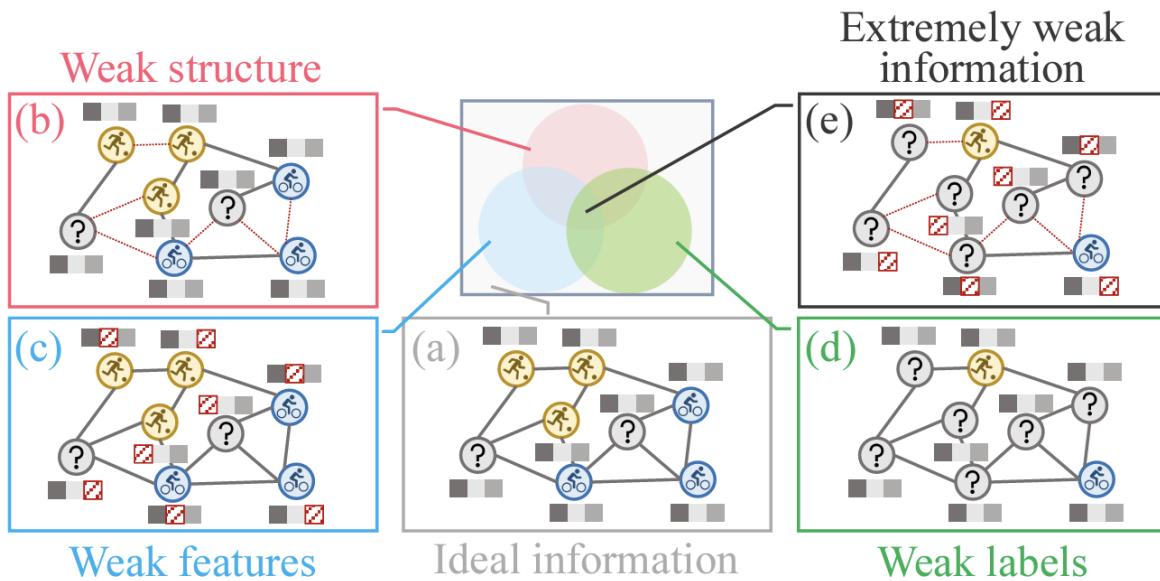


Even more unlabeled data

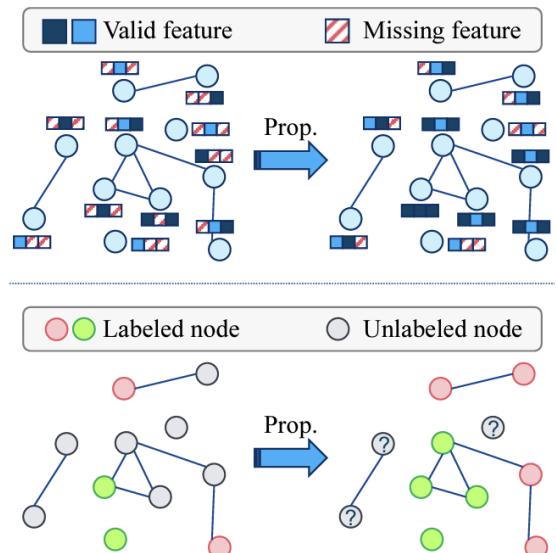


Weak-Supervision on Limited Labeled Data: D²PT

- **Motivation:** Design a universal and effective GNN for graph learning with weak information (GLWI). *Disclaimer: This work is for static graphs.*
- **Method:** Execute effective information propagation in GNNs.



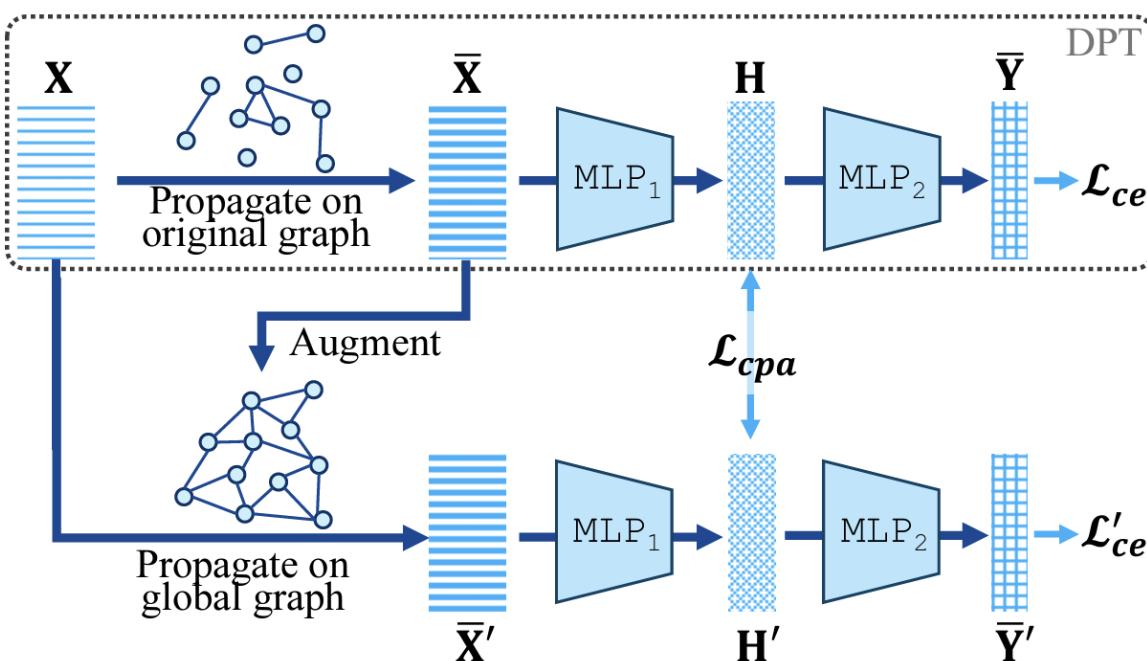
(a) Weak structure leads to stray nodes



(b) Stray nodes hinder feature completion and supervision signal spreading

Weak-Supervision on Limited Labeled Data: D²PT

- **Motivation:** Design a universal and effective GNN for graph learning with weak information (GLWI). *Disclaimer: This work is for static graphs.*
- **Method:** Execute effective information propagation in GNNs.



$$\begin{aligned} \mathbf{X}^{(0)} &= \mathbf{X}, \\ \mathbf{X}^{(t+1)} &= (1 - \alpha) \tilde{\mathbf{A}} \mathbf{X}^{(t)} + \alpha \mathbf{X}, \\ \bar{\mathbf{X}} &= \mathbf{X}^{(T)}, \end{aligned}$$

$$\begin{aligned} p_j &= \sum_{\arg\max_i(\bar{Y}_i)=j} \frac{s_i Z_i}{S_j}, \quad p'_j = \sum_{\arg\max_i(\bar{Y}'_i)=j} \frac{s_i Z'_i}{S_j}, \\ \mathcal{L}_{cpa} &= -\frac{1}{2c} \sum_{j=1}^c \left(\log \frac{f(p_j, p'_j)}{\sum_{q \neq j} f(p_j, p'_q)} + \log \frac{f(p_j, p'_j)}{\sum_{q \neq j} f(p_q, p'_j)} \right) \end{aligned}$$

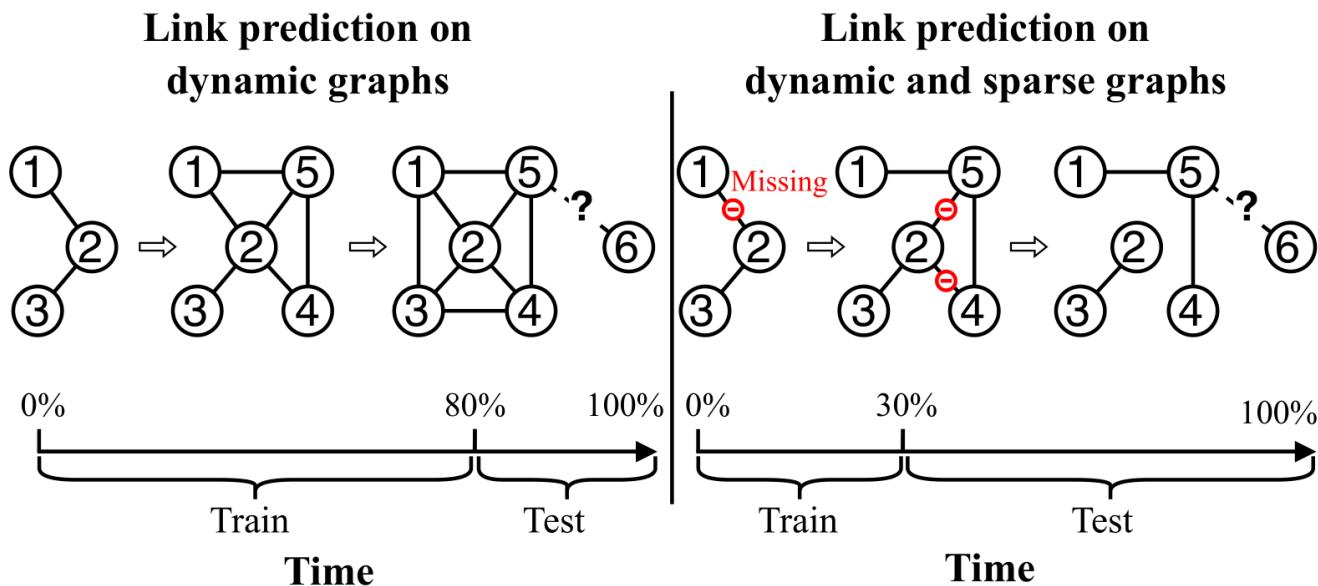
Weak-Supervision on Limited Labeled Data: D²PT

☐ Experiments: Classification accuracy in extreme GLWI scenario.

Methods	Cora	CiteSeer	PubMed	Amz. Photo	Amz. Comp.	Co. CS	Co. Physics	ogbn-arxiv
GCN	46.80±3.20	42.31±3.60	62.30±2.82	83.44±1.61	71.22±1.24	84.14±1.40	88.83±1.44	59.83±0.31
GAT	48.79±5.89	43.86±4.43	63.07±2.82	82.33±1.00	71.81±4.51	82.30±2.01	88.53±1.65	57.47±0.31
APPNP	54.91±5.18	43.59±5.07	64.77±3.56	83.78±1.08	<u>73.46±2.84</u>	84.59±1.73	90.22±0.99	58.59±1.02
SGC	48.06±6.00	40.56±3.51	61.89±3.30	82.12±1.26	71.36±2.12	83.82±1.50	88.20±1.22	54.10±0.73
Pro-GNN	53.28±7.27	44.92±1.62	OOM	OOM	OOM	OOM	OOM	OOM
IDGL	46.95±4.60	45.98±2.64	63.35±4.19	<u>83.85±1.16</u>	73.09±1.94	83.73±1.97	OOM	OOM
GEN	52.72±3.82	49.49±3.70	OOM	OOM	OOM	OOM	OOM	OOM
SimP-GCN	48.72±3.89	48.31±4.45	64.25±3.80	82.94±1.84	71.17±2.06	<u>87.13±1.30</u>	89.72±1.43	OOM
GINN	47.07±3.80	41.82±2.09	61.80±2.97	82.09±1.21	70.33±1.09	81.78±1.25	OOM	OOM
GCN _{MF}	47.13±2.74	43.64±4.72	61.91±4.05	83.22±0.98	73.17±2.11	83.49±1.88	88.56±1.56	55.43±0.66
M3S	56.15±3.89	52.13±5.03	63.65±5.01	82.64±2.96	71.96±1.49	83.25±2.07	88.99±3.04	OOM
CGPN	59.29±1.54	<u>54.62±2.74</u>	64.47±1.17	82.60±2.36	72.36±2.70	82.92±2.54	OOM	OOM
Meta-PN	53.05±5.97	45.60±5.84	64.88±4.02	82.46±1.91	72.19±3.11	83.95±2.43	89.28±1.01	56.32±0.86
GRAND	<u>63.38±1.40</u>	46.23±4.44	63.97±3.82	82.66±4.85	71.84±5.20	84.93±1.50	90.33±0.67	55.49±0.47
DPT	56.37±5.97	46.06±4.56	<u>65.08±3.13</u>	82.84±1.46	73.13±1.70	84.32±2.10	<u>90.42±1.13</u>	<u>60.87±0.18</u>
D ² PT	66.00±1.20	56.99±2.23	66.43±2.45	84.12±1.66	74.96±1.79	<u>85.62±1.75</u>	91.41±0.89	61.59±0.59

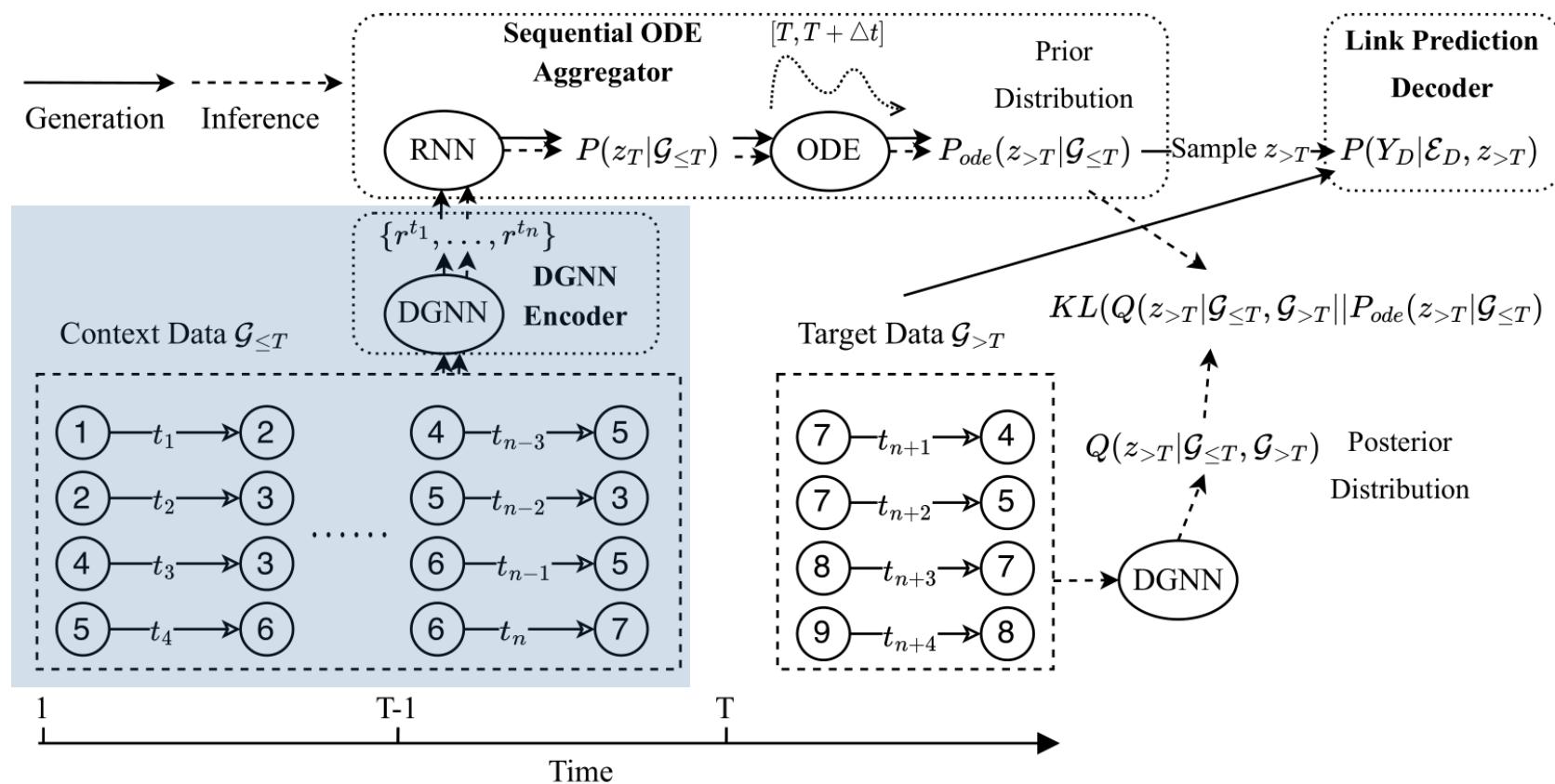
Weak-Supervision on Sparse Temporal Graph: GSNOP

- **Objective:** Address the situation where there is not enough historical data.
- **Motivation:** Missing links are common. How to learn better representation on sparse graphs and prevent overfitting.



Weak-Supervision on Sparse Temporal Graph: GSNOP

- Method: Treat the link prediction as a dynamic-changing stochastic process and employ neural process.



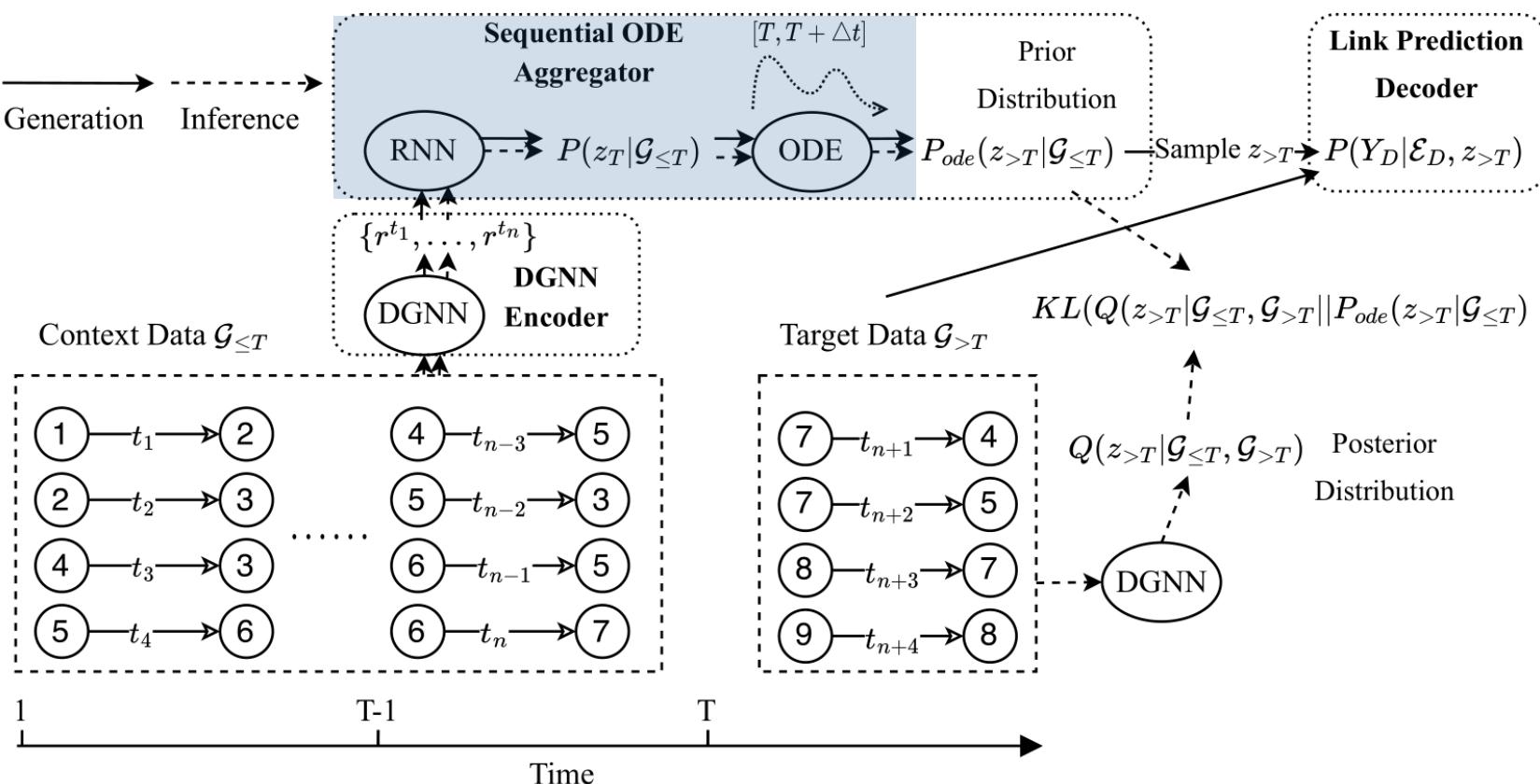
$$h^t = \text{DGNN}\left(h^{t-1}, \text{agg}(N_{<t}(v))\right)$$

$$r^t = \text{MLP}\left(h_i^t || h_j^t || y\right) + t_{emb}$$

$$y = \begin{cases} 1, e_C(t) \in \mathcal{G}_{\leq T} \\ 0, e_C(t) \notin \mathcal{G}_{\leq T} \end{cases}$$

Weak-Supervision on Sparse Temporal Graph: GSNOP

- Method: Treat the link prediction as a dynamic-changing stochastic process and employ neural process.



$$r^t = \text{MLP}(h_i^t || h_j^t || y) + t_{emb}$$

$$\mathbf{r}_t = \text{RNN}(\mathbf{r}_{t-1}, \mathcal{G}_t), t > 1,$$

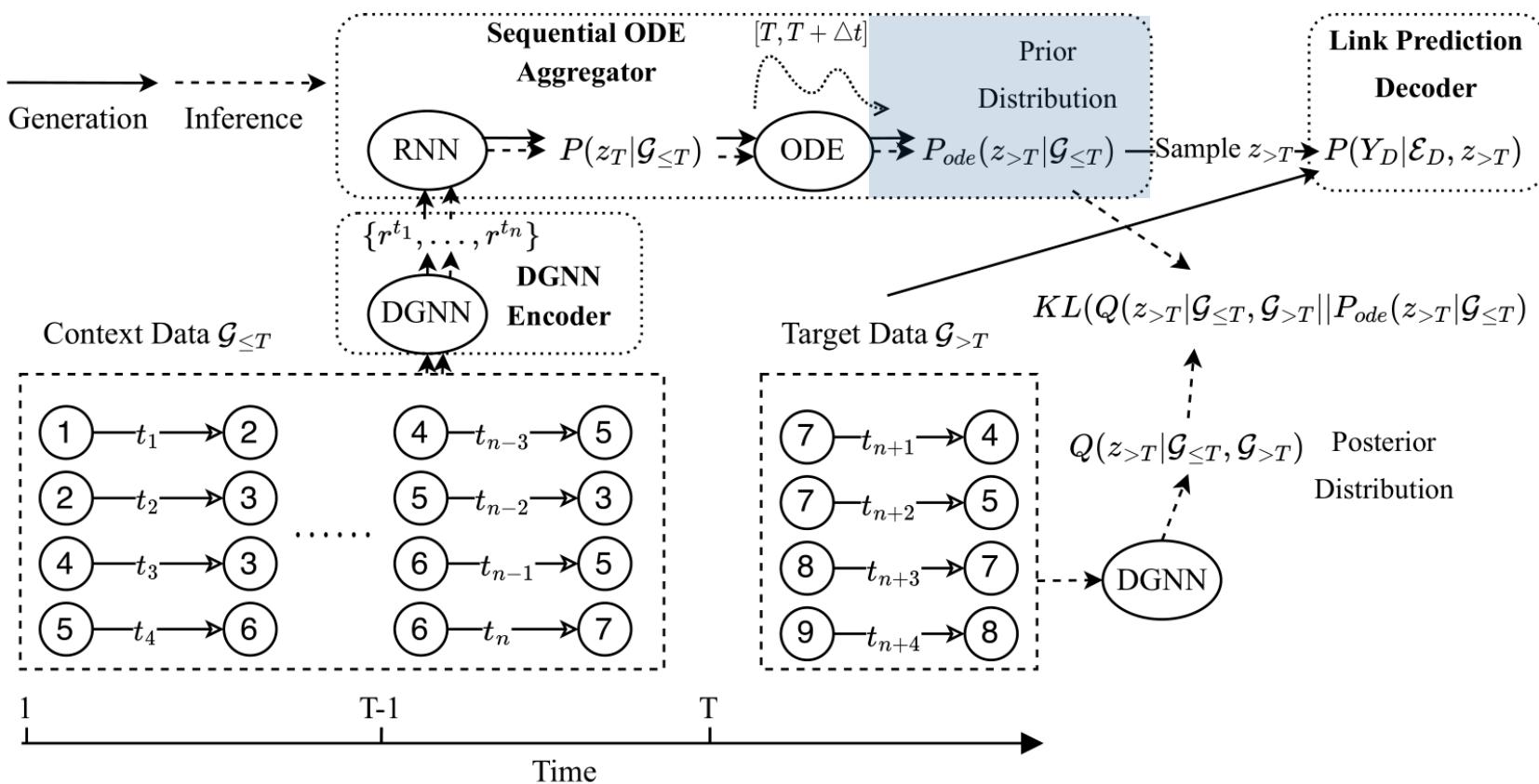
$$\mathbf{r}_1 = \frac{1}{|\mathcal{G}_{\leq 1}|} \sum_{e_C(t) \in \mathcal{G}_{\leq 1}} t^1$$

$$\mathbf{r}_{>T} = \mathbf{r}_T + \int_T^{T+\Delta t} f_{ode}(\mathbf{r}_t, t) dt$$

$$f_\theta(\mathbf{r}_t, t) = \text{Tanh}(\text{MLP}(\mathbf{r}_t + t_{emb}))$$

Weak-Supervision on Sparse Temporal Graph: GSNOP

- Method: Treat the link prediction as a dynamic-changing stochastic process and employ neural process.



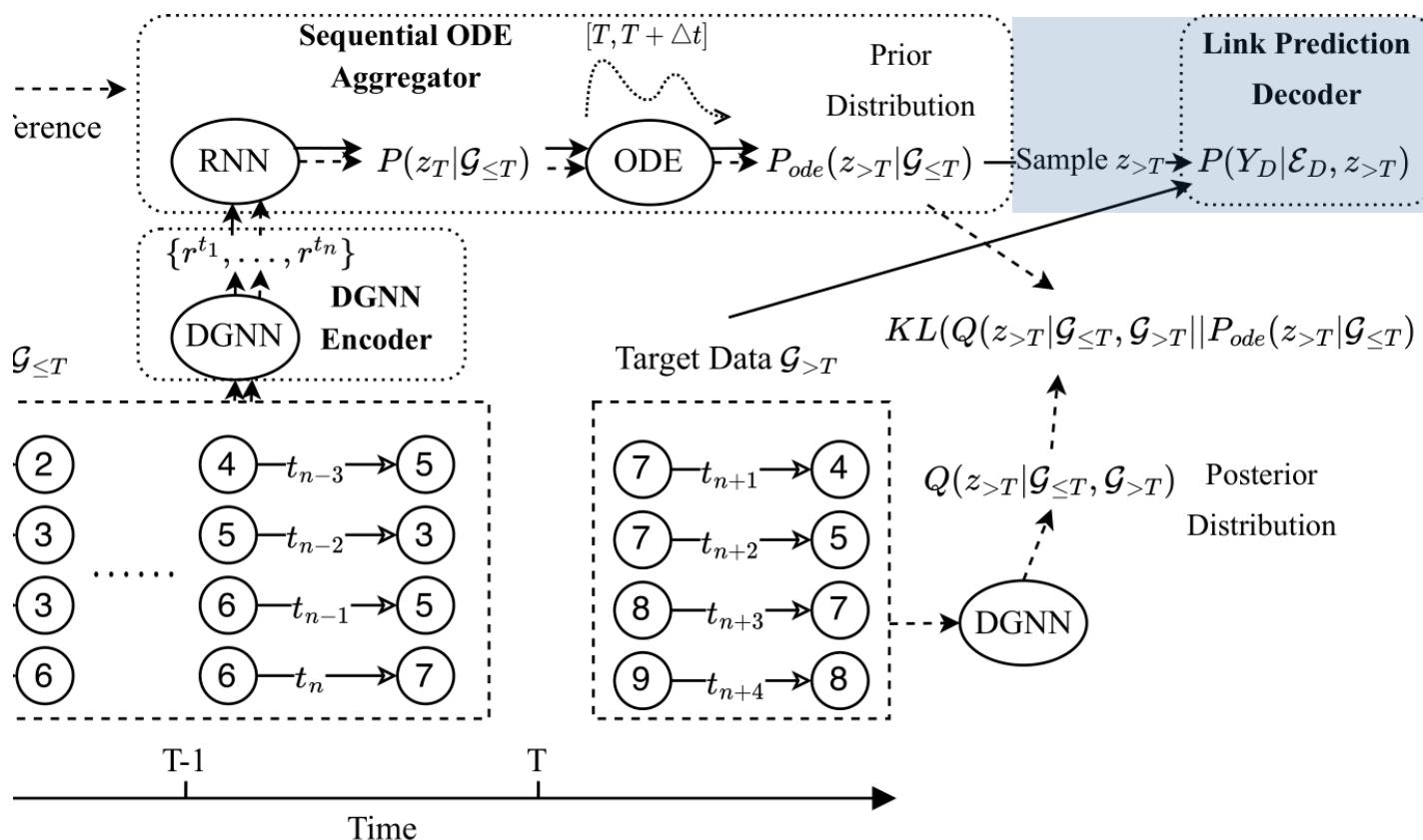
$$\chi = \text{MLP}(\mathbf{r}_{>T}),$$

$$\mu(\mathbf{r}_{>T}) = \text{ReLU}(\text{MLP}(\chi)),$$

$$\sigma(\mathbf{r}_{>T}) = 0.1 + 0.9 * \text{Sigmoid}(\text{MLP}(\chi)).$$

Weak-Supervision on Sparse Temporal Graph: GSNOP

- Method: Treat the link prediction as a dynamic-changing stochastic process and employ neural process.



$$h_i^t = \text{DGNN}(v_i, N_{<t}(v_i)), h_j^t = \text{DGNN}(v_j, N_{<t}(v_i))$$

$$\text{Sample } z_{>T} \sim \mathcal{N}(\mu(\mathbf{r}_{>T}), \sigma(\mathbf{r}_{>T}))$$

$$\tilde{h}_i^t = \text{ReLU}(\text{MLP}(h_j^t || z_{>T})), \tilde{h}_j^t = \text{ReLU}(\text{MLP}(h_j^t || z_{>T}))$$

$$\hat{y} = \text{Sigmoid}(\text{MLP}(\tilde{h}_i^t || \tilde{h}_j^t)),$$

Weak-Supervision on Sparse Temporal Graph: GS NOP

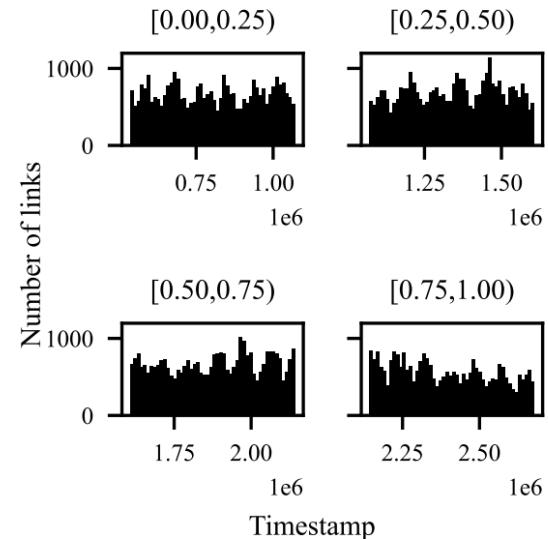
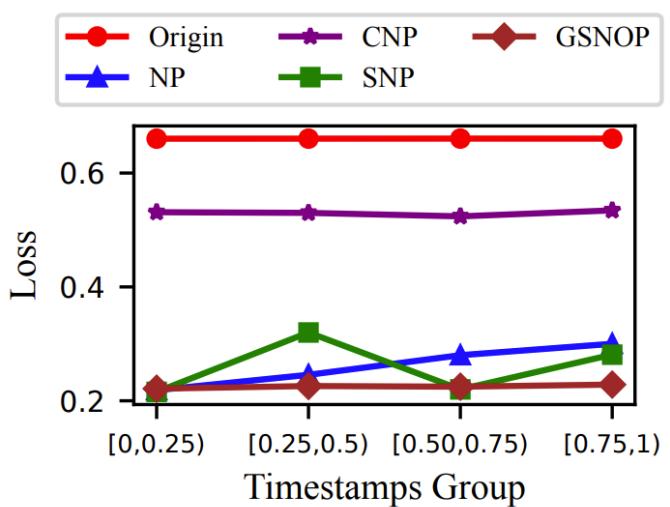
- Experiments: Split the graph with 30%-20%-50% and 10%-10%-80% ratio. Results on multiple datasets. And comparison with other neural process variant

Method	WIKI_0.3			REDDIT_0.3			MOOC_0.3			WIKI_0.1			REDDIT_0.1			MOOC_0.1		
	AP	MRR	AP	MRR	AP	MRR	AP	MRR										
JODIE	0.3329	0.5595	0.6320	0.8101	0.6690	0.8802	0.1820	0.4476	0.3324	0.6401	0.7100	0.9063						
JODIE+GS NOP	0.4258	0.6086	0.8171	0.8922	0.6716	0.8853	0.2979	0.5273	0.4705	0.7260	0.7196	0.9066						
DySAT	0.3881	0.6680	0.5705	0.7887	0.6441	0.8807	0.3872	0.6601	0.5353	0.7853	0.6705	0.8968						
DySAT+GS NOP	0.5816	0.7716	0.7406	0.8443	0.6528	0.8811	0.3910	0.6628	0.5738	0.7887	0.6718	0.8971						
TGAT	0.3253	0.5229	0.8343	0.8797	0.5183	0.7828	0.2766	0.4638	0.4833	0.6523	0.1919	0.4228						
TGAT+GS NOP	0.3701	0.5348	0.8395	0.8820	0.5448	0.7905	0.3366	0.4961	0.5240	0.6807	0.1874	0.4581						
TGN	0.5541	0.7788	0.7621	0.8587	0.7200	0.9007	0.5676	0.7631	0.6749	0.8029	0.7677	0.9087						
TGN+GS NOP	0.6633	0.7816	0.8307	0.8935	0.7445	0.9117	0.6568	0.7674	0.7148	0.8263	0.7714	0.9174						
APAN	0.2431	0.5496	0.6166	0.7799	0.6601	0.8774	0.0504	0.1857	0.5601	0.7415	0.7016	0.8967						
APAN+GS NOP	0.4570	0.6918	0.7119	0.8560	0.6614	0.8790	0.1996	0.5607	0.6126	0.7606	0.7052	0.8982						

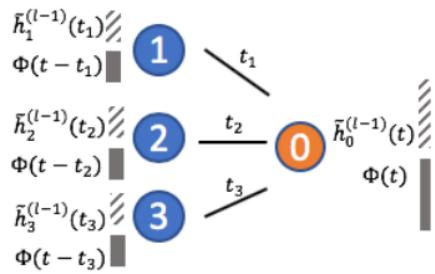
Method	WIKI_0.3									
	Origin		NP		CNP		SNP		GS NOP	
	AP	MRR	AP	MRR	AP	MRR	AP	MRR	AP	MRR
JODIE	0.3868	0.6241	0.4063	0.6118	0.3255	0.5943	0.4173	0.6023	0.4258	0.6086
DySAT	0.3757	0.6727	0.4226	0.6818	0.3494	0.6552	0.4199	0.6821	0.5816	0.7716
TGAT	0.3277	0.5171	0.3456	0.5147	0.2881	0.4680	0.3462	0.5173	0.3701	0.5348
TGN	0.5611	0.7789	0.6632	0.7837	0.6539	0.7766	0.6577	0.7809	0.6633	0.7816
APAN	0.2191	0.5085	0.0784	0.2701	0.0987	0.3195	0.1090	0.3503	0.4570	0.6918

Weak-Supervision on Sparse Temporal Graph: GS NOP

- Experiments: Demonstrate the effectiveness of Sequential ODE aggregator. GS NOP models the distribution derivative and maintain relatively low losses across different timestamps.

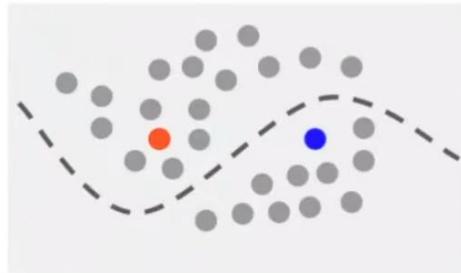


Data-Efficient TGNN Checkpoint



Self-Supervised Learning

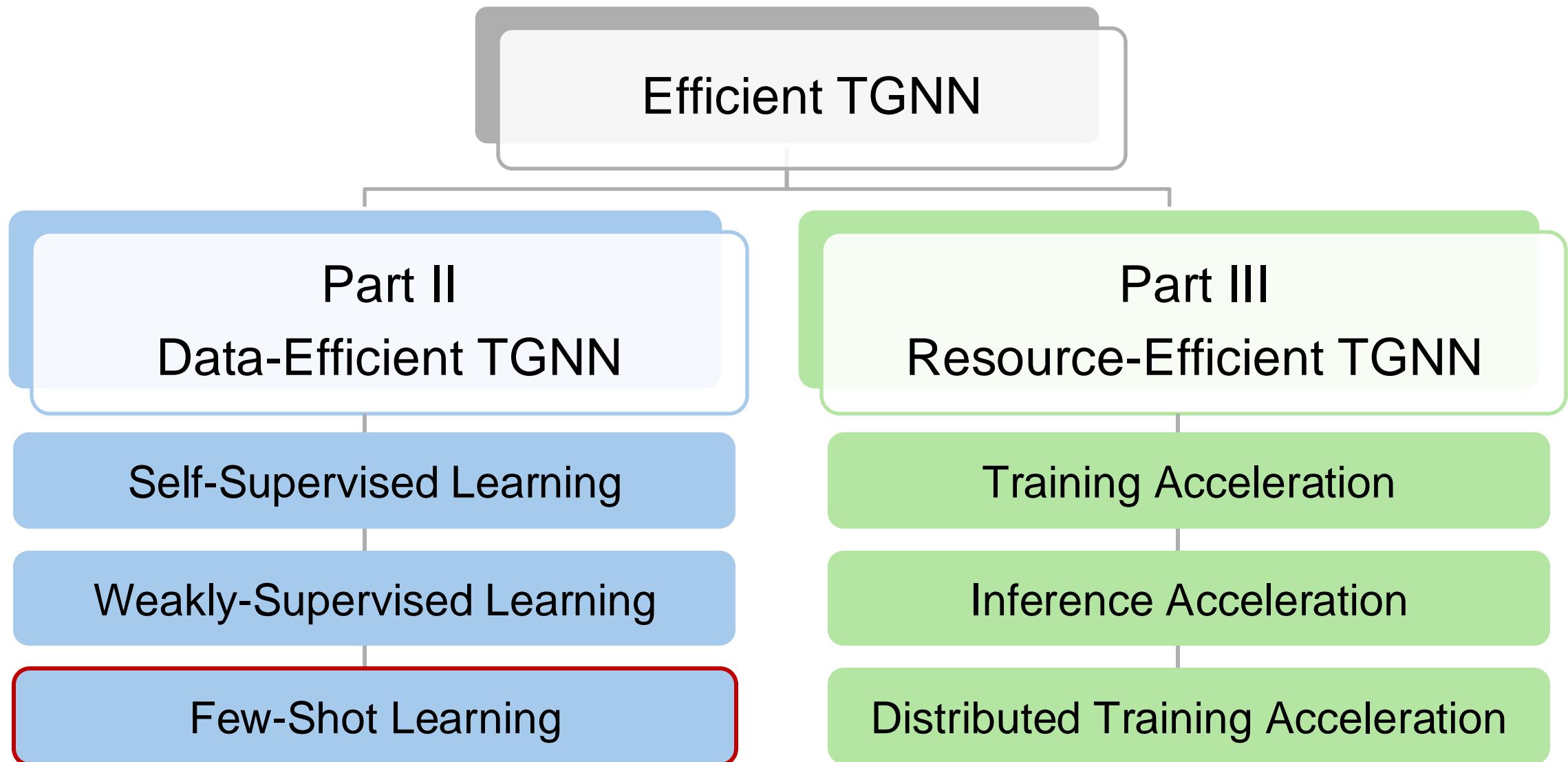
- Introduction & Background
- Future Link Reconstruction
- Contrastive Learning
- Multiview Approach



Weakly-Supervised Learning

- Introduction & Background
- Limited Labeled Data
- Sparse Temporal Graph

Scope of This Tutorial



Few-Shot Learning on Temporal Graphs

- **Introduction & Background**
- **Few-Shot Learning for New Nodes**
- **Few-Shot Learning for New Time Intervals**
- **Few-Shot Learning for New Classes**

Introduction – Few-Shot Learning

- ❑ New tasks and new data distributions are emerging as time goes by, where each new task is associated limited amount of labeled data.
- ❑ What are new tasks?

