Scalable and Certifiable Graph Unlearning: Overcoming the Approximation Error Barrier

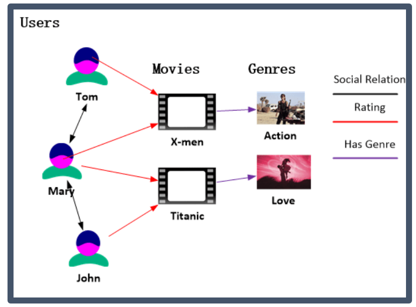
Lu Yi, Zhewei Wei*

Contact: yilu@ruc.edu.cn

https://github.com/luyi256/ScaleGUN

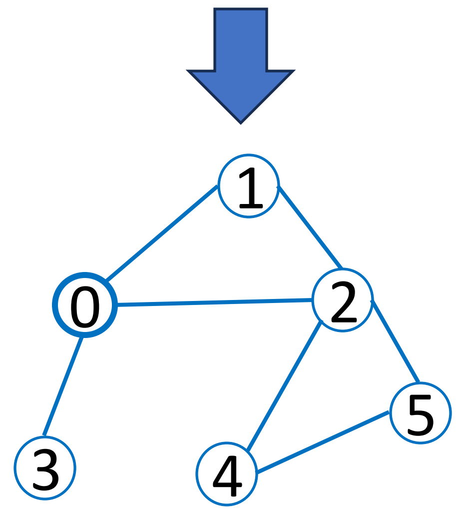
Graph

- Graphs are everywhere!
- Graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$
  - Node set $\boldsymbol{V}$
  - Edge set $\boldsymbol{E}$
  - Adjacency matrix $\boldsymbol{A}$
  - Degree matrix $\boldsymbol{D}$
  - Normalized Laplacian matrix $\boldsymbol{P} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$
  - Feature matrix $\boldsymbol{X} \in \mathcal{R}^{n \times f}$



