GNNDELETE: A GENERAL UNLEARNING STRATEGY FOR GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of graph unlearning, wherein graph neural network (GNN) model is trained to specified accuracy and then deployed while a sequence of requests arrives to delete graph elements (nodes, edges) from the model. As GNN models are used in real-world implementation, this problem is increasingly vital to address—for example, when a user seeks to hide their connections with others in a social graph or when relationships in a knowledge graph become irrelevant or are not true anymore. To unlearn information from a trained GNN, its influence on both GNN model weights as well as on representations of neighbors in the graph must be deleted from the model. However, existing methods using retraining and weight modification either degrade model weights shared across all nodes or are ineffective because of strong dependency of deleted edges on their local graph neighborhood. Realizing these pitfalls, we formalize the required properties for graph unlearning in the form of Deleted Edge Consistency and Neighborhood Influence and develop GNNDELETE, a model-agnostic layer-wise operator that optimize both properties for unlearning tasks. GNNDELETE updates latent representations to delete nodes and edges from the model while keeping the rest of the learned knowledge intact. Experiments on six real-world and two knowledge graphs show that GNNDELETE outperforms existing graph unlearning models by up to 36.9% in AUC on link prediction tasks and 22.5% in AUC on distinguishing deleted edges from non-deleted edges. GNNDELETE efficient—e.g., it takes 12.3x less time and 9.3x less space than retraining from scratch on WordNet18. Our code is available here.

1 INTRODUCTION

Graph neural networks (GNN) are increasingly used in real-world implementations (Xu et al., 2022; 2019; Huang et al., 2021; Morselli Gysi et al., 2021), and the underlying graphs evolve over time in most deployed GNNs. Traditional machine learning approaches often work offline, wherein a model is trained once using the full training dataset and then locked for inference with few, if any, updates to the model. In contrast, online training can update the model using new training data points as they become available (Orabona, 2019; Nagabandi et al., 2018). However, neither offline nor online learning can deal with data deletion (Cao & Yang, 2015b; Ginart et al., 2019a)—the task of removing all traces of a data point from a model without sacrificing model performance. When data needs to be deleted from a model, the model must be updated accordingly. For example, GNNs must implement privacy provisions for preserving personal privacy (e.g., California Consumer Privacy Act (CCPA) and General Data Protection Regulation (GDPR)), meaning that endowing GNNs with data deletion capability is important yet sparsely studied in the literature (Ginart et al., 2019b; Fu et al., 2022).

At the same time, GNNs received significant interest from the learning community due to the ubiquity of graph-structured data (Hu et al., 2020; Ying et al., 2019). However, despite methods developed for general machine unlearning, none of them can be readily applied to GNNs because graph machine unlearning (as we show in this paper) is fundamentally different because of dependencies between nodes connected by edges. In addition, since the current success of GNN models relies on the idea of "exchanging information in the local neighborhood," an adversarial agent can trivially infer the presence of the data from its neighbors if the impact of data on its local neighborhood is not limited. Given the broad range of GNN applications and the lack of general graph unlearning methods, there is a need to design algorithms that enable GNN models to unlearn what they have previously learned. This would mean that unavailable, outdated, or privacy-concerned graph elements are not used by the

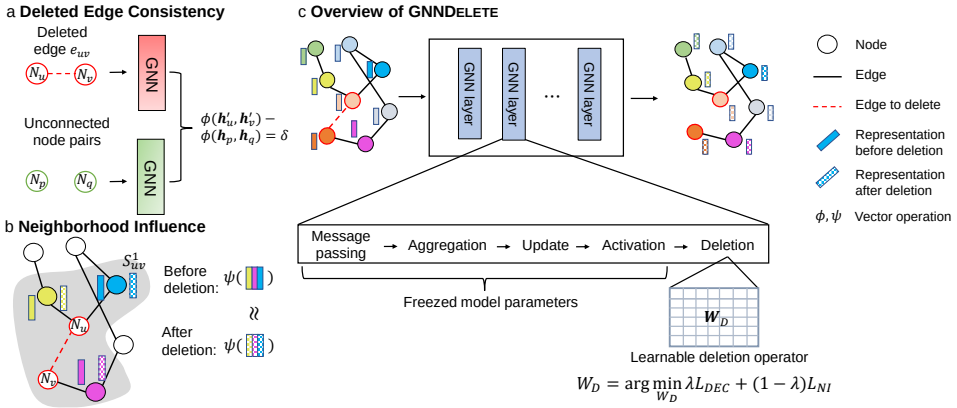

Figure 1: a. Illustration of Deleted Edge Consistency: It suggests that the predicted probability of deleted edges after unlearning should be random, such that it looks like the deleted data was not used for training before. **b**. Illustration of Neighborhood Influence: It implies that an appropriate unlearning should not change the representations of the local neighborhood (nodes in the subgraph, not nodes themselves) to maintain the original causality. **c**. Overview of GNNDELETE: Given a trained GNN model and edge deletion request, GNNDELETE outputs unlearned representations efficiently by only learning a small deletion operator W_D. It also ensures representation quality by minimizing a loss function that satisfies the two key properties proposed above.

model anymore, preventing security concerns and performance drops. In this paper, we take a step towards building an efficient and accurate machine unlearning method for GNNs.

Designing graph unlearning methods is nonetheless challenging. Removing data alone is insufficient to comply with recent demands for increased data privacy because models trained on the original data may still contain information about their patterns and features. A naive approach is to delete the data and retrain a model from scratch. However, this can be prohibitively expensive, especially in large datasets. Recently, many efforts have been made to achieve efficient unlearning based on the idea of exact learning (Brophy & Lowd, 2021; Sekhari et al., 2021; Hase et al., 2021; Ullah et al., 2021). The core idea is to retrain several independent models by dividing datasets into separate shards and then aggregating their predictions during inference. Such methods guarantee the removal of all information associated with the deleted data. However, in the context of GNN, dividing graphs destroys the structure of the input graph and thus leads to poor performance on edge-level tasks, such as link prediction. To address this issue, Chen et al. (2021c) applies the exact learning to the graph-structured data by proposing a particular graph partition method that can preserve the structural information and weighted prediction aggregation for inference. However, this is still less efficient as the cost increase as the number of shards grows. In addition, choosing the optimal number of shards is still unresolved and may require extra hyperparameter tuning. Several approximation-based approaches Guo et al. (2019); Ullah et al. (2021); He et al. (2021); Shibata et al. (2021) avoid retraining a model from scratch on multiple subsets. While these approaches have shown promise, Mitchell et al. (2022) show that the methods produce model updates that can harm model performance.

Present Work. We introduce GNNDELETE[1], a model-agnostic approach for graph unlearning. We formalize two key GNN deletion properties (Section 4.1): 1) *Deleted Edge Consistency:* Predicted probability for deleted edges of the unlearned model should be similar to those for nonexistent edges. The property enforces GNNDELETE to unlearn information such that deleted edges masquerade as unconnected nodes. 2) *Neighborhood Influence:* We establish a connection between graph unlearning and Granger causality (Granger, 1969) to ensure that local subgraphs after deletion are not affected and thus maintain the original predictive dependencies. However, existing graph unlearning methods do not consider this essential property, meaning they do not consider local connectivity influence, leading to sub-optimal deletion. Using both properties, we introduce GNNDELETE, a layer-wise deletion operator to update node representations. When receiving deletion requests, GNNDELETE freezes the model and learns additional small gated weight matrices that are shared across all nodes. Unlike existing methods that attempt to retrain several small models from scratch or directly update model weights which can be inefficient and suboptimal, GNNDELETE uses small learnable matrices for inference without changing GNN model weights, achieving both efficiency and scalability. To

[1]Code: `https://anonymous.4open.science/r/Anonymous-repo-GNNDelete-D7F5/`.

optimize GNNDELETE, we specify a novel objective function (described in Section 4.3) that satisfies Deleted Edge Consistency and Neighborhood Influence, yielding strong overall deletion.

