Unlink to Unlearn: Simplifying Edge Unlearning in GNNs

Jiajun Tan[♠,♡], Fei Sun[♠], Ruichen Qiu[♡], Du Su[♠], Huawei Shen[♠♡]
[♠]CAS Key Laboratory of AI Safety & Security,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[♡]University of Chinese Academy of Sciences, Beijing, China
{tanjiajun22s,sunfei,sudu,shenhuawei}@ict.ac.cn

ABSTRACT

As concerns over data privacy intensify, unlearning in Graph Neural Networks (GNNs) has emerged as a prominent research frontier in academia. This concept is pivotal in enforcing the *right to be forgotten*, which entails the selective removal of specific data from trained GNNs upon user request. Our research focuses on edge unlearning, a process of particular relevance to real-world applications. Current state-of-the-art approaches like GNNDelete can eliminate the influence of specific edges, yet our research has revealed a critical limitation in these approaches, termed *over-forgetting*. It occurs when the unlearning process inadvertently removes excessive information beyond specific data, leading to a significant decline in prediction accuracy for the remaining edges. To address this issue, we have identified the loss functions of GNNDelete as the primary source of the over-forgetting phenomenon. Furthermore, our analysis also suggests that loss functions may not be essential for effective edge unlearning.

Building on these insights, we have simplified GNNDelete to develop Unlink to Unlearn (UtU), a novel method that facilitates unlearning exclusively through unlinking the forget edges from graph structure. Our extensive experiments demonstrate that UtU delivers privacy protection on par with that of a retrained model while preserving high accuracy in downstream tasks. Specifically, UtU upholds over 97.3% of the retrained model's privacy protection capabilities and 99.8% of its link prediction accuracy. Meanwhile, UtU requires only constant computational demands, underscoring its advantage as a highly lightweight and practical edge unlearning solution.

KEYWORDS

Machine Unlearning, Graph Neural Networks, Over-forgetting

ACM Reference Format:
Jiajun Tan, Fei Sun, Ruichen Qiu, Du Su, Huawei Shen. 2024. Unlink to Unlearn: Simplifying Edge Unlearning in GNNs. In *Proceedings of International World Wide Web Conference (WWW '24)*. ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

WWW '24, May 13–17, 2024, Singapore

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Although Graph Neural Networks (GNNs) have achieved significant success in various tasks [20], this advancement inherently comes with the risk of privacy leakage, as training data, rich in sensitive personal information, can be implicitly "remembered" within model parameters. In response to these privacy concerns, recent legislation [13, 16] has granted individuals with the *right to be forgotten*, enabling them to request service provider for removing their private data from online platforms. Consequently, the concept of machine unlearning [3] has emerged, allowing quick and efficient removal of specific data from a trained model, rather than retraining a new model from scratch. In addition to complying with data owners' requests for data removal, machine unlearning is also a crucial technique for rectifying models affected by poisoned, noisy, or outdated training data [18].

In this paper, we focus on edge unlearning, a key unlearning scheme in graphs, owing to its pivotal role in real-world applications such as safeguarding edge privacy in social networks. Consider the scenario where individuals in online social networks may seek to conceal certain private social connections. In these instances, GNNs that have been trained on these graphs require timely updates to eliminate any influence of the data intended to be forgotten, while preserving performance on retrained edges.

Recently, GNNDelete has achieved state-of-the-art performance in edge unlearning, however, we observed a considerable decline in its prediction accuracy for edges in the retained training set, especially for those resembling or closely associated with the edges subjected to unlearning. We introduce the term *over-forgetting* to describe such phenomenon, where an unlearning algorithm inadvertently eliminates an excessive amount of information from the retained data. In this context, "excessive" refers to a scenario where the performance of the unlearned model on retained data deteriorates significantly compared to a model retrained from scratch using only the retained data.