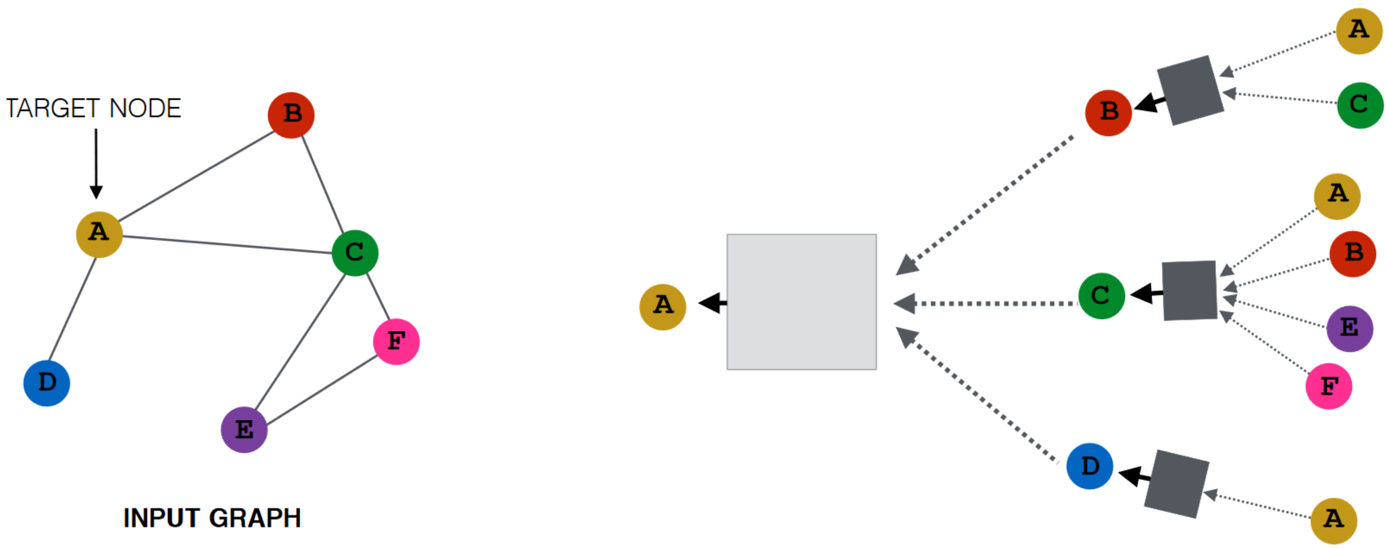
Recommendation Network

Social Network



Graph Neural Networks

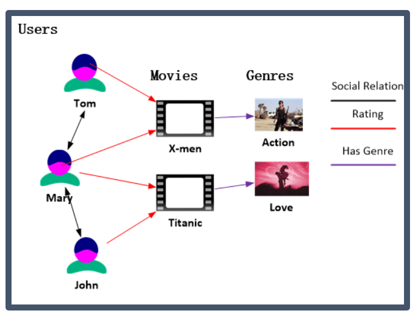
- Graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$
  - Normalized Laplacian matrix $\boldsymbol{P} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$
  - Feature matrix $\boldsymbol{X} \in \mathcal{R}^{n \times f}$
- Graph Neural Networks
  - Aggregating information from neighbors, aka **graph propagation**



  - GCN: $H = \text{softmax}\left(\widetilde{\boldsymbol{P}}\left(\sigma(\widetilde{\boldsymbol{P}}\boldsymbol{X}\boldsymbol{W}_0)\right)\boldsymbol{W}_1\right)$
  - SGC: $H = \text{softmax}(\widetilde{\boldsymbol{P}}^2\boldsymbol{X}\boldsymbol{W})$
  - Applications: recommending products for users; detecting suspicious fraudsters

Privacy meets GNNs

- What if I want to delete my account from social/shopping apps?
- Regulations ensure the "Right to be forgotten"
  - European Union's General Data Protection Regulation (GDPR)
  - California Consumer Privacy Act (CCPA)
  - Canadian Consumer Privacy Protection Act (CPPA)
- Deleting my data from the database is insufficient
- My data was used to train the GNN model!

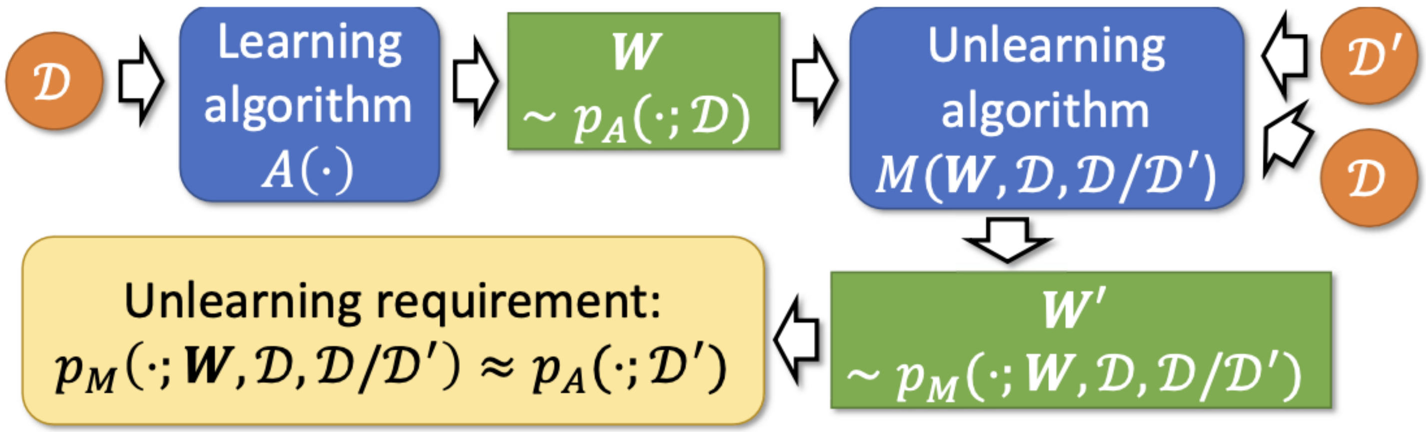


Recommendation Network

Social Network

Graph Unlearning (GU)

