SLADE: Dynamic Anomaly Detection in Edge Streams without Labels via Self-Supervised Learning



Jongha Lee Sunwoo Kim Kijung Shin

Anomalies in Real-world Networks

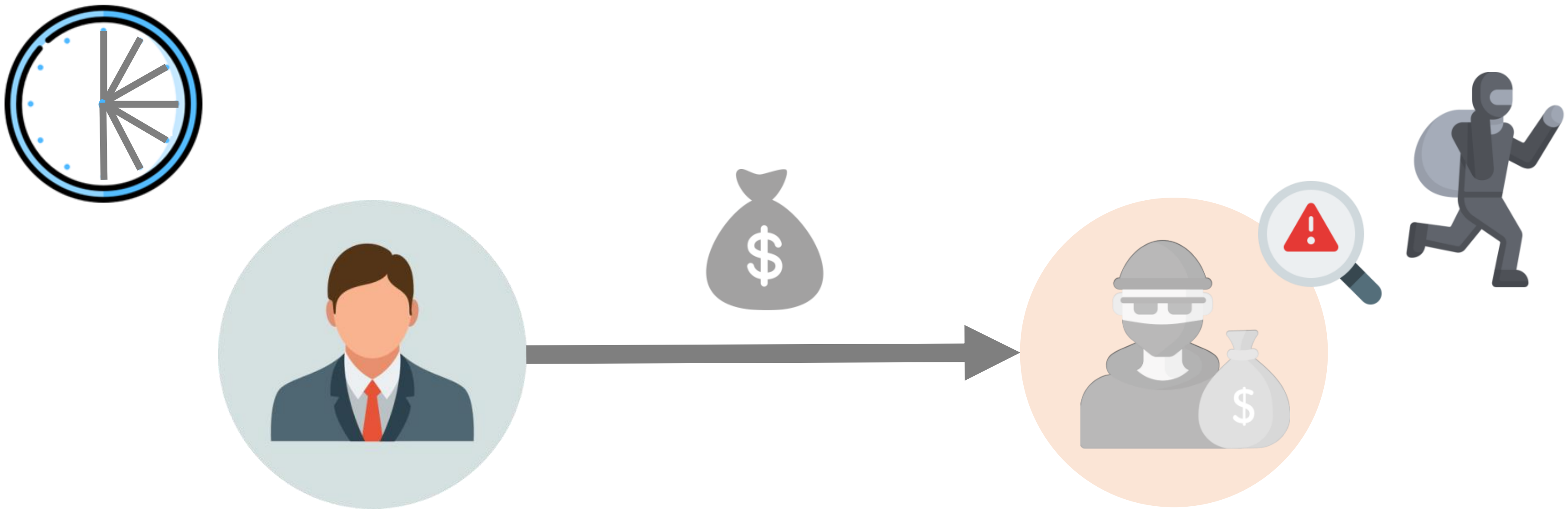
In real-world networks, various anomalies exist and harm normal users

- Account hijackers in social media
- Spammers in email networks
- Fraudsters in financial networks

Challenge 1. Time Delay in Detection

In anomaly detection, detection time delay can cause severe damage

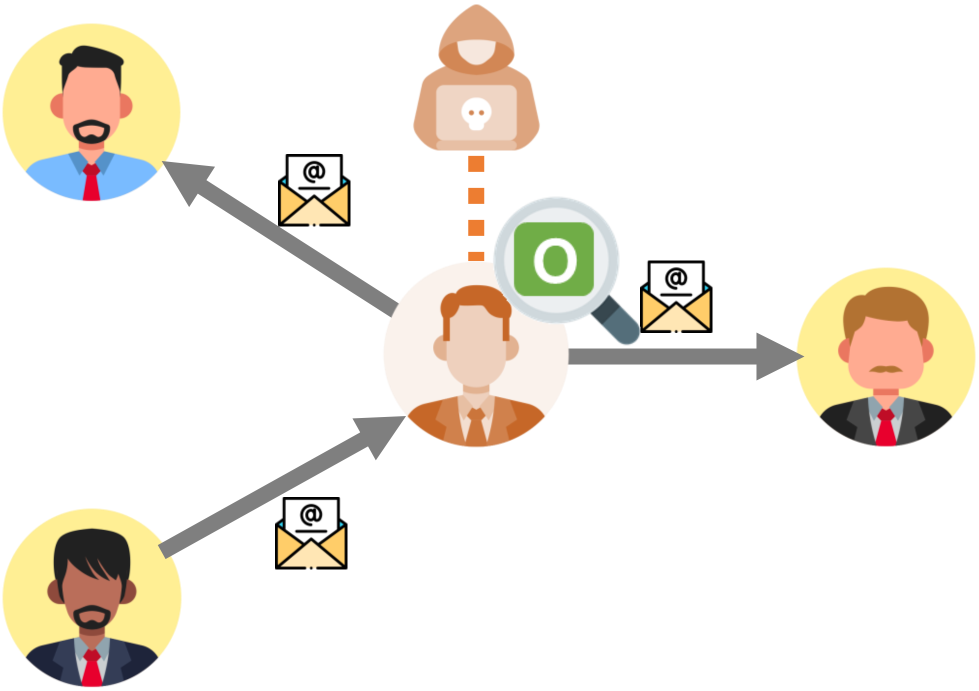
- We should minimize the detection time delay to take proper action immediately



Challenge 2. Dynamically Changing States

The state of users in real-world networks can change over time

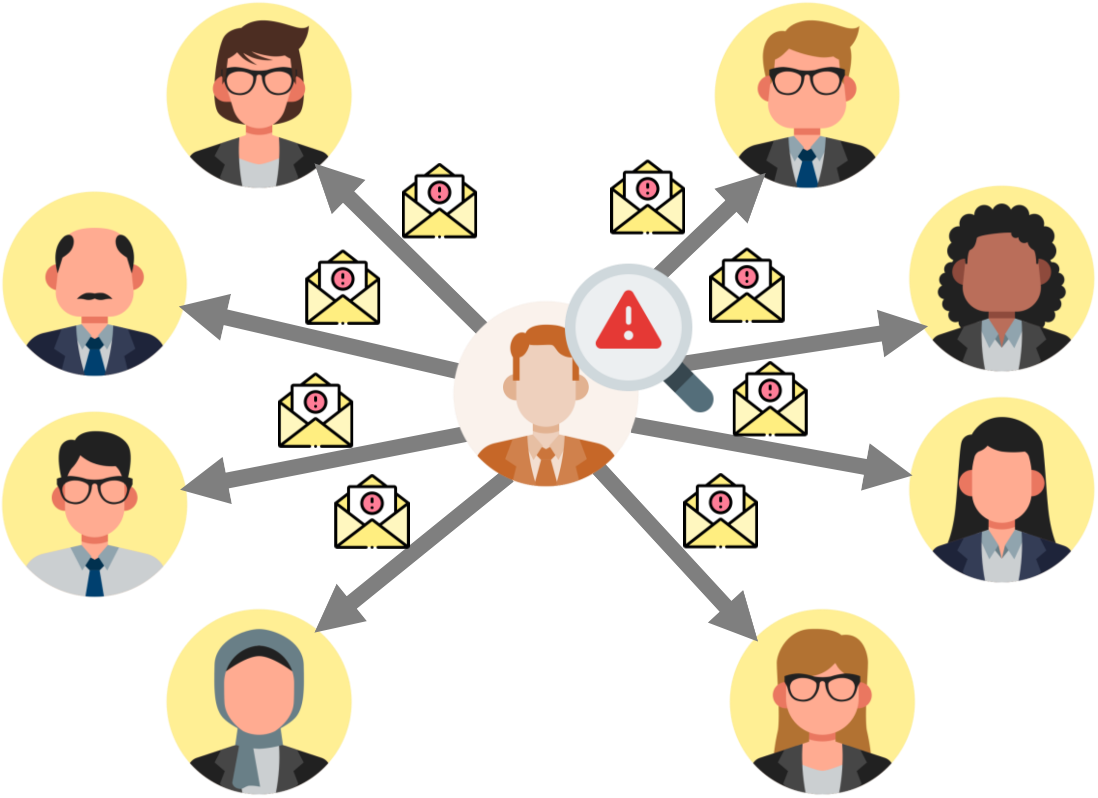
- For example, a state can be either normal or abnormal
- We need more complicated models to detect complex dynamic anomalies



