Graph Learning in Financial Networks

Collaboration with Tianyu Du, Fan-yun Sun, Jure Leskovec

Jiaxuan You

Stanford University

Financial Networks

- **Financial Networks:** Describe financial entities and their connections

International banking

- *Nodes: Countries*
- *Edges: Capital flows*

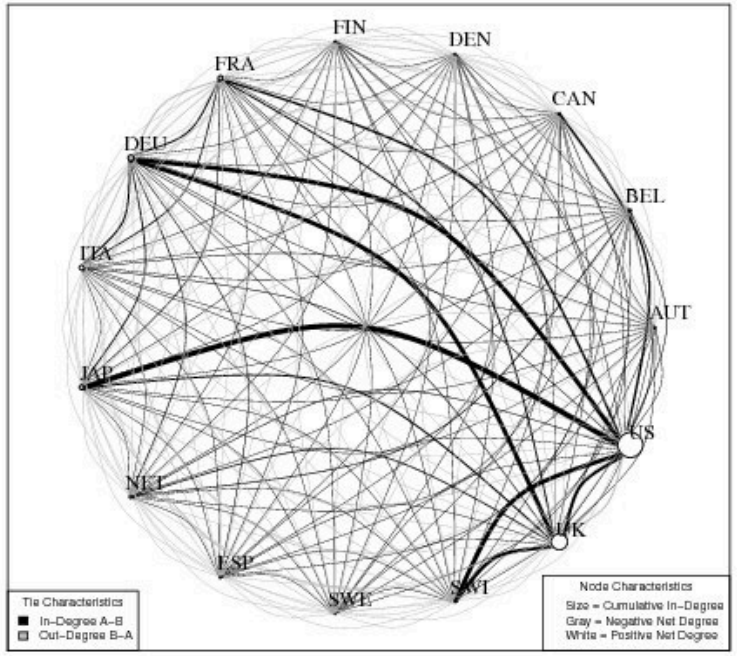


Image credit: The Political Economy of Global Finance: A Network Model

Bitcoin transactions

- *Nodes: BTC wallets*
- *Edges: Transactions*

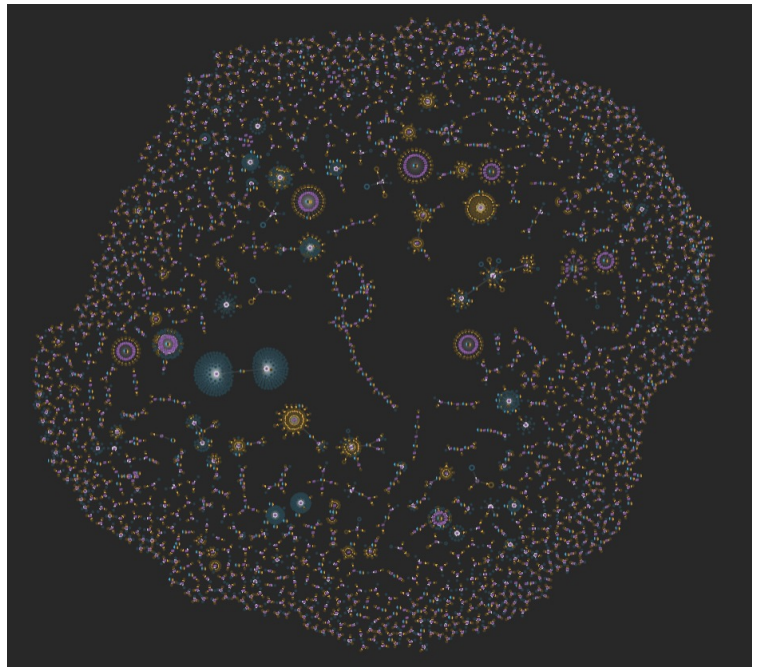
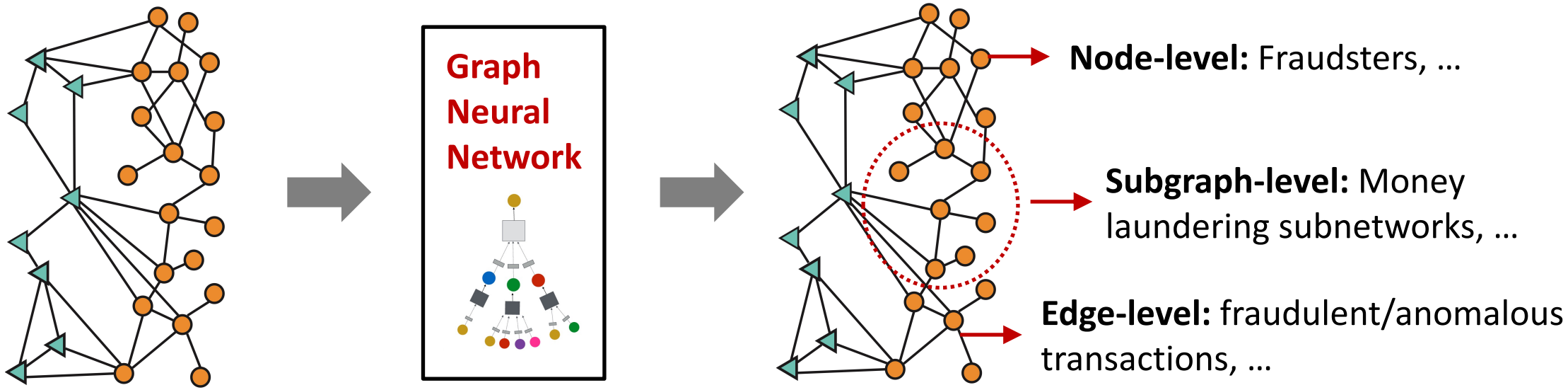


Image credit: https://dailyblockchain.github.io/

Graph Learning in Financial Networks

- **Goal: A graph learning framework for financial networks**
  - **Applications:** Fraud detection, Anti-money laundering, Anomaly detection
  - **Solution: Graph representation learning!**

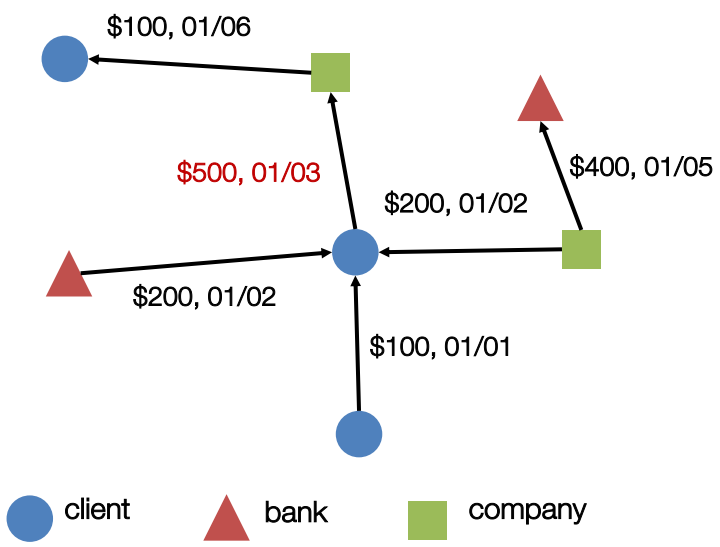


Input: Financial networks

Output: Predictions

Why Graph Representation?

