Towards Securing Graph Neural Networks in MLaaS

Xingliang Yuan

School of Computing and Information Systems

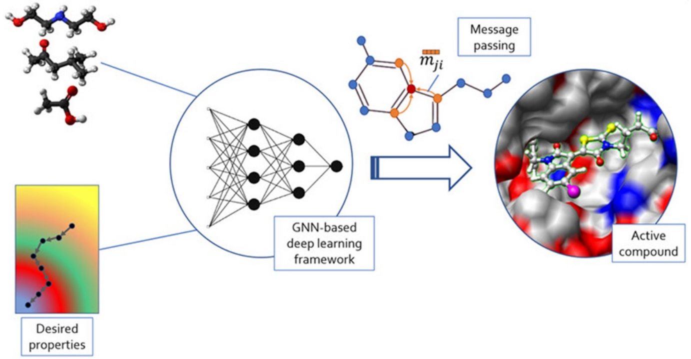
The University of Melbourne

29 June 2024 @ The 86th IFIP WG 10.4 Workshop

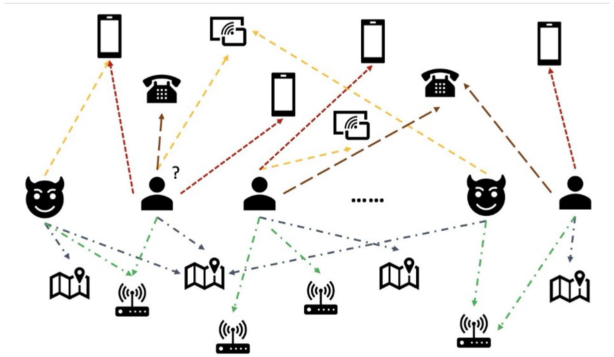
Outline

- Privacy-preserving Machine Learning for GNNs
- Addressing Training Data Misuse in GNNs

GNN: Powerful for Analysing Interconnected Information



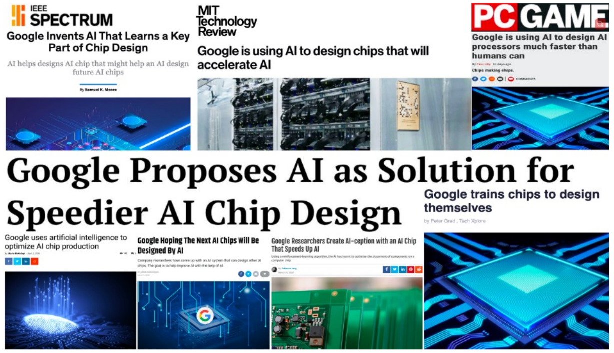
Drug Discovery



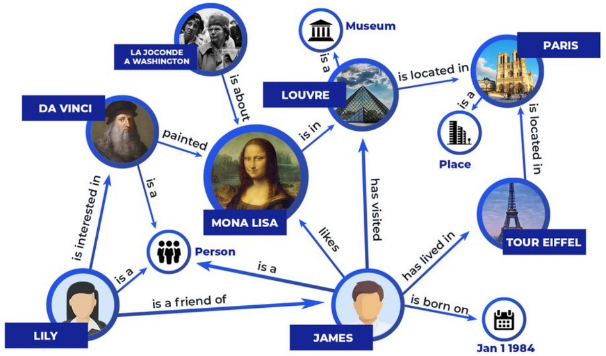
Fraud Detection

Social Networks

Self-driving



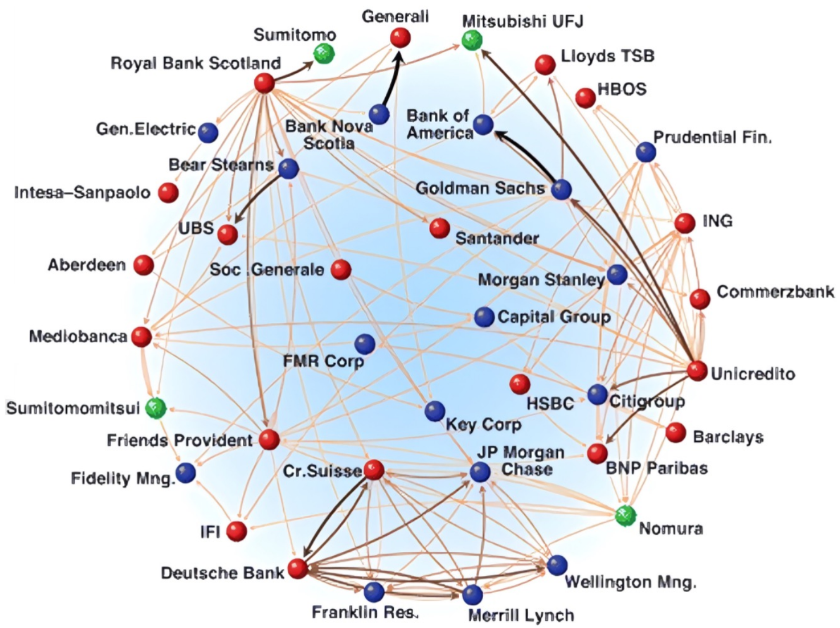
Chip Design



Knowledge Graph

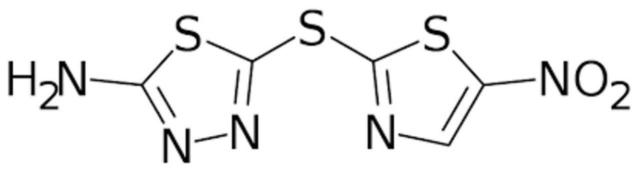
GNN Tasks

Node Classification
(Graph Convolutional Network [Kipf *et al.* (ICLR'17)])

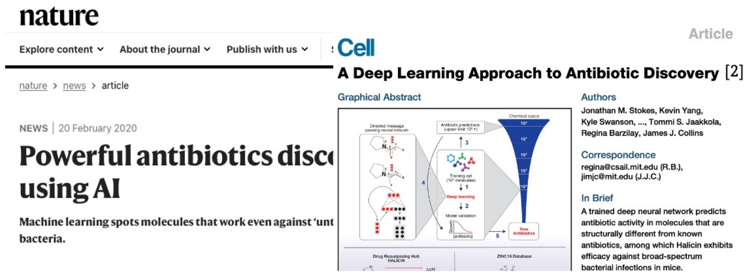


Bank

Graph Classification
(GraphSAGE [Hamilton *et al.* NIPS'17])



Halicin [3]


nature

Explore content · About the journal · Publish with us

nature > news > article

NEWS | 20 February 2020

Powerful antibiotics disc using AI

Machine learning spots molecules that work even against 'unt bacteria.

Cell

Article

A Deep Learning Approach to Antibiotic Discovery [2]

Graphical Abstract

Authors

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

Correspondence

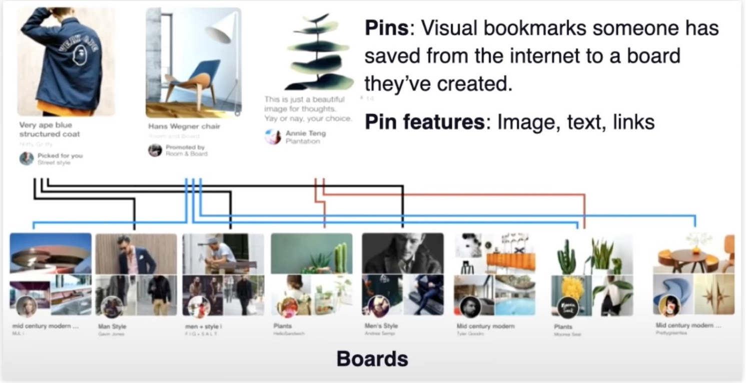
regina@csail.mit.edu (R.B.), jimjc@mit.edu (J.J.C.)

In Brief

A trained deep neural network predicts antibiotic activity in molecules that are structurally different from known antibiotics, among which Halicin exhibits efficacy against broad-spectrum bacterial infections in mice.

Drug discovery

Link Prediction
(GraphSAGE [Hamilton *et al.* NIPS'17])



Recommendation systems

GNNs in Machine Learning as a Service (MLaaS)

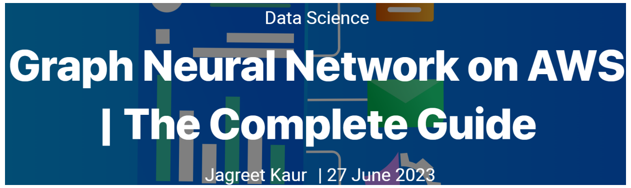
GNN is increasingly featured on MLaaS platforms

- **Amazon**: SageMaker Support for DGL
- **Google**: Neo4j & Google Cloud Vertex AI
- **Microsoft**: Azure ML Spektral

amazon | science

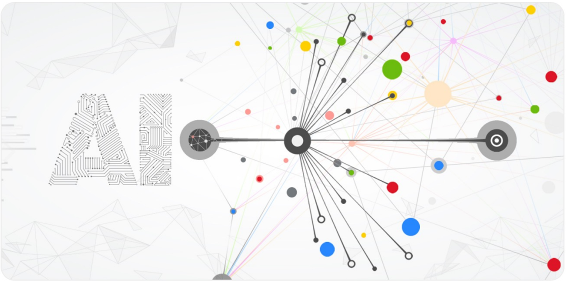
MACHINE LEARNING

How AWS uses graph neural networks to meet customer needs



Use graphs for smarter AI with Neo4j and Google Cloud Vertex AI

January 13, 2022



Ben Lackey

Director, Global Cloud Channel Architecture at Neo4j

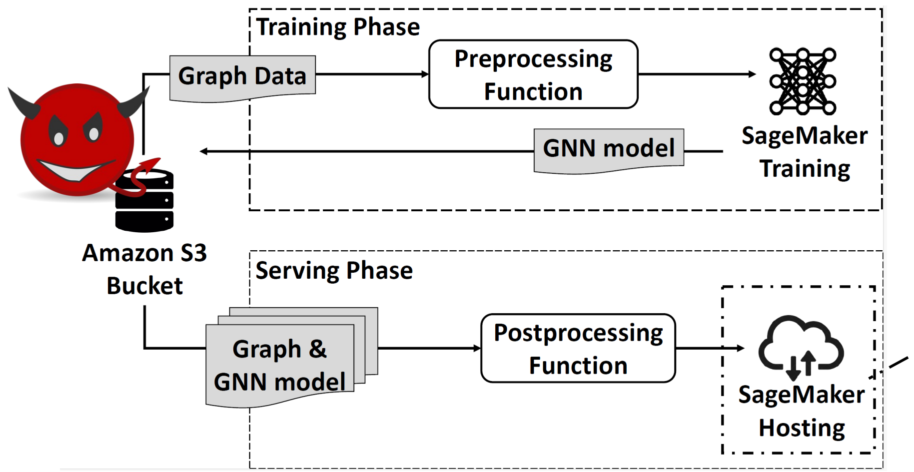
AWS Machine Learning Blog

Build a GNN-based real-time fraud detection solution using Amazon SageMaker, Amazon Neptune, and the Deep Graph Library

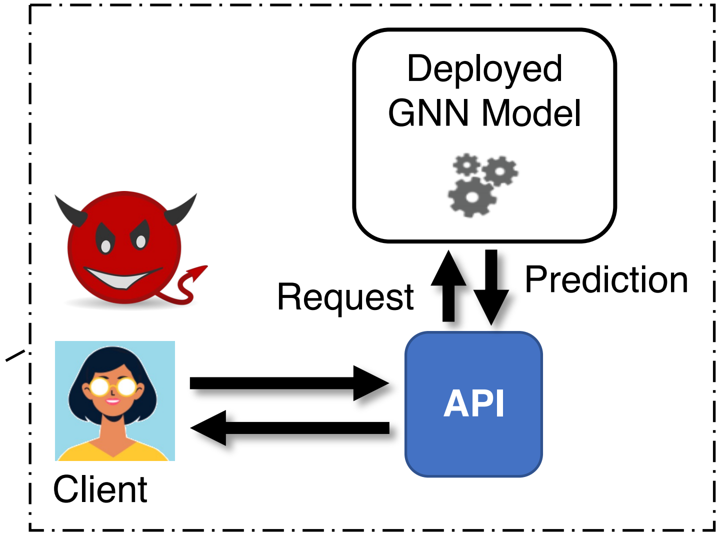
by Jian Zhang, Haozhu Wang, and Mengxin Zhu | on 11 AUG 2022 | in Amazon Neptune, Amazon SageMaker, Artificial Intelligence | Permalink | Comments | Share

Fraudulent activities severely impact many industries, such as e-commerce, social media, and financial services. Frauds could cause a significant loss for businesses and consumers. American consumers reported losing more than $5.8 billion to frauds in 2021, up more than 70% over 2020. Many techniques have been used to detect fraudsters—rule-based filters, anomaly detection, and machine learning (ML) models, to name a few.

