Reading Group

Zheng Huang

ROLAND: Graph Learning Framework for Dynamic Graphs

Jiaxuan You
jiaxuan@cs.stanford.edu
Stanford University
California, USA

Tianyu Du
tianyudu@stanford.edu
Stanford University
California, USA

Jure Leskovec
jure@cs.stanford.edu
Stanford University
California, USA

KDD 2022

Contents

- Background & Intro.
- Proposed Method
- Experiments
- Takeaways

Background

- The problem of learning from dynamic graphs arises in many domains such as fraud detection, anti-money laundering and recommender system.

- Unlike static graphs, dynamic graphs evolve over time, presenting unique challenges
  - Model design
  - Evaluation settings
  - Training strategies

Limitations of Model Design

Existing approaches cannot transfer successful static GNN designs to dynamic graph applications

Existing work

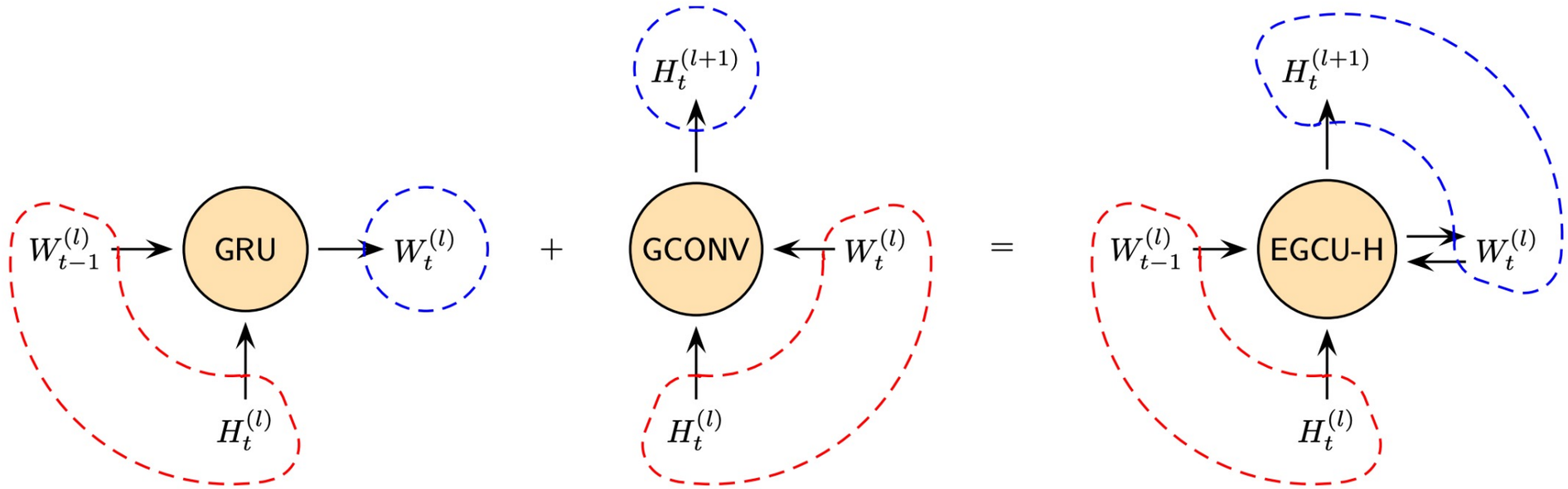
- Treat a GNN as a feature encoder and then build a sequence model on top of the GNN
- Replace the linear layers in the RNN with graph convolution layers

SOTA designs from static GNNs have not been explored for dynamic settings

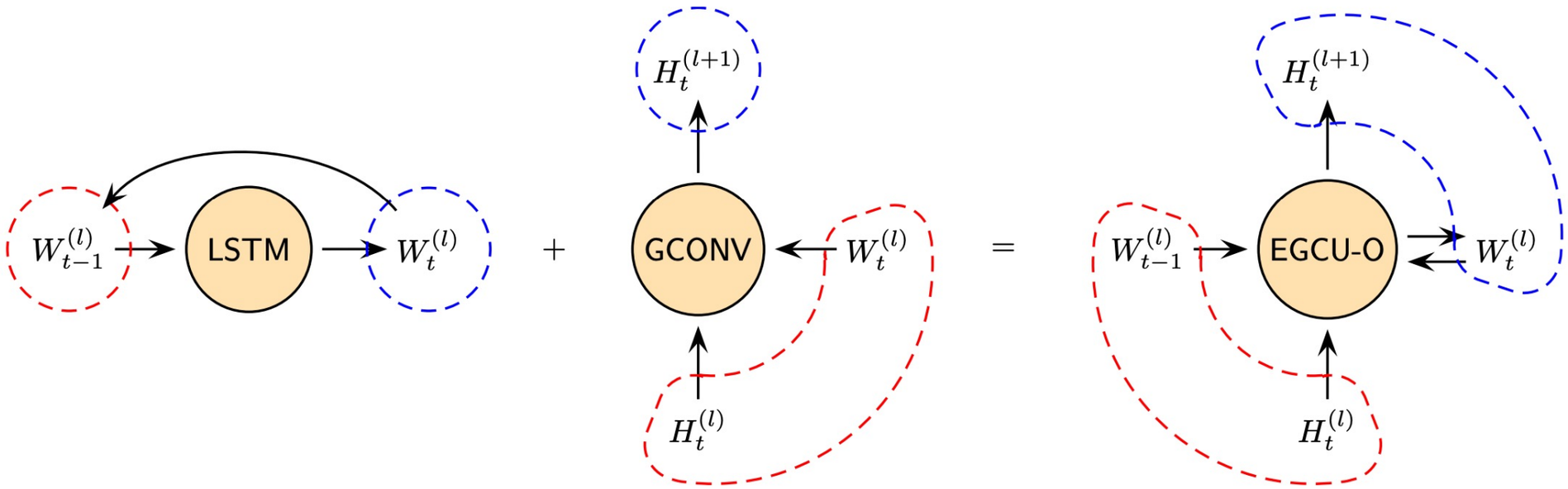
- Skip-connections
- batch normalization

Limitations of Model Design

- Existing work (Build a sequence model on top of the GNN)



(a) EvolveGCN-H, where the GCN parameters are hidden states of a recurrent architecture that takes node embeddings as input.



(b) EvolveGCN-O, where the GCN parameters are input/outputs of a recurrent architecture.