- Transaction-based approach
  - "On 01/03, Client A sends Company B $500"
  - Build models based on transaction attributes
  - **Issues:** ignore the context of a transaction
- **Graph-based approach**
  - Represent transactions as a **dynamic graph**
  - Predictions are made based on the entire graph
  - **Benefits:**
    - Represents a transaction with a broader context
    - Requires fewer feature-engineering

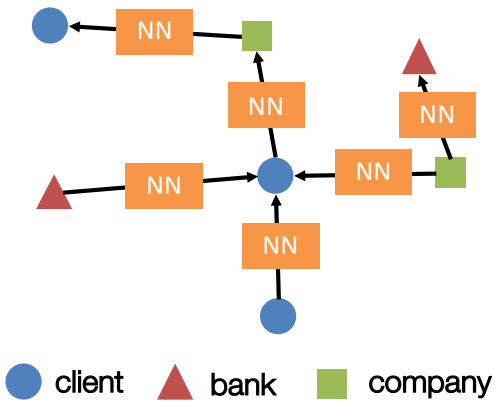


Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

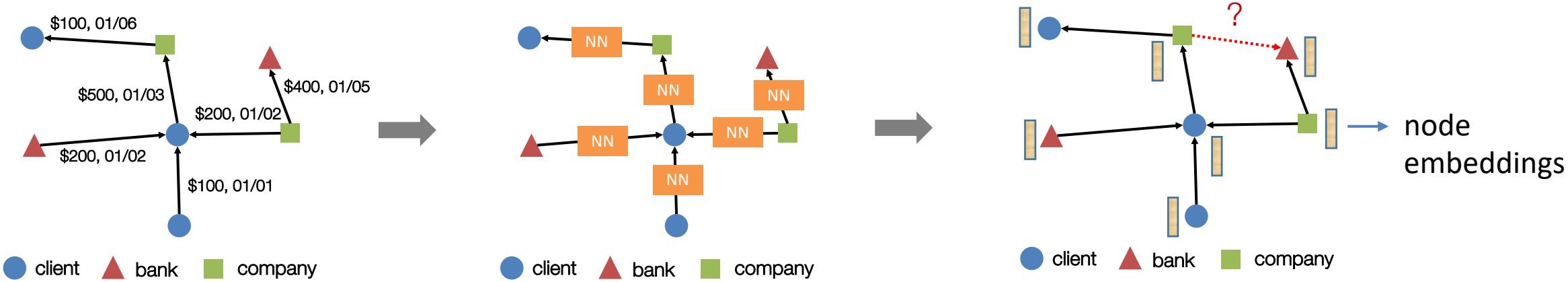


ROLAND: GNN for Financial Networks

- **ROLAND framework:**
  - Transform financial networks as GNN computational graphs
  - Learning from diverse signals

Self-supervised (from raw data)
- Will there be a transaction? **Yes**
- What is the amount? **$500**
- When will it happen? **01/03**
- …

Supervised (from external sources)
- Is it a fraud? **No**
- Is it money laundering? **Yes**
- …