Challenge 2. Dynamically Changing States

The state of users in real-world networks can change over time

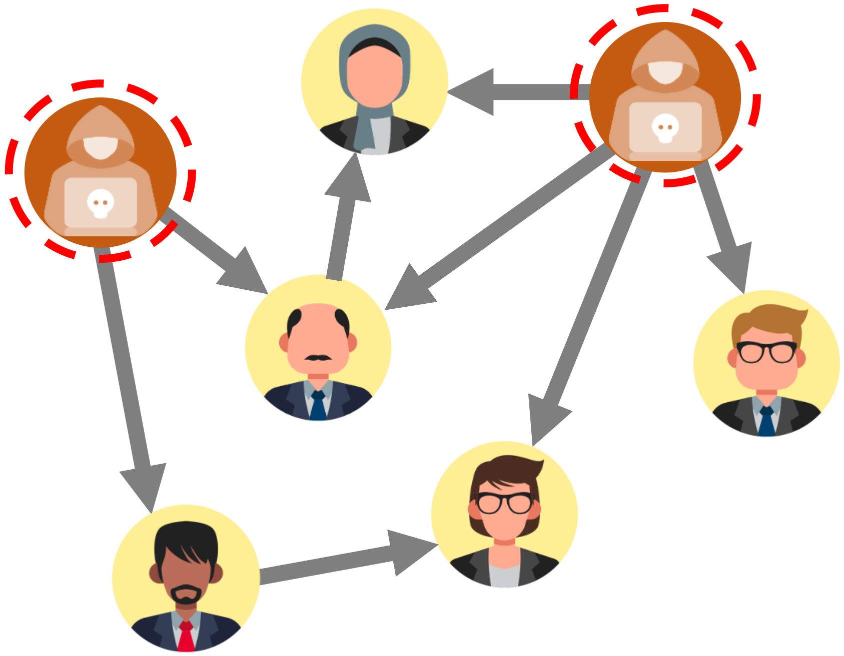
- For example, a state can be either normal or abnormal
- We need more complicated models to detect complex dynamic anomalies



Challenge 3. Absence of Anomaly Labels

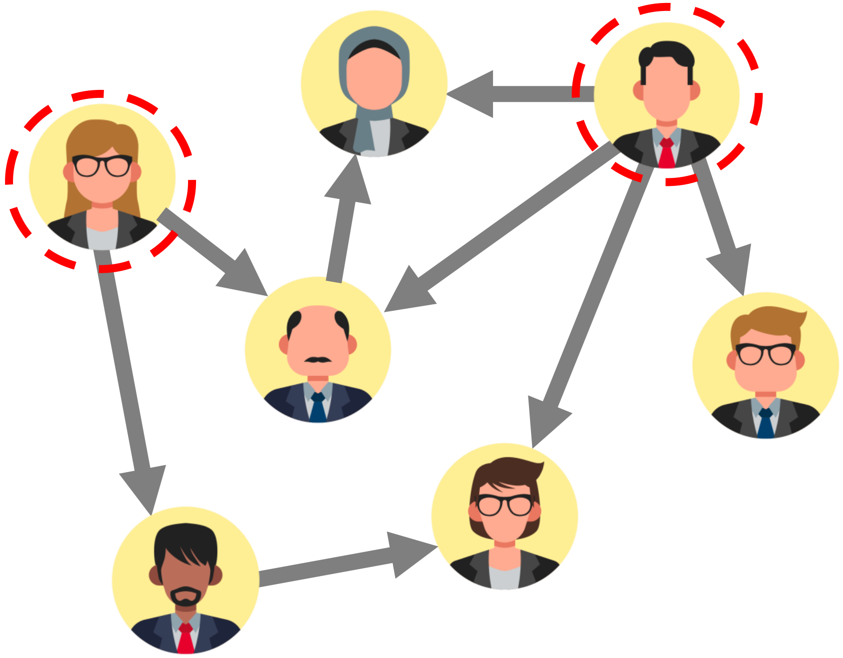
Anomaly labels might be absent in the observable input graph

- We require models that can be trained without label supervision



Ground Truth Labels

(Unobservable)



Input Graph

(Observable)

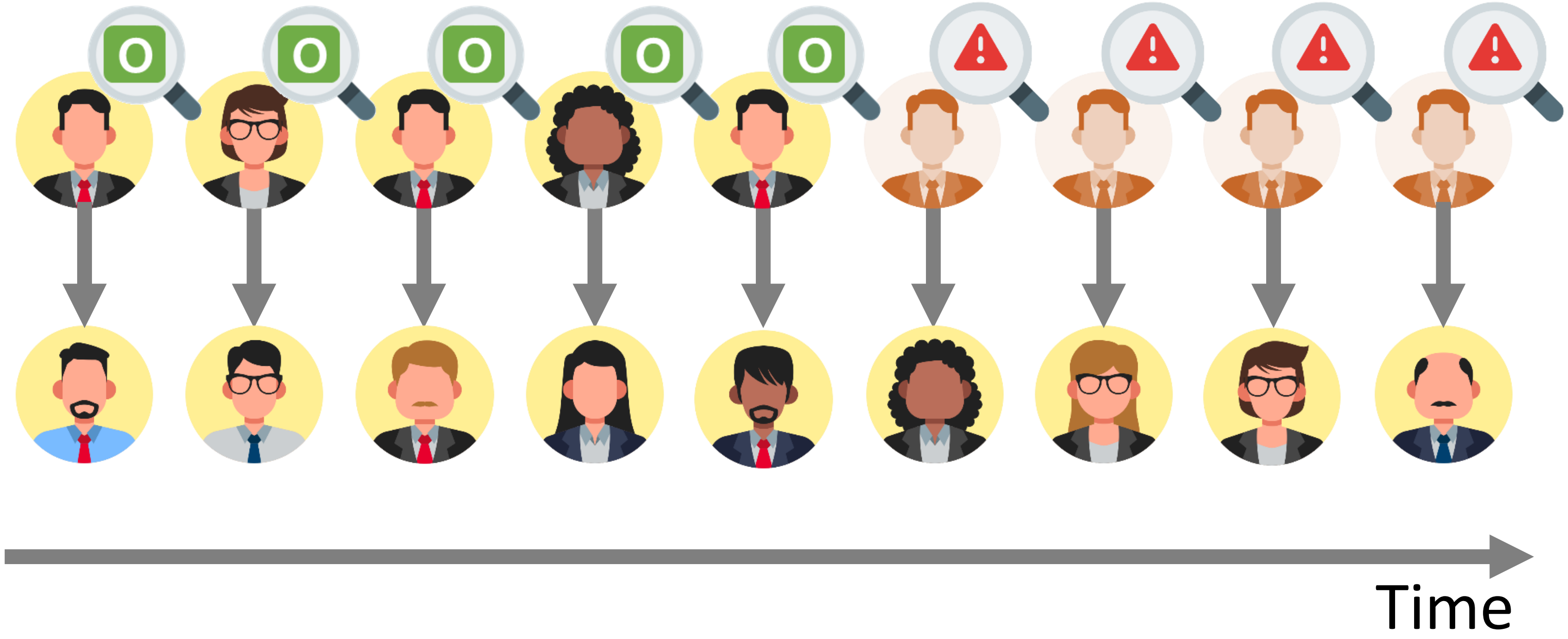
Contents

- Introduction
- Problem Description
- Proposed Method: SLADE
- Experimental Results
- Conclusion

