

PRIVACY-PRESERVING DEEP LEARNING OVER GRAPHS

Sina Sajadmanesh

sajadmanesh@idiap.ch

IDIAP RESEARCH INSTITUTE

SWISS FEDERAL INSTITUTE OF TECHNOLOGY (EPFL)

Information Processing and Communications Lab
Imperial College London
Dec 9, 2020

OUTLINE

1. Introduction and Motivation
2. Graph Neural Networks
3. Privacy Attacks on Graph Neural Networks
4. Privacy-Preserving Graph Neural Network Models
5. Locally Private Graph Neural Networks
6. Research Directions and Conclusion

INTRODUCTION AND MOTIVATION

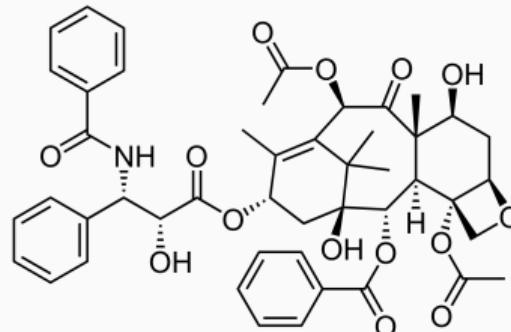
INTRODUCTION

Graphs are **ubiquitous**

Social Networks



Molecules



Knowledge Graphs

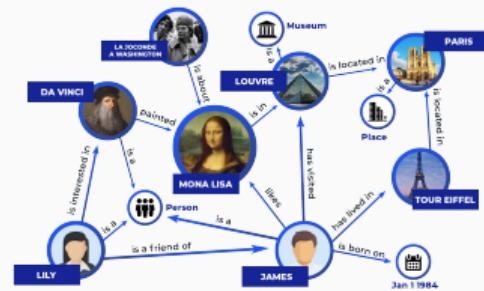
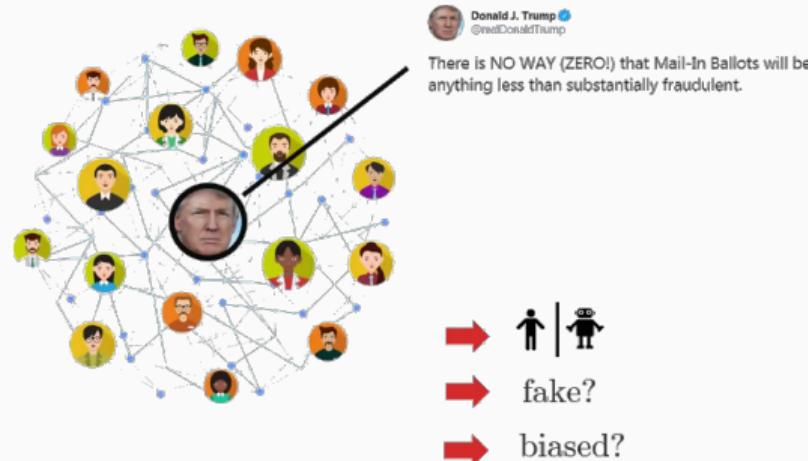


Image source (from left to right): <https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/>, <https://en.wikipedia.org/wiki/Terpenoid>, <https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/>

MACHINE LEARNING TASKS OVER GRAPHS

Node Classification / Regression

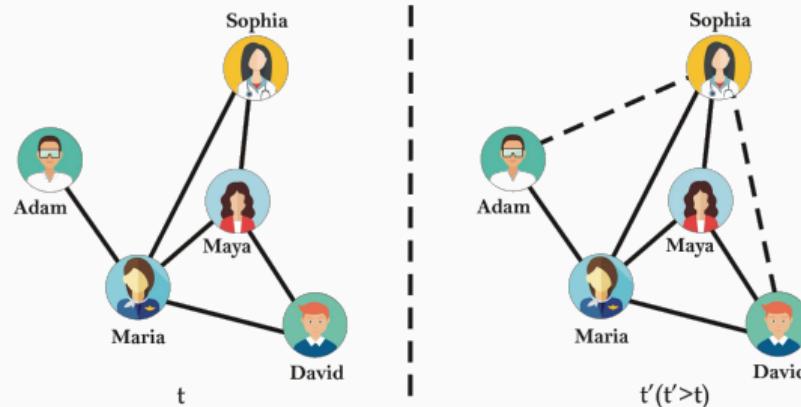
- Given a graph, which is the class label / value of a node?
- Example:** face account detection



MACHINE LEARNING TASKS OVER GRAPHS

Link Prediction

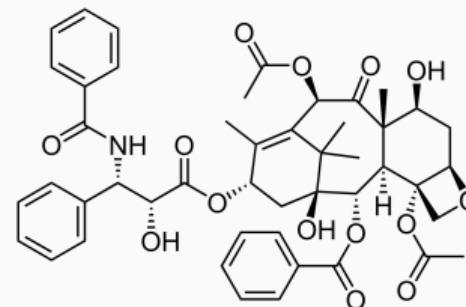
- Given a graph, which links are likely to form?
- Example:** recommendation systems



[Ahmad et al., 2020]

Graph Classification

- Given a graph, predict its label
 - Example:** antibiotic discovery



Antibiotic? Or Not

MACHINE LEARNING TASKS OVER GRAPHS

Graph Decoding to Structured Data

- Given a graph, what is the corresponding structures representation?
- Example:** graph to text, graph to image, ...

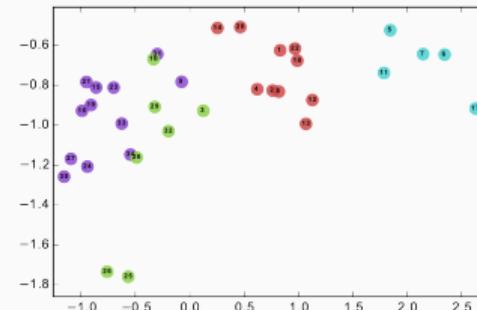
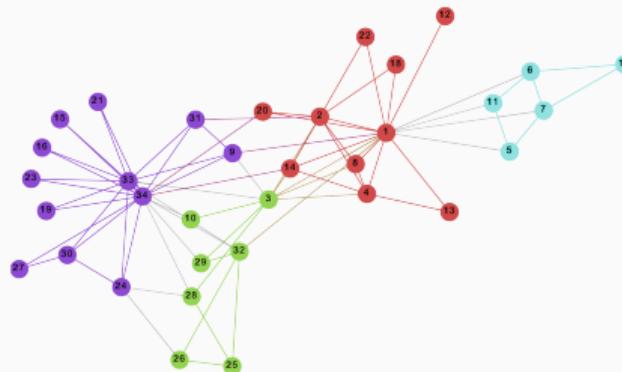