Towards Securing GNNs in MLaaS



Architecture for GNN training and serving



Online prediction

- PPML for GNNs [XLLA**Y**Y24]: “OblivGNN: Oblivious Inference on Transductive and Inductive Graph Neural Network”, USENIX Security, 2024
- Detecting and mitigating data misuse in GNNs [WZYWXP**Y**24]: GraphGuard: Detecting and Counteracting Training Data Misuse in Graph Neural Networks, NDSS, 2024.
- Verifying GNN predictions [W**Y**WLXP24]: “Securing Graph Neural Networks in MLaaS: A Comprehensive Realization of Query-based Integrity Verification”, IEEE S&P, 2024
- Model extraction [WYP**Y**22]:“Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realisation”, AsiaCCS, 2022

OblivGNN: Oblivious Inference on Transductive and Inductive Graph Neural Network

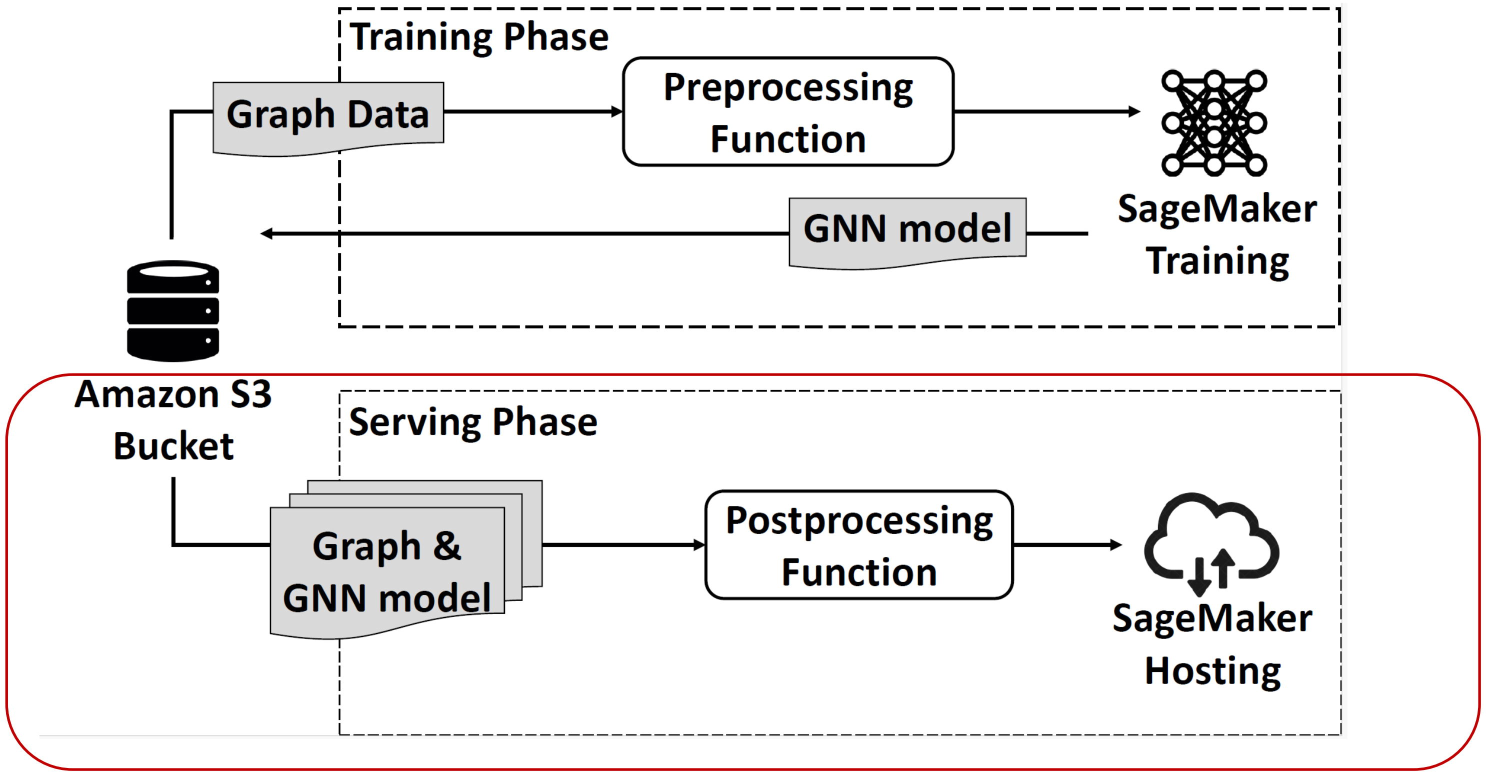
Zhibo Xu[1,2], Shangqi Lai[2], Xiaoning Liu[3], Alsharif Abuadbba[2], **Xingliang Yuan**[1,4], and Xun Yi[3]
[1]*Monash University*, [2]*CSIRO's Data61*, [3]*RMIT University*, [4]*The University of Melbourne*

In the 33rd USENIX Security Symposium

Outline

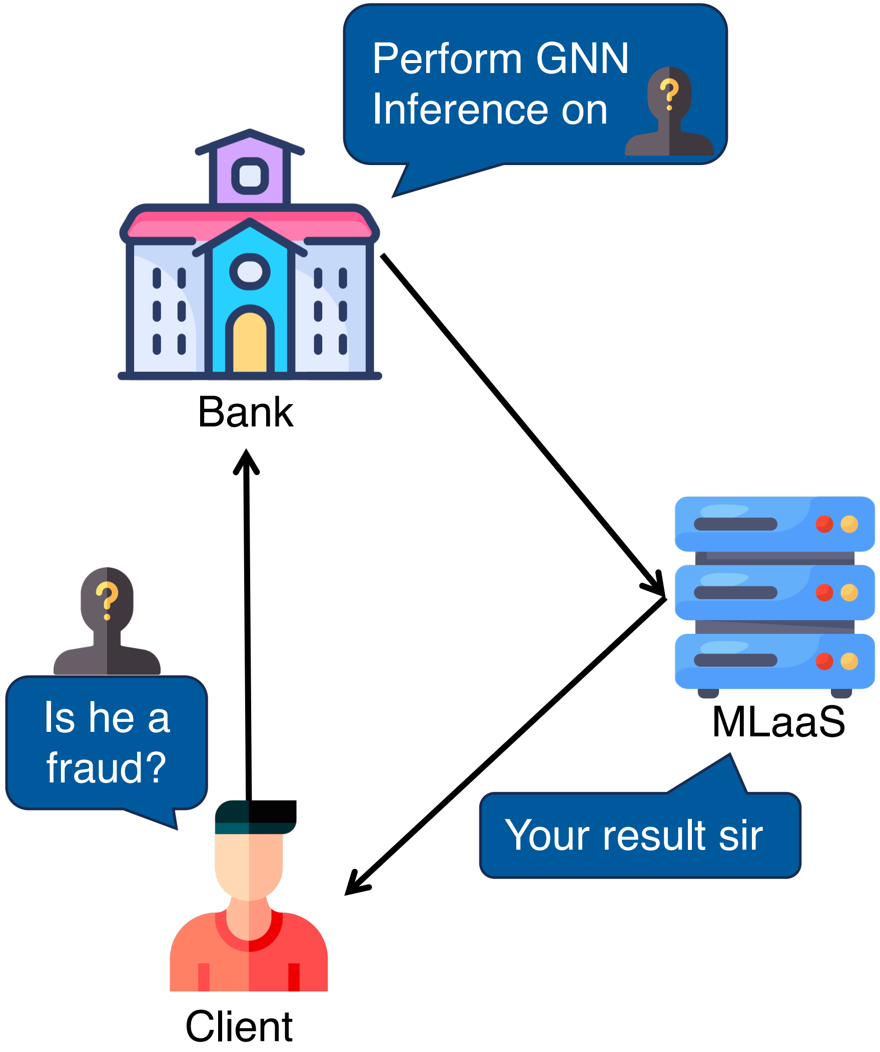
- Introduction
  - Motivation
  - Related Work
- Preliminaries
  - Graph Convolutional Networks and Node Classification
  - Function Secret Sharing
- Protocol
  - Strawman
  - OblivGNN
- Experiments
  - System

GNNs in Machine Learning as a Service (MLaaS)



AWS SageMaker for GNN training and inference

Privacy Concerns



Privacy Concerns:

- <u>Expose sensitive training/inference graph to MLaaS</u>
  - Collecting training graphs often requires a large amount of human, computing, and economic resource
  - Graph data is sensitive by nature, e.g., users' financial transactions, private friendships
- <u>Expose proprietary GNN model parameters to MLaaS</u>

Related Work in Privacy-Preserving Machine Learning

Traditional PPML Frameworks

Trident, Chameleon, Falcon, GAZELLE, MiniONN, Delphi, ABY^3, SecureML, BLAZE, XONN, AriaNN, CryptGPU, SecureNN

Cannot support graph-structured data

PPML for GNNs

SecGNN, CryptoGCN, LinGCN

- Do not offer full protection of graph structure information
  - Leak degree information
  - Do not support the full settings of GNN deployment
- Heavy computation cost (via FHE), heavy communication cost due to the large size of the graph

Outline

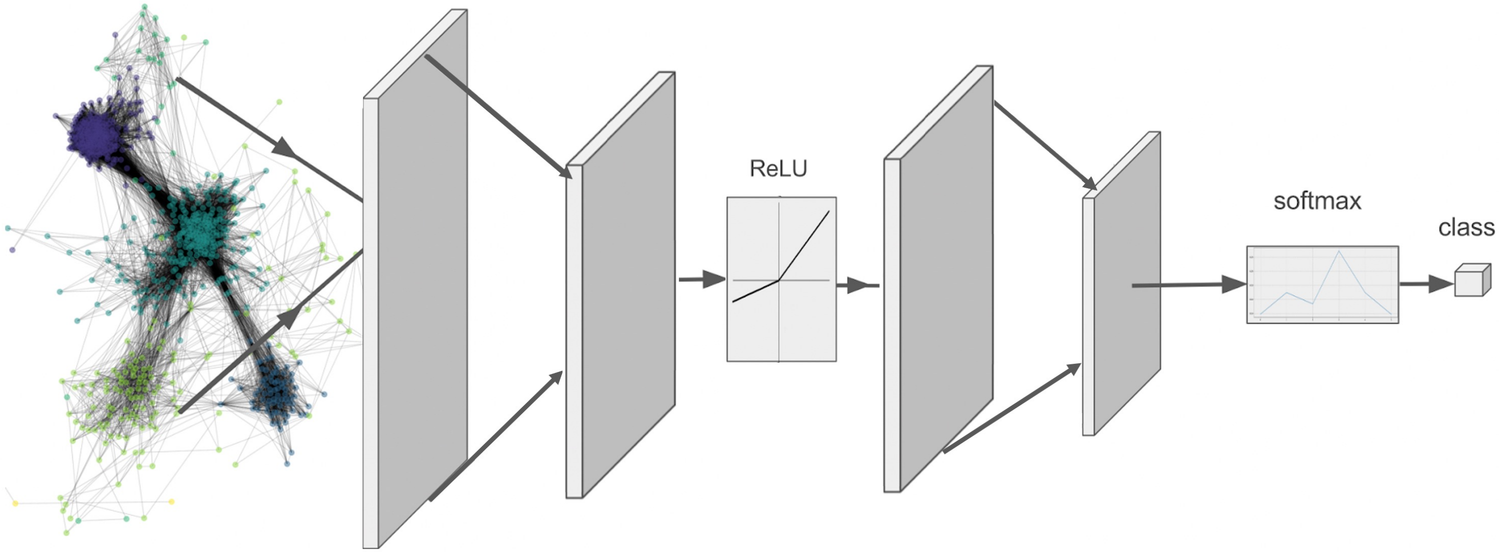
- Introduction
  - Graph Neural Networks
  - Machine Learning as a Service
  - Design Goal
  - Related Work
- Preliminaries
  - Graph Convolutional Networks and Node Classification
  - Function Secret Sharing
- Protocol
  - Strawman
  - OblivGNN
- Experiments
  - System

Preliminaries – Graph Convolutional Network

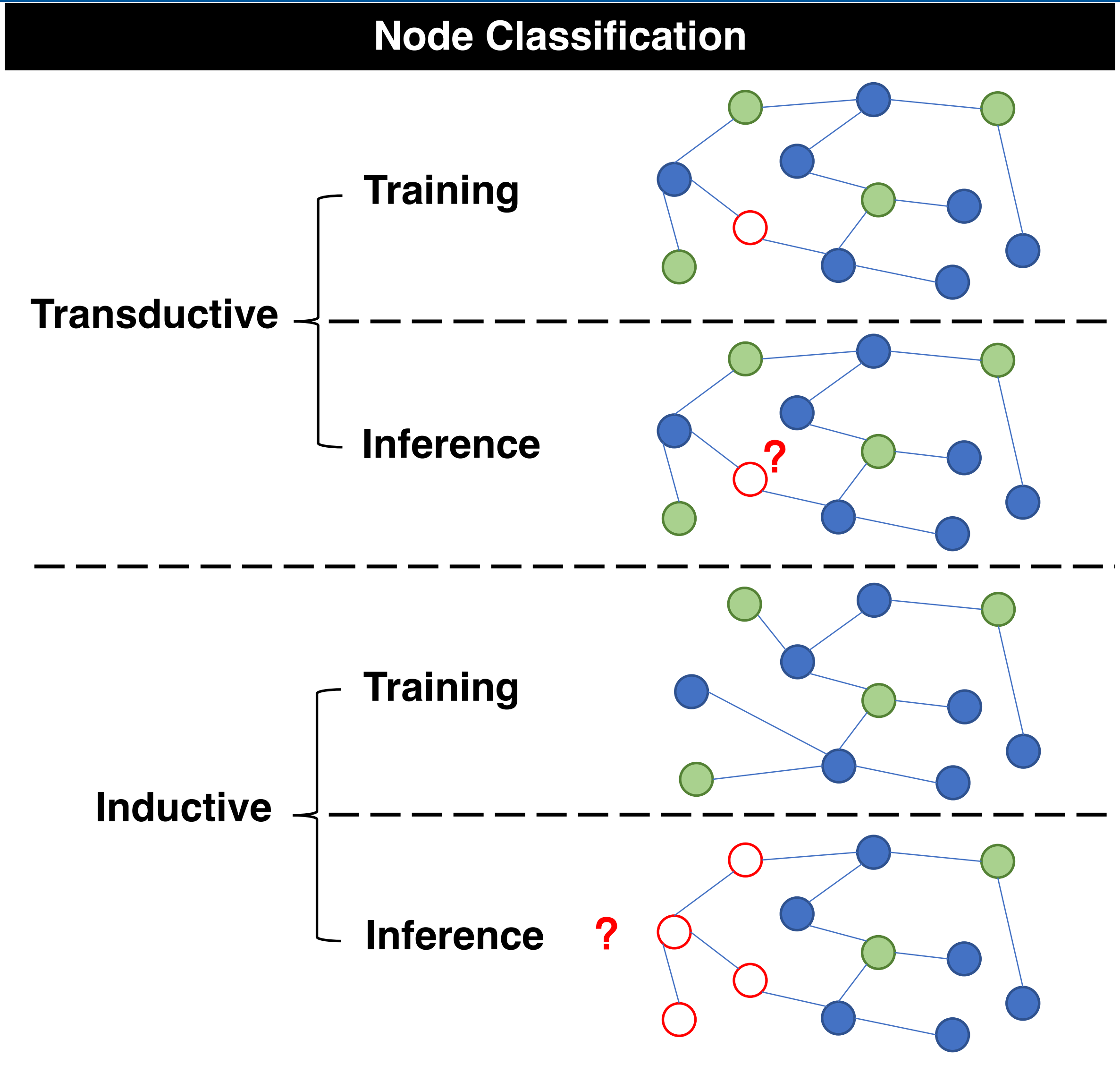
$\mathbf{Z}$=Softmax($\widehat{\mathbf{A}}$ ReLU($\widehat{\mathbf{A}}\widehat{\mathbf{F}}\mathbf{W_1}$)$\mathbf{W_2}$)