Problem Definition

Dynamic Anomaly Detection in Edge Streams

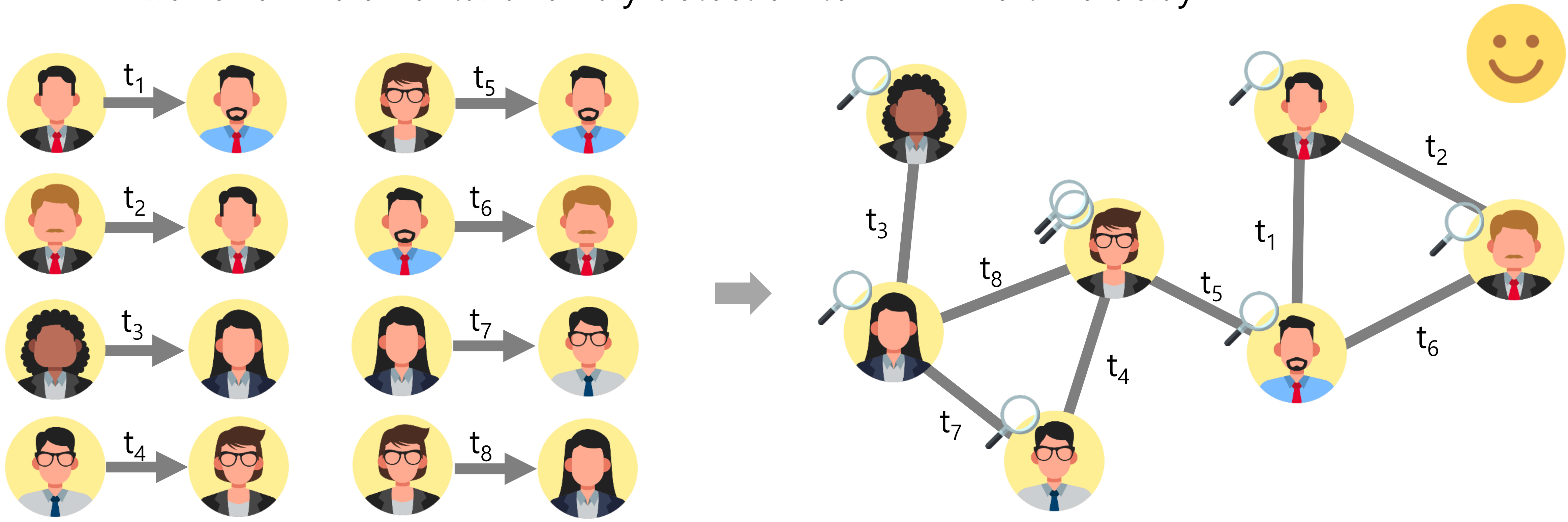
- To classify the current dynamic status of the actor node in edge streams
- The actor node refers to the node that performs an action within the edge
- The dynamic status can be either normal or abnormal



Our Graph Model: CTDG

Continuous Time Dynamic Graph (CTDG)

- The input graph is incrementally updated by each newly arriving edge with its timestamp
- Allows for incremental anomaly detection to minimize time delay



Edge Stream

Continuous Time Dynamic Graph

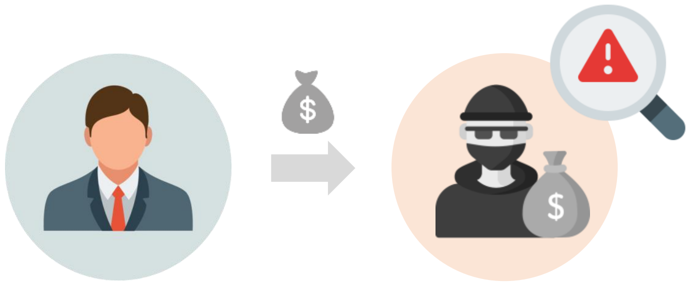
Contents

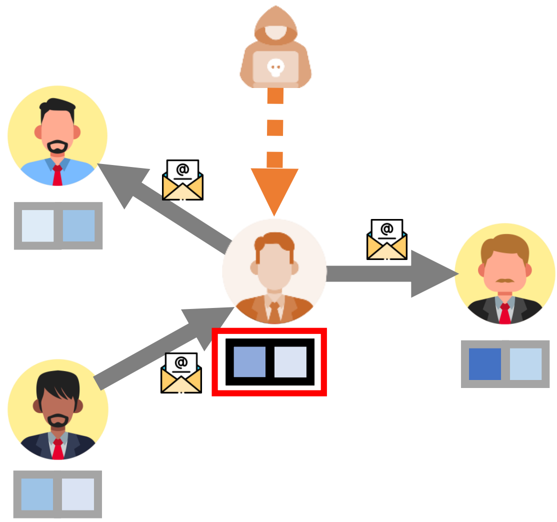
- Introduction
- Problem Description
- Proposed Method: SLADE
- Experimental Results
- Conclusion

Proposed Method: SLADE

Self-supervised Learning for Anomaly Detection in Edge Streams

- Incremental method to minimize time delay
- Dynamic representation of interaction patterns

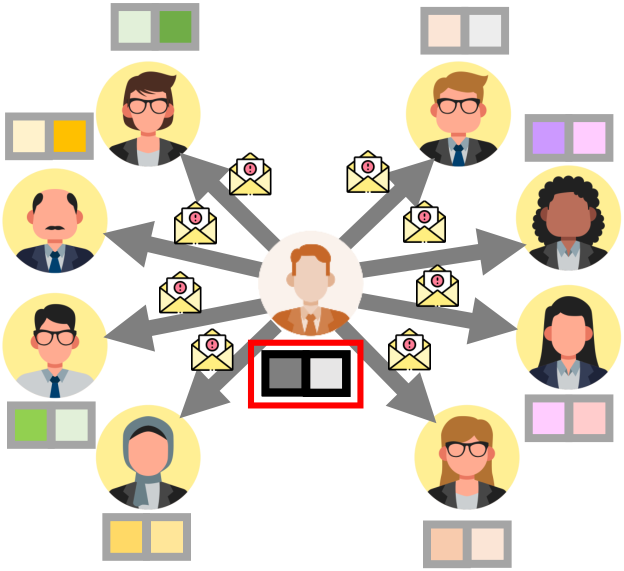




Proposed Method: SLADE

Self-supervised Learning for Anomaly Detection in Edge Streams

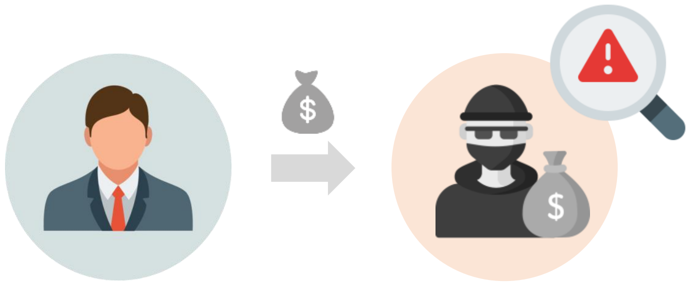
- Incremental method to minimize time delay
- Dynamic representation of interaction patterns

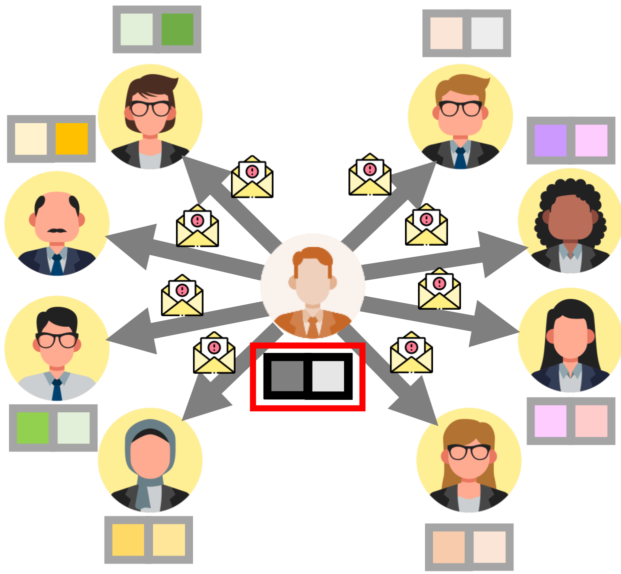


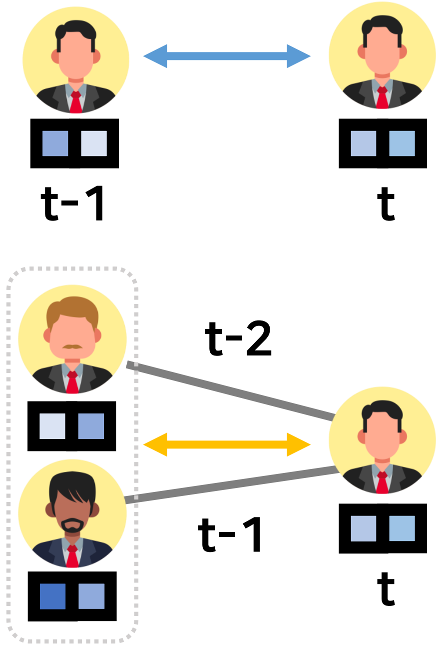
Proposed Method: SLADE

Self-supervised Learning for Anomaly Detection in Edge Streams

- Incremental method to minimize time delay
- Dynamic representation of interaction patterns
- Self-supervised learning without label supervision