Our Contributions. We summarize our contributions as follows: ① We formalize the machine unlearning problem and suggest two helpful properties, *Deleted Edge Consistency* and *Neighborhood Influence*, of machine unlearning in the context of graphs. ② We propose a GNN data deletion approach GNNDELETE. It unlearns the deleted data much more efficiently than existing graph unlearning methods without sacrificing predictive power. ③ We show that GNNDELETE is of particular interest as it is agnostic to the class of models and training methodology and so is extremely flexible. ④ Theoretically, we show the difference between node representations from the baseline and GNNDELETE is bounded, which ensures GNNDELETE's strong performance. ⑤ We provide empirical results for both link and node deletion tasks. The results suggest that GNNDELETE can lead to effective unlearning using $12.3\times$ less time and $9.3\times$ computation than retraining from scratch.

2 RELATED WORK

Machine Unlearning. We organize existing machine unlearning literature into four categories: 1) *Re-training:* Re-training models from scratch to unlearn is simple yet can be inefficient despite efforts to develop efficient data partitioning and re-training (Bourtoule et al., 2021; Wu et al., 2020; Liu et al., 2022b; Cao & Yang, 2015a; Golatkar et al., 2020; Izzo et al., 2021). Nevertheless, partitioning graphs is not easy because the graph structure and learned node representations of the shared graphs can be significantly different from the original graph. Moreover, such methods may not scale to large datasets. *2) Output modification:* To reduce computational overhead, methods, such as UNSIR (Tarun et al., 2021) directly modify model outputs. UNSIR first learns an error matrix that destroys the output and then trains the destroyed model for two epochs with clean data to impair and repair the outputs. On graphs, the error destroys outputs for all edges, and the training after that falls back to re-training the whole model *3) Logit manipulation:* Other methods achieve unlearning by manipulating the model logits (Izzo et al., 2021; Baumhauer et al., 2020). These methods only apply to linear or logit-based models. *4) Weight modification:* Unlearning via weight modification is achieved by running an optimization algorithm. For example, Ullah et al. (2021) proposed a new algorithmic machine unlearning based on noisy stochastic gradient descent. Guo et al. (2019) achieves certified removal based on Newton update. Other weight modification via optimization methods include Thudi et al. (2021a) and Neel et al. (2021). Recent unlearning methods perturb gradients (Liu et al., 2020b) or model weights (Chen et al., 2021a). However, weight modification approaches lack features unique to graphs and incur computation overheads (e.g., calculation of the inverse Hessian). In Appendix A, we elaborate further on machine unlearning methods for other tasks and models.

Graph Unlearning. GraphEraser (Chen et al., 2021c) tries to solve the graph unlearning problem with graph partitioning and efficient retraining. They extend the k-means algorithm to consider both node features and structural information when dividing graphs into shards. A learnable aggregator is optimized to aggregate the predictions from each sharded model. However, GraphEraser (Chen et al., 2021c) is designed for the node classification task, which hinders its applications on other tasks such as link prediction. GraphEditor (Cong & Mahdavi) relies on the closed-form solution of linear GNNs for guaranteed information deletion, including node deletion, edge deletion, and node feature update. Performing additional finetuning can boost predictive performance. However, GraphEditor is restricted to linear structure, which is the same case for most machine unlearning algorithms (not only limited to the graph structured data), Thus, it cannot work with existing trained non-linear GNNs or knowledge graphs. It also has trouble unlearning a large portion of nodes or edges.

Connection with Catastrophic Forgetting and Dynamic Network Embeddings. Despite the fact of forgetting knowledge, we argue that catastrophic forgetting (Kemker et al., 2018) is not a good fit for the machine unlearning setting. Specifically, catastrophic forgetting is 1) not targeted at a particular set of data, 2) not guaranteed to forget, i.e. adversarial agents may still find out the presence of what is forgotten in the training set; 3) not capable of handling an arbitrary amount of deletion requests. Dynamic network embedding (Nguyen et al., 2018) learns network representations by incorporating time information and can deal with data deletion. However, these methods are designed to process graphs that are inherently changing over time. On the other hand, machine unlearning aims at deleting a specific set of training data, while keeping the majority of the underlying graph fixed. It aims at targeted data removal requests and is thus orthogonal to the above two approaches.

Connection with Adversarial Attacks and Defense on GNNs. To determine whether a given data point has been used in the training process of a model, Membership Inference (MI) is a suitable measure for the quality of unlearning (Yeom et al., 2017; Sablayrolles et al., 2019). Defense against MI attack is also a key problem we care about when building unlearning models. Thudi et al. (2022) proposed to use a novel privacy amplification scheme based on a new tighter bound and subsampling strategy. Olatunji et al. (2021) showed that all GNN models are vulnerable to MI attack and proposed two defense mechanisms based on output perturbation and query neighborhood perturbation. Liu et al. (2022a) treat the data to be unlearned as backdoored data. Although such works can serve as an interesting unlearning measure, they aim at defending against MI attacks and can not perform the real unlearning procedure, which is a complementary topic to machine unlearning.

Connection with Techniques for Achieving Privacy in ML Models. Privacy-preserving is another related privacy-preserving learning scheme, such as federated learning (FL) (Konečný et al., 2016). To let FL models unlearn data in a similar privacy-preserving way, Liu et al. (2020a) proposed the first federated unlearning framework by leveraging historical parameter updates. Chen et al. (2021d) proposed a new attack and metrics to improve privacy protection, which provides insights on practical implementations of machine unlearning. Golatkar et al. (2021) proposed machine unlearning in a mixed-privacy setting by splitting the weights into a set of core and forgettable user weights. Different from privacy-preserving algorithms that aim at protecting data privacy during training and inference, the goal of machine unlearning, most of the time, is to retrieve privacy, which has no conflict.

3 PRELIMINARIES

Let $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ denote an attributed graph with $n = |\mathcal{V}|$ nodes, set of edges $\mathcal{E}$, and n_f-dimensional node features $\boldsymbol{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}\}$ where $\boldsymbol{x}_i \in \mathbb{R}^{n_f}$. We use $\boldsymbol{A}$ to denote the adjacency matrix of G and $\deg_G : \mathcal{V} \to \mathbb{N}$ to denote degree distribution of graph G. Further, we use $\mathcal{S}^k_{uv} = (\mathcal{V}^k_{uv}, \mathcal{E}^k_{uv}, \boldsymbol{X}^k_{uv})$ to represent a k-hop *enclosing subgraph* around nodes u and v.