- $\mathbf{W_1}$ and $\mathbf{W_2}$ are two trainable weight matrixes
- $\widehat{\mathbf{A}}$ is the normalized adjacency matrix
- $\widehat{\mathbf{F}}$ is the normalized feature matrix

- Activation functions:
  - $\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
  - Softmax: $z_i = \frac{e^{x_i}}{\sum_{j\in[1,C]} e^{x_i}}, i \in [1, C]$



GNN Settings: Transductive and Inductive



Transductive:
- Unlabelled nodes and their connections *exist* in the *training*
- Graph for training and inference remains the same
- Query is a node ID/set of node IDs

Inductive:
- *Updated* nodes, features, connections appear in the *inference*
- Query is a node ID/set of node IDs

Function Secret Sharing

Function Secret Sharing [Boyle *et al.* CCS'16][Boyle *et al.* EUROCRYPT'21]

Distributed Point Functions:

$\mathrm{KeyGen}(\alpha, \beta) \rightarrow k_0, k_1$

$\mathrm{Eval}(k_b, x) \rightarrow [\![y]\!]_b$

$$\mathrm{Eval}(k_0, x) + \mathrm{Eval}(k_1, x) = \begin{cases} \beta, if\, x = \alpha \\ 0, otherwise \end{cases}$$

Equality Test:

$\mathrm{KeyGen}^{=}(\alpha = \gamma, \beta = 1) \rightarrow k_0^{=}, k_1^{=}$

$\mathrm{Eval}^{=}(k_b^{=}, x) \rightarrow [\![y]\!]_b$

$$\mathrm{Eval}^{=}(k_0, x') + \mathrm{Eval}^{=}(k_1, x') = \begin{cases} y = 1, if\, x' = \gamma \\ 0 \quad , otherwise \end{cases}$$

Comparison:

$\mathrm{KeyGen}^{<}(\alpha = \gamma, \beta = 1) \rightarrow k_0^{<}, k_1^{<}$

$\mathrm{Eval}^{<}(k_b^{<}, x) \rightarrow [\![y]\!]_b$

$$\mathrm{Eval}^{<}(k_0, x') + \mathrm{Eval}^{<}(k_1, x') = \begin{cases} y = 1, if\, x' \leq \gamma \\ 0 \qquad , if\, x' > \gamma \end{cases}$$

Arithmetic FSS:

Multiplication:

$\mathrm{KeyGen}^{\times}(\mathrm{g}^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow k_0^{\times}, k_1^{\times}$

$\mathrm{Eval}^{\times}(k_b^{\times}, x_1', x_2') \rightarrow \mathrm{g}_b^{\circ}\,(x_1 \times x_2) + r_{out}$

$$\mathrm{Eval}^{\times}(k_0^{\times}, x_1', x_2') + \mathrm{Eval}^{\times}(k_1^{\times}, x_1', x_2') = x_1 \times x_2 + r_{out}$$

Addition:

$\mathrm{KeyGen}^{+}(\mathrm{g}^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow k_0^{+}, k_1^{+}$

$\mathrm{Eval}^{+}(k_b^{+}, x_1', x_2') \rightarrow \mathrm{g}_b^{\circ}\,(x_1 + x_2) + r_{out}$

$$\mathrm{Eval}^{+}(k_0^{+}, x_1', x_2') + \mathrm{Eval}^{+}(k_1^{+}, x_1', x_2') = x_1 + x_2 + r_{out}$$

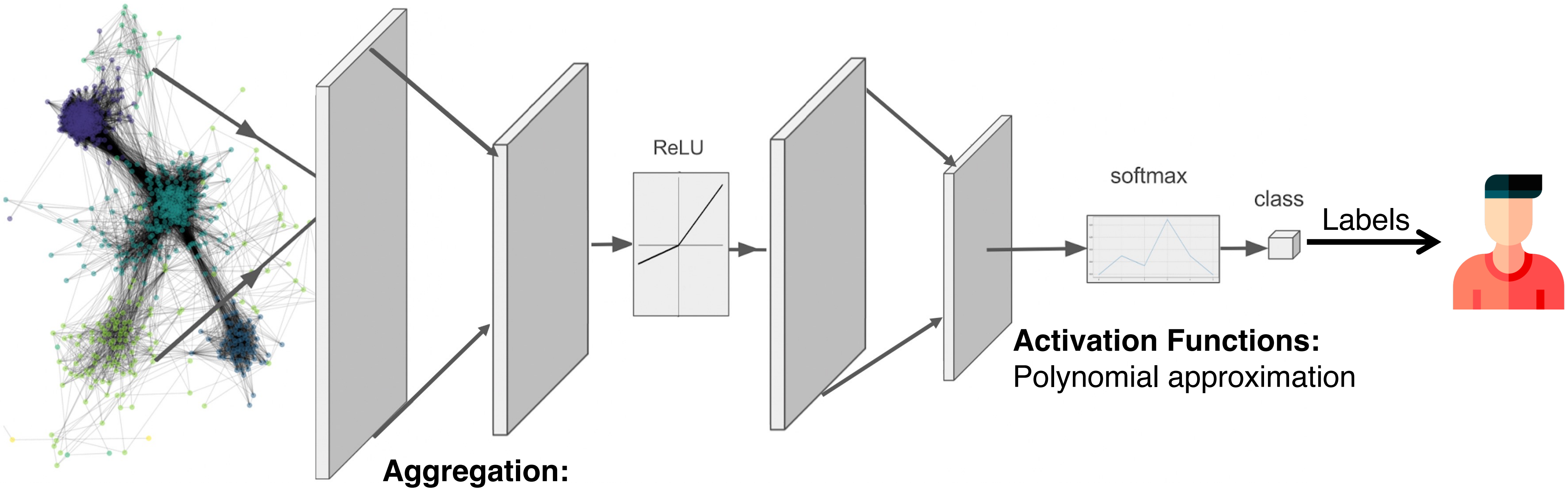
Outline

- Introduction
  - Graph Neural Networks
  - Machine Learning as a Service
  - Design Goal
  - Related Work
- Preliminaries
  - Graph Convolutional Networks and Node Classification
  - Function Secret Sharing
- Protocol
  - Strawman
  - OblivGNN
- Experiments
  - Microbenchmark
  - System

Strawman Approach

Transductive setting

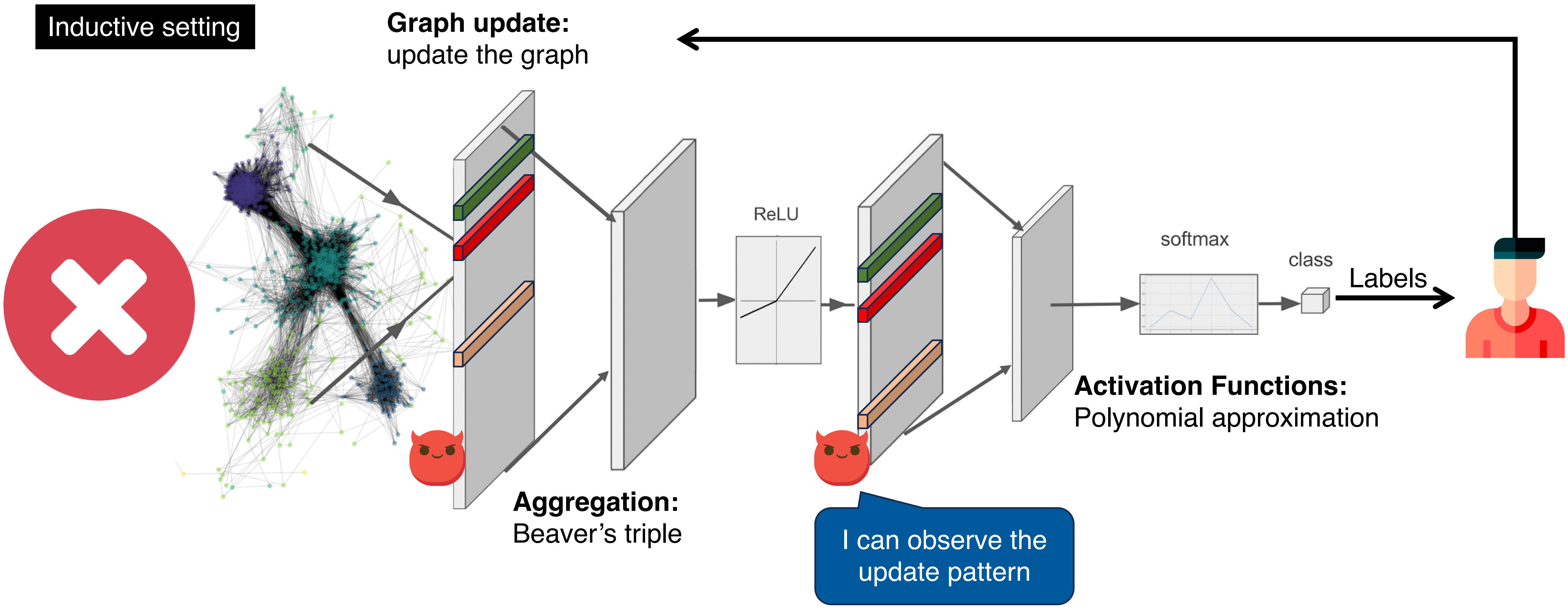
Sharing:
Additive Secret Sharing



Activation Functions:
Polynomial approximation

Aggregation:
Beaver's triple

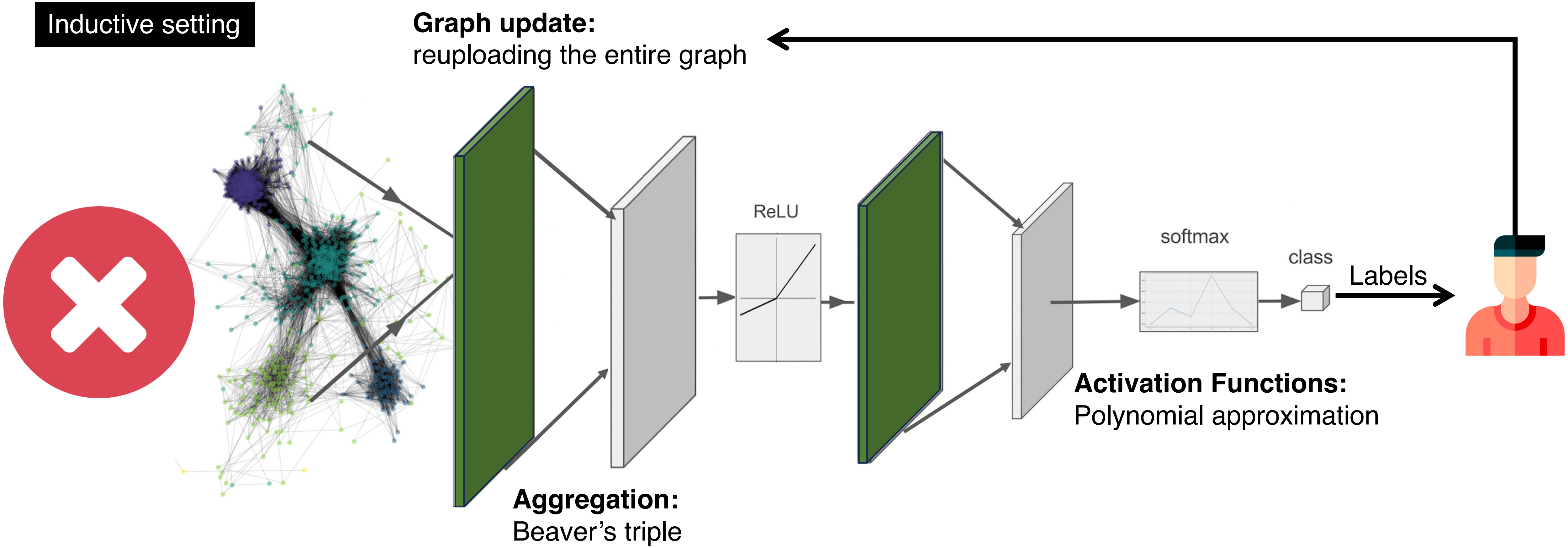
Strawman Approach



Problem: Leak graph update access, suffering from leakage attack [Falzon and Paterson, ESORICS'22]

Strawman Approach

Inductive setting



Problems: the communication cost is significant when re-uploading the updated graph.