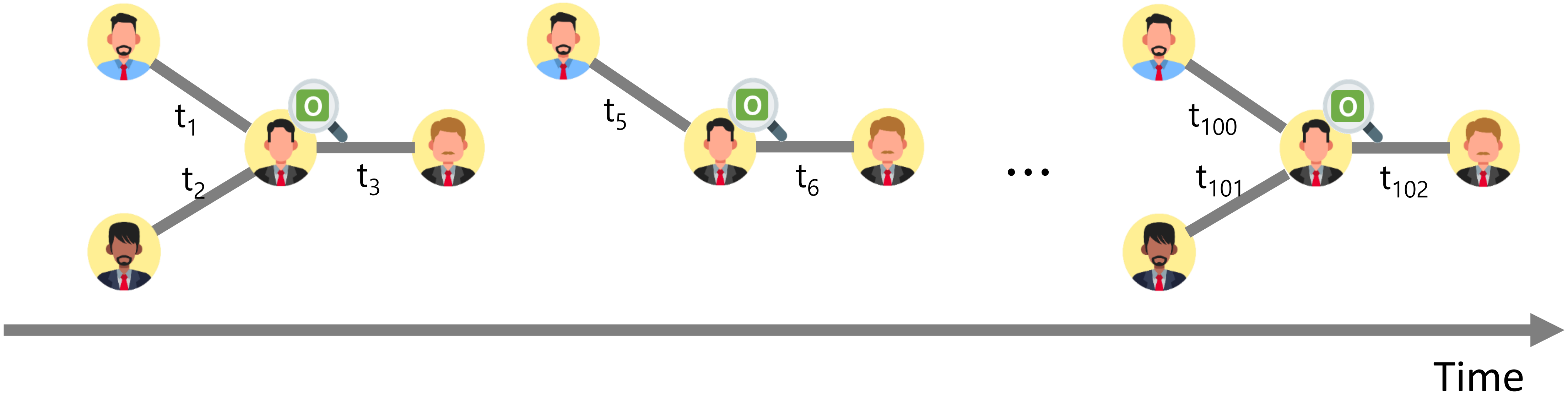




Assumptions for Nodes in a Normal state

A1. Stable Long-term Interaction Patterns

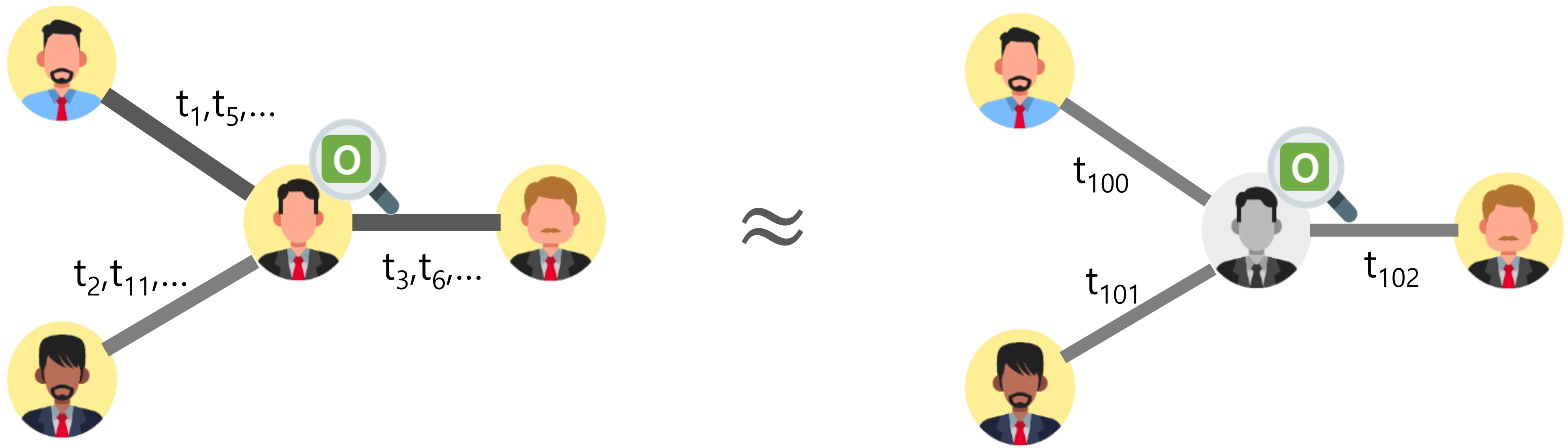
- Repetitively engage in similar interactions over a long-term period
- Show stable long-term interaction patterns in a temporal aspect



Assumptions for Nodes in a Normal state

A2. Potential for Restoration of Patterns

- Easily restore the long-term interaction patterns using recent interaction information
- Show structural similarities between long-term and short-term interaction patterns



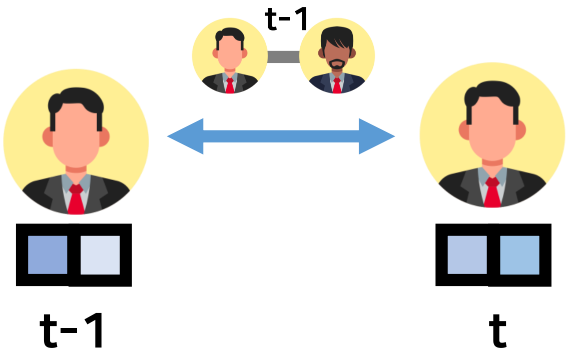
SLADE: (1) Self-supervised Tasks

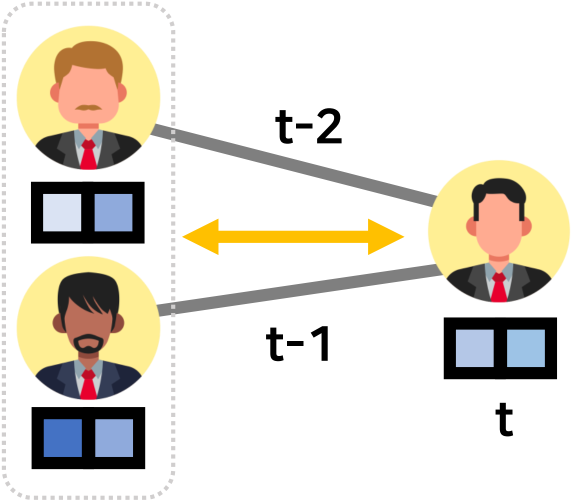
S1. Temporal Contrast

- Aligns with A1 (stable long-term interaction patterns)
- To minimize drift in dynamic node representations within short-time periods

S2. Memory Generation

- Aligns with A2 (potential for restoration of patterns)
- To generate dynamic node representations based on recent interactions