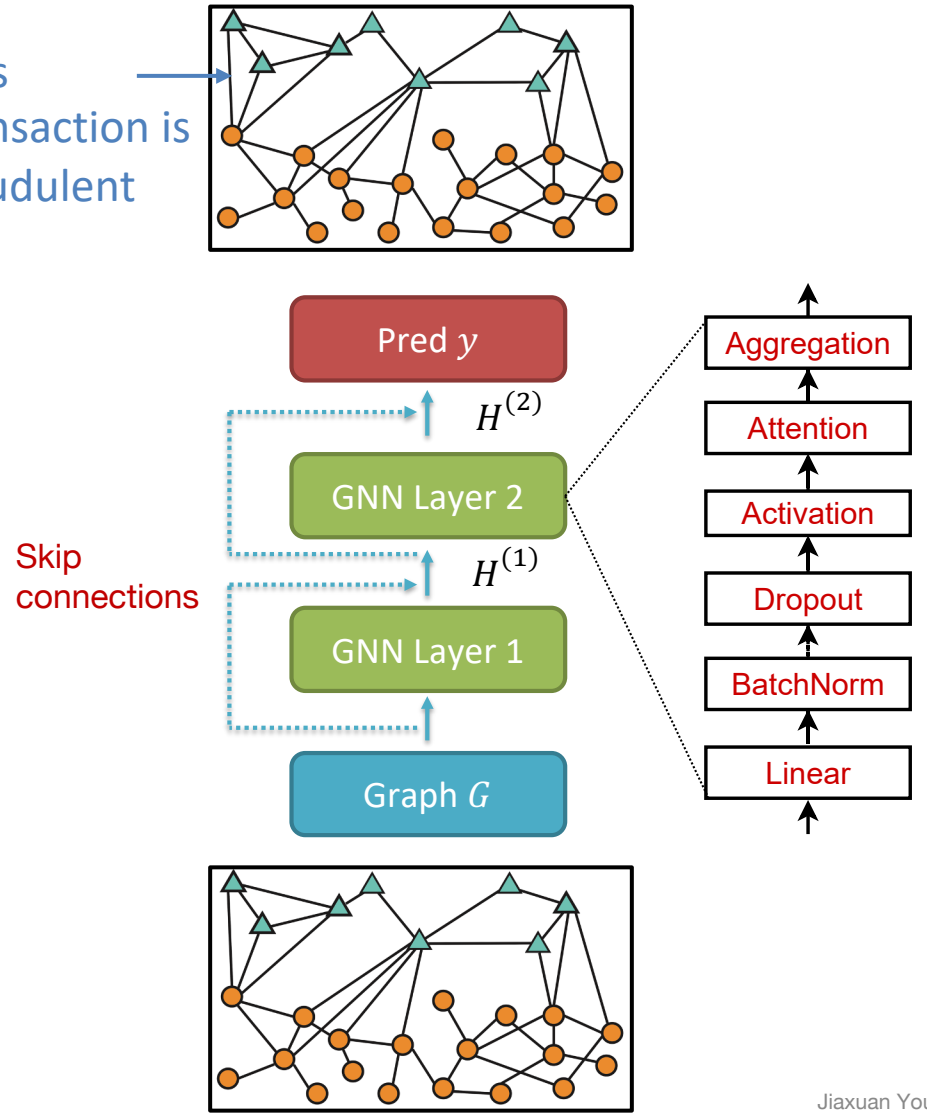
Financial networks

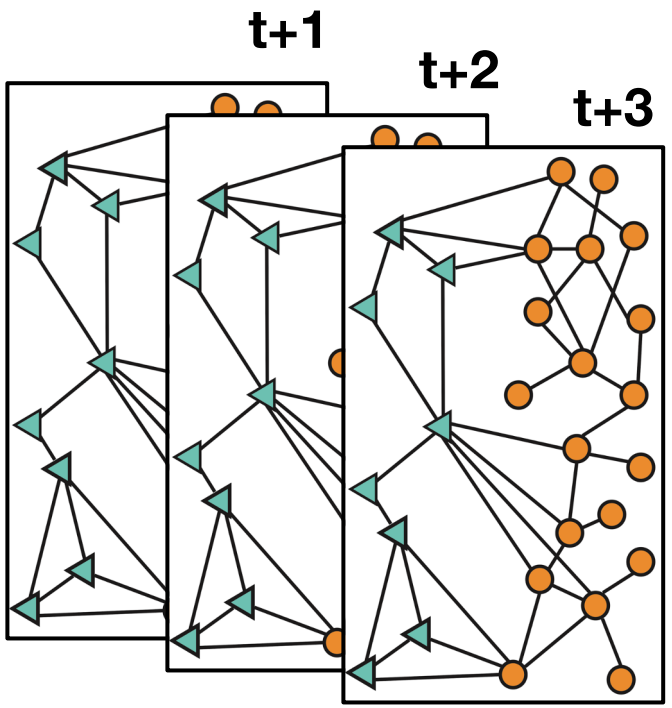
Graph Neural Networks

Learning Signals & Application

ROLAND Model: From Static to Dynamic GNN

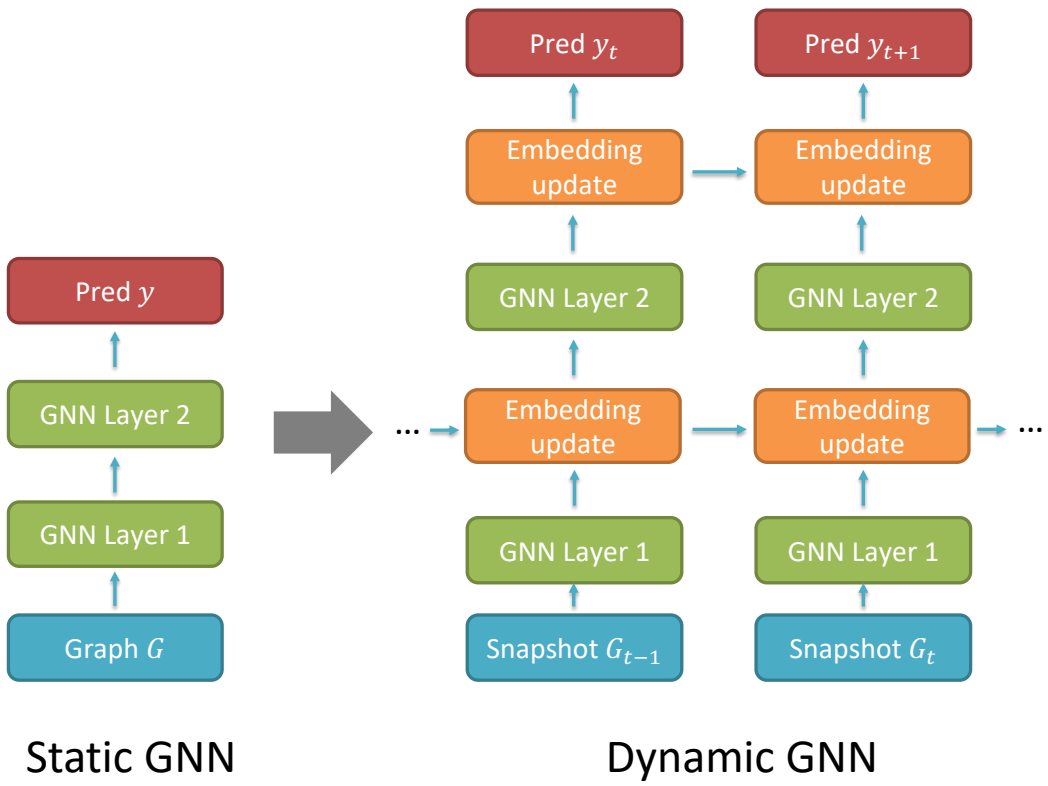
SOTA Static GNN





Question: Can we adapt a SOTA **static GNN** to **dynamic prediction tasks?**

ROLAND Model: From Static to Dynamic GNN



- **Idea: *Recurrently*** update node embeddings ***at each layer***
- Introduce a new module to a static GNN:

Embedding update

Input:

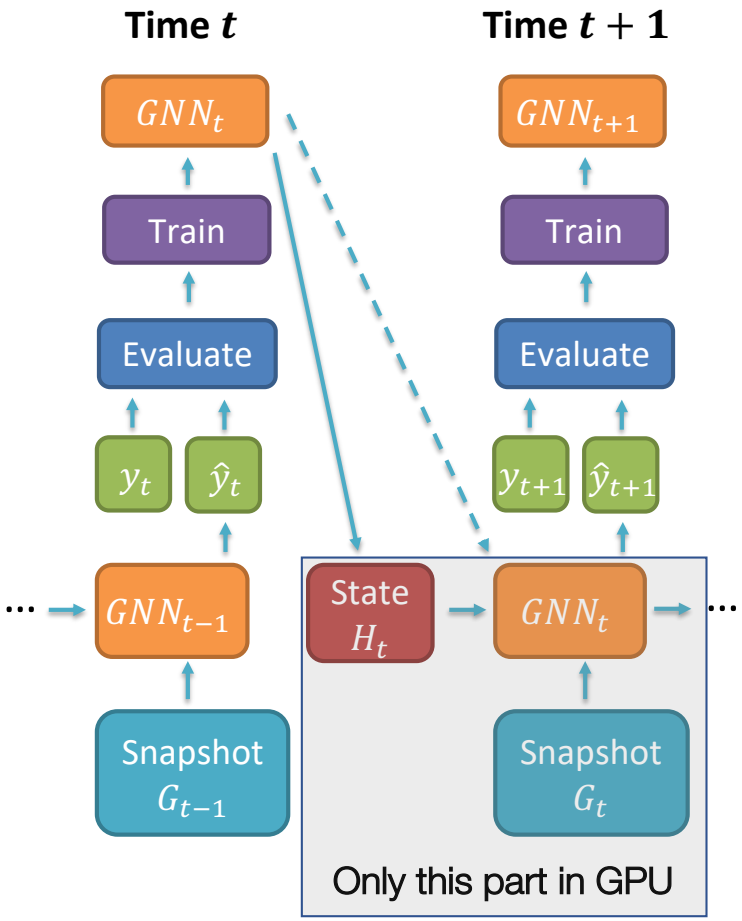
- **Previous embeddings** from the the same layer
- **Current embeddings** from the previous layer

Output: Updated embeddings

- **Benefits:**
  - Simple and effective
  - Benefit from the SOTA designs of a static GNN

ROLAND Training: Efficient Training

Live-update evaluation pipeline



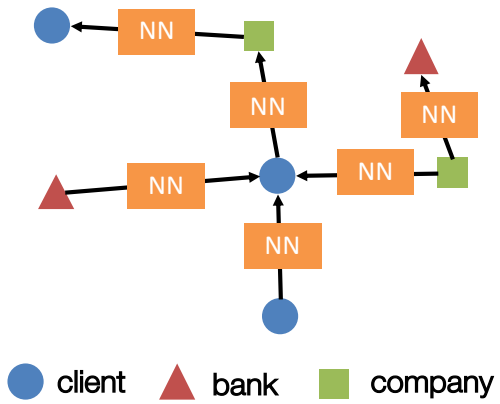
- **Incremental training:**
  - **Only keep these in GPU**
    - GNN Model GNN_t
    - Historical node states H_{t-1}
    - Incoming new graph snapshot G_t
    - Efficient and work well in practice
- **Meta training:**
  - Train a meta-GNN that can **quickly adapt to new data**
  - **Benefits: ROLAND does not need to be frequently retrained**