- Graph unlearning emerges as a solution



- Unlearning algorithm = Retraining? inefficient, impractical for frequent removal requests ☹
- Goal: A more efficient approach than retraining, while maintaining comparable model performance.

Image source: Chien et al., 2022

Exact V.S. Approximate GU

- Exact GU
  - Equivalent to retraining, privacy is fully preserved 😃
  - inefficient ☹
- Approximate GU without guarantees
  - privacy preservation is uncertain ☹
  - efficient 😃
- Approximate GU with certified guarantees
  - privacy preservation is ensured with bounded error 😃
  - efficient 😃

Certified Unlearning

- Definition
  - $\mathcal{D}$: original dataset, A: model, $h \in \mathcal{H}$: hypothesis space, $\mathcal{D}'$: dataset post-removal
  - $M(A(\mathcal{D}), \mathcal{D}, \mathcal{D}')$: An unlearning mechanism
  - (ϵ, δ)-certified unlearning:

    Given $\delta > 0$, $\epsilon > 0$, $\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D}, \mathcal{D}' \subseteq \mathcal{X}$:

$$P(M(A(\mathcal{D}), \mathcal{D}, \mathcal{D}') \in \mathcal{T}) \leq e^{\epsilon} P(A(\mathcal{D}') \in \mathcal{T}) + \delta, \text{ and}$$
$$P(A(\mathcal{D}') \in \mathcal{T}) \leq e^{\epsilon} P(M(A(\mathcal{D}), \mathcal{D}, \mathcal{D}') \in \mathcal{T}) + \delta$$

    - Approximately equivalent in terms of their probability distributions
  - A common solution: Newton update removal mechanism

$$\mathbf{w}^{-} = \mathbf{w}^{\star} + [\nabla^2 L(\mathbf{w}^{\star}, \mathcal{D}')]^{-1} [\nabla L(\mathbf{w}^{\star}, \mathcal{D}) - \nabla L(\mathbf{w}^{\star}, \mathcal{D}')]$$

    - add perturbation to the loss function

$$L_{\mathbf{b}}(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^{n} \ell(\mathbf{w}^{\top} \mathbf{Z}_i, y_i) + \frac{\lambda n}{2} \| \mathbf{w} \|_2^2 + \mathbf{b}^{\top} \mathbf{w}$$

Exiting Certified GU

- Motivation: Existing certified GU cannot scale to large graphs!

Algorithm 1: Existing GU

Input: Graph data $\mathcal{D} = (\mathbf{X}, \mathbf{Y}, \mathbf{A})$, sequence of removal requests $R = \{R_1, \cdots, R_k\}$, loss function l, parameters $L, \{w_\ell\}, r_{\max}, \alpha, \gamma_2, \epsilon, \delta$

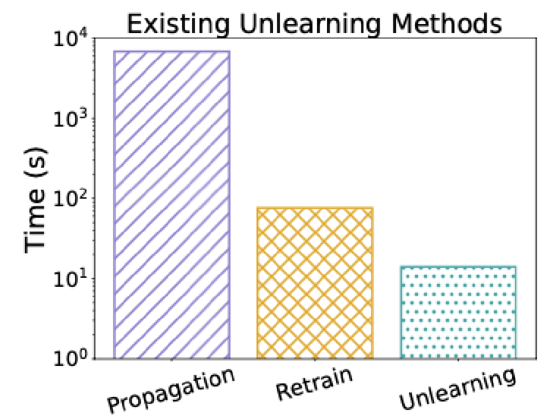
Compute the embedding matrix $\mathbf{Z}$;