- ❑ New nodes



A new politician



A new user



A new post

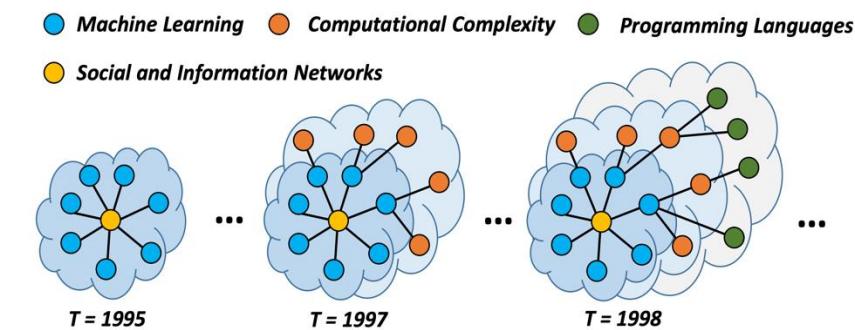
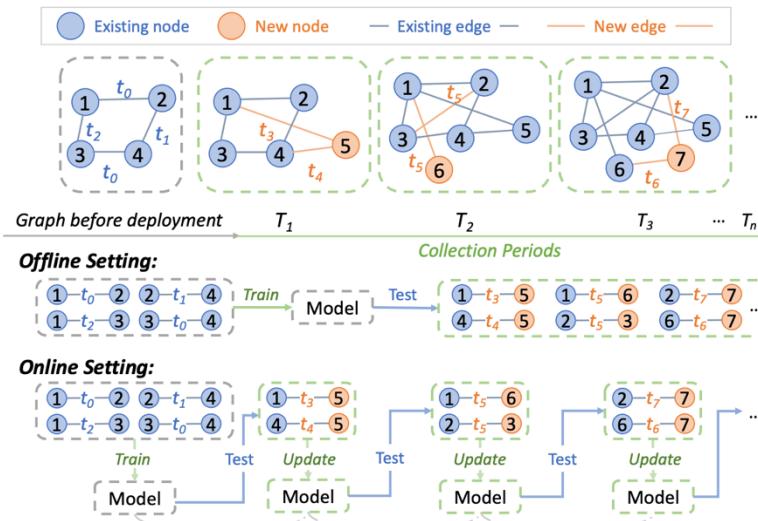


A new product



A new query

- ❑ New time interval



Background: Meta-Learning for Few-Shot Learning

□ Meta-learning formulation

Training task 1

Support set



Query set



Training task 2 . . .

Support set



Query set



Test task 1 . . .

Support set

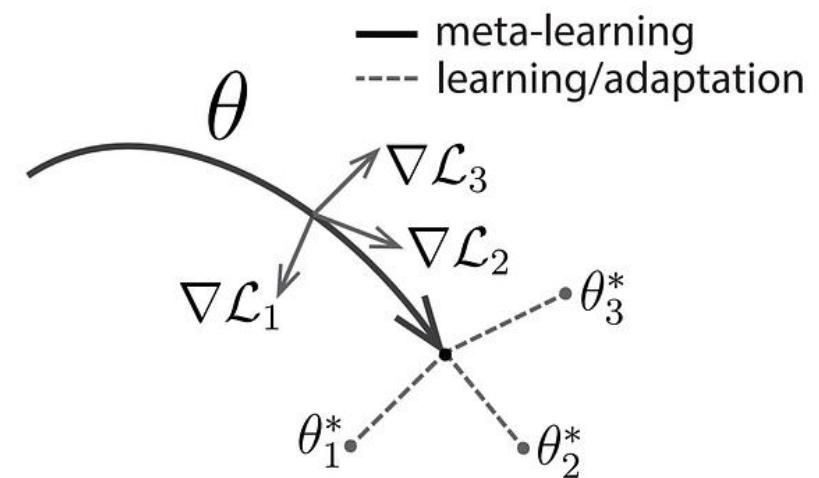
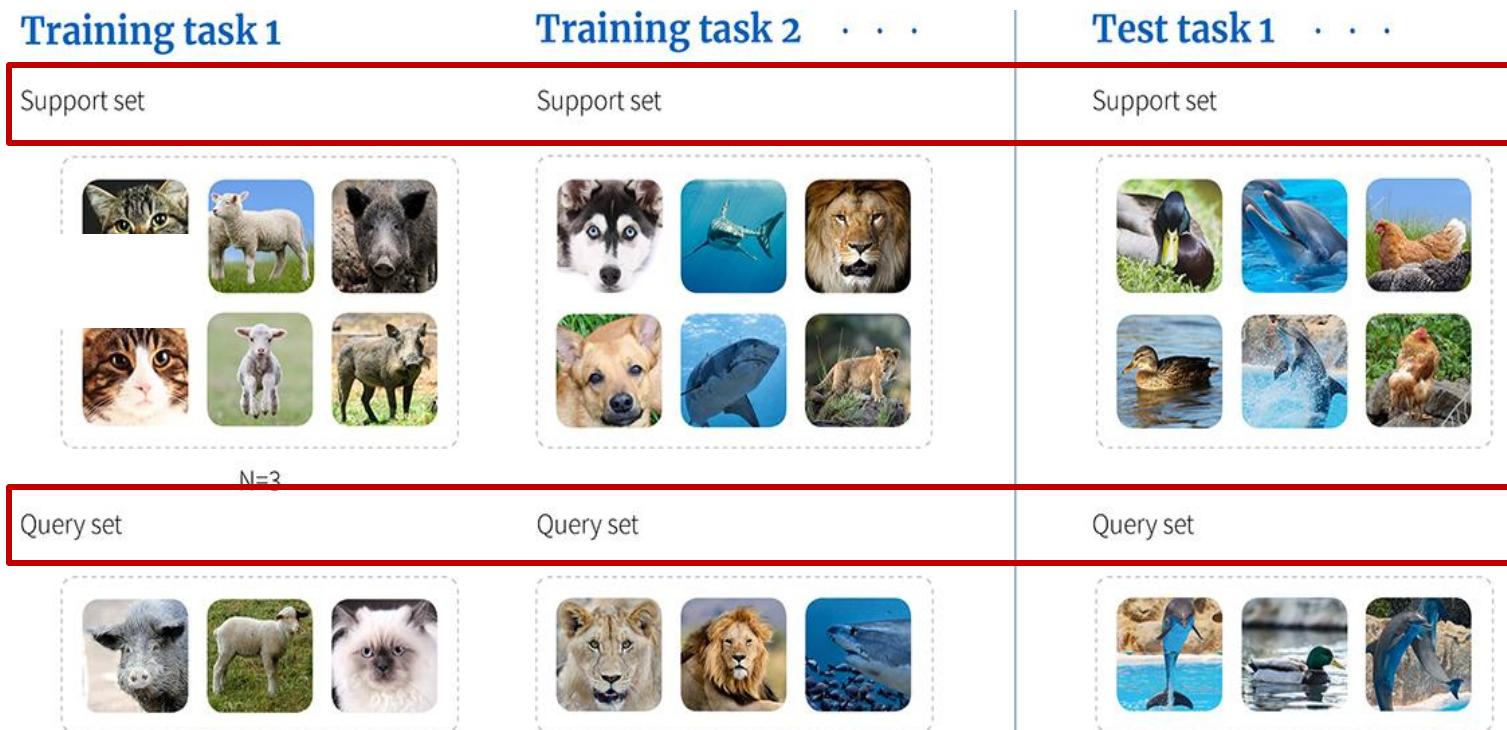


Query set



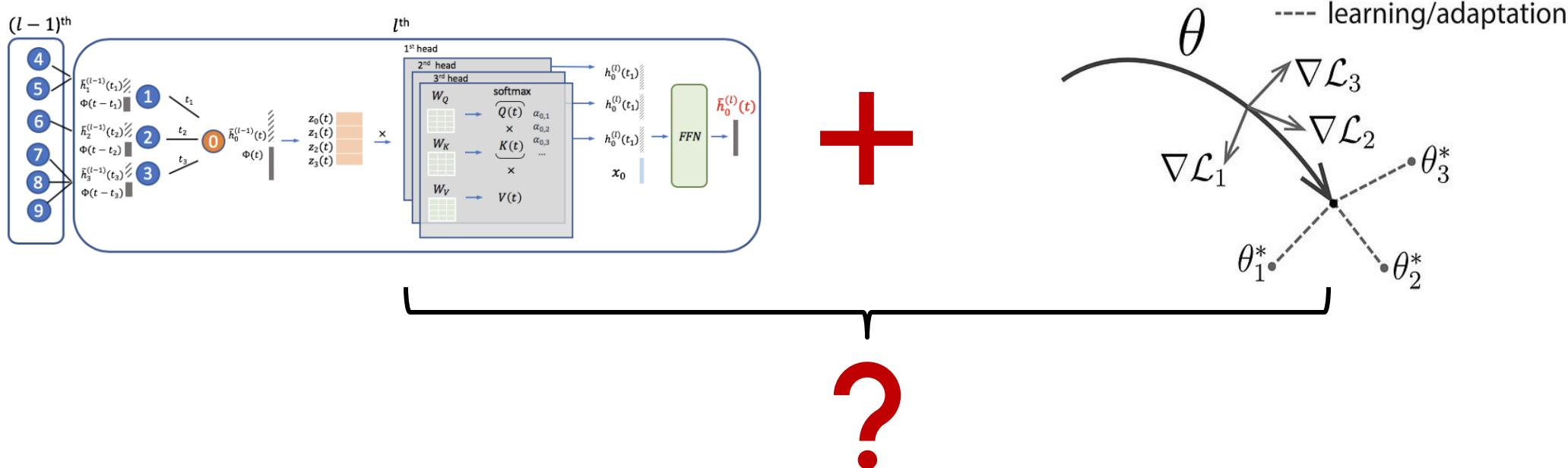
Background: Meta-Learning for Few-Shot Learning

- Model-agnostic meta-learning (MAML)
 - Support set & query set
 - Bi-level optimization



Challenges on Temporal Graphs

- **Challenge 1:** optimize model for new tasks from insufficient information.
- **Challenge 2:** design meta-learning algorithms that are compatible with TGNNs.
- **Challenge 3:** learn temporally robust parameters against distribution evolution over time.



New Nodes on Temporal Graphs

- New nodes continuously join graphs:



A new politician



A new user



A new post

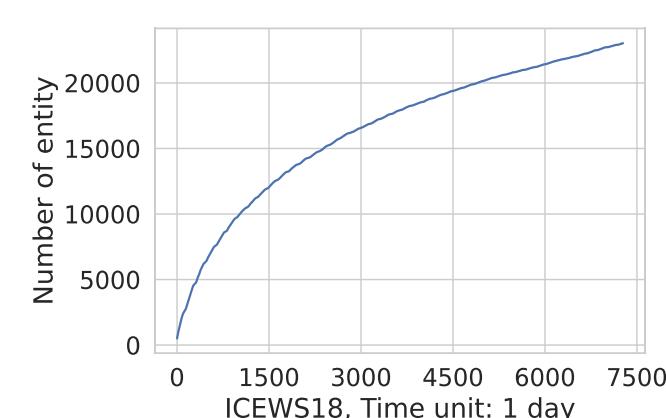
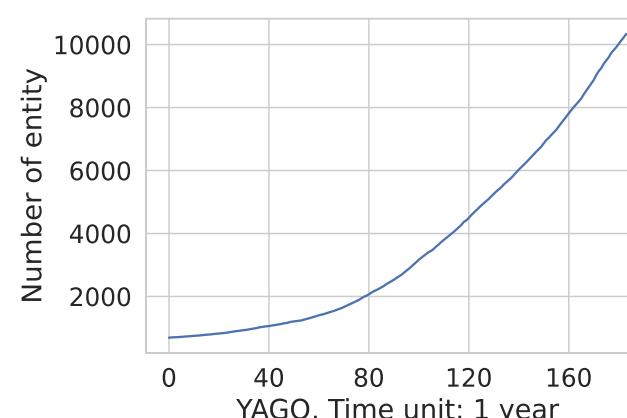
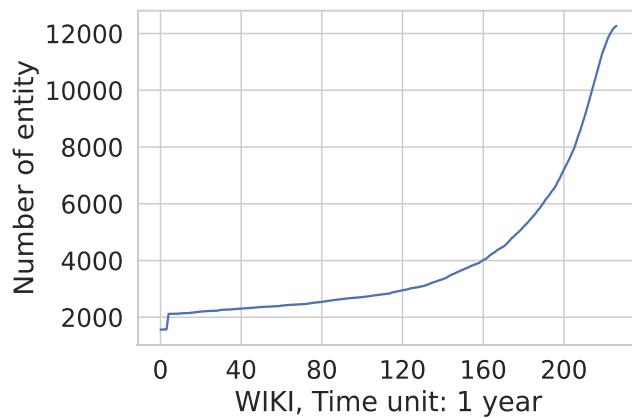


A new product



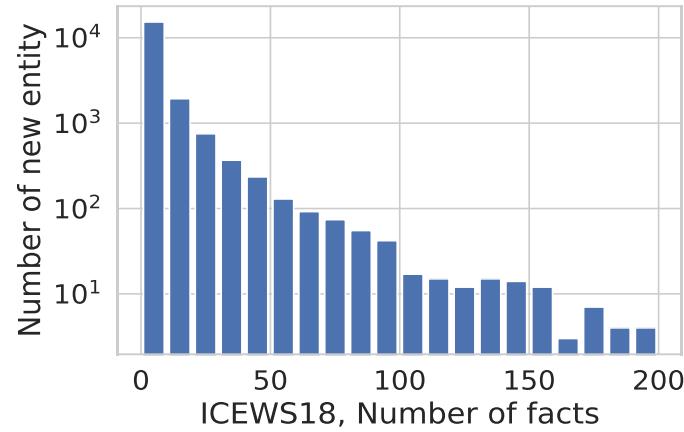
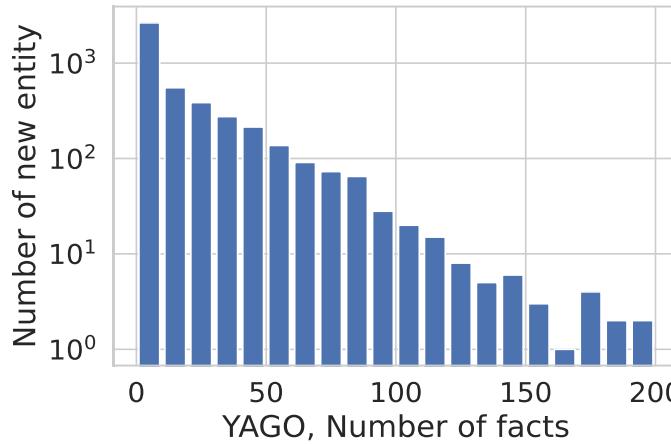
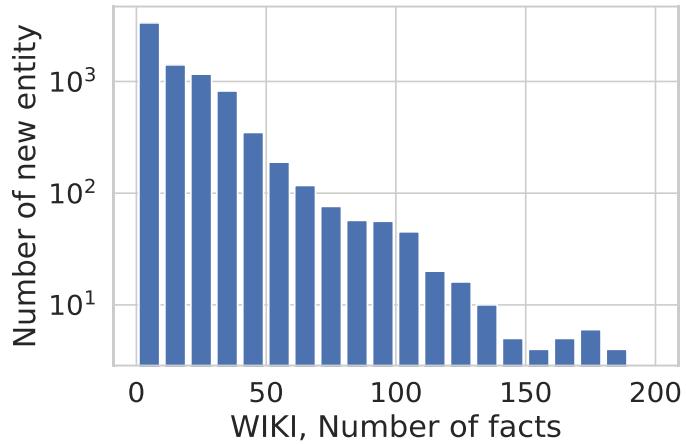
A new query

- 61.3% of WIKI entities are new entities (firstly appear in the last 25% time steps):



New Nodes on Temporal Graphs

- Can we directly utilize existing reasoning methods on new entities?
- New nodes are initially associated with a few links:



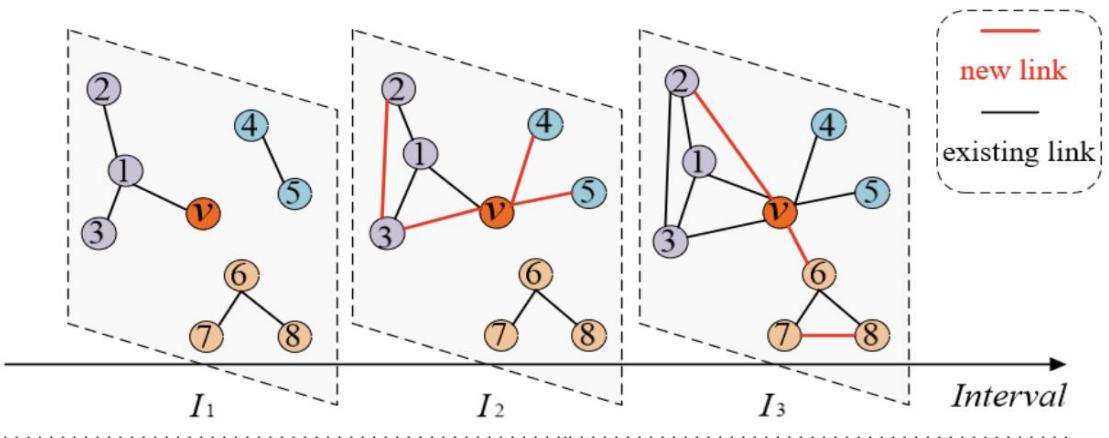
Few-shot link prediction on TGs

Predict future links for newly emerging entities based on extremely limited observations on temporal graphs.

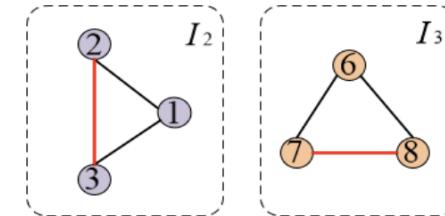
Few-Shot Learning for New Nodes: MetaDyGNN

- **Objective:** Address challenge 1&2
 - Challenge 1: optimize model for new tasks from insufficient information.
 - Challenge 2: design meta-learning algorithms that are compatible with TGNNs.
- **Motivation:** node-wise & time-wise general knowledge

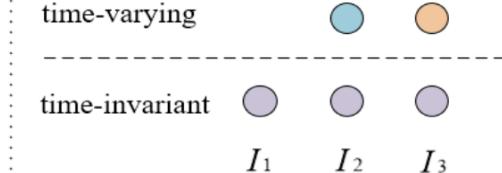
(a) The link formation process of nodes in a dynamic network



(b) General knowledge across nodes

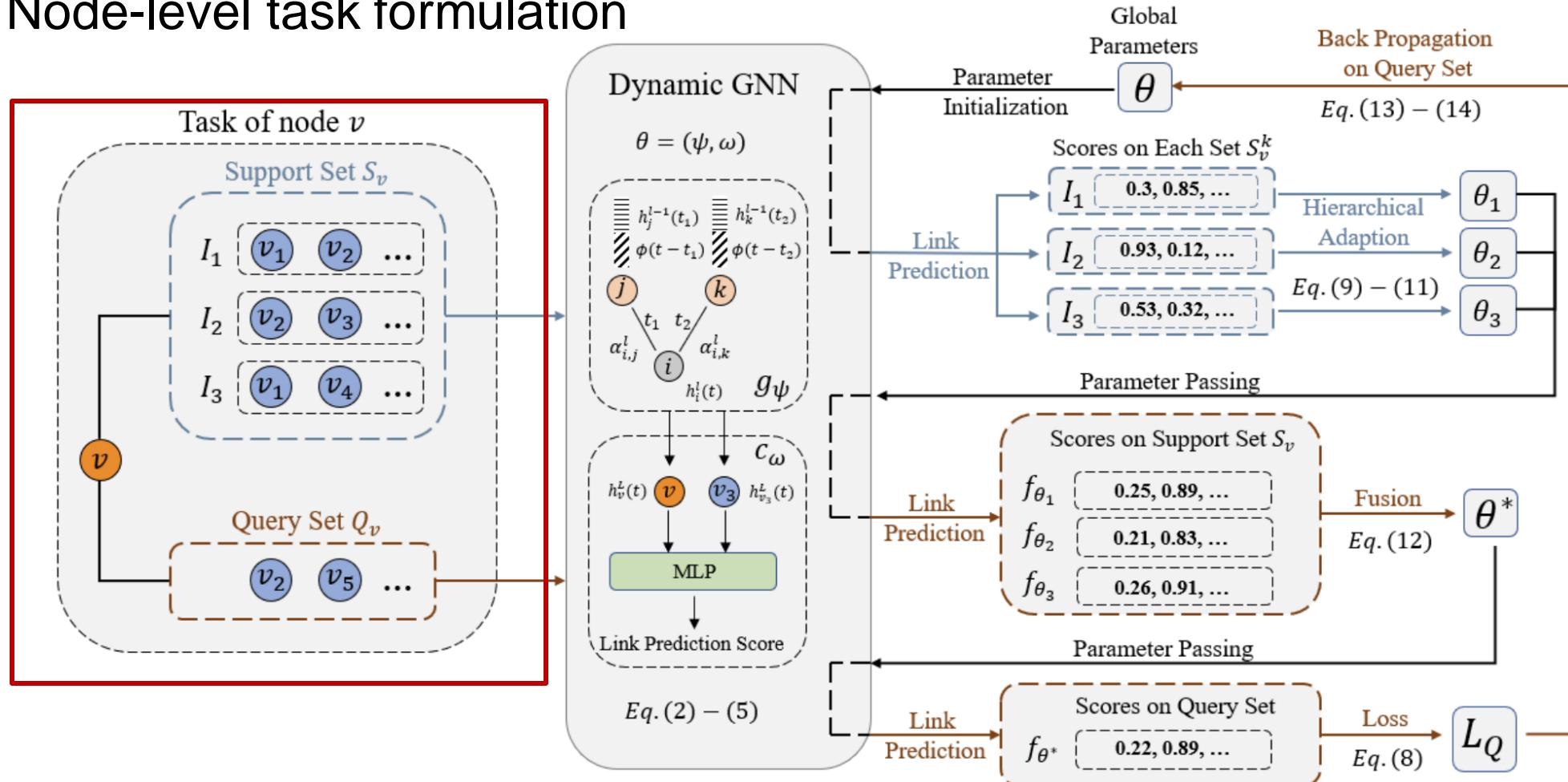


(c) General knowledge across intervals for node v 's preference



Few-Shot Learning for New Nodes: MetaDyGNN

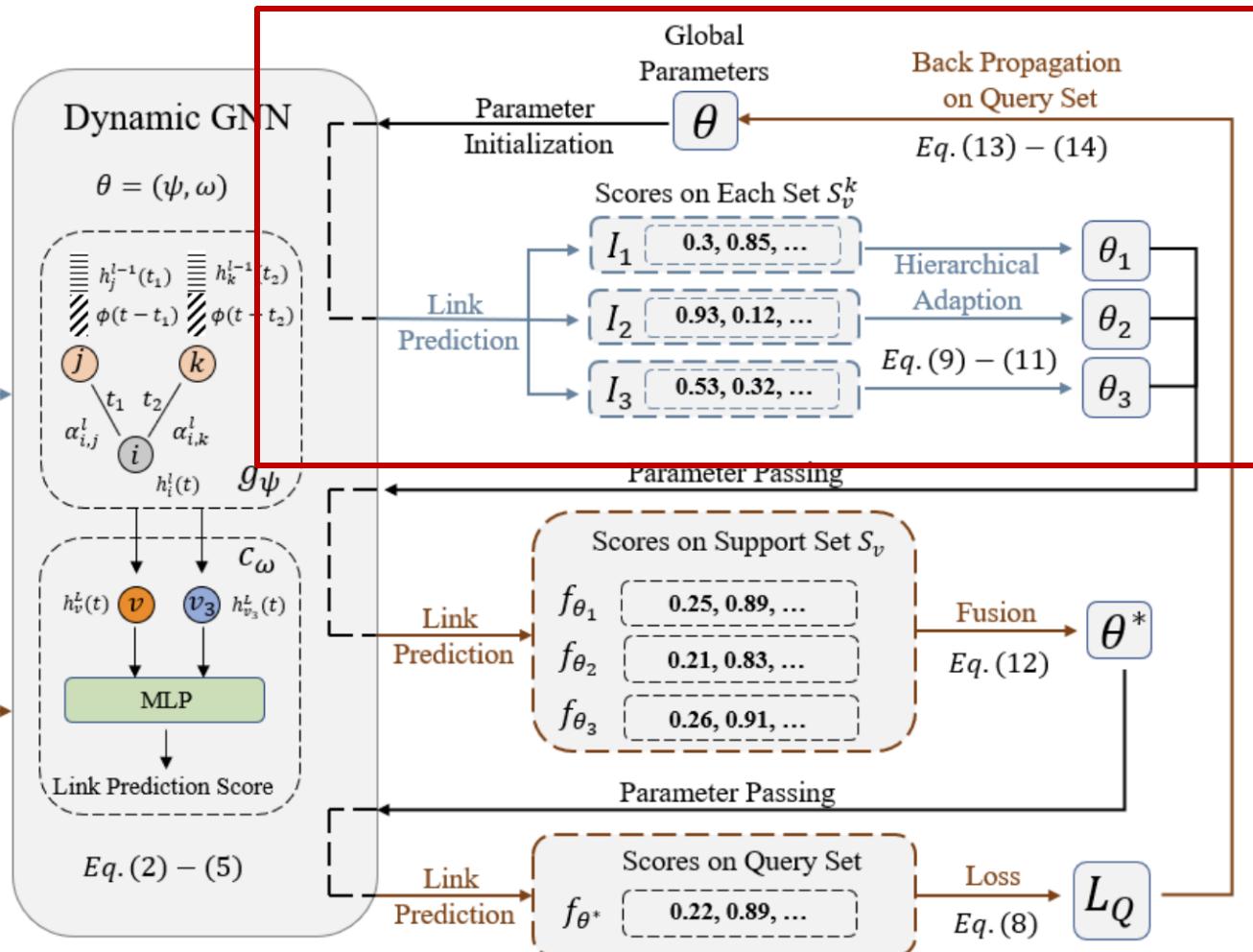
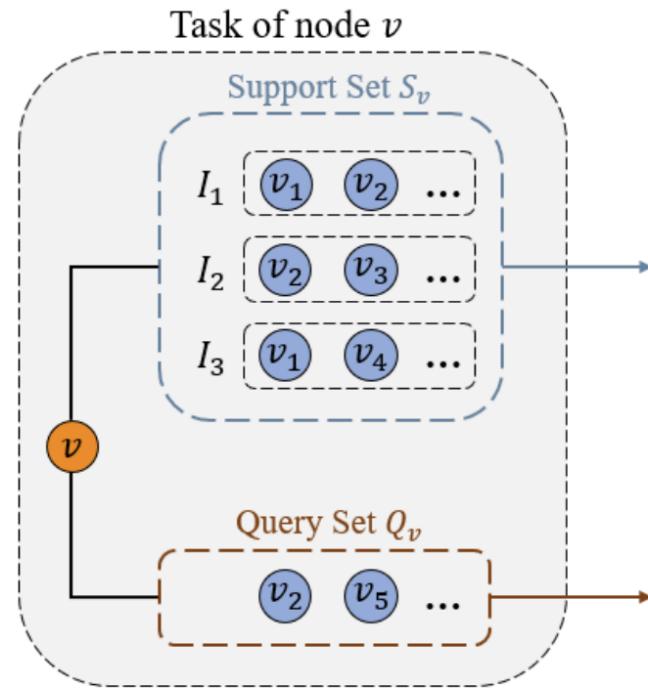
- Method: customized MAML with TGAT backbone
- Node-level task formulation



Few-Shot Learning for New Nodes: MetaDyGNN

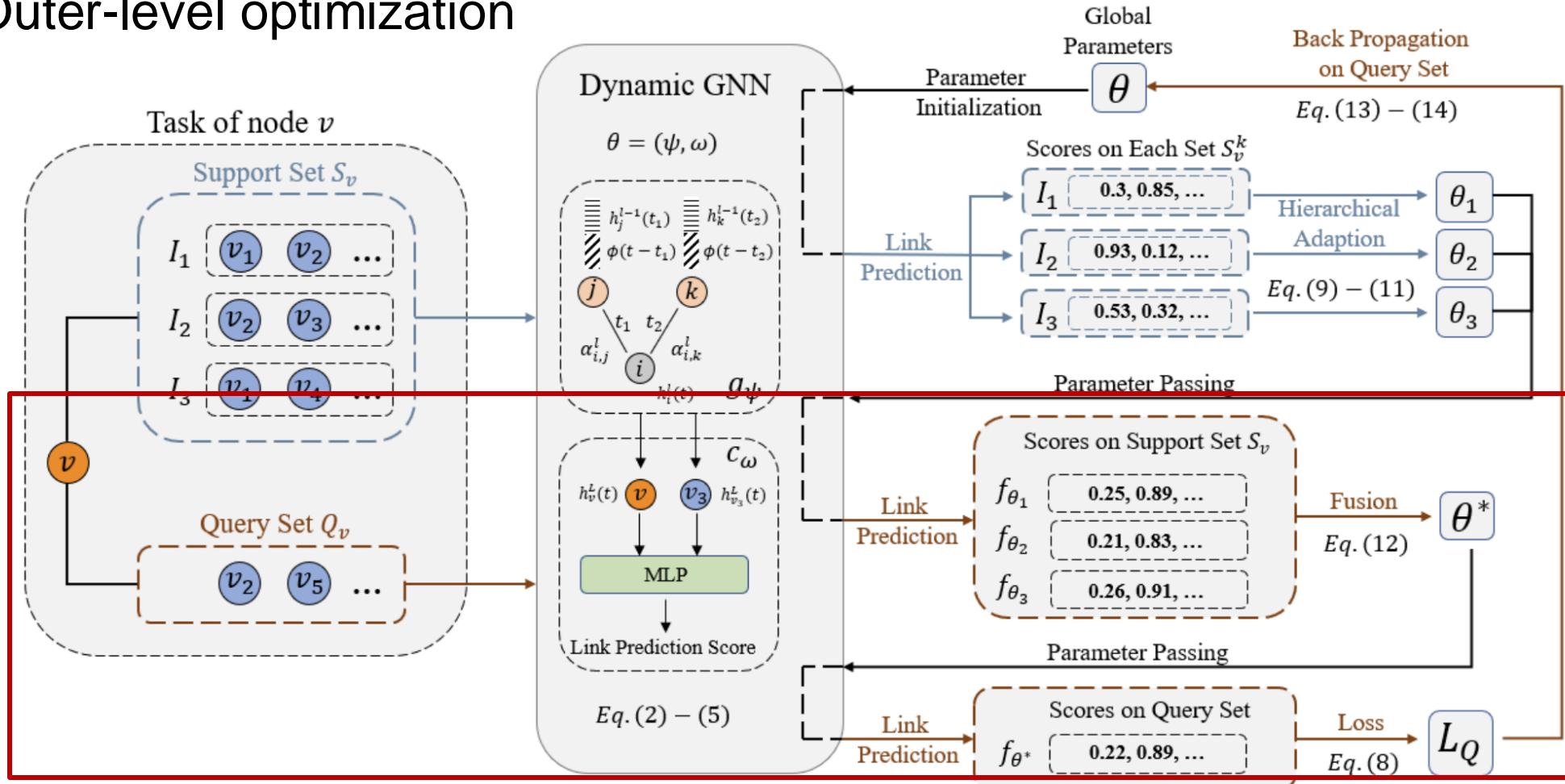
- Method: customized MAML with TGAT backbone

- Node-wise adaptation
- Time-wise adaptation



Few-Shot Learning for New Nodes: MetaDyGNN

- Method: customized MAML with TGAT backbone
- Outer-level optimization



Few-Shot Learning for New Nodes: MetaDyGNN

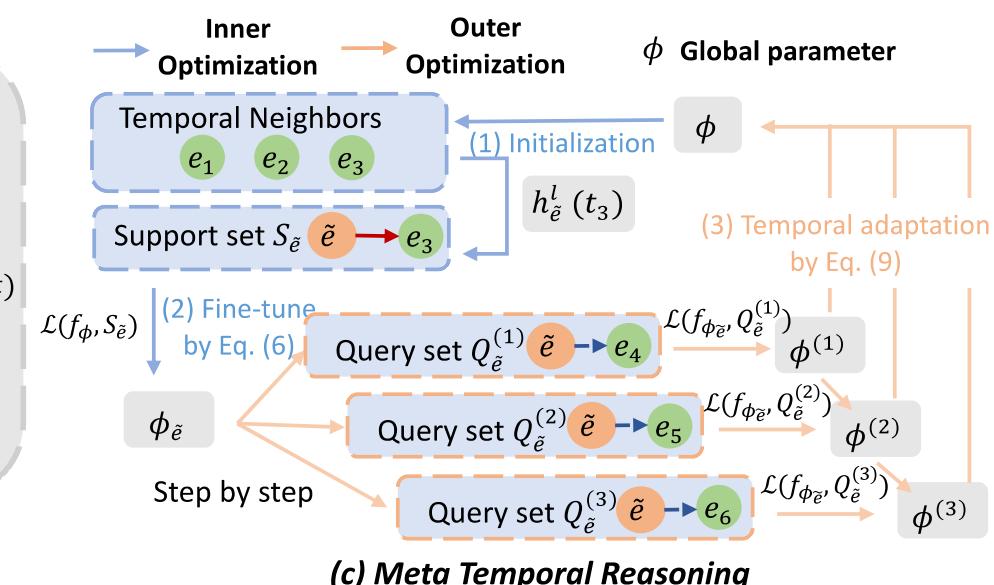
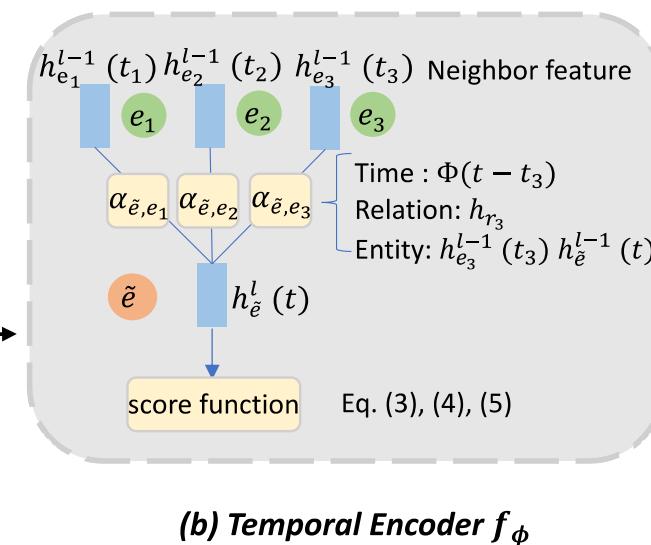
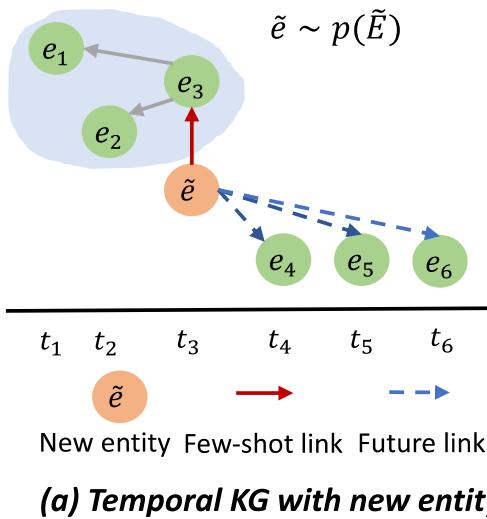
☐ Experiments: 2/4/5-shot link prediction

Model/Result	Wikipedia			Reddit			DBLP		
	ACC(%)	AUC(%)	Macro-F1(%)	ACC(%)	AUC(%)	Macro-F1(%)	ACC(%)	AUC(%)	Macro-F1(%)
GraphSAGE	75.84	77.17	74.76	89.32	93.21	88.28	72.17	76.35	71.12
GAT	76.03	78.76	74.94	89.86	93.36	88.64	73.18	77.16	72.56
DyRep	58.01	62.54	56.93	62.14	66.08	61.27	59.14	66.73	57.79
EvolveGCN	56.36	63.15	54.27	58.21	62.64	57.13	57.98	64.14	56.92
TGAT	87.15	90.44	86.89	93.53	95.45	93.27	76.32	81.13	76.26
Meta-GNN	78.92	80.96	77.13	86.27	90.87	85.44	75.48	79.64	74.81
GraphSAGE+MAML	79.54	81.21	77.49	87.41	91.34	85.91	76.18	80.17	75.12
TGAT+MAML	83.29	85.04	83.04	87.46	91.07	87.09	73.86	78.03	72.67
MetaDyGNN	95.21	96.02	94.32	96.03	97.89	95.98	83.15	88.76	82.15
<i>Improvement</i>	9.25	6.17	8.55	2.67	2.56	2.91	8.95	9.40	7.72

Model/Result/AUC(%)	2-shot			4-shot			6-shot		
	Wikipedia	Reddit	DBLP	Wikipedia	Reddit	DBLP	Wikipedia	Reddit	DBLP
TGAT	90.20	95.20	81.01	90.21	95.23	80.98	90.23	95.34	81.03
Meta-GNN	75.81	82.96	74.03	77.89	87.69	76.63	79.03	89.92	78.84
GraphSAGE+MAML	76.21	83.34	74.17	78.42	88.31	77.17	79.21	90.34	79.17
TGAT+MAML	80.14	88.67	73.04	81.24	89.67	76.24	83.07	92.67	78.12
MetaDyGNN	90.31	95.76	82.14	92.75	96.87	83.83	94.82	97.21	86.24
<i>Improvement</i>	0.12	0.59	1.39	2.81	1.72	3.51	5.08	1.96	6.42

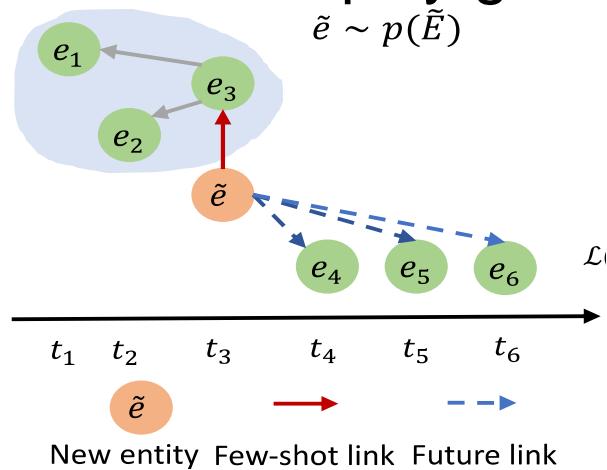
Few-Shot Learning for New Nodes: MetaTKGR

- **Objective:** Further address challenge 3
 - Challenge 3: learn temporally robust parameters against distribution evolution
- **Motivation:** utilize prior estimation as temporal regularizers against distribution shift

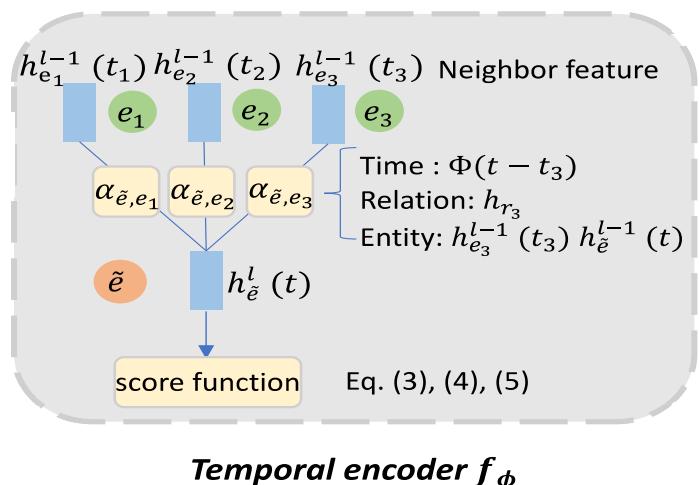


Few-Shot Learning for New Nodes: MetaTKGR

- Method: Simplify general knowledge as sample & aggregation capability



- Temporal sampler:
 - Sample temporal neighbors;
 - Time bounded breadth first search;
- Temporal aggregation:
 - Aggregate information from sampled temporal neighbors;
 - Achieve by trainable attention weights;

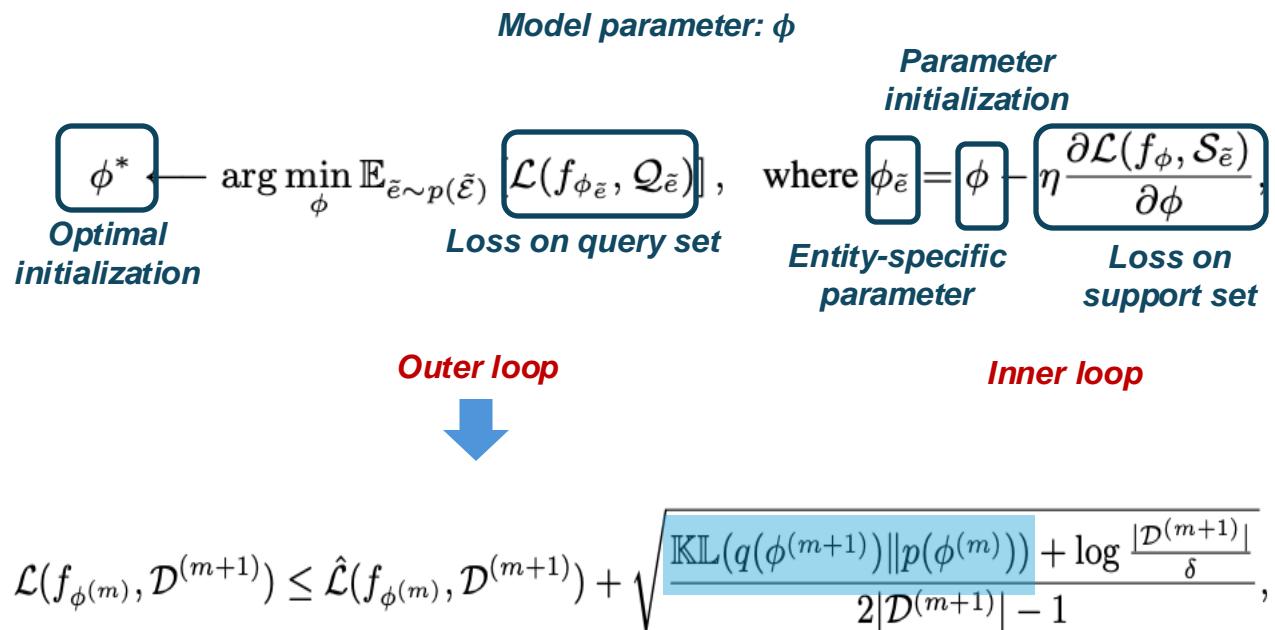


$$\mathbf{h}_{\tilde{e}}^l(t) = \sigma \left(\sum_{(e_i, r_i, t_i) \in \mathcal{N}_{\tilde{e}}(t)} \alpha_{\tilde{e}, e_i} \left(\mathbf{h}_{e_i}^{l-1}(t_i) \mathbf{W} \right) \right)$$

$$\alpha_{\tilde{e}, e_i} = \frac{\exp(q_{\tilde{e}, e_i})}{\sum_{(e_k, r_k, t_k) \in \mathcal{N}_{\tilde{e}}(t)} \exp(q_{\tilde{e}, e_k})} \quad q_{\tilde{e}, e_i} = \mathbf{a} \left(\mathbf{h}_{\tilde{e}}^{l-1} \| \mathbf{h}_{e_i}^{l-1} \| \mathbf{h}_{r_i} \| \Phi(t - t_i) \right)$$

Few-Shot Learning for New Nodes: MetaTKGR

- **Method:** Propose bi-level temporal meta-learning framework
 - Inner loop: fine-tune on support set from initialization;
 - Outer loop: train on query set for **robust** global initialization



Few-Shot Learning for New Nodes: MetaTKGR

□ Setting:

- View entities appearing in last ~25% time interval as new entities.