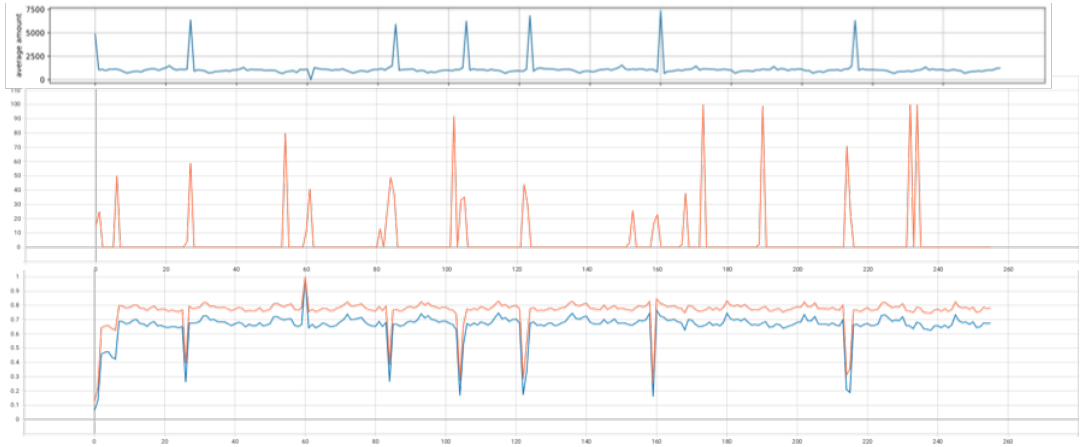
Overview of This Talk

ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

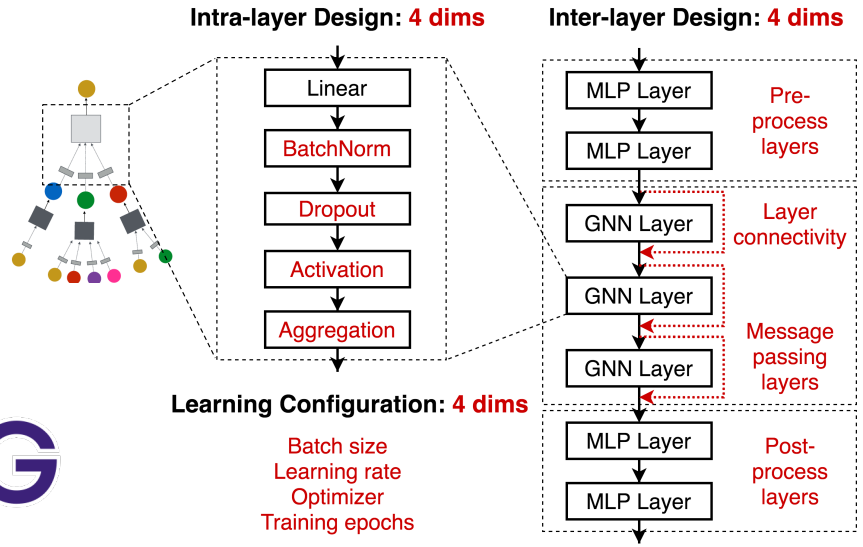




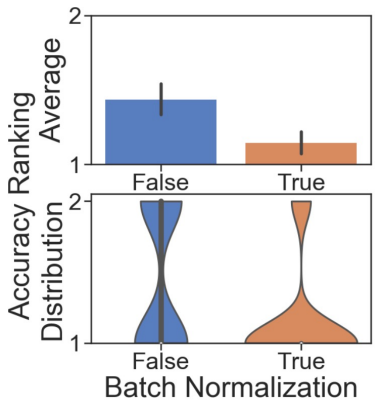
ROLAND Implementation: GraphGym

- **GraphGym** offers a standardized pipeline for graph learning
  - Design Space of Graph Neural Networks (NeurIPS 2020 spotlight)
  - **Now officially part of PyG 2.0! (Stay tuned for my next talk :)**

General GNN design space



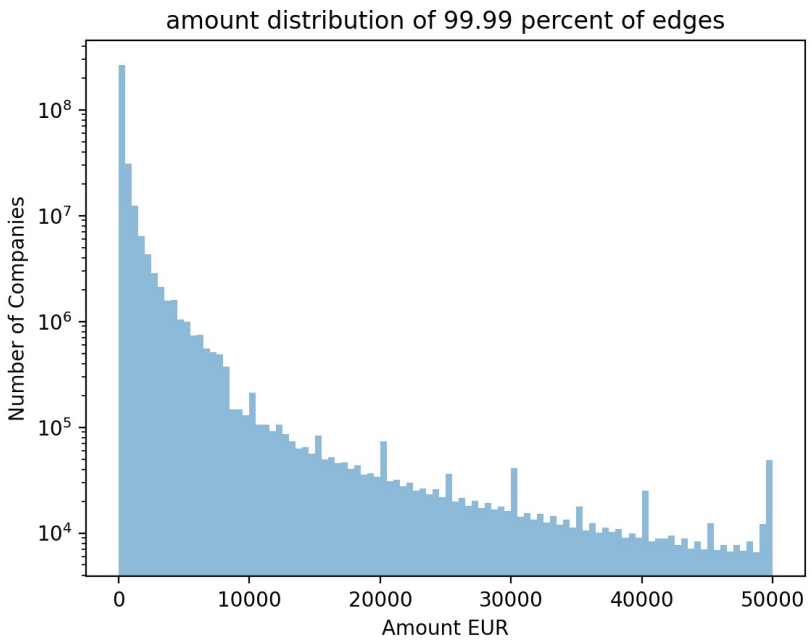
Clean interface for comparing designs

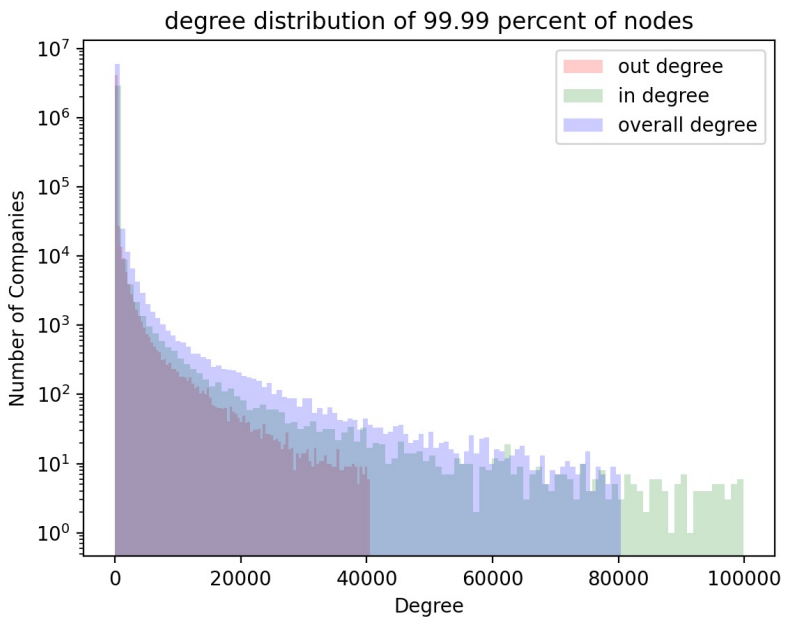


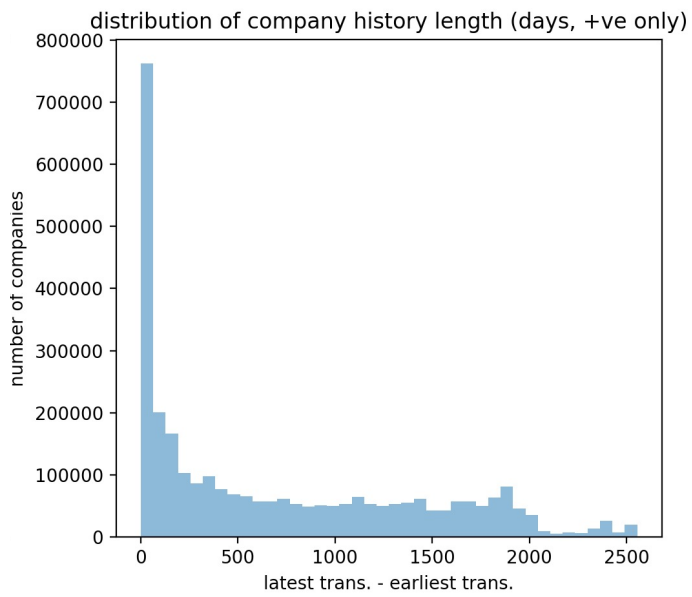
- We **implement the financial prediction pipeline with GraphGym**
  - We adopt many **optimal design choices from GraphGym**

An Example Financial Dataset

- **Transactions between companies** captured by a bank clearing system
  - **336 million** transactions, **6 million** accounts, **6 years** of daily data







Task 1: Classify Fraud Transactions

- **Task setting:** Supervised learning
  - ***Unbalanced labels:*** 2% of all the transactions are fraudulent
  - ***Random dataset split:*** 80% training, 10% validation, 10% testing
  - ***Metric:*** AUC on test set (random guess has AUC 0.5, higher is better)
- **Best non-deep learning baseline**
  - **AUC: ~0.8**
- **Simple GNN implementation using out-of-the-box PyG**
  - **AUC: 0.81**
- **ROLAND framework works much better!**
  - **AUC: 0.94**

