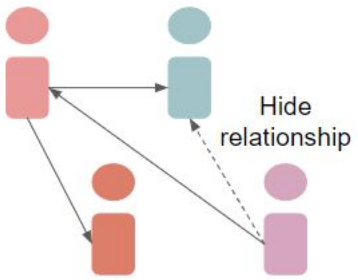
GNNDelete: A General Strategy for Unlearning in Graph Neural Networks

Jiali Cheng[1], George Dasoulas[2,*], Huan He[2,*], Chirag Agarwal[3], Marinka Zitnik[2]

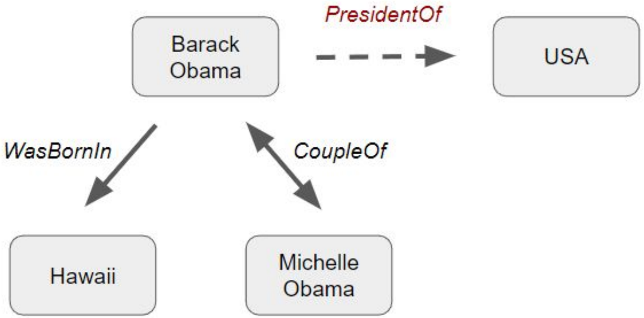
[1]University of Massachusetts Lowell, [2]Harvard University, [3]Adobe

Motivation for Graph Unlearning

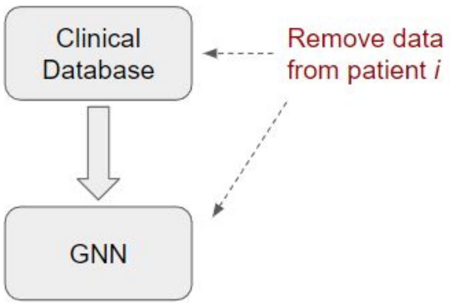
- Graph unlearning involves deleting graph elements such as nodes, node labels, and relationships from trained graph neural network (GNN) models
- Retraining models from scratch is not feasible. Needed are **efficient methods for model editing**



Graph elements become irrelevant or inaccurate



Underlying graphs evolving over time



Sensitive data and growing demands for privacy

Why is Graph Unlearning Challenging?

1. Graph elements exert strong influence on other elements with dependencies between nodes connected by edges
   Challenge: **Existing machine unlearning methods are unsuitable for data with underlying geometric and relational structure**

2. Graph models make predictions by propagating messages across local neighborhoods
   Challenge: **Adversarial agents can infer the presence of graph elements from their local neighbors. Merely removing data from the graph is not sufficient**

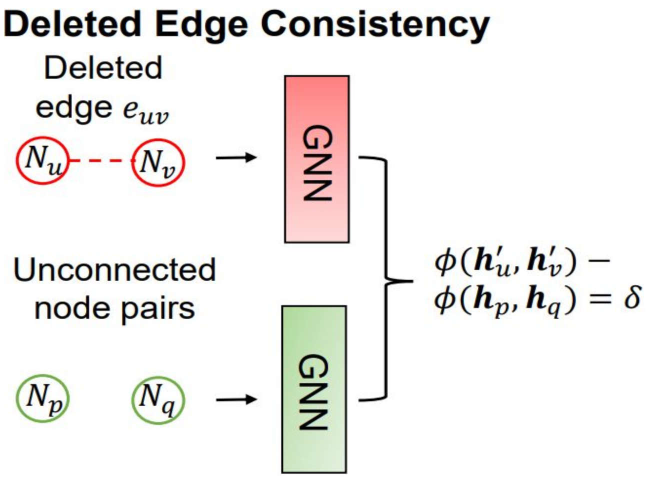
3. GNNs share model weights across many (often all) nodes or edges in the graph
   Challenge: **Naively perturbing model weights deteriorates model performance. Methods developed for other modalities are not suitable for graphs**

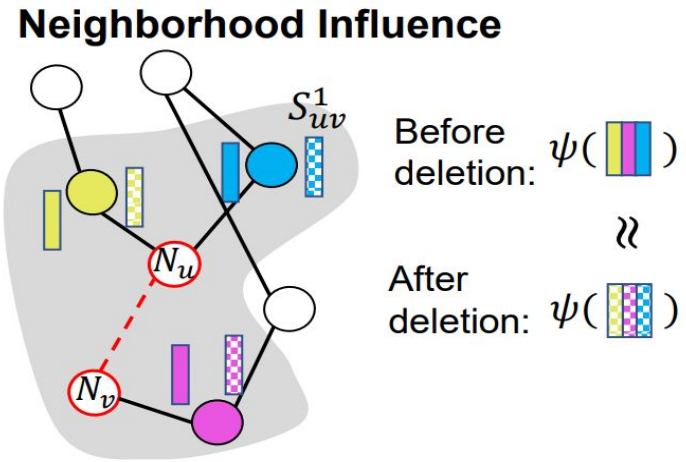
Requirements for Successful Graph Deletion