□ Baselines:

- **Static KGE:** TransE, TransR, RotatE;
- **Temporal KGE:** RE-NET;
- **Few-shot KGE:** LAN, I-GEN, T-GEN;
- **Few-shot temporal graph:** MetaDyGNN;

□ Observation:

- MetaTKGR generally outperforms all baselines for 1,2,3-shot experiments.

Table 2: The results of 3-shot temporal knowledge graph reasoning. Average results on 5 independent runs are reported. * indicates the statistically significant improvements over the best baseline, with p -value smaller than 0.001. We report standard deviation in Appendix A.5

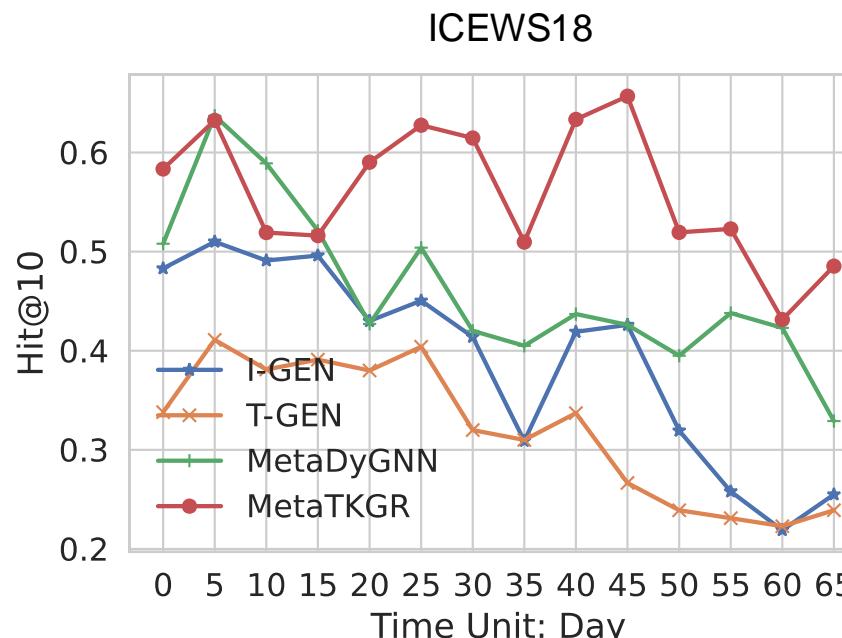
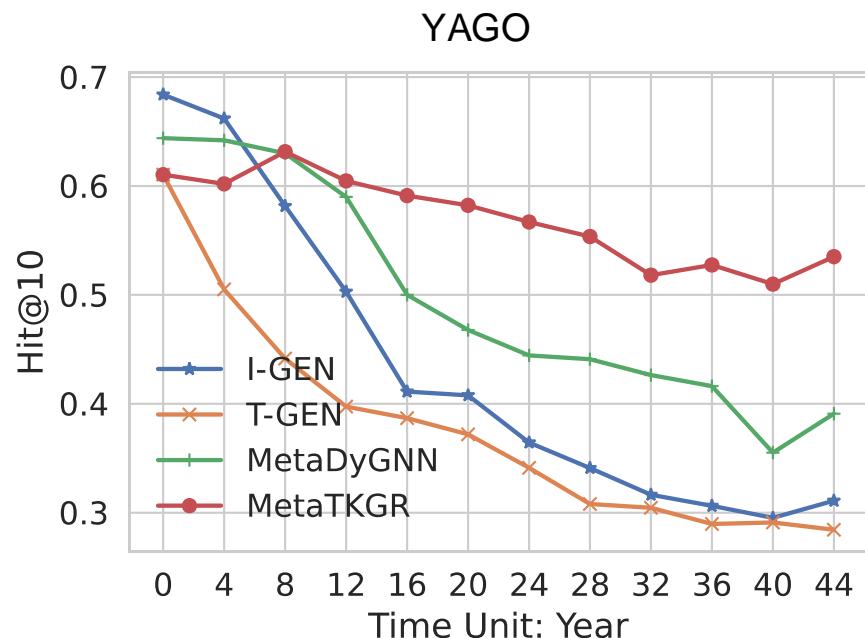
Models	YAGO				WIKI				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.223	0.158	0.242	0.360	0.161	0.111	0.171	0.236	0.058	0.052	0.062	0.098
TransR	0.234	0.165	0.259	0.382	0.183	0.138	0.188	0.245	0.061	0.062	0.073	0.109
RotatE	0.241	0.182	0.278	0.409	0.232	0.171	0.223	0.284	0.078	0.074	0.082	0.128
RE-NET	0.261	0.210	0.298	0.410	0.261	0.210	0.251	0.331	0.232	0.139	0.241	0.369
LAN	0.230	0.154	0.247	0.352	0.185	0.133	0.201	0.287	0.207	0.119	0.234	0.321
I-GEN	0.303	0.238	0.323	0.420	0.221	0.179	0.229	0.264	0.212	0.120	0.251	0.346
T-GEN	0.292	0.218	0.310	0.394	0.234	0.185	0.222	0.271	0.169	0.122	0.184	0.265
MetaDyGNN	0.350	0.270	0.379	0.511	0.309	0.238	0.309	0.459	0.307	0.216	0.309	0.469
MetaTKGR	0.370*	0.303*	0.416*	0.558*	0.329*	0.253*	0.335*	0.489*	0.335*	0.249*	0.340*	0.527*
Gains (%)	5.68	12.21	9.80	9.19	6.26	6.38	8.47	6.35	9.11	14.85	9.89	12.4

Table 3: The results of 1-shot and 2-shot experiment. We report complete results in Appendix A.5.

Models	YAGO				WIKI				ICEWS18			
	1-shot		2-shot		1-shot		2-shot		1-shot		2-shot	
	MRR	H@10										
TransE	0.183	0.268	0.193	0.304	0.144	0.186	0.146	0.213	0.049	0.077	0.058	0.086
TransR	0.189	0.270	0.198	0.312	0.160	0.183	0.160	0.225	0.050	0.080	0.060	0.090
RotatE	0.215	0.280	0.210	0.359	0.175	0.190	0.201	0.268	0.068	0.098	0.070	0.091
RE-NET	0.221	0.304	0.233	0.390	0.212	0.259	0.239	0.294	0.185	0.250	0.200	0.341
LAN	0.196	0.269	0.200	0.310	0.174	0.275	0.162	0.273	0.170	0.301	0.188	0.317
I-GEN	0.238	0.321	0.237	0.402	0.181	0.241	0.223	0.287	0.199	0.320	0.177	0.337
T-GEN	0.247	0.331	0.260	0.379	0.202	0.245	0.240	0.319	0.131	0.262	0.161	0.259
MetaDyGNN	0.269	0.396	0.316	0.496	0.241	0.371	0.271	0.390	0.249	0.420	0.269	0.441
MetaTKGR	0.294*	0.428*	0.356*	0.526*	0.277*	0.419*	0.309*	0.441*	0.295*	0.496*	0.301*	0.500*
Gains (%)	9.43	8.04	12.69	6.14	14.64	12.93	14.04	13.20	18.45	17.87	11.47	13.39

Few-Shot Learning for New Nodes: MetaTKGR

- MetaTKGR produces relatively robust performance over time;
- Performance curve of ICEWS18 fluctuates a lot, due to much shorter time unit (day).

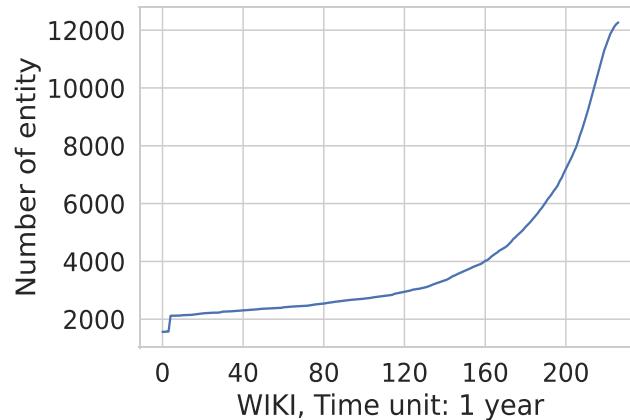


Limitations of MetaTKGR

- ❑ **Challenge 2:** design meta-learning algorithms that are compatible with TGNN
- ❑ **Insufficient consideration of new node patterns:**
 - ❑  **Few-shot initial links**
 - ❑  **Topology information: e.g., geometry pattern, ...**
 - ❑  **Total amount: exponential growth**

Distribution of New Nodes on Temporal Graphs

❑ Exponential growth:



Number of nodes over time.

❑ Latent space should have large capacity to learn discriminative representations:

❑ Euclidean space: area grows polynomially w.r.t. radius.

❑ Harder model adaptation:

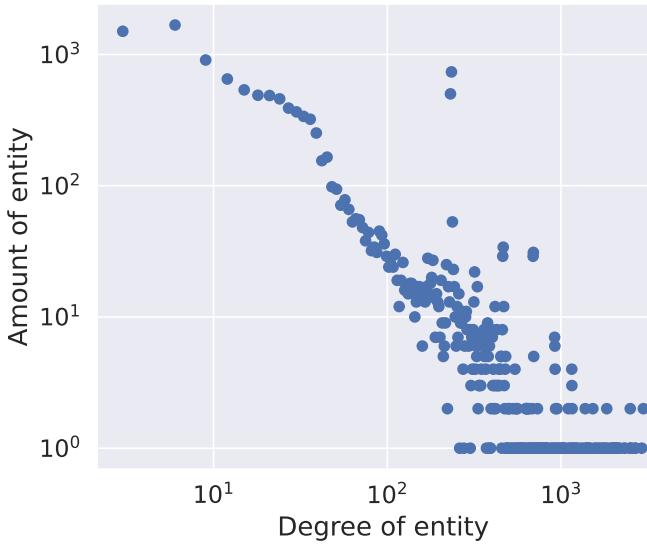
❑ Change to higher-dimensional space;

❑ Enlarge radius of area.

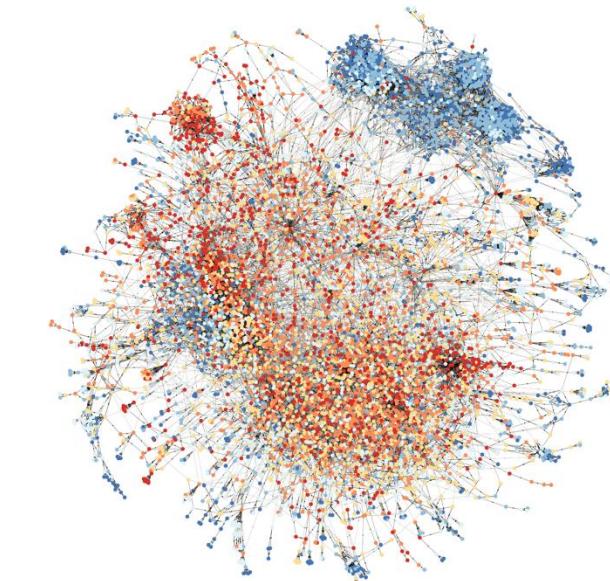
Distribution of New Nodes on Temporal Graphs

- Power-law distribution and hierarchical structures:

Distribution and graph hierarchy visualization on Wiki graphs.



Power-law distribution



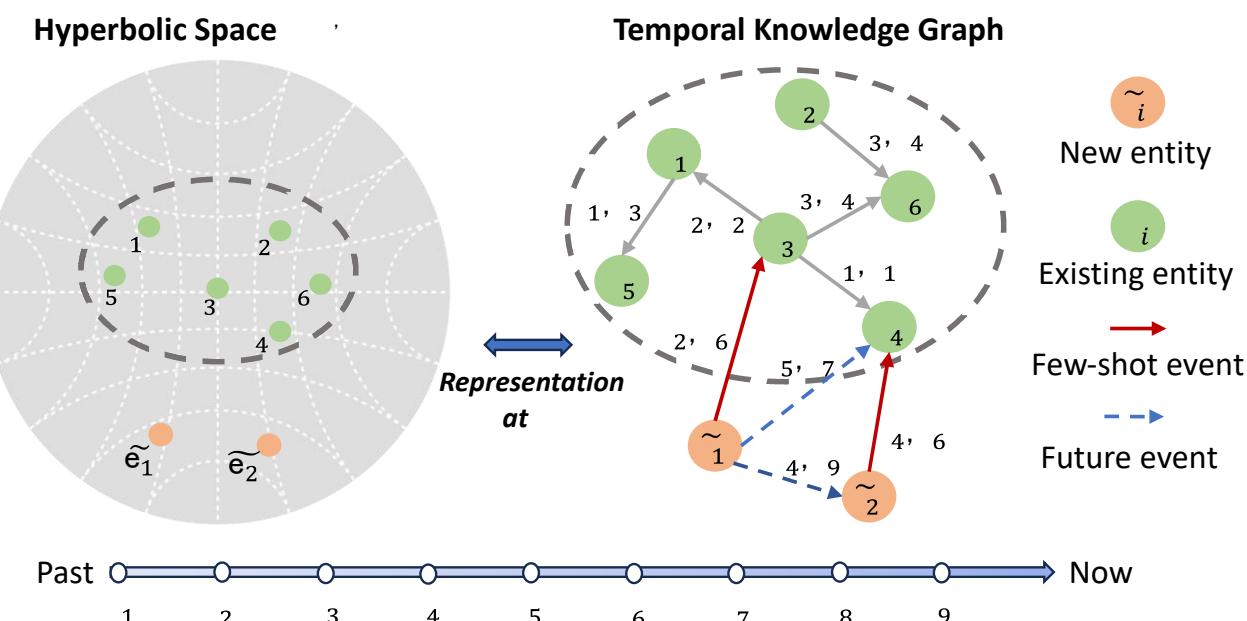
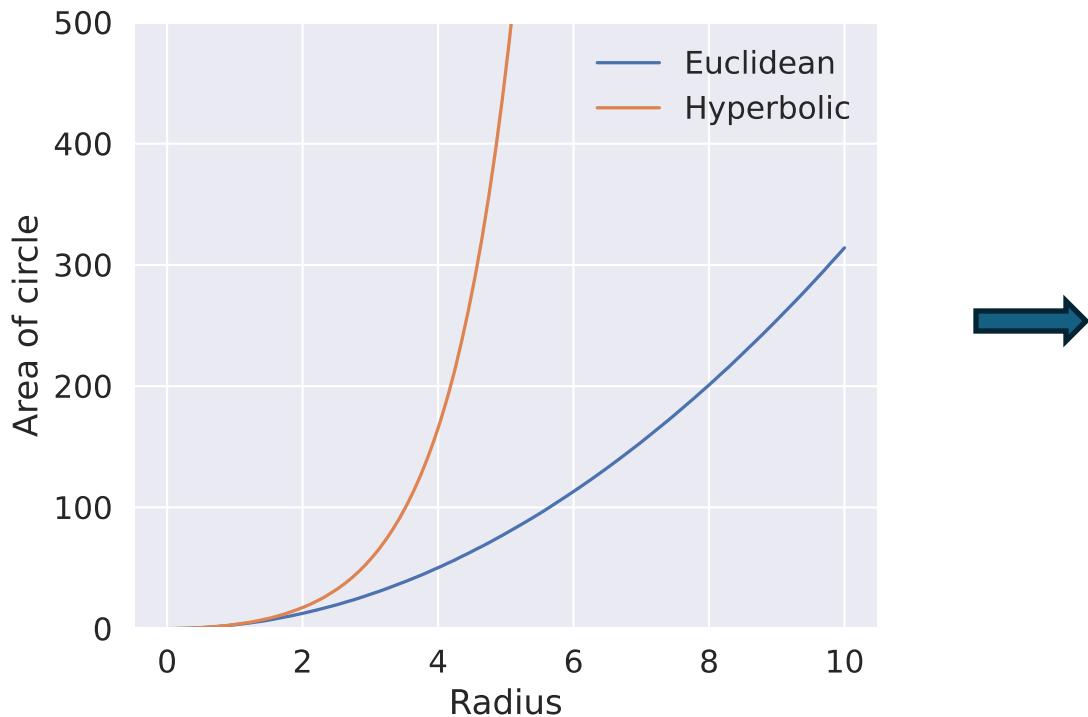
*Graph hierarchy
existing (red) v.s. new (blue)*

- Latent geometry should be good at capturing power-law distribution and hierarchical structures :

- Euclidean space: capture grid-structured data

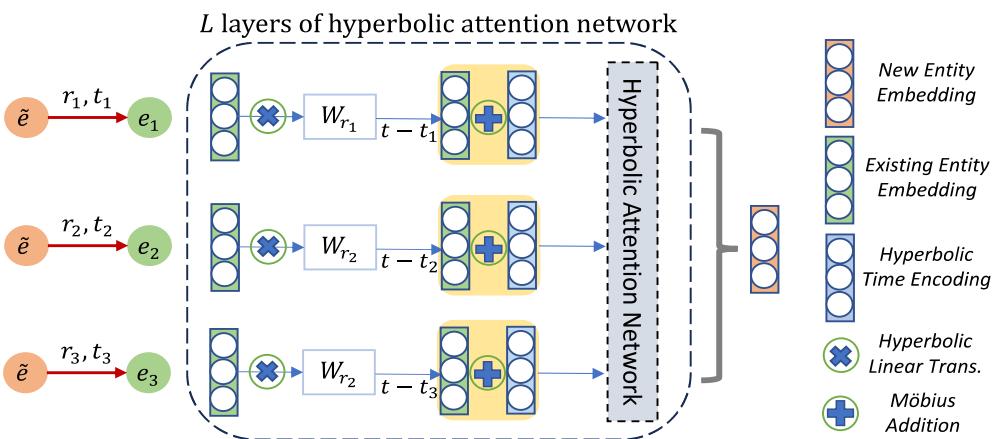
Few-Shot Learning for New Nodes: MetaHKG

- Motivation: represent temporal graphs on a hyperbolic space that yields easier model adaptation for new nodes.



Few-Shot Learning for New Nodes: MetaHKG

- Method: temporal graph representation on hyperbolic space
 - Hyperbolic attention layers



- Hyperbolic time encoding

$$\phi_{\mathcal{B}}(t) = \frac{\tanh(\sqrt{cd}/2)}{\sqrt{c/2}d} (\cos(\omega_1 t), \sin(\omega_1 t), \dots, \cos(\omega_{d/2} t), \sin(\omega_{d/2} t))$$

$$\begin{aligned} \mathcal{K}_{\mathcal{B}}(t_i + \Delta t, t_j + \Delta t) &= \langle \phi_{\mathcal{B}}(t_i + \Delta t), \phi_{\mathcal{B}}(t_j + \Delta t) \rangle_{\mathcal{B}} \\ &= \mathcal{K}_{\mathcal{B}}(t_i, t_j) \end{aligned}$$

$$\begin{aligned} \mathbf{h}_{e_i, r_i, t_i}^{\mathcal{B}, l}(t) &= \mathbf{W}_{r_i}^l \otimes^c \mathbf{h}_{e_i}^{\mathcal{B}, l-1}(t_i) \oplus^c \phi_{\mathcal{B}}(t - t_i) \\ \mathbf{h}_{\tilde{e}}^{\mathcal{B}, l}(t) &= \exp_0^c \left(\sigma \left(\sum_{(e_i, r_i, t_i) \in \mathcal{N}_{\tilde{e}}(t)} \alpha_{\tilde{e}, e_i} \log_0^c \left(\mathbf{h}_{e_i, r_i, t_i}^{\mathcal{B}, l}(t) \right) \right) \right) \\ \alpha_{\tilde{e}, e_i} &= \frac{\exp(q_{\tilde{e}, e_i})}{\sum_{(e_k, r_k, t_k) \in \mathcal{N}_{\tilde{e}}(t)} \exp(q_{\tilde{e}, e_k})} \\ q_{\tilde{e}, e_i} &= \frac{1}{\sqrt{d}} \left(\mathbf{W}_k^l \otimes^c \mathbf{h}_{e_i, r_i, t_i}^{\mathcal{B}, l}(t) \right)^T \cdot \left(\mathbf{W}_q^l \otimes^c \mathbf{h}_{\tilde{e}}^{\mathcal{B}, l-1}(t) \right) \end{aligned}$$

THEOREM 3.1 (TRANSLATION INVARIANCE). Let the temporal kernel be $\mathcal{K}_{\mathcal{B}}(t_i, t_j) = \langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}}$, where $\phi_{\mathcal{B}}(t)$ is defined in Eq. (6). Then the temporal kernel is translation-invariant, i.e., $\mathcal{K}_{\mathcal{B}}(t_i + \Delta t, t_j + \Delta t) = \langle \phi_{\mathcal{B}}(t_i + \Delta t), \phi_{\mathcal{B}}(t_j + \Delta t) \rangle_{\mathcal{B}} = \mathcal{K}_{\mathcal{B}}(t_i, t_j)$, for any constant Δt .

PROOF. By Eq. (6), the following equation holds:

$$\langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}} = \frac{\tanh^2(\sqrt{cd}/2)}{cd^2/2} \langle \phi_{\mathcal{E}}(t_i), \phi_{\mathcal{E}}(t_j) \rangle,$$

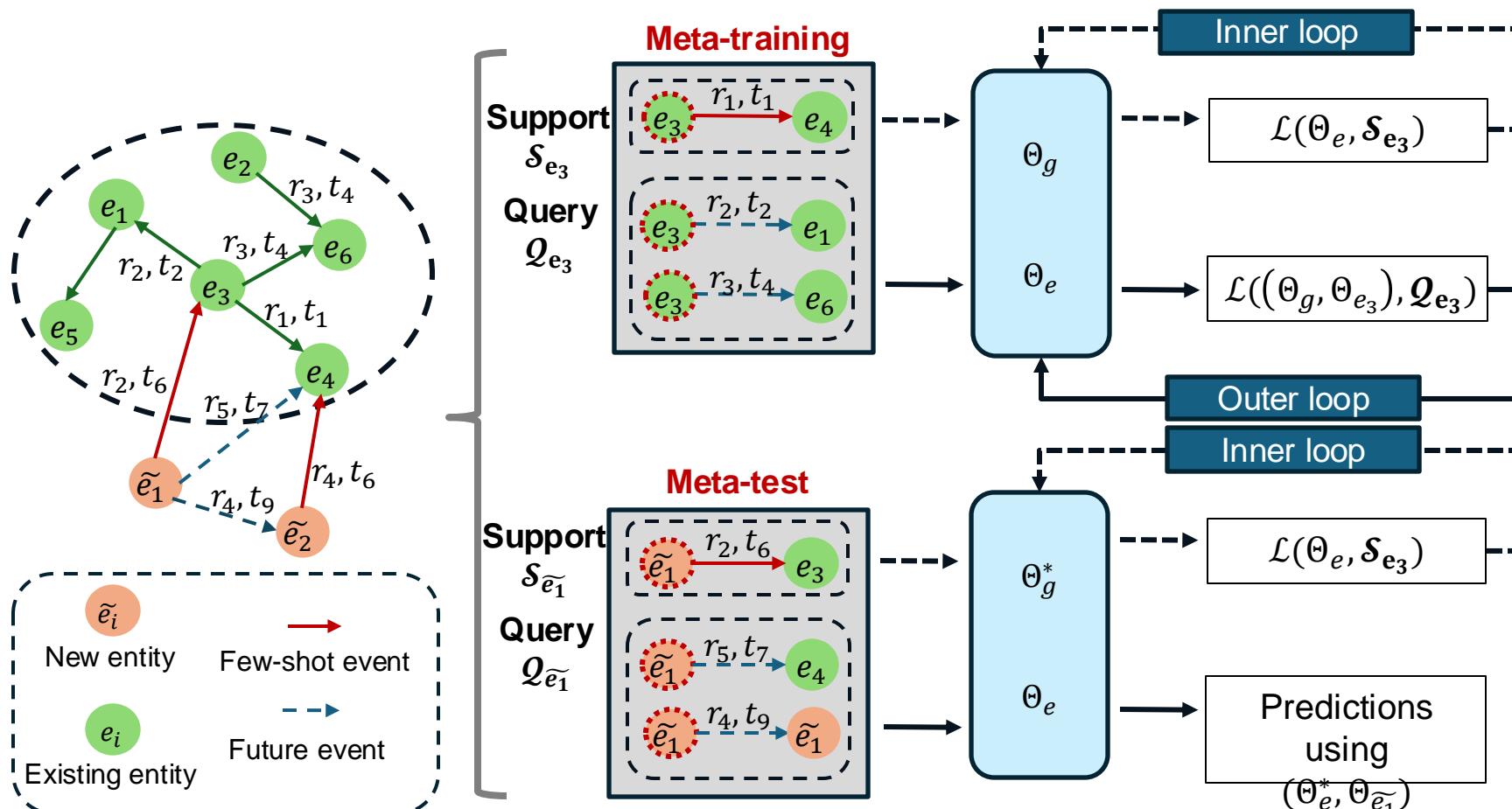
In the Euclidean space, according to the Bochner's theorem and [11], we have $\mathcal{K}(t_i, t_j) = \langle \phi_{\mathcal{E}}(t_i), \phi_{\mathcal{E}}(t_j) \rangle = \psi(t_i - t_j)$, where there exists a function ψ that is translation invariant. Therefore, we have the following for $\mathcal{K}_{\mathcal{B}}(t_i, t_j)$:

$$\mathcal{K}_{\mathcal{B}}(t_i, t_j) = \langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}} = \psi_{\mathcal{B}}(t_i - t_j),$$

$$\text{where } \psi_{\mathcal{B}} = f \circ \psi \text{ and } f(x) = \frac{\tanh^2(\sqrt{cd}/2)}{cd^2/2} x.$$

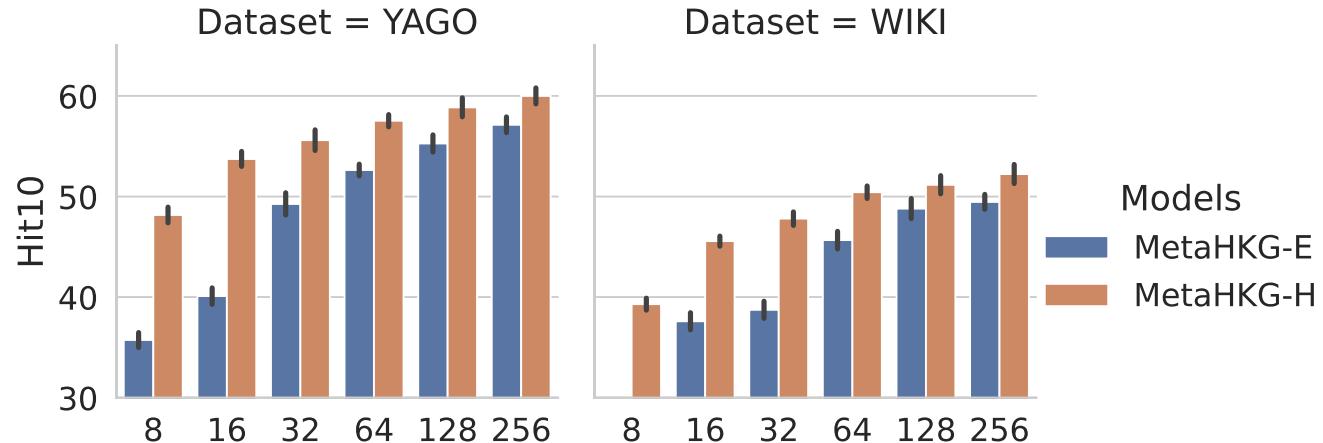
Few-Shot Learning for New Nodes: MetaHKG

- **Global parameters** Θ_g : relation embedding, time embedding, trainable curvature.
- **Entity-specific parameters** Θ_e : node embedding, attention parameters.



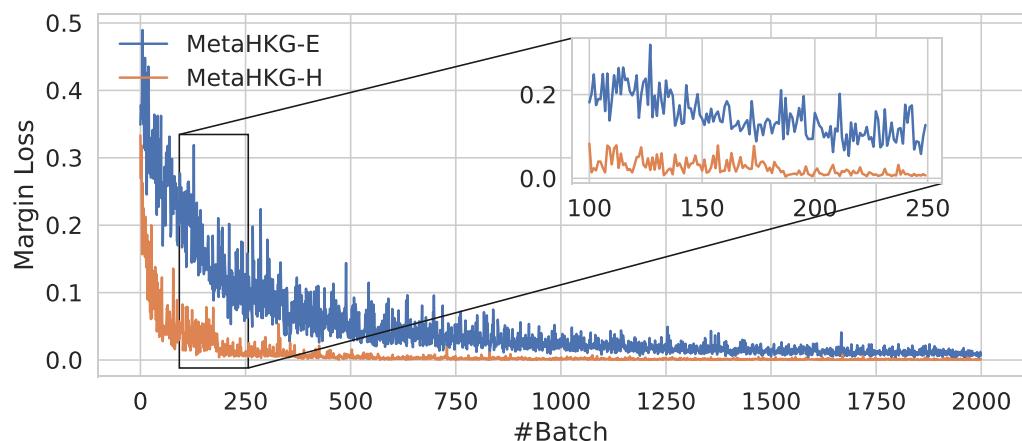
Few-Shot Learning for New Nodes: MetaHKG

- ❑ Lower-dimension of representation: 32-d (H) vs. 128/256-d (E)



*MetaHKG-E: Euclidean
MetaHKG-H: Hyperbolic*

- ❑ Stable training process because of easier model adaptation:



*MetaHKG-E: Euclidean
MetaHKG-H: Hyperbolic*

Few-Shot Learning for New Time Intervals

- ❑ New edges/nodes continually emerge over time
- ❑ The link prediction models need efficient updates to maintain satisfying performance over time.

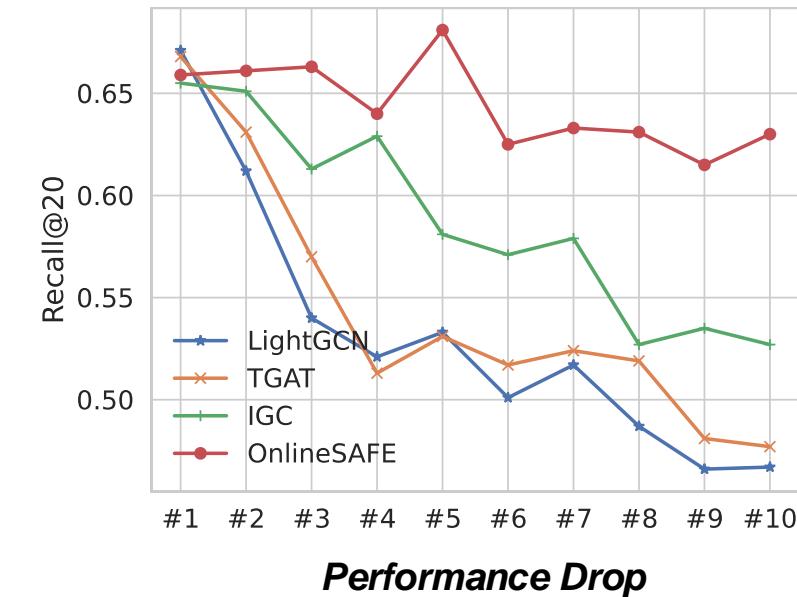
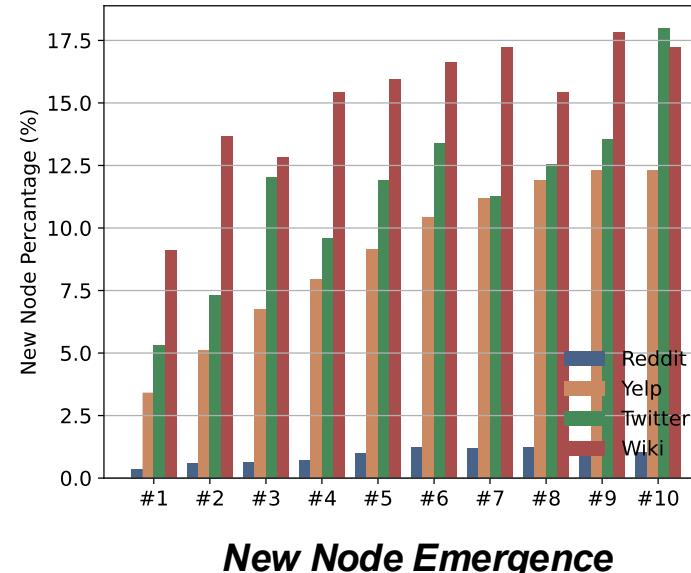
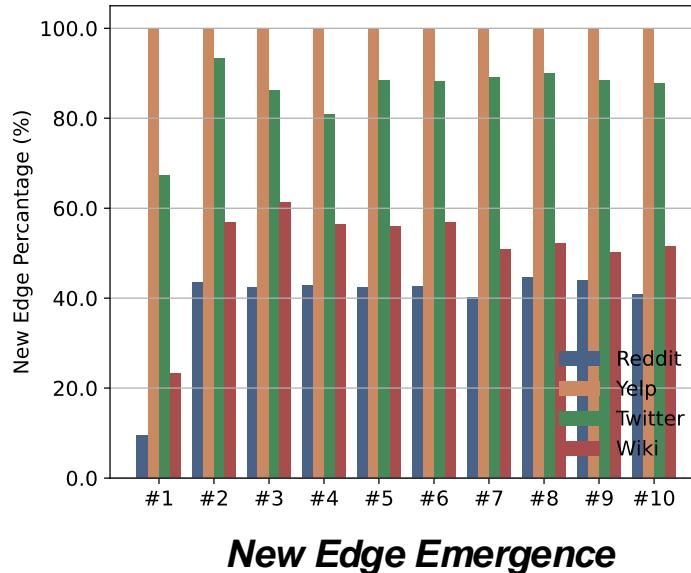
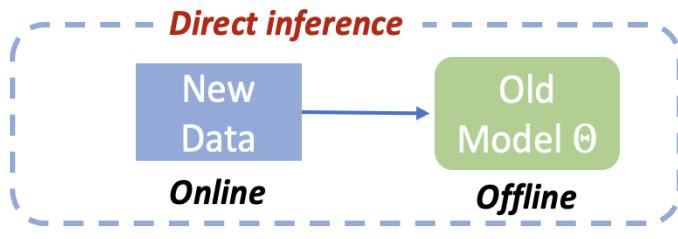


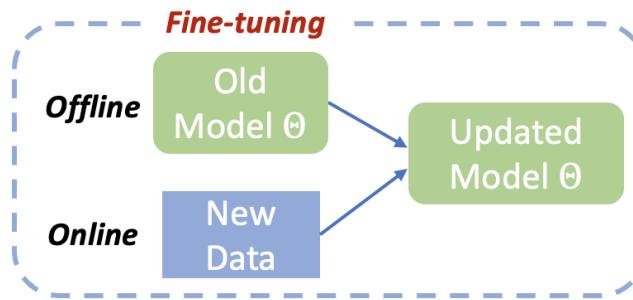
Figure shows percentage of new nodes/edges per time interval
(wiki/reddit: per ~ 1 day; Twitter: per ~2 weeks; Yelp: per ~4 months)

Few-Shot Learning for New Time: Existing Solutions

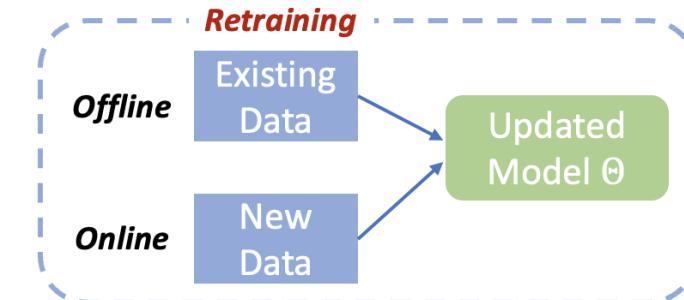
❑ Naïve strategies



Update time: 0s
Performance: 0.561



Update time: 40.5s
Performance: 0.730



Update time: 1395.8s
Performance: 0.744

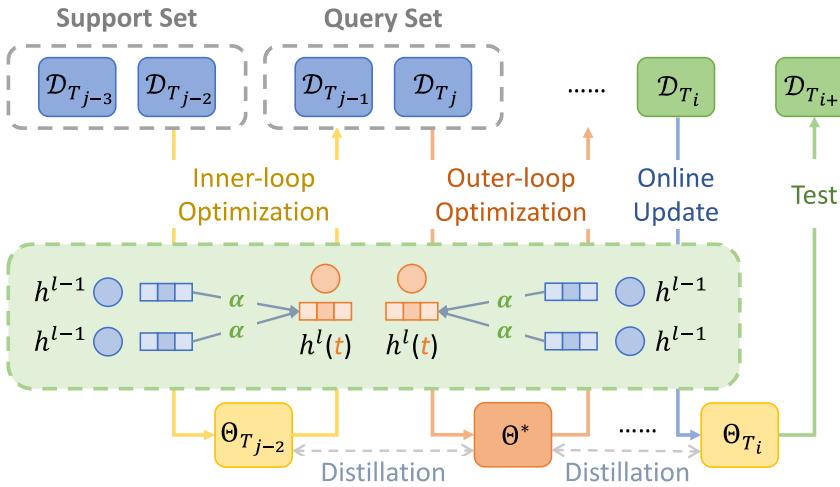
Result comparison on Reddit data (~11,000 nodes, ~0.67M temporal edges, daily update)

❑ Worse than existing static solutions

	Settings	Offline Setting				Online Setting	
		Models	First	Last	Dacay	Avg.	Strategy
Static model:	LightGCN	0.751	0.607	↓ 19.2%	0.661	Fine-tuning	0.741
	IGC	-	-	-	-	IGC	0.754
Temporal model:	TGAT	0.745	0.642	↓ 13.8%	0.681	Retraining	<u>0.744</u>
						<i>Better during offline</i>	<i>But worse during online</i>

Few-Shot Learning for New Time: **TGOnline**

- **Motivation:** knowledge distillation from global parameters:
- **Objective:** avoid overfitting exclusively on new data.
- How to enhance generalization of the global parameters: **meta optimization**



*Optimal
parameters*

$$\Theta^* \leftarrow \arg \min_{\Theta} \mathbb{E}_{T_i} [\mathcal{L}(\Theta^{T_i}, Q^{T_i})] \text{ (Outer Loop),}$$

$$\text{where } \Theta^{T_i} = \Theta^{T_i} - \eta \frac{\partial \mathcal{L}^{\text{Online}}(\Theta^{T_i}, S^{T_i})}{\partial \Theta^{T_i}} \text{ (Inner Loop),}$$

Meta task: adapt global knowledge on recently collected new data to update the model for future link prediction

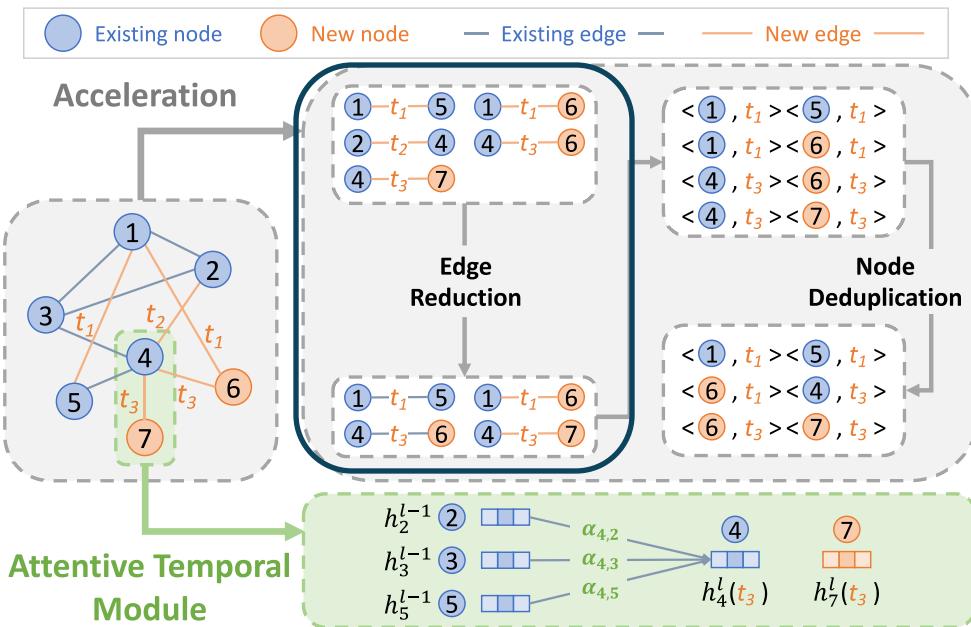
Support set S : edges from several recent time steps.