[Johnson et al., 2018]

GRAPH REPRESENTATION LEARNING

Graph Representation Learning

- A key step to applying machine learning algorithms over graphs
- Learn representation of nodes (or graphs) in a low-dimensional space
- **Graph embeddings algorithms:** learn node embeddings directly from topological structure
- **Graph neural networks:** learn how to compute node representation based on local network neighborhood



[Perozzi et al., 2014]

Graphs could be sensitive

- Users' personal attributes, financial transactions, medical/biological networks, ...
- Machine learning algorithms should preserve the privacy of individuals in graph data

Private Machine Learning on Graphs

- Privacy-preserving ML methods are mostly designed for **non-relational** data
- Specific techniques need to be developed to address privacy issues in graphs
- Privacy-preserving graph representation learning tries to fill this gap

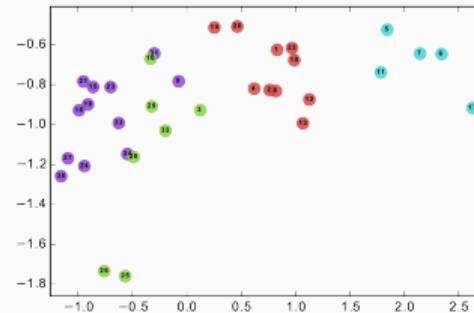
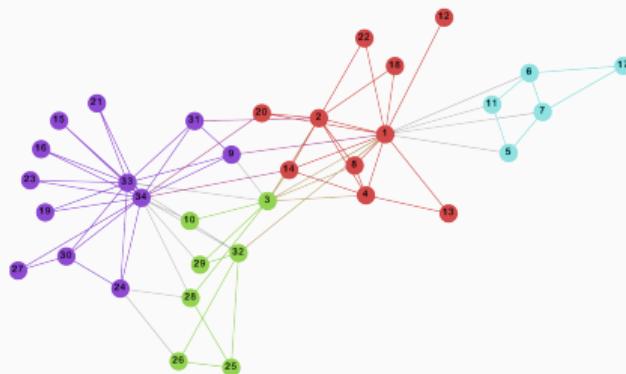
GRAPH NEURAL NETWORKS

GRAPH EMBEDDING

Input: Graph $G = (V, E)$, with node set V and link set E

Objective: Embed each node into a continuous vector space such that similarity in the embedding space approximates similarity in the graph:

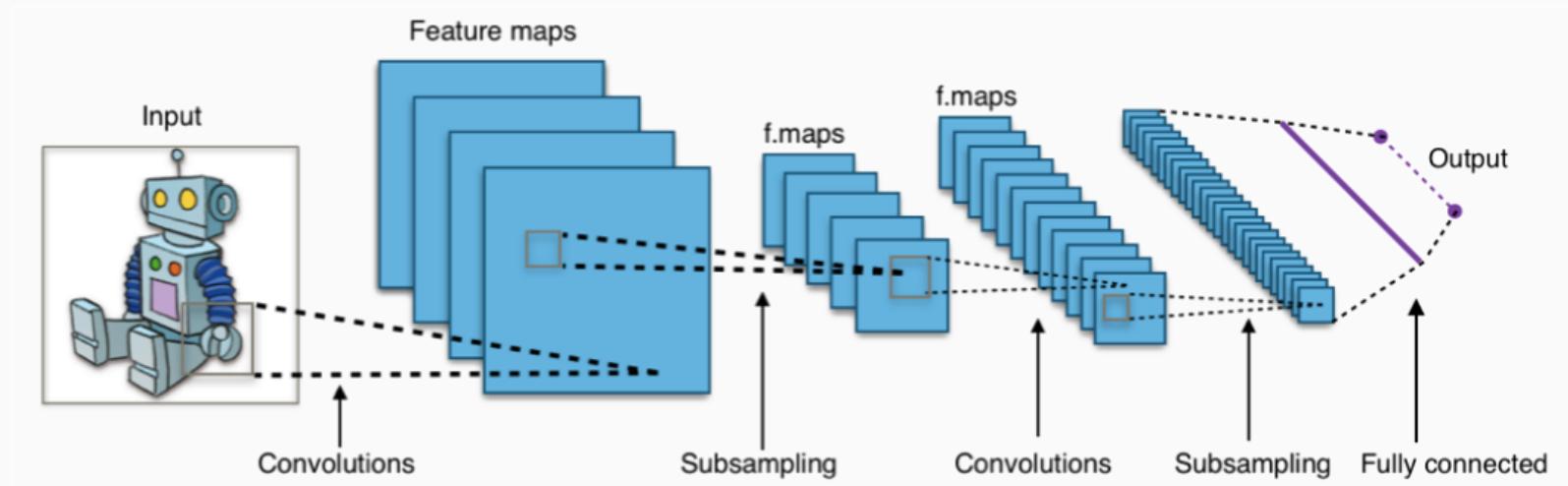
$$\text{similarity}(u, v) \approx \mathbf{z}_u^T \mathbf{z}_v$$



[Perozzi et al., 2014]

THE POWER OF DEEP LEARNING

Modern deep learning excels at exploiting **grid-structured data**



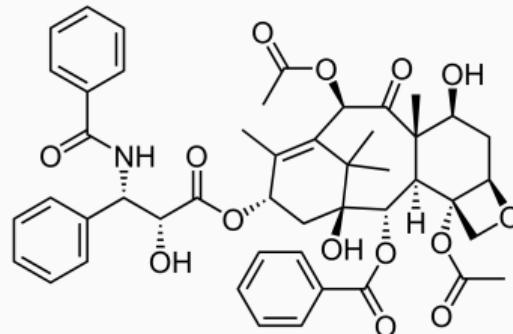
GRAPH-STRUCTURED DATA

But graphs are **combinatorial structures**, have arbitrary sizes, and contain multi-modal information