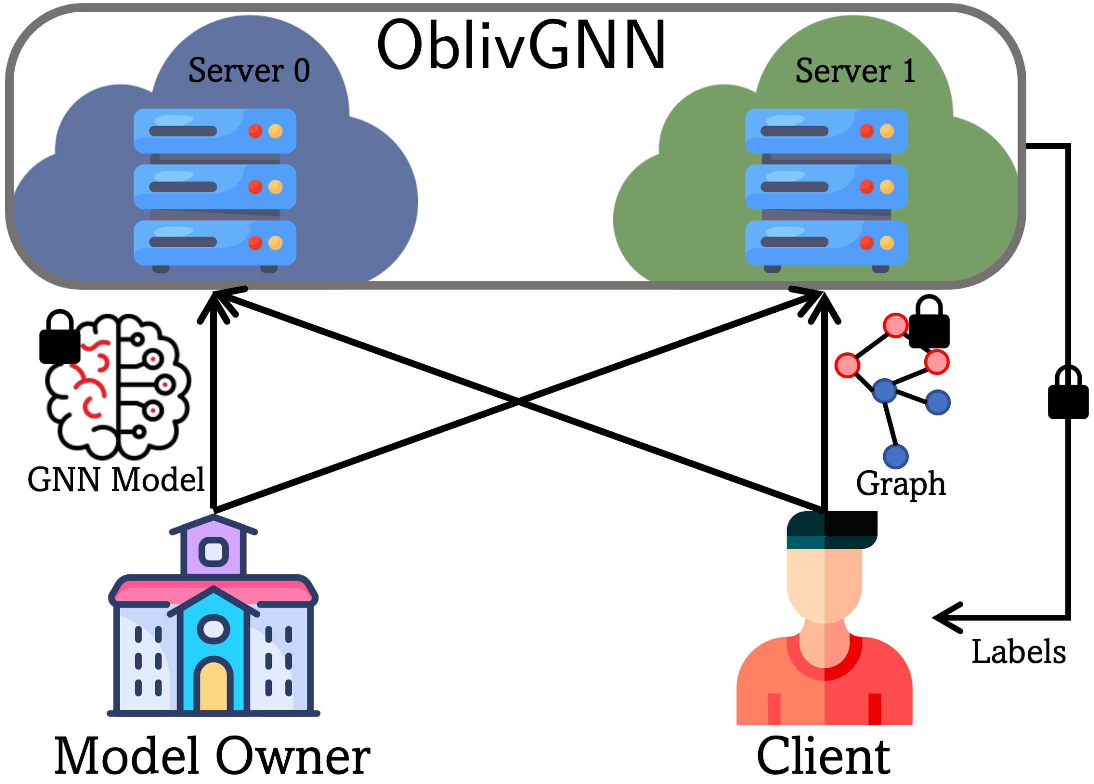
Research Questions

1. How to enable secure GNN inference in the *transductive* and *inductive* settings?

2. How to achieve data *obliviousness* with semi-honest security?

3. How to achieve high efficiency while achieving the above goals?

Protocol - Architecture

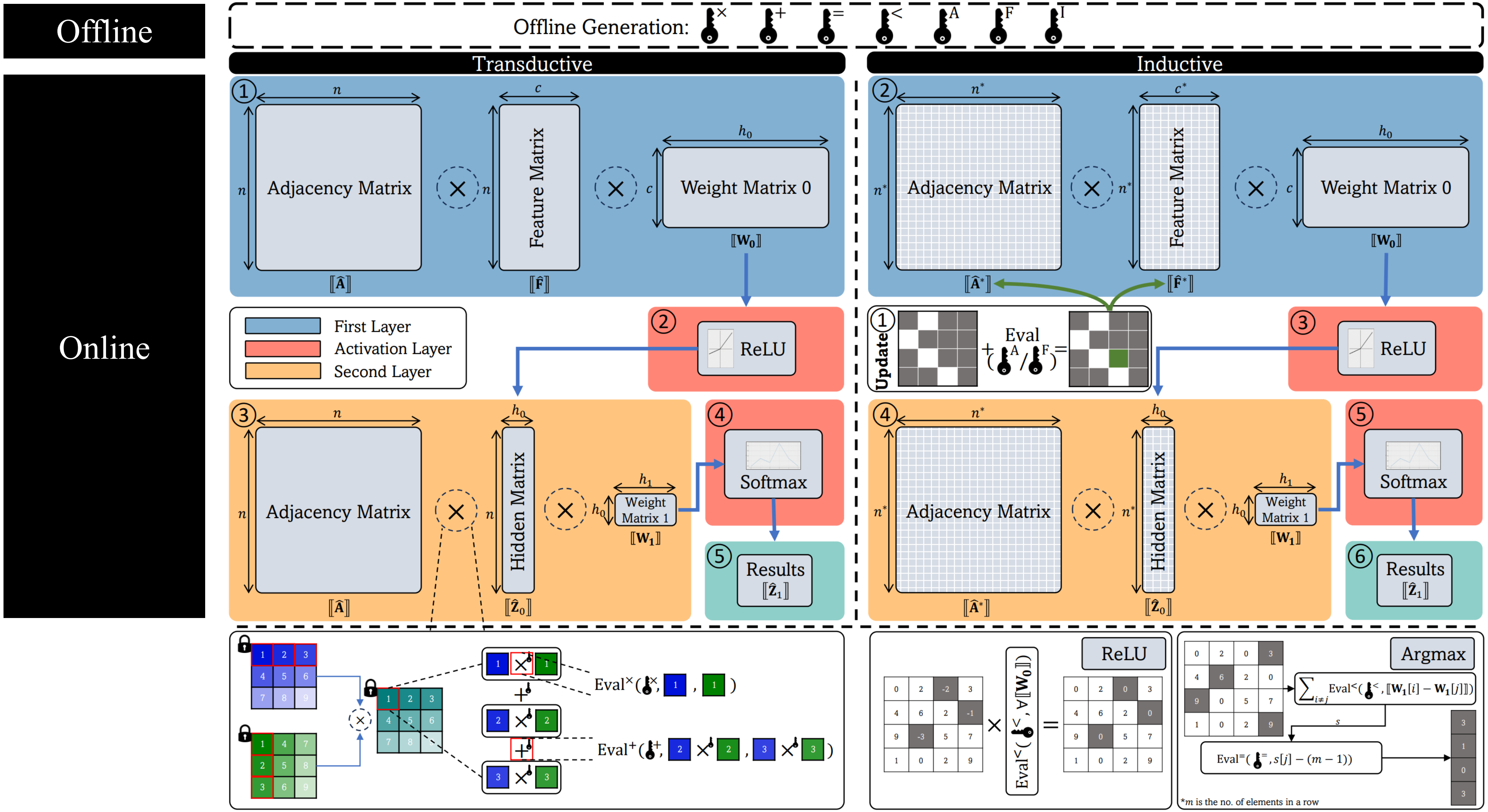


- Semi-honest Servers
- Non-colluding

Protocol – Security Guarantee

- Protect graph information
  - Adjacency Matrix $\hat{\mathbf{A}}$
  - Feature Matrix $\hat{\mathbf{F}}$
- Protect model information
  - Weight Matrix $\mathbf{W_0}$ and $\mathbf{W_1}$
- Protect access pattern to the graph structure $\hat{\mathbf{A}}$ and node feature $\hat{\mathbf{F}}$
- Protect client queries and inference results

OblivGNN Approach



OblivGNN Approach

Offline

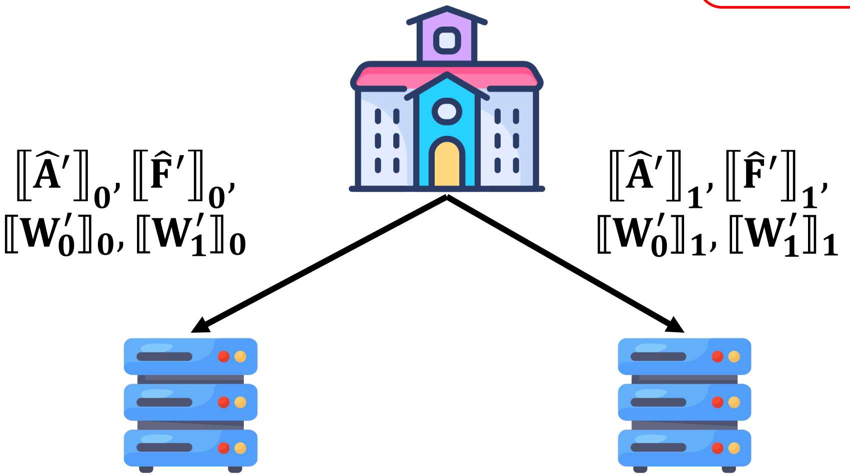
- <u>Masking & secret share GNN model</u>

Masks for Arithmetic FSS gates

Adjacency matrix: $\widehat{\mathbf{A}}' \leftarrow \widehat{\mathbf{A}} + r_{in}^1/r_{in}^2$

Feature matrix: $\hat{\mathbf{F}}' \leftarrow \hat{\mathbf{F}} + r_{in}^1/r_{in}^2$

Weight matrices: $\mathbf{W}_{0,1}' \leftarrow \mathbf{W}_{0,1} + r_{in}^1/r_{in}^2$



- Two servers need to *recover* the ASS shares (masked data) before operating FSS circuits
- Matrices stored in secret shares to facilitate update

OblivGNN Approach

Offline

- Key generation

FSS Key Pool Generation

Multiplication:

$\mathrm{KeyGen}^{\times}(g^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow k_0^{\times}, k_1^{\times}$: $\boxed{\text{FSS Multiplication keys}}$

$\mathrm{Eval}^{\times}(k_b^{\times}, x_1', x_2') \rightarrow g_b^{\circ}(x_1 \times x_2) + r_{out}$

Addition:

$\mathrm{KeyGen}^{+}(g^{\circ}, r_{in}^{1}, r_{in}^{2}, r_{out}) \rightarrow k_0^{+}, k_1^{+}$: $\boxed{\text{FSS Addition keys}}$

$\mathrm{Eval}^{+}(k_b^{+}, x_1', x_2') \rightarrow g_b^{\circ}(x_1 + x_2) + r_{out}$

DPF Key Pool Generation

$\left.\begin{array}{l} k^{\mathrm{A}}\text{: DPF Node Update keys} \\ k^{\mathrm{F}}\text{: DPF Feature Update keys} \\ k^{\mathrm{I}}\text{: DPF Client Query keys} \end{array}\right\}$ Online keys

$k^{=}$: DPF Equality Test keys

$k^{<}$: DPF Comparison keys

ObliviGNN Approach

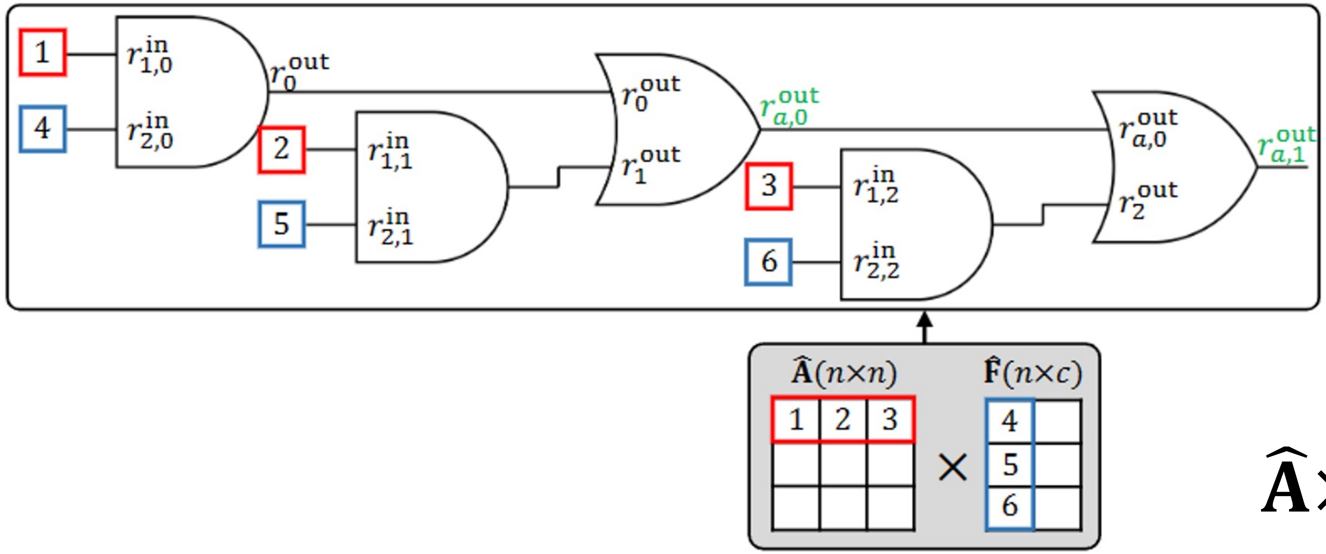
Online – Oblivious Aggregation

$$\mathrm{Eval}^{\times}(k_0^{\times}, x_1', x_2') + \mathrm{Eval}^{\times}(k_1^{\times}, x_1', x_2') = x_1 \times x_2 + r_{out}$$

$$\mathrm{Eval}^{+}(k_0^{+}, x_1', x_2') + \mathrm{Eval}^{+}(k_1^{+}, x_1', x_2') = x_1 + x_2 + r_{out}$$

Example:

$$x_1' = x_1 + r_{in}^1$$
$$x_2' = x_2 + r_{in}^2$$



$\hat{\mathbf{A}} \times \hat{\mathbf{F}}$ $\times$ $\mathbf{W}$ …

$\boxed{1} \times \boxed{4} = \mathrm{Eval}^{\times}(k^{\times}, 1, \boxed{4})\ \boxed{}$

$\boxed{2} \times \boxed{5} = \mathrm{Eval}^{\times}(k^{\times}, 2, \boxed{5})\ \boxed{}$

$\mathrm{Eval}^{+}(k^{+},\ )$

$\boxed{3} \times \boxed{6} = \mathrm{Eval}^{\times}(k^{\times}, 3, \boxed{6})\ \boxed{}$