Query set Q : edges from several future time steps.

Inner loop: distillation-guided fine-tuning (support set).
Outer loop: extract global knowledge (query set).

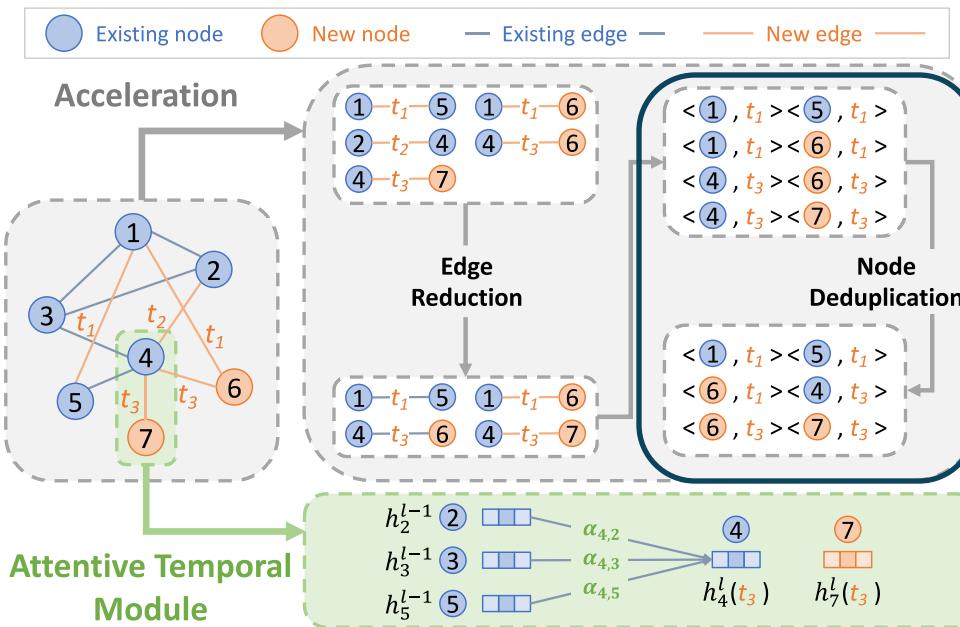
Few-Shot Learning for New Time: TGOnline

- Objective: Improve efficiency:
 - Edge reduction



New edge **(2, 4, t2)** is skipped as it connects two nodes within **2-hop neighborhood** in previous time intervals.

- Node deduplication



Node computation **(1, t1), (4, t3)** are removed because of duplication.

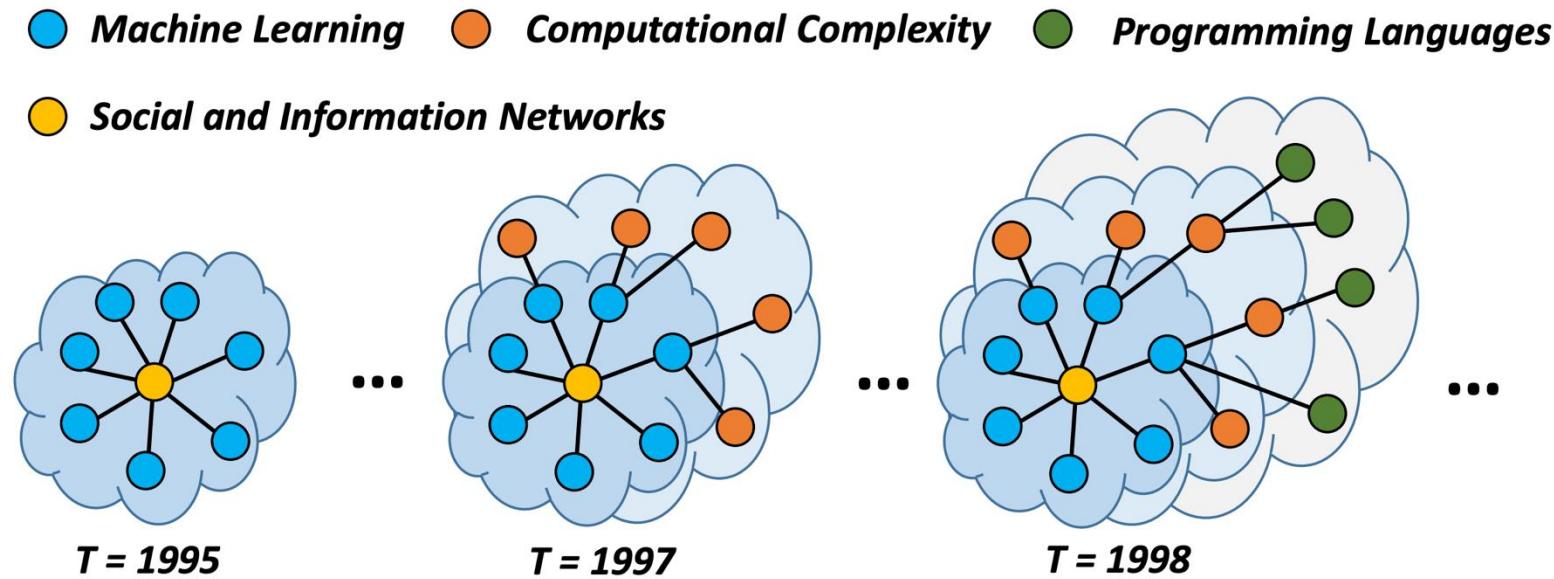
Few-Shot Learning for New Time: **TGOnline**

☐ Experiment: online setting

Dataset	Wiki		Reddit		Twitter		Yelp	
Performance	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF	0.662 ± 0.013	0.319 ± 0.005	0.750 ± 0.017	0.522 ± 0.004	0.172 ± 0.007	0.099 ± 0.002	0.183 ± 0.002	0.125 ± 0.002
GAE	0.481 ± 0.018	0.221 ± 0.012	0.708 ± 0.013	0.481 ± 0.017	0.600 ± 0.000	0.307 ± 0.001	0.213 ± 0.001	0.091 ± 0.000
GAT	0.540 ± 0.007	0.328 ± 0.012	0.706 ± 0.007	0.490 ± 0.007	0.340 ± 0.009	0.154 ± 0.023	0.091 ± 0.002	0.038 ± 0.001
GIN	0.409 ± 0.016	0.193 ± 0.006	0.568 ± 0.055	0.345 ± 0.050	0.519 ± 0.009	0.287 ± 0.008	0.220 ± 0.001	0.093 ± 0.001
LightGCN	0.698 ± 0.003	0.503 ± 0.003	0.741 ± 0.001	0.561 ± 0.003	0.568 ± 0.001	0.291 ± 0.001	0.224 ± 0.003	0.092 ± 0.001
GRU4Rec	0.080 ± 0.001	0.046 ± 0.001	0.048 ± 0.001	0.037 ± 0.000	0.056 ± 0.000	0.030 ± 0.000	0.026 ± 0.000	0.010 ± 0.000
JODIE	0.239 ± 0.015	0.198 ± 0.024	0.211 ± 0.011	0.183 ± 0.025	0.139 ± 0.015	0.098 ± 0.007	OOM	OOM
EGCN-H/O	0.089 ± 0.002	0.039 ± 0.001	0.471 ± 0.007	0.285 ± 0.004	0.250 ± 0.010	0.124 ± 0.007	OOM	OOM
VGRNN	0.048 ± 0.030	0.025 ± 0.016	0.389 ± 0.073	0.193 ± 0.036	0.389 ± 0.073	0.193 ± 0.036	0.165 ± 0.014	0.071 ± 0.006
Euler	0.040 ± 0.010	0.018 ± 0.005	0.484 ± 0.032	0.242 ± 0.017	0.600 ± 0.003	0.334 ± 0.002	0.070 ± 0.021	0.028 ± 0.010
DySAT	0.442 ± 0.010	0.224 ± 0.002	0.668 ± 0.002	0.426 ± 0.007	0.410 ± 0.011	0.176 ± 0.011	0.020 ± 0.000	0.007 ± 0.000
DIDA	0.601 ± 0.007	0.510 ± 0.009	0.617 ± 0.015	0.392 ± 0.025	0.551 ± 0.016	0.391 ± 0.012	0.242 ± 0.002	0.136 ± 0.002
TGAT	0.664 ± 0.010	0.529 ± 0.008	0.744 ± 0.011	0.618 ± 0.017	0.604 ± 0.010	0.231 ± 0.006	0.215 ± 0.012	0.121 ± 0.013
SPMF	0.585 ± 0.007	0.358 ± 0.006	0.741 ± 0.001	0.507 ± 0.002	0.022 ± 0.003	0.007 ± 0.001	0.166 ± 0.001	0.100 ± 0.001
SML	0.374 ± 0.023	0.190 ± 0.017	0.704 ± 0.013	0.455 ± 0.016	0.500 ± 0.071	0.250 ± 0.049	0.177 ± 0.010	0.111 ± 0.003
IGC	0.685 ± 0.014	0.526 ± 0.011	0.754 ± 0.011	0.589 ± 0.010	0.577 ± 0.002	0.335 ± 0.005	0.248 ± 0.008	0.132 ± 0.006
ROLAND	0.681 ± 0.015	0.536 ± 0.028	0.757 ± 0.009	0.592 ± 0.019	0.561 ± 0.015	0.324 ± 0.021	0.231 ± 0.012	0.129 ± 0.021
MetaDyGNN	0.667 ± 0.030	0.510 ± 0.018	0.752 ± 0.023	0.610 ± 0.031	0.581 ± 0.030	0.350 ± 0.019	0.251 ± 0.024	0.129 ± 0.011
TGOnline	0.716 ± 0.017	0.575 ± 0.005	0.807 ± 0.006	0.645 ± 0.010	0.644 ± 0.002	0.475 ± 0.002	0.272 ± 0.005	0.157 ± 0.002
Gains (%)	2.6	8.7	7.0	4.4	5.9	8.0	9.8	15.4

Few-Shot Learning for New Classes

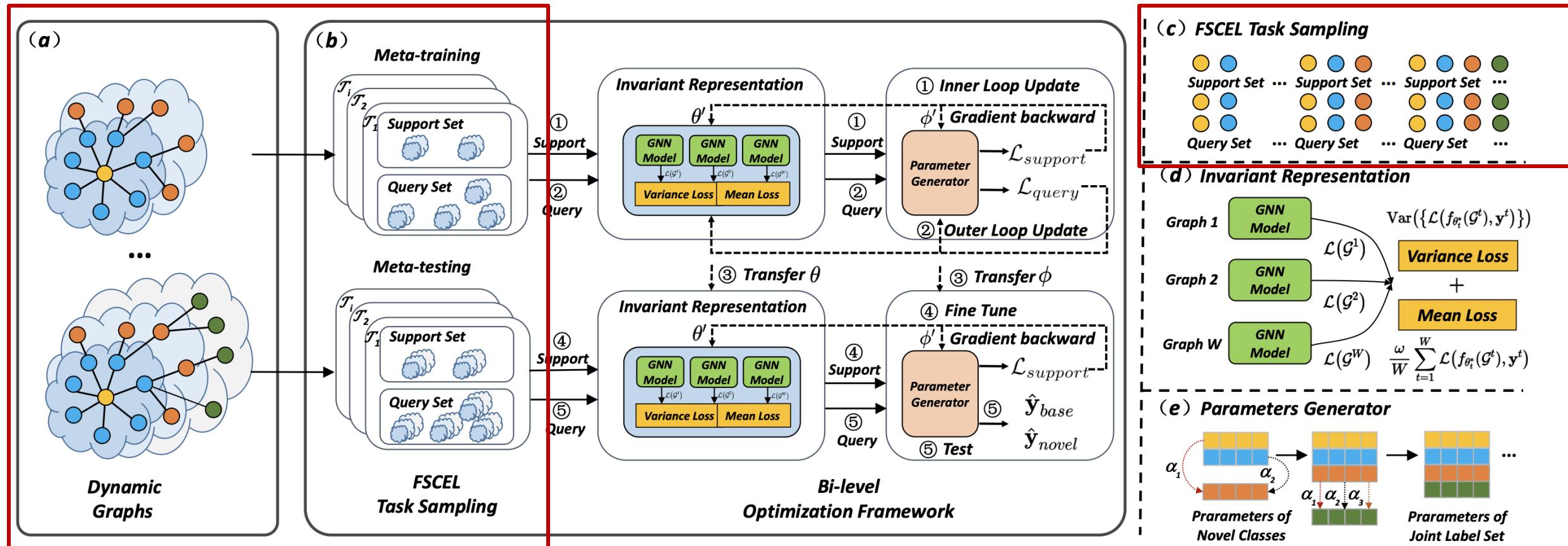
- ❑ Novel classes usually start with few samples, resulting in a label scarcity problem.



- ❑ Few-shot class evolutionary learning problem.

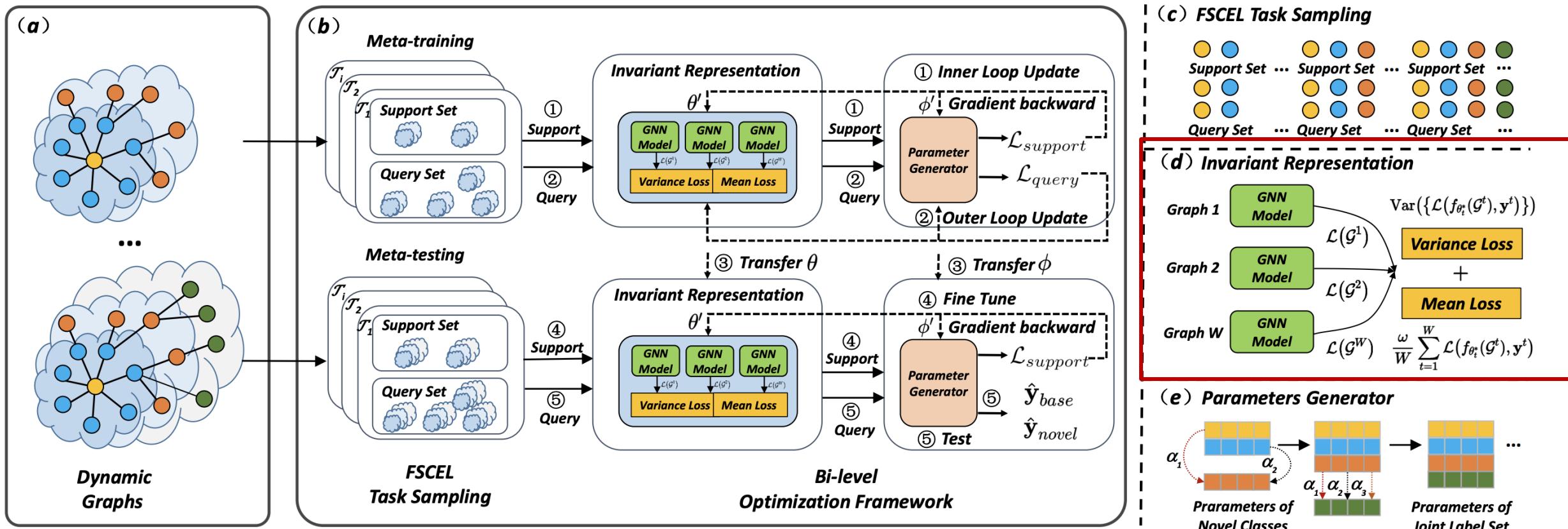
Few-Shot Learning for New Classes: LGP

- LGP is composed by:
 - a FSCEL task sampling strategy;



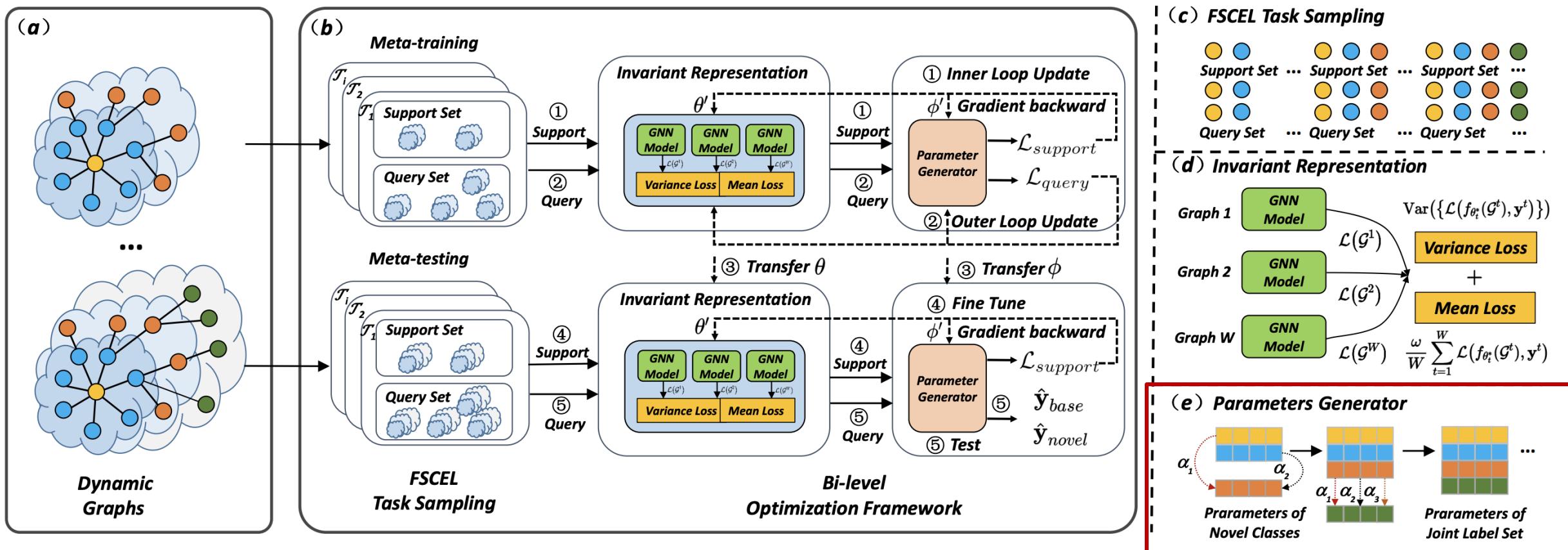
Few-Shot Learning for New Classes: LGP

- LGP is composed by:
 - an invariant representation model $f\theta$;



Few-Shot Learning for New Classes: LGP

- LGP is composed by:
 - a class evolution aware parameter generator $g_\phi \dots$

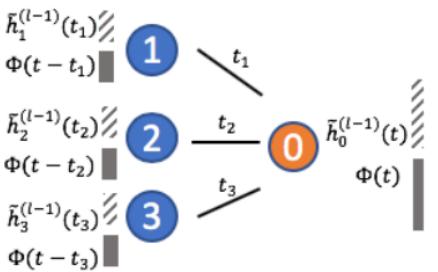


Few-Shot Learning for New Classes: LGP

☐ Experiments on few-shot class evolutionary task

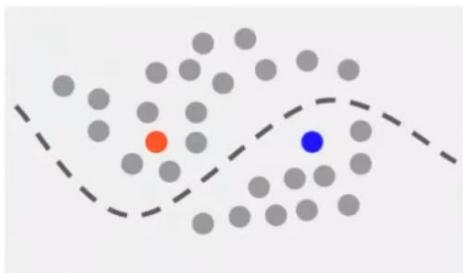
Methods	Test 1: 2006-2007		Test 2: 2007-2008		Test 3: 2008-2009		Test 4: 2009-2010		Test 5: 2010-2011	
Classes: $ C $	Base:20	Novel:4	Base:20	Novel:6	Base:20	Novel:8	Base:20	Novel:10	Base:20	Novel:14
Meta-GCN	25.1 ± 1.2	22.3 ± 2.1	15.9 ± 0.5	27.2 ± 1.0	20.7 ± 0.5	12.3 ± 2.0	8.5 ± 0.9	10.6 ± 0.4	5.2 ± 0.9	8.9 ± 0.5
Meta-SGC	31.4 ± 1.9	37.4 ± 1.6	13.4 ± 0.8	17.1 ± 0.7	12.2 ± 0.7	12.0 ± 1.3	13.5 ± 0.5	9.8 ± 0.6	9.7 ± 0.5	6.0 ± 0.5
G-Meta	60.6 ± 2.9	46.0 ± 1.5	35.8 ± 2.1	33.3 ± 1.3	26.0 ± 1.6	28.5 ± 4.5	22.7 ± 2.1	27.6 ± 0.8	11.2 ± 7.0	20.2 ± 1.9
PNet-SGC	25.7 ± 0.6	15.4 ± 0.5	27.9 ± 0.1	18.8 ± 0.3	21.1 ± 0.6	14.4 ± 0.2	23.4 ± 0.4	17.6 ± 0.4	21.0 ± 0.1	13.3 ± 0.1
PNet-GPR	30.9 ± 1.1	26.4 ± 8.1	29.0 ± 1.5	18.3 ± 3.1	27.6 ± 1.4	12.5 ± 1.1	21.1 ± 1.8	9.0 ± 1.7	22.4 ± 1.3	9.3 ± 2.8
LGP-SGC	51.2 ± 0.4	62.0 ± 2.1	47.2 ± 0.9	50.1 ± 3.1	44.2 ± 1.4	47.8 ± 4.8	36.8 ± 0.6	45.3 ± 2.1	32.8 ± 2.9	38.6 ± 1.6
LGP-GPR	45.9 ± 6.9	52.4 ± 3.2	43.6 ± 6.3	51.9 ± 3.9	42.2 ± 5.7	38.1 ± 4.9	35.2 ± 4.5	42.6 ± 2.6	34.9 ± 4.2	36.2 ± 2.2

Part II Summary: Data-Efficient TGNN



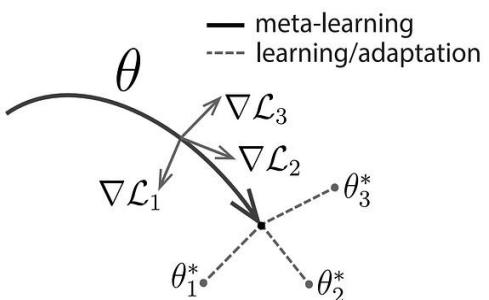
Self-Supervised Learning

- Introduction & Background
- Reconstruction
- Contrastive Learning
- Multiview Approach



Weakly-Supervised Learning

- Introduction & Background
- Limited Labeled Data
- Sparse Temporal Graph



Few-Shot Learning

- Introduction & Background
- New node adaptation
- New time adaptation
- New class adaptation

Q & A

Contents

- Part I – Introduction
- Part II – Data-Efficient Temporal Graph Neural Network
- **30-min Coffee Break**
- Part III – Resource-Efficient Temporal Graph Neural Network
- Part IV – Discussion and Future Directions

Towards Efficient Temporal Graph Learning: Algorithms, Frameworks, and Tools

Coffee Break 15:30 – 16:00

University of Illinois Urbana-Champaign

{ruijiew2, wanyu2, dsun18, charithm, zaher}@illinois.edu

Time: 1:45 PM – 17:30 PM, October 21, 2024

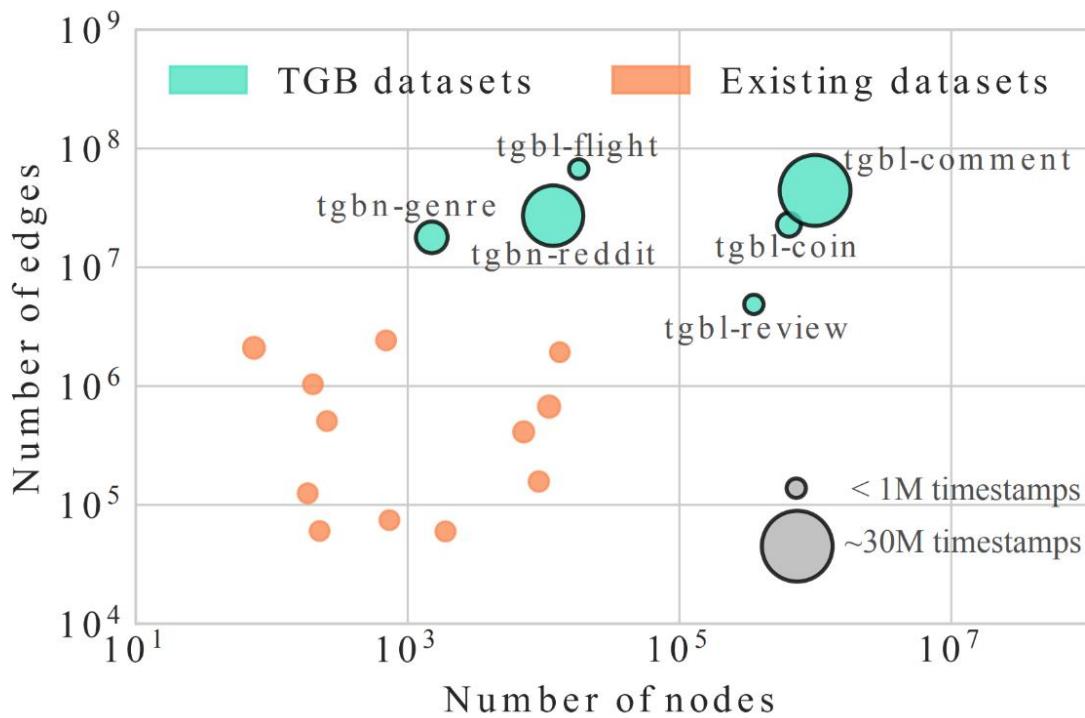
Location: Room 120C, Boise Centre, Boise, ID

Webpage: <https://wjerry5.github.io/cikm2024-tutorial/>

Contents

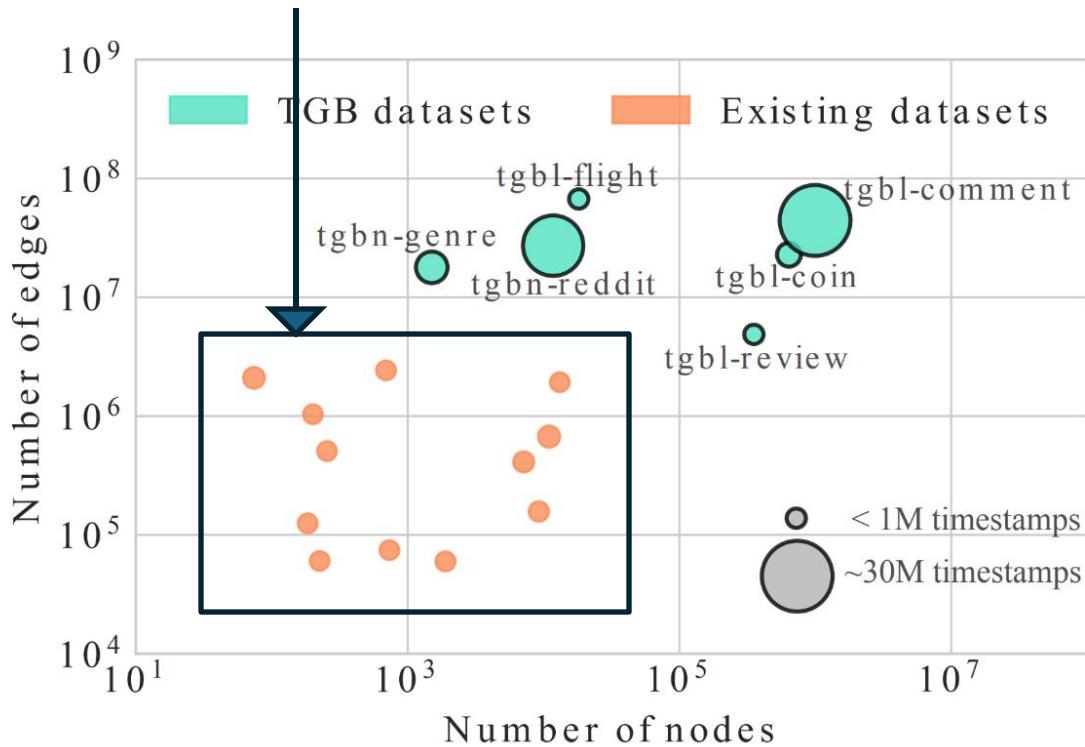
- Part I – Introduction
- Part II – Data-Efficient Temporal Graph Neural Network
- 30-min Coffee Break
- Part III – Resource-Efficient Temporal Graph Neural Network
- Part IV – Discussion and Future Directions

Why Resource Efficiency?



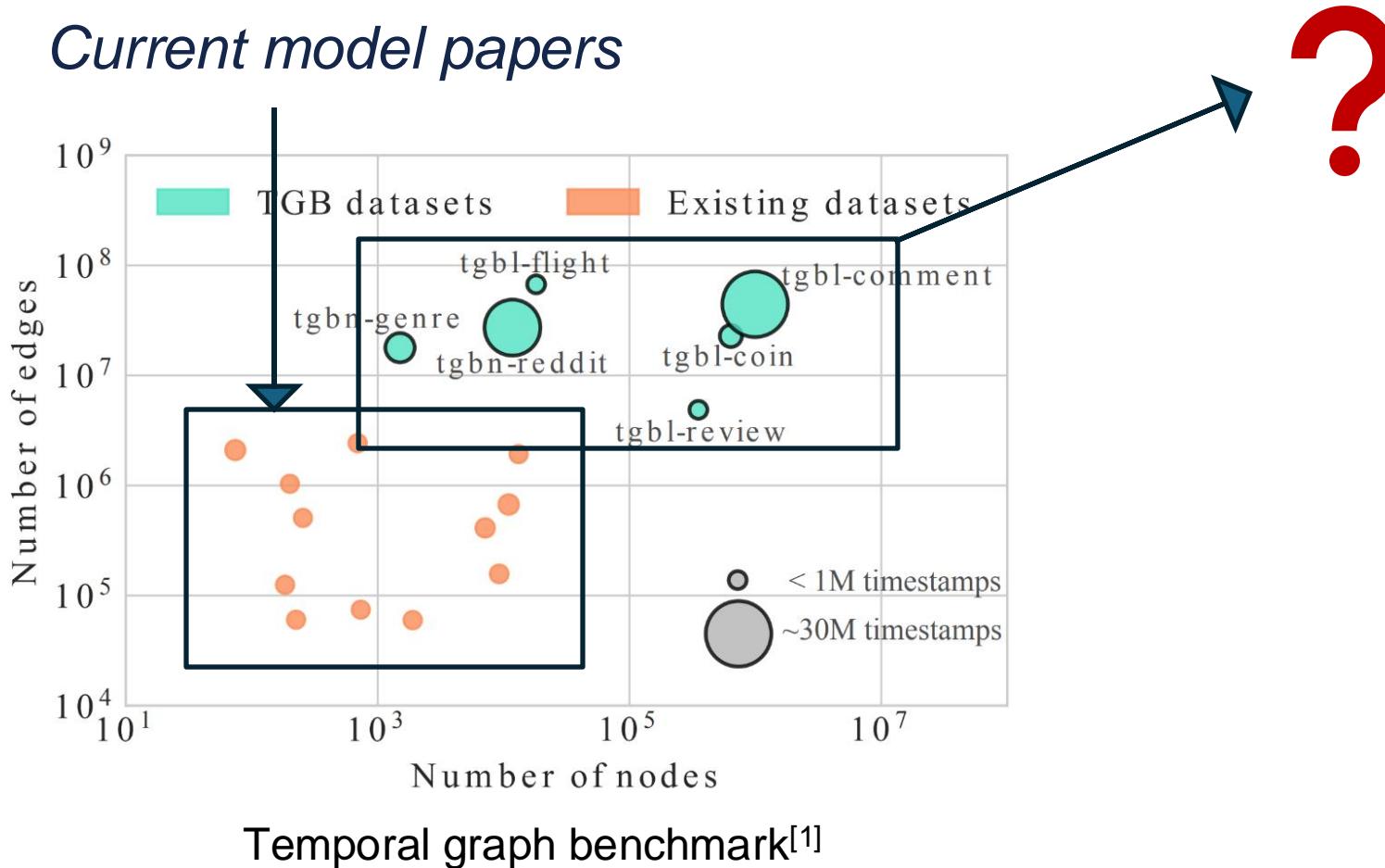
Why Resource Efficiency?

Current model papers



Why Resource Efficiency?

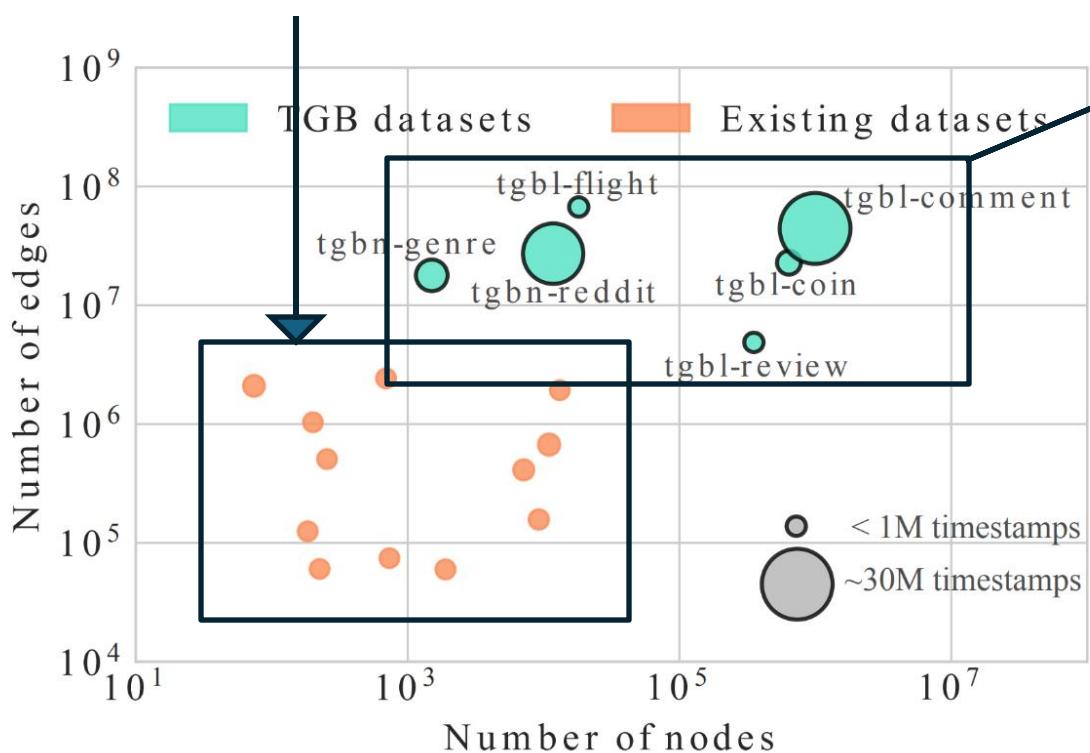
Current model papers



Temporal graph benchmark^[1]

Why Resource Efficiency?

Current model papers



Problems:

- Days to converge
- Out-of-Memory (OOM)
- Static GNN Frameworks?

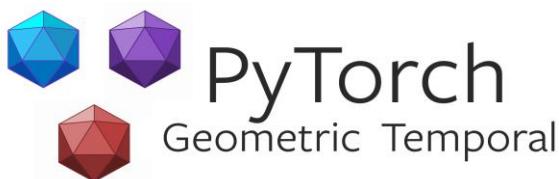


Why Resource Efficiency?

- Static GNN frameworks:
- No first-class support for temporal attributes



- DTDG Frameworks



- CTDG Frameworks

TGL: A General Framework for Temporal GNN Training on Billion-Scale Graphs

Hongkuan Zhou^{*}
University of Southern California
hongkuanz@usc.edu

Da Zheng
AWS AI
dzzhen@amazon.com

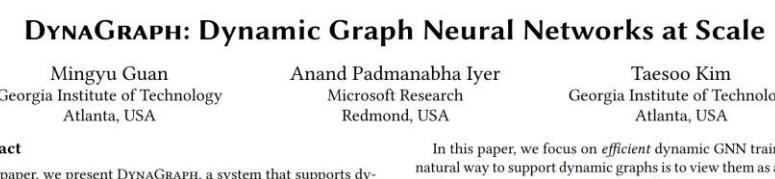
Irat Nisa
AWS AI
nisirat@amazon.com

Vasileios Ioannidis
AWS AI
ivasilei@amazon.com

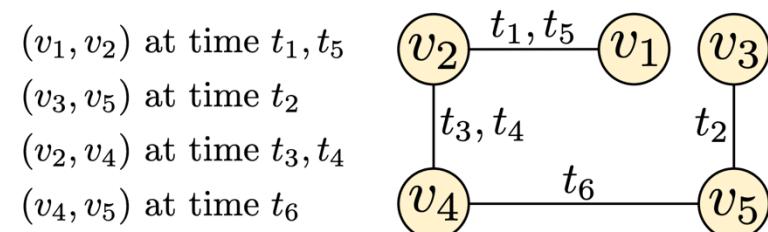
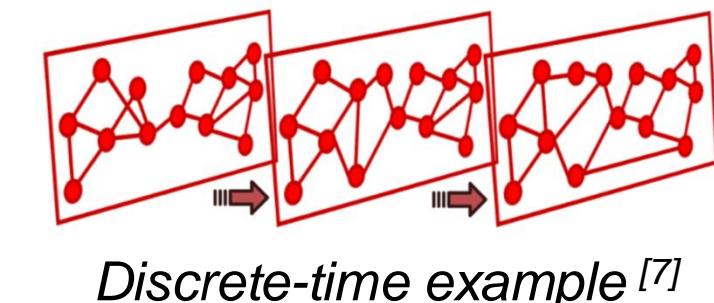
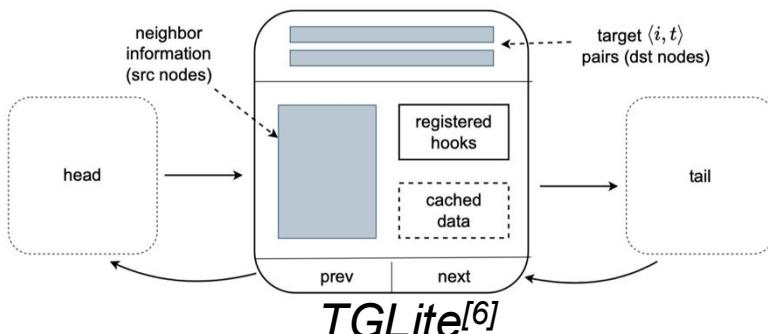
Xiang Song
AWS AI
xiangsx@amazon.com

George Karypis
AWS AI
gkarypis@amazon.com

TGL^[5]



DynaGraph^[4]



Continuous-time example^[7]

Why Resource Efficiency?

- Static GNN frameworks:
 - No first-class support for temporal attributes/operations



Today's Focus: CTDG Frameworks

- DTDG → CTDGs have richer time information with explicit timestamps, which can be crucial for certain applications.



**PyTorch
Geometric Temporal**

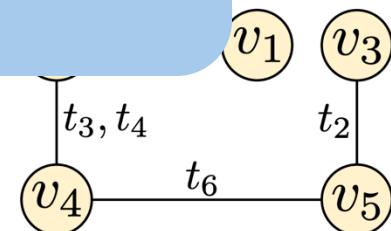
Mingyu Guan
Georgia Institute of Technology
Atlanta, USA

Anand Padmanabha Iyer
Microsoft Research
Redmond, USA

Taesoo Kim
Georgia Institute of Technology
Atlanta, USA

In this paper, we focus on *efficient* dynamic GNN training. A natural way to support dynamic graphs is to view them as a series of static snapshots. We call this representation a *tract*.