Graph Neural Networks (GNNs). A GNN layer g can be expressed as a series of transformation functions: $g(G) = (\textsc{Upd}\circ\textsc{Agg}\circ\textsc{Msg})(G)$ that takes G as input and produces n d-dimensional node representations $\boldsymbol{h}_u$ for $u \in \mathcal{V}$ (Figure 1). Within layer l, MSG specifies neural messages that are exchanged between nodes u and v following edges in $\boldsymbol{A}_{uv}$ by calculating $\boldsymbol{p}^l_{uv} = \textsc{Msg}(\boldsymbol{h}^{l-1}_u, \boldsymbol{h}^{l-1}_v, \boldsymbol{A}_{uv})$. The AGG defines how every node u combines neural messages from its neighbors $\mathcal{N}_u$ and how it computes the aggregated message as $\boldsymbol{P}^l_u = \textsc{Agg}(\{\boldsymbol{p}^l_{uv} | v \in \mathcal{N}_u\})$. Finally, UPD defines how the aggregated messages $\boldsymbol{P}^l_u$ and hidden node states from the previous layer are combined to produce $\boldsymbol{h}^l_u$, i.e., final outputs of l-th layer as follows $\boldsymbol{h}^l_u = \textsc{Upd}(\boldsymbol{P}^l_u, \boldsymbol{h}^{l-1}_u)$. The output of the last GNN layer are final node representations, $\boldsymbol{z}_u = \boldsymbol{h}^L_u$, where L is the total number of GNN layers in the model.

Unlearning in GNN models. Let $\mathcal{E}_d \subseteq \mathcal{E}$ denote edges to be deleted and $\mathcal{E}_r = \mathcal{E} \backslash \mathcal{E}_d$ be remaining edges after removing $\mathcal{E}_d$ from the graph. We use $G_r = (\mathcal{V}_r, \mathcal{E}_r, \boldsymbol{X}_r)$ to indicate the graph after deleting edge $\mathcal{E}_d$, where $\mathcal{V}_r = \{u \in V | \deg_{G_r}(u) > 0\}$ are nodes that remain connected to nodes in G_r, and $\boldsymbol{X}_r$ denotes the corresponding node attributes. While the above mentioned description focuses on edge deletion, GNNDELETE can be used for node editing as well by deleting all edges incident to a node the user wants to delete from the model. Unlearning an edge e_{uv} requires the model to delete all the information and influence associated with e_{uv} as if it has never seen it during the training, but keep the change in downstream performance to a minimum level. Under the current GNN framework, we need to not only modify the predicted probability of e_{uv} but also need to remove its information from its local neighborhood. Hence, post-processing or logit manipulation is not effective for a GNN to delete edges, even if such strategies do not affect the rest of the graph at all. We also denote a classification layer f which takes as input pairs of node representations $\boldsymbol{h}^L_u$ and $\boldsymbol{h}^L_v$ and outputs prediction probabilities for the edges e_{uv}. Finally, given L GNN layers and f, a GNN model $m : G \to \mathbb{R}^{|\mathcal{E}|}$ can be written as: $m(G) = (f \circ g^L \cdots \circ g^1)(G)$. Similarly, the corresponding unlearned model $m' : G \to \mathbb{R}^{|\mathcal{E}_r|}$ can be written as: $m'(G_r) = (f \circ g'^L \cdots \circ g'^1)(G_r)$ as a composition of L unlearned GNN layers.

4 OUR APPROACH: GNNDELETE

Effectively deleting edges from graphs requires that the GNN model ignores the importance of the existence of the edges in set $\mathcal{E}_d$. Given the graph G, the model should be unable to recognize

whether a deleted edge e_{uv} is part of the graph and ignore any influence that a deleted edge has in its neighborhood. To this end, we i) present two properties for effective graph unlearning and ii) propose a layer-wise deletion operator that can be applied to any GNN for deleting a given set of edges $\mathcal{E}_d$.

Problem Formulation (Graph Unlearning). *Given is a graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, a fully trained GNN model $m(G)$, and a set of edges $\mathcal{E}_d$ to delete from $m(G)$. Then, graph unlearning aims to unlearn every edge $e_{uv} \in \mathcal{E}_d$ from the GNN model $m(G)$ such that the unlearned model $m'(G)$ is close to the model that would have been output had deleted edges $\mathcal{E}_d$ been omitted from training by satisfying Deleted Edge Consistency property for $e_{uv} \in \mathcal{E}_d$ and Neighborhood Influence property for $e_{uv} \notin \mathcal{E}_d$.*

4.1 Deletion Properties for Graph Neural Networks

Deleting information from a graph is not a trivial task because the representations of nodes and edges are dependent on the combined neighborhood representations. The following two properties show intuitive assumptions over the deletion operator for effective unlearning in GNNs:

1) Deleted Edge Consistency. The predicted probability from the unlearned model m' for an edge e_{uv} should be *random*, such that it's hard to determine whether it is a true edge or not. More specifically, the unlearned GNN layer g'^l should effectively forget the edge information, and consequently, the combination of the node representations that are present in the deleted edge should be *random*. Formally, we have the following property definition:

Definition 1 (Deleted Edge Consistency)**.** *Let e_{uv} denote an edge to be deleted, g^l be the l-th layer in a GNN with output node representation vectors $\boldsymbol{h}_u^l$, and the unlearned GNN layer g'^l with $\boldsymbol{h}_u'^l$. The unlearned layer g'^l satisfies the Deleted Edge Consistency property if it minimizes the difference between node-pair representations $\phi(\boldsymbol{h}_u', \boldsymbol{h}_v')$, and $\phi(\boldsymbol{h}_p, \boldsymbol{h}_q)$ of two randomly chosen nodes $p, q \in \mathcal{V}$:*

$$\mathop{\mathbb{E}}_{p,q \in_R \mathcal{V}} [\phi(\boldsymbol{h}_u'^l, \boldsymbol{h}_v'^l) - \phi(\boldsymbol{h}_p, \boldsymbol{h}_q)] = \delta, \tag{1}$$

where ϕ is a readout function (e.g., the dot product, concatenation, etc.) that combines the node representations $\boldsymbol{h}_u'^l$ and $\boldsymbol{h}_v'^l$, $\in_R$ denotes the random choice from $\mathcal{V}$, and δ is an infinitesimal constant.

2) Neighborhood Influence. While the notion of causality has been used in explainable machine learning, to the best of our knowledge, we propose the first effort of modifying a knowledge graph using a causal perspective. Formally, removing edge e_{uv} from the graph requires unlearning the influence of e_{uv} from the subgraphs of both nodes u and v. In this work, we propose the *Neighborhood Influence* property which leverages the notion of Granger causality (Granger, 1969; Bressler & Seth, 2011) and declares a causal relationship $\psi(\{\boldsymbol{h}_u | u \in \mathcal{S}_{uv}\}) \rightarrow e_{uv}$ between variables $\psi(\{\boldsymbol{h}_u | u \in \mathcal{S}_{uv}\})$ and e_{uv} if we are better able to predict edge e_{uv} using all available node representations in $\mathcal{S}_{uv}$ than if the information apart from $\psi(\{\boldsymbol{h}_u | u \in \mathcal{S}_{uv}\})$ had been used. Here, $\psi(\cdot)$ is an operator that combines the node representations in subgraph $\mathcal{S}_{uv}$. In the context of graph unlearning, if the absence of node representations decreases the prediction confidence of an edge, then there is a causal relationship between the node representation and the e_{uv} prediction.