$\mathrm{Eval}^{+}(k^{+},\ )$

= output+r_{out}

OblivGNN Approach

Online – Oblivious Activation Function

ReLU

[Ryffel *et al.* PoPETs'22]

1) Each P_b: DPF.Comp$(z[i])$
2) Each P_b: OblivBitFlip

1) $[\![\mathfrak{b}]\!]_0 = \text{Eval}(k_0, [\![z]\!]_0)$, $[\![\mathfrak{b}]\!]_1 = \text{Eval}(k_1, [\![z]\!]_1)$
2) $[\![\mathfrak{b}']\!]_0 = 0 - [\![\mathfrak{b}]\!]_0$, $[\![\mathfrak{b}']\!]_1 = 1 - [\![\mathfrak{b}]\!]_1$
3) $\mathfrak{b}' = [\![\mathfrak{b}']\!]_0 + [\![\mathfrak{b}']\!]_1 = 1 - ([\![\mathfrak{b}]\!]_0 + [\![\mathfrak{b}]\!]_1) = 1 - \mathfrak{b}$

Softmax

[Mohassel *et al.* IEEE S&P'21]
[Keller *et al.* CCS'20]

$$\mathbf{Z}[i] := \begin{cases} \frac{\text{OblivReLU}(z[i])}{\sum_i \text{OblivReLU}(z[i])}, & \text{if } \sum_i \text{OblivReLU}(z[i]) > 0 \\ 1/L, & \text{otherwise} \end{cases}$$

Argmax

[Ryffel *et al.* PoPETs'22]

1) Each P_b: $[\![s[j]]\!]_b \leftarrow \sum_{i \neq j} \text{DPF.Comp}([\![z[i] - z[j]]\!]_b)$ Finding the largest element
2) Each P_b: $[\![z'[j]]\!]_b \leftarrow \text{DPF.Equa}([\![s[j] - (L-1)]\!]_b)$ Locating the largest element

OblivGNN Approach – *Inductive* Protocol

Online – Oblivious Graph Update

New Node Insertion

- Introduce *new* nodes
- Do NOT modify the existing graph

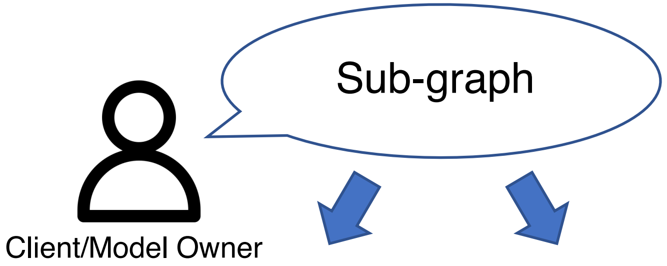
Existing Graph Update

- Modify the *existing* graph
  - Obliviously update adjacency matrix
  - Obliviously update feature matrix

OblivGNN Approach – *Inductive* Protocol

Online – Oblivious Graph Update

New Node Insertion



Sub-graph Adjacency Matrix

Sub-graph Feature Matrix

- Protect connections of new nodes
- Leak graph size

	v_0	v_1	$v_{\dots}$	v_n
v_0	1	1	…	0
v_1	1	1	…	0
$v_{\dots}$	…	…	…	…
v_n	0	0	…	1

Adjacency Matrix (secret shared)

Append shares →

	v_0	v_1	$v_{\dots}$	v_n	v_{n+1}
v_0	1	1	…	0	1
v_1	1	1	…	0	1
$v_{\dots}$	…	…	…	…	…
v_n	0	0	…	1	1
v_{n+1}	1	1	…	1	1

Inserted Adjacency Matrix

	f_0	$f_{\dots}$	f_c
v_0	1	…	0
v_1	1	…	1
$v_{\dots}$	…	…	…
v_n	0	…	1

Feature Matrix (secret shared)

Append shares →

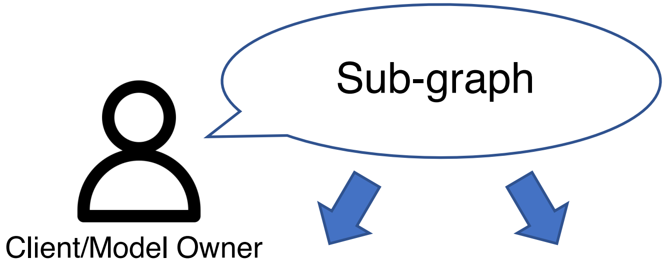
	f_0	$f_{\dots}$	f_c
v_0	1	…	0
v_1	0	…	1
$v_{\dots}$	…	…	…
v_n	0	…	1
v_{n+1}	1	…	0

Inserted Feature Matrix

OblivGNN Approach – *Inductive* Protocol

Online – Oblivious Graph Update

Existing Graph Update



Sub-graph Adjacency Matrix

Sub-graph Feature Matrix

$\mathrm{KeyGen}(0||1,1) \rightarrow k_0^{\mathrm{A}}, k_1^{\mathrm{A}}$

$\mathrm{KeyGen}(1||0,1) \rightarrow k_0^{\mathrm{A}}, k_1^{\mathrm{A}}$

	0	1	…	n
0	1	0	…	0
1	0	1	…	0
…	…	…	…	…
n	0	0	…	1

Adjacency Matrix $[\widehat{\mathbf{A}}]$ (secret shared)

$+$

$\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, 0||0)$

$\leftarrow \boxed{\mathrm{Eval}(k_b^{\mathrm{A}}, 1||0)}$

$\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, \ldots)$

$\leftarrow \mathrm{Eval}(k_b^{\mathrm{A}}, n||0)$

$\Rightarrow$

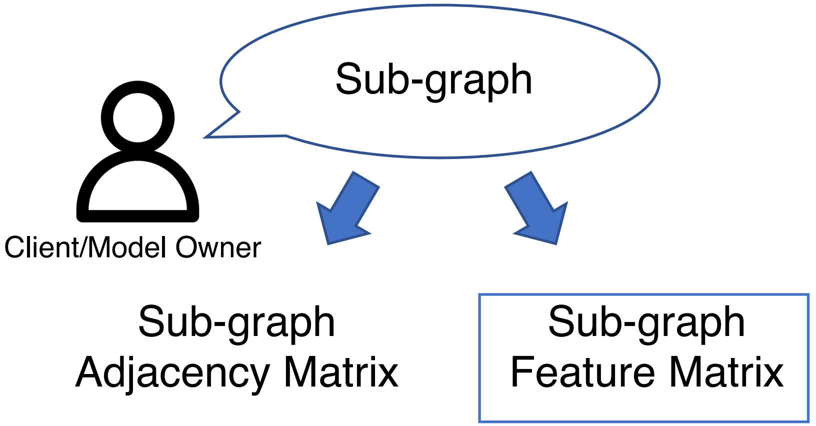
	0	1	…	n
0	1	1	…	0
1	1	1	…	0
…	…	…	…	…
n	0	0	…	1

Updated Adjacency Matrix

OblivGNN Approach – *Inductive* Exclusive Ops

Online – Oblivious Graph Update

Existing Graph Update



$\mathrm{KeyGen}(1||0, F_{\Delta}(v_i)) \rightarrow k_0^{\mathrm{F}}, k_1^{\mathrm{F}}$

$$\begin{array}{c|ccc} & 0 & \dots & c \\ \hline 0 & 1 & \dots & 0 \\ 1 & \boxed{1} & \dots & 1 \\ \dots & \dots & \dots & \dots \\ n & 0 & \dots & 1 \end{array} + \begin{array}{l} \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, 0||0) \\ \leftarrow \boxed{\mathrm{Eval}(k_b^{\mathrm{F}}, 1||0)} \\ \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, \dots) \\ \leftarrow \mathrm{Eval}(k_b^{\mathrm{F}}, n||0) \end{array} \Rightarrow \begin{array}{c|ccc} & 0 & \dots & c \\ \hline 0 & 1 & \dots & 0 \\ 1 & \boxed{0} & \dots & 1 \\ \dots & \dots & \dots & \dots \\ n & 0 & \dots & 1 \end{array}$$

Feature Matrix $[\hat{\mathbf{F}}]$
(secret shared)

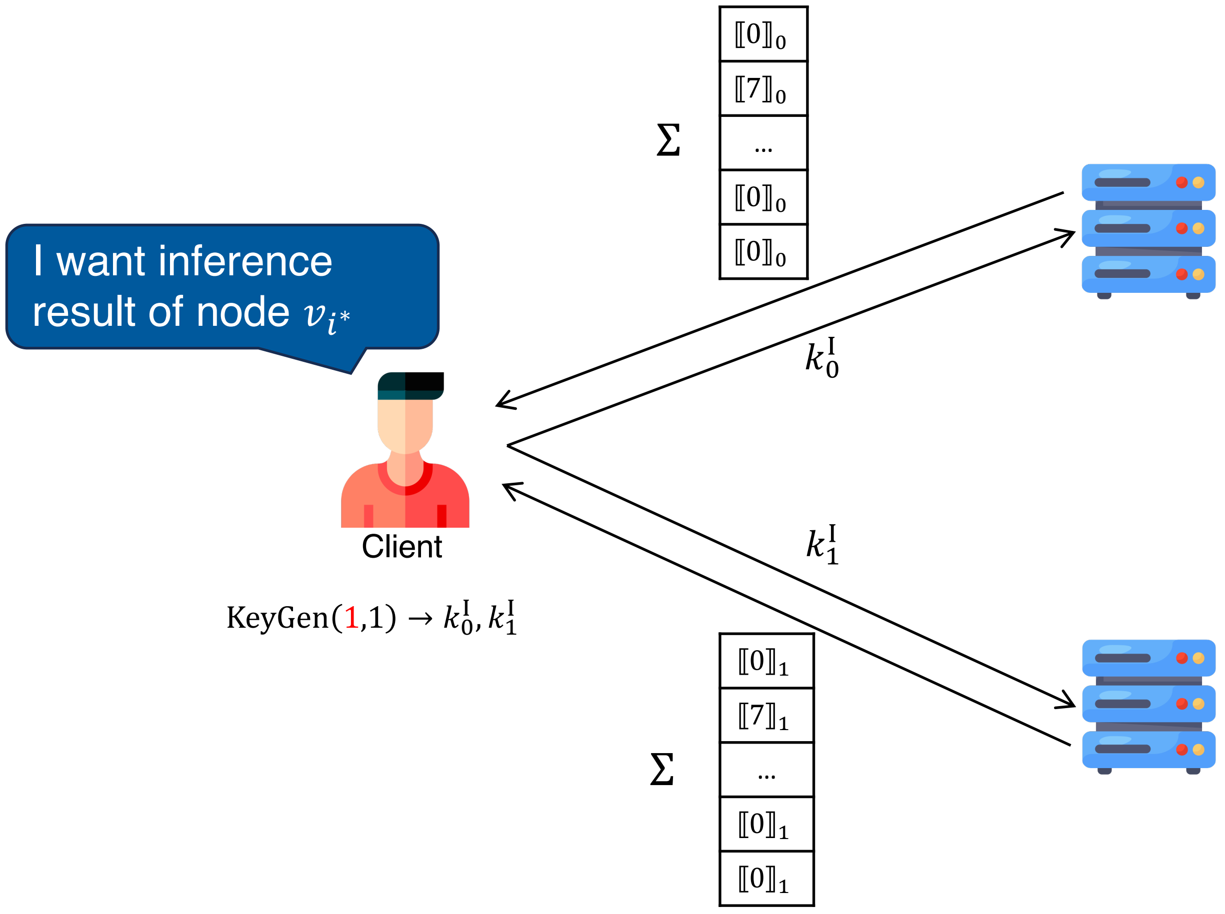
Updated Feature Matrix

Perform oblivious graph updates via DPF write

To further hide graph size, perform DPF full domain evaluation over the graph with padding

OblivGNN Approach

Online – Client Query



Server 0 Inference Results (Masked)

0	2	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, 0) = [\![0]\!]_0$
1	7	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, 1) = [\![1]\!]_0$
…	…	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, \dots) = [\![0]\!]_0$
n-1	4	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, n) = [\![0]\!]_0$
n	6	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, n+1) = [\![0]\!]_0$

Server 1 Inference Results (Masked)

0	2	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, 0) = [\![0]\!]_1$
1	7	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, 1) = [\![1]\!]_1$
…	…	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, \dots) = [\![0]\!]_1$
n-1	4	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, n) = [\![0]\!]_1$
n	6	×	$\mathrm{Eval}(k_b^{\mathrm{I}}, n+1) = [\![0]\!]_1$

Outline

- Introduction
  - Graph Neural Networks
  - Machine Learning as a Service
  - Design Goal
  - Related Work
- Preliminaries
  - Graph Convolutional Networks and Node Classification
  - Function Secret Sharing
- Protocol
  - Strawman
  - OblivGNN
- Experiments
  - System

Experiments

- **Platform**
  - Server
    - 3.70GHz Intel(R) Xeon(R) E-2288G CPU
    - 64GB RAM and 128GB external storage
    - Ubuntu 20.04.5 LTS
  - MP-SPDZ [Keller et al. (CCS'20)]

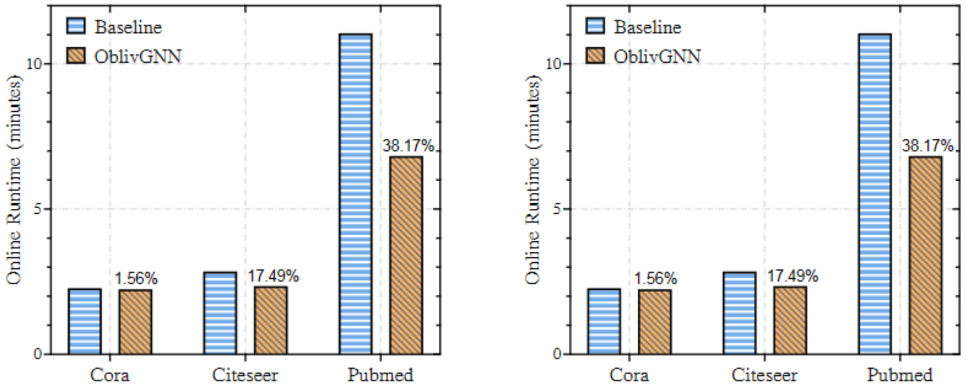
- **Datasets**
  - Cora, Citeseer and Pubmed

Dataset	Nodes	Feature	Edge	Classes
Cora	2708	1433	5429	7
Citeseer	3327	3703	4732	6
Pubmed	19717	500	44338	3

- **Baseline**
  - Baseline: pure additive secret shares for inference.
  - OblivGNN: additive secret shares with FSS for oblivious inference.