(v_3, v_5) at time t_2
 (v_2, v_4) at time t_3, t_4
 (v_4, v_5) at time t_6



CTDG Frameworks

TGL: A General Framework for Temporal GNN Training on Billion-Scale Graphs

Hongkuan Zhou*
University of Southern California
hongkuaz@usc.edu

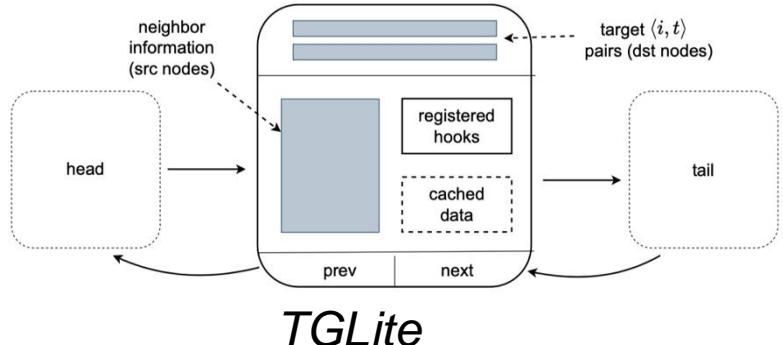
Da Zheng
AWS AI
dzzhen@amazon.com

Irat Nisa
AWS AI
nisirat@amazon.com

Vasileios Ioannidis
AWS AI
ivasilei@amazon.com

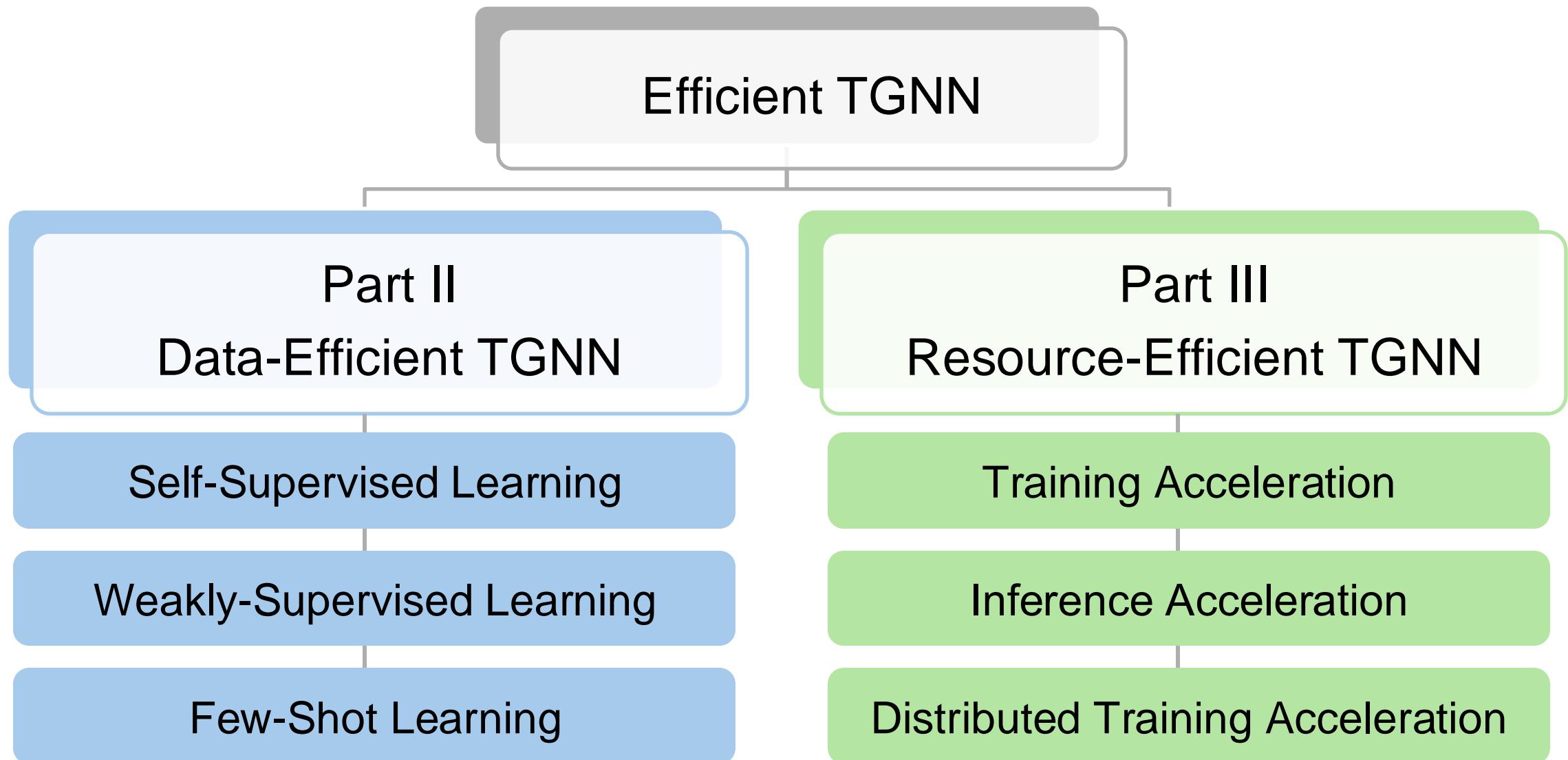
Xiang Song
AWS AI
xiangsx@amazon.com

George Karypis
AWS AI
gkarypis@amazon.com

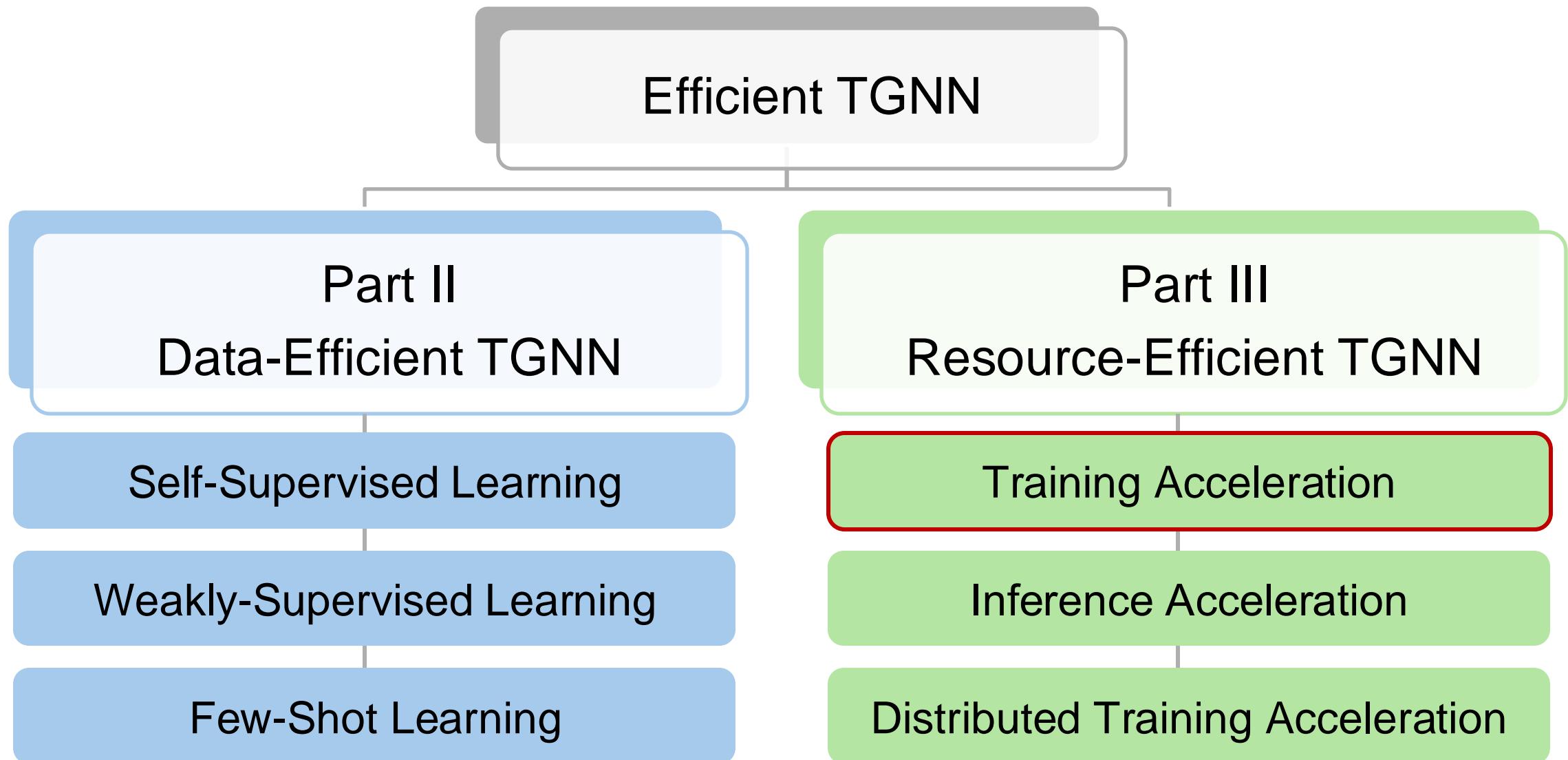


Continuous-time example

Scope of This Tutorial

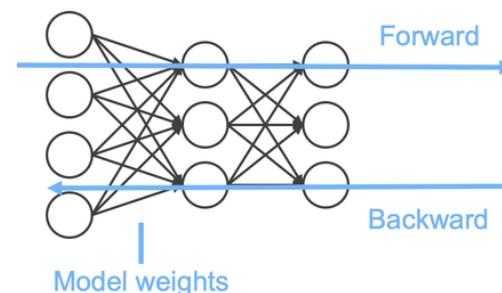
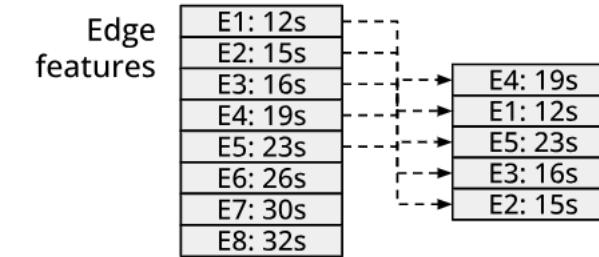
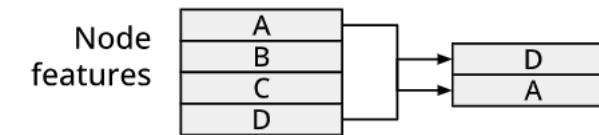
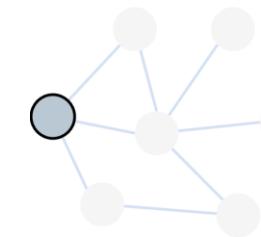


Scope of This Tutorial



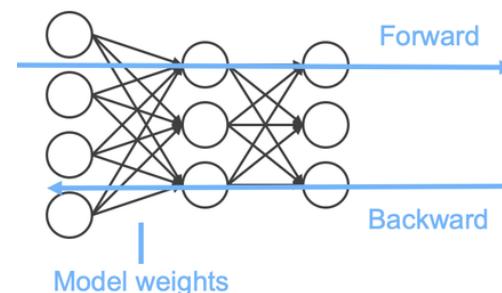
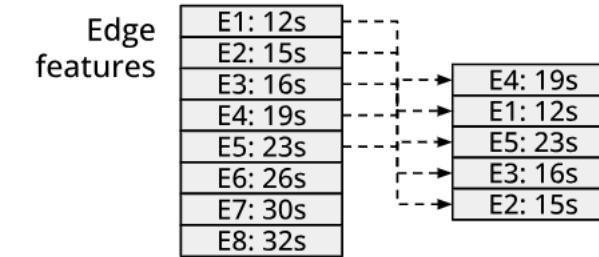
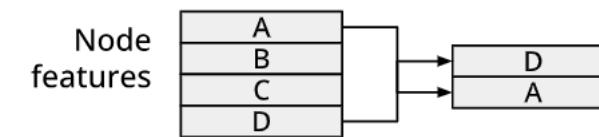
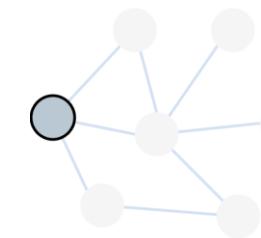
Training Acceleration

- **Introduction & Background**
- **Neighbor Sampling Acceleration**
- **Feature Fetching/Update Acceleration**
- **Model Computation Acceleration**



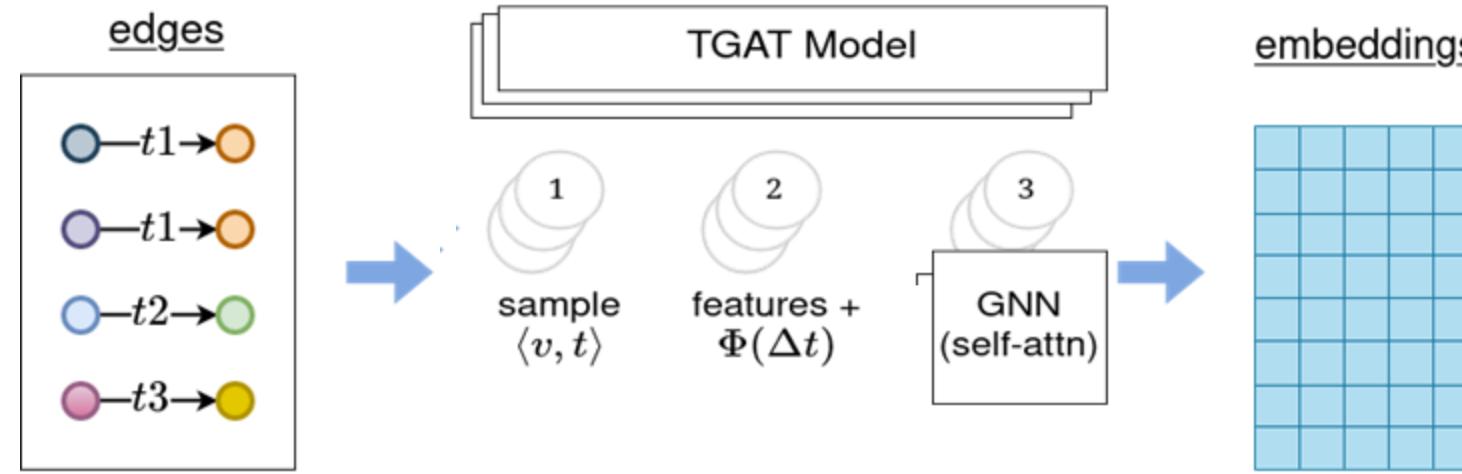
Training Acceleration

- **Introduction & Background**
- **Neighbor Sampling Acceleration**
- **Feature Fetching/Update Acceleration**
- **Model Computation Acceleration**



Background: TGNN Training Pipeline

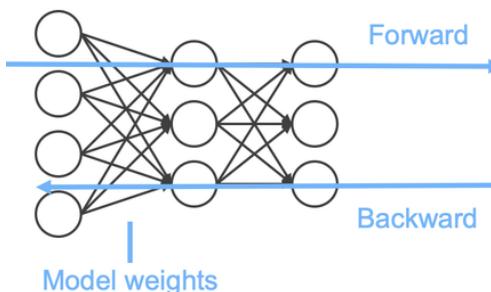
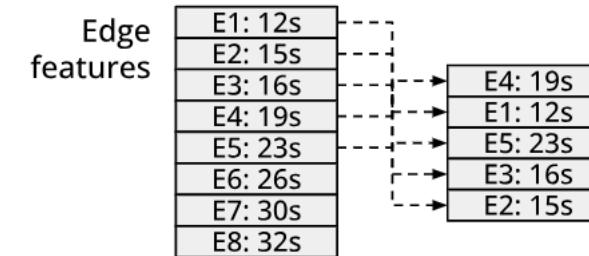
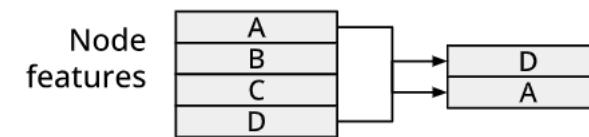
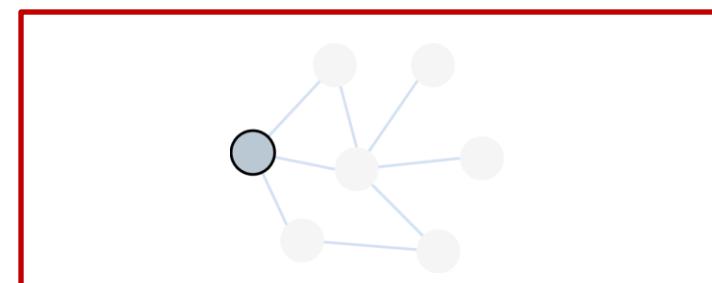
TGAT^[1] Example



- 1) For each node, sample **recent** temporal neighbors.
- 2) Encode neighbor edge **time deltas** into time features.
Aggregate node + time features using **self-attention**.
- 3) Repeat for a total of L layers (L hops).

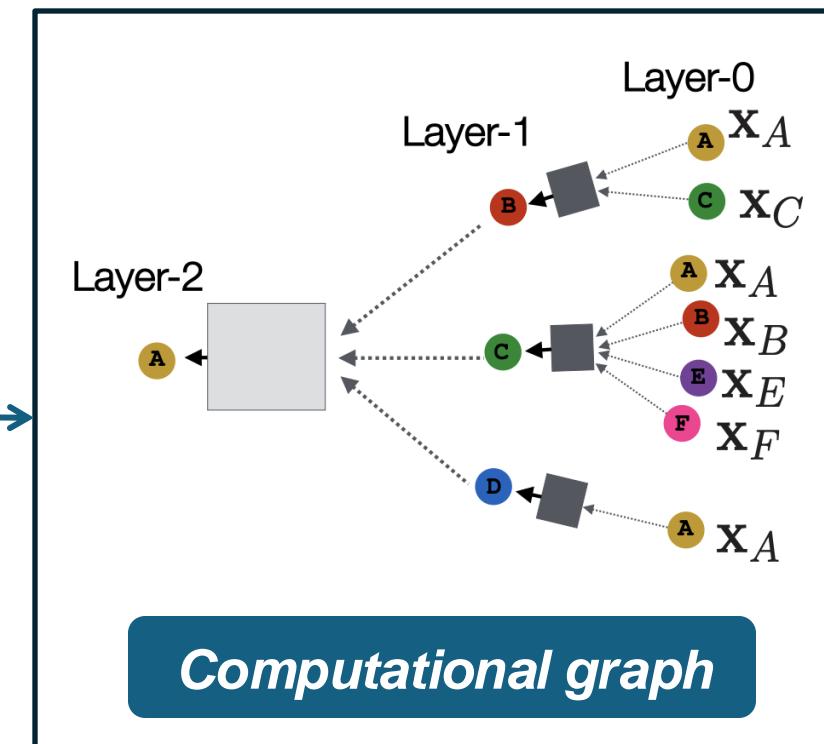
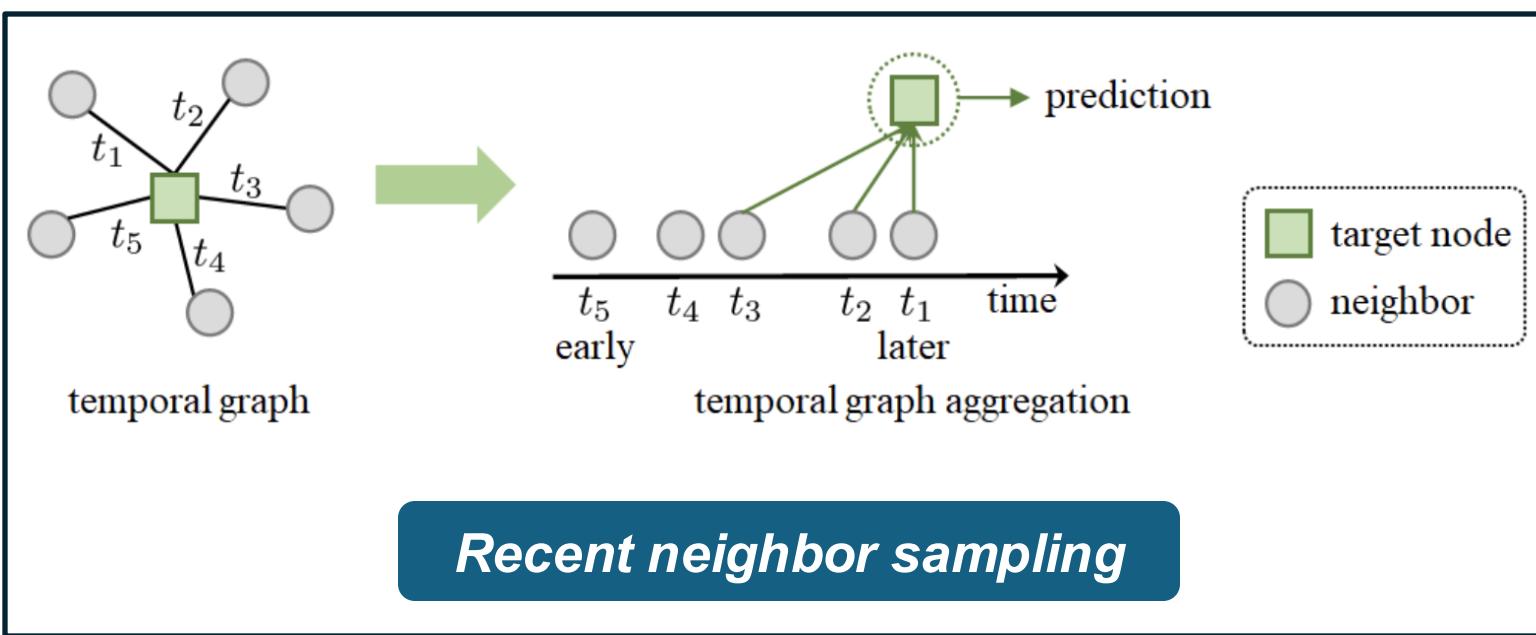
Training Acceleration

- Introduction & Background
- Neighbor Sampling Acceleration
- Feature Fetching/Update Acceleration
- Model Computation Acceleration



Background: Neighbor Sampling

- Input: Graph structure; Output: Computational graph
- Neighbor nodes connected by edges with **smaller** timestamps
 - No information leakage
- Most **recent** neighbor sampling
 - Superior performance & widely adopted



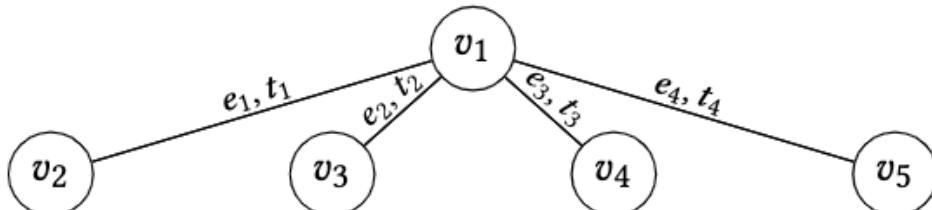
Parallel Temporal Sampler: TGL^[1]

□ Motivation

- Sampling neighbors on temporal graphs is **complex**.
- Timestamps of neighbors need to be considered.

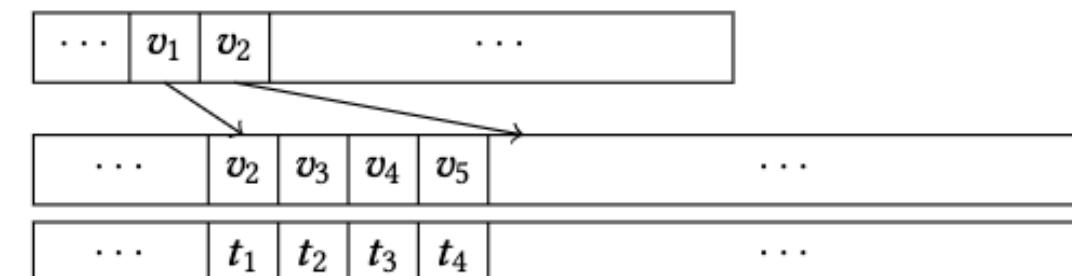
□ Method

- Temporal-CSR(T-CSR) data structure
- Parallel sampling



Graph structure

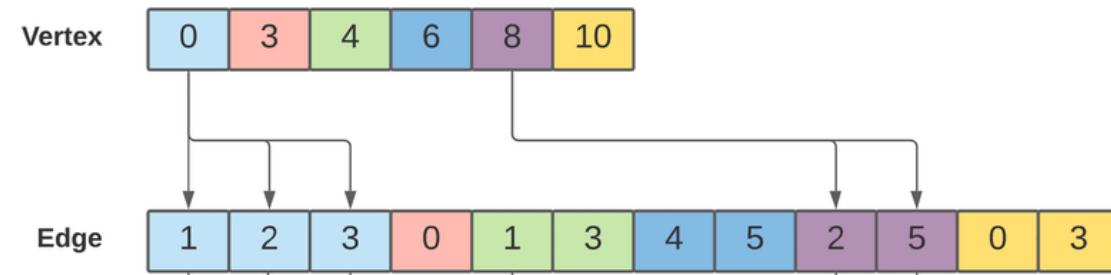
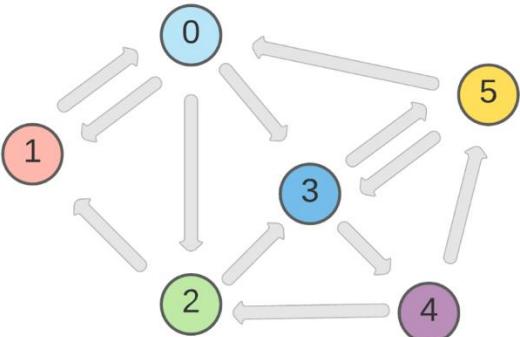
indptr
indices
times



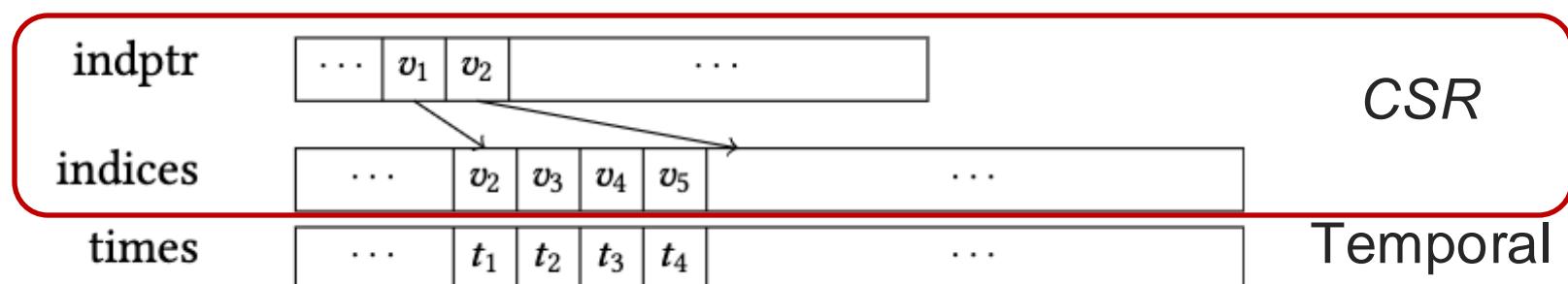
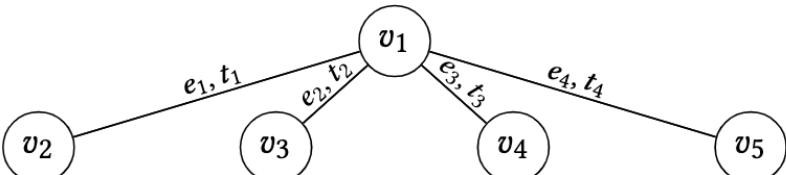
T-CSR

TGL Parallel Temporal Sampler: T-CSR

- CSR: compressed sparse row

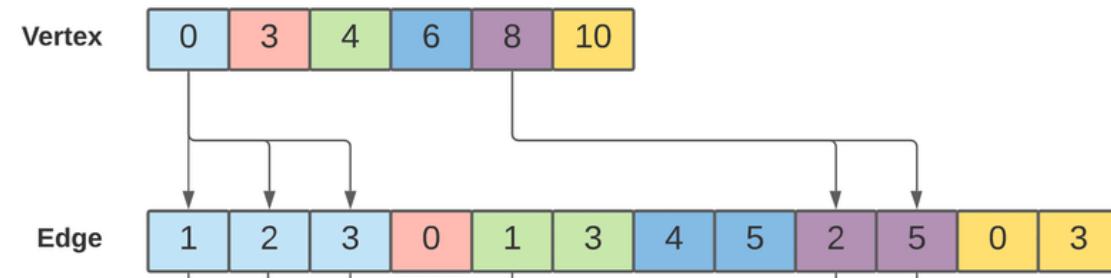
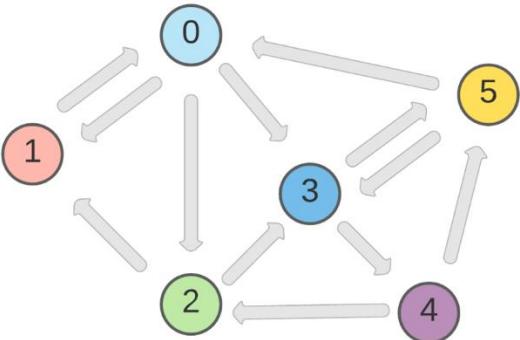


- Temporal-CSR: For each node, T-CSR sorts the outgoing edges according to their timestamps.

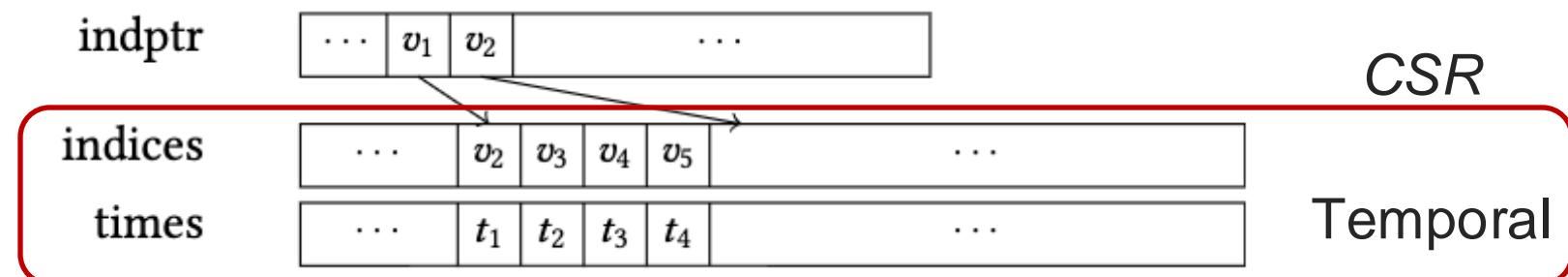
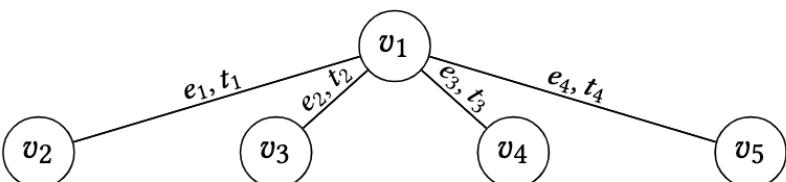


TGL Parallel Temporal Sampler: T-CSR

- CSR: compressed sparse row



- Temporal-CSR: For each node, T-CSR **sorts** the outgoing edges according to their timestamps.



$$t_1 \leq t_2 \leq t_3 \leq t_4$$

TGL Parallel Temporal Sampler: Algorithm

Data: sorted T-CSR G

Input: root nodes n with timestamp t_n , number of layer L , number of neighbors in each layer k_l

Output: DGL MFGs

for l in $0..L$ do

 if $l \geq 0$ then

 | set n and t_n to sampled neighbors in $l - 1$;

 end

 binary search for each node

 foreach $n \in n$ in parallel do

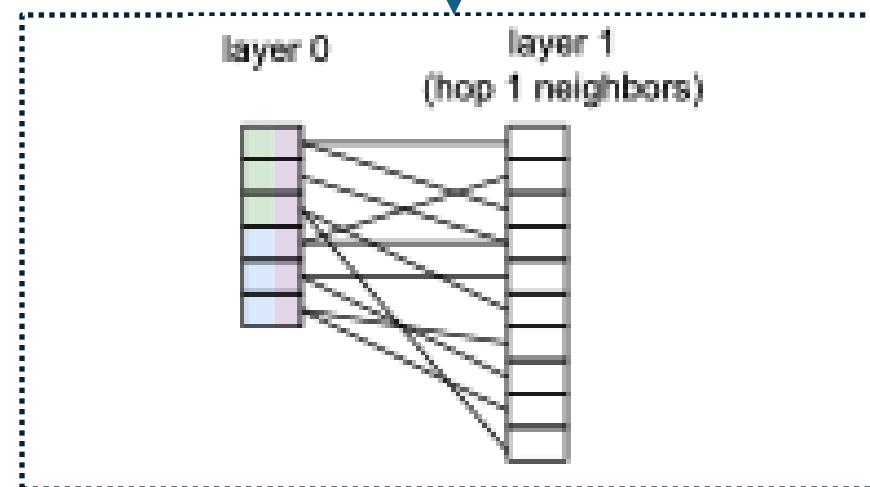
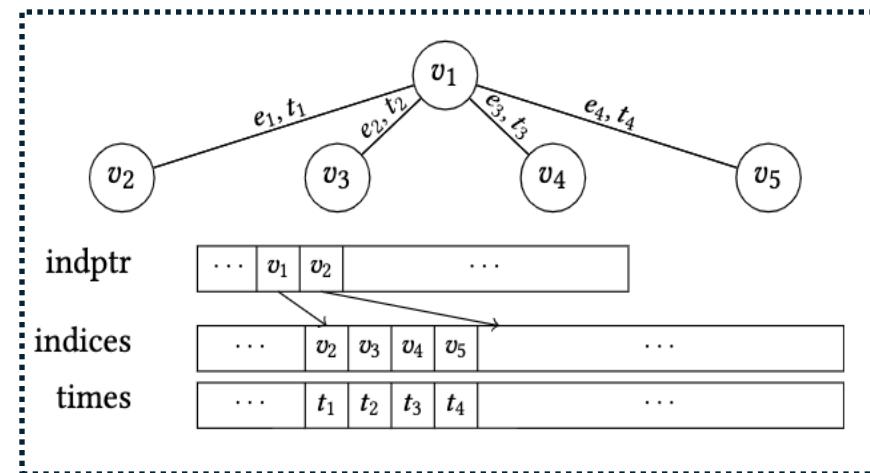
 | sample k_l neighbors within the snapshot S_s ;

 end

 generate DGL MFGs;

end

end



TMFG

TGL Parallel Temporal Sampler: Algorithm

Data: sorted T-CSR G

Input: root nodes n with timestamp t_n , number of layer L ,
number of neighbors in each layer k_l

Output: DGL MFGs

for l in $0..L$ **do**

if $l \geq 0$ **then**

 | set n and t_n to sampled neighbors in $l - 1$;

end

 binary search for each node

foreach $n \in n$ **in parallel do**

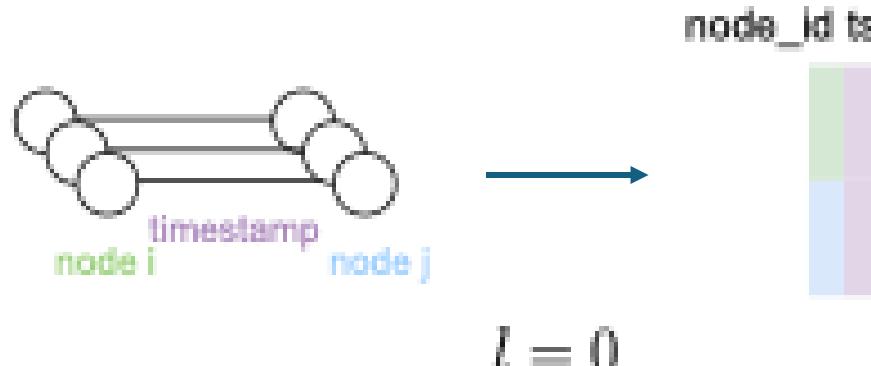
 | sample k_l neighbors within the snapshot S_s ;

end

 generate DGL MFGs;

end

end



TGL Parallel Temporal Sampler: Algorithm

Data: sorted T-CSR G

Input: root nodes n with timestamp t_n , number of layer L ,
number of neighbors in each layer k_l

Output: DGL MFGs

for l in $0..L$ **do**

```

    if  $l \geq 0$  then
        | set  $n$  and  $t_n$  to sampled neighbors in  $l - 1$ ;
    end
    binary search for each node

```

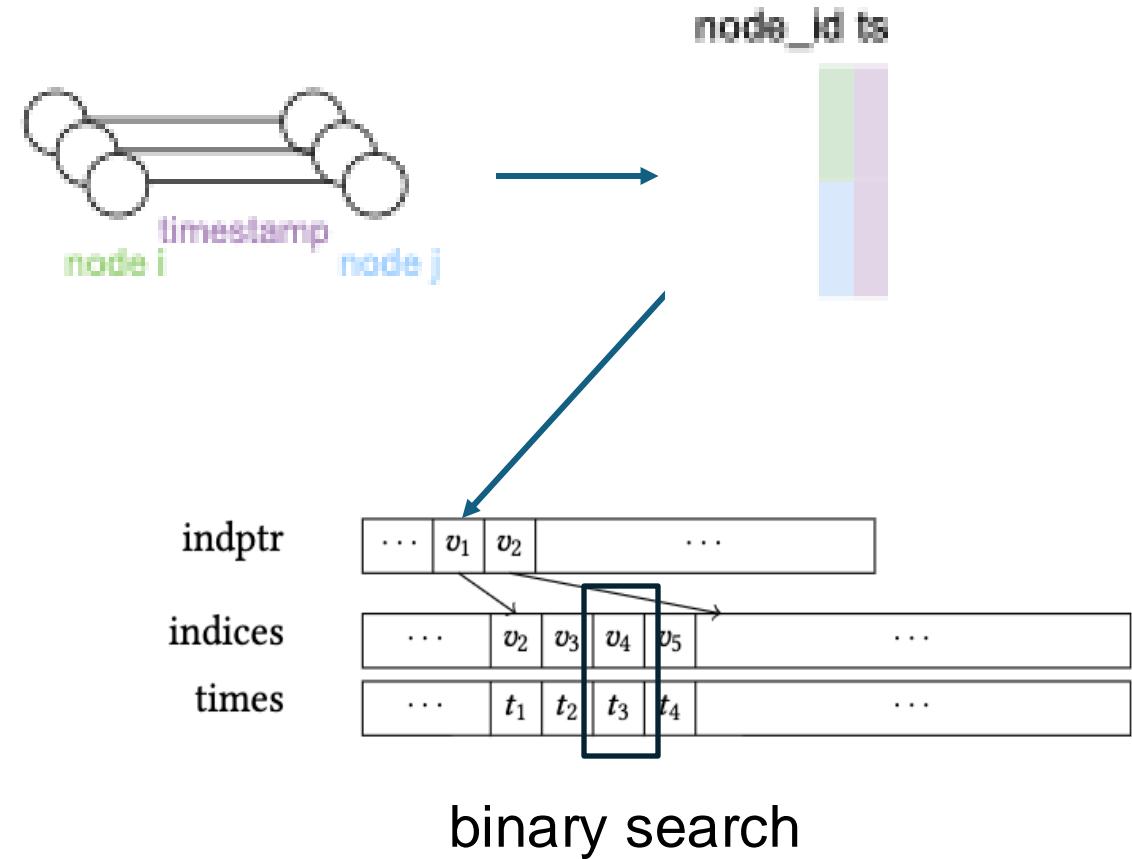
```

        foreach  $n \in n$  in parallel do
            | sample  $k_l$  neighbors within the snapshot  $S_s$ ;
        end
    
```

generate DGL MFGs;

end

end



TGL Parallel Temporal Sampler: Algorithm

Data: sorted T-CSR G

Input: root nodes n with timestamp t_n , number of layer L ,
number of neighbors in each layer k_l

Output: DGL MFGs

for l in $0..L$ **do**

if $l \geq 0$ **then**

set n and t_n to sampled neighbors in $l - 1$;

end

binary search for each node

foreach $n \in n$ **in parallel do**

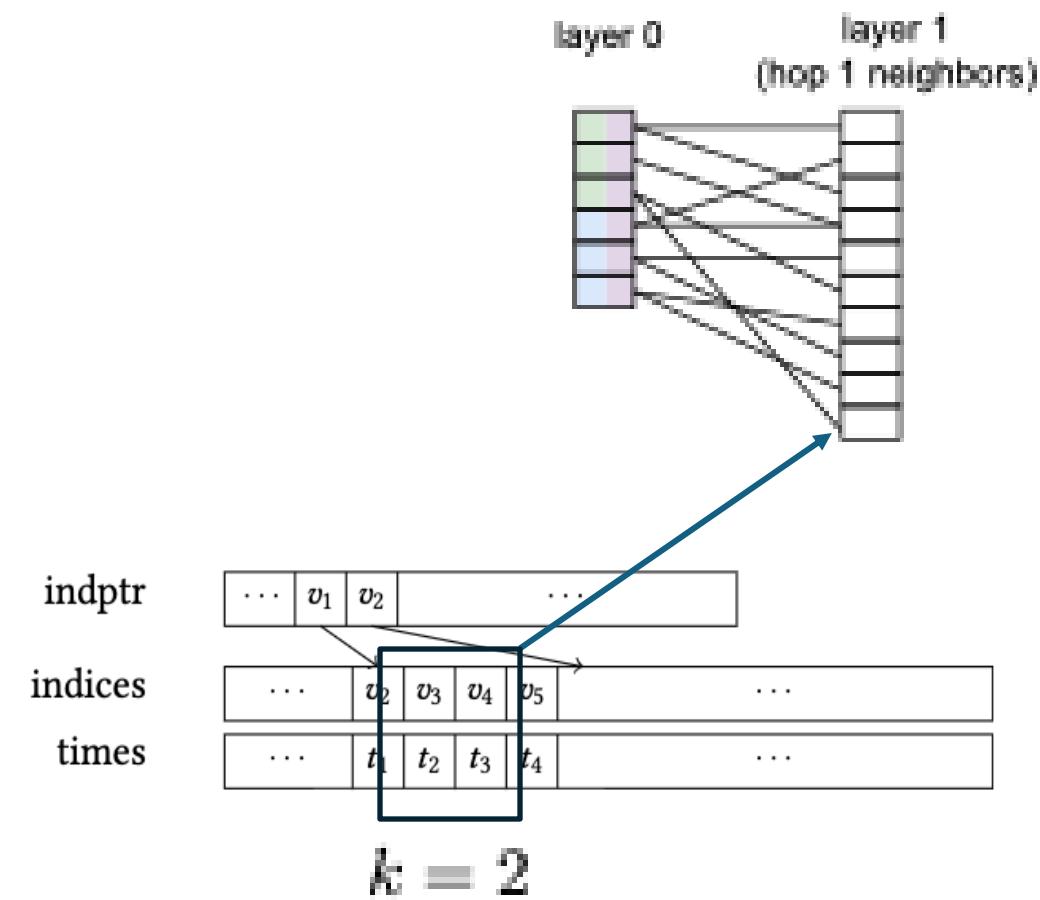
sample k_l neighbors within the snapshot S_s ;

end

generate DGL MFGs;

end

end



GPU-based Graph Sampling: GNNFlow^[1]