Limitations of Evaluation Settings

Existing works ignore the evolving nature of data and models

- The first eight months of data will be used as training
- 9th month as validation
- Last month as the test set

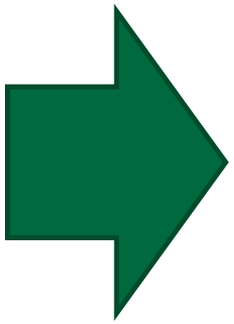
A non-updated model is used for evaluation which means the model gets stale over time

Limitations of Training Strategies

- Most existing training methods for dynamic GNNs usually require keeping the entire or a large portion of the graph in GPU
  - Store all the historical node embeddings in GPU memory (BPTT)
- There is little research on making dynamic GNNs generalize and quickly adapt to new data

Proposed Method

- Limitations of Model Design
- Limitations of Evaluation Settings
- Limitations of Training Strategies

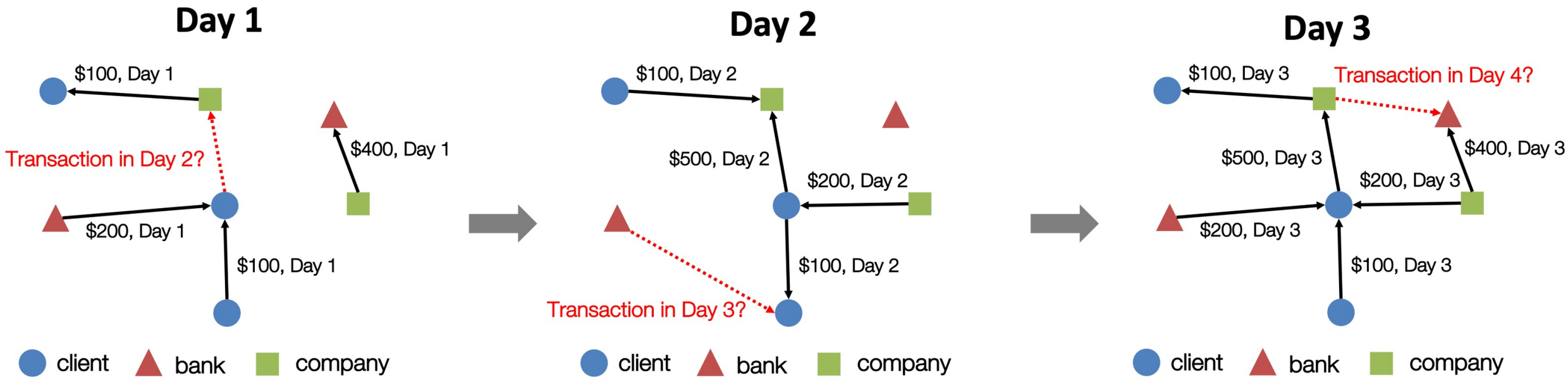


- ROLAND Model
- ROLAND Evaluation
- ROLAND Training

Proposed Method

ROLAND Model

- Any GNN for static graphs can be extended for dynamic graph use cases
- Use case (dynamic transaction graph):



Use information up to time t to predict potential edges at time t+1

Proposed Method

ROLAND Model

- Any GNN for static graphs can be extended for dynamic graph use cases
  - A new viewpoint for static GNNs
    - The node embeddings at different GNN layers are viewed as hierarchical node states.
  - To generalize a static GNN to a dynamic setting
    - Define how to update these hierarchical nodes states based on newly observed nodes and edges.
      - Update-modules: Moving average, MLP, GRU
    - Keep the design of a given static GNN and use it on a dynamic graph

ROLAND Model

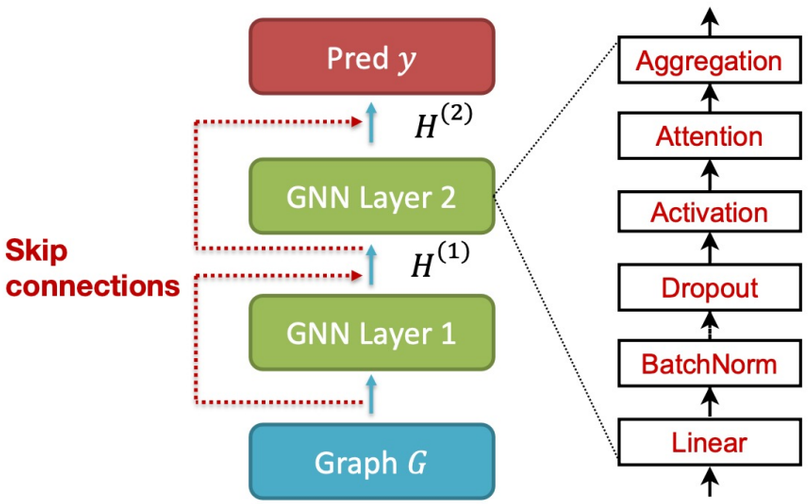
- Static graph $G = (V, E), V = \{v_1, ..., v_n\}$ is the node set and $E \subseteq V \times V$ is the edge set. $X = \{\mathbf{x}_v \mid v \in V\}$ is the node feature and $F = \{\mathbf{f}_{uv} \mid (u, v) \in E\}$ is the edge feature.
- Dynamic graphs
  - Each node v and e has a timestamp.
  - A dynamic graph can be represented as a sequence of snapshots $\mathcal{G} = \{G_t\}_{t=1}^{T}$, where each snapshot is a static graph $G_t = (V_t, E_t)$, $V_t = \{v \in V \mid \tau_v = t\}$ and $E_t = \{e \in E \mid \tau_e = t\}$

ROLAND Model

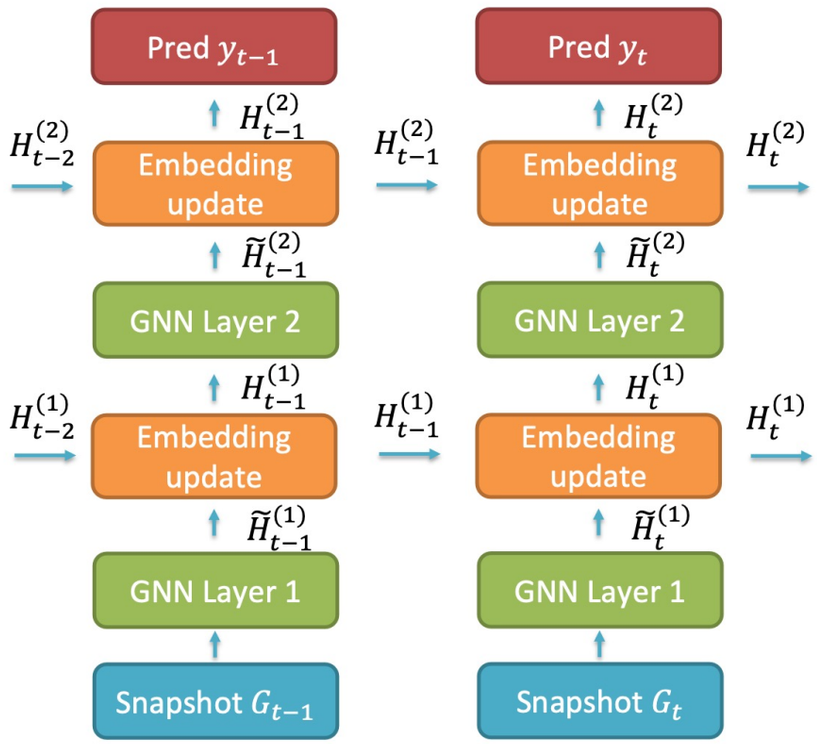
Extend H from static embeddings to dynamic node states