Experiments – System

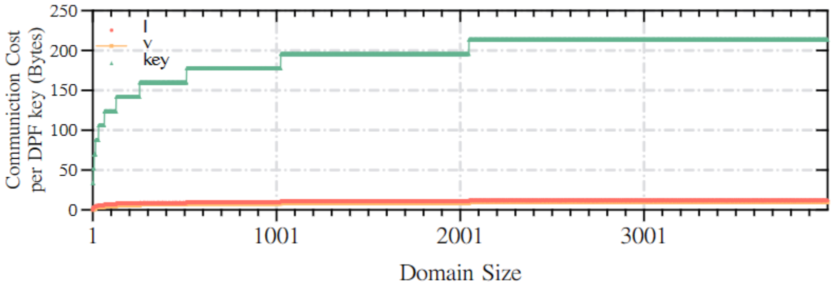
System Online Runtime:



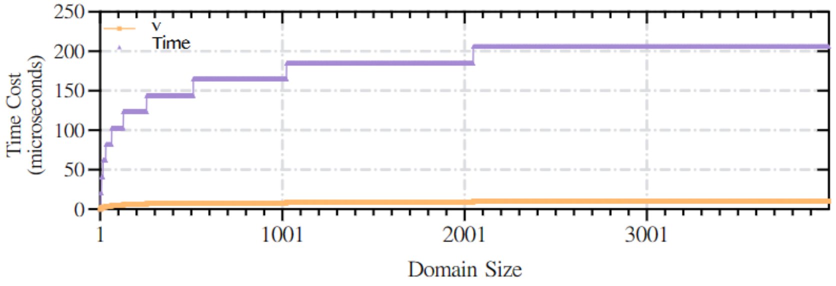
(a) Transductive (b) Inductive

Graph Update Cost:

Logarithm growth with graph size



(a) Node/Feature Update Key Size



(b) Node/Feature Update Time Cost

Online Communication (GB):

Reduction: 10× - 151×

	Baseline	**OblivGNN**
Cora	34.21	**0.29**
Citeseer	61.81	**0.41**
Pubmed	16.33	**1.65**

Future Work

- To enable efficient encrypted GNN training
- To scale PPML for GNNs for large graphs
- To deploy encrypted GNN training and inference protocols to GPU

Outline

- Privacy-preserving Machine Learning for GNNs
- Addressing Training Data Misuse in GNNs

GraphGuard: Detecting and Counteracting Training Data Misuse in Graph Neural Networks

Bang Wu, He Zhang, Xiangwen Yang, Shuo Wang, Minhui Xue,

Shirui Pan and **Xingliang Yuan**

In the Network and Distributed System Security Symposium (NDSS), 2024

MONASH University

CSIRO

DATA 61

SHANGHAI JIAO TONG UNIVERSITY

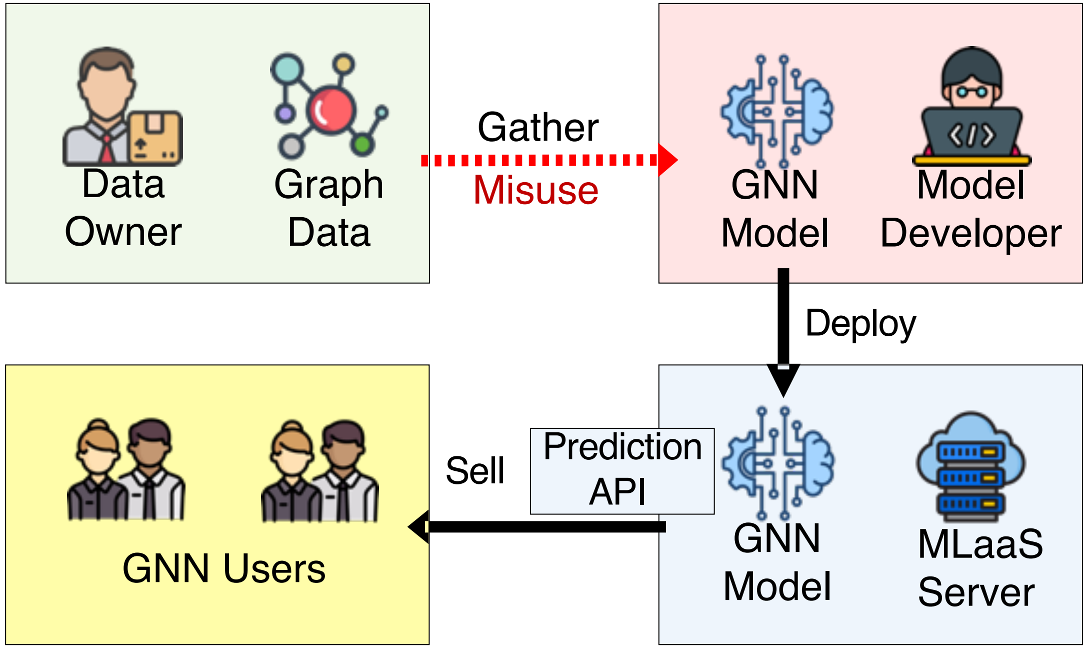
Griffith UNIVERSITY

Data Misuse aginst GNNs in MLaaS

GNN deployment raise data misuse concerns.

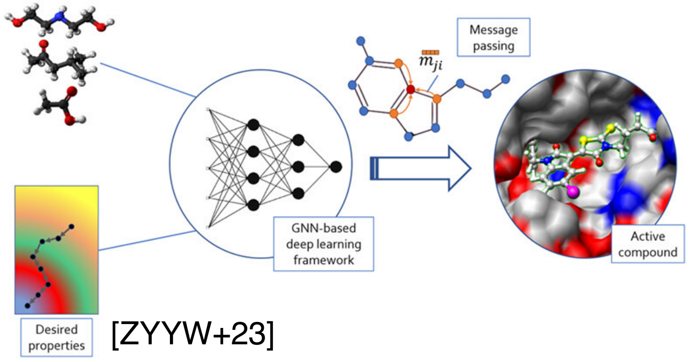
GNN development

1. **Gather data** for GNN training
2. **Deploys GNNs.**
3. **Sell API** to GNN users.



Data Misuse in GNNs

Graphs can be illegally/unintentionally collected for GNN training!

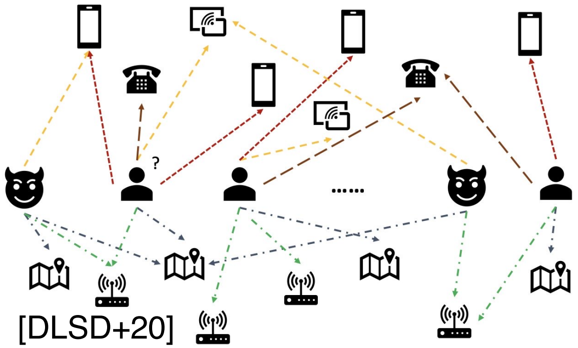


[ZYYW+23]

Drug Discovery

Mislead GNN prediction

Data error leads to incorrect predictions

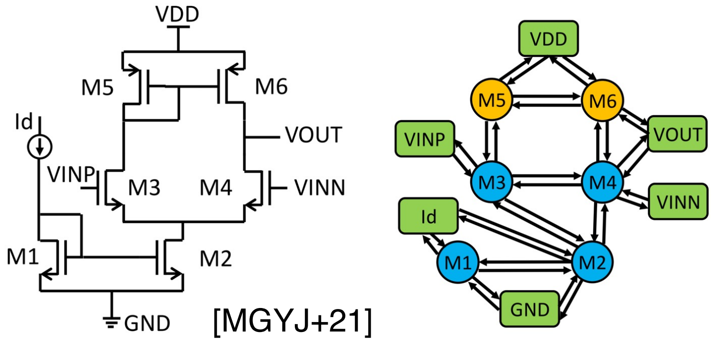


[DLSD+20]

Fraud Detection

Leverage sensitive data against privacy attacks

Transaction records are private



[MGYJ+21]

Chip Design

Compromise IP of data owners

Chip floorplan needs intellectual effort

How to deal with data misuse?

- **Detection**--*Membership Inference*
  - Identify if a specific graph has been used without authorization.
    - *Stealing Links [HJBG+]*
    - *Node-Level Membership Inference [HWWB+21]*
    - *Graph-level Membership Inference [WYPY21]*

- **Mitigation**--*Machine Unlearning*
  - Make the GNN model forget about misused graph data.
    - *GraphEraser [CZWB+22]*
    - *GNNDelete [CDHA+23]*

[HJBG+21] [HWWB+21] He, Xinlei, et al. "Node-level membership inference attacks against graph neural networks." *arXiv* 2021.
[WYPY21] Wu, Bang, et al. "Adapting membership inference attacks to GNN for graph classification: Approaches and implications." *ICDM* 2021.
[CZWB+22] Chen, Min, et al. "Graph unlearning." *CCS* 2022.
[CDHA+23] Cheng, Jiali, et al. "GNNDelete: A General Strategy for Unlearning in Graph Neural Networks." *ICLR 2023*.

Requirements of Mitigating Data Misuse in MLaaS

- Task Requirements

  R1 - Misuse Detection - Detect the data misused GNNs

  R2 - Misuse Mitigation - Remove the impact of misused data to the model

- (MLaaS) Setting Requirements

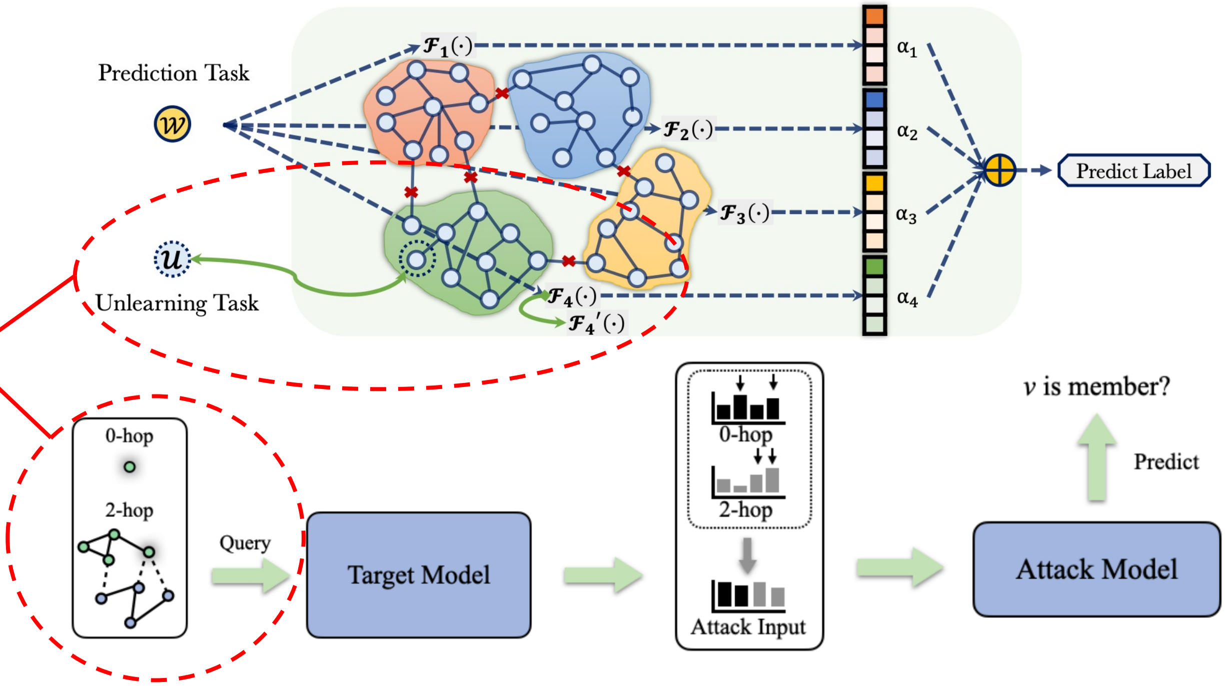
  R3 - Data Privatisation - Keep sensitive information about the graph locally

  R4 - GNN Model Agnostic - No assumption on GNN training/model architecture

Prior Work: Not Applicable to MLaaS

- **Assume that the server can access the exact training samples;**

Querying the exact training graph

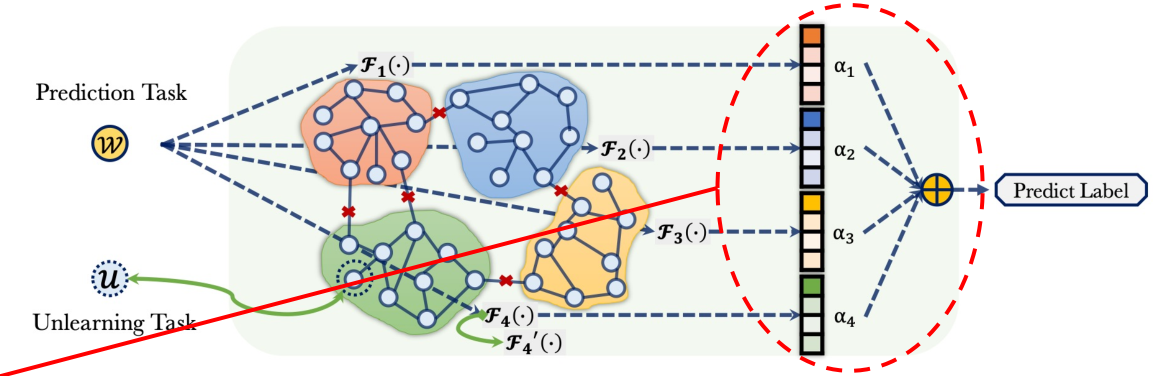


[HWWB+21]