Shown is a motivating example of deleting a single edge e_{uv}

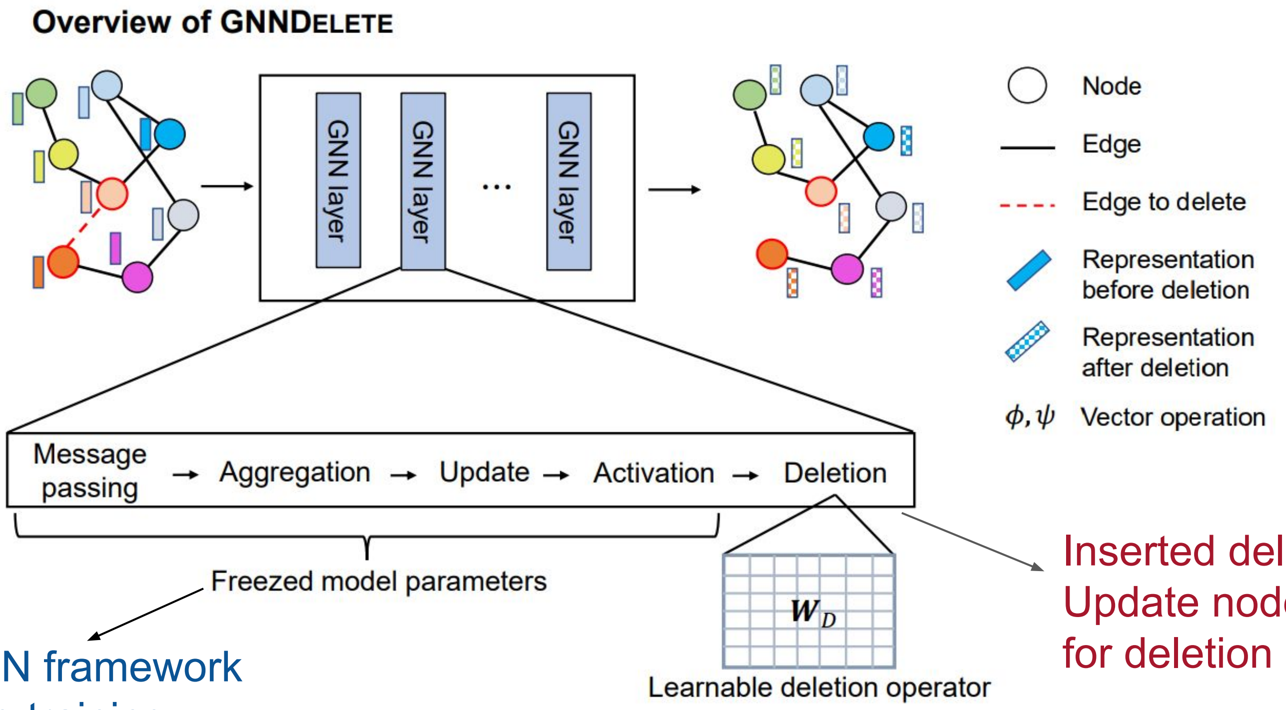
After deletion, GNNDelete treats e_{uv} as an unconnecetd node pair

After deletion, GNNDelete keeps node embeddings close to original ones





Layer-wise Deletion Operator in GNNDelete



$$W_D = \arg\min_{W_D} \lambda L_{DEC} + (1 - \lambda) L_{NI}$$

Standard GNN framework
Frozen during training

Inserted deletion operator
Update node embeddings for deletion

Graph Neural Network Model Unlearning

$$\mathcal{L}^l_{\text{DEC}} = \mathcal{L}(\{[\boldsymbol{h}'^l_u; \boldsymbol{h}'^l_v] | e_{uv} \in \mathcal{E}_d\}, \{[\boldsymbol{h}^l_u; \boldsymbol{h}^l_v] | u, v \in_R \mathcal{V}\})$$

Deleted edges Similar to **Unconnected node pairs**

$$\mathcal{L}^l_{\text{NI}} = \mathcal{L}(\|_w \{\boldsymbol{h}'^l_w | w \in S^l_{uv} / e_{uv}\}, \|_w \{\boldsymbol{h}^l_w | w \in S^l_{uv}\})$$

Embeddings after deletion Similar to **Embeddings before deletion**

$$\boldsymbol{W}^{l*}_D = \arg\min_{\boldsymbol{W}^l_D} \mathcal{L}^l = \arg\min_{\boldsymbol{W}^l_D} \lambda \mathcal{L}^l_{\text{DEC}} + (1-\lambda)\mathcal{L}^l_{\text{NI}}$$

Local update

$$\text{DEL}^l = \begin{cases} \phi & \text{if } w \in S^l_{uv} \\ \mathbb{1} & \text{otherwise} \end{cases}$$

Only train deletion operators.
Other parameters are frozen.