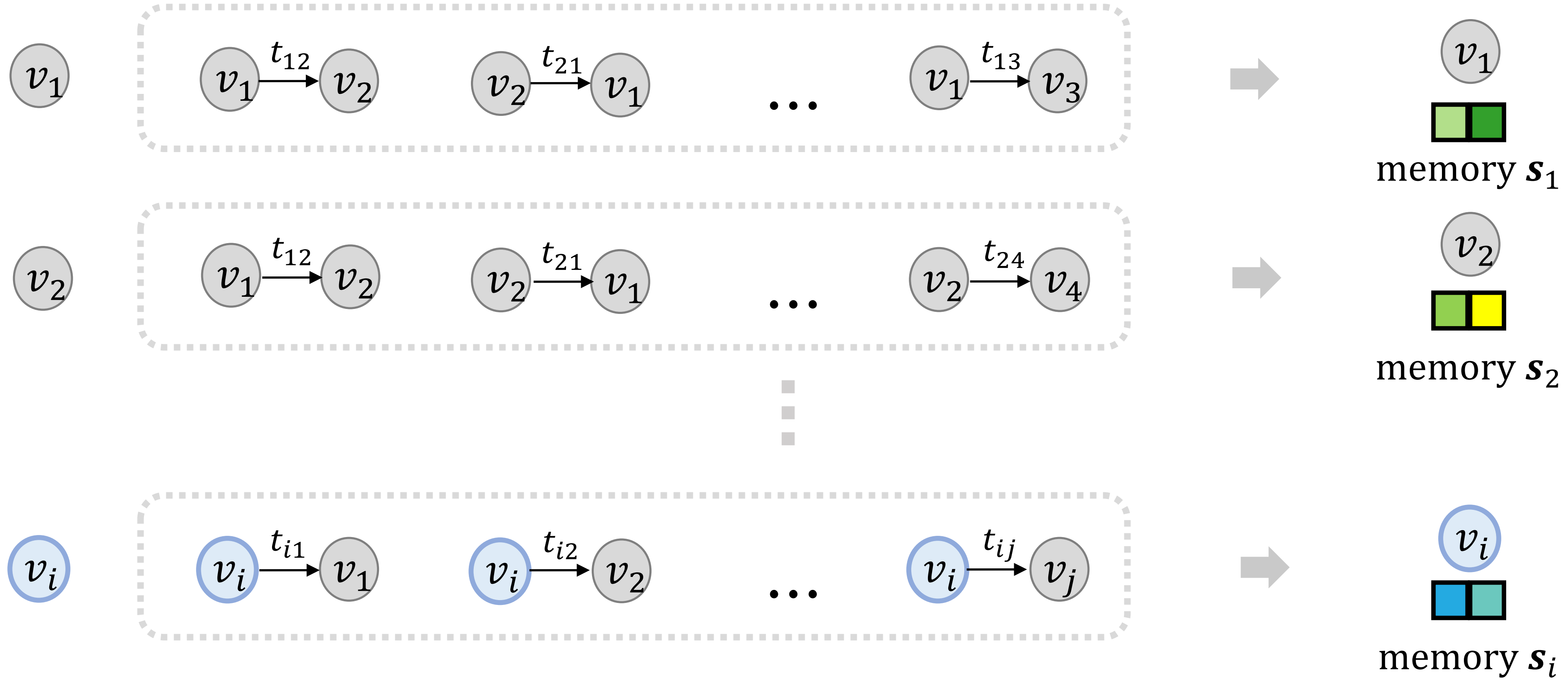




SLADE: (2) Model Architecture

Module1. Memory Module

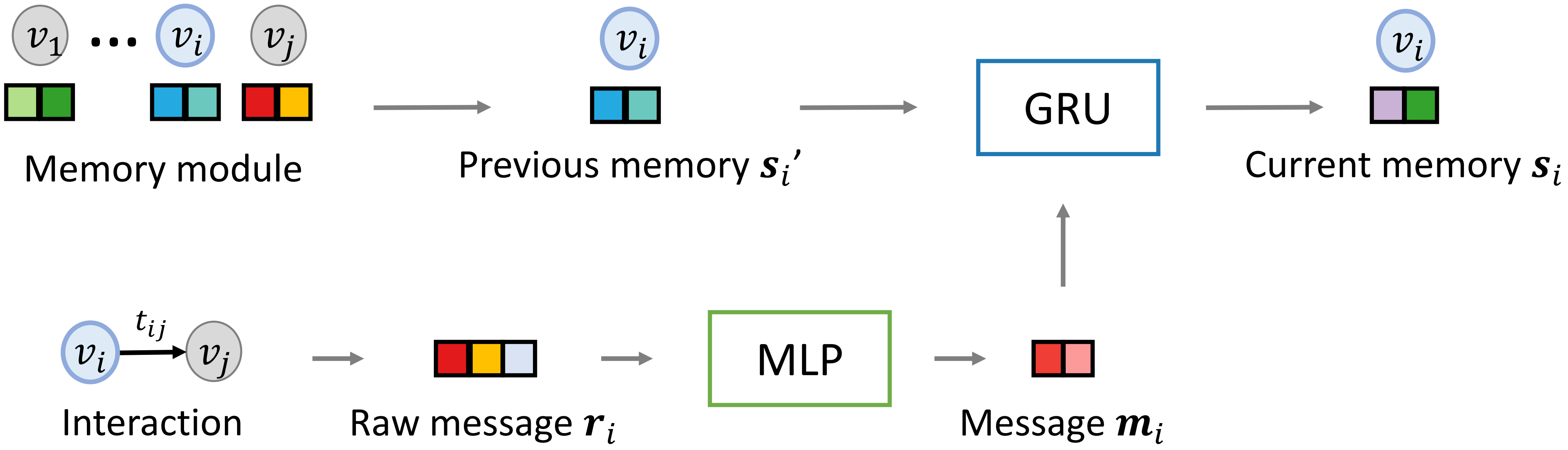
- Time-evolving parameter vectors representing the long-term interaction patterns



SLADE: (2) Model Architecture

Module2. Memory Updater

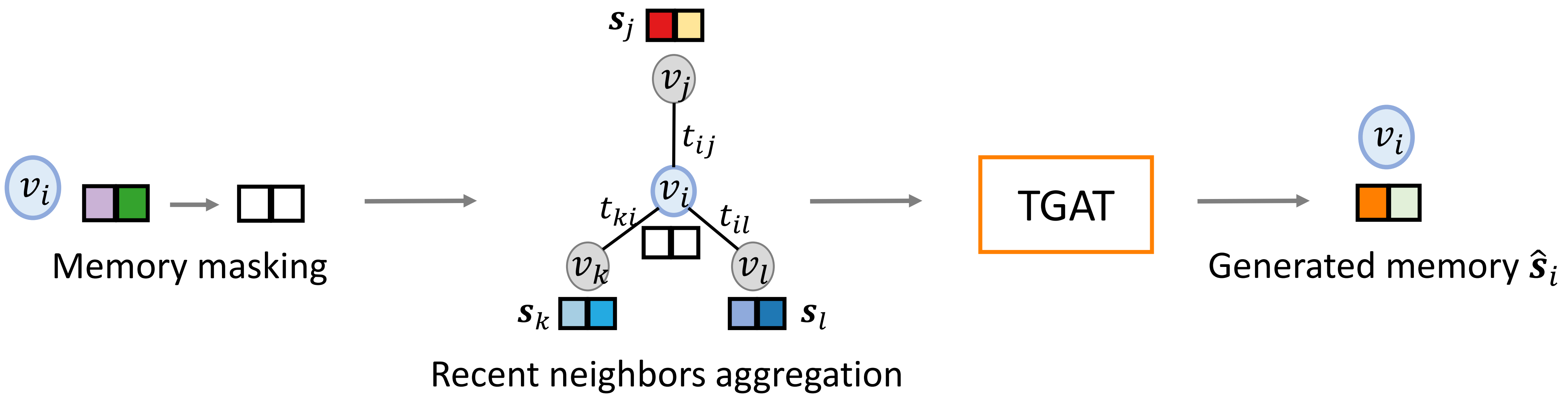
- To update its memory vector whenever a node participates in a new interaction
- Through this process, the long-term interaction patterns can be stored in memory



SLADE: (2) Model Architecture

Module3. Memory Generator

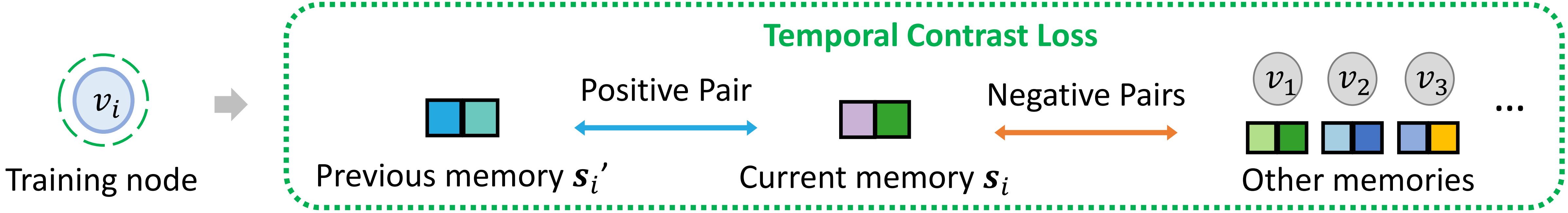
- To generate the memory vectors based on recent interactions after masking
- Generated memory represents short-term interaction patterns of a node