Task 2: Forecast Future Transactions

- **Task setting:** Self-supervised/Unsupervised learning
  - ***Rolling prediction:*** On each day, use all the historical information to predict the transactions on the next day
  - ***Metric:*** Mean reciprocal ranking (MMR) of ground-truth future transactions

	Bitcoin-OTC	Bitcoin-Alpha	UCI-Message
GCN [14]	0.0025	0.0031	0.1141
DynGEM [12]	0.0921	0.1287	0.1055
dyngraph2vecAE [8]	0.0916	0.1478	0.0540
dyngraph2vecAERNN [8]	**0.1268**	**0.1945**	0.0713
EvolveGCN-H [30]	0.0690	0.1104	0.0899
EvolveGCN-O [30]	0.0968	0.1185	**0.1379**
ROLAND Moving Average	0.0468 ± 0.0022	0.1399 ± 0.0107	0.0649 ± 0.0049
ROLAND MLP	0.0778 ± 0.0024	0.1561 ± 0.0114	0.0875 ± 0.0110
ROLAND GRU	$\mathbf{0.2203 \pm 0.0167}$	$\mathbf{0.2885 \pm 0.0123}$	$\mathbf{0.2289 \pm 0.0618}$
Improvement over best baseline	73.74%	43.33%	65.99%

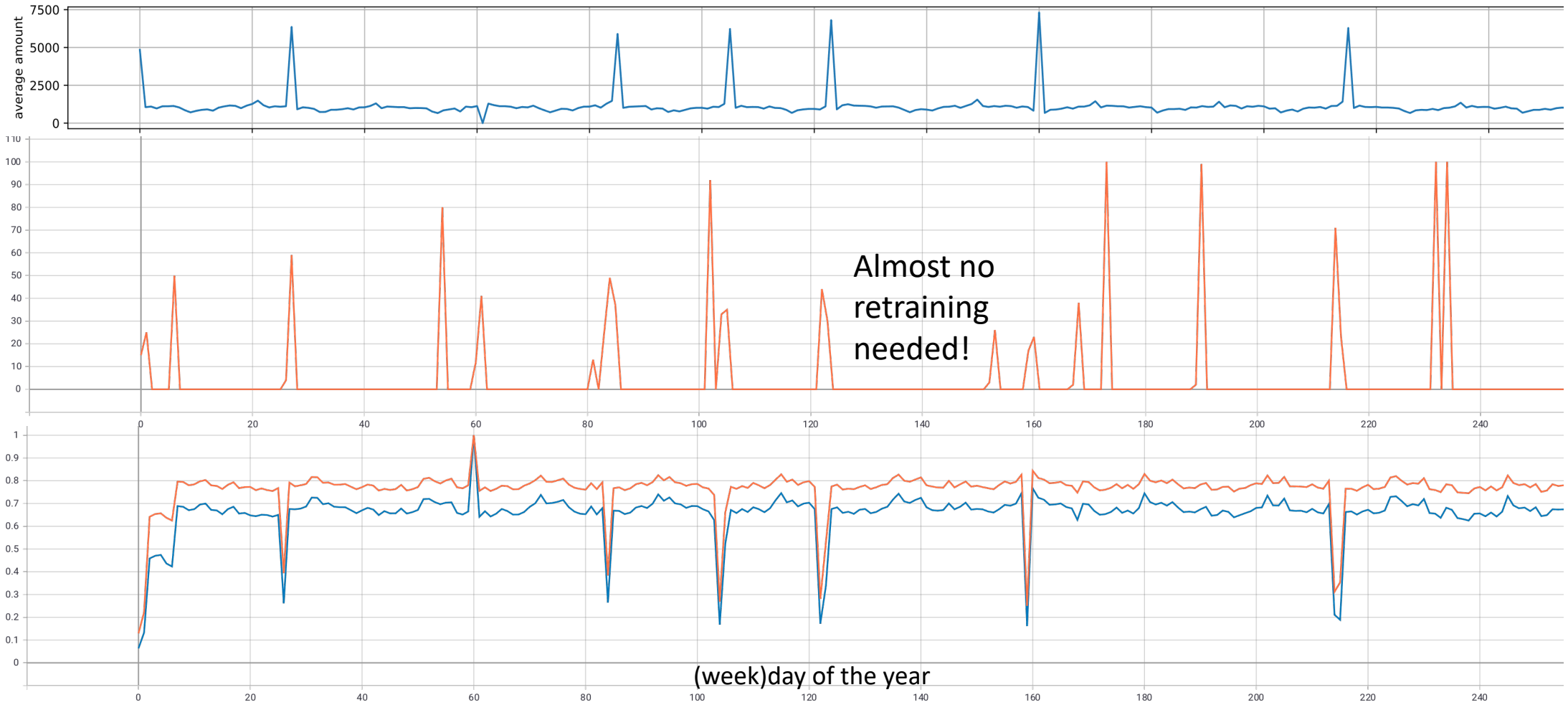
ROLAND significantly outperforms SOTA baselines

Analysis: ROLAND's Performance over Time

Transaction amount

# of retraining epochs

Test MRR
Recall@1

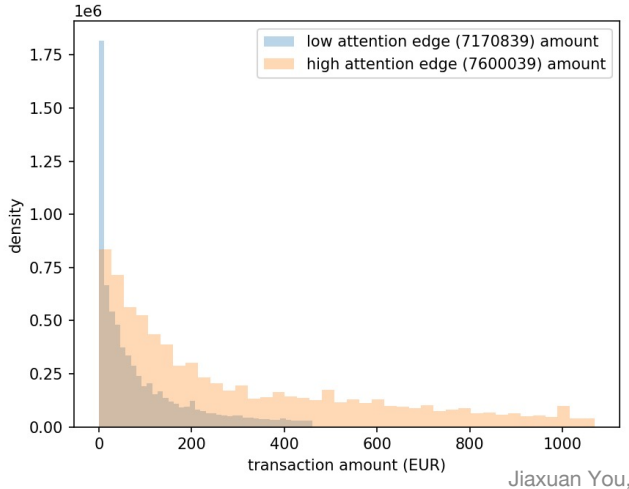


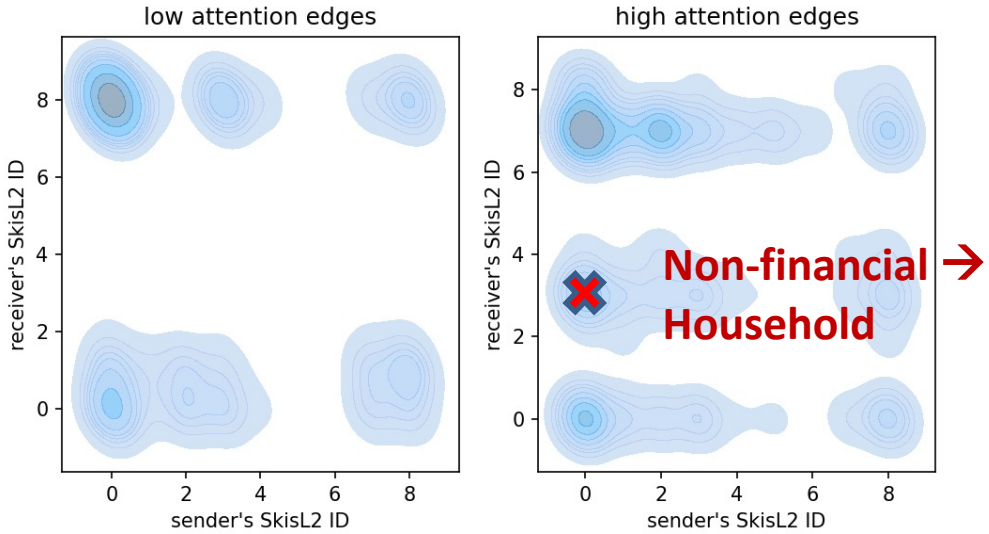
ROLAND maintains a robust performance over transaction pattern changes