In this study, we address the challenge of over-forgetting by introducing **U**nlink-**t**o-**U**nlearn (**UtU**). Our investigation has revealed deficiencies in the design of GNNDelete's loss functions. The first loss function, which is designed to eliminate the influence of forget edge, unfortunately, opts for an unsuitable optimization objective, being the primary contributor to the phenomenon of over-forgetting. The other, designed to alleviate the issue of over-forgetting, fails to prevent the performance decline of retaining edges as intended. In light of these findings, we deprecate both loss functions in GNNDelete and derive UtU, a novel approach that facilitates edge unlearning exclusively by unlinking forget edges from the original graph structure. Our method eliminates the necessity for complex parameter optimization, thereby reducing computation overhead by orders of magnitude. Our experimental evaluations indicate that UtU's performance on downstream tasks, its efficacy in unlearning, and its output distribution are more aligned with those of the retrained model, which is broadly regarded as the gold standard of unlearning.

2 PRELIMINARIES

2.1 Operation of GNNs

Consider a graph $G = (V, E)$ with node set V and edge set E. Each node $v_i \in V$ is often associated with a feature vector $\boldsymbol{x_i}$. A GNN model $\mathcal{M}$, parameterized by $\boldsymbol{\theta}$, is composed of multiple GNN layers, which process node features and graph structural information to generate node embeddings via the message passing mechanism [8].

Initialized by $\boldsymbol{h}_i^0 = \boldsymbol{x_i}$ for each node v_i, the operation of l-th GNN layer can be formally expressed as follows:

$$\boldsymbol{m}_i^l = \mathsf{msg}(\boldsymbol{h}_i^{l-1}, \{\boldsymbol{h}_j^{l-1} \mid j \in \mathcal{N}(i)\}), \tag{1}$$

$$\boldsymbol{h}_i^l = \mathsf{upd}(\boldsymbol{h}_i^{l-1}, \boldsymbol{m}_i^l) \tag{2}$$

where $\mathcal{N}(i)$ denotes neighborhood of v_i, and $\mathsf{msg}(\cdot)$ and $\mathsf{upd}(\cdot)$ vary among different GNN types, representing message function and update function. The final output of $\mathcal{M}$ is the node embeddings of last layer, denoted by $\boldsymbol{h} = \mathcal{M}(G, \theta)$.

2.2 Edge Unlearning on GNNs

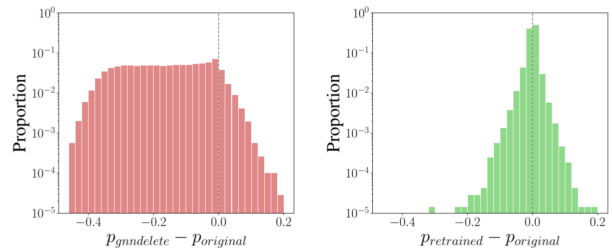
Let $\mathcal{M}_0$ be a randomly initialized GNN, and let the original model $\mathcal{M}^*$ be trained using a learning algorithm $\mathcal{A}(\mathcal{M}_0, G)$ on graph G. The forget set, defined as $E_d \subseteq E$, contains the edges requested for removal, while the retain set $E_r = E \backslash E_d$ represents the remaining edges in the training graph G. The objective of unlearning is to devise an unlearning process $\mathcal{U}$ that renders the unlearned model $\mathcal{M}_u = \mathcal{U}(\mathcal{M}^*, G, E_d)$ indistinguishable from the retrained model $\mathcal{M}_r = \mathcal{A}(\mathcal{M}_0, G_r)$, with $G_r = (V, E_r)$ being the retain graph.

A variety of unlearning algorithms have been developed to address edge unlearning requests in GNNs. [4, 17] leverage SISA paradigm [2], where only the sub-model corresponding to the removed data needs to be retrained and then aggregates its result with other sub-models. Other strategies directly update the original model, including leveraging the influence function [18], providing a theoretical guarantee of unlearning via differential privacy [6, 19], projecting parameters to irrelevant subspace [7].