Social Networks



Molecules



Knowledge Graphs

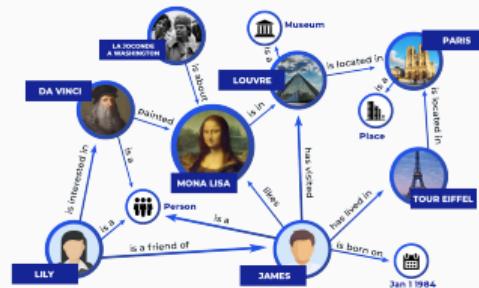
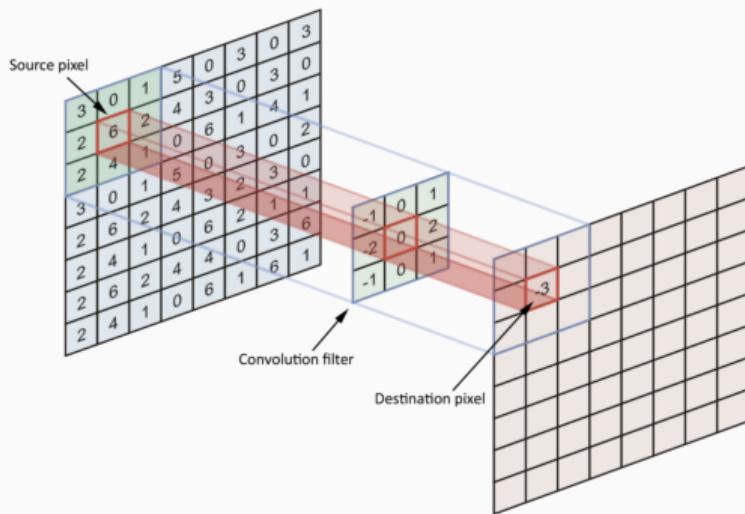


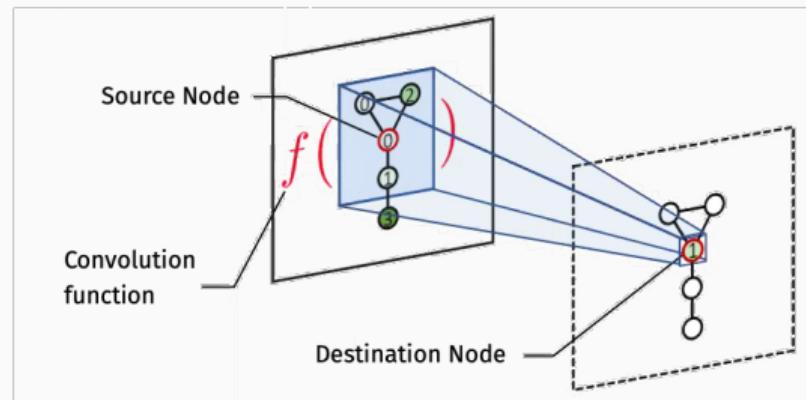
Image source (from left to right): <https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/>, <https://en.wikipedia.org/wiki/Terpenoid>,
<https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/>

GRAPH CONVOLUTION

Image Convolution



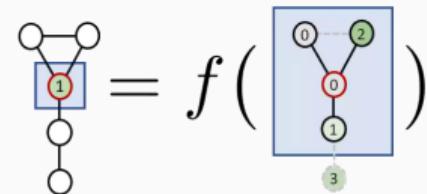
Graph Convolution



INSIDE THE GRAPH CONVOLUTION

Input: an input representation vector \mathbf{h}_v for each node v

Output: a new representation vector \mathbf{h}'_v for each node v



$$\mathbf{h}'_v = f(\{\mathbf{h}_u : u \in \mathcal{N}(v)\}) = \text{UPDATE} \left(\text{AGGREGATE} (\{\mathbf{h}_u : u \in \mathcal{N}(v)\}) \right)$$

- **AGGREGATE** is a permutation invariant function (e.g., sum, mean, max)
- **UPDATE** is a neural network (e.g., MLP)

SPECIAL CASES

Graph Convolutional Network (GCN) [Kipf and Welling, 2017]

$$\text{AGGREGATE}(\{h_u : u \in \mathcal{N}(v)\}) = \sum_{u \in \mathcal{N}(v)} \frac{h_u}{\sqrt{|\mathcal{N}(u)|} \sqrt{|\mathcal{N}(v)|}}$$

Graph Sample and Aggregate (GraphSAGE) [Hamilton et al., 2017]

$$\text{AGGREGATE}(\{h_u : u \in \mathcal{N}(v)\}) = \text{CONCAT}\left(h_v, \text{MEAN}(\{h_u : u \in \mathcal{N}(v)\})\right)$$

Graph Isomorphism Network (GIN) [Xu et al., 2018]

$$\text{AGGREGATE}(\{h_u : u \in \mathcal{N}(v)\}) = (1 + \epsilon) \cdot h_v + \text{SUM}(\{h_u : u \in \mathcal{N}(v)\})$$

THE GRAPH NEURAL NETWORK MODEL

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; Feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$

Output: Embedding vector \mathbf{z}_v for all $v \in \mathcal{V}$

Initialization: $\mathbf{h}_v^0 = \mathbf{x}_v$ for all $v \in \mathcal{V}$

for $l = 1$ to L **do**

foreach node $v \in \mathcal{V}$ **do**

$$\mathbf{h}_{\mathcal{N}(v)}^l = \text{AGGREGATE}_l \left(\{\mathbf{h}_u^{l-1} : u \in \mathcal{N}(v)\} \right)$$

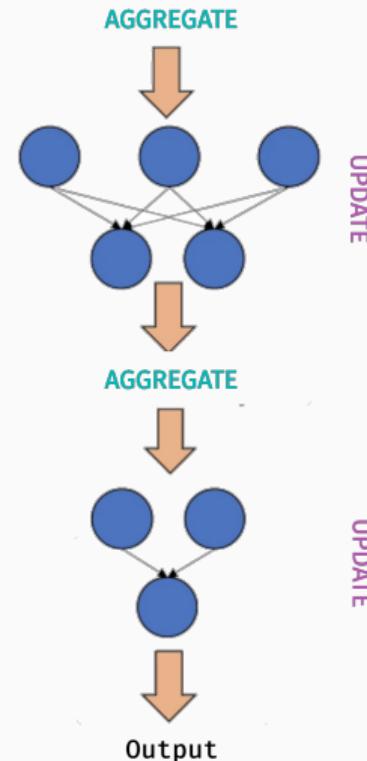
$$\mathbf{h}_v^l = \text{UPDATE}_l \left(\mathbf{h}_{\mathcal{N}(v)}^l \right)$$

end

end

return $\mathbf{z}_v = \mathbf{h}_v^L$: for all $v \in \mathcal{V}$

You can feed these embeddings into any **loss function** to train the network parameters



PRIVACY ATTACKS ON GRAPH NEURAL NETWORKS

[Duddu et al., 2020] Quantifying Privacy Leakage in Graph Embedding, NeurIPS PPML