Here, we characterize the notion of deletion by extending Granger causality to local subgraph causality, i.e., an edge e_{uv} dependent on the subgraphs associated with both nodes u and v. In particular, removing e_{uv} should not affect the predictions of $\mathcal{S}_{uv}$ yielding the following property:

Definition 2 (Neighborhood Influence)**.** *Let $e_{uv} \in \mathcal{E}_d$ denote an edge in G to be deleted, g^l be the l-th layer in a GNN with output node representation vectors $\boldsymbol{h}_u^l$, and the unlearned GNN layer g'^l with $\boldsymbol{h}_u'^l$. The unlearned layer g'^l satisfies the Neighborhood Influence property if it minimizes the difference of all node-subset representations $\psi(\{\boldsymbol{h}_w^l | w \in \mathcal{S}_{uv}\})$ comprising e_{uv} with their corresponding node-subset representations $\psi(\{\boldsymbol{h}_w'^l | w \in \mathcal{S}_{uv/e_{uv}}\})$ where e_{uv} is deleted, i.e.,*

$$\psi(\{\boldsymbol{h}_w^l | w \in \mathcal{S}_{uv}\}) - \psi(\{\boldsymbol{h}_w'^l | w \in \mathcal{S}_{uv/e_{uv}}\})] = \delta, \tag{2}$$

where ψ is an operator that combines the elements of S_{uv} (e.g., concatenation, summation, etc.), $\mathcal{S}_{uv/e_{uv}}$ represent the subgraph excluding the information from e_{ij}, and δ is an infinitesimal constant.

4.2 Layer-Wise Deletion Operator

Perturbing weights or any other hyperparameter of the GNN model m can affect the decisions for multiple nodes and edges of graph G because we allow information to pass through the network

from the local neighborhood of each node. Hence, to ensure effective deletion of an edge e_{uv}, it is important to eliminate signals with minor contributions for predicting edges and develop knobs that can tune/perturb any source of node/edge information that aid in the prediction of e_{uv}. To provide the flexibility to delete the knowledge of specific nodes/edges, we introduce a model-agnostic deletion operator DEL that can be applied to any type of GNN layer.

Deletion Operator. Following the notations of Section 3, for the l-th GNN layer $g^l(G)$ with output dimension d^l, we define an extended GNN layer with unlearning capability as $(\text{DEL}^l \circ g^l)(G)$ with the same output dimension d^l . Given an edge e_{uv} that is to be removed, DEL is applied to the node representations and is defined as:

$$\text{DEL}^l = \begin{cases} \phi & \text{if } w \in S^l_{uv} \\ \mathbb{1} & \text{otherwise} \end{cases}, \tag{3}$$

where $\mathbb{1}$ is the identity function, and $\phi : \mathbb{R}^{n \times d^l} \rightarrow \mathbb{R}^{n \times d^l}$ can be any differentiable function that takes as input the output node representations of g. In this work, ϕ is considered as an MLP with weight parameters $\boldsymbol{W}^l_D$. Similarly to other GNN operators, the weights $\boldsymbol{W}^l_D$ of our DEL operator are shared across all nodes to achieve efficiency and scalability.

Local Update. Defining an operator that acts only in the local neighborhood $\mathcal{S}_{uv}$ enables targeted unlearning, keeping the previously learned knowledge intact as much as possible. If node u is within the local neighborhood of e_{uv}, DEL is activated. For other nodes, DEL remains deactivated and does not affect the hidden states of the nodes. This ensures that the model will not forget the knowledge it has gained before during training and the predictive performance on $\mathcal{E} \backslash S_{uv}$ will not drop.

By applying the deletion operator DEL to every GNN layer, we expect the final representations to reflect the unlearned information in the downstream task. Next, we show a theoretical observation over the unlearned node representations that indicates a stable behavior of the deletion operator:

Theorem 1. *(Bounding edge prediction using initial model m and unlearned model m') Let e_{uv} be an edge to be removed, $\boldsymbol{W}^L_D$ be the weight matrix of the deletion operator DEL^L, and normalized Lipschitz activation function $\sigma(\cdot)$. Then, the norm difference between the dot product of the final node representations from the initial model $\boldsymbol{z}_u, \boldsymbol{z}_v$ and from the unlearned one $\boldsymbol{z}'_u, \boldsymbol{z}'_v$ is bounded by:*

$$\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle - \langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle \geq -\frac{1 + \|\boldsymbol{W}^L_D\|^2}{2} \|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2, \tag{4}$$

where $\boldsymbol{W}^L_D$ denotes the weight matrix of the deletion operator for the l-th GNN layer.

The proof is in Appendix B. By Theorem 1, $\langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle$, and consequently the prediction probability for edge e_{uv} from the unlearned model cannot be dissimilar from the baseline. Nevertheless, DEL^l is a layer-wise operator, which provides stable node embeddings as compared to the initial ones.

4.3 MODEL UNLEARNING

Moving from a layer-wise operator to the whole GNN model, our method GNNDELETE applies DEL to every GNN layer, leading to a total number of trainable parameters $\sum_l (d^l)^2$. As the number of trainable parameters in GNNDELETE is independent of the size of the graph, it is compact and scalable to larger graphs and the number of deletion requests. Considering the properties defined in Section 4.1, we design two loss functions and compute them in a layer-wise manner. Specifically, for the l-th GNN layer we first compute the *Deleted Edge Consistency* loss:

$$\mathcal{L}^l_{\text{DEC}} = \mathcal{L}(\{[\boldsymbol{h}'^l_u; \boldsymbol{h}'^l_v] | e_{uv} \in \mathcal{E}_d\}, \{[\boldsymbol{h}^l_u; \boldsymbol{h}^l_v] | u, v \in_R \mathcal{V}\}), \tag{5}$$

and the *Neighborhood Influence* loss:

$$\mathcal{L}^l_{\text{NI}} = \mathcal{L}(\|_w \{\boldsymbol{h}'^l_w | w \in \mathcal{S}^l_{uv} / e_{uv}\}, \|_w \{\boldsymbol{h}^l_w | w \in \mathcal{S}^l_{uv}\}), \tag{6}$$

where $[\boldsymbol{h}'^l_u; \boldsymbol{h}'^l_v]$ denotes the concatenation of two vectors, and $\|$ denotes the concatenation of multiple vectors. Note that according to Equations 1 and 2, we choose the functions ϕ, ψ to be the concatenation operators. During the backward pass, the deletion operator at the l-th GNN layer is only optimized based on the weighted total loss at the l-th layer, i.e.

$$\boldsymbol{W}^{l^*}_D = \arg\min_{\boldsymbol{W}^l_D} \mathcal{L}^l = \arg\min_{\boldsymbol{W}^l_D} \lambda \mathcal{L}^l_{\text{DEC}} + (1 - \lambda) \mathcal{L}^l_{\text{NI}}, \tag{7}$$

where $\lambda \in [0, 1]$ is a regularization coefficient that balances the trade-off between the two properties.

Broad Applicability of GNNDELETE. GNNDELETE treats node representations in a model-agnostic manner, allowing us to consider graph unlearning in models beyond GNNs. Graph transformers (Ying et al., 2021; Rampášek et al., 2022) have been proposed recently as an extension of the Transformer architecture (Vaswani et al., 2017) for learning predictions on graphs. The DEL operator can be, also, applied after the computation of the node representations in such models. For example, the $\text{MPNN}_e^l(\boldsymbol{X}^l, \boldsymbol{E}^l, \boldsymbol{A})$ layer in GraphGPS (Rampášek et al., 2022, Equation 2) can be replaced with the unlearned version $(\text{DEL} \circ \text{MPNN}_e^l)$. Similarly, in the Graphormer layer, DEL operator can be applied after the multi-head attention MHA layer (Rampášek et al., 2022, Equation 8).

5 EXPERIMENTS

Next, we present our empirical results for GNNDELETE. In particular, we address the following questions: **Q1**) How does GNNDELETE perform as compared to baselines? **Q2**) Can GNNDELETE be extended to unlearning nodes and node features? **Q3**) How does the interplay between Deleted Edge Consistency and Neighborhood Influence property affect deletion performance?

