GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation

Sina Sajadmanesh Idiap Research Institute, EPFL

Joint work with Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez

August 2022

- **Graph Neural Networks** (GNNs) are state-of-the-art algorithms for learning on graphs
  - *Tasks*: node classification, link prediction, ...
  - *Applications*: recommendation systems, credit issuing, traffic forecasting, drug discovery, ...

- **Graph Neural Networks** (GNNs) are state-of-the-art algorithms for learning on graphs
    - *Tasks*: node classification, link prediction, ...
    - *Applications*: recommendation systems, credit issuing, traffic forecasting, drug discovery, ...
- Graph data could be privacy-sensitive and contain personal information
    - e.g., social networks, financial networks, ...

Introduction

- **Graph Neural Networks** (GNNs) are **state-of-the-art** algorithms for learning on graphs
  - *Tasks*: node classification, link prediction, ...
  - *Applications*: recommendation systems, credit issuing, traffic forecasting, drug discovery, ...
- Graph data could be **privacy-sensitive** and contain personal information
  - e.g., social networks, financial networks, ...
- GNN's are vulnerable to **privacy attacks**
  - Link stealing attack [He et al., 2021a, Wu et al., 2021]
  - Node membership inference attack [Olatunji et al., 2021, He et al., 2021b]

Introduction

- **Graph Neural Networks** (GNNs) are **state-of-the-art** algorithms for learning on graphs
  - *Tasks*: node classification, link prediction, ...
  - *Applications*: recommendation systems, credit issuing, traffic forecasting, drug discovery, ...
- Graph data could be **privacy-sensitive** and contain personal information
  - e.g., social networks, financial networks, ...
- GNN's are vulnerable to **privacy attacks**
  - Link stealing attack [He et al., 2021a, Wu et al., 2021]
  - Node membership inference attack [Olatunji et al., 2021, He et al., 2021b]

Our Goal:

*Making GNNs privacy-preserving using **Differential Privacy***

$\mathbf{A}$: Adjacency matrix

$\mathbf{X}$: Input node features

$\mathbf{Y}$: Predicted node labels

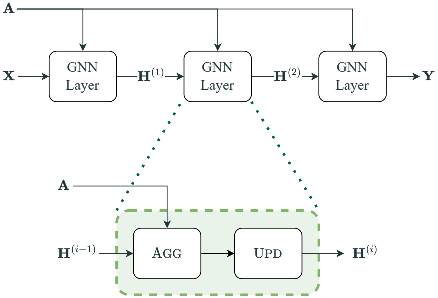
$\mathbf{H}^{(i)}$: Hidden node representations of layer i

Agg: Aggregation function

- e.g., summation: $\textsc{Agg}(\mathbf{H}, \mathbf{A}) = \mathbf{A}^T \cdot \mathbf{H}$

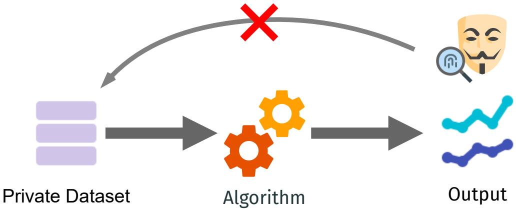
Upd: Learnable update function

- e.g., an MLP



Differential Privacy

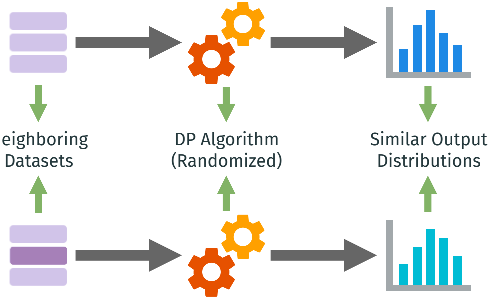
- An algorithm is executed on the private dataset and the output is publically released
- An adversary should not be able to learn about the private dataset by analyzing the output



Differential Privacy [Dwork et al., 2006]

Randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring datasets $D \simeq D'$ and all sets of outputs S:

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D') \in S] + \delta$$



Differential Privacy [Dwork et al., 2006]

Randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring datasets $D \simeq D'$ and all sets of outputs S:

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D') \in S] + \delta$$

- The probability bound captures how much protection we get
  - ϵ quantifies information leakage
    - Often called **privacy budget**
  - δ allows for a small probability of failure
    - Usually very small ($\delta <<$ *inverse number of records*)

Differential Privacy [Dwork et al., 2006]

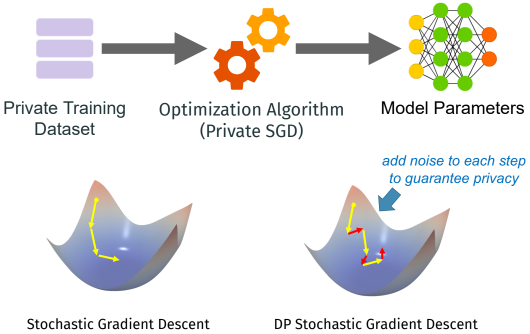
Randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring datasets $D \simeq D'$ and all sets of outputs S:

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D') \in S] + \delta$$

- The neighboring relation captures what is protected
  - **Standard DP:** D and D' differ by at most **one record**
  - **Edge-level DP:** D and D' are graphs differing by at most **one edge**
  - **Node-level DP:** D and D' are graphs differing by at most **one node** (and all its adjacent edges)

Differentially Private ML

Differentially private learning is possible with noisy gradient descent



DP-SGD Algorithm [Abadi et al., 2016]

input : Data $\{\vec{x}_1 \ldots, \vec{x}_N\}$, learning rate η, batch size B, epochs T, clipping threshold C, noise variance σ^2,

Initialize $\vec{\theta}_0$ randomly

for $t \in [T \cdot \frac{N}{B}]$ **do**

  Sample a batch $\vec{B}_t$ by selecting each $\vec{x}_i$ independently with probability $\frac{B}{N}$

  For each $\vec{x}_i \in \vec{B}_t$: $\vec{g}_t(\vec{x}_i) \leftarrow \nabla_{\vec{\theta}_t} L(\vec{\theta}_t, \vec{x}_i)$ `// compute per-sample gradients`

  $\tilde{\vec{g}}_t(\vec{x}_i) \leftarrow \mathrm{clip}(\vec{g}_t(\vec{x}_i), C)$ `// clip gradients to max norm` C

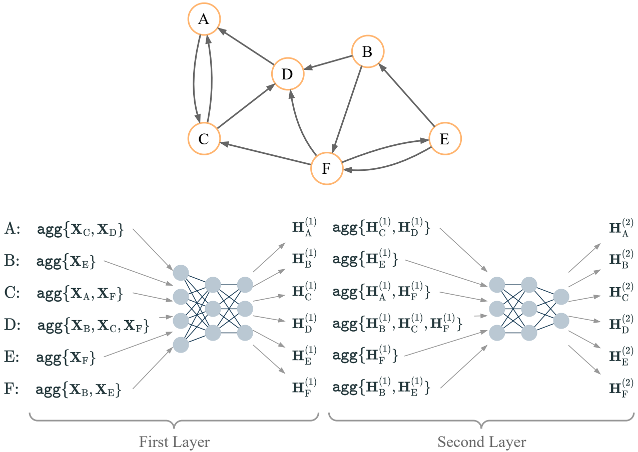
  $\tilde{\vec{g}}_t \leftarrow \frac{1}{B}\left(\sum_{\vec{x}_i \in \vec{B}_t} \tilde{\vec{g}}_t(\vec{x}_i) + \mathcal{N}(0, \sigma^2 \vec{I})\right)$ `// add Gaussian noise with variance` σ^2

  $\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t - \eta \tilde{\vec{g}}_t$ `// SGD step`