2.3 GNNDelete

Recently, GNNDelete [5] surpasses various baseline methods, showing a strong capability to unlearn selected edges by a learning-to-unlearn framework. It first inserts a linear transformation ϕ with learnable parameters after each GNN layer, while freezing the original model's parameters. ϕ is only applied to nodes in l-hop enclosing subgraph of forget edge e_{uv}, namely S_{uv}^l , by transforming each node's original embedding into unlearned embedding: $\boldsymbol{h}_i'^l = \phi^l(\boldsymbol{h}_i^l)$; for other nodes, their embeddings remain unchanged.

Two loss functions, *Deleted Edge Consistency* (DEC) loss $\mathcal{L}_{DEC}$ and *Neighborhood Influence* (NI) loss $\mathcal{L}_{NI}$, are then computed layer-wise. Generally, DEC loss is intended for unlearning edges in E_d, while the NI loss aims to repair node embeddings in S_{uv}^l. We will discuss their design in more detail in 3.2. During the backward pass, the parameters of ϕ are optimized based on the weighted total loss,



(a) Unlearn by GNNDelete. **(b) Retrain from scratch.**

Figure 1: An intuitive demonstration of over-forgetting, comparing the difference of retaining edge predictions before and after forgetting. The x-axis represents the change in predicted probabilities after unlearning, and the y-axis shows the distribution of these changes across the retain set E_r.

which is represented as:

$$\mathcal{L}^l = \lambda \mathcal{L}_{DEC}^l + (1 - \lambda)\mathcal{L}_{NI}^l, \tag{3}$$

GNNDelete uses $\lambda = 0.5$ to report its performance, claiming this setting can achieve the best overall performance.

3 UTU: UNLINK TO UNLEARN

While GNNDelete reports prominent unlearning ability, it suffers from the issue of over-forgetting. In this section, we first provide a formal introduction to the over-forgetting problem, subsequently revealing the connection between DEC loss and over-forgetting. By removing inappropriate loss functions in GNNDelete, we further introduce our simplified approach, **UtU**, aiming at alleviating the issue of over-forgetting.

3.1 Over-forgetting in Edge Unlearning

As previously introduced, over-forgetting refers to the phenomenon where the performance of samples in the retained set significantly deteriorates after unlearning. Figure 1 shows the over-forgetting observed on GNNDelete. Despite only 5% of the edges being unlearned, a substantial 92.4% of the retained edges experience performance decrement. Conversely, the retrained model's predictions for the majority of retained edges remain mostly unchanged. In order to quantitatively assess over-forgetting, we compare performance of unlearned model against a retrained model to gauge the impact of the forgetting procedure.

For link prediction task, the probability of the existence of the edge between nodes (v_i, v_j) is predicted by integrating their final embeddings $\boldsymbol{h}_i^L, \boldsymbol{h}_j^L$ using a score function $\varphi(\cdot)$, as $p_{ij} = \varphi(\boldsymbol{h}_i^L, \boldsymbol{h}_j^L)$. For $e_{ij} \in E_r$, we identify over-forgetting if the predicted probability of e_{ij} in $\mathcal{M}_u$ decreases compared to $\mathcal{M}_r$, i.e., $\Delta p_{ij} = \varphi(\boldsymbol{h}_i', \boldsymbol{h}_j') - \varphi(\boldsymbol{h}_i^r, \boldsymbol{h}_j^r) < 0$, where $\boldsymbol{h}'$ and $\boldsymbol{h}^r$ represent embeddings generated by models $\mathcal{M}_u$ and $\mathcal{M}_r$, respectively. Typically, our focus is on the overall performance decline across E_r, which can be measured as $\Delta p_r = \mathsf{mean}(\Delta p_{ij}), \forall e_{ij} \in E_r$.

3.2 Analysis on Unlearning Target

Now we discuss the design of GNNDelete's loss functions to find out the source of over-forgetting. The Deleted Edge Consistency (DEC) loss minimizes the difference between predictions of forget edges e_{uv} and random-chosen node pairs:

$$\mathcal{L}^l_{\mathrm{DEC}} = \mathrm{mse}(\{[\boldsymbol{h}'^l_u;\boldsymbol{h}'^l_v] \mid e_{uv} \in E_d\}, \{[\boldsymbol{h}^l_p;\boldsymbol{h}^l_q] \mid p,q \in_R V\}), \quad (4)$$

where $[\cdot;\cdot]$ denotes the concatenation of two vectors. mse refers to Mean-Squared Error.