5.1 EXPERIMENTAL SETUP

Datasets. We evaluate GNNDELETE on several widely-used graphs at various scales. We use 5 homogeneous graphs: Cora (Bojchevski & Günnemann, 2018), PubMed (Bojchevski & Günnemann, 2018), DBLP (Bojchevski & Günnemann, 2018), CS (Bojchevski & Günnemann, 2018), OGB-Collab (Hu et al., 2020), and 2 heterogeneous graphs: OGB-BioKG (Hu et al., 2020), and WordNet18RR (Dettmers et al., 2018). Table 5 includes details on graph datasets and deleted edges.

GNNs and Baselines. We test with four GNN architectures and two graph types to show the flexibility of our GNNDELETE operator. In particular, we test on GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019) for homogeneous graphs, and R-GCN (Schlichtkrull et al., 2018) and R-GAT (Chen et al., 2021b) for heterogeneous graphs. We consider four baseline methods: i) GRAPHEDITOR (Cong & Mahdavi), a method that finetunes on a closed-form solution of linear GNN models; ii) GRAPHERASER (Chen et al., 2021c), a re-training-based machine unlearning method for graphs; iii) GRADASCENT, which performs gradient ascent on $\mathcal{E}_d$ with cross-entropy loss and iv) DESCENT-TO-DELETE (Neel et al., 2021), a general machine unlearning method.

Implementation Details. We test GNNDELETE on edge deletion tasks, and show that it can also handle node deletion and node feature update tasks. In particular, we show the effectiveness of GNNDELETE on two different settings: i) an easier setting where information far away from the test set is deleted, and ii) a harder setting where information close to testset is deleted. For edge deletion tasks, we delete a varying size of $\mathcal{E}_d$ between $[0.5\% - 5.0\%]$ of total edges, with a step size of 0.5%. For larger datasets like OGB (Hu et al., 2020), we limit the maximum deletion ratio to 2.5%. We report the average and standard error of the unlearning performance across five independent runs. Regarding the evaluating metrics, we use AUROC to evaluate GNNDELETE for the link prediction task, as well as the Membership Inference (MI) (Thudi et al., 2021a) for the node deletion. An elaborate discussion of the evaluation metrics can be found in Appendix C.1. Moreover, in Appendix C.2 we describe two sampling strategies that we apply on $\mathcal{E}_d$.

5.2 **Q1**: RESULTS – COMPARISON TO BASELINES

We compare GNNDELETE to the baseline unlearning techniques and present the results in Tables 1-2. Across three GNN architectures, we find that GNNDELETE achieves the best performance on the test edge set $\mathcal{E}_t$, outperforming GRAPHEDITOR and GRAPHERASER by 13.9% and 38.8%. Further, we observe that GNNDELETE achieves the highest AUROC on $\mathcal{E}_d$, outperforming GRAPHEDITOR and GRAPHERASER by 32.2% and 25.4%. GNNDELETE even outperforms RETRAIN-FROM-SCRATCH by 21.7% under this setting, proving its capability of effectively unlearning the deleted edges. Interestingly, none of the existing baseline methods have comparable performance to GNNDELETE on these performance metrics, including GRAPHERASER, which ignores the global connectivity pattern and overfit to specific shards, and GRAPHEDITOR, whose choice of linear architecture strongly limits the power of the GNN unlearning. In contrast to Table 1, results in Table 2 show that

Table 1: AUROC (↑) performance of graph unlearning methods on deleting $\mathcal{E}_t$ and $\mathcal{E}_d$ from the DBLP dataset. Shown is the average AUROC performance on deleting 2.5% of the total edges. The best performance is marked in **bold**, and the second best is underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$. See Appendix Tables 10-11 for results on other datasets and ratios of deleted edges.

Model	GCN		GAT		GIN	
	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
RETRAIN	0.964 ±0.003	0.506 ±0.013	0.956 ±0.002	0.525 ±0.012	0.931 ±0.005	0.581 ±0.014
GRADASCENT	0.555 ±0.066	0.594 ±0.063	0.501 ±0.020	0.592 ±0.017	0.700 ±0.025	0.524 ±0.017
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
GRAPHERASER	0.527 ±0.002	0.500 ±0.000	0.538 ±0.013	0.500 ±0.000	0.517 ±0.009	0.500 ±0.000
GRAPHEDITOR	0.776 ±0.025	0.432 ±0.009	0.776 ±0.025	0.432 ±0.009	0.776 ±0.025	0.432 ±0.009
GNNDELETE	**0.934 ±0.002**	**0.748 ±0.006**	**0.914 ±0.007**	**0.774 ±0.015**	**0.897 ±0.006**	**0.740 ±0.015**

Table 2: AUROC (↑) performance of graph unlearning methods on deleting $\mathcal{E}_t$, and $\mathcal{E}_d$ from the WordNet18 dataset. Shown is the average AUROC performance on deleting 2.5% of the total edges. The best performance is marked in **bold**, and the second best is underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$. Graph unlearning on heterogeneous graphs does not apply to GRAPHEDITOR (i.e., N/A) as it supports unlearning tasks using only linear GNNs on homogeneous graphs. See Appendix Tables 14-15 for results on other datasets and ratios of deleted edges.

Model	R-GCN		R-GAT	
	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
RETRAIN	0.800 ±0.005	0.580 ±0.006	0.891 ±0.005	0.783 ±0.009
GRADASCENT	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
GRAPHERASER	0.512 ±0.003	0.500 ±0.000	0.545 ±0.015	0.500 ±0.000
GRAPHEDITOR	N/A	N/A	N/A	N/A
GNNDELETE	**0.751 ±0.006**	**0.845 ±0.007**	**0.893 ±0.002**	**0.786 ±0.004**

retraining is ineffective when deleting edges on large graphs. Our results demonstrate that baselines like DESCENT-TO-DELETE and GRADASCENT lose almost their predictive prowess in making meaningful predictions and distinguishing deleted edges because the weight updates are independent of the unlearning task and affect all the nodes, including nodes associated with $\mathcal{E}_t$. Please refer to the Appendix for results on PubMed (Tables 8-9), DBLP (Tables 10-11), OGB-Collab (Tables 12-13), and WordNet18 (Tables 14-15) using a deletion ratio of 0.5%, 2.5%, and 5%.

Results in Table 3 show the Membership Inference (MI) performance of baselines and GNNDELETE for the DBLP and Wordnet18 using a deletion ratio of 2.5%. It shows that GNNDELETE outperforms baselines for most GNN models, highlighting its effectiveness in hiding deleted data. Across five GNN architectures, we find that GNNDELETE improves on the MI ratio score of all baselines: GRAPHEDITOR (+9.7%), GRAPHERASER (+22.5%), RETRAIN-FROM-SCRATCH (+8.6%), GRADASCENT (+23.5%), and DESCENT-TO-DELETE (+2.4%).

5.3 Q2: RESULTS – OTHER UNLEARNING TASKS AND GNNDELETE'S EFFICIENCY

Node Deletion Task. Here, we show the flexibility of GNNDELETE in handling node learning tasks. We delete 100 nodes and their associated edges from the training data and evaluate the performance of the unlearning method using the Membership Inference attack metric. Results in Table 6 show that GNNDELETE outperforms baselines on node classification performance while deleting nodes. GNNDELETE outperforms GRAPHEDITOR by 1.8% accuracy and 1.9% F1 score. It is also 1.4% better than GRAPHEDITOR in terms of membership inference attacks. See Table 21 in Appendix for results of node feature update.