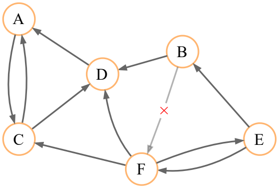
end

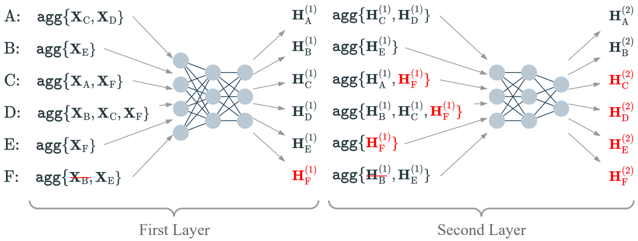
output: $\vec{\theta}_{\frac{TN}{B}}$

DP GNN Challenges: Exploding Sensitivity

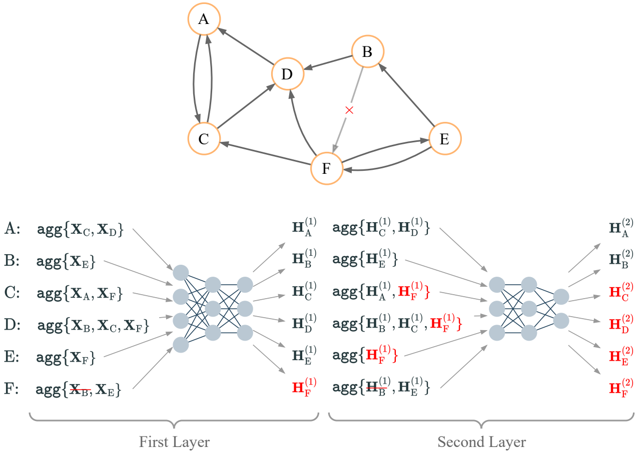


DP GNN Challenges: Exploding Sensitivity



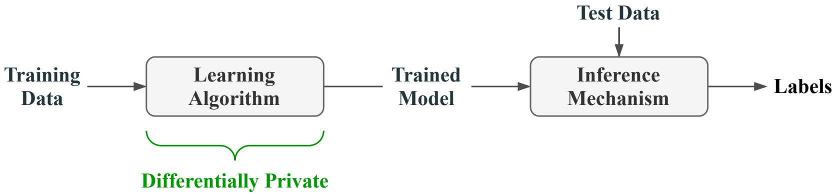


DP GNN Challenges: Exploding Sensitivity



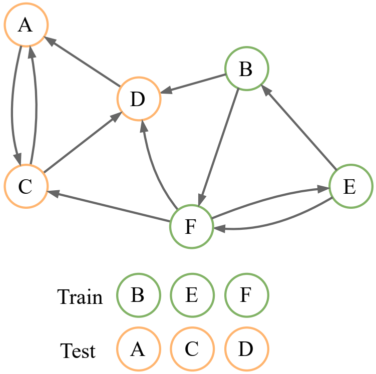
The number of affected outputs = $\mathcal{O}(\text{max degree}^{\text{num layers}})$

Private Learning: Standard Neural Nets

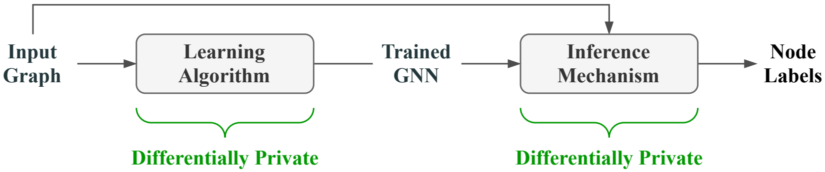


Inference is independent of the training data

- GNN re-uses graph data for inference
- Private information leaks at inference, even with a private model