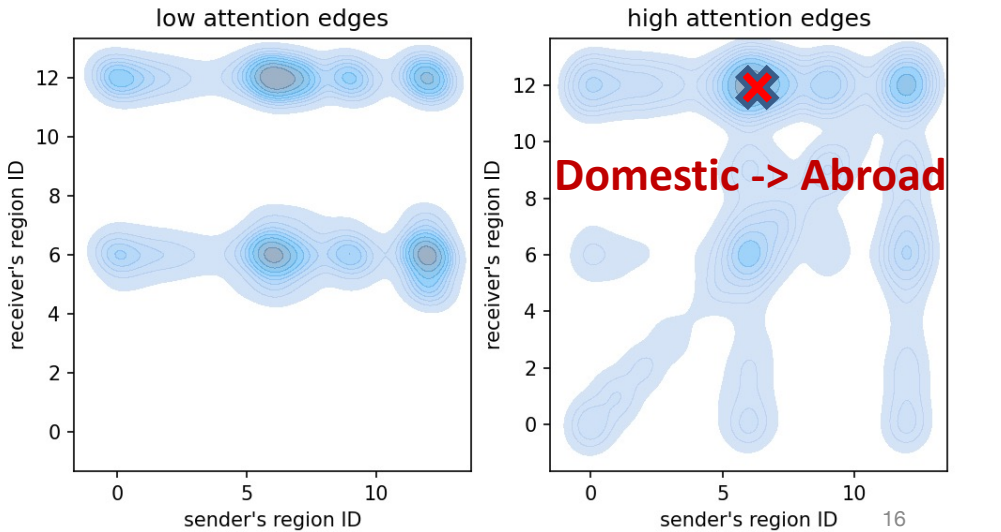
Analysis: Interpreting ROLAND's Predictions

- **Question:** What kind of transactions the model paid attention to?
- **Method**: Inspect **edges' attention scores of the last GNN layer**.
  - Select edges with low/high attention scores.
  - Get the feature value distribution of these edges

Higher attention to large amount transactions





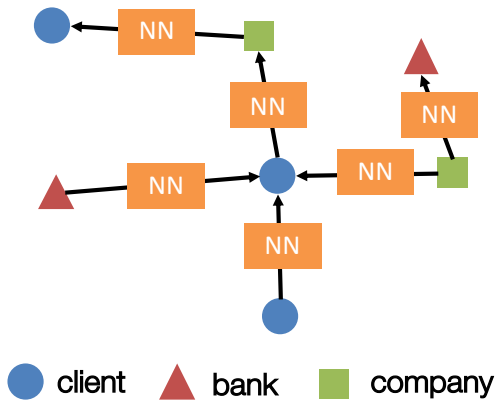


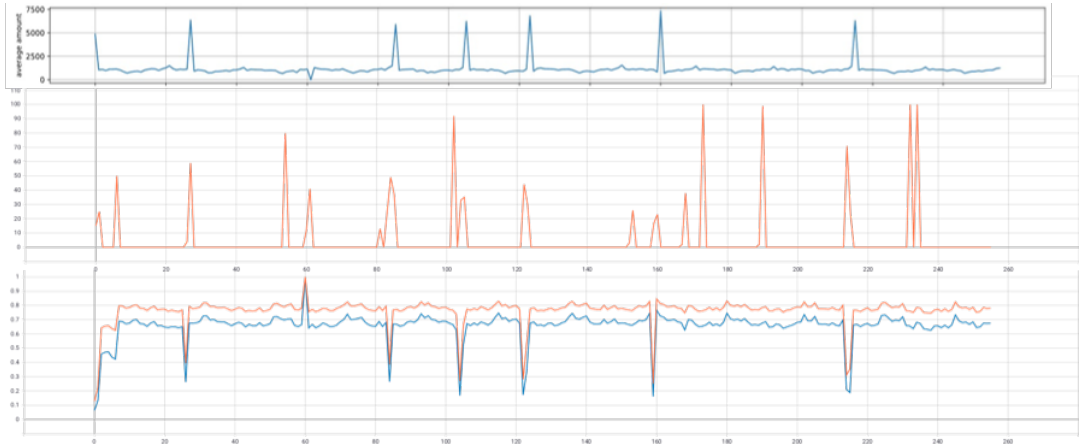
Overview of This Talk

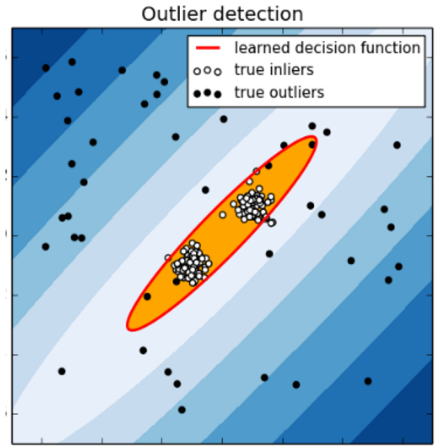
ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection

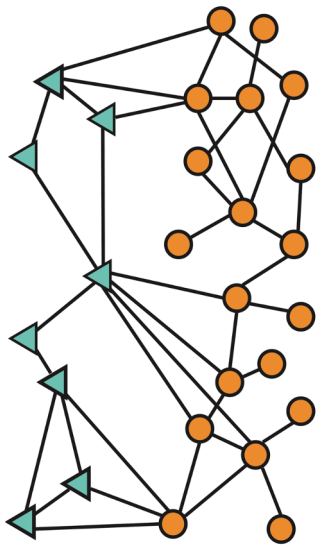




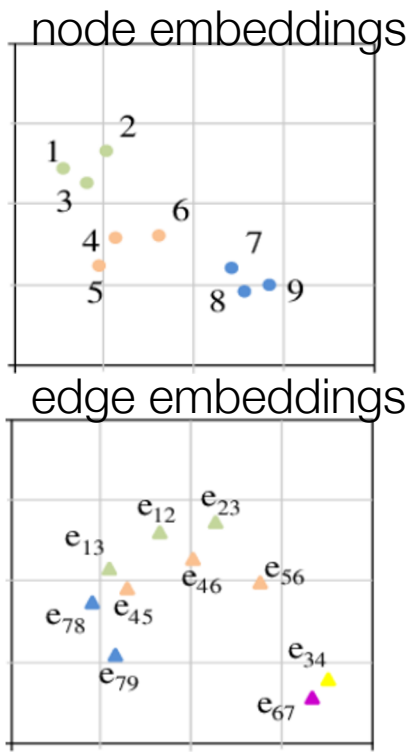


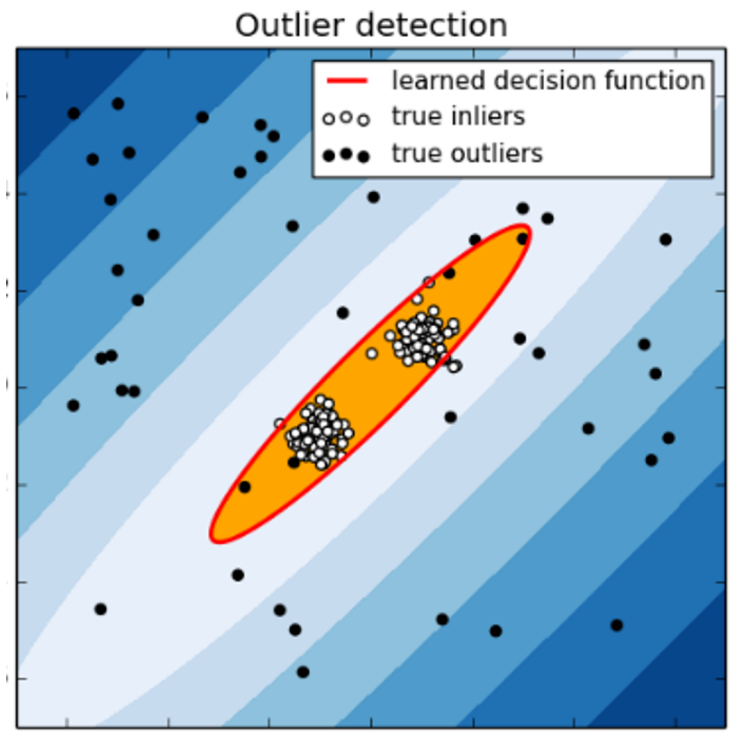
Application: Anomaly Detection

- **Goal:** Detect anomalous transactions **without external supervision**
  - Combine Graph Representation Learning with Outlier Detection