[He et al., 2021] Stealing Links from Graph Neural Networks, USENIX Security

Membership Inference Attack

- Infer whether a given node is part of the target graph

Attribute Inference Attack

- Infer sensitive attributes of a node in the target graph

Link Inference Attack

- Infer whether a given pair of nodes are connected in the target graph

THREAT MODEL

Adversary have **back-box access** to a trained GNN

- The GNN is trained for node classification
- The GNN can be queried to retrieve embeddings or predictions



Example

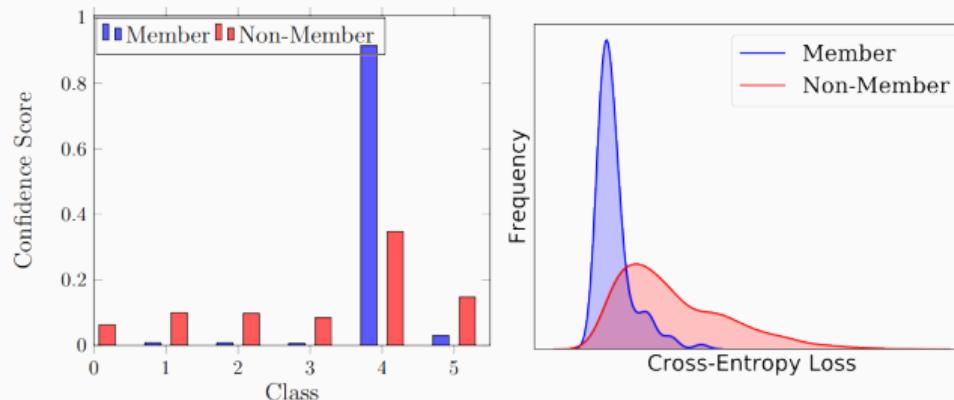
- GNN-based fake account detection service
- Machine Learning as a Service
- Publishing graph embeddings for research purposes

Different attacks may need extra background knowledge

MEMBERSHIP INFERENCE ATTACK

Exploits the **statistical difference** between the prediction confidence on **training** and **testing** data

- GNNs are more confident when predicting labels for the training data
- Nodes with high output confidence are likely members of the training graph



[Duddu et al., 2020]

Confidence Attack (unsupervised)

- Compare the highest prediction confidence of the given node to a **threshold**
- If above the threshold, then member

Shadow Attack (supervised)

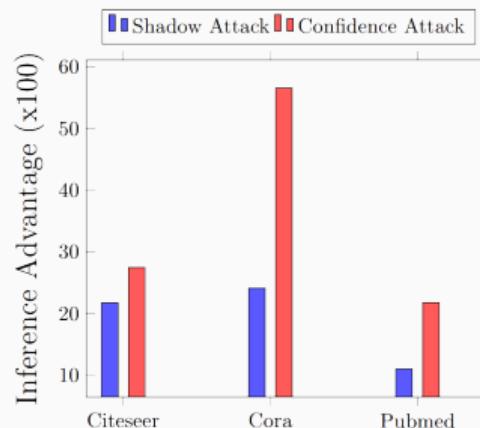
- Uses an **auxiliary graph** sampled from the training graph
- Train a **similar GNN** over the auxiliary graph and get predictions
- Train a **binary classifier** with prediction scores as features to predict the membership status in the auxiliary graph
- Predict the membership of nodes in the original graph using the learned classifier

MEMBERSHIP INFERENCE ATTACK

Adversary advantage metric

- Estimates model information leakage compared to the random guess

$$I_{adv} = 2 \times (Acc - 0.5)$$



[Duddu et al., 2020]

Exploits the fact that similar users have similar attributes

- Similar users are connected
- The connectivity of users is captured by embeddings

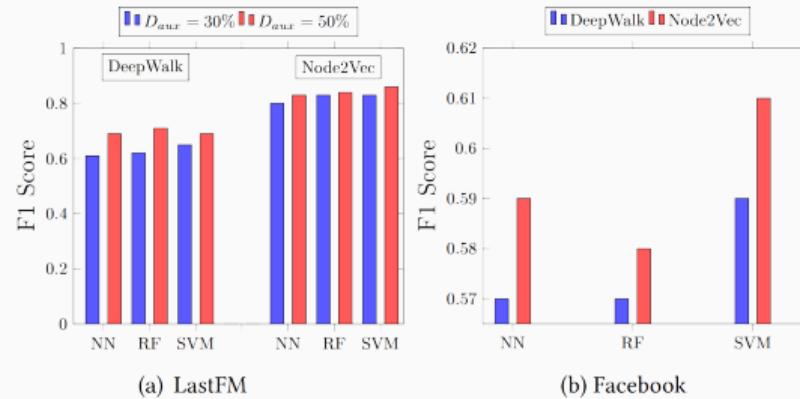
Requirements

- Needs **embeddings**, not predictions
- Requires an **subset of nodes with sensitive attributes** revealed

ATTRIBUTE INFERENCE ATTACK

Attack Methodology

- Train a classifier with the embedding of the auxiliary graph's nodes as features and their sensitive attribute as label
- Use the trained classifier to predict the sensitive attribute of any given node in the original graph



[Duddu et al., 2020]