Private Learning: Graph Neural Nets



Both training and inference should be private

Our Approach: Aggregation Perturbation

- The sensitivity of a single aggregation step is easily computed
  - Only **one node** is affected for **edge-level DP**
  - Maximum **D nodes** are affected for **node-level DP** (D is maximum degree)
- Aggregation Perturbation: adding noise to output of the aggregation step
  - Prevents the exploding sensitivity problem by composing differentially private aggregation steps
  - Ensures inference privacy
- Applying aggregation perturbation to the conventional GNNs is costly
  - Every forward pass of the model consumes privacy budget
  - The excessive noise results in poor performance

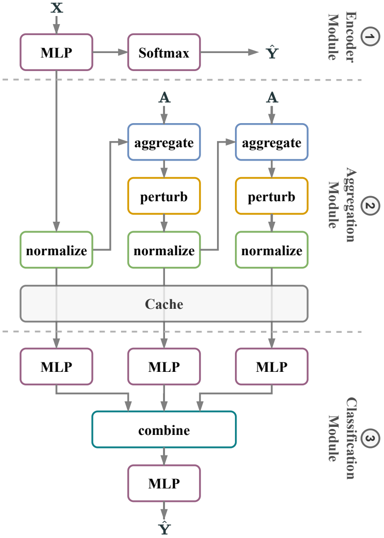
Our Approach: Aggregation Perturbation

- The sensitivity of a single aggregation step is easily computed
  - Only **one node** is affected for **edge-level DP**
  - Maximum **D nodes** are affected for **node-level DP** (D is maximum degree)
- Aggregation Perturbation: adding noise to output of the aggregation step
  - Prevents the exploding sensitivity problem by composing differentially private aggregation steps
  - Ensures inference privacy
- Applying aggregation perturbation to the conventional GNNs is costly
  - Every forward pass of the model consumes privacy budget
  - The excessive noise results in poor performance

Need to tailor the GNN architecture to the private learning setting!

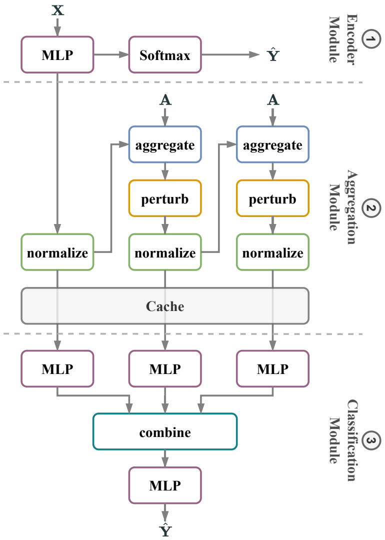
GNN with Aggregation Perturbation (GAP)

1. Encoder Module
    - Learns to encode node features into lower-dimensional representations
    - Does not use graph adjacency information



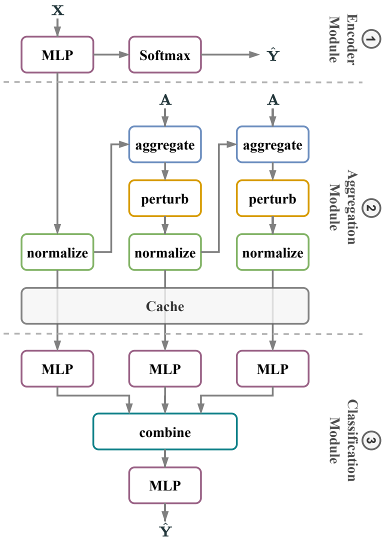
GNN with Aggregation Perturbation (GAP)

1. Encoder Module
   - Learns to encode node features into lower-dimensional representations
   - Does not use graph adjacency information
2. Aggregation Module
   - Computes aggregated node representations at multiple hops privately using the aggregation perturbation approach
   - Uses graph adjacency information