Input: Financial networks





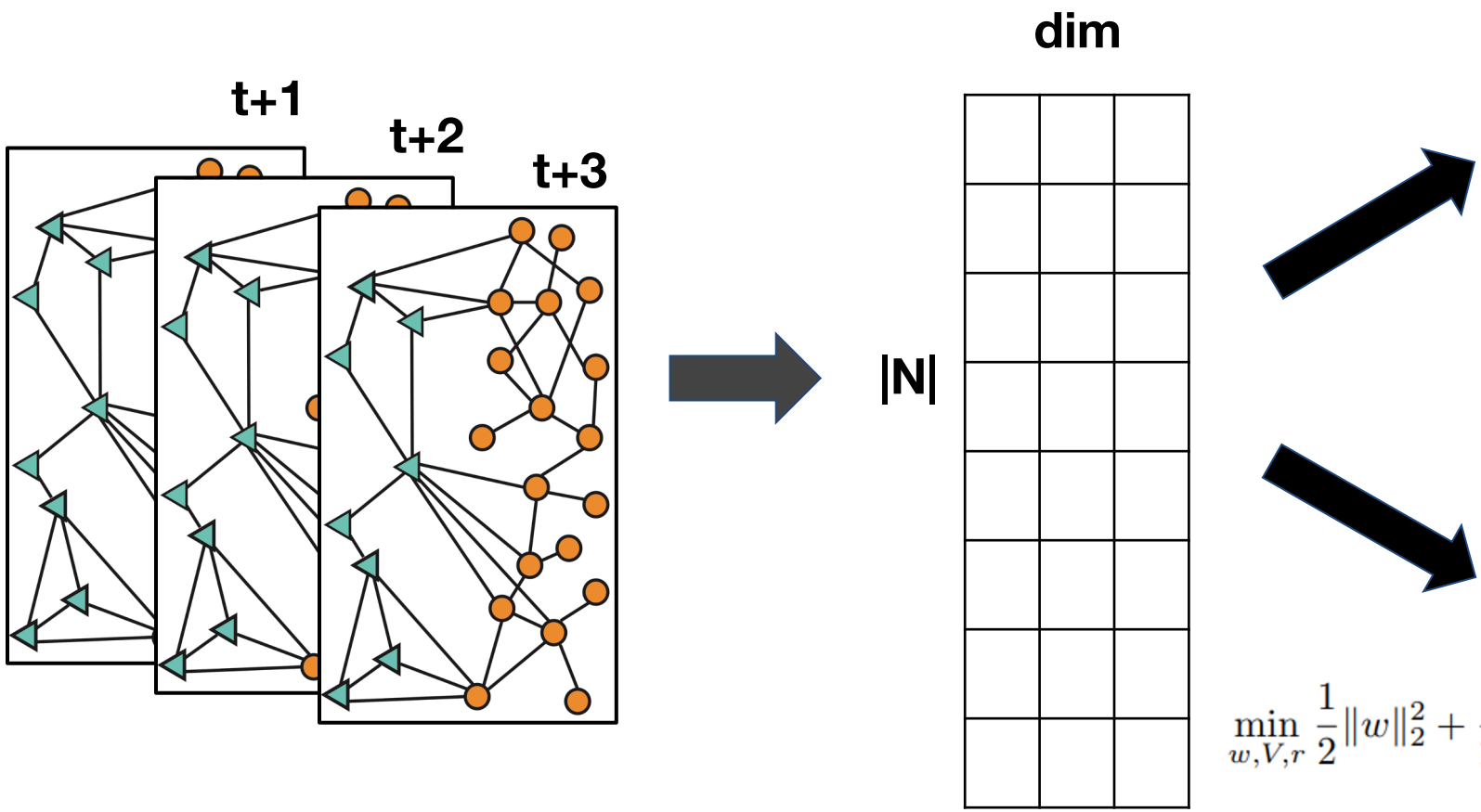
Self-supervised objectives

"Indicators and patterns of money laundering or terrorist financing"

Real world Indicators (from baselgovernance.org)	**Objectives for GNN**
investment funds sent to countries of concern	Predict country
Large cash deposits into (company) accounts	Predict amount
unusual subject of transaction	Predict subject of transaction (link prediction)
high volume of transactions within a short period	predict acculumated amount
multiple individuals sending funds to the one beneficiary	predict # transactions

Objectives for Anomaly Detection

- **Solution:** ROLAND framework + diverse objectives



Self-supervised Objectives

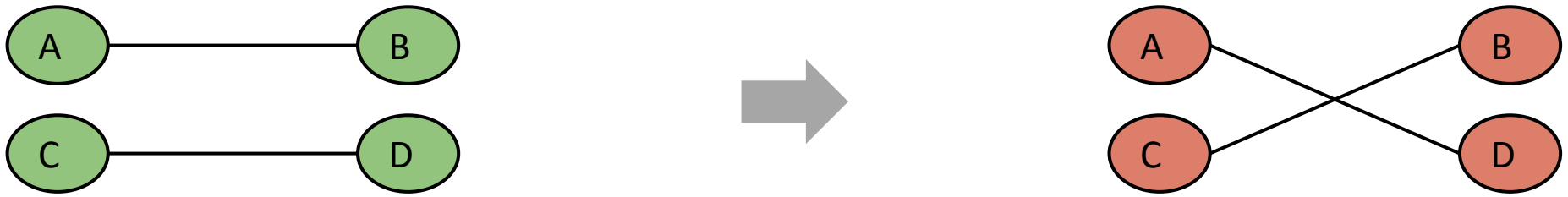
- Objective 1: predict amount
- Objective 2: predict recipient
- Objective 3: predict country
- …

Outlier Detection Objectives

$$\min_{w,V,r} \frac{1}{2}\|w\|_2^2 + \frac{1}{2}\|V\|_F^2 + \frac{1}{\nu} \cdot \frac{1}{N} \sum_{n=1}^{N} \max(0, r - \langle w, g(V\mathbf{X}_{n:})\rangle) - r$$

Quantitative Evaluation for Anomaly Detection

- **Task:** Detect anomalous transactions **without supervision**
  - **Goal:** Simulate real-world anomaly detection use cases
  - Corrupt the dataset to see if ROLAND can extract them
    - Report *precision/recall* of corrupted edges detected by ROLAND
  - Ways of corrupting the dataset:
    - **Amount corruption (~1% transactions)**
    - **Country corruption (~1% transactions)**
    - **Recipient corruption (~1% transactions) → Rewire !**