- View the node embeddings as hierarchical node states
  - As a result, we only need to define how to update these hierarchical nodes states based on newly observed nodes and edges
- Embedding matrix $H = \{H^{(1)}, ..., H^{(L)}\}$ captures the multi-hop node neighbor information

a) A static GNN with modern architectural design options



b) Extend static GNNs to dynamic GNNs



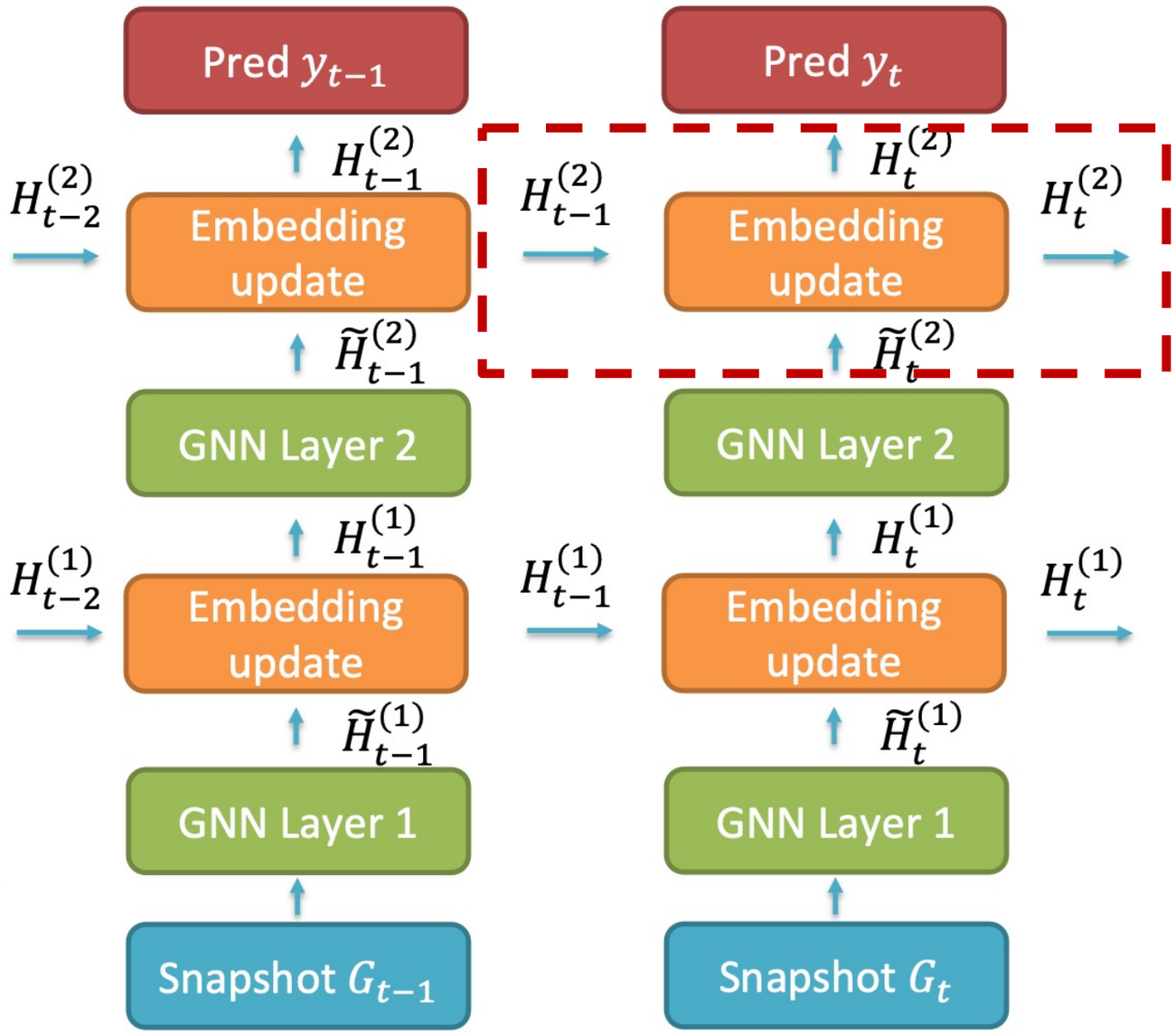
Hierarchical node state:
$H_t = \{H_t^{(1)}, \dots, H_t^{(L)}\}$

ROLAND Model

Update modules

- Update model embeddings $H_t^{(l)}$ hierarchically and dynamically
- Can be inserted to any static GNN
  - The new level L node state depends on lower layer node state and historical node state

$$H_t^{(l)} = \textsc{Update}^{(l)}(H_{t-1}^{(l)}, \tilde{H}_t^{(l)}), \quad \tilde{H}_t^{(l)} = \textsc{Gnn}^{(l)}(H_t^{(l-1)})$$