- Motivation: Large overhead to construct graphs

Framework	Load and Build	Partition	Training
TGL [73]	0.5 hr	N/A	2.5 hr
DGL [72]	0.7 hr	2.7 hr	5.2 hr

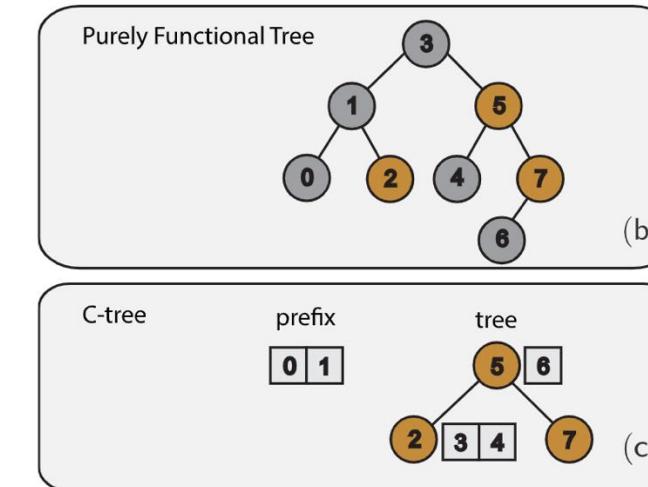
Overhead of building and partitioning large graphs

GPU-based Graph Sampling: GNNFlow

- **Motivation:** Large overhead to construct graphs
- **Challenge:** Data structure to support efficient temporal k-hop sampling
 - *Adjacent list*: traversing whole list is inefficient for long lists
 - *Block-based adjacent list*: careful selection to balance efficiency and memory use (millions of snapshots for large graphs, equal to the number of edges)
 - *Tree-of-trees*: doesn't support temporal k-hop sampling; not optimized for GPU operations

Framework	Load and Build	Partition	Training
TGL [73]	0.5 hr	N/A	2.5 hr
DGL [72]	0.7 hr	2.7 hr	5.2 hr

Overhead of building and partitioning large graphs

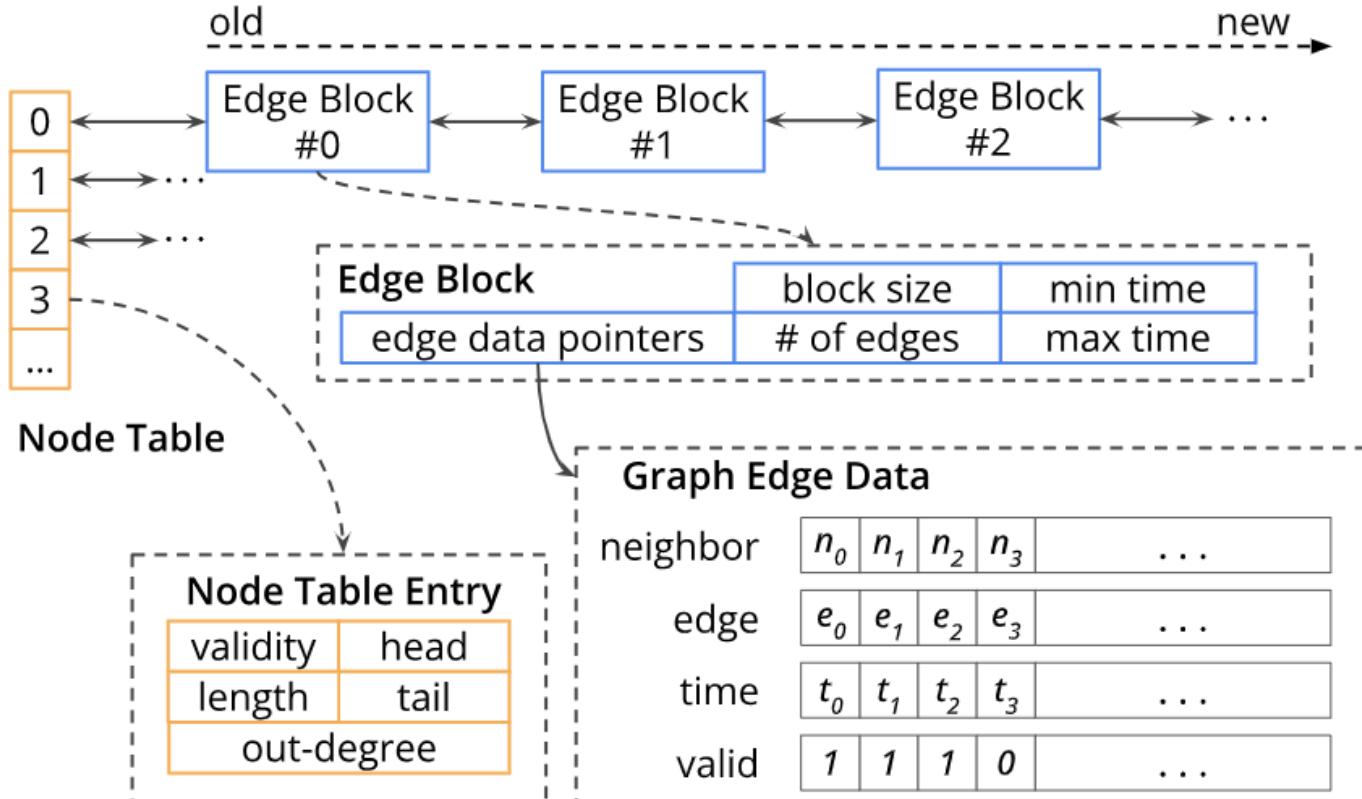


Tree-of-trees^[1]

[1] Dhulipala et. al., Low-latency graph streaming using compressed purely-functional trees.

GNNFlow Graph Sampling: In-memory Graph Storage

- Block-based neighborhood storage
 - A *node table*: contiguous array of nodes
 - A list of *edge blocks*: chronologically ordered doubly-linked list



Insertion: append new edge to tail/add into a new block

Deletion: mark the corresponding validity fields

Out-of-memory: offload old data to a file

GNNFlow GPU-based Graph Sampling: Algorithm

Input: block-based graph storage (node table V), target nodes \mathbf{n} , start timestamps t_s , end timestamps t_e , neighborhood sample size f

Output: temporal neighbors for each $n \in \mathbf{n}$

parallel for $n \in \mathbf{n}, t_s \in \mathbf{t}_s, t_e \in \mathbf{t}_e$ **do**

 candidates = []

$lst = V[n]$

$b = lst.tail$ // the newest block

while $b \neq null$ **do**

if $t_e < b.t_{min}$: $b = b.prev$ **continue**

if $t_s > b.t_{max}$: **break**

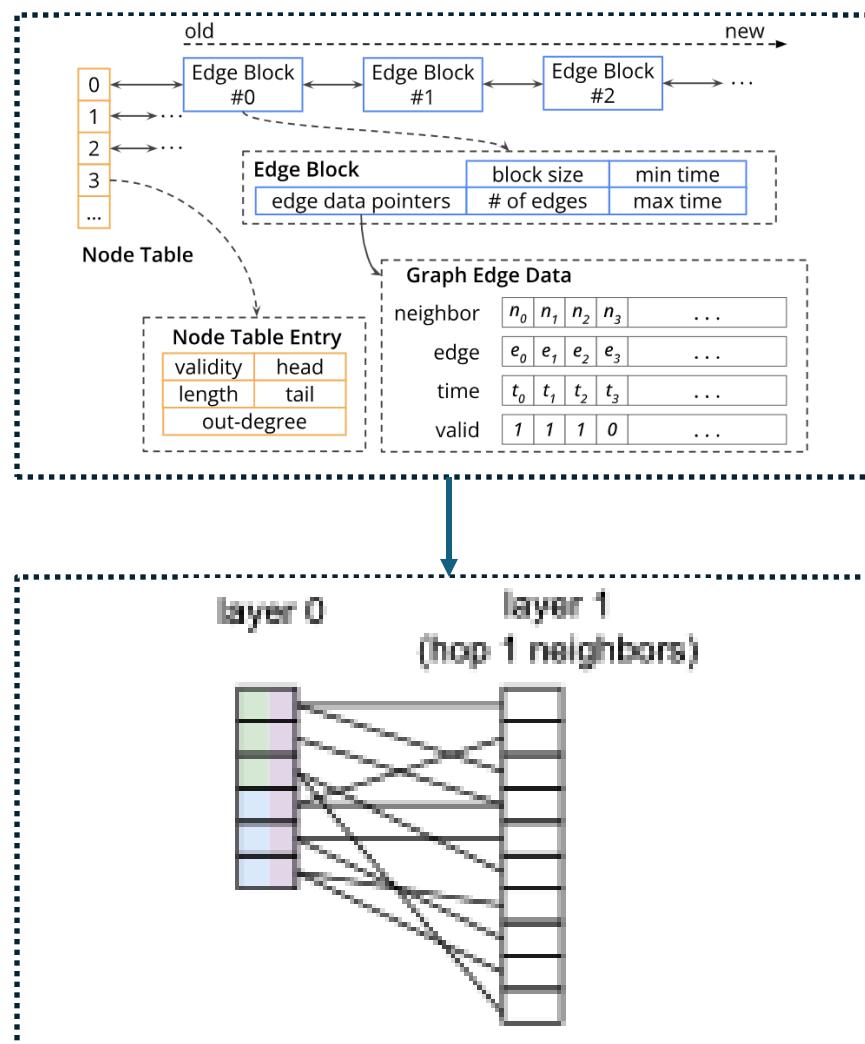
 find neighbors within time window $[t_s, t_e)$ using binary search in b and add them to candidates

$b = b.prev$

end while

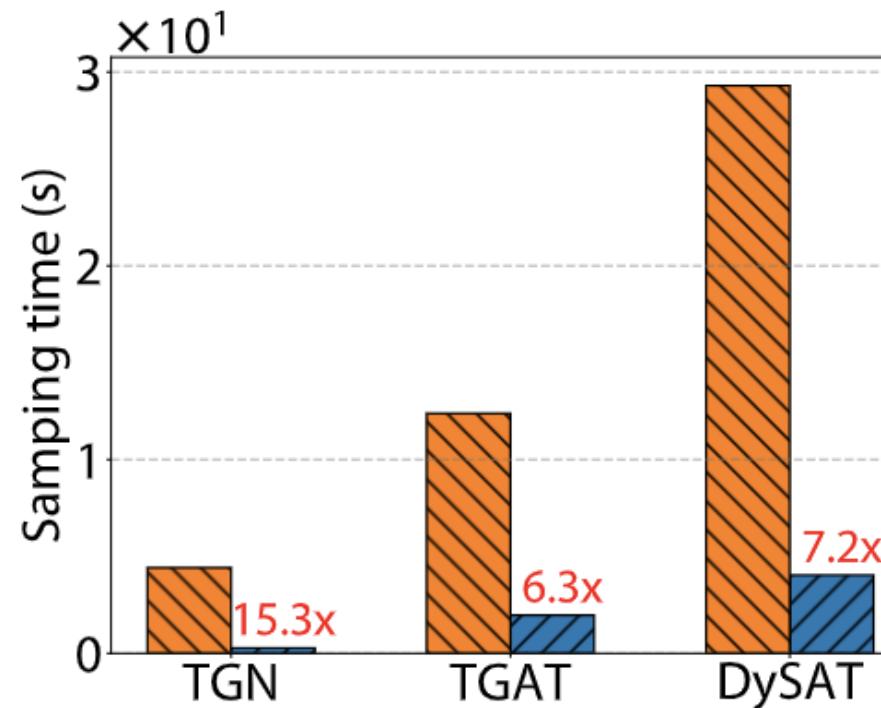
 sample f neighbors from candidates

end for



GNNFlow GPU-based Graph Sampling: Speedup

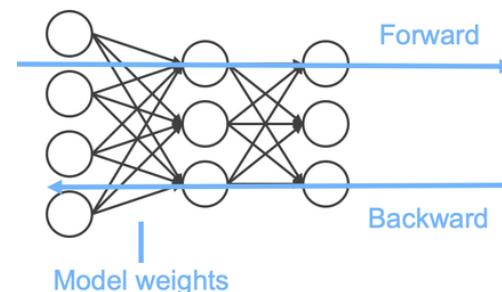
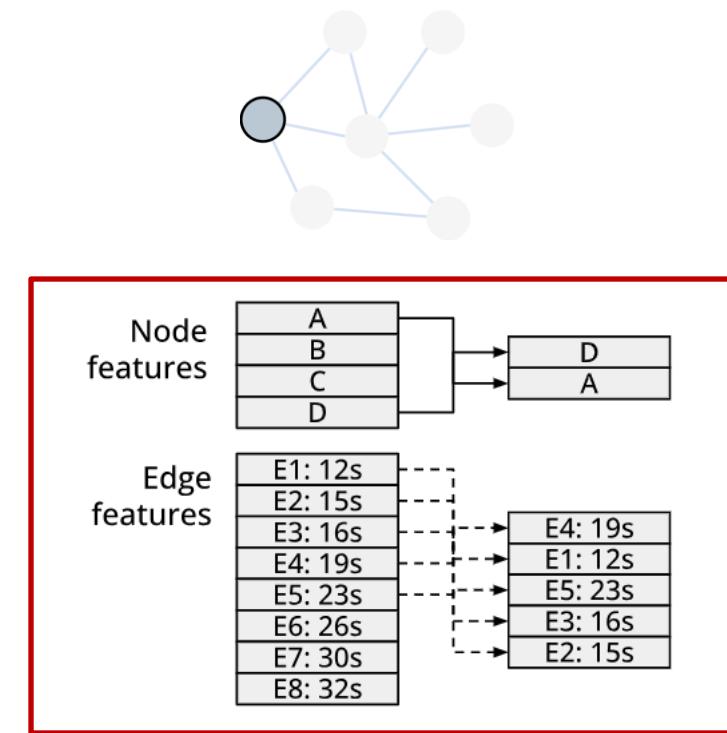
 GNNFlow  TGL



Sampling time: 7.2x to 15.3x speedup

Training Acceleration

- Introduction & Background
- Neighbor Sampling Acceleration
- Feature Fetching/Update Acceleration
- Model Computation Acceleration



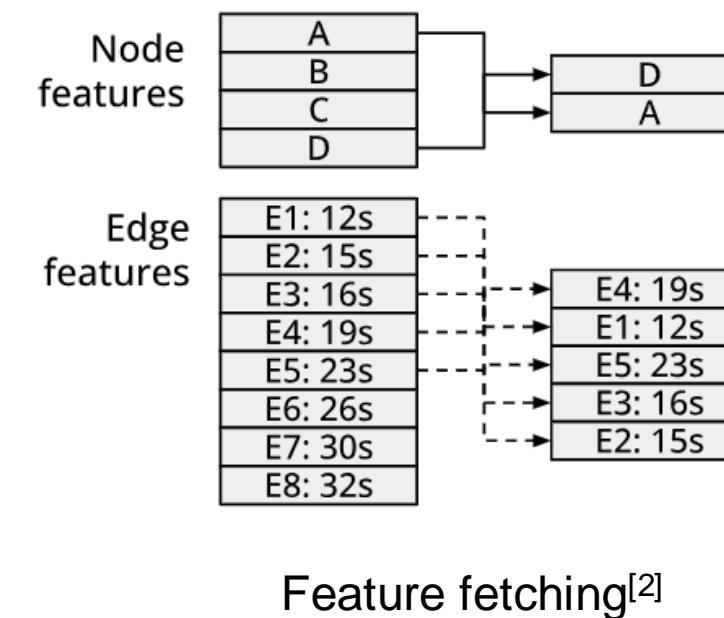
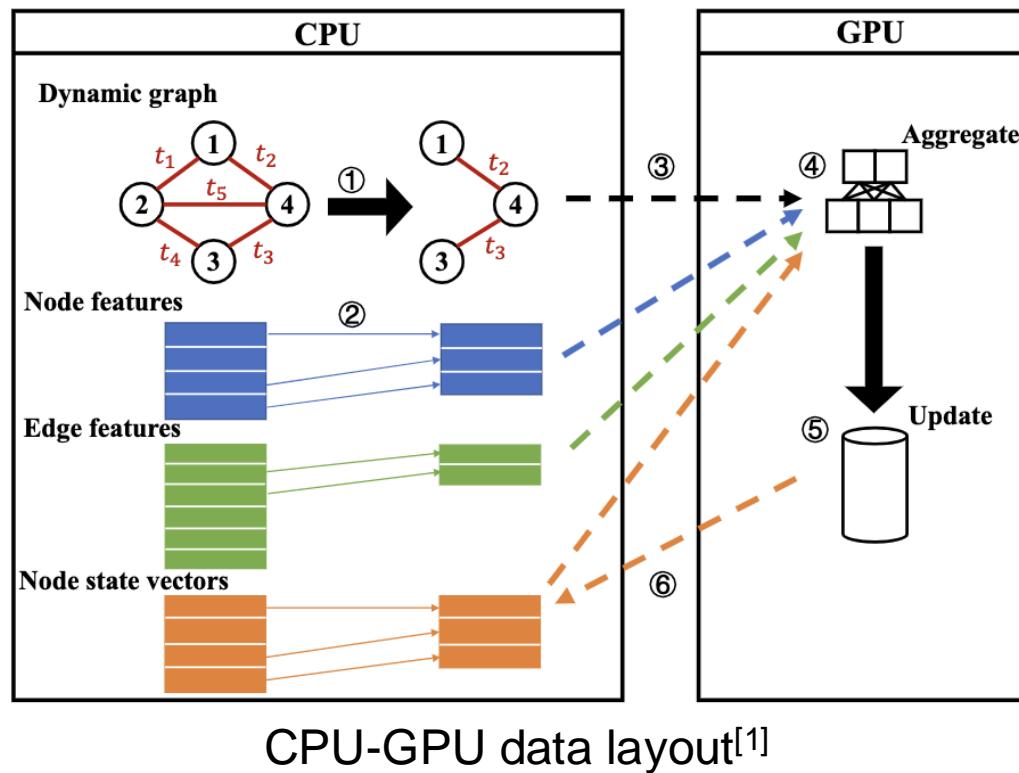
Background: Feature Fetching & Feature Update

- Feature fetching:

 - Fetch feature data corresponding to sampled graph to GPU memory.

- Feature Update:

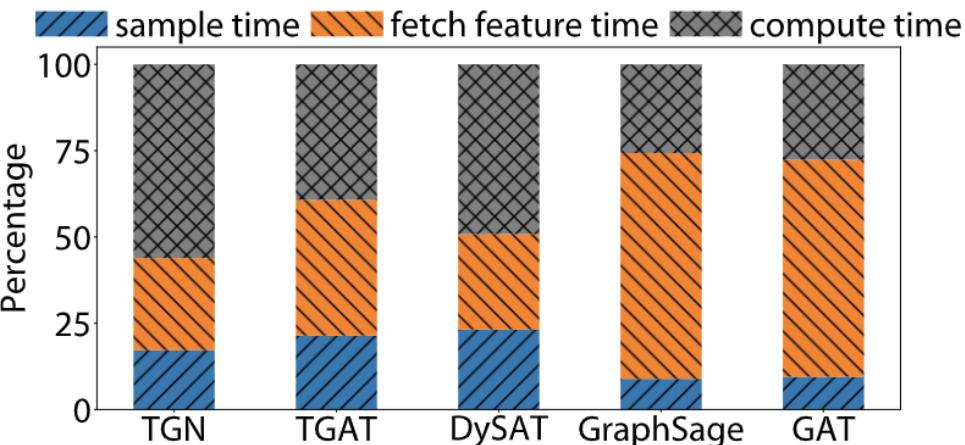
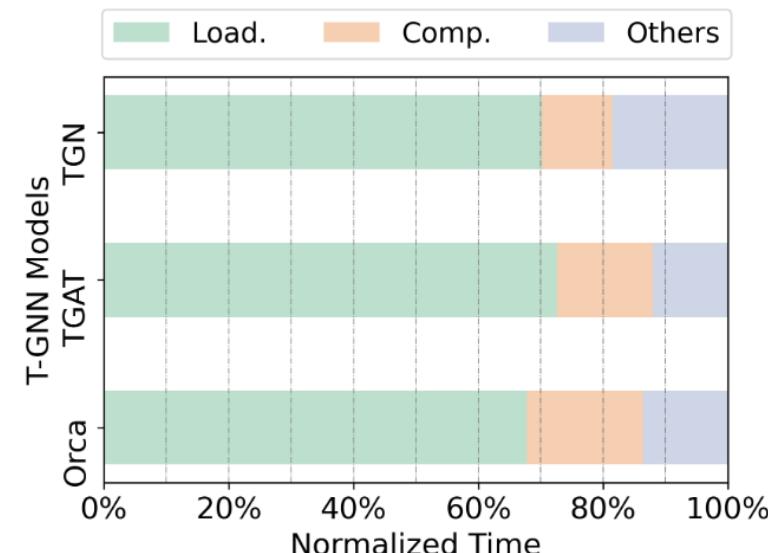
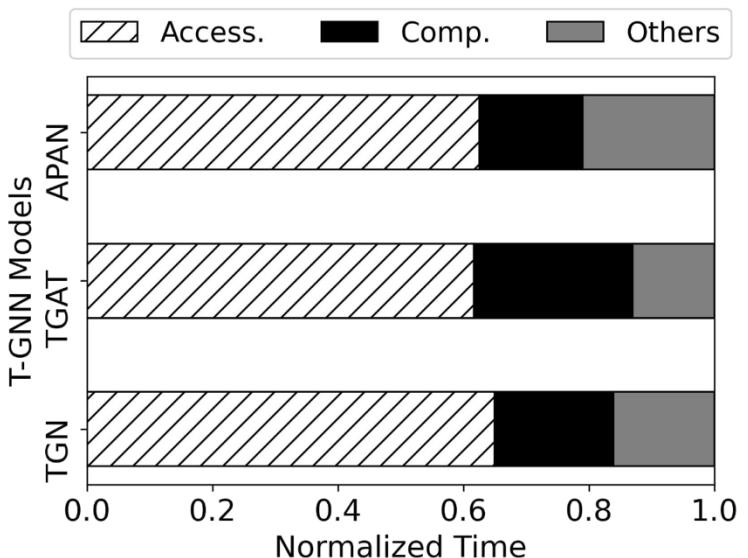
 - Write back updated node features (node state vectors).



[1] Gao et. al., ETC: Efficient Training of Temporal Graph Neural Networks over Large-Scale Dynamic Graphs.

[2] Zhong et. al., GNNFlow: A Distributed Framework for Continuous Temporal GNN Learning on Dynamic Graphs.

Motivation: Data Access Bottleneck

GNNFlow^[1]*ETC*^[2]*SIMPLE*^[3]

[1] Zhong et. al., GNNFlow: A Distributed Framework for Continuous Temporal GNN Learning on Dynamic Graphs.

[2] Gao et. al., ETC: Efficient Training of Temporal Graph Neural Networks over Large-Scale Dynamic Graphs.

[3] Gao et. al., SIMPLE: Efficient Temporal Graph Neural Network Training at Scale with Dynamic Data Placement.

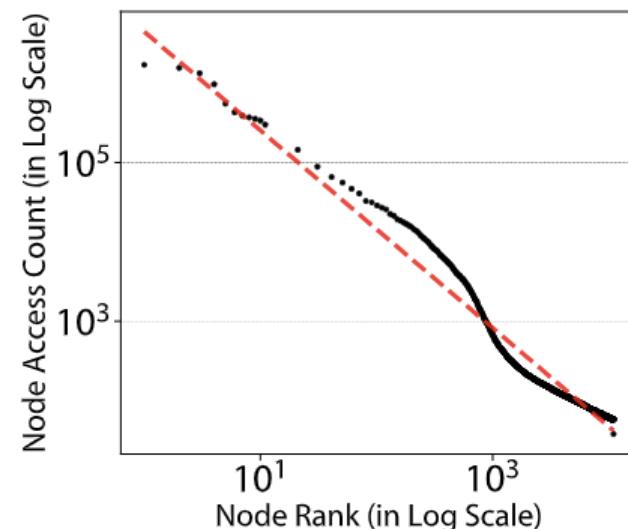
GPU Feature Cache: GNNFlow

□ Observation:

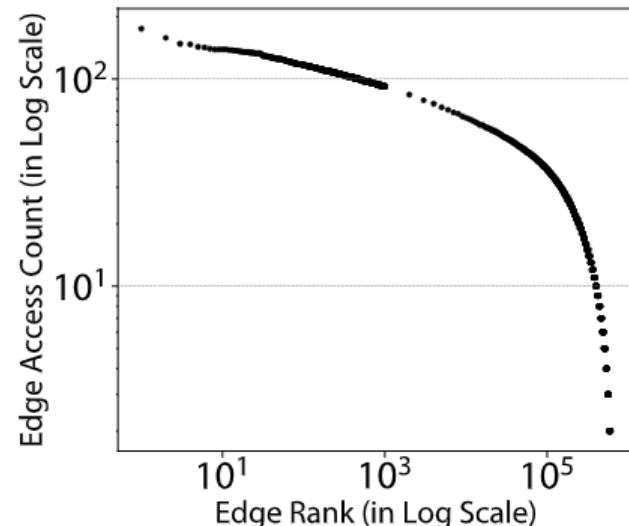
- Opportunity in *data reuse*: duplication in the nodes and edges sampled between adjacent continuous leaning rounds
- Data access pattern:
 - power-law node access; exponential edge access

□ Method:

- Cache reuse and restoration
 - Inter-round duplication
 - Save cache content
- Vectorized cache
- Parallel access and update



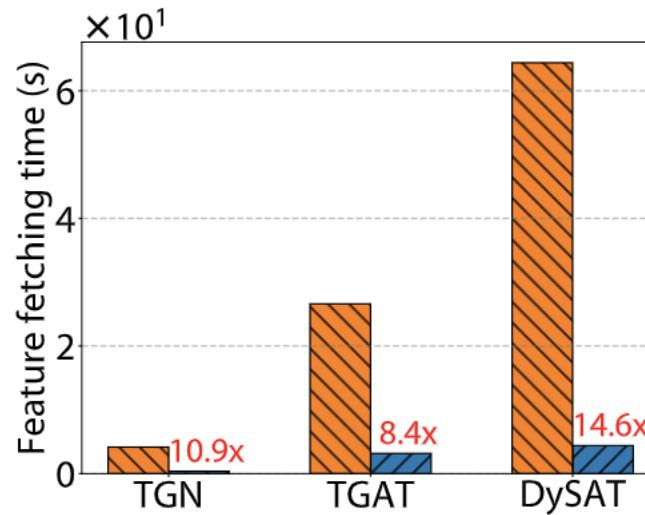
Power-law



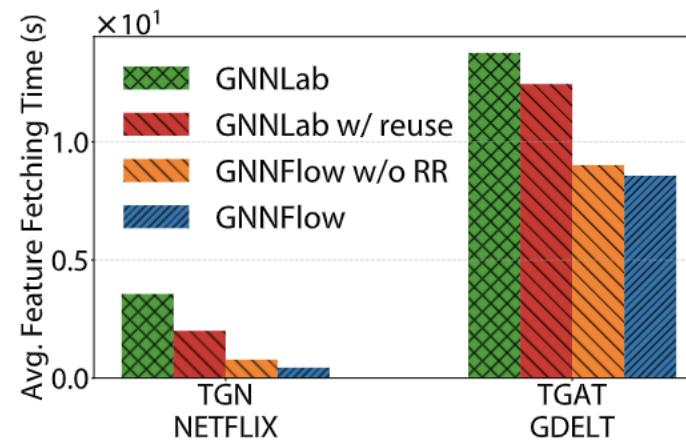
Exponential

GNNFlow GPU Feature Cache: Evaluation

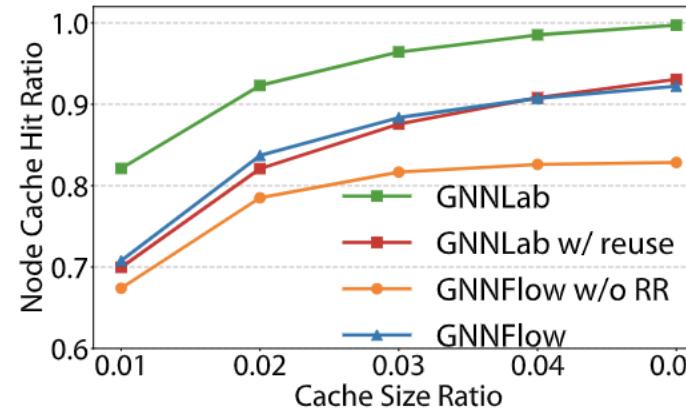
 GNNFlow  TGL



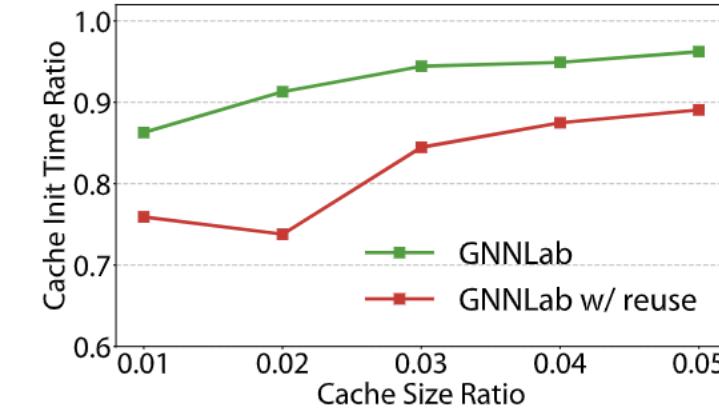
Feature fetching: 10.9x to 14.6x speedup



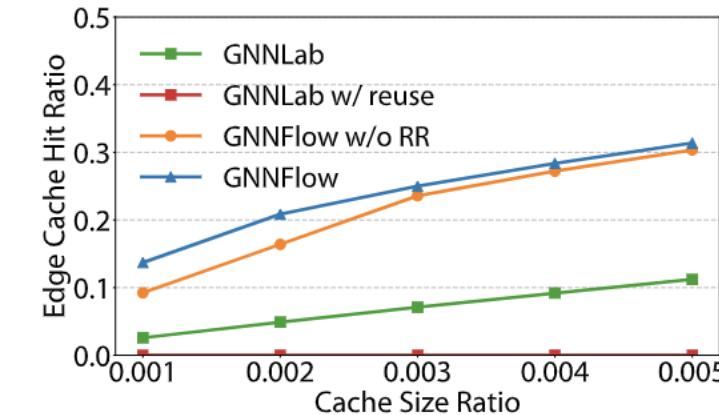
(a) Effect on different models



(c) Effect of node feature



(b) Cache init. time ratio



(d) Effect of edge feature

Reducing Data Loading Volume: ETC^[1]

□ Observation:

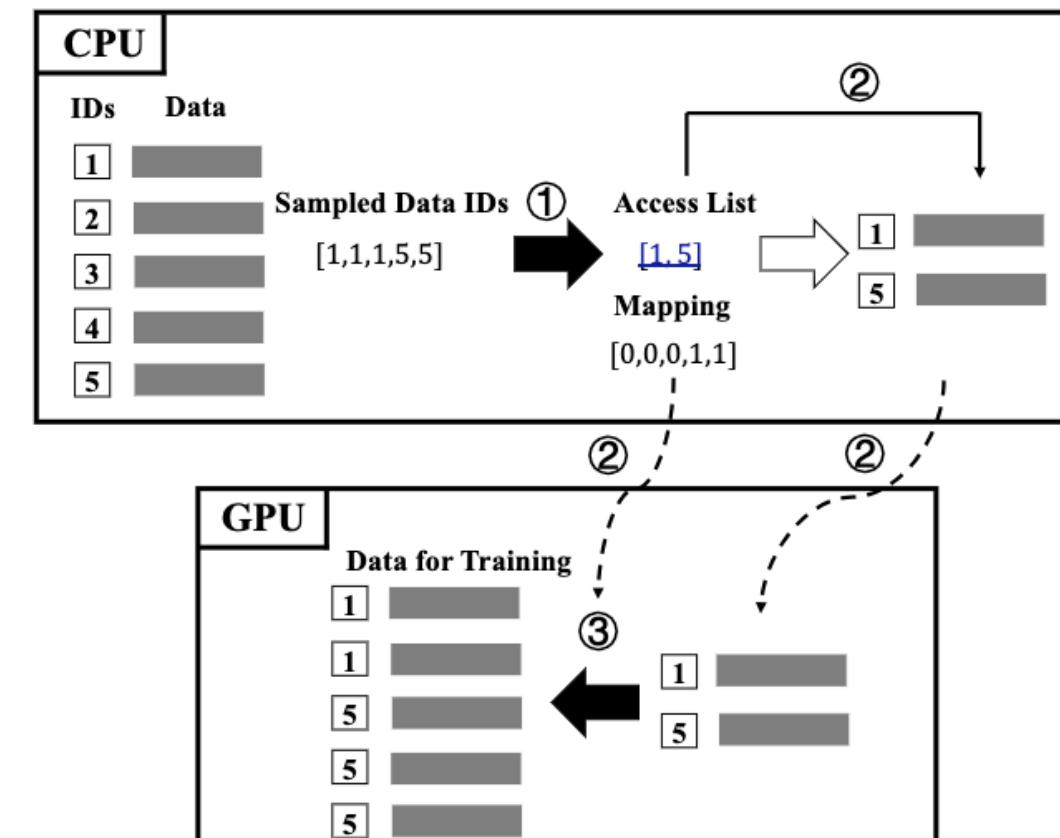
- Redundant data access can take up over 80% of the total data access volume

□ Method: three-step data access policy *Supra*

- Identify redundant access
- Access unique data
- Reconstruct originally required data

□ Inter-batch pipeline:

- Ensure lower overhead
- Concurrent model computation and data access
- Other works adopting pipeline:
 - TGL^[2], DistTGL^[3]

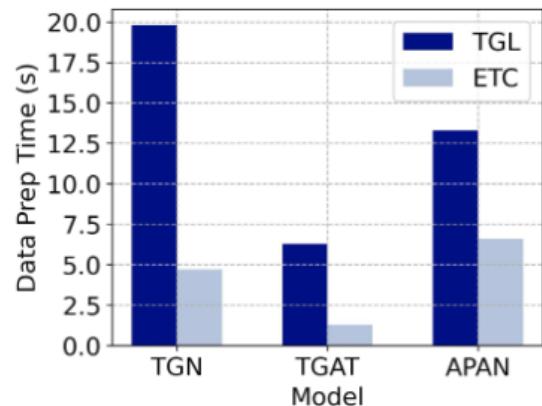


[1] Gao et. al., ETC: Efficient Training of Temporal Graph Neural Networks over Large-Scale Dynamic Graphs.

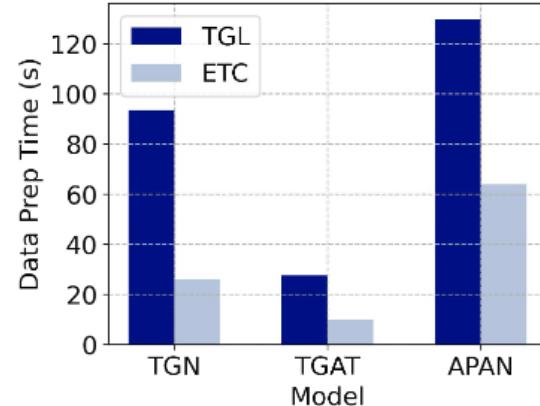
[2] Zhou et. al., TGL: A General Framework for Temporal GNN Training on Billion-Scale Graphs.

[3] DistTGL: Distributed Memory-Based Temporal Graph Neural Network Training.