Quantitative Evaluation for Anomaly Detection

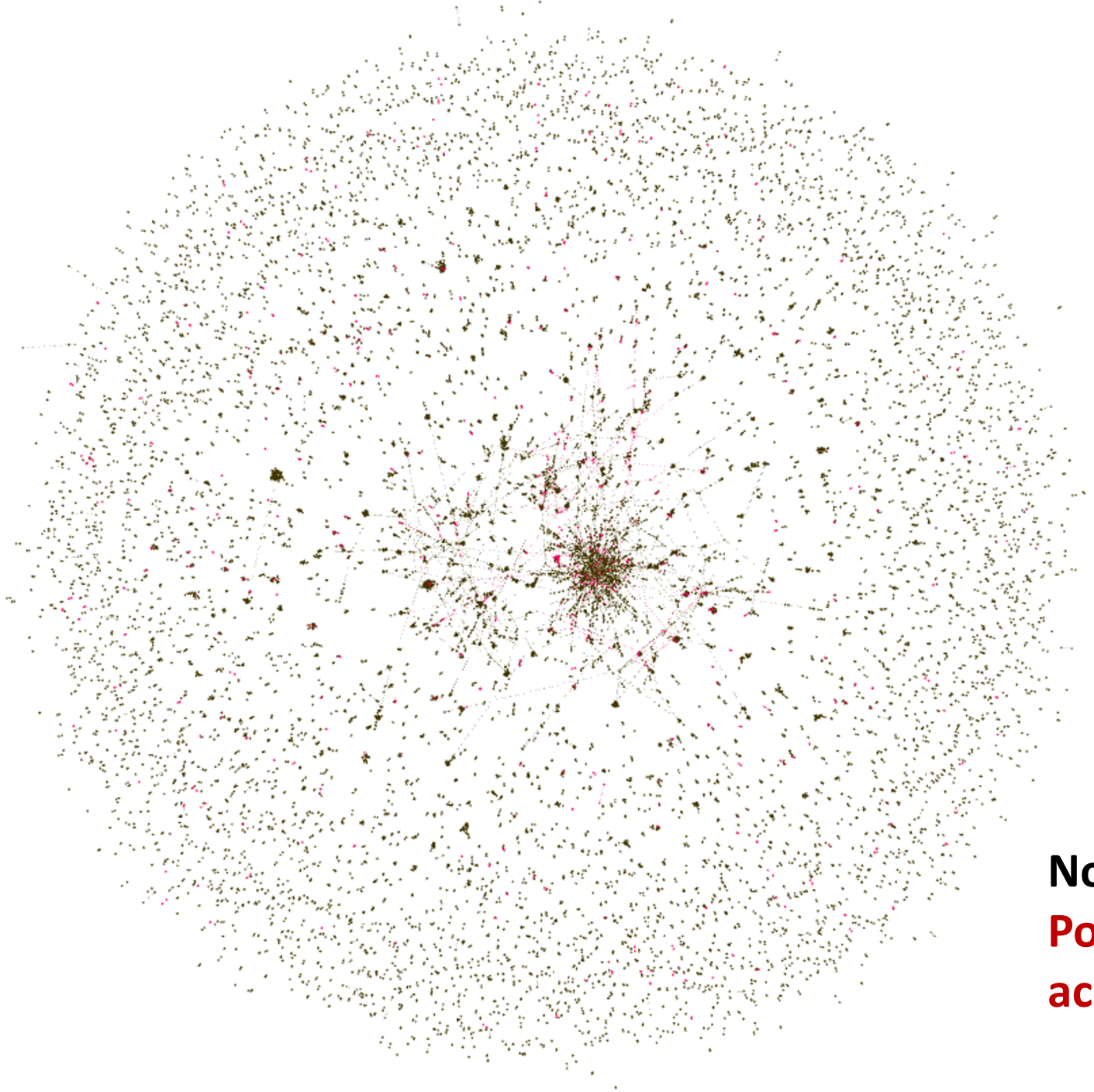
Deep learning baseline:

+14.7% Precision **+14.2% Recall**

	Precision (%)				Recall (%)			
	Naive Baseline	Link Prediction + One Class SVM	Multi-head Self-supervised + One Class SVM	Multi-head Self-supervised End to end	Naive Baseline	Link Prediction + One Class SVM	Multi-head Self-supervised + One Class SVM	Multi-head Self-supervised End to end
Amount corruption (~1%) - 20% corruption by x2	0.34	1.06	0.94	**1.32**	4.39	13.92	12.25	**17.25**
- 20% corruption by x5	0.47	1.00	0.95	**1.09**	6.02	12.93	12.33	**14.14**
- 20% corruption by x10	0.52	1.03	1.05	**1.13**	6.80	13.29	13.60	**14.65**
- 20% corruption by x100	0.37	1.07	1.05	**1.09**	4.83	**14.03**	13.57	**14.03**
- 20% corruption by x1000	0.32	1.10	1.09	**1.12**	4.07	14.33	14.18	**14.48**
Amount corruption (~1%)	1.97	4.71	4.74	**5.24**	5.26	12.64	12.70	**13.95**
Country Corruption (~1%)	0.64	2.27	2.41	**2.65**	3.30	11.68	12.40	**13.66**
Recipient Corruption (~1%)	0.63	1.37	1.55	**1.65**	4.36	9.53	10.82	**11.39**
Total Corruption (~3%)	3.15	7.48	7.92	**8.58**	4.61	10.94	11.59	**12.49**

Rule-based baseline:

+172% Precision **+171% Recall**



Normal accounts

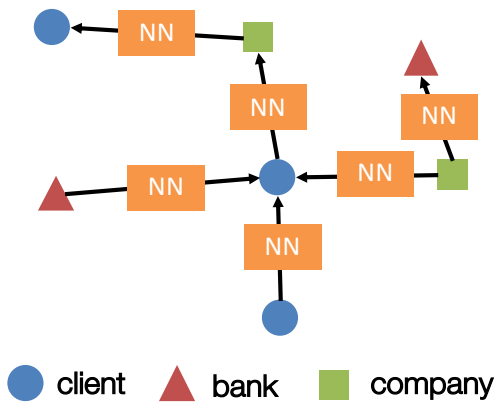
Potential anomalous accounts

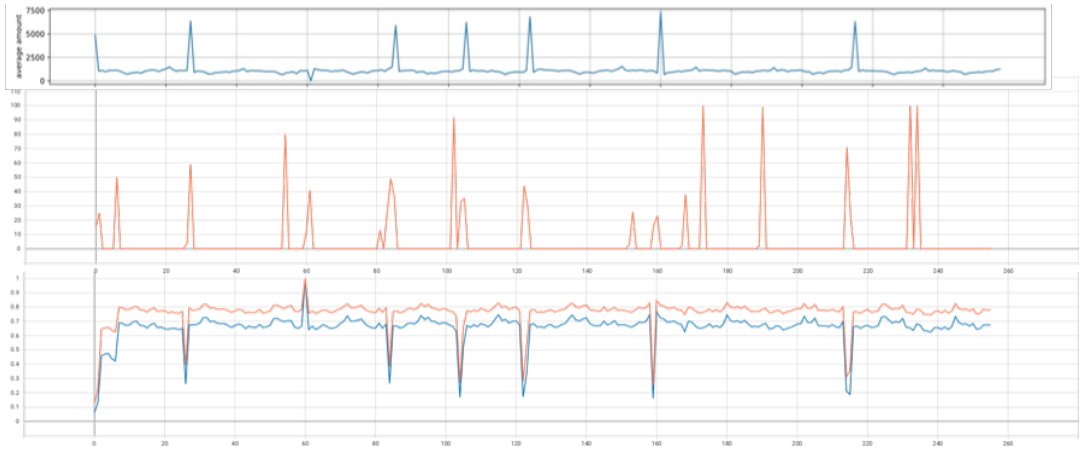
Overview of This Talk

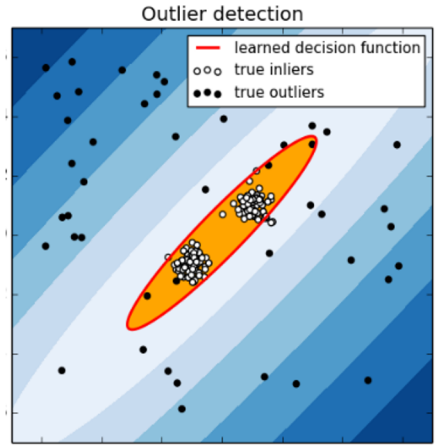
ROLAND: GNN for Financial Networks

Results & Insights from ROLAND

Application: Anomaly Detection







Resources: ROLAND (in submission)

Design Space for Graph Neural Networks

Learn more about GNN: PyG.org

PyG