ROLAND Model

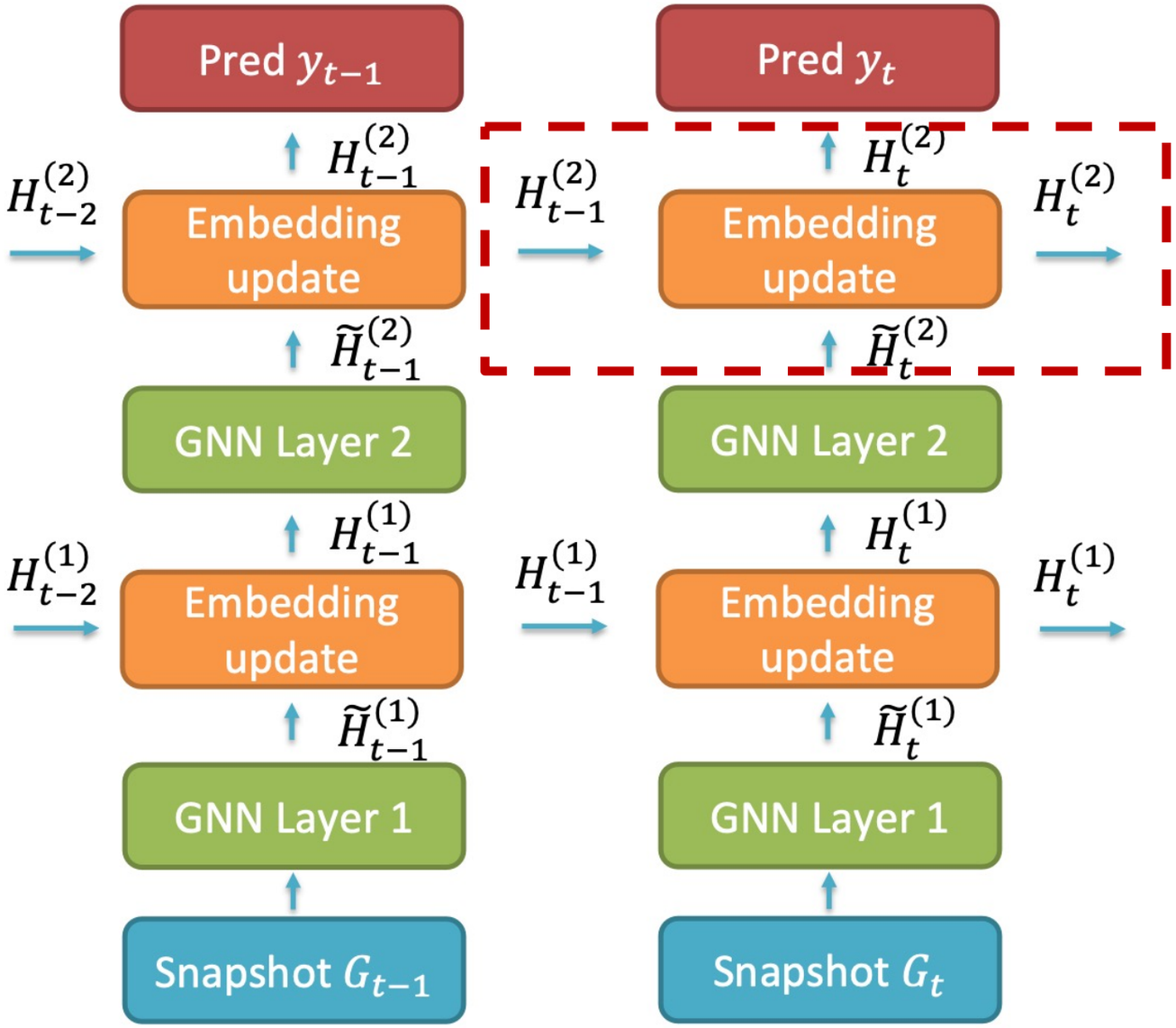
Update modules

- Three embedding update methods
- Moving average

$$H_{t,v}^{(l)} = \kappa_{t,v} H_{t-1,v}^{(l)} + (1-\kappa_{t,v}) \tilde{H}_{t,v}^{(l)}$$

$$\kappa_{t,v} = \frac{\sum_{\tau=1}^{t-1} |E_\tau|}{\sum_{\tau=1}^{t-1} |E_\tau| + |E_t|} \in [0, 1]$$

  - Capture dynamics of embedding and free from trainable parameters



- MLP

$$H_t^{(l)} = \textsc{Mlp}(\textsc{Concat}(H_{t-1}^{(l)}, \tilde{H}_t^{(l-1)}))$$

- GRU

$$H_t^{(l)} = \textsc{Gru}(H_{t-1}^{(l)}, \tilde{H}_t^{(l)})$$

ROLAND Model

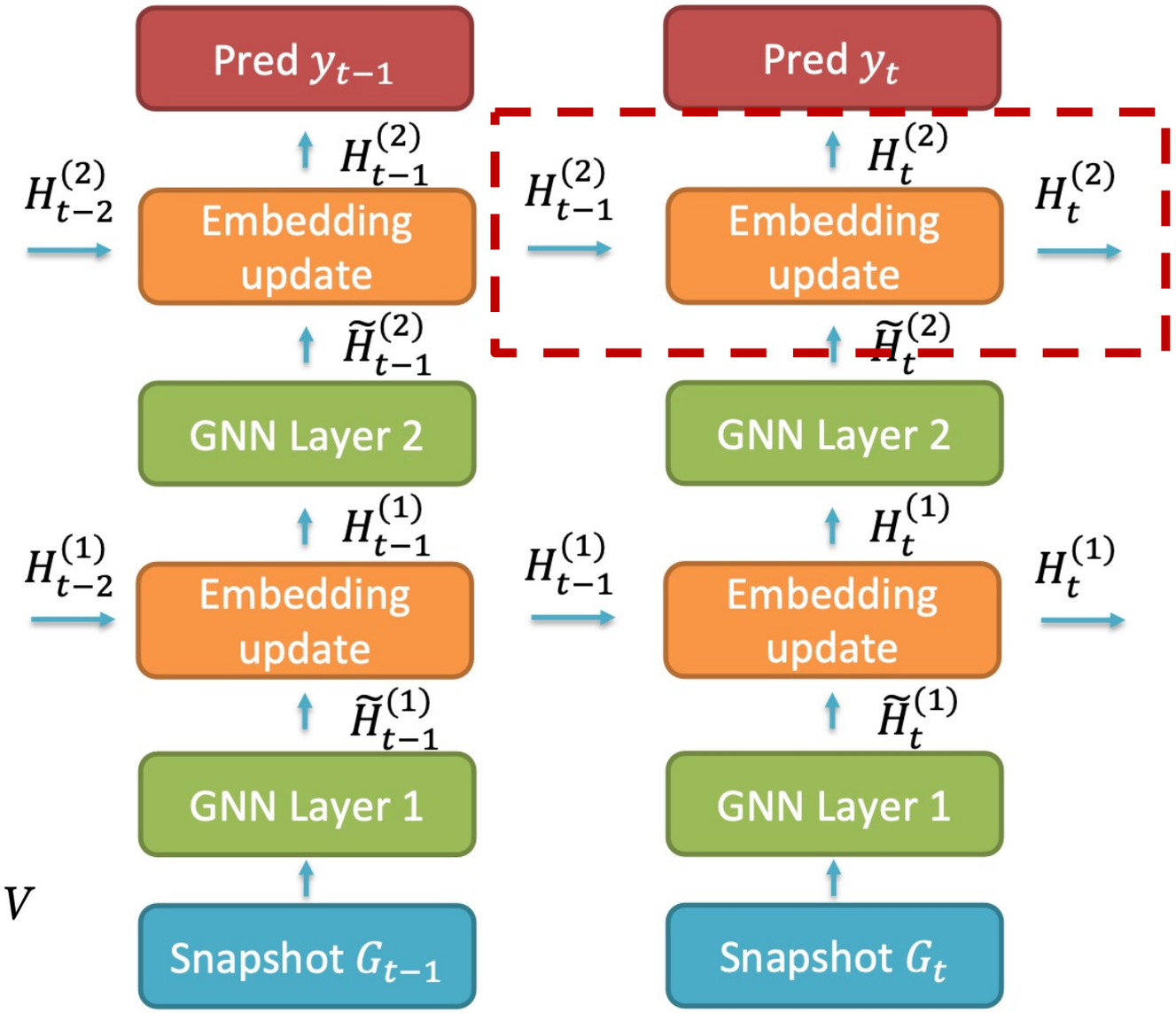
- GNN Design

$$\mathbf{m}_{u\to v}^{(l)} = \mathbf{W}^{(l)}\,\textsc{Concat}\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}, \mathbf{f}_{uv}\right),$$

$$\mathbf{h}_v^{(l)} = \textsc{Agg}^{(l)}\left(\{\mathbf{m}_{u\to v}^{(l)} \mid u \in \mathcal{N}(v)\}\right) + \mathbf{h}_v^{(l-1)}$$