SLADE: (3) Training Objective

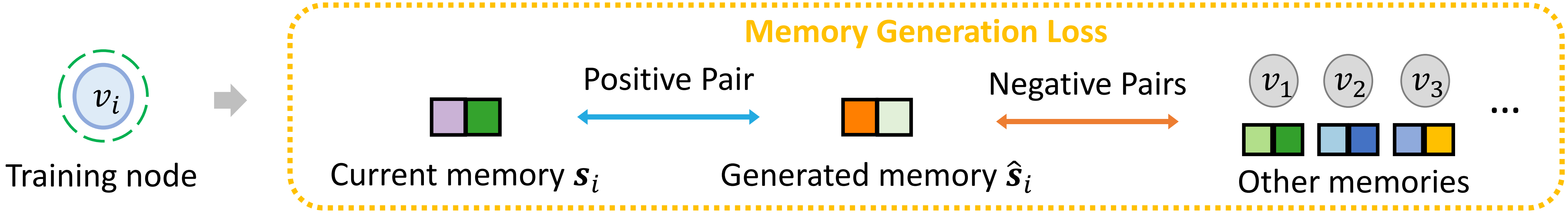
Temporal Contrast Loss

- To minimize drift in memories within a short time interval for S1 (temporal contrast)



Memory Generation Loss

- To accurately generate memories from recent interactions for S2 (memory generation)



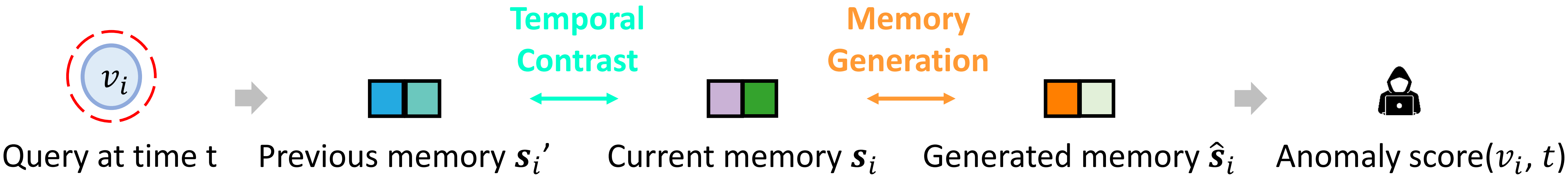
SLADE: (4) Anomaly Scoring

Temporal Contrast Score

- How much each node deviates from A1 (stable long-term interaction patterns)
  - ✓ How well the model performs S1 (temporal contrast) task for each node

Memory Generation Score

- How much each node deviates from A2 (potential for restoration of patterns)
  - ✓ How well the model performs S2 (memory generation) task for each node



Contents

- Introduction
- Problem Description
- Proposed Method: SLADE
- Experimental Results
- Conclusion

Research Questions

RQ1) Accuracy

- ✓ How accurately does SLADE detect anomalies, compared to other baselines?

RQ2) Speed

- ✓ Does SLADE exhibit a detection speed constant with respect to the graph size?

RQ3) Type Analysis

- ✓ Can SLADE accurately detect various types of anomalies?

Experimental Settings

Datasets

- ✓ 2 social networks (Wikipedia, Reddit)
- ✓ 2 financial networks (Bitcoin-alpha, Bitcoin-OTC)
- ✓ 2 email networks (Email-EU) with anomaly injection

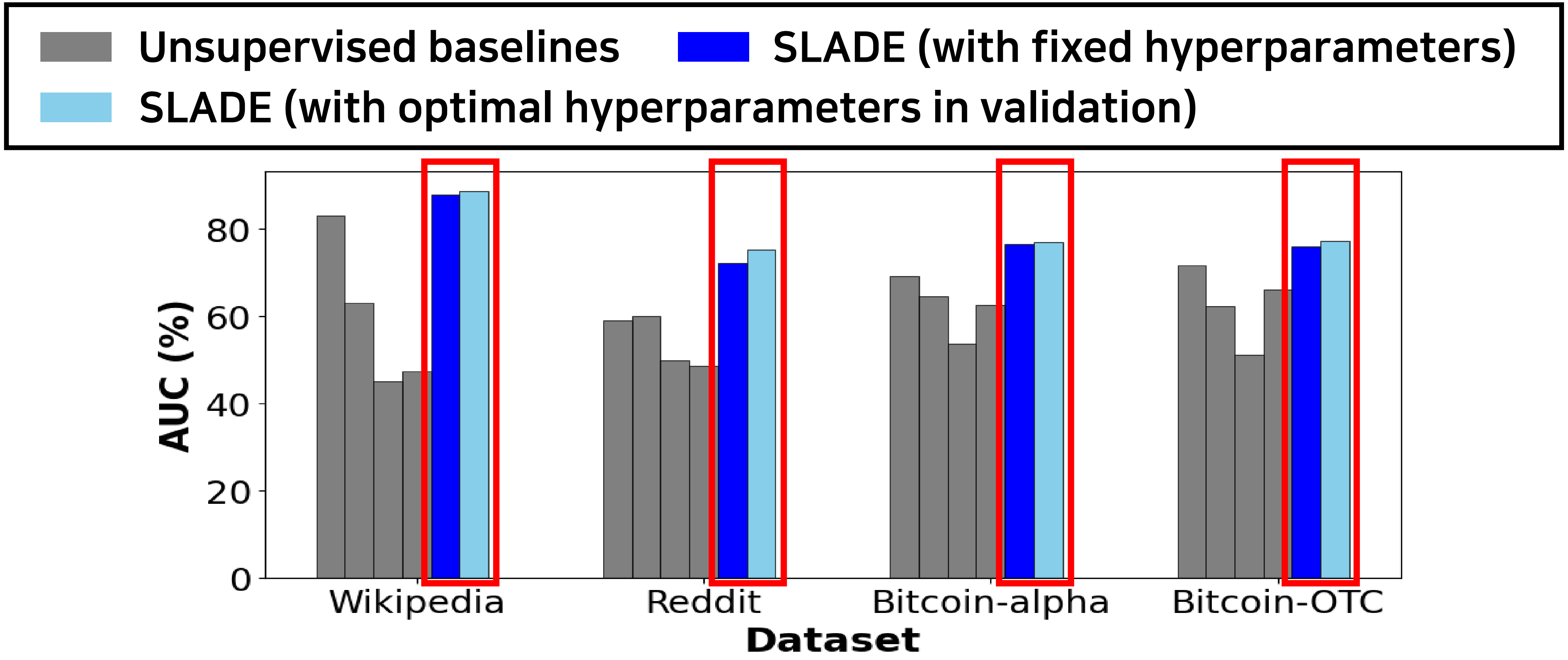
Baselines

- ✓ Unsupervised: 4 rule-based methods (SedanSpot, MIDAS-R, F-FADE, Anoedge-l)
- ✓ Supervised: 5 neural network-based methods (JODIE, Dyrep, TGAT, TGN, SAD)