Exploits the similarity of prediction posteriors for connected nodes

- If two nodes are connected, then their prediction scores are likely similar

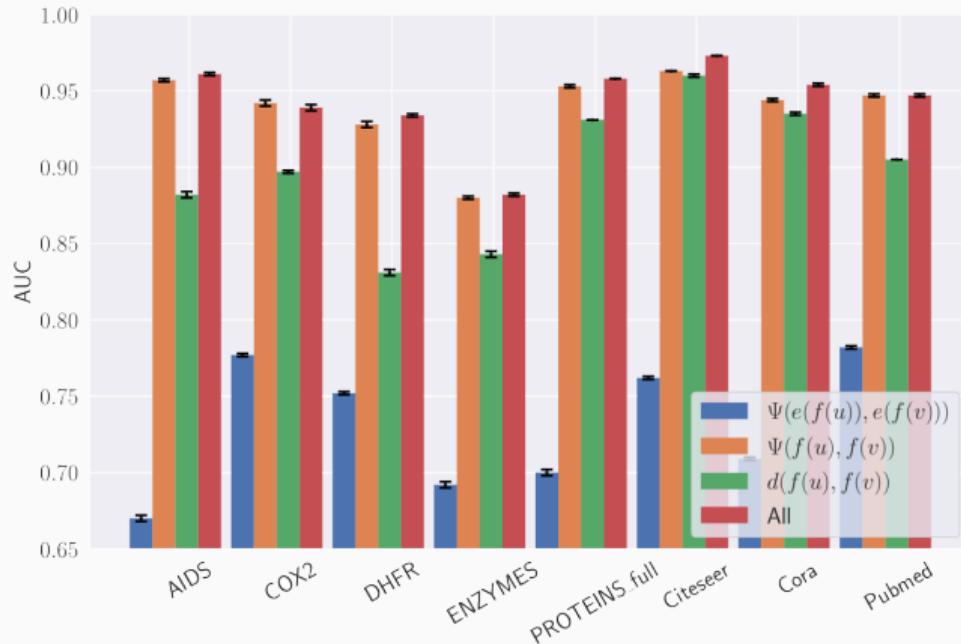
Requirements

- Requires access to an **auxiliary subgraph** of the original graph

Attack Methodology

- Obtain the prediction scores from the target GNN for every node pair in the auxiliary graph
- Extract features from the obtained scores for each node pair
 - features based on **distance metrics** (cosine, euclidean, etc), **vector operations** (average, hadamard product, etc), and **entropy**
- Train an MLP using the extracted features and the link state in the auxiliary graph
- Use the trained MLP to infer the link between any node pair in the original graph

LINK INFERENCE ATTACK



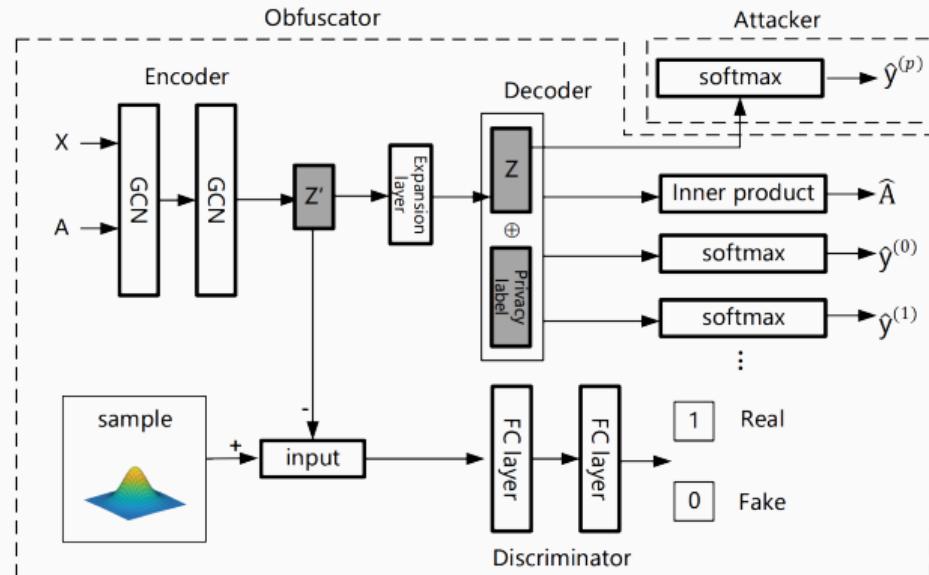
[He et al., 2021]

PRIVACY-PRESERVING GRAPH NEURAL NETWORK MODELS

PRIVACY-PRESERVING GRAPH NEURAL NETWORK MODELS

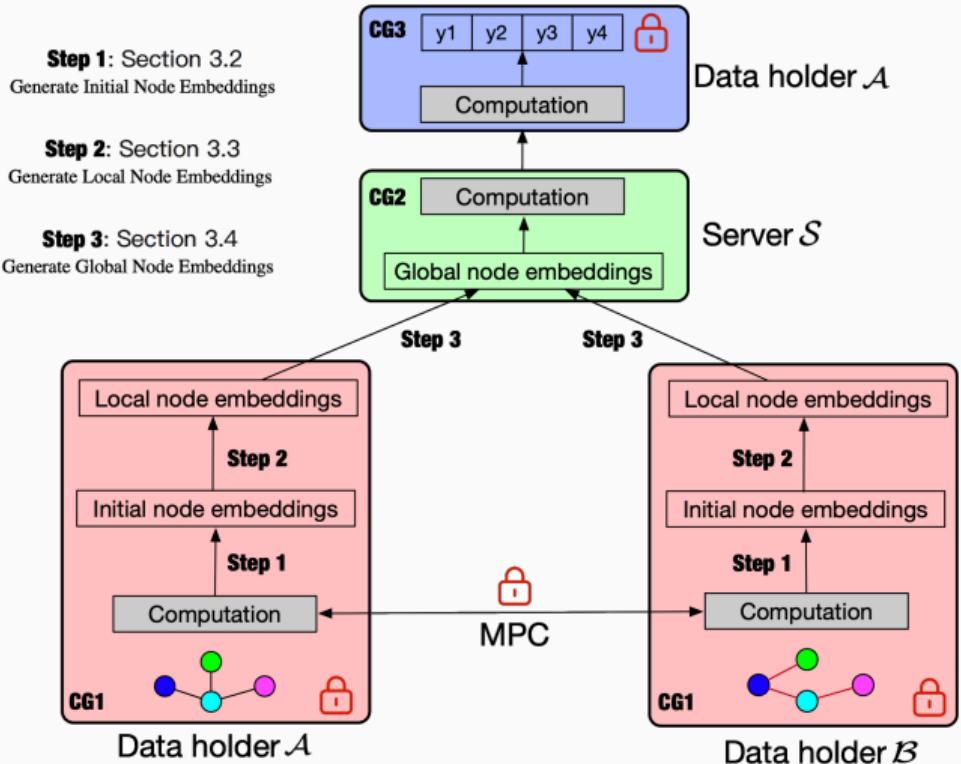
Adversarial Privacy-Preserving Graph Embedding Against Inference Attack [Li et al., 2020]

- **Setting:** graph embeddings are released publicly
- **Goal:** preserving information about the graph structure and utility node attributes
- **Privacy:** mitigating the inference of sensitive node attributes
- **Approach:** adversarial learning



Privacy-Preserving GNN for Node Classification [Zhou et al., 2020]