- Based on the new node embeddings, ROLAND predicts the probability of a future edge from node u to v by an MLP

$$y_t = \textsc{Mlp}(\textsc{Concat}(\mathbf{h}_{u,t}^{(L)}, \mathbf{h}_{v,t}^{(L)})), \forall (u,v) \in V \times V$$



ROLAND Evaluation

Live-update evaluation setting for dynamic graphs

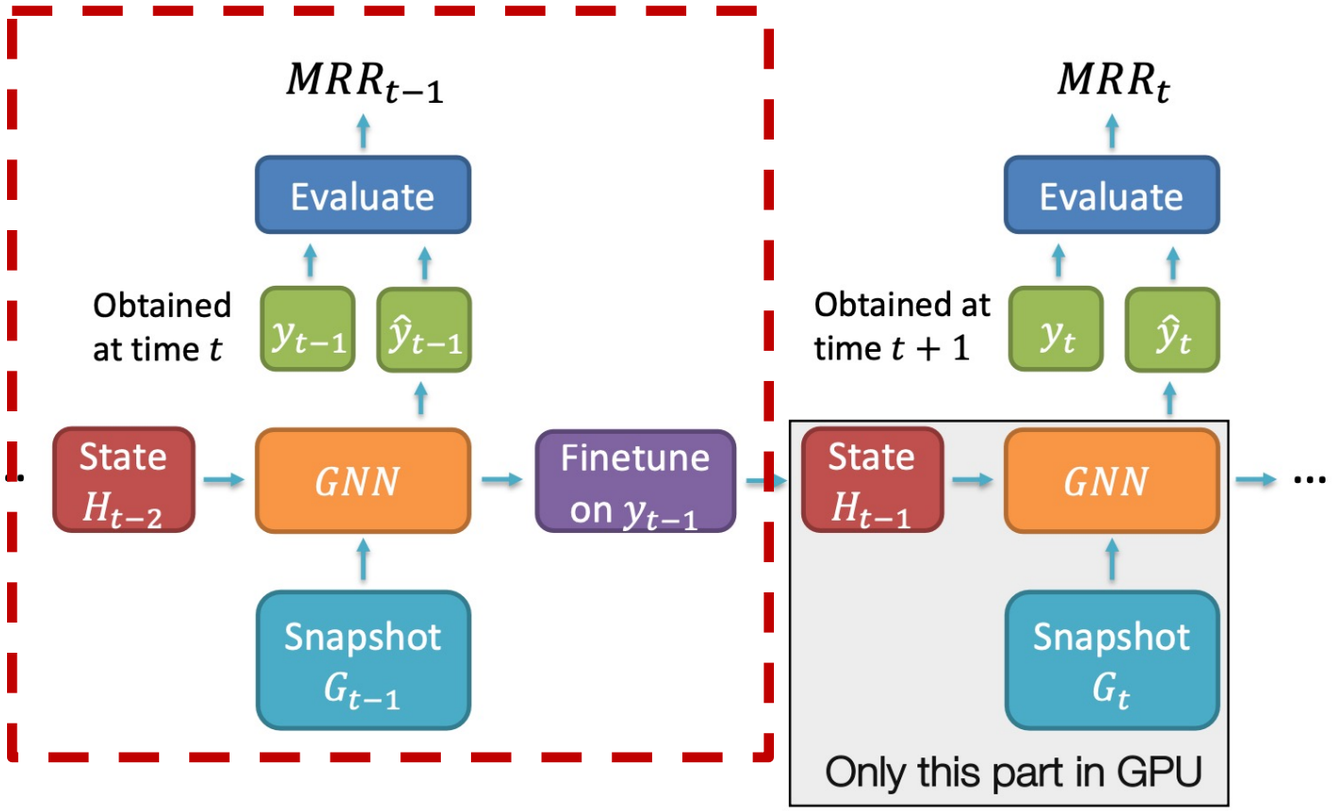
- Existing works ignore the evolving nature of data and models
- Model to be fine-tuned using the new graph snapshot
- Mimicking real-world use cases, in which the model is constantly updated to fit evolving data

ROLAND Evaluation

Live-update evaluation setting for dynamic graphs

(1) Fine-tune GNN model using newly observed label y_{t-1}

(2) Making predictions using historical state and current snapshot



Algorithm 2 ROLAND live-update evaluation

Input: Dynamic graph $\mathcal{G} = \{G_1, \ldots, G_T\}$, link prediction labels $y_1, \ldots, y_T$, number of snapshots T, GNN$(\cdot)$ defined in Algorithm 1
Output: Performance MRR, model GNN