[CZWB+22]

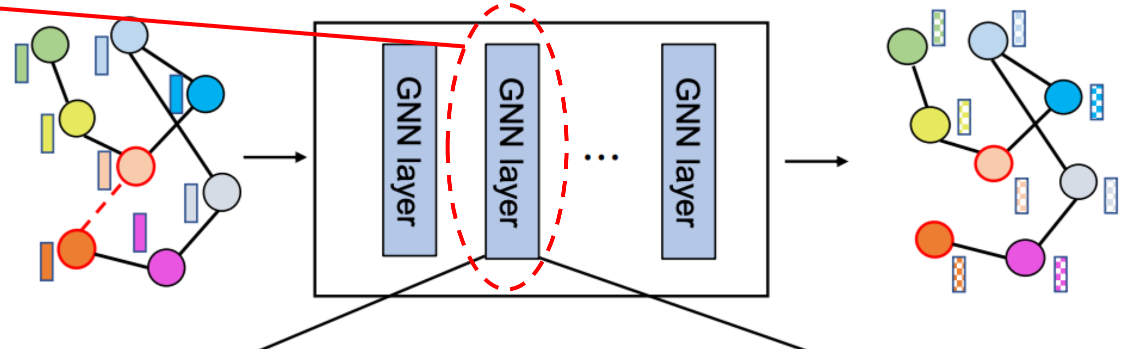
Prior Work: Not Applicable to MLaaS

- **Require modifications in the GNN architecture or training process.**



[HWWB+21]

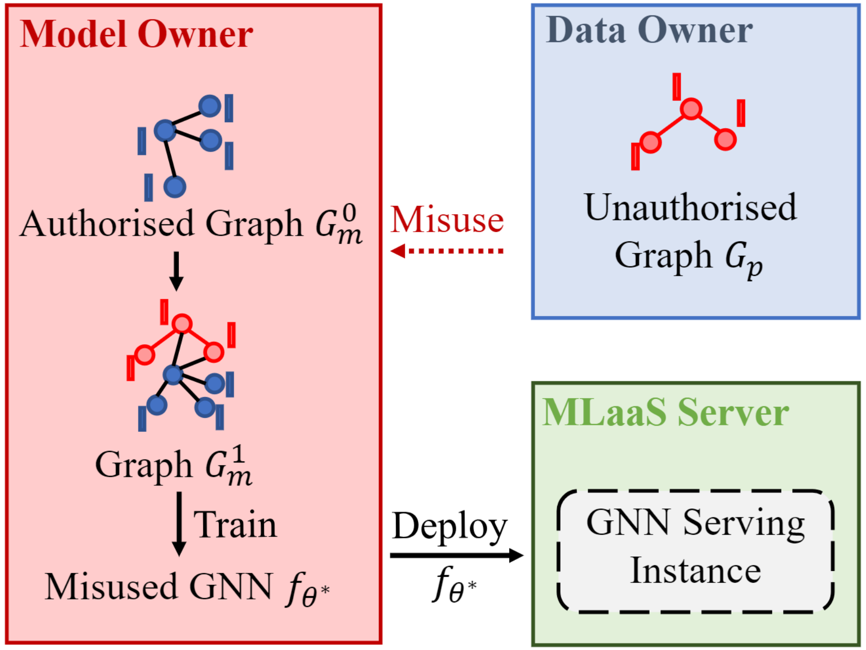
Additional functions in GNNs



[CDHA+23]

Our Design -- *GraphGuard*

- Identify if G_p is used in f_{θ^*} training (**R1**)
  - Membership inference
- Eliminate the impact of G_p on f_{θ^*} (**R2**)
  - Unlearning
- Do not leverage the graph structure (**R3**)
- Utilize only standard APIs in MLaaS (**R4**)

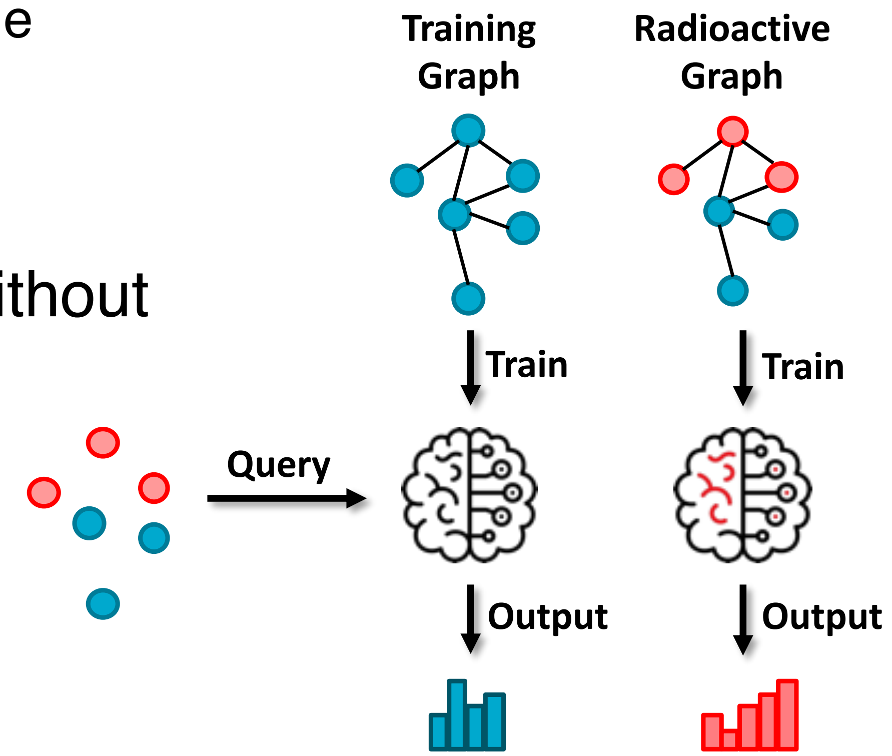


GraphGuard - Detection

- Detection goal

  Detect data misuse (**R1**) via API (**R4**) without the graph structure (**R3**).

GNNs trained on them react differently for specific node attribute queries.



- How to perform membership inference without the graph structure?
  - Prior study: proactive MIA. [SDSJ20]
  - Our design: **radioactive graph**

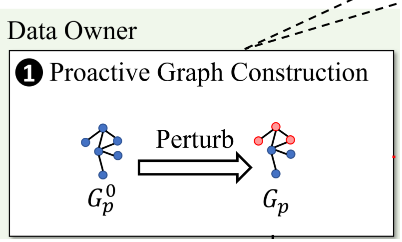
[SDSJ20] Sablayrolles, A., et.al. Radioactive data: tracing through training. ICML 2020.

GraphGuard - Detection

Pipeline:

1. Revise node attributes from G_p^0 to G_p before publishing graph

$$
\begin{aligned}
&\max_{G_p} d(\mathcal{A}(f_{\theta_1^*}(\hat{G}_p)), \mathcal{A}(f_{\theta_0^*}(\hat{G}_p))),\\
&s.t.\ \theta_1^* = \arg\min_{\theta} L(f_\theta(G_m^1)),\\
&\qquad \theta_0^* = \arg\min_{\theta} L(f_\theta(G_m^0)),
\end{aligned}
$$

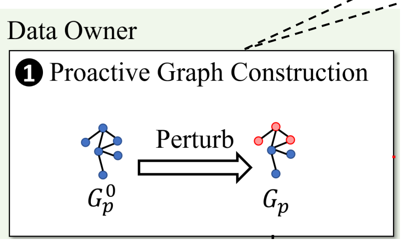


GraphGuard - Detection

Pipeline:

1. Revise node attributes from G_p^0 to G_p before publishing graph
2. Data misuse during training
3. GNN being deployed

$$
\begin{aligned}
&\max_{G_p} \; d(\mathcal{A}(f_{\theta_1^*}(\hat{G}_p)), \mathcal{A}(f_{\theta_0^*}(\hat{G}_p))),\\
&s.t. \;\; \theta_1^* = \arg\min_{\theta} L(f_\theta(G_m^1)),\\
&\qquad\; \theta_0^* = \arg\min_{\theta} L(f_\theta(G_m^0)),
\end{aligned}
$$

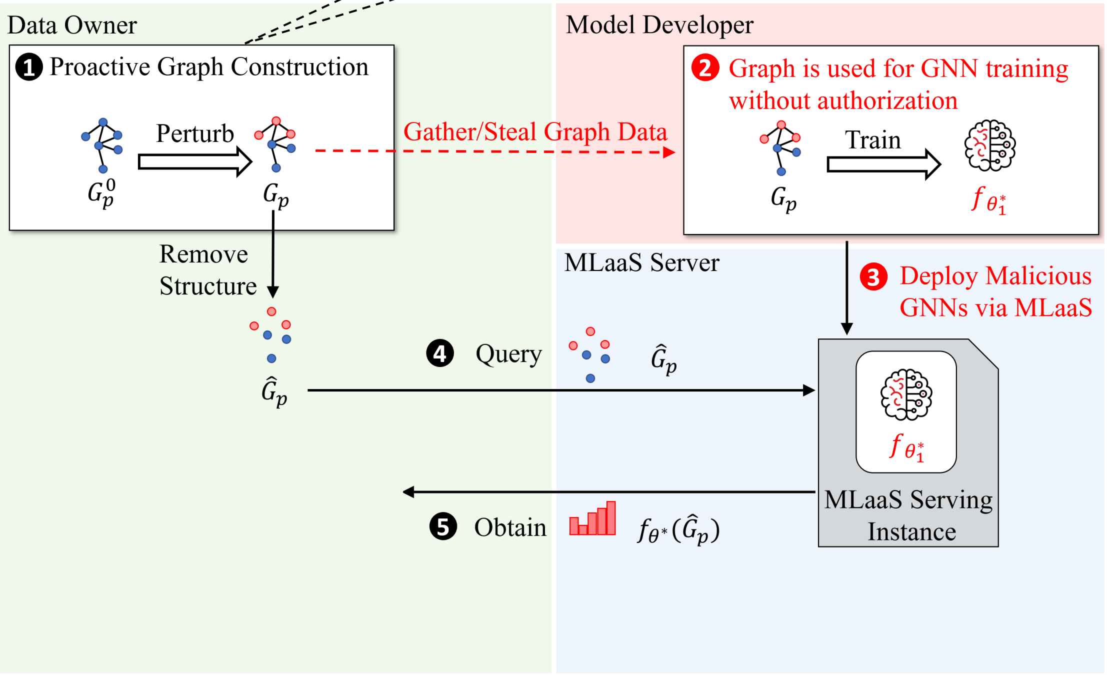


GraphGuard - Detection

Pipeline:

1. Revise node attributes from G_p^0 to G_p before publishing graph
2. Data misuse during training
3. GNN being deployed
4. Query graph $\hat{G}_p$ with node attributes only (without structure)
5. Obtain predictions $f_{\theta^*}(\hat{G}_p)$

$$\begin{aligned}
&\max_{G_p} d(\mathcal{A}(f_{\theta_1^*}(\hat{G}_p)), \mathcal{A}(f_{\theta_0^*}(\hat{G}_p))), \\
&s.t.\ \theta_1^* = \arg\min_{\theta} L(f_\theta(G_m^1)), \\
&\qquad \theta_0^* = \arg\min_{\theta} L(f_\theta(G_m^0)),
\end{aligned}$$

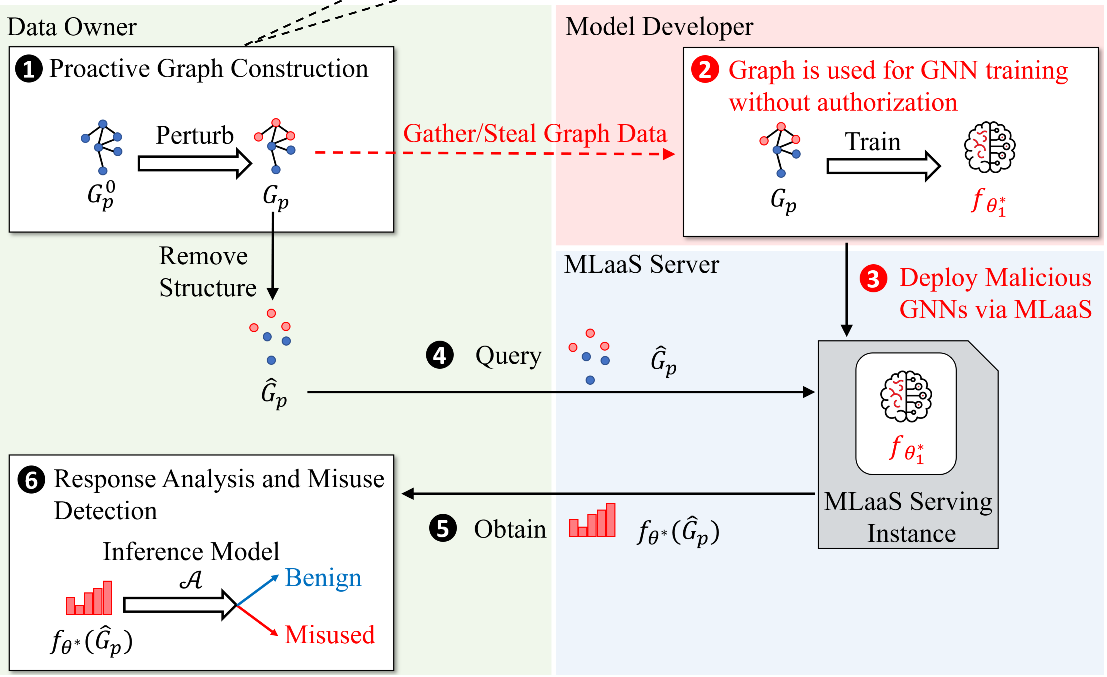


GraphGuard - Detection

Pipeline:

1. Revise node attributes from G_p^0 to G_p before publishing graph
2. Data misuse during training
3. GNN being deployed
4. Query graph $\hat{G}_p$ with node attributes only (without structure)
5. Obtain predictions $f_{\theta^*}(\hat{G}_p)$
6. Membership inference $\hat{\mathcal{A}}$

$$\max_{G_p} d(\mathcal{A}(f_{\theta_1^*}(\hat{G}_p)), \mathcal{A}(f_{\theta_0^*}(\hat{G}_p))),$$
$$s.t.\ \theta_1^* = \arg\min_{\theta} L(f_\theta(G_m^1)),$$
$$\theta_0^* = \arg\min_{\theta} L(f_\theta(G_m^0)),$$



GraphGuard - Mitigation

- Mitigation goal

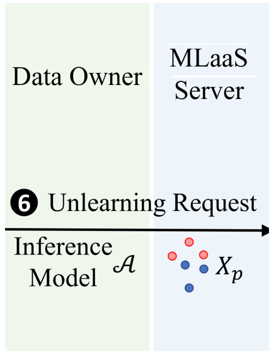
  Perform unlearning (**R2**) by fine-tuning the target GNNs (**R4**) without utilising the exact graph structure (**R3**).