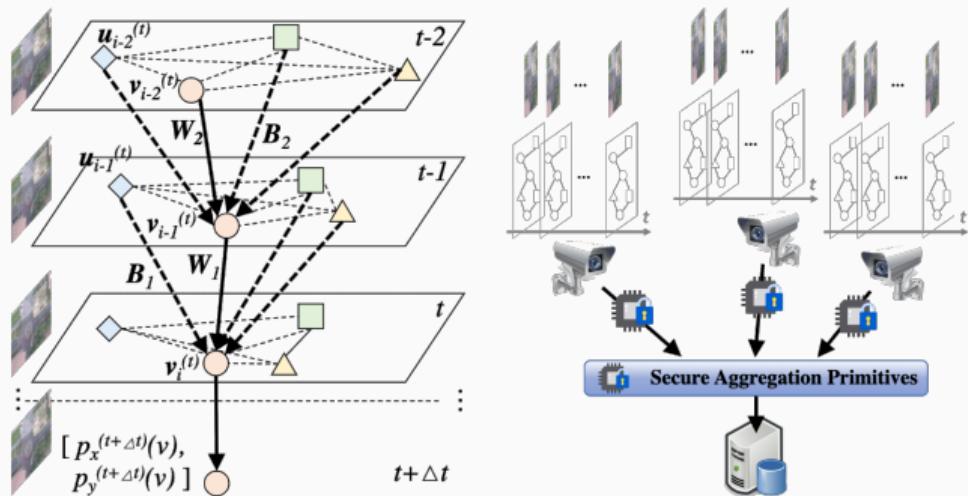
- **Setting:** the graph is partitioned vertically across multiple parties
- **Goal:** learning a global GNN collaboratively
- **Privacy:** keep node features and link information local to each party
- **Approach:** split learning + secure multiparty computation



PRIVACY-PRESERVING GRAPH NEURAL NETWORK MODELS

Federated Dynamic GNN with Secure Aggregation [Jiang et al., 2020]

- **Setting:** each camera has its own graph sequence
- **Goal:** learning a global GNN collaboratively to predict future object positions
- **Privacy:** keep node features and link information local to each camera
- **Approach:** federated learning + secure multiparty computation



LOCALLY PRIVATE GRAPH NEURAL NETWORKS

[Sajadmanesh and Gatica-Perez, 2020]

PROBLEM DEFINITION

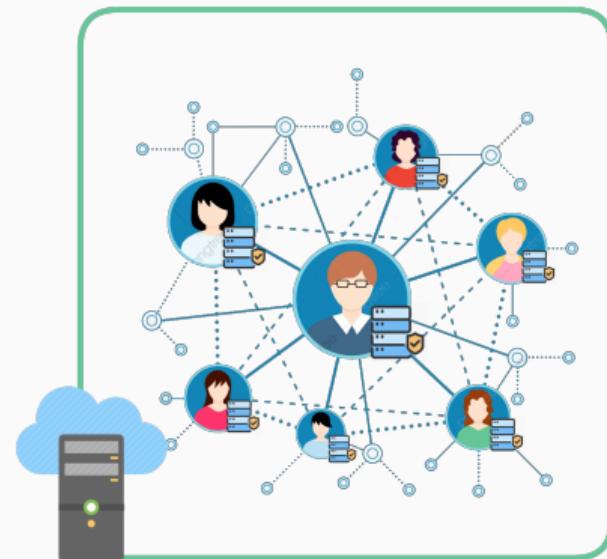
Privacy-Preserving GNN learning with node-level privacy

Setting:

- The server has access to a graph
- Each node has a private feature vector
- Node features are inaccessible by the server

Problem:

- How to learn a GNN without letting the private features leave the nodes?



Why not federated learning?

- Message passing must be done at **node side**
- Each node requires the private features of its adjacent nodes
 - If sent in plain text → **privacy violation!**
 - If sent using SMC → FL + SMC = **massive communication!**
- **Result:** message passing at node side is not feasible

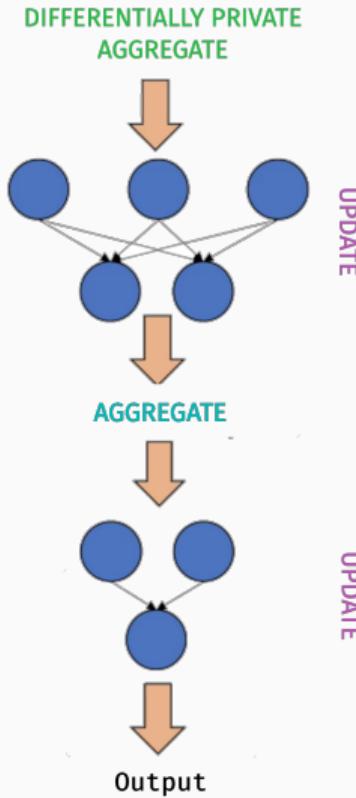
OUR APPROACH

Let's keep the model on the server

- We only need to calculate the first layer's **AGGREGATE** function privately!
- SMC? It's vulnerable to differential attack!

Idea: make **AGGREGATE** differentially private by input perturbation!

- Individual features are not necessary, **only aggregated features** are needed
- Node features can be privatized by injecting noise using **Local Differential Privacy (LDP)**
- The neighborhood aggregation will **cancel out the injected noise** in the features



Local Differential Privacy [Kasiviswanathan et al., 2011]

- De facto standard for computing aggregated statistics over private data
- **Key idea:** data holders send perturbed data to the aggregator that are meaningless individually, but can approximate the target statistic when aggregated.
- Composed of two steps:
 1. **Data collection:** each data holder perturbs his data x using a randomized mechanism \mathcal{M} , returning $y = \mathcal{M}(x)$ to the aggregator.
 2. **Estimation:** the aggregator computes the target statistic (e.g. mean)
- Randomization in \mathcal{M} provides **plausible deniability** to data holders
- However, the aggregator must not be able to infer initial data x by observing y and having arbitrary **background knowledge**

Local Differential Privacy

Given $\epsilon > 0$, a randomized mechanism \mathcal{M} satisfies ϵ -local differential privacy if for all possible pairs of user's private data x and x' , and for all possible outputs y of \mathcal{F} , we have:

$$\Pr[\mathcal{M}(x) = y] \leq e^\epsilon \Pr[\mathcal{M}(x') = y]$$

Interpretation

- Any input value x is almost as likely (depending on ϵ) to produce the same output y
- An adversary cannot distinguish between x and x' by observing y

The outline of our **locally private GNN** (LPGNN) algorithm

1. Each node perturbs its private feature vector using an LDP mechanism and sends it to the server
2. The server uses the received perturbed feature vectors and estimates the first layer's **AGGREGATE** function
3. The training proceeds with forward and backward propagation as usual
4. Return to step 2 if stopping criteria has not met

But it's not that easy! there are challenges...

Challenge #1: High-dimensional features

- The total privacy budget for a single node scales with the number of features
→ Too much privacy leakage!
- Trivial solution: perturb individual features with ϵ/d privacy budget
→ Too much noise!