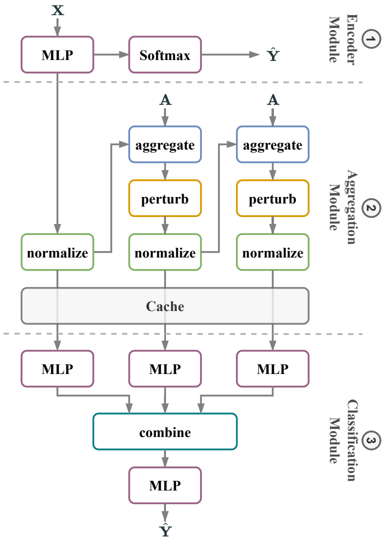
GNN with Aggregation Perturbation (GAP)

1. Encoder Module
    - Learns to encode node features into lower-dimensional representations
    - Does not use graph adjacency information
2. Aggregation Module
    - Computes aggregated node representations at multiple hops privately using the aggregation perturbation approach
    - Uses graph adjacency information
3. Classification Module
    - Learns to perform node-wise classification based on aggregated node representations
    - Does not re-use graph adjacency information



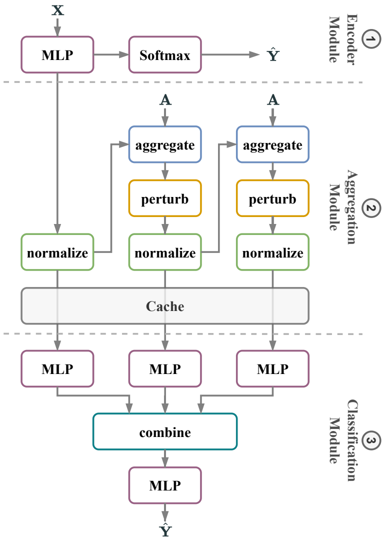
Advantages of GAP Architecture

✓ Edge-level DP

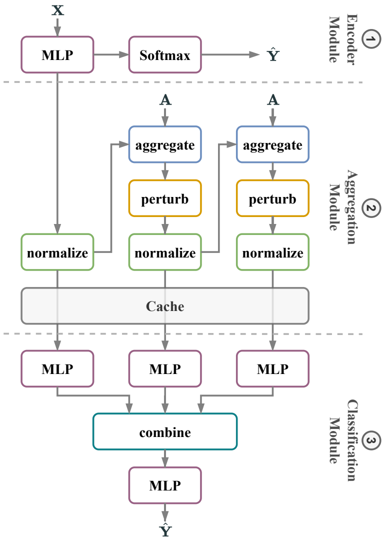


Advantages of GAP Architecture

- ✓ Edge-level DP
- ✓ Node-level DP through combination with DP-SGD
    - For bounded-degree graphs

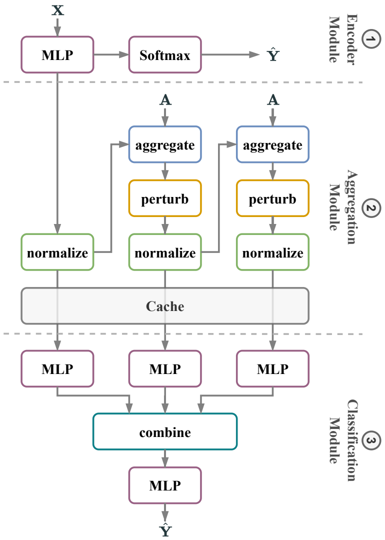


- ✓ Edge-level DP
- ✓ Node-level DP through combination with DP-SGD
    - For bounded-degree graphs
- ✓ Multi-hop aggregations



Advantages of GAP Architecture

- ✓ Edge-level DP
- ✓ Node-level DP through combination with DP-SGD
  - For bounded-degree graphs
- ✓ Multi-hop aggregations
- ✓ Zero-cost inference privacy