Time and Space Efficiency. We demonstrate that GNNDELETE is time-efficient as compared to most unlearning baselines. For all methods, we use a 2-layer GCN/R-GCN architecture with a trainable entity and relation embeddings with 128, 64, and 32 hidden dimensions trained on three datasets (PubMed, CS, and OGB-Collab). We present the results of wall-clock time vs. graph size in Figure 2 and observe that GNNDELETE consistently takes much less time than existing graph unlearning methods. Specifically, GNNDELETE is $\mathbf{12.3}\times$ faster than RETRAIN-FROM-SCRATCH on WordNet. For smaller graphs like DBLP, GNNDELETE takes 185 seconds less (18.5% faster)

Table 3: MI ratio (↑) performance of graph unlearning methods on DBLP and the WordNet18 datasets. Shown is the average MI ratio performance on deleting 5% of the total edges. The best performance is marked in **bold**, and the second best is underlined. Graph unlearning on heterogeneous graphs does not apply to GRAPHEDITOR (i.e., N/A) as it supports unlearning tasks using only linear GNNs on homogeneous graphs.

	DBLP			WordNet18	
Model	GCN	GAT	GIN	R-GCN	R-GAT
RETRAIN	1.255 ±0.207	1.223 ±0.151	1.200 ±0.177	1.250 ±0.091	1.215 ±0.125
GRADASCENT	1.180 ±0.061	1.112 ±0.109	1.123 ±0.103	1.169 ±0.066	1.112 ±0.106
D2D	1.264 ±0.000	1.264 ±0.000	**1.264 ±0.000**	**1.268 ±0.000**	1.268 ±0.000
GRAPHERASER	1.101 ±0.032	1.182 ±0.104	1.071 ±0.113	1.199 ±0.048	1.173 ±0.104
GRAPHEDITOR	1.189 ±0.193	1.189 ±0.193	1.189 ±0.193	N/A	N/A
GNNDELETE	**1.266 ±0.106**	**1.338 ±0.122**	1.254 ±0.159	1.264 ±0.143	**1.280 ±0.144**

than the pre-training stage of GRAPHEDITOR. Despite taking less time, the predictive performance of DESCENT-TO-DELETE and GRADASCENT is poor compared to GNNDELETE because they are not tailored to incorporate the graph structure for unlearning. Regarding the space efficiency, we measure the number of training parameters and show that GNNDELETE has the smallest model size. In addition, the number of training parameters does not scale with respect to the graph size, proving the efficiency and scalability of GNNDELETE. For instance, GNNDELETE takes $\mathbf{9.3}\times$ less computation than GRAPHERASER. Due to space limitation, we refer to Appendix Table 7 for additional results and details.

5.4 **Q3**: RESULTS – DELETED EDGE CONSISTENCY VS. NEIGHBORHOOD INFLUENCE

We conduct ablations on two key properties of GNNDELETE, namely the Deleted Edge Consistency and Neighborhood Influence, by varying the regularization parameter λ in Equation 7. Results in Table 4 show that both properties are necessary to high AUROC on $\mathcal{E}_t$ and $\mathcal{E}_d$. We observe that GNNDELETE focuses on Neighborhood Influence as λ decreases, explaining why the performance on $\mathcal{E}_t$ is close to the original while the model cannot distinguish $\mathcal{E}_d$ from the remaining edges. For higher values of λ, GNNDELETE focuses on optimizing the Deleted Edge Consistency property and can better distinguish between $\mathcal{E}_d$ and $\mathcal{E}_r$. In summary, we observe an improvement of 5.56% in the average AUROC for $\lambda = 0.5$.

Table 4: Ablation study on the interplay of Deleted Edge Consistency and Neighborhood Influence property on the DBLP dataset. Gap is calculated as: $|\text{AUROC}(\mathcal{E}_t) - \text{AUROC}(\mathcal{E}_d)|$. Best overall deletion performance is achieved for $\lambda = 0.5$, indicating that both properties are necessary to successfully delete information from the GNN model while minimizing negative effects on overall model performance.

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)

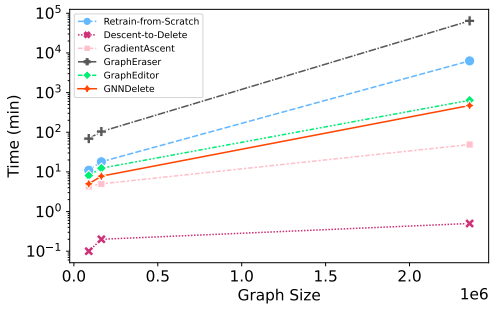

Figure 2: Efficiency comparison in terms of time versus graph size on three datasets (PubMed, CS, and OGB-Collab) of GNNDELETE with the baseline methods. We can observe GNNDELETE scales better than existing graph unlearning methods.

6 CONCLUSION

In this work, we propose a flexible and easy-to-use deletion operator that can be applied to any type of GNN model. Along with the deletion operator, we introduce Deleted Edge Consistency and Neighborhood Influence as two properties that can contribute to more effective graph unlearning. Combining the operator with the two properties, we introduce a novel loss function for the unlearning

task. We evaluate GNNDELETE in the link prediction task and we show that our method outperforms the existing graph unlearning models for the case of edge deletion, showing on parallel competitive results for node deletion. Overall, the performance behavior across a variety of problems and the ease of use demonstrate the potential of GNNDELETE to become the graph unlearning workhorse.

REFERENCES

Nasser Aldaghri, Hessam Mahdavifar, and Ahmad Beirami. Coded machine unlearning. *IEEE Access*, 9:88137–88150, 2021.

Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *arXiv preprint arXiv:2002.02730*, 2020.

Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1ZdKJ-0W.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

Steven L Bressler and Anil K Seth. Wiener–granger causality: a well established methodology. *Neuroimage*, 2011.

Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pp. 1092–1104. PMLR, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480, 2015a. doi: 10.1109/SP.2015.35.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480. IEEE, 2015b.

Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511997. URL https://doi.org/10.1145/3485447.3511997.

Kongyang Chen, Yiwen Wang, and Yao Huang. Lightweight machine unlearning in neural network. *arXiv preprint arXiv:2111.05528*, 2021a.

Meiqi Chen, Yuan Zhang, Xiaoyu Kou, Yuntao Li, and Yan Zhang. r-gat: Relational graph attention network for multi-relational graphs. *arXiv*, 2021b.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. *arXiv preprint arXiv:2103.14991*, 2021c.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 896–911, 2021d.

Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Zero-shot machine unlearning. *arXiv preprint arXiv:2201.05629*, 2022.

Weilin Cong and Mehrdad Mahdavi. Grapheditor: An efficient graph representation learning and unlearning approach.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Shaopeng Fu, Fengxiang He, and Dacheng Tao. Knowledge removal in sampling-based bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dTqOcTUOQO.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making AI forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems*, 32, 2019a.

Antonio A. Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. *Making AI Forget You: Data Deletion in Machine Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019b.

Shashwat Goel, Ameya Prabhu, and Ponnurangam Kumaraguru. Evaluating inexact unlearning requires revisiting forgetting. *arXiv preprint arXiv:2201.06640*, 2022.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9301–9309, 2020.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 792–801, 2021.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, 1969.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 16319–16330. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/87f7ee4fdb57bdfd52179947211b7ebb-Paper.pdf.

Peter Hase, Mona T. Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srini Iyer. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *ArXiv*, abs/2111.13654, 2021.