$\mathbf{w} \leftarrow$ the model trained on the training set of $\mathcal{D}$ with the approximate embeddings $\mathbf{Z}$;

Accumulated unlearning error $\beta \leftarrow 0$;

for $R_i \in R$ **do**

- $\mathcal{D}' \leftarrow$ the updated dataset according to R_i;
- $\mathbf{Z}' \leftarrow$ update the embedding matrix according to the removal request R_i;
- $\beta \leftarrow$ update the accumulated error;
- **if** $\beta > \alpha\epsilon / \sqrt{2\log(1.5/\delta)}$ **then**
  - $\mathbf{w} \leftarrow$ the model retrained on the training set of $\mathcal{D}'$;
  - $\beta \leftarrow 0$;
- **else**
  - $\mathbf{w} \leftarrow$ update the model parameters;



(a)

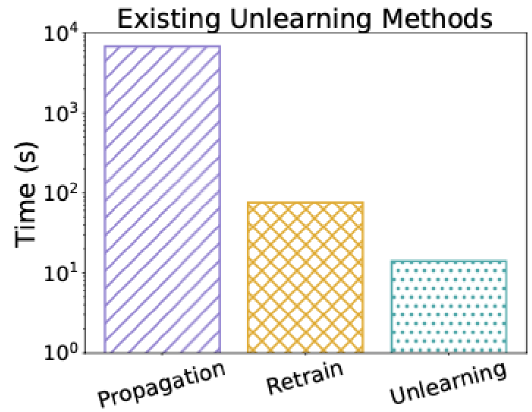
Re-propagation for each removal request is the primary bottleneck!

How can we scale up Certified GU

- Problem A. How can we accelerate graph propagation?
  - Can scalable techniques for graph learning be applied to GU?

  But all of them introduce approximation error to node embeddings

- Problem B. How can we ensure certified guarantees for GU despite the impact of approximation error?
  - Certified GU requires bounded "model error".
  - Model error: $\|\nabla L(\mathbf{w}^{-}, \mathcal{D}')\|$
  - $\boldsymbol{L}(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^{n} \ell(\mathbf{w}^{\top}\mathbf{Z}_i, y_i) + \frac{\lambda n}{2} \| \mathbf{w} \|_2^2$
  - $\mathbf{Z} = \sum_{\ell=0}^{L} w_\ell \left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)^{\ell} \mathbf{X}$

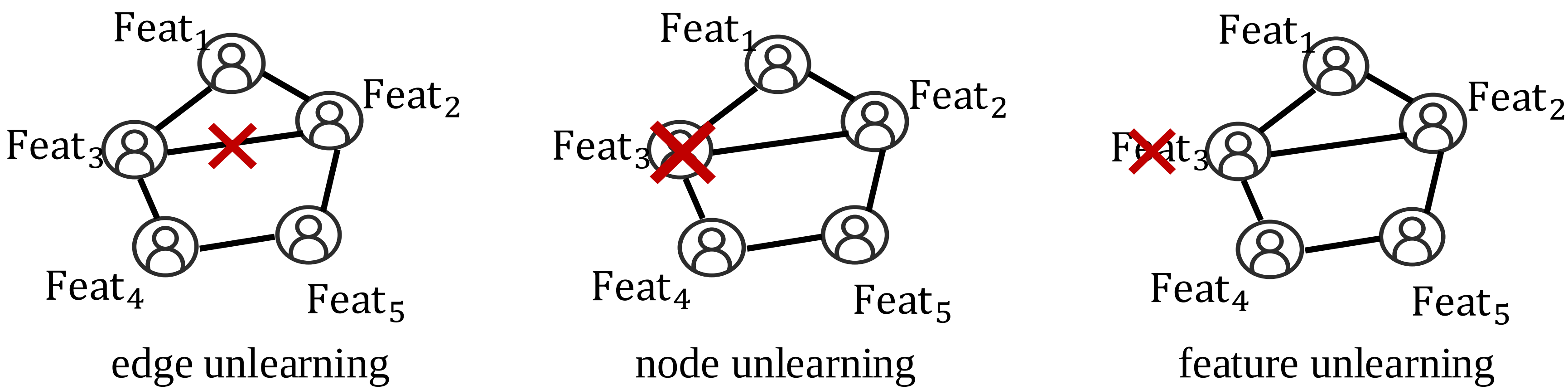


(a)

Re-propagation for each removal request is the primary bottleneck!