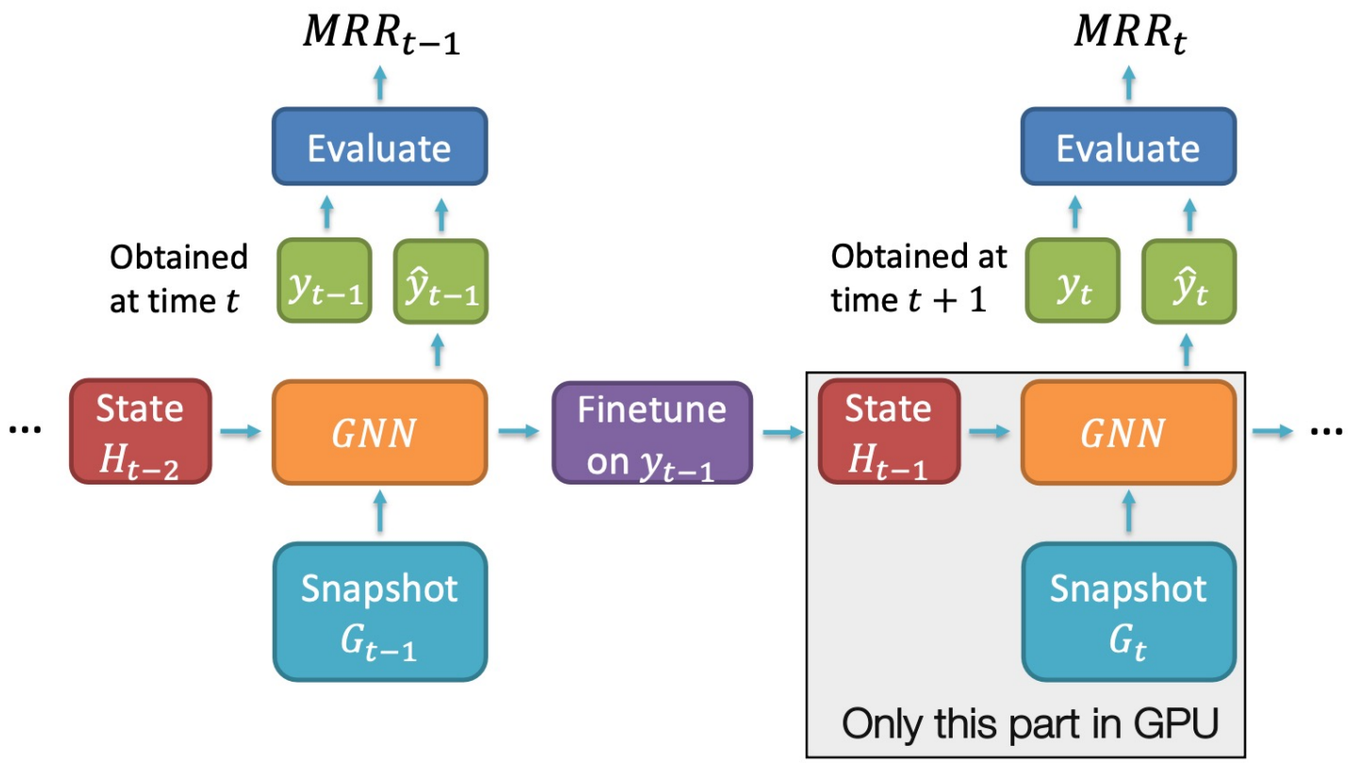
Initialize hierarchical node state H_0
for $t = 2, \ldots, T$ **do**
  Collect link prediction labels $y_{t-1} = y_{t-1}^{(train)} \cup y_{t-1}^{(val)}, y_t$
  while $\text{MRR}_{t-1}^{(val)}$ is increasing **do**
    $H_{t-1}, \hat{y}_{t-1} \leftarrow \text{GNN}(G_{t-1}, H_{t-2})$, $\hat{y}_{t-1} = \hat{y}_{t-1}^{(train)} \cup \hat{y}_{t-1}^{(val)}$
    Update GNN via backprop based on $\hat{y}_{t-1}^{(train)}, y_{t-1}^{(train)}$
    $\text{MRR}_{t-1}^{(val)} \leftarrow \text{EVALUATE}(\hat{y}_{t-1}^{(val)}, y_{t-1}^{(val)})$
  $H_t, \hat{y}_t \leftarrow \text{GNN}(G_t, H_{t-1})$
  $\text{MRR}_t \leftarrow \text{EVALUATE}(\hat{y}_t, y_t)$
$\text{MRR} = \sum_{t=2}^{T} \text{MRR}_t / (T-1)$

ROLAND Training

A truncated version of back-propagation-through-time (BPTT) based on the evaluation method

Only keep the incoming new graph snapshot and historical node states in GPU

We can train GNNs on a dynamic transaction network with 56 million edges (13 times larger than



ROLAND Training

- A truncated version of back-propagation-through-time (BPTT)
  - Only keep the incoming new graph snapshot and historical node states in GPU
    - We can train GNNs on a dynamic transaction network with 56 million edges (13 times larger than existing benchmarks in terms of edges per snapshot)
- Formulate prediction tasks over dynamic graphs as a meta-learning problem
  - Treat making predictions in different periods (e.g., every day) as different tasks arriving sequentially
  - Find a meta-model, which serves as a good initialization that is used to derive a specialized model for future prediction tasks quickly

ROLAND Training

- Initialize GNN with $\text{GNN}^{(meta)}$
- Use back-propagation with early stopping to fine-tune the model for the next prediction task
- We updates the meta-model by computing the moving average of the trained models

$$(1-\alpha)\text{GNN}^{(meta)} + \alpha\text{GNN}$$

- $\alpha \in [0, 1]$ is the smoothing factor

Algorithm 3 ROLAND training algorithm

Input: Graph snapshot G_t, link prediction label y_t, hierarchical node state H_{t-1}, smoothing factor α, meta-model $\text{GNN}^{(meta)}$
Output: Model GNN, updated meta-model $\text{GNN}^{(meta)}$

1: $\text{GNN} \leftarrow \text{GNN}^{(meta)}$
2: Move GNN, G_t, H_{t-1} to GPU
3: **while** $\text{MRR}_t^{(val)}$ is increasing **do**
4: $\quad H_t, \hat{y}_t \leftarrow \text{GNN}(G_t, H_{t-1}), \hat{y}_t = \hat{y}_t^{(train)} \cup \hat{y}_t^{(val)}$
5: $\quad$ Update GNN via backprop based on $\hat{y}_t^{(train)}, y_t^{(train)}$
6: $\quad \text{MRR}_t^{(val)} \leftarrow \text{EVALUATE}(\hat{y}_t^{(val)}, y_t^{(val)})$
7: Remove GNN, G_t, H_{t-1} from GPU
8: $\text{GNN}^{(meta)} \leftarrow (1-\alpha)\text{GNN}^{(meta)} + \alpha\text{GNN}$

Experiments

Dataset

(1) BSI-ZK and (2) BSI-SVT: Financial transactions among companies

(3) Bitcon-OTC and (4) Bitcon-Alpha: Who-trusts-whom networks of people who trade on the OTC/Alpha platform

(5) UCI-Message: Private messages sent on an online social network system among students

(6) Reddit-Title and (7) Reddit-Body: Networks of hyperlinks in titles and bodies of Reddit posts, respectively. Each hyperlink represents a directed edge between two subreddits

(8) AS-733: The Autonomous systems dataset of traffic flows among routers

Experiments

Dataset statistics

	# Edges	# Nodes	Range	Snapshot Frequency	# Snapshots
BSI-ZK	56,194,191	1,744,561	Jan 01, 2008 - Dec 30, 2008	daily	257
AS-733	11,965,533	7,716	Nov 8, 1997 - Jan 2, 2000	daily	733
Reddit-Title	571,927	54,075	Dec 31, 2013 - Apr 30, 2017	weekly	178
Reddit-Body	286,561	35,776	Dec 31, 2013 - Apr 30, 2017	weekly	178
BSI-SVT	190,133	89,564	Jan 27, 2008 - Dec 30, 2008	weekly	49
UCI-Message	59,835	1,899	Apr 15, 2004 - Oct 26, 2004	weekly	29
Bitcoin-OTC	35,592	5,881	Nov 8, 2010 - Jan 24, 2016	weekly	279
Bitcoin-Alpha	24,186	3,783	Nov 7, 2010 - Jan 21, 2016	weekly	274