For any $\delta \in (0, 1)$, number of hops $K \geq 0$, and noise standard deviation $\sigma > 0$, GAP's training algorithm satisfies edge-level (ϵ, δ)-DP with:

$$\epsilon = \frac{K}{2\sigma^2} + \sqrt{2K \log(1/\delta)}/\sigma$$

For any $\delta \in (0, 1)$, number of nodes N, batch-size $0 < B < N$, number of epochs T, gradient clipping threshold $C > 0$, number of hops $K \geq 0$, maximum cut-off degree $D \geq 1$, and noise standard deviation $\sigma > 0$, GAP's training algorithm satisfies node-level (ϵ, δ)-DP with:

$$\begin{aligned}\epsilon \leq \min_{\alpha} \quad & 2T\frac{N}{B}\frac{1}{\alpha-1}\log\left\{\left(1-\frac{B}{N}\right)^{\alpha-1}\left(\alpha\frac{B}{N}-\frac{B}{N}+1\right)+\binom{\alpha}{2}\left(\frac{B}{N}\right)^{2}\left(1-\frac{B}{N}\right)^{\alpha-2}e^{\frac{C^2}{\sigma^2}}\right. \\ & \left.+\sum_{l=3}^{\alpha}\binom{\alpha}{l}\left(1-\frac{B}{N}\right)^{\alpha-l}\left(\frac{B}{N}\right)^{l}e^{(l-1)(\frac{C^2 l}{2\sigma^2})}\right\}+\frac{DK\alpha}{2\sigma^2}+\frac{\log(1/\delta)}{\alpha-1} \\ \text{s.t.} \quad & \alpha > 1.\end{aligned}$$

Experiment Settings: Datasets

- **Task:** Node Classification

Dataset	Classes	Nodes	Edges	Features	Avg. Degree
Facebook	6 Year	26,406 User	2,117,924 Friendship	501	62
Reddit	8 Community	116,713 Post	46,233,380 Mutual User	602	209
Amazon	10 Category	1,790,731 Product	80,966,832 Mutual Purchase	100	22

Experiment Settings: Competing Methods

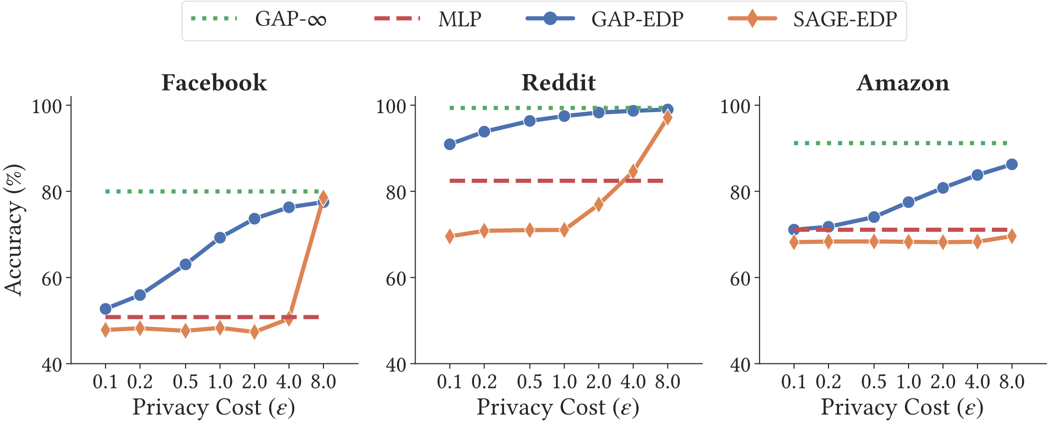
- Edge-Level Private Methods
  - **GAP-EDP:** Our edge-level private method
  - **SAGE-EDP:** Graph-SAGE with adjacency matrix perturbation [Wu et al., 2021]
  - **MLP:** Simple MLP model that does not use the graph edges
- Node-Level Private Methods
  - **GAP-NDP:** Our node-level private method
  - **SAGE-NDP:** 1-layer Graph-SAGE with gradient perturbation [Daigavane et al., 2021]
  - **MLP-DP:** Simple MLP model trained with DP-SGD
- None-Private Methods
  - **GAP-∞:** GAP without noise
  - **SAGE-∞:** Standard Graph-SAGE [Hamilton et al., 2017]