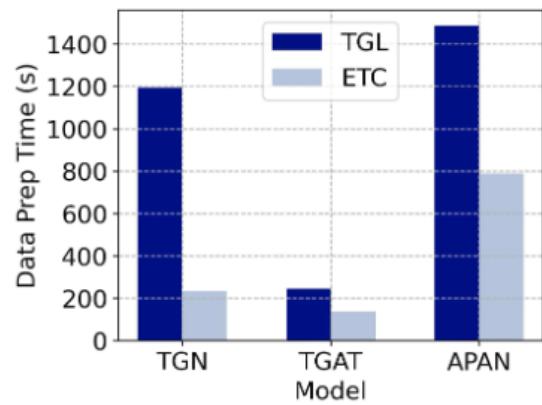
ETC Reducing Data Loading Volume: Evaluation



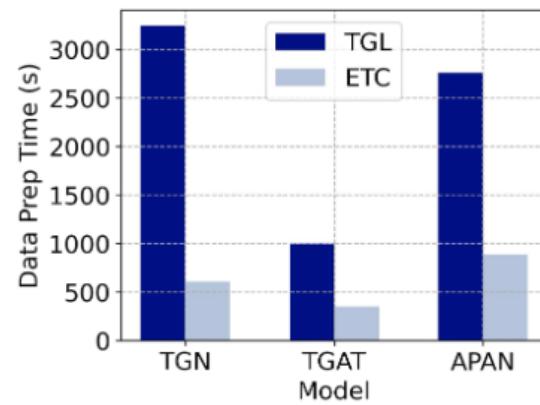
(a) LastFM



(b) Wiki-Talk



(c) Stack-Overflow



(d) GDELT

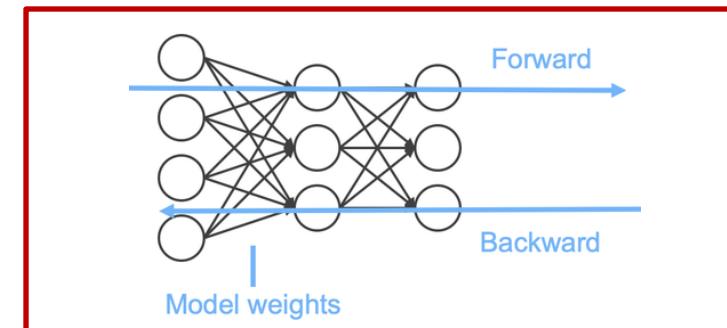
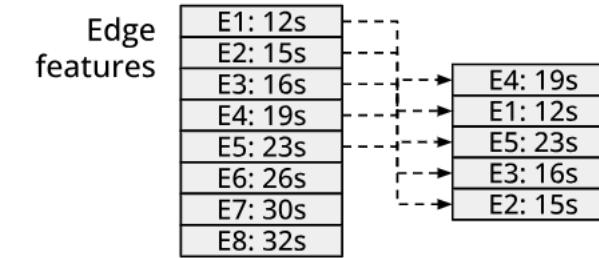
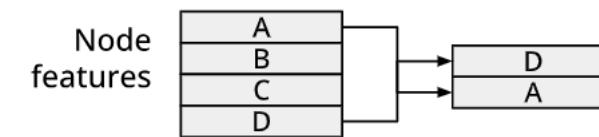
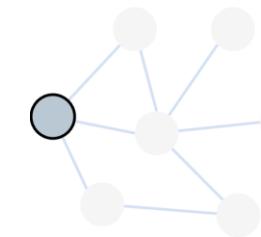
Per-epoch data preparation time (s)

	LastFM	Wiki-Talk	Stack-Overflow	GDELT
TGL	14.2 (4.9×)	66.3 (4.2×)	906.1 (3.9×)	2599.6 (6.1×)
ETC	2.9	15.9	233.3	427.5

Per-epoch data access time (s): 4.9x to 6.1x speedup

Training Acceleration

- **Introduction & Background**
- **Neighbor Sampling Acceleration**
- **Feature Fetching/Update Acceleration**
- **Model Computation Acceleration**

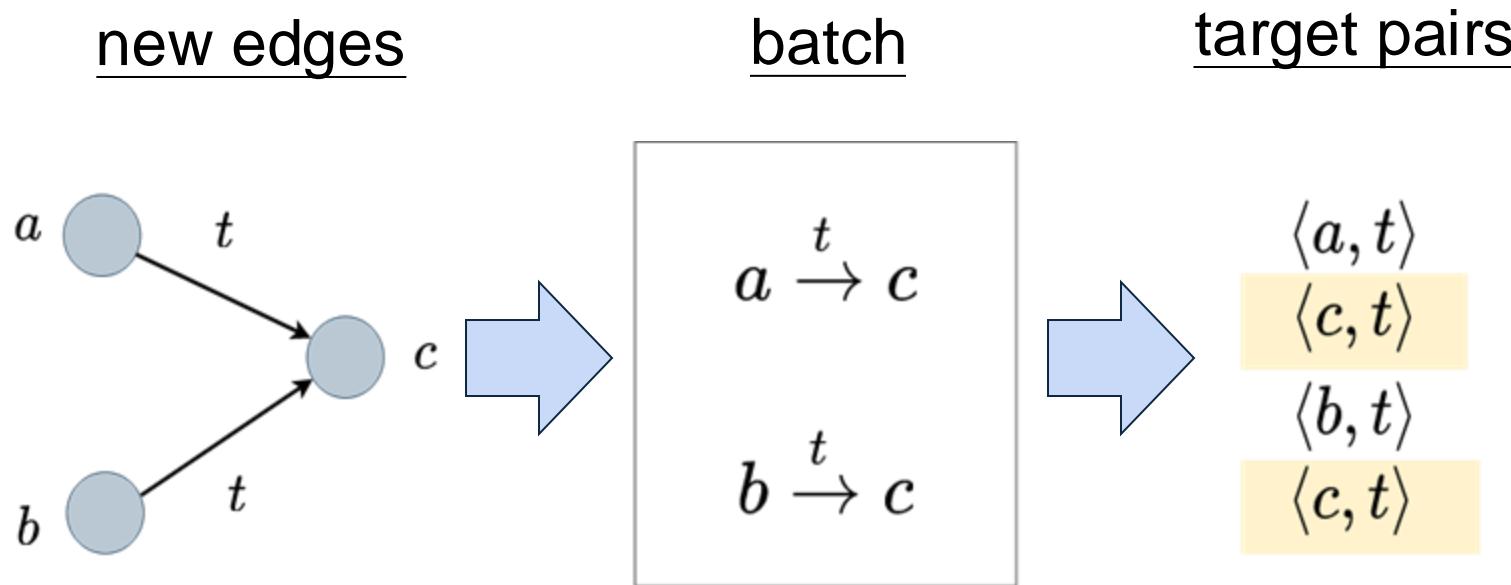


Removing Computation Redundancy: TGLite^[1]

□ Observation:

- Duplication from batched edges
- Nodes share common neighbors, leading to duplicate target pairs in batch processing

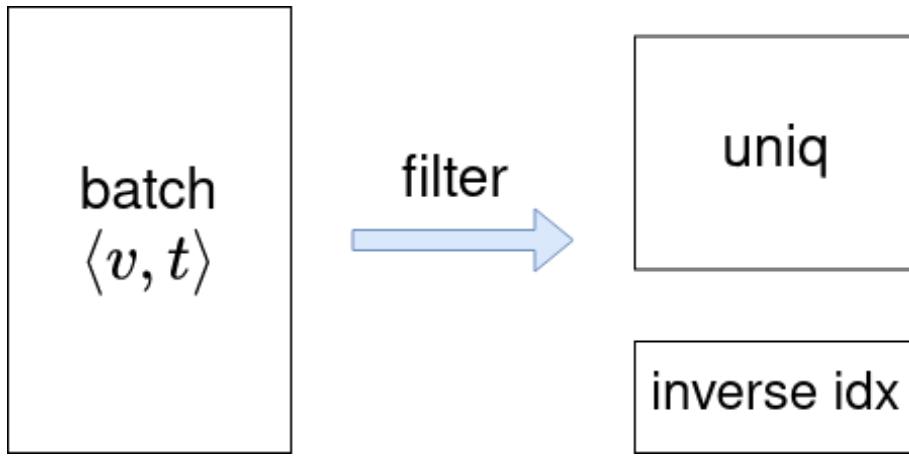
□ Method: deduplication



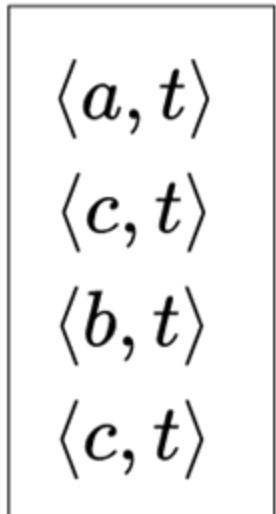
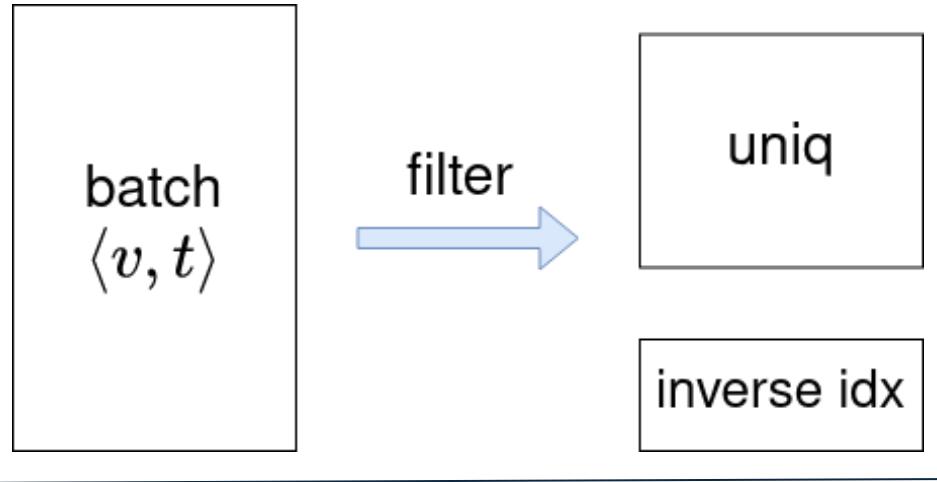
Dataset	TGAT layer 1	TGAT layer 2
jodie-lastfm	48%	0%
jodie-mooc	74%	2%
jodie-reddit	41%	0%
jodie-wiki	68%	0%
snap-email	55%	19%
snap-msg	70%	16%
snap-reddit	35%	8%

[1] Wang et. al., TGLite: A Lightweight Programming Framework for Continuous-Time Temporal Graph Neural Networks

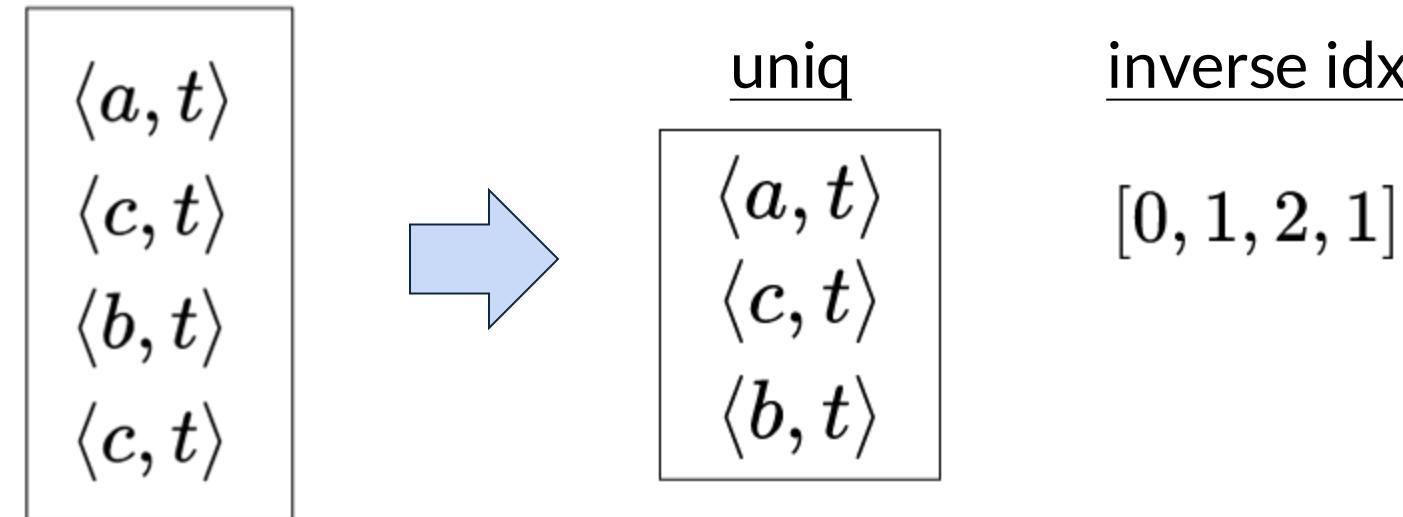
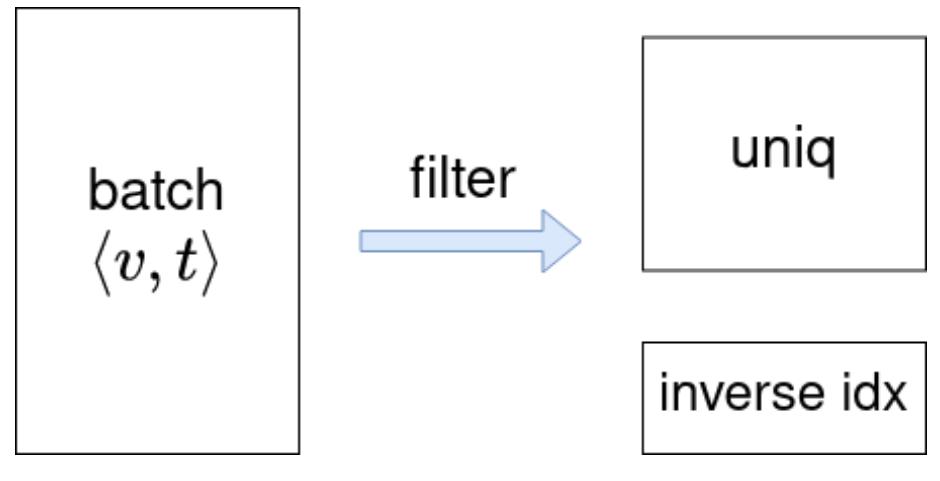
Removing Computation Redundancy: TGLite



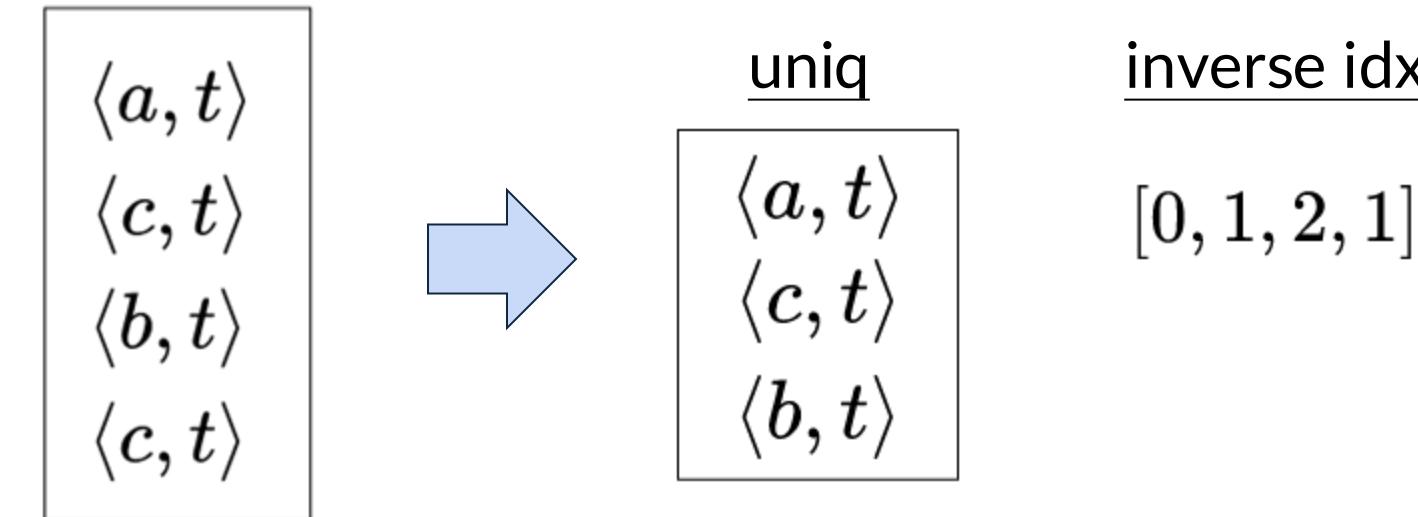
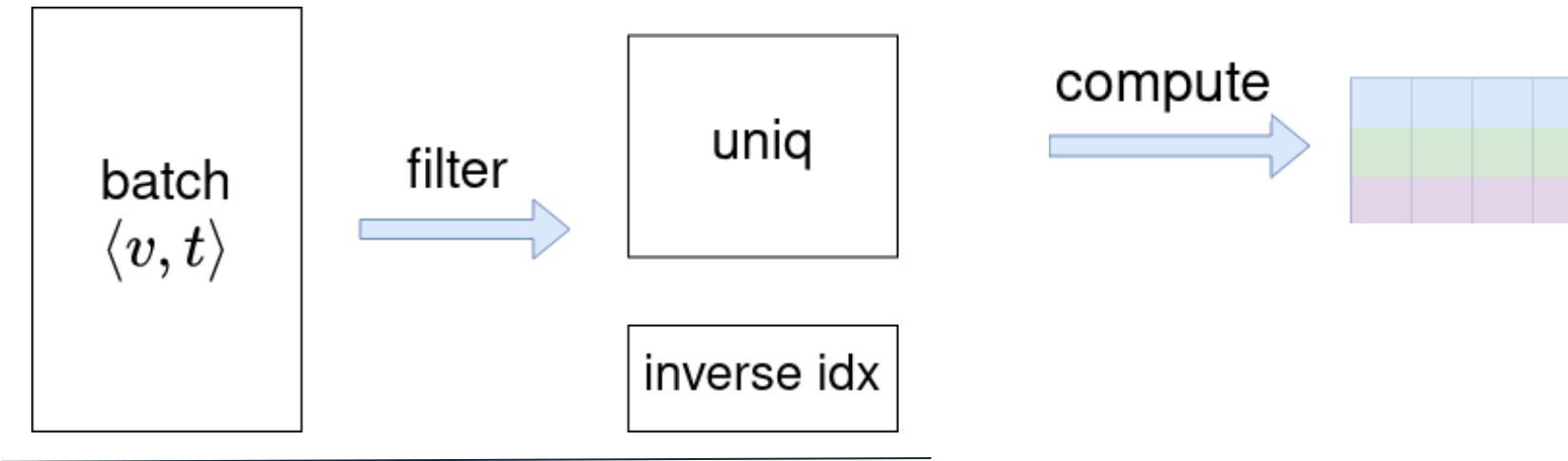
Removing Computation Redundancy: TGLite



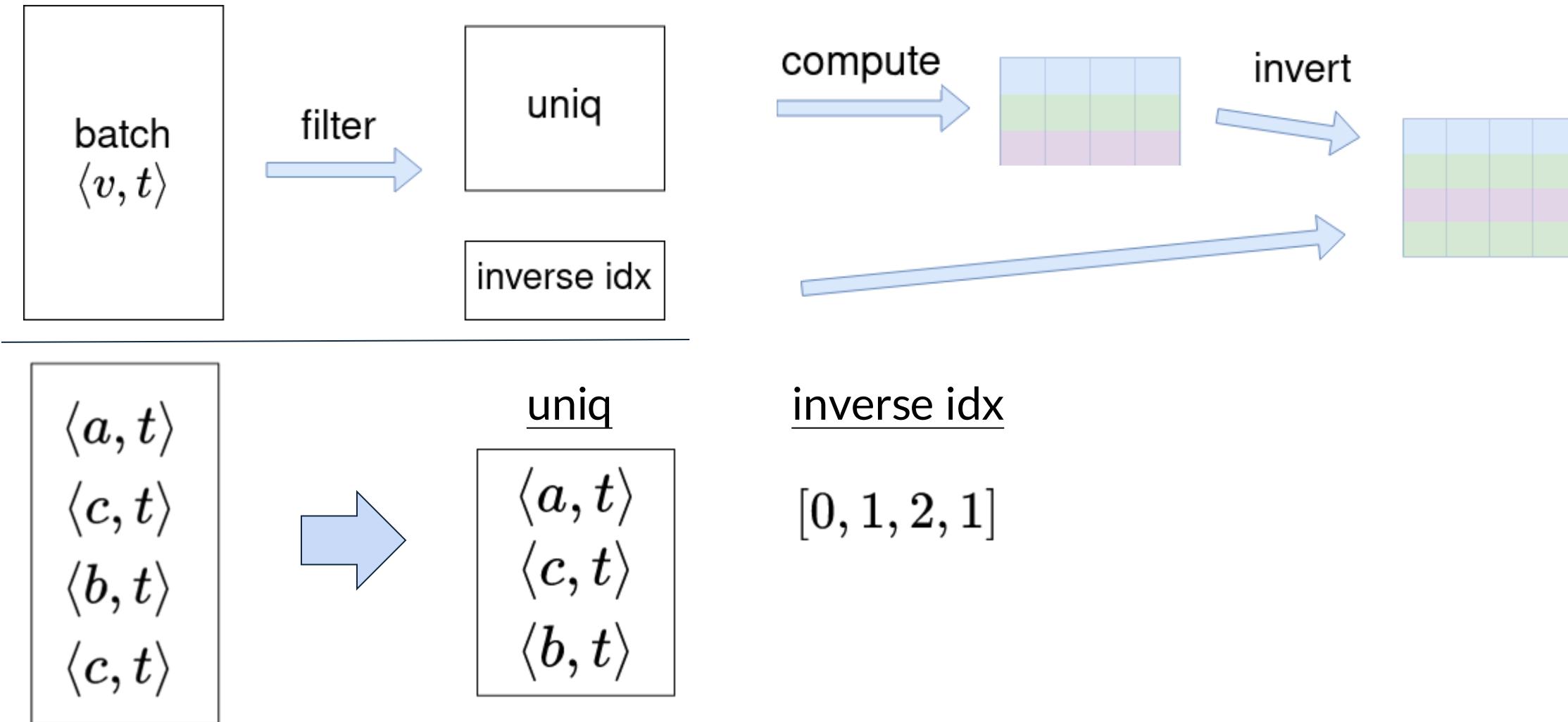
Removing Computation Redundancy: TGLite



Removing Computation Redundancy: TGLite

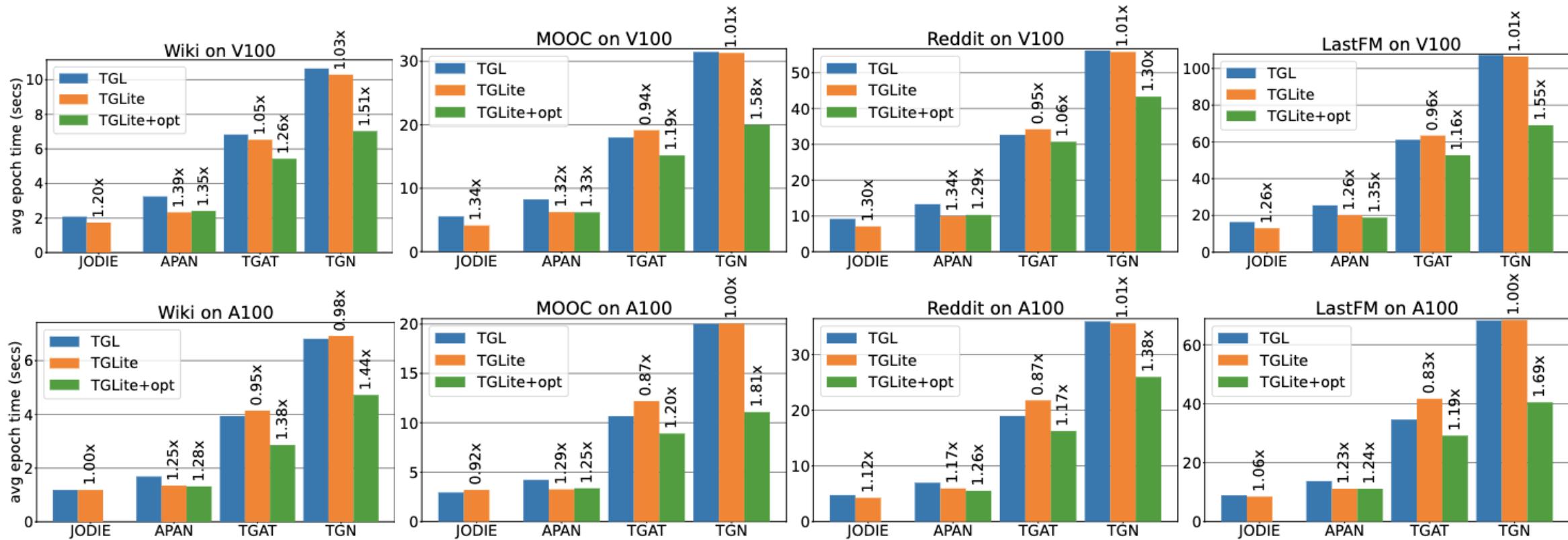


Removing Computation Redundancy: TGLite



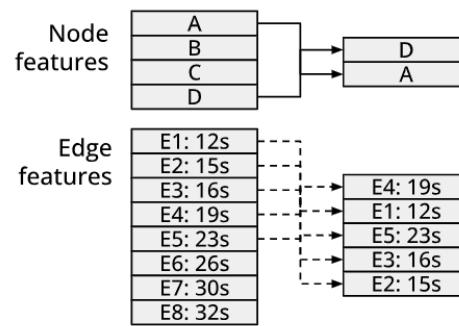
TGLite Deduplication: Evaluation

lower = better



Training time per epoch (seconds) for different datasets with data residing on GPU device memory.

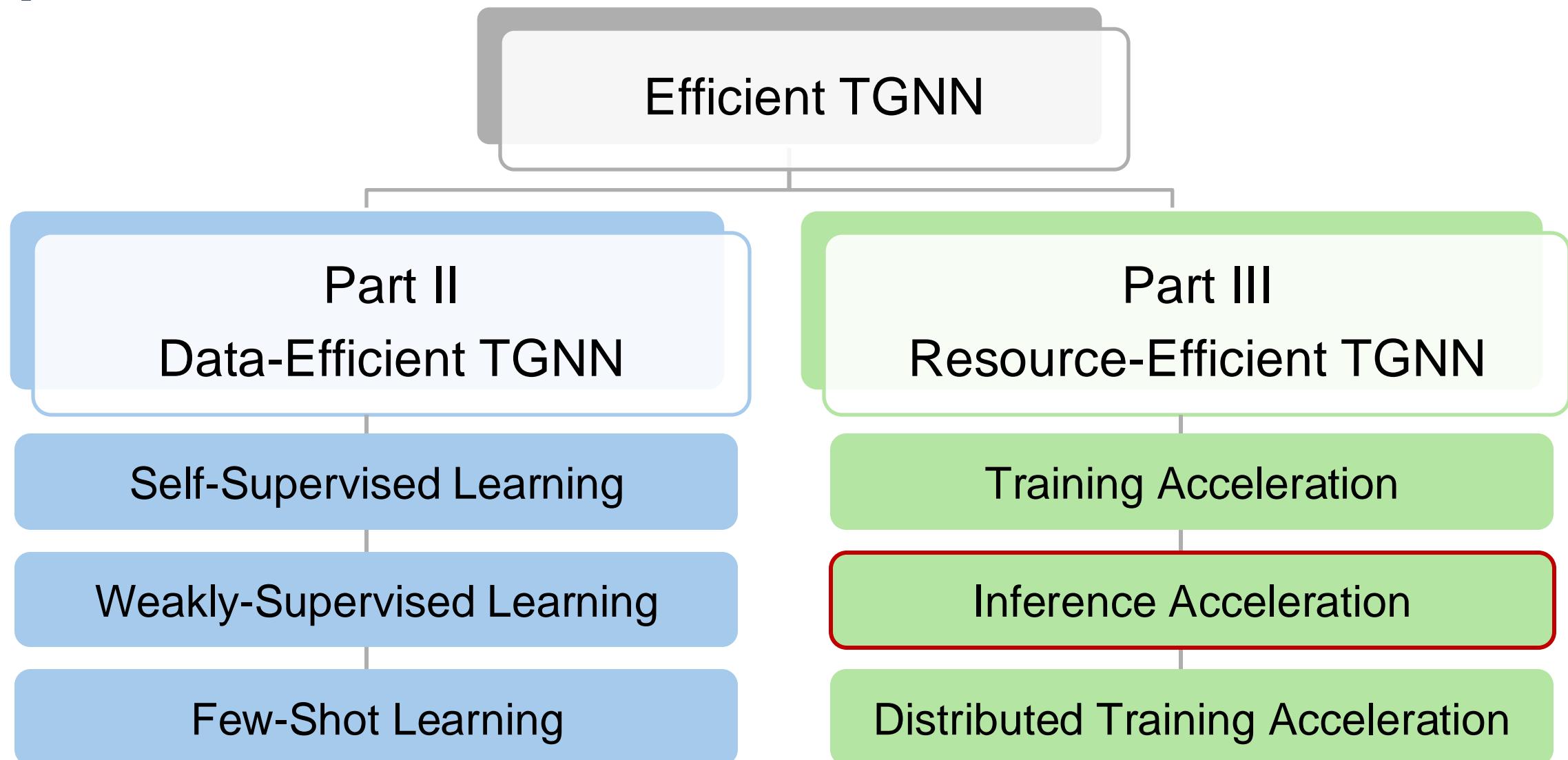
Part III Summary: Resource-Efficient TGNN



Training acceleration

- ❑ Introduction & Background
- ❑ Parallel neighbor sampler
- ❑ Data access optimization
- ❑ Model computation deduplication

Scope of This Tutorial



Inference Acceleration

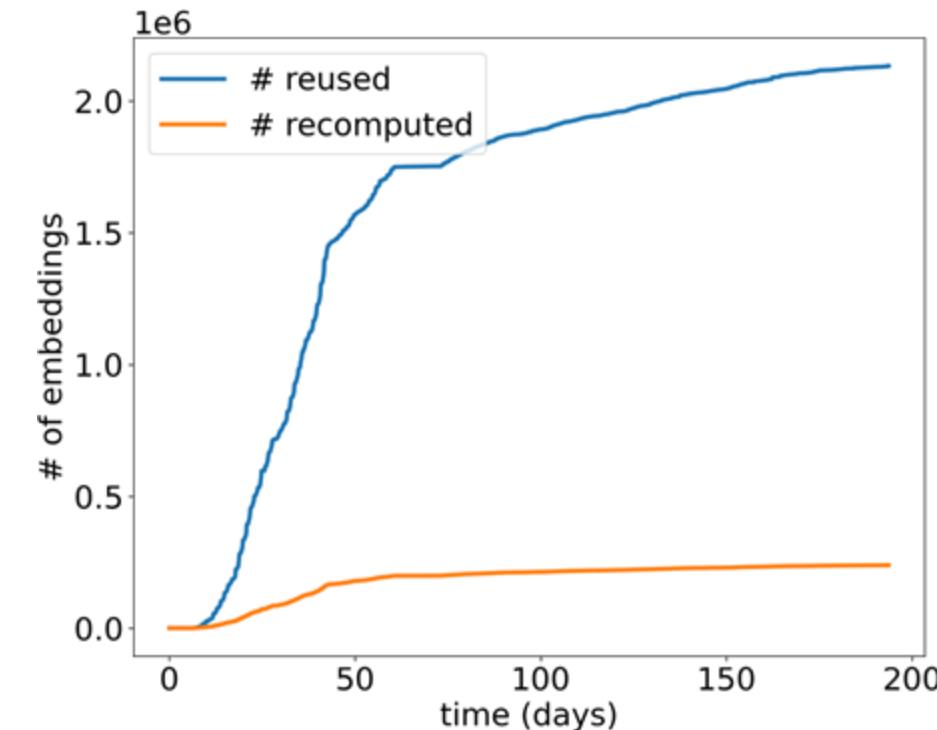
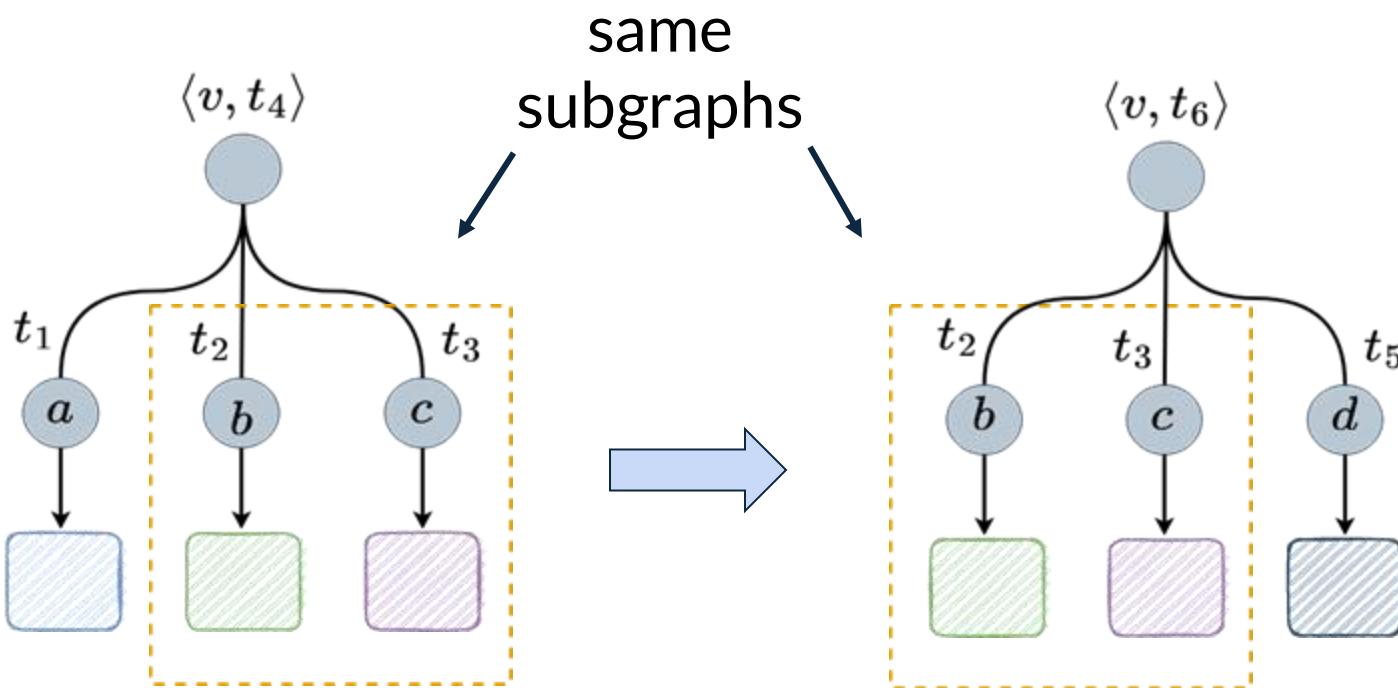
- **Introduction & Background**
- **Embedding Memoization**
- **Time Encoding Precomputation**

Inference Acceleration

- **Introduction & Background**
- **Embedding Memoization**
- **Time Encoding Precomputation**

Embedding Memoization: TGLite

- **Observation:**
 - Redundant embeddings over time
 - Given same target pair, sampled temporal subgraph is the same
- **Method:** Embedding memoization



Embeddings reused vs.
(re)computed over time

Inference Acceleration

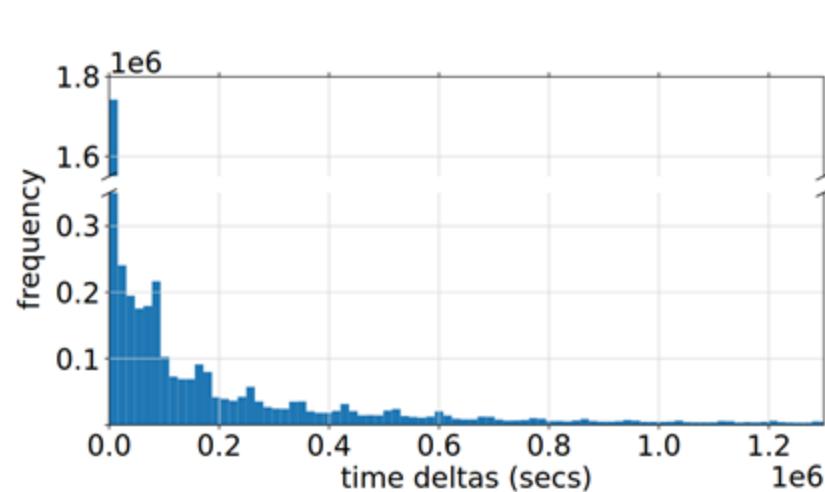
- **Introduction & Background**
- **Embedding Memoization**
- **Time Encoding Precomputation**

Time Encoding Precomputation: TGLite

- **Observation:** repeated time-encodings
- **Method:** precompute time window ahead-of-time and reuse time-encoding vectors

$$\boxed{z_i(t) = h_i^{(l-1)}(t) \parallel \dots \parallel \Phi(0)}$$
$$z_j(t) = h_j^{(l-1)}(t_j) \parallel \dots \parallel \Phi(t - t_j)$$

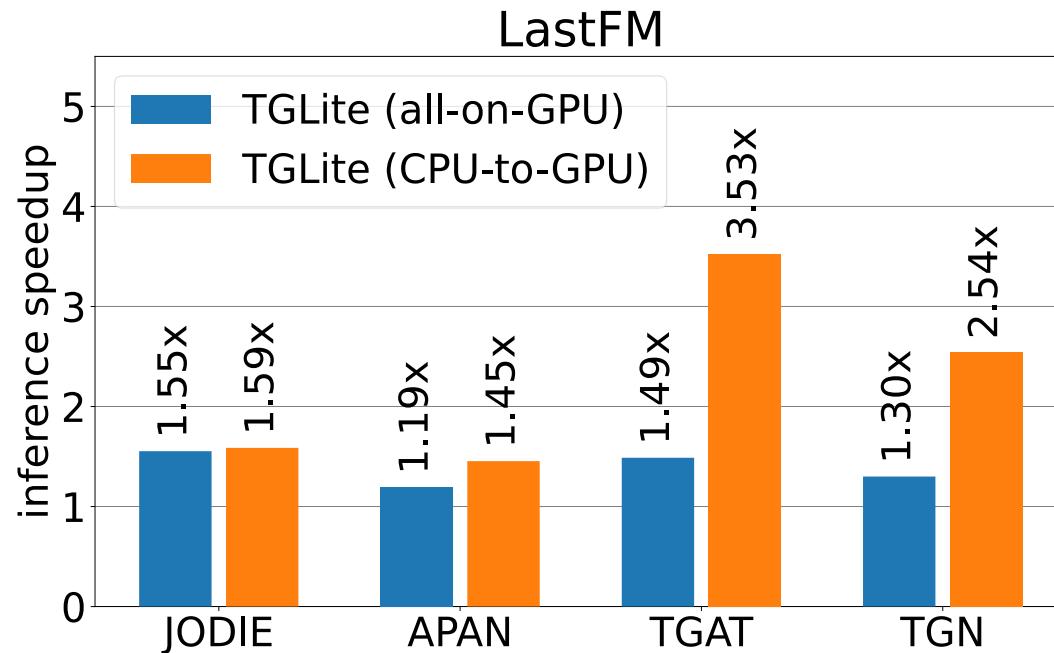
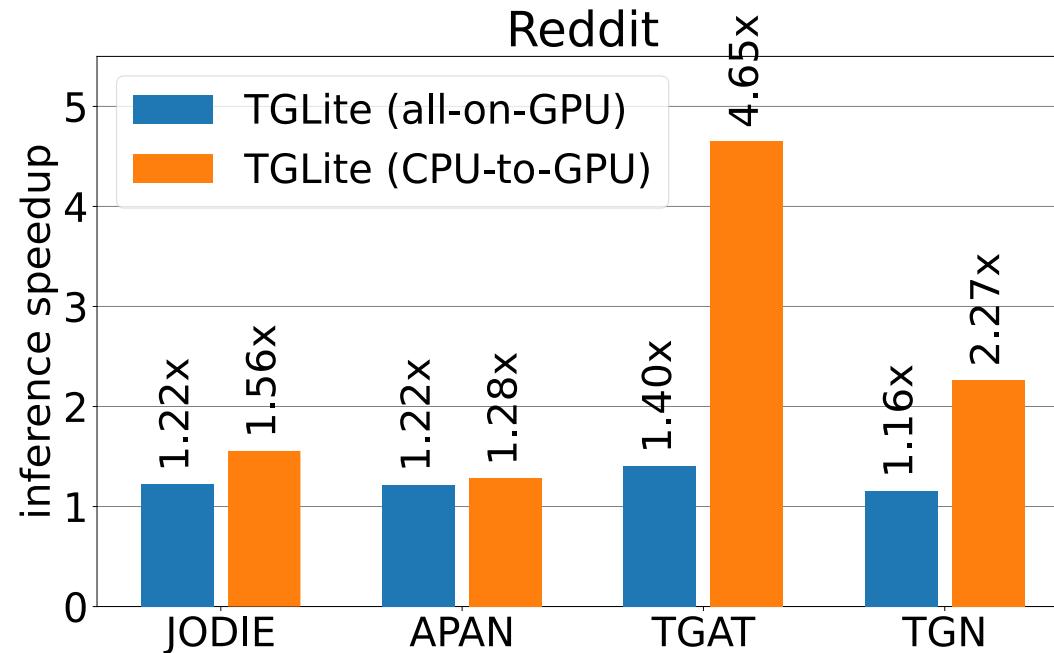
always 0
(constant)



Distribution of time deltas
processed by time-encoder

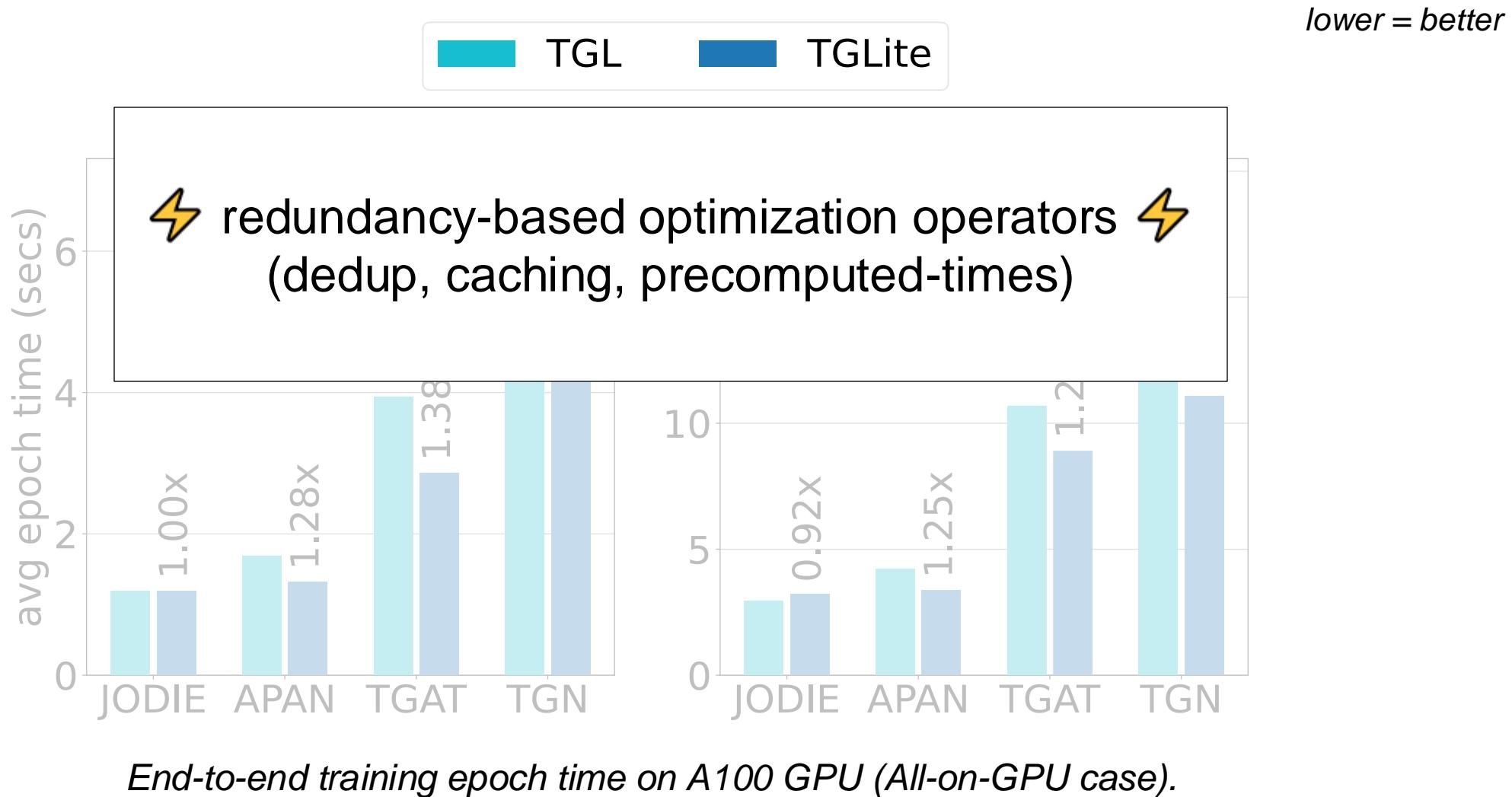
TGLite Inference Optimizations: Evaluation

higher = better

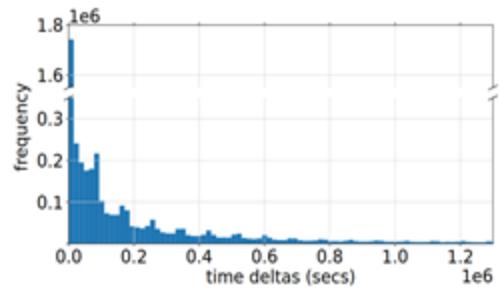
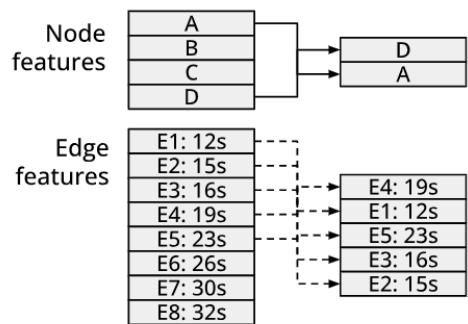


Relative inference speedup of TGLite against respective TGL baseline on A100 GPU.