How can we scale up Certified GU

- Problem B. How can we ensure certified guarantees for GU despite the impact of approximation error?



  - Challenge 1. Unlearning a single edge, node, or feature impacts multiple node embeddings.
  - Challenge 2. Scalable propagation techniques yield approximate embeddings, but model error is defined based on exact embeddings.

ScaleGUN: Problem A

- How can we accelerate graph propagation ?
  - the dynamic approximate propagation techniques
  - Existing dynamic approximate propagation techniques can only be applied to Personalized PageRank (PPR):

$$\mathbf{Z} = \sum_{\ell=0}^{\infty} \alpha(1-\alpha)^{\ell} \left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{\ell} \mathbf{X}$$

  - We extend it to Generalized PageRank (GPR):

$$\mathbf{Z} = \sum_{\ell=0}^{L} w_{\ell} \left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{\ell} \mathbf{X}$$

    - GPR is more common and thus enhance the generalizability of our ScaleGUN.
  - Result: ScaleGUN processes a random edge removal request in constant time.

ScaleGUN: Problems B

- How can we ensure certified guarantees for GU under the impact of approximation error? ➔ How can we quantify the model error?

Algorithm 1: Existing GU

Input: Graph data $\mathcal{D} = (\mathbf{X}, \mathbf{Y}, \mathbf{A})$, sequence of removal requests $R = \{R_1, \cdots, R_k\}$, loss function l, parameters $L, \{w_\ell\}, r_{\max}, \alpha, \gamma_2, \epsilon, \delta$

Compute the embedding matrix $\mathbf{Z}$;

$\mathbf{w} \leftarrow$ the model trained on the training set of $\mathcal{D}$ with the approximate embeddings $\mathbf{Z}$;

Accumulated unlearning error $\beta \leftarrow 0$;

for $R_i \in R$ **do**

  $\mathcal{D}' \leftarrow$ the updated dataset according to R_i;

  $\mathbf{Z}' \leftarrow$ update the embedding matrix according to the removal request R_i;

  $\beta \leftarrow$ update the accumulated error;

  if $\beta > \alpha\epsilon / \sqrt{2\log(1.5/\delta)}$ **then**

    $\mathbf{w} \leftarrow$ the model retrained on the training set of $\mathcal{D}'$;

    $\beta \leftarrow 0$;

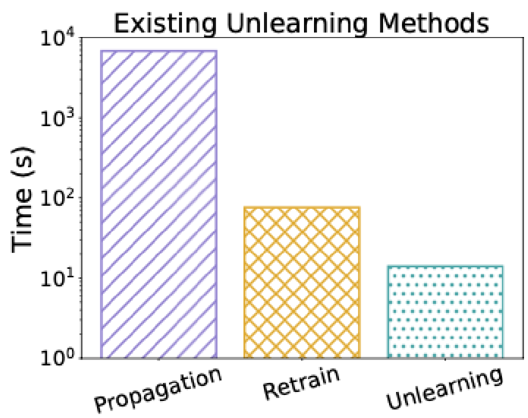
  else

    $\mathbf{w} \leftarrow$ update the model parameters;