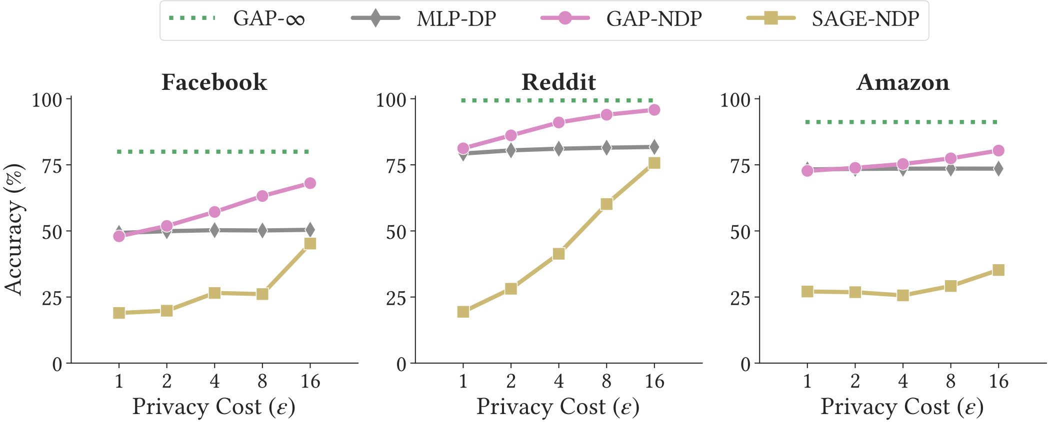
Comparison of Non-Private Methods

Accuracy of Non-Private Methods

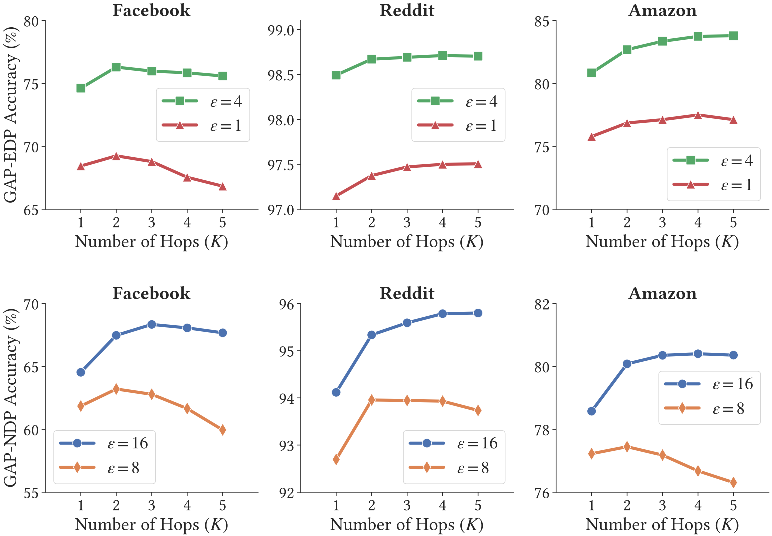
Method	Facebook	Reddit	Amazon
GAP-∞	80.0 ± 0.48	**99.4 ± 0.02**	91.2 ± 0.07
SAGE-∞	**83.2 ± 0.68**	99.1 ± 0.01	**92.7 ± 0.09**