However, the use of random node pairs $p, q \in_R V$ as optimization targets for the forget set E_d is not suitable. The optimization of DEC loss will introduce structural noise by encouraging embeddings of nodes in E_d to reflect random connections rather than actual graph topology, leading to inaccurate representations and prediction results. Moreover, this noise propagates to neighboring nodes by message passing mechanism as pointed out in Sec. 2.1, degrading the embedding quality on a broader scale and exacerbating the issue of over-forgetting. Besides, connected nodes in a graph tend to share similar attributes or belong to the same class according to the homophily hypothesis [11]. Despite being removed due to unlearning requests, the samples in E_d originate from pre-existing edges in the graph, implying that their end nodes ought to exhibit strong homophily, and should not be equated with arbitrarily selected node pairs.

Neighborhood Influence (NI) loss base on the idea that removing e_{uv} should not affect the predictions of its enclosing subgraph S_{uv}:

$$\mathcal{L}^l_{\mathrm{NI}} = \mathrm{mse}(\|_w\{[\boldsymbol{h}'^l_w] \mid w \in S^l_{uv}/e^l_{uv}\}, \|_w\{[\boldsymbol{h}^l_w] \mid w \in S^l_{uv}\}), \quad (5)$$

where $\|$ signifies the concatenation of multiple vectors.

NI loss guides the adaptation of the unlearned embedding $\boldsymbol{h}'^l_w$ to the original embedding, acting as a regularization term for node embeddings in S^l_{uv} to mitigate the structural noise induced by DEC loss.

However, note that node features and edges are combined as GNN's input, the original embedding is generated using original graph without removing e_{uv}'s influence. That means the unlearned embedding will still contain structural information of edges in the forget set, which weakens its ability to repair over-forgetting.

3.3 UtU: a Minimalist Approach

Given the preceding analysis, we propose to eliminate the DEC loss due to its selection of an unsuitable target for forgetting. Following the removal of DEC loss, the NI loss, initially serving as a corrective measure for DEC loss, is also deemed unnecessary. We then propose that edge unlearning can be effectively achieved by only altering the input edge indexes to that of retain graph G_r, which can be named **Unlink to Unlearn**:

$$\boldsymbol{h}' = \mathcal{M}^*(G_r;\theta^*), \quad (6)$$

where θ^* denotes parameters of original model $\mathcal{M}^*$.

This approach is grounded on the insights of GNN operations. The edges in the graph mainly facilitate message passing between node features, as delineated in Eq. 1. Most GNN models' parameters are located during the node features' update step, as specified in Eq. 2, which does not involve edge utilization. Therefore, removing a forgotten edge during inference can effectively block message propagation from neighboring nodes linked to the forget set. This action alone suffices to eliminate the edge's influence from the model, thereby achieving our unlearning objective.

It is worth noting that UtU is remarkably efficient through its minimalistic design. Deleting an edge from the graph structure only requires $O(1)$ time complexity, making UtU a nearly "zero-cost" edge unlearning solution.

Table 1: Statistical Overview of the Datasets.

Dataset	# Nodes	# Edges	# Features	# Classes
CoraFull	19,793	126,842	8,710	70
PubMed	19,717	88,648	500	3
CS	18,333	163,778	6,805	15
OGB-collab	235,868	2,238,104	128	N/A

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. We conduct the experiments on four real-world datasets, including citation networks: CoraFull [1] and PubMed [1], and collaboration networks: CS [14] and OGB-collab [9]. Table 1 shows details of these datasets. In accordance with previous work, we split 90% edges for the training set, 5% for validation, and 5% for test.

Baselines. For GNN backbones, we choose the most widely-used 2-layer GNNs: GCN [10], GAT [15], and GIN [21]. For unlearning methods, we consider the following baseline methods: Retrain from scratch, Gradient Ascent, GIF [18], and GNNDelete [5]. We also set a variant of GNNDelete by only removing DEC loss, namely GNNDelete-NI, for comparison.