Yingzhe He, Guozhu Meng, Kai Chen, Jinwen He, and Xingbo Hu. Deepobliviate: A powerful charm for erasing data residual memory in deep neural networks. *ArXiv*, abs/2105.06209, 2021.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf H Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=8nvgnORnoWr.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2008–2016. PMLR, 2021.

Rahif Kassab and Osvaldo Simeone. Federated generalized bayesian learning via distributed stein variational gradient descent. *IEEE Transactions on Signal Processing*, 70:2180–2192, 2022.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

Yuyuan Li, Xiaolin Zheng, Chaochao Chen, and Junlin Liu. Making recommender systems forget: Learning and unlearning for erasable recommendation, 2022. URL https://arxiv.org/abs/2203.11491.

Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federated unlearning. *arXiv preprint arXiv:2012.13891*, 2020a.

Yang Liu, Zhuo Ma, Ximeng Liu, Jian Liu, Zhongyuan Jiang, Jianfeng Ma, Philip Yu, and Kui Ren. Learn to forget: Machine unlearning via neuron masking. *arXiv preprint arXiv:2003.10933*, 2020b.

Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. Backdoor defense with machine unlearning. *arXiv preprint arXiv:2201.09538*, 2022a.

Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pp. 1749–1758, 2022b. doi: 10.1109/INFOCOM48880.2022.9796721.

Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

Deisy Morselli Gysi, Ítalo Do Valle, Marinka Zitnik, Asher Ameli, Xiao Gan, Onur Varol, Susan Dina Ghiassian, JJ Patten, Robert A Davey, Joseph Loscalzo, et al. Network medicine framework for identifying drug-repurposing opportunities for covid-19. *Proceedings of the National Academy of Sciences*, 118(19):e2025581118, 2021.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl, 2018.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962. PMLR, 2021.

Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, WWW '18. International World Wide Web Conferences Steering Committee, 2018.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33:16025–16036, 2020.

Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. *arXiv preprint arXiv:2101.06570*, 2021.

Francesco Orabona. A modern introduction to online learning, 2019. URL https://arxiv.org/abs/1912.13213.

Nishchal Parne, Kyathi Puppaala, Nithish Bhupathi, and Ripon Patgiri. Machine unlearning: Learning, polluting, and unlearning for spam email. *arXiv preprint arXiv:2111.14609*, 2021.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2022. URL https://arxiv.org/abs/2205.12454.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *ICML*, 2019.

Sebastian Schelter, Stefan Grafberger, and Ted Dunning. *HedgeCut: Maintaining Randomised Trees for Low-Latency Machine Unlearning*. Association for Computing Machinery, New York, NY, USA, 2021. URL https://doi.org/10.1145/3448016.3457239.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=pvCLqcsLJ1N.

Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In *IJCAI*, 2021.

David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247*, 2020.

Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *arXiv preprint arXiv:2111.08947*, 2021.

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. *arXiv preprint arXiv:2109.13398*, 2021a.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. *arXiv preprint arXiv:2110.11891*, 2021b.

Anvith Thudi, Ilia Shumailov, Franziska Boenisch, and Nicolas Papernot. Bounding membership inference. *arXiv preprint arXiv:2202.12232*, 2022.

Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pp. 4126–4142. PMLR, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. *arXiv preprint arXiv:2110.11794*, 2021.

Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. DeltaGrad: Rapid retraining of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2020. URL https://proceedings.mlr.press/v119/wu20b.html.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PzcvxEMzvQC.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2017. URL https://arxiv.org/abs/1709.01604.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=OeWooOxFwDa.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. *GNNExplainer: Generating Explanations for Graph Neural Networks*. Curran Associates Inc., Red Hook, NY, USA, 2019.

A Related Work: Cont'd

Machine Unlearning for Other Tasks. Given the increasing necessity of machine unlearning, researches have studied machine unlearning algorithms tailored for other tasks. For example, Chen et al. (2022) and Li et al. (2022) propose machine unlearning frameworks for recommendation systems by considering the collaborative information. In addition, recent unlearning methods use Bayesian and latent models (Nguyen et al., 2020; Kassab & Simeone, 2022; Sekhari et al., 2021). Machine unlearning (MU) has also been applied for other tasks, including verification of MU (Sommer et al., 2020), re-definition of MU (Thudi et al., 2021b), test of MU (Goel et al., 2022), zero-shot MU (Chundawat et al., 2022), tree-based MU (Schelter et al., 2021), coded MU (Aldaghri et al., 2021), pruning based federated MU (Wang et al., 2021), adaptive sequence deletion MU (Gupta et al., 2021), MU for k-means (Ginart et al., 2019a), MU for features and labels (Warnecke et al., 2021), MU for spam email detection (Parne et al., 2021), MU for linear models (Mahadevan & Mathioudakis, 2021). While the above mentioned methods have shown new directions for machine unlearning, they are not comparable to our work as we focus on general graph unlearning instead of a specific ML task.

B Proof for Theorem 1

For the L-th GNN layer, we assume for sake of simplicity that the final node representation $\boldsymbol{z}_u$ for a node u is computed as:

$$\boldsymbol{z}_u = \sigma\Big(\boldsymbol{W}_1 \boldsymbol{h}_u^{L-1} + \sum_{v \in \mathcal{N}_u} \boldsymbol{W}_2 \mathbf{h}_v^{L-1}\Big), \tag{8}$$

where σ is the sigmoid function, and, and $\boldsymbol{W}_1, \boldsymbol{W}_2$ are the weight parameters for the GNN layer. Similarly, for the unlearned L-th layer we have:

$$\boldsymbol{z}'_u = \sigma(\boldsymbol{W}_D^L \boldsymbol{z}_u^L), \tag{9}$$

where $\boldsymbol{W}_D^L$'s are the weight parameters for the DEL operator. For a given edge $e_{uv} \in \mathcal{E}_d$ that is to be deleted, we expect for the dot product $\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle$ to be maximized, as it is an existent edge in the initial graph, while $\langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle$ to be fixed, so that the edge probability is close to 0.5, following the *Deleted Edge Consistency* property. Specifically, the difference of the two terms can be bounded as:

$$\begin{aligned}
\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle - \langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle &= \frac{1}{2}(\|\boldsymbol{z}_u\|^2 + \|\boldsymbol{z}_v\|^2 - \|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2) - \frac{1}{2}(\|\boldsymbol{z}'_u\|^2 + \|\boldsymbol{z}'_v\|^2 + \|\boldsymbol{z}'_u - \boldsymbol{z}'_v\|^2) \\
\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle - \langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle &\overset{\text{Normalization}}{=} 1 - \frac{1}{2}\|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2 - 1 - \frac{1}{2}\|\boldsymbol{z}'_u - \boldsymbol{z}'_v\|^2
\end{aligned} \tag{10}$$

Then, simplifying Equations 8 and 9, we have:

$$\begin{aligned}
\|\boldsymbol{z}'_u - \boldsymbol{z}'_v\| &= \|\sigma(\boldsymbol{W}_D^L \boldsymbol{z}_u) - \sigma(\boldsymbol{W}_D^L \boldsymbol{z}_v)\| \\
&\overset{\text{Lipschitz } \sigma}{\leq} \|\boldsymbol{W}_D^L \boldsymbol{z}_u - \boldsymbol{W}_D^L \boldsymbol{z}_v\| \\
&\overset{\text{Cauchy-Schwartz}}{\leq} \|\boldsymbol{W}_D^L\| \|\boldsymbol{z}_u - \boldsymbol{z}_v\|
\end{aligned}$$

and applying that to Equation 10:

$$\begin{aligned}
\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle - \langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle &\geq -\frac{1}{2}\|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2 - \frac{1}{2}\|\boldsymbol{W}_D^L\|^2 \|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2 \\
\langle \boldsymbol{z}_u, \boldsymbol{z}_v \rangle - \langle \boldsymbol{z}'_u, \boldsymbol{z}'_v \rangle &\geq -\frac{1}{2}(1 + \|\boldsymbol{W}_D^L\|^2) \|\boldsymbol{z}_u - \boldsymbol{z}_v\|^2
\end{aligned}. \tag{11}$$

C Experiments

In this section, we give further details on the experimental setup. We firstly present the used evaluation metrics for the edge and node deletion, then we describe the sampling strategies for $\mathcal{E}_t$, and $\mathcal{E}_d$, dataset statistics, and finally we present the full results for the model efficiency and the prediction performance of the models.