Edge-Level DP Accuracy-Privacy Trade-Off

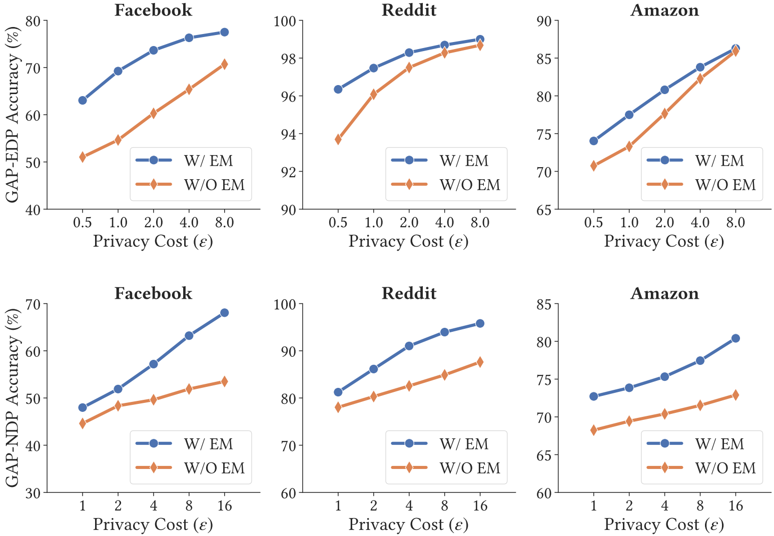



GAP-∞
MLP-DP
GAP-NDP
SAGE-NDP
Facebook
Reddit
Amazon
Accuracy (%)
100
75
50
25
0
1
2
4
8
16
Privacy Cost (ε)

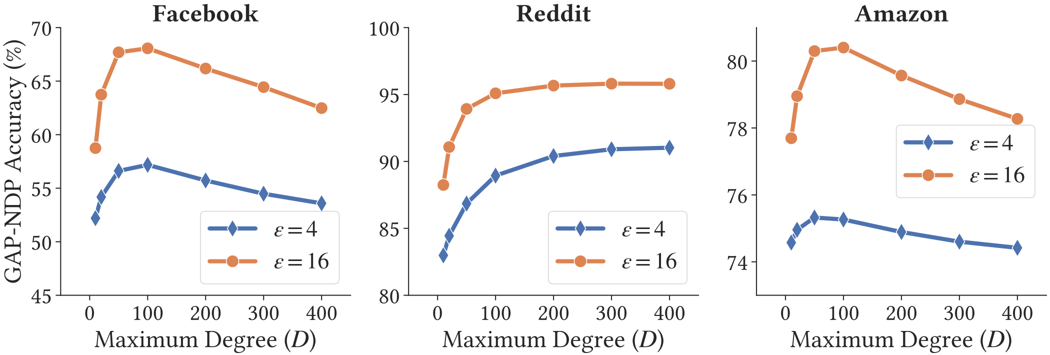
Effect of the Number of Hops



Effect of the Encoder Module



Effect of the Maximum Degree



Conclusion

- Implementing DP in GNNs is challenging
  - Exploding sensitivity
  - Inference privacy
- Our contribution: GAP
  - Ensures both edge-level and node-level DP
  - Supports multi-hop aggregations
  - Provides inference privacy

Thank You!

Questions?

sajadmanesh@idiap.ch

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016).
Deep learning with differential privacy.
In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Daigavane, A., Madan, G., Sinha, A., Thakurta, A. G., Aggarwal, G., and Jain, P. (2021).
Node-level differentially private graph neural networks.
arXiv preprint arXiv:2111.15521.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006).
Calibrating noise to sensitivity in private data analysis.
In *Theory of cryptography conference*, pages 265–284. Springer.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017).
Inductive representation learning on large graphs.
In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

He, X., Jia, J., Backes, M., Gong, N. Z., and Zhang, Y. (2021a).
Stealing links from graph neural networks.
In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

He, X., Wen, R., Wu, Y., Backes, M., Shen, Y., and Zhang, Y. (2021b).
Node-level membership inference attacks against graph neural networks.
arXiv preprint arXiv:2102.05429.

References III

Olatunji, I. E., Nejdl, W., and Khosla, M. (2021).
Membership inference attack on graph neural networks.
arXiv preprint arXiv:2101.06570.

Wu, F., Long, Y., Zhang, C., and Li, B. (2021).
Linkteller: Recovering private edges from graph neural networks via influence analysis.
arXiv preprint arXiv:2108.06504.