Tasks. Following the common practice in [5, 19], we train all models on link prediction task and then perform edge unlearning. Forget edges are randomly chosen from the training set. We vary the proportion of forget edges from 0.1% to 5% to examine algorithm performance under different scenarios. There are few scenarios where more than 5% of edges need to be unlearned simultaneously.

Metrics. We adopt ROC-AUC to evaluate downstream tasks for link prediction. To assess the effectiveness of unlearning, we compare the unlearned model against the retrained model using JS divergence and ROC-AUC of MI Attack. Additionally, we use Δp, introduced in Sec. 3.1, to compare over-forgetting.

Implementation. We follow the default hyper-parameter setting of all baselines, and metrics are reported across an average of five independent runs[1].

4.2 Experiment Result and Anlysis

4.2.1 Downstream Task. In this part, we compare the utility of unlearned models obtained by different unlearning methods, as we anticipate that unlearning will not harm the performance of GNN on downstream tasks. ROC-AUC was used to determine the model's ability to predict hidden test edges. Results are shown in Table 2, where UtU performs the best in most settings, with the closest gap of 0.001 on average compared to retraining.

4.2.2 Unlearning Efficacy. The model after unlearning should treat the forget edges as if it had never seen them before. Hence, we expect the predictions on these edges to be similar to those of a model that has been retrained from scratch.

Following [20], we use the activation distance measured by JS divergence, along with membership inference (MI) attack [12] to assess whether the model has truly achieved the effect of unlearning. Figure 2 illustrates that the outputs of forget edges from UtU closely mirror those from a model retrained from scratch. Table 3 shows that the resistance to MI Attack of UtU is generally more aligned with retraining than baseline methods.

[1]The code will be released after acceptance.

Table 2: AUC (↑) on Link Prediction. Forget Set: 5.0% edges.

Dataset	Model	Retrain	GradAscent	GIF	GNNDelete	GNNDelete-NI	UtU
CoraFull	GCN	0.967	0.563	0.964	0.922	**0.967**	0.965
	GAT	0.963	0.766	0.926	0.934	0.947	**0.964**
	GIN	0.961	0.596	0.742	0.897	0.958	**0.960**
PubMed	GCN	0.970	0.375	0.924	0.934	0.968	**0.969**
	GAT	0.936	0.766	0.774	0.890	0.927	**0.933**
	GIN	0.939	0.545	0.842	0.887	0.938	**0.942**
CS	GCN	0.968	0.786	0.950	0.947	0.968	**0.970**
	GAT	0.963	0.846	0.941	0.943	0.958	**0.963**
	GIN	0.960	0.583	0.520	0.900	0.959	**0.960**
OGB-collab	GCN	0.985	0.406	0.971	0.925	0.981	**0.987**
	GAT	0.971	0.755	0.744	0.924	0.960	**0.971**
	GIN	0.925	0.683	0.500	0.805	0.912	**0.913**

Table 3: AUC on MI Attack. Forget Set: 5.0% edges.

Dataset	Model	Retrain	GradAscent	GIF	GNNDelete	GNNDelete-NI	UtU
CoraFull	GCN	0.580	0.500	0.529	0.712	0.531	0.528
	GAT	0.582	0.500	0.547	0.719	0.543	0.541
	GIN	0.586	0.510	0.508	0.721	0.586	0.605
PubMed	GCN	0.616	0.500	0.555	0.652	0.555	0.551
	GAT	0.624	0.500	0.574	0.708	0.576	0.578
	GIN	0.603	0.519	0.552	0.752	0.662	0.625
CS	GCN	0.593	0.500	0.580	0.566	0.573	0.577
	GAT	0.574	0.500	0.543	0.615	0.540	0.542
	GIN	0.580	0.653	0.497	0.666	0.592	0.590
OGB-collab	GCN	0.515	0.500	0.538	0.542	0.528	0.547
	GAT	0.560	0.500	0.533	0.484	0.478	0.554
	GIN	0.571	0.502	0.500	0.511	0.555	0.556
Avg Diff. with Retrain (%)			6.67	4.40	5.53	2.21	**1.58**