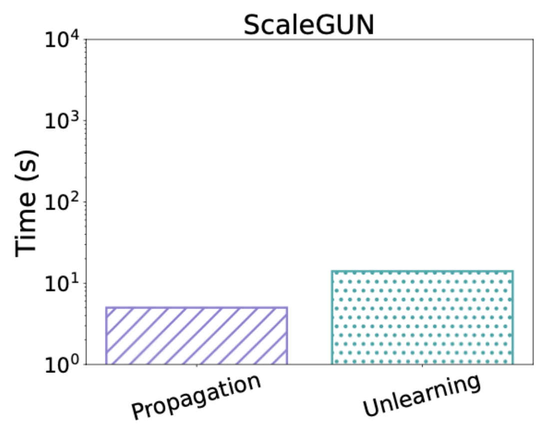
1. Connect pre- and post-removal node embeddings **using the propagation framework!**

2. Establish the relationship between the approximate embeddings and the exact embeddings via bounded approximation error.

ScaleGUN: Our results



(a)



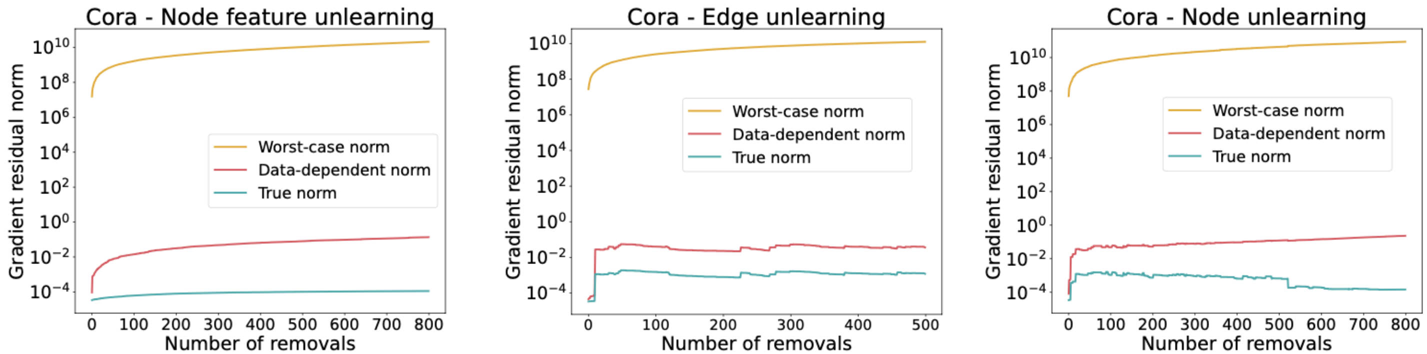
(b)

Time costs per 5,000-random-edge batch removal for existing unlearning methods v.s. ScaleGUN on ogbn-papers100M (2-hop SGC model).

Speedup: 6000 seconds → 20 seconds !

ScaleGUN: Our results

- Derive the worst-case/data-dependent bound of the model error across three unlearning scenarios.



Theorem 4.2 (Worst-case bound of node feature unlearning). *Suppose that Assumption 4.1 holds and the feature of node u is to be unlearned. If $\forall j \in [F]$, $\left\|\hat{\mathbf{Z}}\mathbf{e}_j - \mathbf{Z}\mathbf{e}_j\right\| \le \epsilon_1$, we have*

$$\|\nabla \mathcal{L}(\mathbf{w}^-, \mathcal{D}')\| \le \left(\frac{c\gamma_1}{\lambda}F + c_1\sqrt{F(n_t-1)}\right)\left(\epsilon_1 + \frac{8\gamma_1 F}{\lambda(n_t-1)} \cdot \sqrt{d(u)}\right).$$