TGLite End-to-End Training Time



Part III Summary: Resource-Efficient TGNN



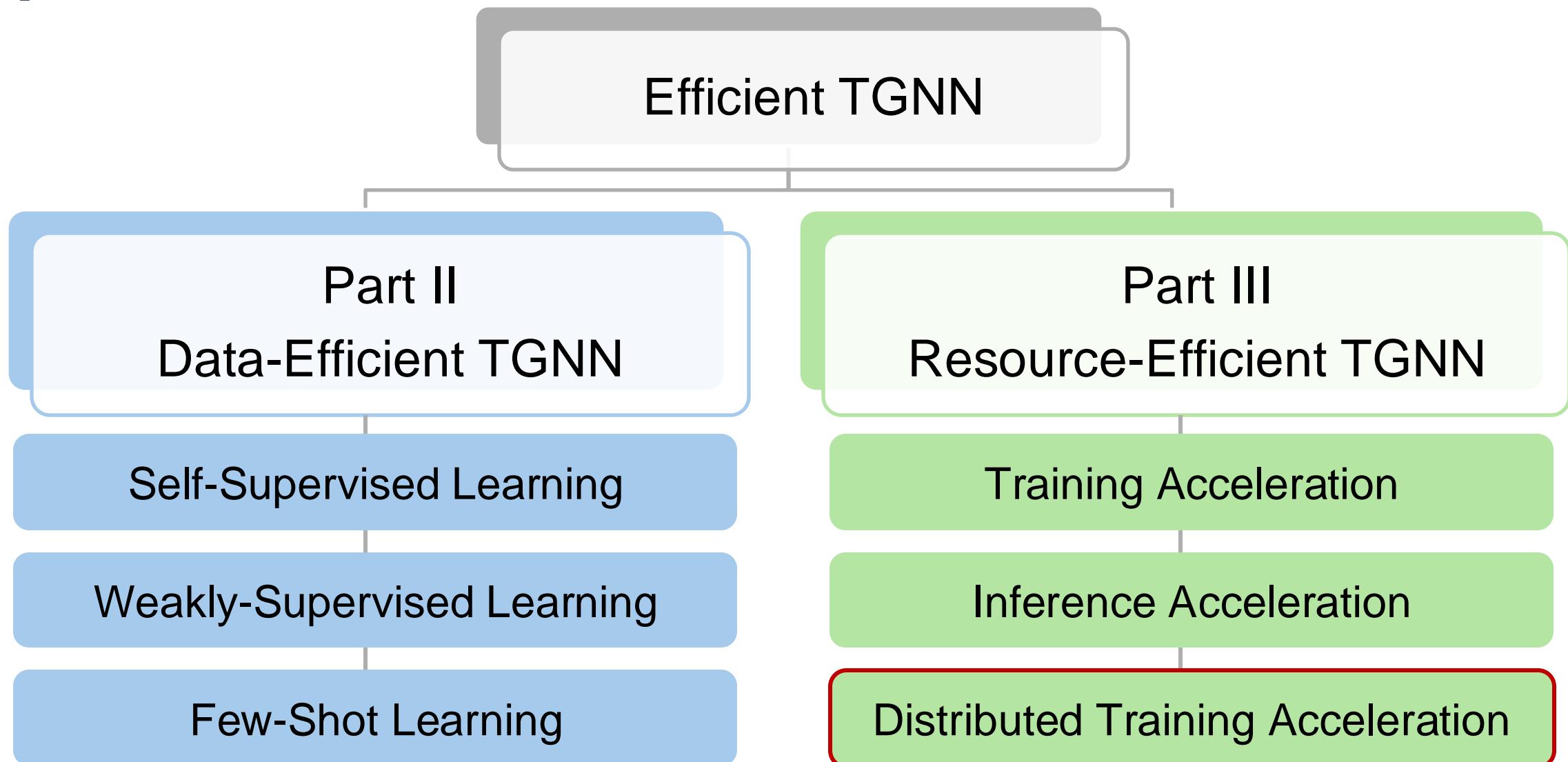
Training acceleration

- ❑ Introduction & Background
- ❑ Parallel neighbor sampler
- ❑ Data access optimization
- ❑ Model computation deduplication

Inference acceleration

- ❑ Introduction & Background
- ❑ Embedding memoization
- ❑ Time encoding precomputation

Scope of This Tutorial



Distributed Training Acceleration

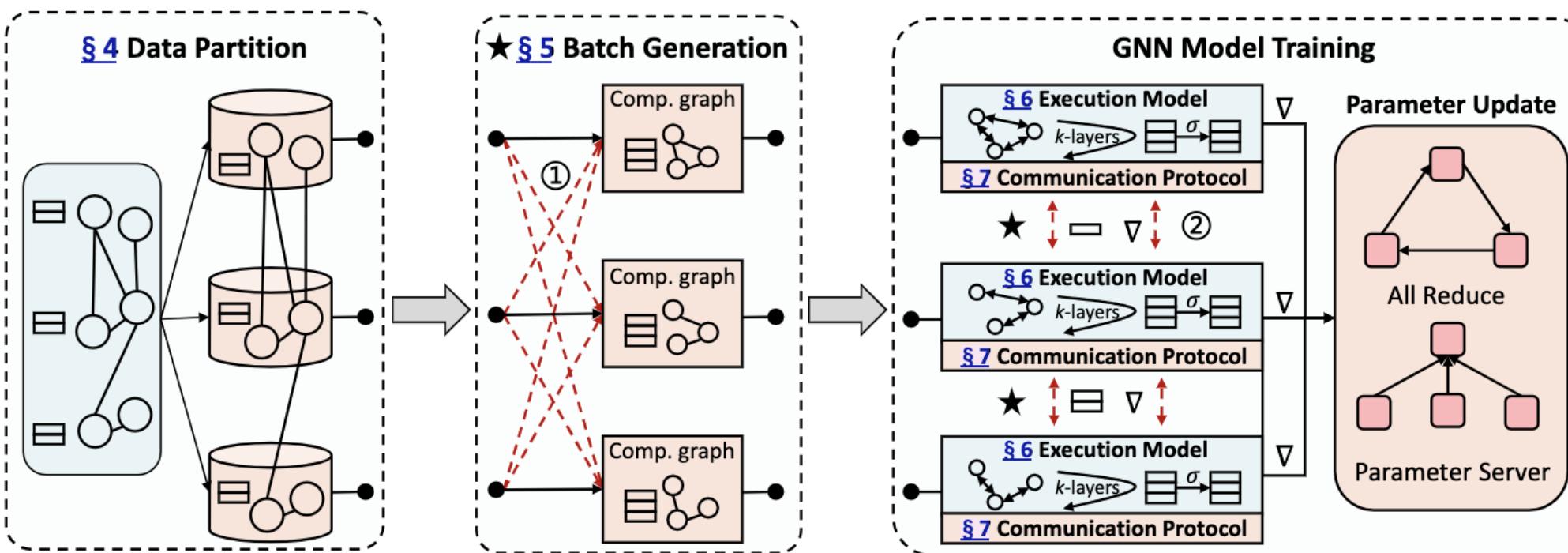
- **Introduction & Background**
- **Distributed Data Storage**
- **Distributed Sampling**
- **Data Parallel Training**

Distributed Training Acceleration

- **Introduction & Background**
- **Distributed Data Storage**
- **Distributed Sampling**
- **Data Parallel Training**

Background: Distributed GNN Training

- **Graph partition:** graph data exceeds a single machine's RAM capacity.
- **Data parallelism:** Data parallel training splits the input data across multiple devices, processes it in parallel on separate model replicas, and combines the gradients to update a shared model.

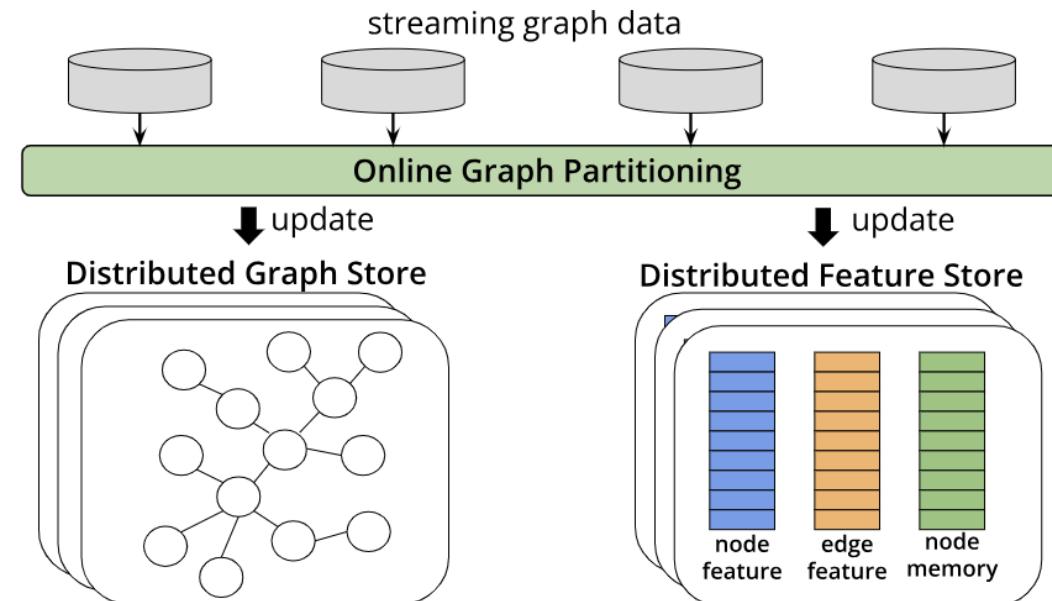
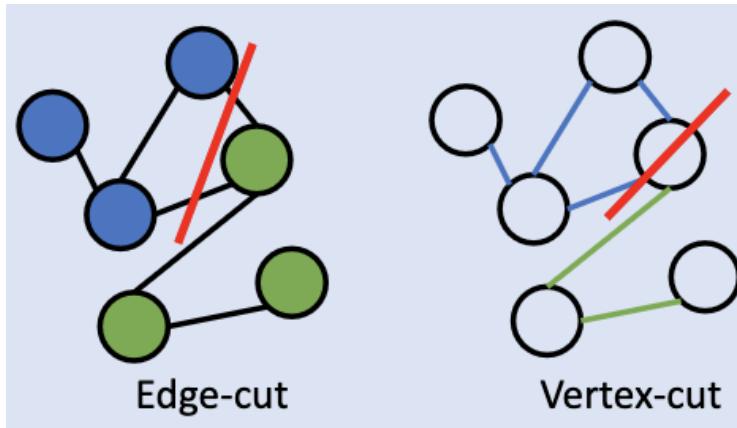


Distributed Training Acceleration

- **Introduction & Background**
- **Distributed Data Storage**
- **Distributed Sampling**
- **Data Parallel Training**

Distributed Data Storage: GNNFlow

- Edge-cut model
 - Less cross-machine communication compared to node-cut model
 - Existing edge-cut method like METIS is not suitable for temporal graphs due to high recomputation cost.
- Online hash partitioning: $\text{hash}(n) \% P$, n is node id, P is number of machines

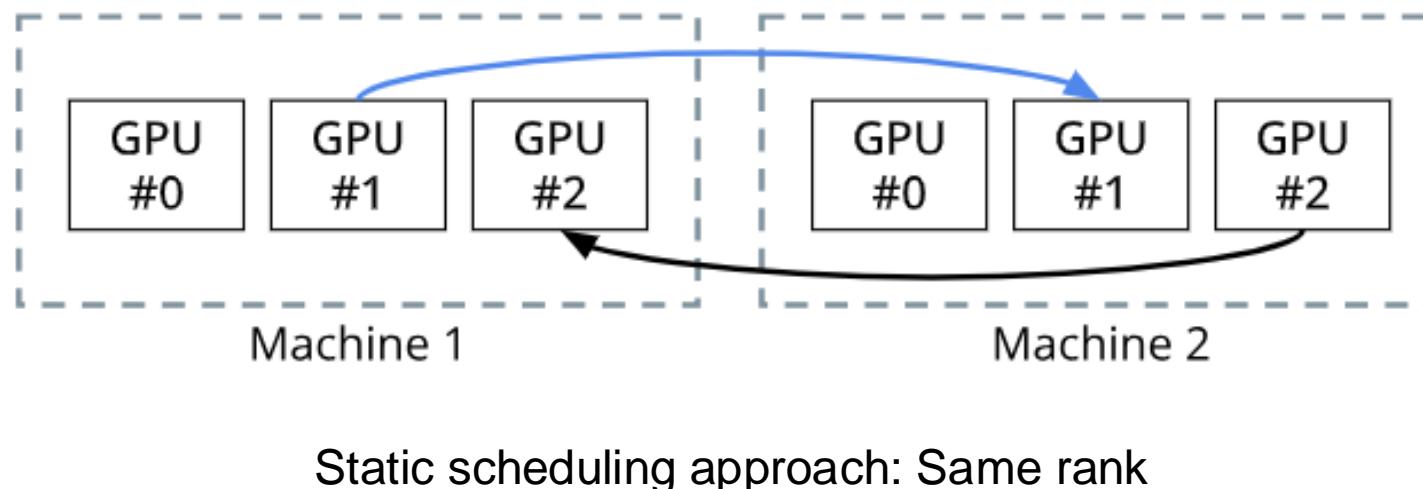


Distributed Training Acceleration

- **Introduction & Background**
- **Distributed Data Storage**
- **Distributed Sampling**
- **Data Parallel Training**

Distributed Sampling: GNNFlow

- Workload balance
 - Remote neighborhood sampling
 - Static scheduling approach
- Empirically validation
 - Average coefficient of variance of sampling times
 - <0.06



Distributed Training Acceleration

- **Introduction & Background**
- **Distributed Data Storage**
- **Distributed Sampling**
- **Data Parallel Training**

Data Parallel Training: DistTGL

- Epoch parallelism
- Memory parallelism

(a) Mini-batch Parallelism

	P0	P1	P2
Itr. 0	$i \text{ R}$	$i + 1 \text{ W}$	$i + 2 \text{ R}$
Itr. 1	$i + 3 \text{ R}$	$i + 4 \text{ R}$	$i + 5 \text{ W}$
Itr. 2	$i + 6 \text{ R}$	$i + 7 \text{ R}$	$i + 8 \text{ R}$
Itr. 3	$i + 9 \text{ R}$	$i + 10 \text{ R}$	$i + 11 \text{ R}$

(b) Epoch Parallelism

	P0	P1	P2
Itr. 0	$i \text{ R}$	$i - 1$	$i - 2$
Itr. 1	$i + 1 \text{ R}$	i	$i - 1$
Itr. 2	$i + 2 \text{ R}$	$i + 1$	i
Itr. 3	$i + 3 \text{ R}$	$i + 2$	$i + 1$

reorder

	P0	P1	P2
	$i \text{ R}$	$i - 2$	$i - 1$
	i	$i + 1 \text{ R}$	$i - 1$
	i	$i + 1$	$i + 2 \text{ R}$
	$i + 3 \text{ R}$	$i + 1$	$i + 2$

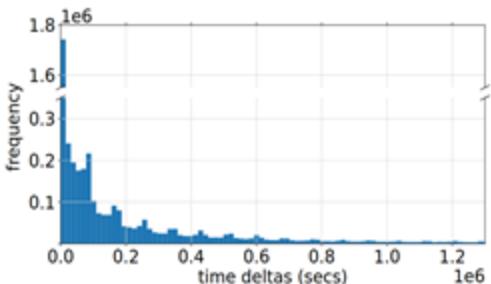
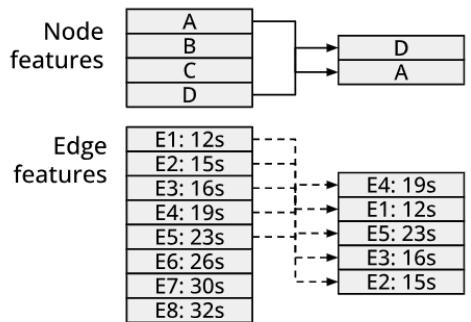
(c) Memory Parallelism

	P0	P1	P2
Itr. 0	0 R	2 R	4 R
Itr. 1	1 R	3 R	5 R
Itr. 2	0 R	2 R	4 R
Itr. 3	1 R	3 R	5 R

reorder

	P0	P1	P2
	0 R	2 R	4 R
	1 R	3 R	5 R
	2 R	4 R	0 R
	3 R	5 R	1 R

Part III Summary: Resource-Efficient TGNN



Training acceleration

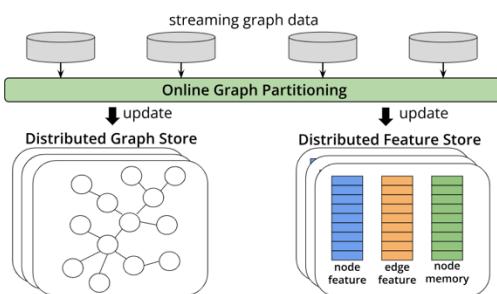
- ❑ Introduction & Background
- ❑ Parallel neighbor sampler
- ❑ Data access optimization
- ❑ Model computation deduplication

Inference acceleration

- ❑ Introduction & Background
- ❑ Embedding memoization
- ❑ Time encoding precomputation

Distributed acceleration

- ❑ Introduction & Background
- ❑ Distributed data storage
- ❑ Distributed sampling
- ❑ Data-parallel training



TGNN Frameworks

- TGL^[1]: Lacks programming interface and continuous-time optimizations
- TGLite^[2]: Ready-to-use library

```

import torch
import tglite as tg

class TGAT(torch.nn.Module):
    def __init__(self, ctx: tg.TContext, ...):
        super().__init__()  https://github.com/amazon-science/tgl
        self.ctx = ctx
        self.sampler = tg.TSampler(20, strategy='recent')
        ...

    def forward(self, batch: tg.TBatch):
        ...

graph = tg.TGraph(edges, times, ...)
ctx = tg.TContext(graph)
model = TGAT(ctx, ...)

```

TGLite API

CONFIG.YML

```

memory:
    - type: 'none'
      dim_out: 0
gnn:
    - arch: 'transformer_attention'
      layer: 2
      att_head: 2
      dim_time: 100
      dim_out: 100

if memory_param['type'] == 'node':
    if memory_param['memory_update'] == 'gru':
        self.memory_updater = GRUMemoryUpdater(memory_param, 2 * memory_pa
    elif memory_param['memory_update'] == 'rnn':
        self.memory_updater = RNNMemoryUpdater(memory_param, 2 * memory_pa
    elif memory_param['memory_update'] == 'transformer':
        self.memory_updater = TransformerMemoryUpdater(memory_param, 2 * me
    else:
        raise NotImplementedError

```

TGL API

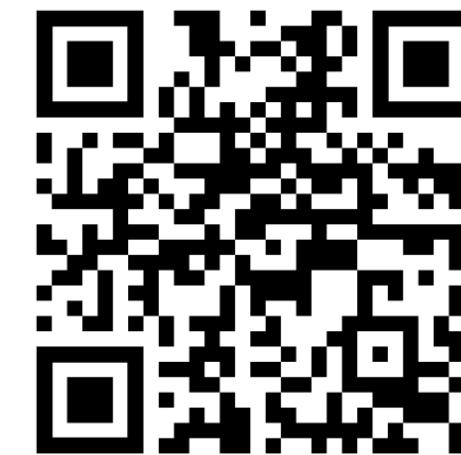
[1] <https://github.com/amazon-science/tgl>
[2] <https://github.com/ADAPT-uiuc/tglite>

TGLite Demo

<https://github.com/ADAPT-uiuc/tglite>



<https://tglite.readthedocs.io>

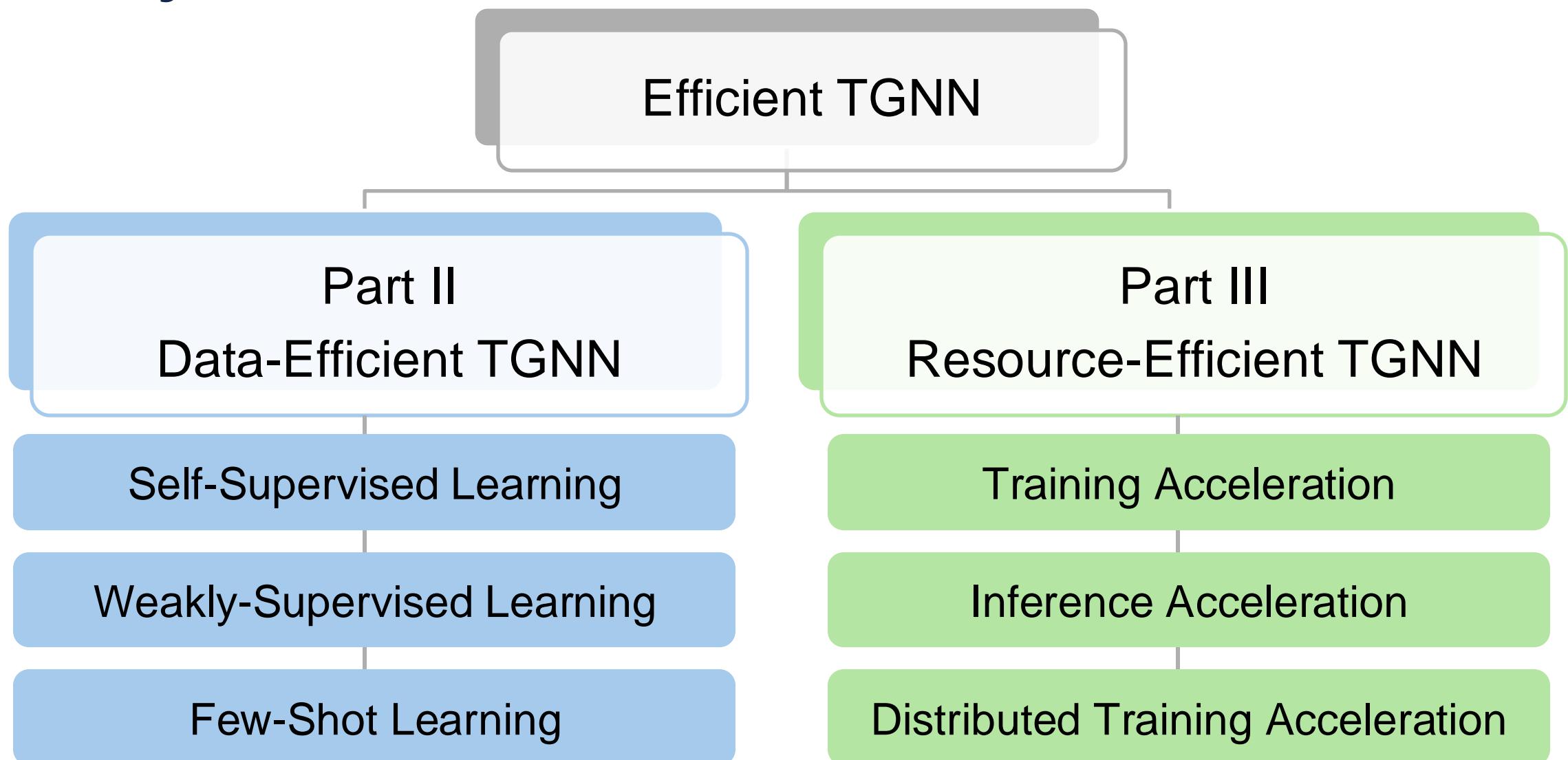


Q & A

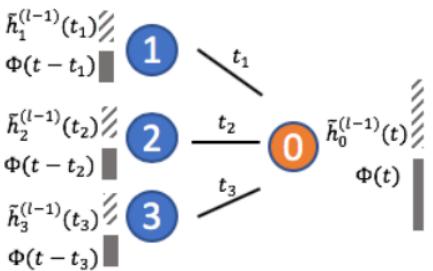
Contents

- **Part I – Introduction**
- **Part II – Data-Efficient Temporal Graph Neural Network**
- **30-min Coffee Break**
- **Part III – Resource-Efficient Temporal Graph Neural Network**
- **Part IV – Discussion and Future Directions**

Summary of This Tutorial

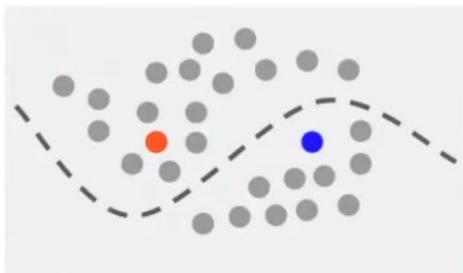


Part II Summary: Data-Efficient TGNN



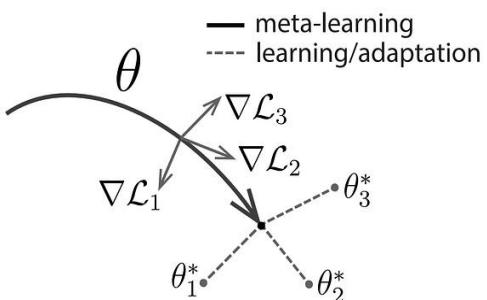
Self-Supervised Learning

- Introduction & Background
- Reconstruction
- Contrastive Learning
- Multiview Approach



Weakly-Supervised Learning

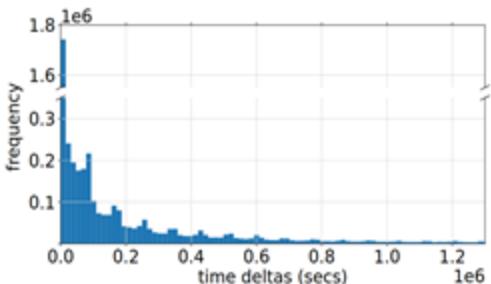
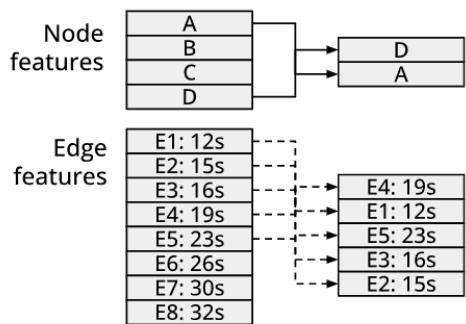
- Introduction & Background
- Limited Labeled Data
- Sparse Temporal Graph



Few-Shot Learning

- Introduction & Background
- New node adaptation
- New time adaptation
- New class adaptation

Part III Summary: Resource-Efficient TGNN



Training acceleration

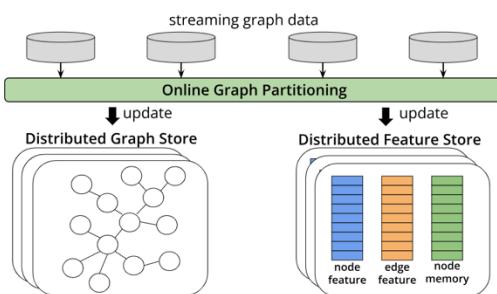
- Introduction & Background
- Parallel neighbor sampler
- Data access optimization
- Model computation deduplication

Inference acceleration

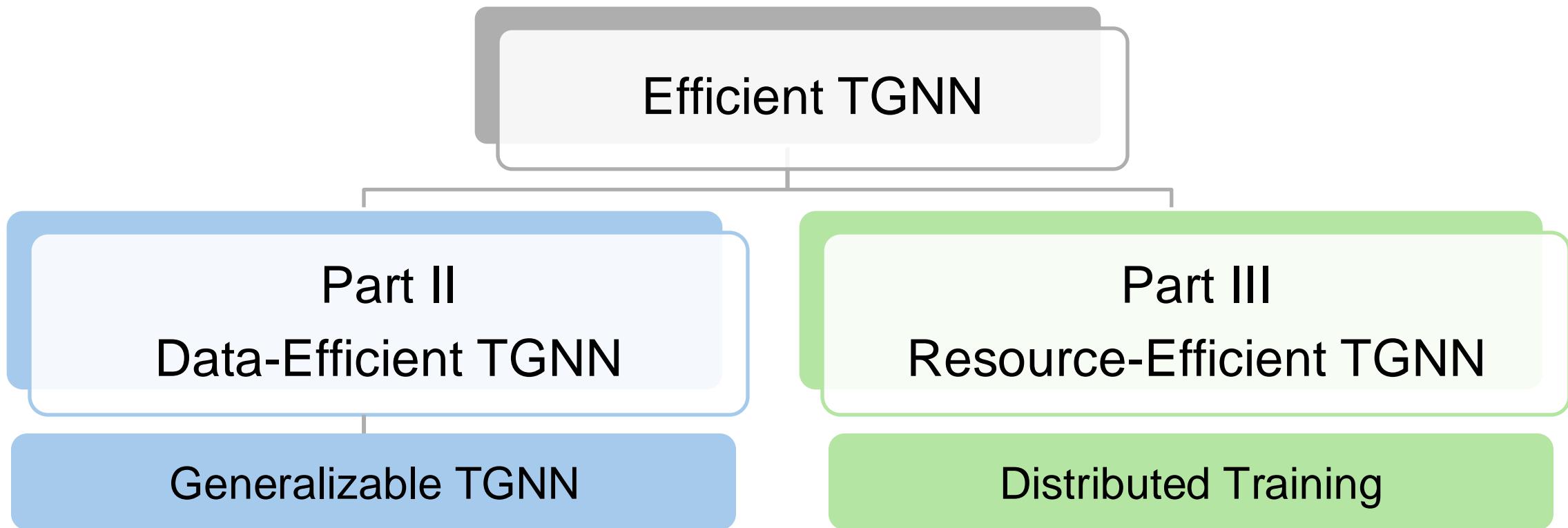
- Introduction & Background
- Embedding memoization
- Time encoding precomputation

Distributed acceleration

- Introduction & Background
- Distributed data storage
- Distributed sampling
- Data-parallel training



Future Trends

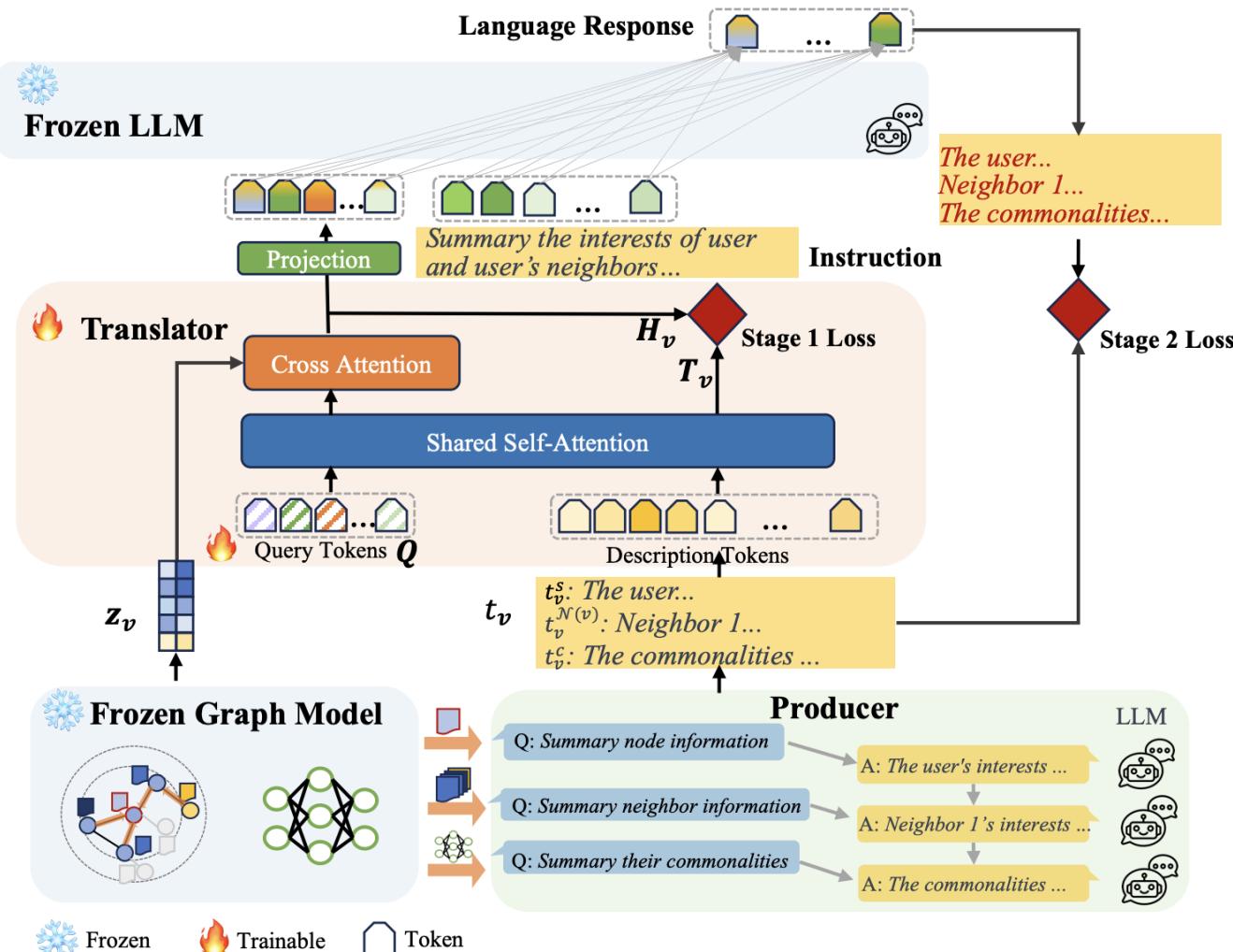


Generalizable TGNNs: LLM Solutions

- **Objective:** Borrow LLMs' generalizability to eliminate dependency on task-related labels.

- **Challenges:**
 - Time token formulation
 - Repeated LLM fine-tuning in different time

- **Future Directions:**
 - Align TGNNs with LLMs
 - Lightweight TGNN-based adapter



Generalizable TGNNs: Graph Foundation Model Solutions

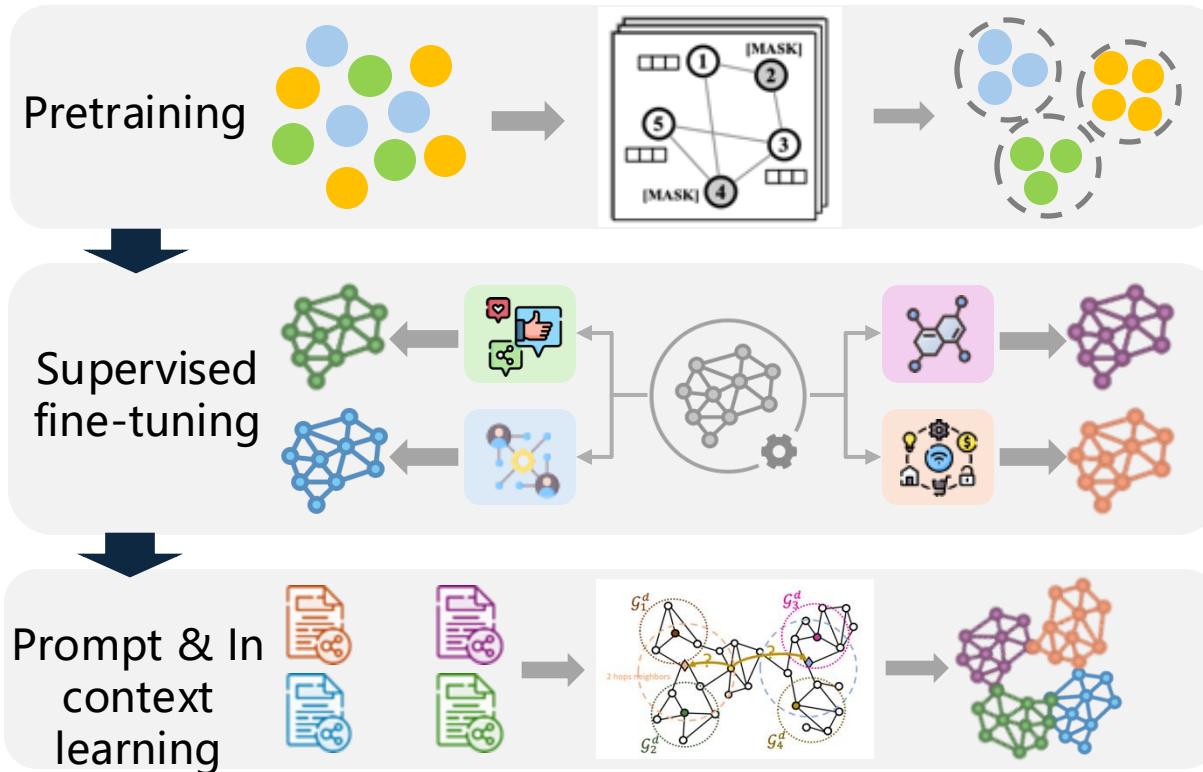
- **Objective:** Fine-tune GFM to eliminate dependency on task-related labels.

- **Challenges:**

- Lack of unified pretraining objective
- Homophily vs. heterophily

- **Future Directions:**

- Unify pretraining objective via temporal subgraph relevance prediction.
- Adaptive TGNNs on homophily and heterophily graphs.



Distributed Training: Adaptive Graph Partition

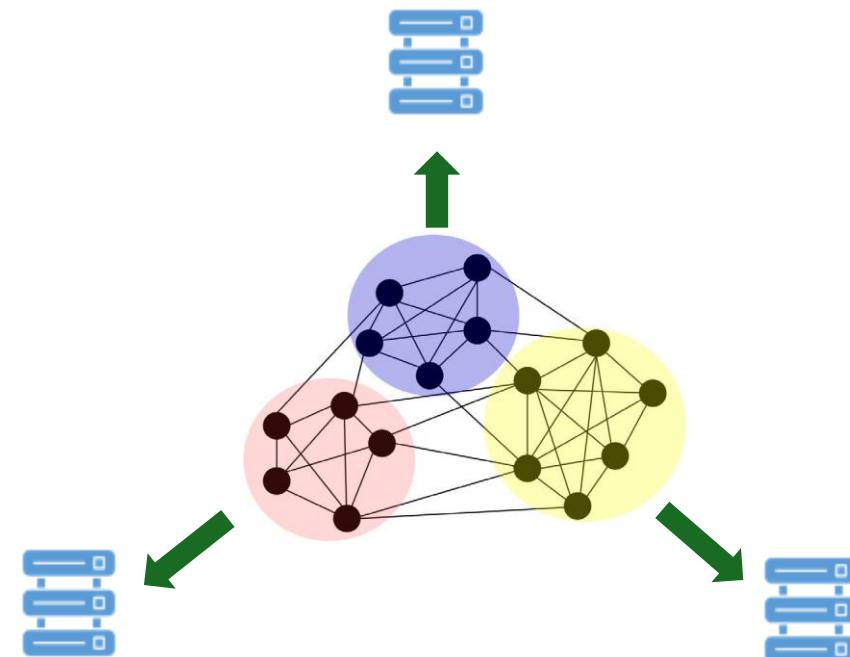
- ❑ **Objective:** Partition large-scale graphs into distributed nodes according to node conditions.

- ❑ **Challenges:**

- ❑ Chronological order v.s. graph partition
 - ❑ Load balance

- ❑ **Future Directions:**

- ❑ Adaptive graph partition in online setting
 - ❑ Dynamic load balance mechanism



Distributed Training: Heterogeneous Devices

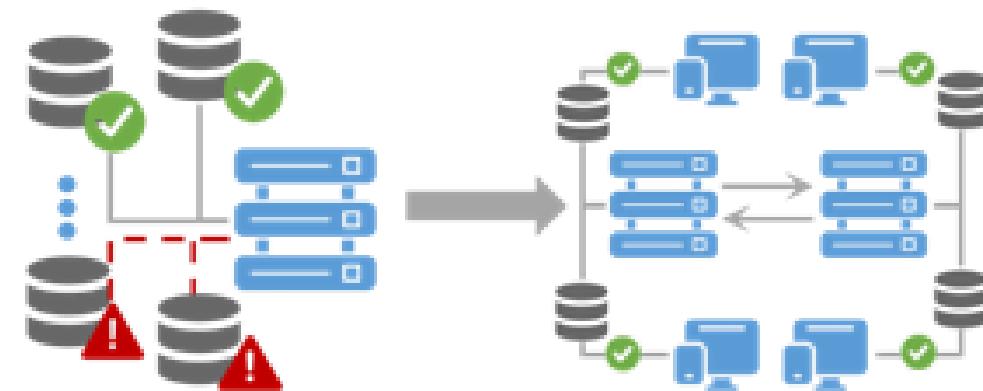
- ❑ **Objective:** Enable distributed training on a set of heterogeneous devices.

- ❑ **Challenges:**

- ❑ Network throughput bottleneck
- ❑ Privacy concern

- ❑ **Future Directions:**

- ❑ On-device offloading of TGNNs
- ❑ Federated TGNN training frameworks



Q & A