Experiments

Task: The ROLAND framework is evaluated over the future link prediction

- At each time t, the model utilizes information accumulated up to time t to predict edges in snapshot t + 1.
- For each node u with positive edge (u, v) at t + 1, we randomly sample 1000 negative edges emitting from u and identify the rank of edge (u, v)'s prediction score among all other negative edges based on mean reciprocal rank (MRR)

Experiments

Baselines

(1) EvolveGCN-H and (2) EvolveGCN-O utilizes an RNN to dynamically update weights of internal GNNs, which allows the GNN model to change during the test time

(3) T-GCN uses a GNN into the GRU by replacing linear transformations in GRU with graph convolution operators

(4) GCRN-GRU and (5) GCRN-LSTM modify TGCN by capturing temporal information using either GRU or LSTM

(6) GCRN constructs node features using a Chebyshev spectral graph convolution layer to capture spatial information; afterward, node features are fed into an LSTM cell for temporal information

Experiments

Baselines with BPTT training vs. ROLAND

By default, the baseline models are trained with BPTT, which requires storing all the historical node embeddings in GPU. This training strategy cannot scale to large graphs.

	BSI-ZK	AS-733	Reddit-Title	Reddit-Body	BSI-SVT	UCI-Message	Bitcoin-OTC	Bitcoin-Alpha
Baseline Models with standard training								
EvolveGCN-H	N/A, OOM	N/A, OOM	N/A, OOM	**0.148 ± 0.013**	**0.031 ± 0.016**	0.061 ± 0.040	0.067 ± 0.035	0.079 ± 0.032
EvolveGCN-O	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.015 ± 0.006	0.071 ± 0.009	0.085 ± 0.022	0.071 ± 0.025
GCRN-GRU	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.080 ± 0.012	N/A, OOM	N/A, OOM
GCRN-LSTM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	**0.083 ± 0.001**	N/A, OOM	N/A, OOM
GCRN-Baseline	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.069 ± 0.004	**0.152 ± 0.011**	**0.141 ± 0.005**
TGCN	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.054 ± 0.024	0.128 ± 0.049	0.088 ± 0.038
Baseline Models with ROLAND Training								
EvolveGCN-H	N/A, OOM	0.251 ± 0.079	0.165 ± 0.026	0.102 ± 0.010	0.032 ± 0.008	0.057 ± 0.012	0.076 ± 0.022	0.054 ± 0.015
EvolveGCN-O	0.396	0.163 ± 0.002	0.047 ± 0.004	0.033 ± 0.001	0.018 ± 0.003	0.066 ± 0.012	0.032 ± 0.004	0.034 ± 0.002
GCRN-GRU	N/A, OOM	**0.344 ± 0.001**	0.338 ± 0.006	0.217 ± 0.004	0.050 ± 0.004	0.089 ± 0.004	0.173 ± 0.003	0.140 ± 0.004
GCRN-LSTM	N/A, OOM	0.341 ± 0.001	0.344 ± 0.005	0.216 ± 0.000	0.051 ± 0.002	0.091 ± 0.010	0.174 ± 0.004	**0.146 ± 0.005**
GCRN-Baseline	0.754	0.336 ± 0.002	0.351 ± 0.001	0.218 ± 0.002	0.054 ± 0.002	**0.095 ± 0.008**	**0.183 ± 0.002**	0.145 ± 0.003
TGCN	**0.831**	0.343 ± 0.002	**0.391 ± 0.004**	**0.251 ± 0.001**	**0.157 ± 0.004**	0.080 ± 0.015	0.083 ± 0.011	0.069 ± 0.013
ROLAND results								
Moving Average	0.819	0.309 ± 0.011	0.362 ± 0.007	0.289 ± 0.038	0.177 ± 0.006	0.075 ± 0.006	0.120 ± 0.002	0.0962 ± 0.010
MLP-Update	0.834	0.329 ± 0.021	0.395 ± 0.006	0.291 ± 0.008	**0.217 ± 0.003**	0.103 ± 0.010	0.154 ± 0.010	0.148 ± 0.012
GRU-Update	**0.851**	**0.340** ± 0.001	**0.425** ± 0.015	**0.362** ± 0.002	0.205 ± 0.014	**0.112** ± 0.008	**0.194** ± 0.004	**0.157** ± 0.007
Improvement over the best baseline	2.40%	-1.16%	8.70%	44.22%	38.21%	17.89%	6.01%	7.53%

Experiments

Baselines with ROLAND training vs. ROLAND.

Re-implement baseline models and adapt them to be trained with ROLAND training.

	BSI-ZK	AS-733	Reddit-Title	Reddit-Body	BSI-SVT	UCI-Message	Bitcoin-OTC	Bitcoin-Alpha
Baseline Models with standard training								
EvolveGCN-H	N/A, OOM	N/A, OOM	N/A, OOM	**0.148 ± 0.013**	**0.031 ± 0.016**	0.061 ± 0.040	0.067 ± 0.035	0.079 ± 0.032
EvolveGCN-O	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.015 ± 0.006	0.071 ± 0.009	0.085 ± 0.022	0.071 ± 0.025
GCRN-GRU	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.080 ± 0.012	N/A, OOM	N/A, OOM
GCRN-LSTM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	**0.083 ± 0.001**	N/A, OOM	N/A, OOM
GCRN-Baseline	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.069 ± 0.004	**0.152 ± 0.011**	**0.141 ± 0.005**
TGCN	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	N/A, OOM	0.054 ± 0.024	0.128 ± 0.049	0.088 ± 0.038
Baseline Models with ROLAND Training								
EvolveGCN-H	N/A, OOM	0.251 ± 0.079	0.165 ± 0.026	0.102 ± 0.010	0.032 ± 0.008	0.057 ± 0.012	0.076 ± 0.022	0.054 ± 0.015
EvolveGCN-O	0.396	0.163 ± 0.002	0.047 ± 0.004	0.033 ± 0.001	0.018 ± 0.003	0.066 ± 0.012	0.032 ± 0.004	0.034 ± 0.002
GCRN-GRU	N/A, OOM	**0.344 ± 0.001**	0.338 ± 0.006	0.217 ± 0.004	0.050 ± 0.004	0.089 ± 0.004	0.173 ± 0.003	0.140 ± 0.004
GCRN-LSTM	N/A, OOM	0.341 ± 0.001	0.344 ± 0.005	0.216 ± 0.000	0.051 ± 0.002	0.091 ± 0.010	0.174 ± 0.004	**0.146 ± 0.005**
GCRN-Baseline	0.754	0.336 ± 0.002	0.351 ± 0.001	0.218 ± 0.002	0.054 ± 0.002	**0.095 ± 0.008**	**0.183 ± 0.002**	0.145 ± 0.003
TGCN	**0.831**	0.343 ± 0.002	**0.391 ± 0.004**	**0.251 ± 0.001**	**0.157 ± 0.004**	0.080 ± 0.015	0.083 ± 0.011	0.069 ± 0.013
ROLAND results								
Moving Average	0.819	0.309 ± 0.011	0.362 ± 0.007	0.289 ± 0.038	0.177 ± 0.006	0.075 ± 0.006	0.120 ± 0.002	0.0962 ± 0.010
MLP-Update	0.834	0.329 ± 0.021	0.395 ± 0.006	0.291 ± 0.008	**0.217 ± 0.003**	0.103 ± 0.010	0.154 ± 0.010	0.148 ± 0.012
GRU-Update	**0.851**	**0.340** ± 0.001	**0.425** ± 0.015	**0.362** ± 0.002	0.205 ± 0.014	**0.112** ± 0.008	**0.194** ± 0.004	**0.157** ± 0.007
Improvement over the best baseline	2.40%	-1.16%	8.70%	44.22%	38.21%	17.89%	6.01%	7.53%