Results 1: Excellent Performance on Edge Deletion Benchmarks

GNNDelete outperforms all baselines by 30.7 ($\mathcal{E}_t$) and 25.1 ($\mathcal{E}_d$) on average, other graph unlearning methods by 24.1% ($\mathcal{E}_t$) and 28.5 ($\mathcal{E}_d$) on average

Predict test edges accurately

Distinguish between deleted and non-deleted edges

Reference

Other graph unlearning methods

Model	GCN		GAT		R-GCN		R-GAT	
	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
RETRAIN	0.964 ±0.003	0.506 ±0.013	0.956 ±0.002	0.525 ±0.012	0.800 ±0.005	0.580 ±0.006	0.891 ±0.005	0.783 ±0.009
GRADASCENT	0.555 ±0.066	0.594 ±0.063	0.501 ±0.020	0.592 ±0.017	0.490 ±0.001	0.502 ±0.002	0.490 ±0.001	0.492 ±0.003
D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
GRAPHERASER	0.527 ±0.002	0.500 ±0.000	0.538 ±0.013	0.500 ±0.000	0.512 ±0.003	0.500 ±0.000	0.545 ±0.015	0.500 ±0.000
GRAPHEDITOR	0.776 ±0.025	0.432 ±0.009	-	-	N/A	N/A	N/A	N/A
CERTUNLEARN	0.718 ±0.032	0.475 ±0.011	-	-	N/A	N/A	N/A	N/A
GNNDELETE	**0.934 ±0.002**	**0.748 ±0.006**	**0.914 ±0.007**	**0.774 ±0.015**	**0.751 ±0.006**	**0.845 ±0.007**	**0.893 ±0.002**	**0.786 ±0.004**

GNNDelete achieves the highest AUROC on both settings using different architectures

Results 2: Ablation Study

On the interplay of Deleted Edge Consistency (DEC) and Neighborhood Influence (NI)

$$\boldsymbol{W}_D^{l^*} = \arg\min_{\boldsymbol{W}_D^l} \mathcal{L}^l = \arg\min_{\boldsymbol{W}_D^l} \lambda \mathcal{L}_{\text{DEC}}^l + (1-\lambda)\mathcal{L}_{\text{NI}}^l$$

Considering performance on both $\mathcal{E}_t$ and $\mathcal{E}_d$.

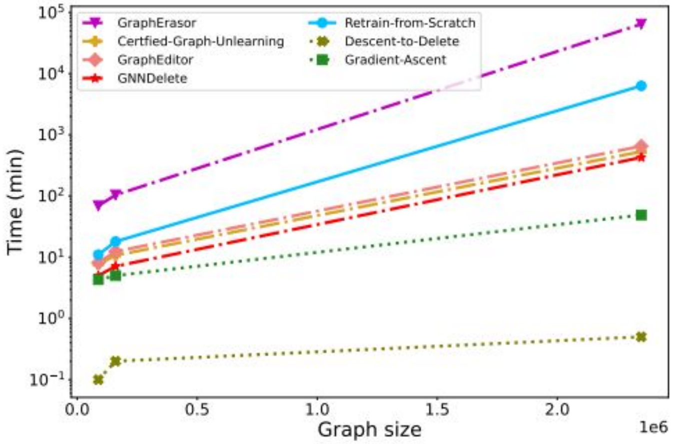
Deleted Edge Consistency and **Neighborhood Influence** are both necessary for successful unlearning on graphs

λ	AUROC on $\mathcal{E}_t$	AUROC on $\mathcal{E}_d$	Avg. AUROC (Gap)
0.0	0.964 ±0.003	0.492 ±0.012	0.728 (0.473)
0.2	0.961 ±0.003	0.593 ±0.011	0.777 (0.368)
0.4	0.950 ±0.005	0.691 ±0.010	0.821 (0.259)
0.5	0.934 ±0.002	0.748 ±0.006	**0.841 (0.185)**
0.6	0.927 ±0.001	0.739 ±0.006	0.834 (0.188)
0.8	0.893 ±0.003	0.759 ±0.008	0.823 (0.134)
1.0	0.858 ±0.004	0.757 ±0.004	0.808 (0.101)

Results 3: GNNDelete is Computationally Efficient

GNNDelete demonstrates efficiency in terms of both its training time and the number of trainable parameters it requires

Training time



Trainable parameters

Model	OGB-Collab	OGB-BioKG
RETRAIN	5,216	12,009,792
GRADASCENT	5,216	12,009,792
D2D	5,216	12,009,792
GRAPHERASER	52,160	120,097,920
GRAPHEDITOR	5,216	N/A
CERTUNLEARN	5,216	N/A
GNNDELETE	5,120	5,120

Compared to GraphEraser: GNNDelete saves **~9x training time** and **~10x & ~23000x space**.

GNNDelete is a General Strategy for Graph Unlearning!

- GNNDelete is a novel deletion operator that is flexible and easy-to-use and **can be used with any graph neural network (GNN) model**
- We formulate two **key requirements that graph unlearning methods must satisfy, Deleted Edge Consistency** and **Neighborhood Influence** through which we can unlearn graph elements and retain strong predictive performance
- GNNDelete achieves **state-of-the-art performance across a wide range of deletion tasks including edge deletion, node deletion, and node feature unlearning**

openreview.net/pdf?id=X9yCkmT5Qrl

github.com/mims-harvard/GNNDelete