RQ1) Accuracy

SLADE outperforms other baselines in dynamic anomaly detection

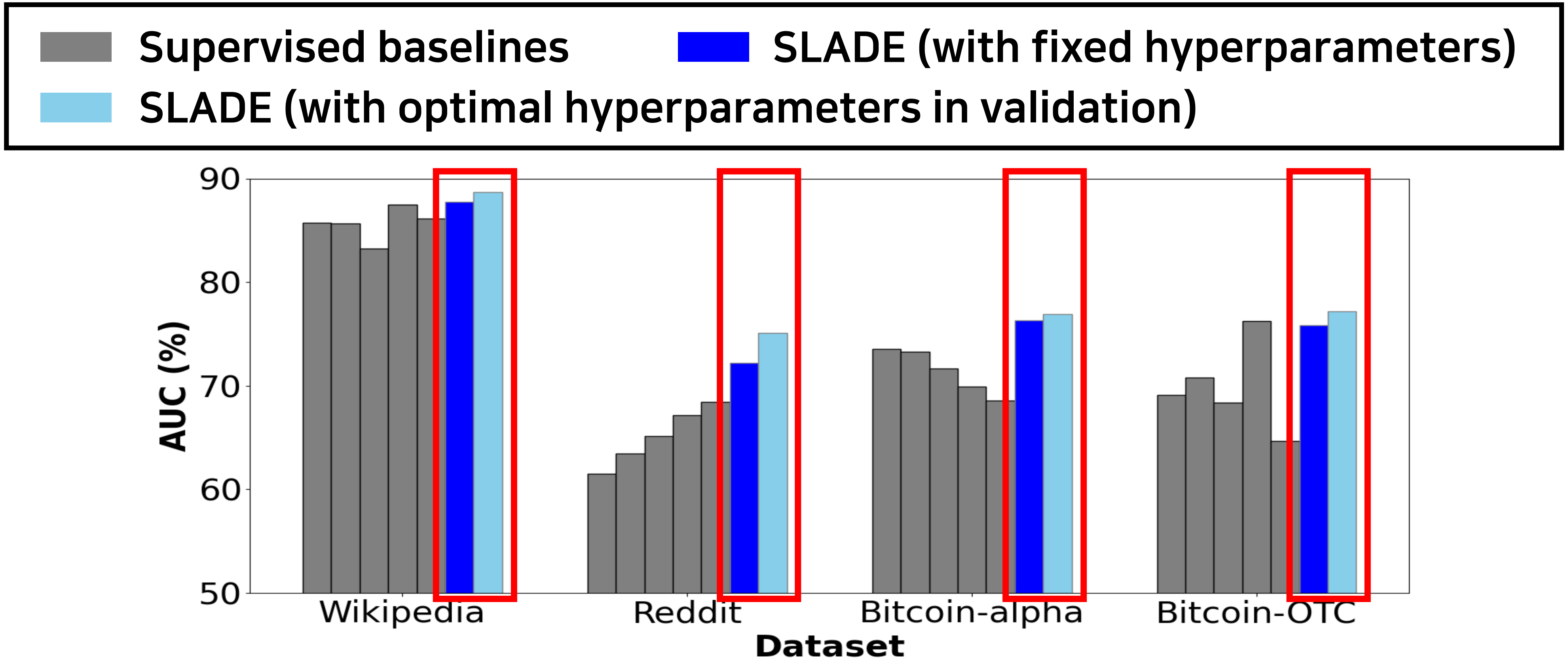
- Including unsupervised methods and supervised methods relying on label supervision



RQ1) Accuracy

SLADE outperforms other baselines in dynamic anomaly detection

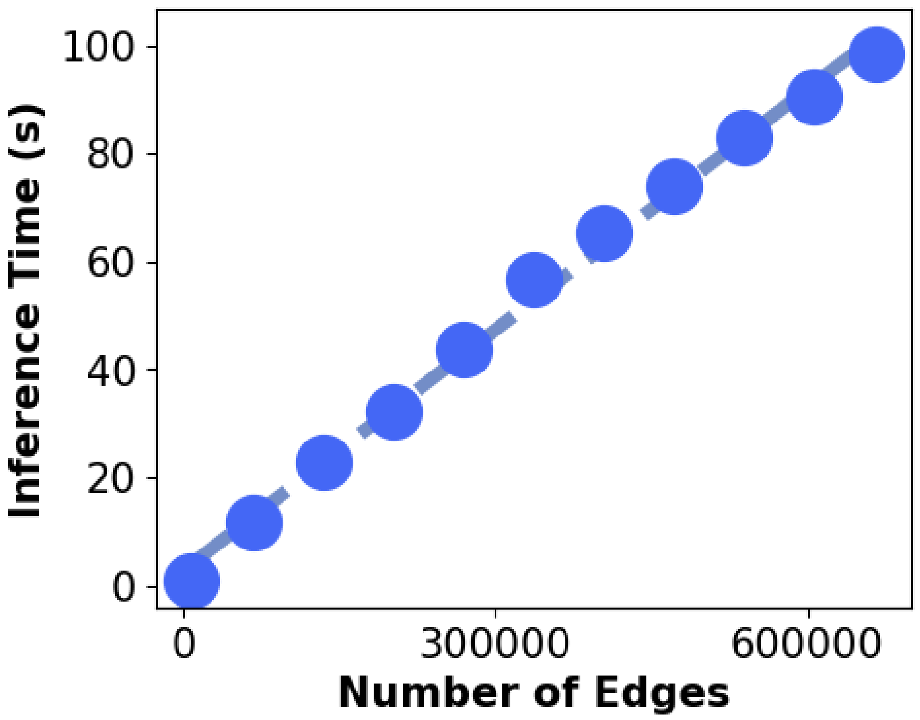
- Including unsupervised methods and supervised methods relying on label supervision

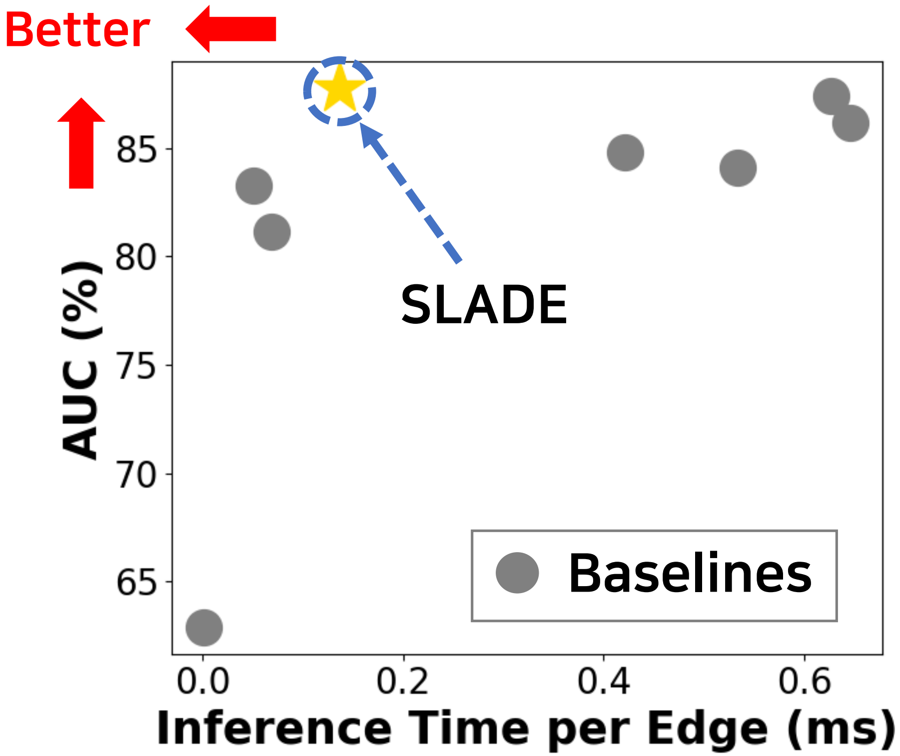


RQ2) Speed

SLADE minimize the detection time delay based on an incremental algorithm

- Maintains a constant inference time per edge regardless of graph size
- Offers the best trade-off between performance and inference time



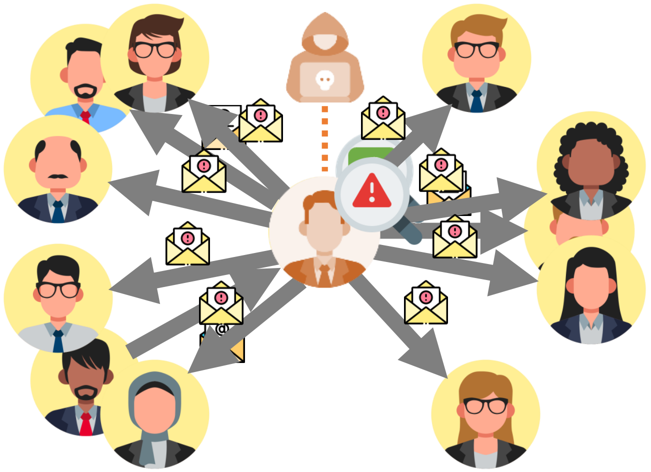


RQ3) Type Analysis

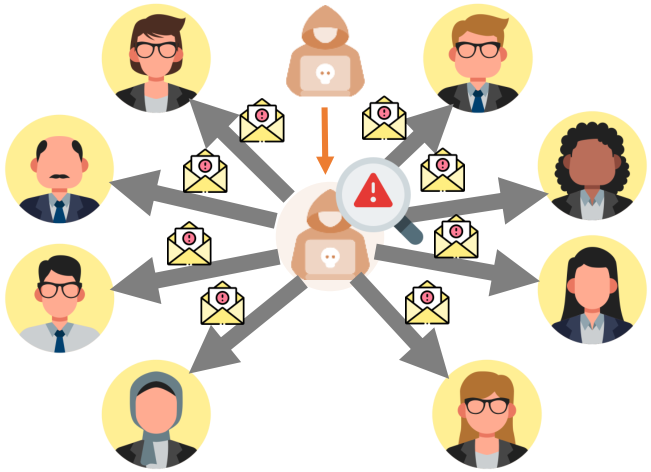
T1. Hijacked Anomalies (deviate from A1 and A2)