Theorem 4.3 (Worst-case bound of edge unlearning). *Suppose that Assumption 4.1 holds, and the edge (u,v) is to be unlearned. If $\forall j \in [F]$, $\left\|\hat{\mathbf{Z}}\mathbf{e}_j - \mathbf{Z}\mathbf{e}_j\right\| \le \epsilon_1$, we can bound $\|\nabla \mathcal{L}(\mathbf{w}^-, \mathcal{D}')\|$ by*

$$\frac{4c\gamma_1 F}{\lambda n_t} + \left(\frac{c\gamma_1}{\lambda}F + c_1\sqrt{Fn_t}\right)\left(\epsilon_1 + \frac{2\gamma_1 F}{\lambda n_t}(2\epsilon_1 + \frac{4}{\sqrt{d(u)}} + \frac{4}{\sqrt{d(v)}})\right).$$

Theorem 4.4 (Worst-case bound of node unlearning). *Suppose that Assumption 4.1 holds and node u is removed. If $\forall j \in [F]$, $\left\|\hat{\mathbf{Z}}\mathbf{e}_j - \mathbf{Z}\mathbf{e}_j\right\| \le \epsilon_1$, we can bound $\|\nabla \mathcal{L}(\mathbf{w}^-, \mathcal{D}')\|$ by*

$$\frac{4c\gamma_1 F}{\lambda(n_t-1)} + \left(\frac{c\gamma_1}{\lambda}F + c_1\sqrt{F(n_t-1)}\right)\left(\epsilon_1 + \frac{2\gamma_1 F}{\lambda(n_t-1)}(2\epsilon_1 + 4\sqrt{d(u)} + \sum_{w\in\mathcal{N}(u)}\frac{4}{\sqrt{d(w)}})\right).$$

The approximation error only marginally increases the model error of unlearning, while still ensuring the total model error remains bounded.

ScaleGUN: Our results

- ScaleGUN achieves high efficiency while maintaining accuracy comparable to retraining.

Table 1: Test accuracy (%), total unlearning cost (s) and propagation cost (s) per batch edge removal for **linear** models (large graphs).

ogbn-arxiv					ogbn-products					ogbn-papers100M		
N	Retrain	CGU	CEU	ScaleGUN	$N(\times 10^3)$	Retrain	CGU	CEU	ScaleGUN	$N(\times 10^3)$	Retrain	ScaleGUN
0	57.83	57.84	57.84	57.84	0	56.24	56.23	56.23	56.23	0	59.99	59.72
25	57.83	57.83	57.83	57.84	1	56.23	56.22	56.22	56.22	2	59.71	59.61
50	57.82	57.83	57.83	57.83	2	56.22	56.21	56.21	56.21	4	59.55	59.30
75	57.82	57.82	57.82	57.82	3	56.21	56.21	56.21	56.20	6	59.89	59.16
100	57.81	57.82	57.82	57.82	4	56.20	56.20	56.20	56.19	8	59.46	59.18
125	57.81	57.81	57.81	57.81	5	56.19	56.19	56.19	56.19	10	59.26	59.14
Total	2.66	2.28	2.08	**0.91**	Total	101.90	92.37	95.79	**8.76**	Total	6764.31	**53.51**
Prop	1.73	1.68	1.68	**0.70**	Prop	98.48	91.24	94.63	**8.35**	Prop	6703.44	**6.14**

Table 2: Test accuracy (%), total unlearning cost (s) and propagation cost (s) per node feature/node removal for **linear** models on ogbn-papers100M.

Feature Unlearning			Node Unlearning		
$N(\times 10^3)$	Retrain	ScaleGUN	$N(\times 10^3)$	Retrain	ScaleGUN
0	59.99	59.72	0	59.99	59.72
2	59.99	59.72	2	59.99	59.75
4	59.99	59.65	4	59.99	59.58
6	59.99	59.47	6	59.99	59.80
8	59.99	59.54	8	59.99	59.56
10	59.99	59.45	10	60.00	59.63
Total	5400.45	**45.29**	Total	5201.88	**60.08**
Prop	5352.84	**6.89**	Prop	5139.09	**21.61**

Table 3: Test accuracy (%), total unlearning cost (s) per batch edge removal for **decoupled** models.