Ablation Study

Effectiveness of Meta-learning

- Run 10 experiments with $\alpha \in \{0.1, 0.2, \ldots, 1.0\}$
- α = 1 corresponds to the baseline that directly uses the previous model to initialize the training of GNN
- Gain: MRR improvement from the best meta-learning setting over the non-meta-learning setting

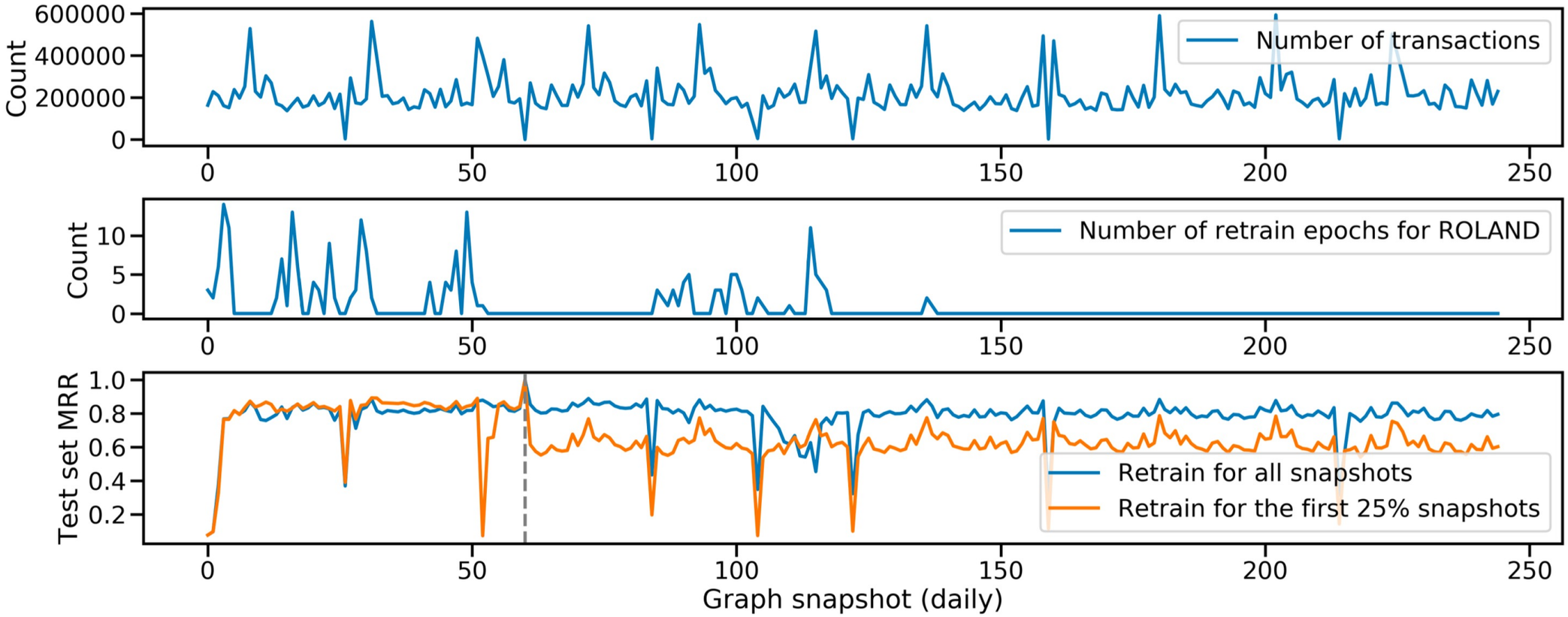
Model	BSI-ZK		AS-733		Reddit-Title		Reddit-Body		BSI-SVT		UCI-Message		Bitcoin-OTC		Bitcoin-Alpha		Average
	α	Gain	α	Gain	α	Gain	α	Gain	α	Gain	α	Gain	α	Gain	α	Gain	
Moving Average	0.5	0.99%	0.4	4.94%	0.7	2.68%	0.5	8.52%	0.5	3.27%	0.7	23.73%	0.9	7.45%	0.6	6.94%	7.33%
MLP-Update	0.5	4.04%	0.9	18.05%	0.7	1.09%	0.4	2.51%	0.4	11.40%	0.2	27.73%	0.3	33.76%	0.6	6.98%	13.19%
GRU-Update	0.7	0.56%	0.5	5.15%	0.1	2.97%	0.5	8.18%	0.8	2.10%	0.5	0.17%	0.9	2.82%	0.8	0.77%	2.84%

- Enabling meta-learning leads to performance gain

Ablation Study

Effectiveness of Model Retraining

- ROLAND can automatically retrain itself to fit the data
- Compare ROLAND with a baseline that stops retraining after training the first 25% of snapshots
  - The baseline performs significantly worse than retraining for all the snapshots.



Takeaways

- Model design: ROLAND describes how to repurpose a static GNN for dynamic settings effectively.
- Training: ROLAND can scale to dynamic graphs with 56 million edges, which is at least 13 times larger than existing benchmarks. In addition, we innovatively formulate predicting over dynamic graphs as a meta-learning problem to achieve fast model adaptation.
- Evaluation: Compared to the common dataset split, ROLAND's live-update evaluation can reflect the evolving nature of data and model

Thank You