- Design intuitions
  - Well-generalized GNNs **do not learn the exact graph structure**
  - Unlearning a subgraph **does not rely on the exact sub-graph structure**

- Our design
  - Leverage MIA for **graph synthesis**
  - Use synthetic graph for unlearning

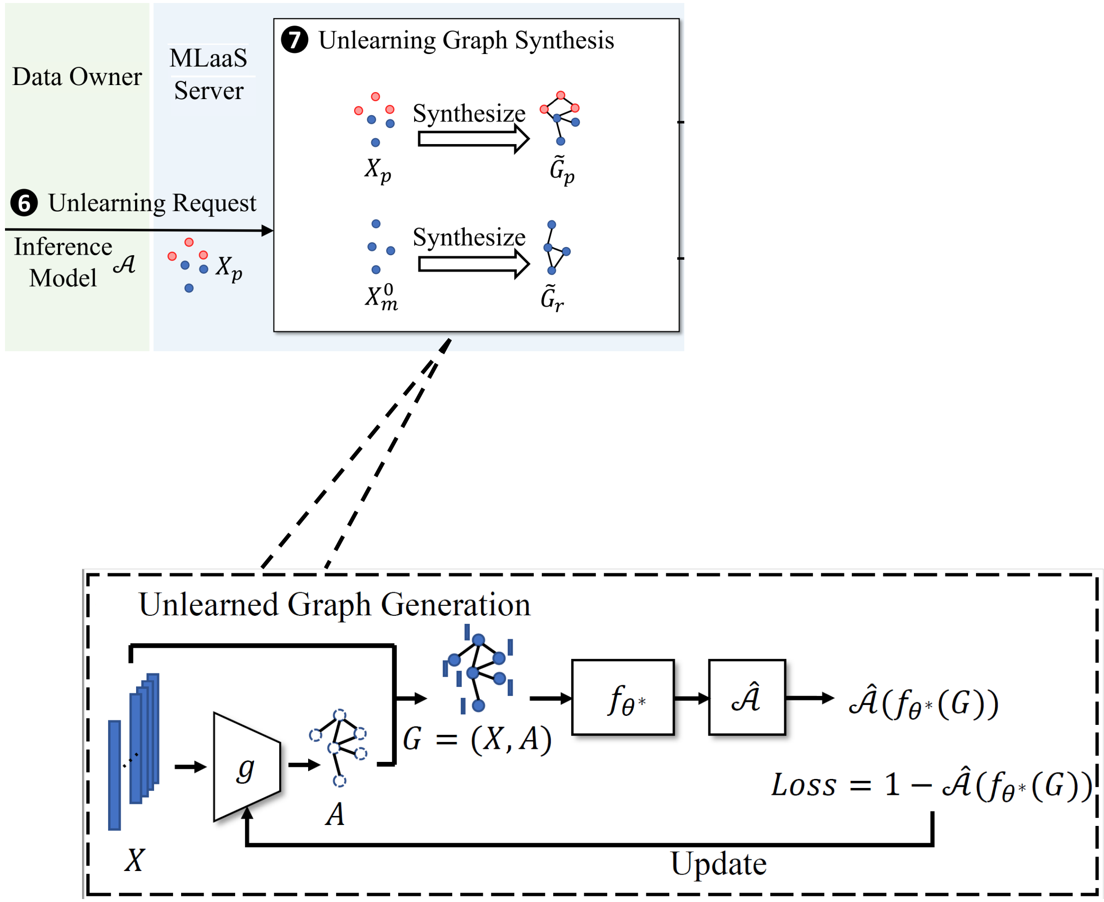
GraphGuard - Mitigation

6. MLaaS receives an unlearning request



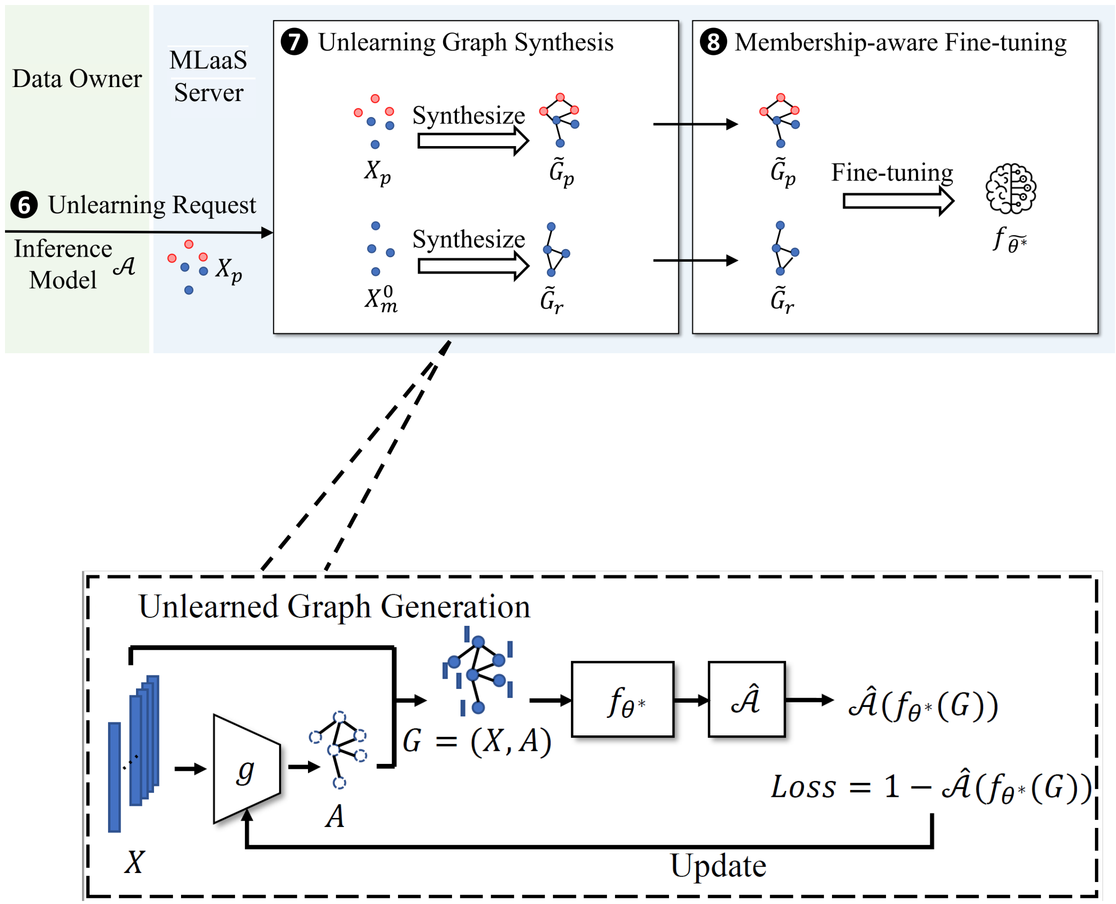
GraphGuard - Mitigation

6. MLaaS receives an unlearning request
7. (1) Data Gathering
   X_p, $\hat{\mathcal{A}}$ from the data owner
   X_m^0 from the model owner
7. (2) Graph Synthesize
   Unlearning graph $\tilde{G}_p$ by X_p, f_{θ^*} and $\hat{\mathcal{A}}$
   Remaining graph $\tilde{G}_r$ by X_m^0 , f_{θ^*} and $\hat{\mathcal{A}}$



GraphGuard - Mitigation

6. MLaaS receives an unlearning request
7. (1) Data Gathering
   X_p, $\hat{\mathcal{A}}$ from the data owner
   X_m^0 from the model owner
7. (2) Graph Synthesize
   Unlearning graph $\tilde{G}_p$ by X_p, f_{θ^*} and $\hat{\mathcal{A}}$
   Remaining graph $\tilde{G}_r$ by X_m^0 , f_{θ^*} and $\hat{\mathcal{A}}$
8. Fine-tuning f_{θ^*} :
   Increase loss on $\tilde{G}_p$
   Decrease loss on $\tilde{G}_r$



Evaluations - Detection

	GCN			GraphSage			GAT			GIN		
	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ
Cora	0.874	0.999	**↑0.125**	0.864	0.999	**↑0.135**	0.927	1.0	**↑0.073**	0.857	1.0	**↑0.143**
Citeseer	0.711	0.999	**↑0.288**	0.822	1.0	**↑0.178**	0.723	0.999	**↑0.276**	0.767	1.0	**↑0.233**
Pubmed	0.906	1.0	**↑0.094**	0.902	1.0	**↑0.098**	1.0	1.0	**0**	0.932	1.0	**↑0.068**
Flickr	1.0	1.0	**0**	0.994	1.0	**↑0.006**	0.996	1.0	**↑0.004**	0.998	1.0	**↑0.002**

Metric - AUC

Observations

- Our design achieve higher detection rates
- Baseline MIA only satisfied R1-Detectable & R4-Model Agnostic

Evaluations - Mitigation

- **Effectiveness** - MIA ASR before/after unlearning

	GCN			GraphSage			GAT			GIN		
	Before	After	Δ	Before	After	Δ	Before	After	Δ	Before	After	Δ
Cora	86.9	51.8	↓ **35.1**	83.3	54.5	↓ **28.8**	85.6	47.5	↓ **38.1**	91.7	47.9	↓ **43.8**
Citeseer	91.3	68.7	↓ **22.6**	81.2	56.1	↓ **25.1**	61.4	60.3	↓ **1.10**	86.2	46.2	↓ **40.0**
Pubmed	93.6	49.2	↓ **44.4**	85.7	53.2	↓ **32.5**	82.4	49.7	↓ **32.7**	84.1	47.6	↓ **36.5**

- **Utility -** Model ACC before/after unlearning

	GCN			GraphSage			GAT			GIN		
	R	U	Δ	R	U	Δ	R	U	Δ	R	U	Δ
Cora	75.7	74.3	↓ **1.2**	67.4	66.5	↓ **0.9**	83.1	81.5	↓ **1.6**	86.4	85.1	↓ **1.3**
Citeseer	81.1	80.0	↓ **1.1**	70.0	68.7	↓ **1.3**	82.2	80.1	↓ **2.1**	79.5	78.9	↓ **0.6**
Pubmed	81.8	79.8	↓ **2.0**	82.5	80.3	↓ **2.2**	83.6	81.3	↓ **2.3**	83.6	82.8	↓ **0.8**

Evaluations - Mitigation

- **Efficiency** - Time cost of retraining and our unlearning method.

	GCN			GraphSage		
	R	Ours	Times(↑)	R	Ours	Times(↑)
Cora	3.615	0.725	$\approx$**4.99**	4.188	0.643	$\approx$**6.51**
Citeseer	1.746	0.375	$\approx$**4.66**	2.023	0.333	$\approx$**6.08**
Pubmed	4.201	3.043	$\approx$**1.38**	4.865	2.670	$\approx$**1.82**

	GAT			GIN		
	R	Ours	Times(↑)	R	Ours	Times(↑)
Cora	3.600	0.720	$\approx$**5.0**	4.26	1.225	$\approx$**3.48**
Citeseer	1.737	0.375	$\approx$**4.63**	2.058	0.613	$\approx$**3.56**
Pubmed	4.190	3.017	$\approx$**1.39**	4.968	5.124	$\approx$**0.97**

Take Away

- **Definition of New Problem**
  - We define the graph misuse in MLaaS-deployed GNNs
- **Requirement Formulation**
  - **Task Requirements**: (R1) detectable, (R2) remedial
  - **(MLaaS) Setting Requirements**: (R3) data privatization, (R4) model agnostic
- **An Integrated Pipeline**
  - **Radioactive data** driven detection technique
  - Unlearning methodology w/o confidential graph structure

- Code: https://github.com/GraphGuard/GraphGuard-Proactive

Artifact Evaluated
NDSS
Available
Functional

Challenges Ahead

- How to enable privacy-preserving auditing for data misuse in the ML pipeline?
  - Will perturbed data be exploited to recover the original data?
- How to enable privacy-preserving unlearning?
  - Will synthesized data be exploited to recover the unlearning request?
- How to enable verifiable machine unlearning?
  - Ensure the execution of unlearning

Thanks! xingliang.yuan@unimelb.edu.au