T2. New or Rarely-interacting Anomalies (deviate from A1 and A2)

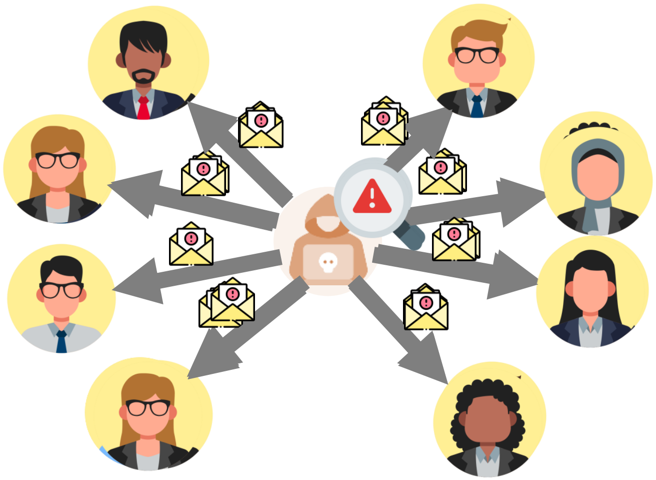
T3. Consistent Anomalies (deviate from A2)



Hijacked Anomalies



New Anomalies

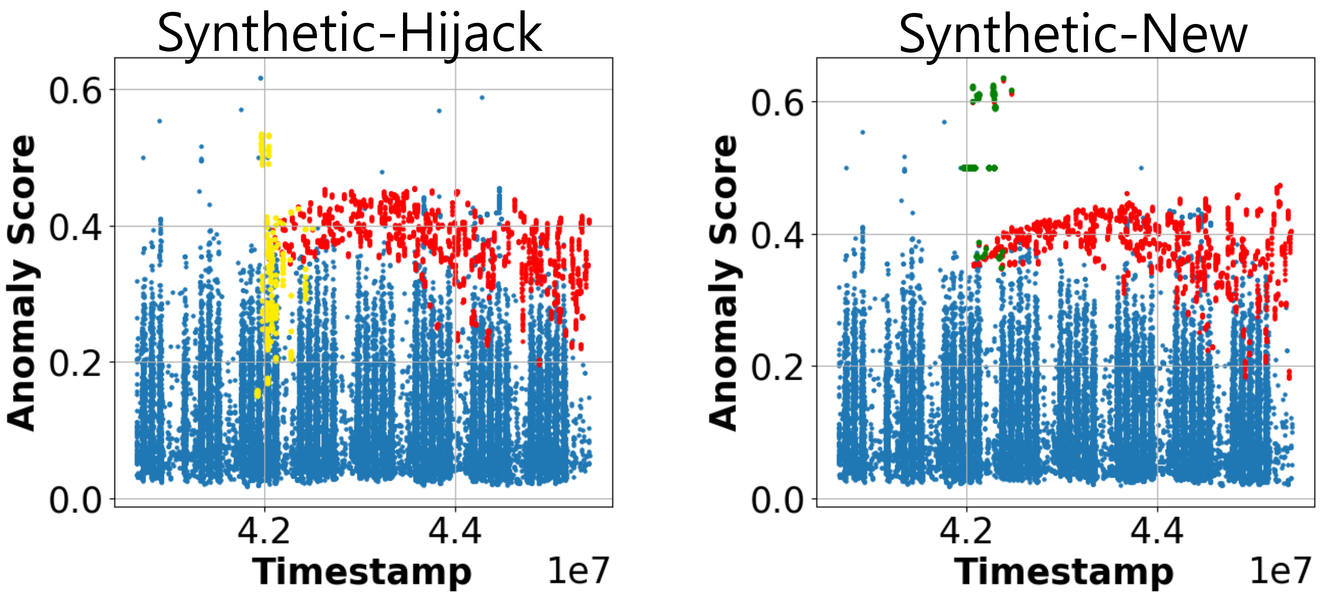


Consistent Anomalies

RQ3) Type Analysis

SLADE can effectively detect below anomaly types without label supervision

- Normal Nodes (blue)
- T1. Hijacked Anomalies (yellow)
- T2. New or Rarely Interacting Anomalies (green)
- T3. Consistent Anomalies (red)



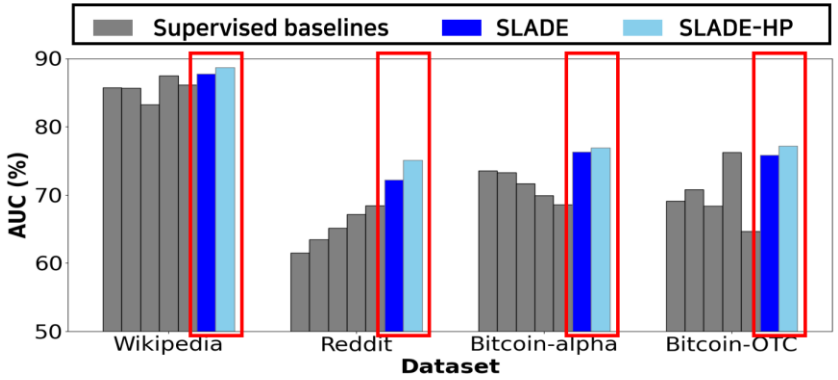
Contents

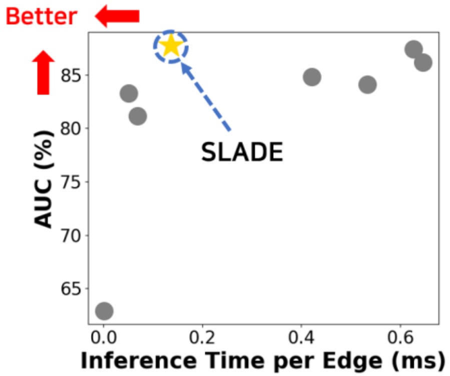
- Introduction
- Problem Description
- Proposed Method: SLADE
- Experimental Results
- Conclusion

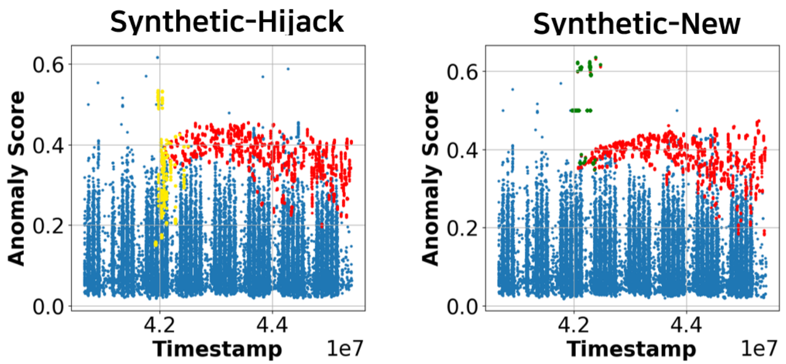
Conclusion

We propose SLADE for the rapid detection of dynamic anomalies in edge streams, without relying on labels

- ✓ Unsupervised: SLADE can detect dynamic anomalies without label supervision
- ✓ Effective: SLADE outperforms other baselines in dynamic anomaly detection
- ✓ Constant Inference Speed: SLADE requires a constant inference time per edge







Paper: https://doi.org/10.1145/3637528.3671845

Github: https://github.com/jhsk777/SLADE