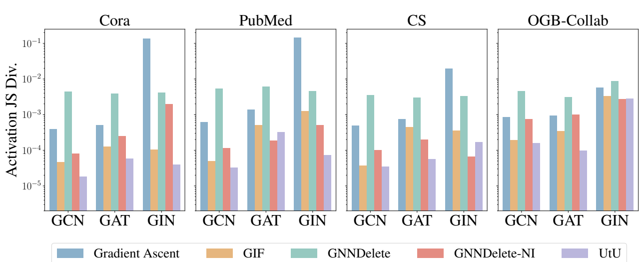


Figure 2: Activation distance (↓) on forget set (5.0% edges).

4.2.3 Over-forgetting evaluation. Figure 3 shows the trend of Δp as the size of the forget set changes. As described in 3.1, Δp represents the average difference of the edge predictions of the retain set, compared with that of retrained. Lower Δp indicates more serious over-forgetting. The results indicate that our method remains unaffected by over-forgetting, regardless of the forget set's size. Furthermore, its predictions of retain set are also highly consistent with those of the retrained model.

5 CONCLUSION

In this work, we address the issue of over-forgetting in the state-of-the-art edge unlearning method, GNNDelete. Our analysis identifies a correlation between its loss functions and the over-forgetting problem. To mitigate this, we introduce a simplified approach named Unlink to Unlearn (UtU). UtU effectively eliminates the influence of forgotten edges by merely modifying the structure of the input graph, thus obstructing the corresponding message-passing paths in GNN during the inference stage. Experimental results demonstrate that UtU acts on par with the retrained model with near-zero computational overhead. Our findings suggest that removing a

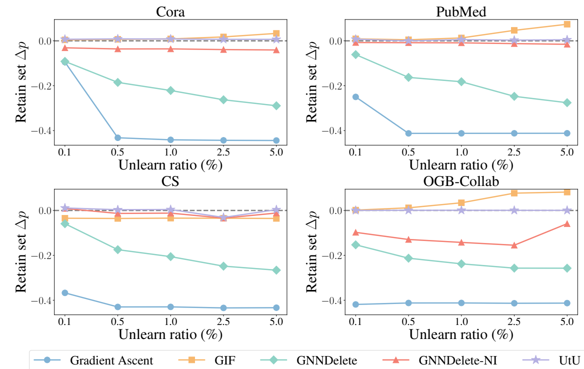


Figure 3: Comparison of over-forgetting on GAT backbone.

small number of edges might have little influence on the model parameters, highlighting an avenue for future research to investigate this phenomenon.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine Unlearning. In *IEEE S&P*. 141–159.

[3] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *IEEE S&P*. 463–480.

[4] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *SIGSAC*. 499–513.

[5] Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. 2022. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *ICLR*.

[6] Eli Chien, Chao Pan, and Olgica Milenkovic. 2023. Efficient Model Updates for Approximate Unlearning of Graph-Structured Data. In *ICLR*.

[7] Weilin Cong and Mehrdad Mahdavi. 2023. Efficiently Forgetting What You Have Learned in Graph Representation Learning via Projection. In *ICAIS*. 6674–6703.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.

[9] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NIPS*, Vol. 33. 22118–22133.

[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[11] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[12] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership inference attack on graph neural networks. In *TPS-ISA*. IEEE, 11–20.

[13] Stuart L Pardau. 2018. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y* 23 (2018), 68.

[14] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[16] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[17] Cheng-Long Wang, Mengdi Huai, and Di Wang. 2023. Inductive Graph Unlearning. *arXiv preprint arXiv:2304.03093* (2023).

[18] Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. 2023. GIF: A General Graph Unlearning Strategy via Influence Function. In *TheWebConf*. 651–661.

[19] Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. 2023. Certified edge unlearning for graph neural networks. In *SIGKDD*. 2606–2617.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE TNNLS* 32, 1 (2021), 4–24.

[21] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.