Multi-bit mechanism: multidimensional feature perturbation

- Uniformly sample m features out of d ones without replacement
- Perturb each sampled feature with ϵ/m privacy budget
- Map the output of the 1-bit mechanism to either -1 or 1
- For the rest of the features (not sampled) return 0

Algorithm 1: Multi-Bit Mechanism

Input : feature vector $\mathbf{x} \in [\alpha, \beta]^d$; privacy budget $\epsilon > 0$; range parameters α and β ; sampling parameter $m \in \{1, 2, \dots, d\}$.

Output: perturbed vector $\mathbf{x}^* \in \{-1, 0, 1\}^d$.

- 1 Let \mathcal{S} be a set of m values drawn uniformly at random without replacement from $\{1, 2, \dots, d\}$
- 2 **for** $i \in \{1, 2, \dots, d\}$ **do**
- 3 $s_i = 1$ if $i \in \mathcal{S}$ otherwise $s_i = 0$
- 4 $t_i \sim \text{Bernoulli} \left(\frac{1}{e^{\epsilon/m} + 1} + \frac{x_i - \alpha}{\beta - \alpha} \cdot \frac{e^{\epsilon/m} - 1}{e^{\epsilon/m} + 1} \right)$
- 5 $x_i^* = s_i \cdot (2t_i - 1)$
- 6 **end**
- 7 **return** $\mathbf{x}^* = [x_1^*, \dots, x_d^*]^T$

ADDRESSING CHALLENGE #1

Approximation of Graph Convolution

- The server can estimate the first layer's graph convolution by:

$$\begin{aligned} \mathbf{x}'_u &= \frac{d(\beta - \alpha)}{2m} \cdot \frac{e^{\epsilon/m} + 1}{e^{\epsilon/m} - 1} \cdot \mathbf{x}_u^* + \frac{\alpha + \beta}{2} \\ \hat{\mathbf{h}}_{\mathcal{N}(v)} &= \text{AGGREGATE} \left(\{\mathbf{x}'_u, \forall u \in \mathcal{N}(v)\} \right) \\ \mathbf{h}_v &= \text{UPDATE} \left(\hat{\mathbf{h}}_{\mathcal{N}(v)} \right) \end{aligned}$$

Theorem 3.1

The multi-bit mechanism satisfies ϵ -LDP for each node.

Proposition 3.5

The optimal value of the sampling parameter m in the multi-bit mechanism is:

$$m^* = \max(1, \min(d, \lfloor \frac{\epsilon}{2.18} \rfloor))$$

Challenge #2: Small-size neighborhood

- Lots of the nodes have too few neighbors (remember Power-Law degree distribution?)
- The neighborhood aggregator cannot cancel out the noise if the neighborhood size is small

ADDRESSING CHALLENGE #2

KProp convolution layer: neighborhood expansion method

- Expands the neighborhood to the nodes that are up to K-hops away
- Applies K consecutive **AGGREGATE** function
- Applies the **UPDATE** function after the K-th **AGGREGATE**
- Trade-off between the aggregation estimation error and the GNN expressive power

Algorithm 2: KProp Convolution Layer

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$; linear aggregator function **AGGREGATE**; Non-linear update function **UPDATE**; step parameter $K \geq 1$;

Output: embedding vector $\mathbf{h}_v, \forall v \in \mathcal{V}$

```
1 for all  $v \in \mathcal{V}$  do in parallel
2    $\mathbf{h}_{\mathcal{N}(v)}^0 = \mathbf{x}_v$ 
3   for  $k = 1$  to  $K$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k =$ 
          AGGREGATE  $\left( \{\mathbf{h}_{\mathcal{N}(u)}^{k-1}, \forall u \in \mathcal{N}(v) - \{v\}\} \right)$ 
5   end
6    $\mathbf{h}_v = \text{UPDATE} \left( \mathbf{h}_{\mathcal{N}(v)}^K \right)$ 
7 endfor
8 return  $\{\mathbf{h}_v, \forall v \in \mathcal{V}\}$ 
```

EXPERIMENTS

Statistics of Datasets

DATASET	#CLASSES	#NODES	#EDGES	#FEATURES	AVG. DEG.
CORA	7	2,708	5,278	1,433	3.90
CITESEER	6	3,327	4,552	3,703	2.74
PUBMED	3	19,717	44,324	500	4.50
FACEBOOK	4	22,470	170,912	4,714	15.21
GITHUB	2	37,700	289,003	4,005	15.33
LASTFM	18	7,624	27,806	7,842	7.29

Comparison methods

- **GCN+RAW**: A standard two-layer GCN trained on raw features (**non-private**)
- **LPGNN**: A two layer GNN (KProp as the first, GCN as the second layer) trained on perturbed features using the multi-bit mechanism (**locally-private**)
- **GCN+RND**: Similar to GCN+RAW, but trained on random features (**fully-private**)
- **GCN+OHD**: Similar to GCN+RAW, but trained on “one-hot degree” features (**fully-private**)

ANALYSIS OF ACCURACY-PRIVACY TRADE-OFF

Micro-F1 score of different methods for node classification under varying privacy budget (ϵ)

DATASET	GCN +RAW	LPGNN				GCN +RND	GCN +OHD
		$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 1.0$	$\epsilon = 2.0$		
CORA	87.5 ± 0.2	81.4 ± 4.8	83.3 ± 1.5	83.6 ± 1.1	83.6 ± 0.7	58.1 ± 7.8	29.3 ± 0.2
CITESEER	74.1 ± 0.3	64.5 ± 1.1	66.0 ± 1.0	66.5 ± 0.9	66.8 ± 0.8	29.6 ± 6.5	27.2 ± 0.1
PUBMED	87.6 ± 0.1	81.9 ± 0.3	82.0 ± 0.3	82.2 ± 0.3	82.2 ± 0.3	53.5 ± 1.1	50.3 ± 0.1
FACEBOOK	94.9 ± 0.1	92.4 ± 0.5	93.2 ± 0.3	93.4 ± 0.3	93.4 ± 0.3	31.8 ± 2.1	63.8 ± 0.3
GITHUB	87.1 ± 0.1	84.1 ± 3.4	85.7 ± 0.8	86.1 ± 0.3	86.2 ± 0.1	74.3 ± 0.0	83.7 ± 0.0
LASTFM	88.2 ± 0.3	76.3 ± 1.5	82.7 ± 1.6	84.3 ± 0.8	84.8 ± 0.7	21.8 ± 1.2	45.3 ± 0.7

EFFECT OF THE MULTI-BIT MECHANISM

Mean absolute error of the multi-bit (MBM), 1-bit (1BM), and the Analytic Gaussian (AGM) mechanisms in estimating the neighborhood aggregation