ogbn-products			ogbn-papers100M		
$N(\times 10^3)$	Retrain	ScaleGUN	$N(\times 10^3)$	Retrain	ScaleGUN
0	74.16	74.25	0	63.39	63.13
1	74.15	74.25	2	63.21	63.05
2	74.16	74.24	4	63.13	62.97
3	74.12	74.24	6	63.05	62.89
4	74.18	74.23	8	62.95	62.80
5	74.10	74.22	10	62.85	62.72
Total	174.23	**14.19**	Total	7958.83	**10.49**

ScaleGUN: Our results

- The effectiveness of edge unlearning
  - Add adversarial edges to the original graph.
  - Remove them in batches
  - Model accuracy improves as more adversarial edges are removed.

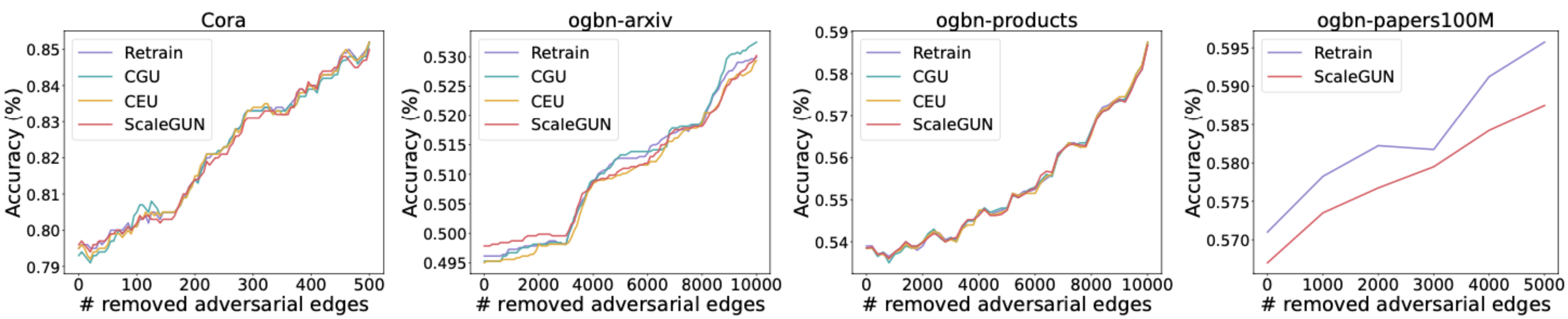


Figure 3: Comparison of unlearning efficacy for **linear** models: Model accuracy v.s. the number of removed adversarial edges.

ScaleGUN: Our results

- The effectiveness of node unlearning
  - Select a set of nodes $\mathcal{V}_d$ to be unlearned
  - Add 100-dim features to all nodes
    - all ones for $\mathcal{V}_d$, all zeros for $\mathcal{V}\setminus\mathcal{V}_d$
  - Nodes in $\mathcal{V}_d$ are assigned to a new class c
  - After unlearning $\mathcal{V}_d$, the model should no longer predict any nodes as class c.

Table 4: Deleted Data Replay Test: The ratio of incorrectly labeled nodes, r_d, r_a, after unlearning.

Model	$r_d = \frac{\lvert\{i \in \mathcal{V}_d \mid \hat{y}_i = c\}\rvert}{\lvert\mathcal{V}_d\rvert}$ (%, ↓)			$r_a = \frac{\lvert\{i \in \mathcal{V} \mid \hat{y}_i = c\}\rvert}{\lvert\mathcal{V}\rvert}$ (%, ↓)		
	Cora	ogbn-arxiv	ogbn-products	Cora	ogbn-arxiv	ogbn-products
Origin	100	90.72	100	52.13	1.01	45.59
Retrain	0	0	0	0	0	0
CGU	0	0	0	0.09	0	0
ScaleGUN	0	0	0	0.08	0	0

Thank you!

The ALGO Lab

-人工智能与大数据基础算法实验室-