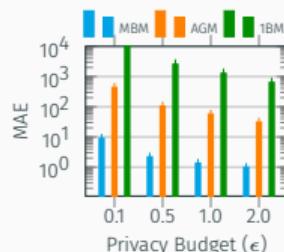
Cora



Citeseer



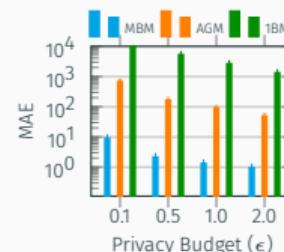
Pubmed



Facebook



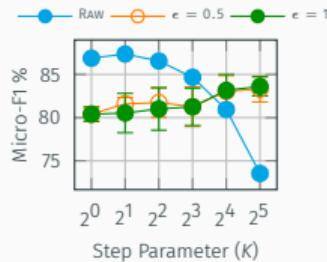
Github



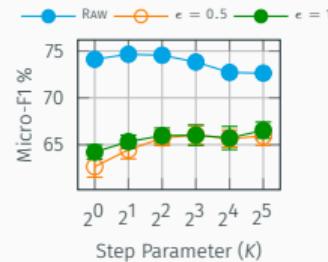
LastFM

EFFECT OF THE KPROP LAYER

Effect of the KProp step parameter (K) on the performance of LPGNN



Cora



Citeseer



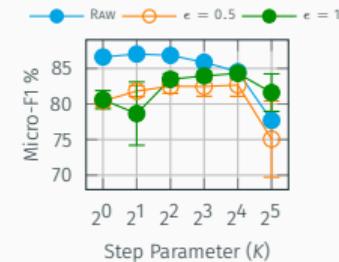
Pubmed



Facebook



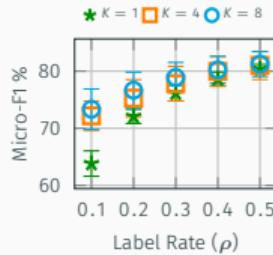
Github



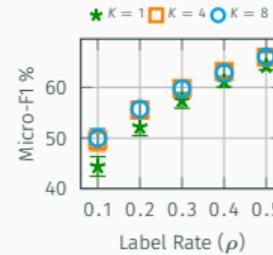
LastFM

EFFECT OF THE LABEL RATE

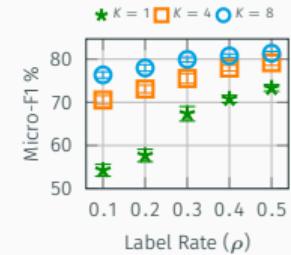
Effect of the label rate on the performance of LPGNN



Cora



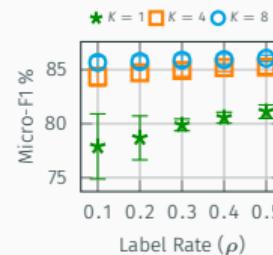
Citeseer



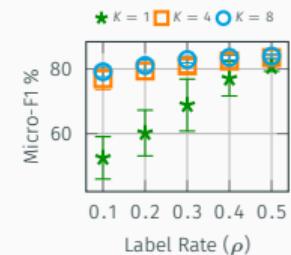
Pubmed



Facebook



Github



LastFM

RESEARCH DIRECTIONS AND CONCLUSION

Differentially Private GNNs

- How to build privacy-preserving GNNs satisfying graph-based notions of differential privacy?
- Edge-DP and Node-DP?

Privacy-Preserving Distributed GNN Learning

- How to remove the trusted server in multi-party GNN training?
- Multi-layer networks?

Privacy-Preserving Deep Graph Learning

- How to learn a latent graph from private data?
- Privacy-preserving graph-based classifier?

CONCLUSION

Graphs are likely to be sensitive

- social connections, financial transactions, disease outbreak, ...

Graph representation algorithms are vulnerable to privacy attacks

- Simple but effective attacks have already been proposed

Common privacy-preserving ML methods cannot trivially be applied on graphs

- e.g., the exhaustive communication cost of federated learning

Privacy-preserving graph representation learning aims to address privacy issue of applying deep learning over graphs

- This is a new-born field of research with lots of opportunities and open questions

If you are interested, please get in touch!

THANK YOU!

Questions?

@sajadmanesh 

sajadmanesh@idiap.ch 

REFERENCES I

-  Ahmad, I., Akhtar, M. U., Noor, S., and Shahnaz, A. (2020).
Missing link prediction using common neighbor and centrality based parameterized algorithm.
Scientific Reports, 10(1):1–9.
-  Duddu, V., Boutet, A., and Shejwalkar, V. (2020).
Quantifying privacy leakage in graph embedding.
arXiv preprint arXiv:2010.00906.
-  Hamilton, W., Ying, Z., and Leskovec, J. (2017).
Inductive representation learning on large graphs.
In *Advances in neural information processing systems*, pages 1024–1034.

REFERENCES II

-  He, X., Jia, J., Backes, M., Gong, N. Z., and Zhang, Y. (2021).
Stealing links from graph neural networks.
In *30th USENIX Security Symposium (USENIX Security 21)*, Vancouver, B.C. USENIX Association.
-  Jiang, M., Jung, T., Karl, R., and Zhao, T. (2020).
Federated dynamic gnn with secure aggregation.
arXiv preprint arXiv:2009.07351.
-  Johnson, J., Gupta, A., and Fei-Fei, L. (2018).
Image generation from scene graphs.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228.

REFERENCES III

-  Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011).
What can we learn privately?
SIAM Journal on Computing, 40(3):793–826.
-  Kipf, T. N. and Welling, M. (2017).
Semi-supervised classification with graph convolutional networks.
In *International Conference on Learning Representations (ICLR)*.
-  Li, K., Luo, G., Ye, Y., Li, W., Ji, S., and Cai, Z. (2020).
Adversarial privacy preserving graph embedding against inference attack.
IEEE Internet of Things Journal.

-  Perozzi, B., Al-Rfou, R., and Skiena, S. (2014).
Deepwalk: Online learning of social representations.
In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.
-  Sajadmanesh, S. and Gatica-Perez, D. (2020).
Locally private graph neural networks.
arXiv preprint arXiv:2006.05535.
-  Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826.

REFERENCES V

-  Zheng, L., Zhou, J., Chen, C., Wu, B., Wang, L., and Zhang, B. (2020).
Asfgnn: Automated separated-federated graph neural network.
arXiv preprint arXiv:2011.03248.
-  Zhou, J., Chen, C., Zheng, L., Zheng, X., Wu, B., Liu, Z., and Wang, L. (2020).
Privacy-preserving graph neural network for node classification.
arXiv preprint arXiv:2005.11903.