Table 5: Statistics of evaluated datasets and maximum number of deleted edges.

Graph	# Nodes	# Edges	# Unique edge types	Max # deleted edges
Cora	19,793	126,842		6,342
PubMed	19,717	88,648		4,432
DBLP	17,716	105,734	N/A	5,286
CS	18,333	163,788		8,189
OGB-Collab	235,868	2,358,104		117,905
WordNet18	40,943	151,442	18	7,072
OGB-BioKG	93,773	5,088,434	51	127,210

C.1 Evaluation Metrics

We choose the following performance metrics to measure the effectiveness of the deletion operation on the set of the deleted edges $\mathcal{E}_d$ and the test set $\mathcal{E}_t$ that contains a subset of the remaining edges:. For the edge deletion case, we have:

- AUROC and AUPRC on the test set $\mathcal{E}_t$: these metrics measure the prediction perfomance of each graph graph learning model over the existent edges of the test set $\mathcal{E}_t$. High values of AUROC and AUPRC on $\mathcal{E}_t$ show that the unlearned model's performance on the original test set is not affected by the deletion of an edge subset.
- AUROC and AUPRC on the set of deleted edges $\mathcal{E}_d$: these metrics quantify the ability of the unlearned models to distinguish deleted edges (contained in $\mathcal{E}_d$) from the remaining edges (contained in $\mathcal{E}_r$). For the computation of the area under the curve, we take into account the total of the deleted edges in $\mathcal{E}_d$ and we sample an equal amount of remaining edges from $\mathcal{E}_r$. Then, we set the labels of the deleted edges equal to 0 (since they do not exist after deletion), and the labels of the remaining edges to 1. Higher values of the AUC on $\mathcal{E}_d$ show that the model is more capable of distinguishing edges in $\mathcal{E}_d$ from edges in $\mathcal{E}_r$.

For the node deletion case, we capitalize on the evaluation for privacy leakage with Membership Inference (MI) (Thudi et al., 2021a) attacks. An unlearned model f' effectively forgets $\mathcal{E}_d$ if an MI attacker returns that $\mathcal{E}_d$ is not present in the training set, i.e., the probability of predicting the presence of $\mathcal{E}_d$ decreases.

- MI Ratio: this metric quantifies the success rate of a Membership Inference (MI) attack, by calculating the ratio of presence probability of $\mathcal{E}_d$ before and after the deletion operator. We adapt the implementation from Olatunji et al. (2021) in our experiments. If the ratio is higher than 1, it means that the model contains less information about $\mathcal{E}_d$. If the ratio is less than 1, it means the model contains more information about $\mathcal{E}_d$.

C.2 Sampling of $\mathcal{E}_t$ and $\mathcal{E}_d$

We sample 5% of the total edges as the test set ($\mathcal{E}_t$) to evaluate the model's performance on link prediction and sample another 5% as validation set for selecting the best model. We propose two sampling strategies for sampling $\mathcal{E}_d$ for edge deletion tasks: i) $\mathcal{E}_{d,\text{OUT}}$ refers to randomly sampling edges outside the 2-hop enclosing subgraph of $\mathcal{E}_t$, i.e., $\mathcal{E}_{d,\text{OUT}} = \{e | e \notin S^2_{\mathcal{E}_t}\}$; and ii) $\mathcal{E}_{d,\text{IN}}$ refers to randomly sampling edges from the 2-hop enclosing subgraph of $\mathcal{E}_t$, i.e., $\mathcal{E}_{d,\text{IN}} = \{e | e \in S^2_{\mathcal{E}_t}\}$. We note that deleting $\mathcal{E}_{d,\text{IN}}$ is more difficult than deleting $\mathcal{E}_{d,\text{OUT}}$ as the deletion operation will have an impact on the local neighborhood of $\mathcal{E}_{d,\text{IN}}$, where $\mathcal{E}_t$ is located. For comparison of these two sampling strategies, please refer to the Appendix Tables 8-9 for results on PubMed, Tables 10-11 on DBLP, Tables 12-13 on OGB-Collab, and Tables 14-15 on WordNet18.

C.3 Evaluated datasets

Table 5 presents the size of the graphs and the maximum number of deleted edges in the experiments, ranging from small to large scales.

Table 6: Node deletion performance on DBLP with 100 nodes deleted. **Bold**: best performance. Underline: second best performance.

Model	Accuracy	F1	MI ratio
RETRAIN	0.845 ±0.008	0.841 ±0.004	1.515 ±0.034
GRADASCENT	0.392 ±0.026	0.341 ±0.035	1.021 ±0.113
D2D	0.250 ±0.000	0.250 ±0.000	**1.755** ±0.065
GRAPHERASER	0.718 ±0.014	0.716 ±0.011	0.975 ±0.083
GRAPHEDITOR	**0.765 ±0.012**	**0.749 ±0.006**	1.260 ±0.088
GNNDELETE	0.748 ±0.013	0.740 ±0.015	1.219 ±0.079

C.4 MODEL EFFICIENCY

Space efficiency is reflected by the number of trainable parameters a model has. We report number of parameters for all methods in Table 7. We can observe that GNNDELETE has the smallest model size, which does not scale with respect to the size of the graph. This proves the efficiency and scalability of GNNDELETE. It is also significantly smaller than GRAPHERASER, where we usually have to divide the original graph into 10 or 20 shards, each requiring a separate GNN model.

Table 7: Space efficiency of unlearning models. All models are trained on OGB-Collab and OGB-BioKG to delete 5.0% of edges, using a 2-layer GCN/R-GCN architecture with 128, 64, 32 as hidden dimensions, with number of shards in GRAPHERASER as 10.

Model	OGB-Collab	OGB-BioKG
RETRAIN-FROM-SCRATCH	5216	12,009,792
GRADASCENT	5216	12,009,792
DESCENT-TO-DELETE	5216	12,009,792
GRAPHERASER	52160	120,097,920
GRAPHEDITOR	5216	N/A
Ours	5120	5120

D FULL RESULTS

We detail all the results from different graphs, GNN architectures, models, and deletion ratios.

Table 8: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using PubMed dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{OUT}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.968 ±0.001	0.687 ±0.023	0.931 ±0.003	0.723 ±0.026	0.941 ±0.004	0.865 ±0.012
	GRADASCENT	0.458 ±0.139	0.539 ±0.091	0.450 ±0.017	0.541 ±0.049	0.518 ±0.122	0.528 ±0.021
	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
	GRAPHERASER	0.529 ±0.013	0.500 ±0.000	0.542 ±0.004	0.500 ±0.000	0.535 ±0.003	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.566 ±0.015	0.657 ±0.006	0.566 ±0.015	0.657 ±0.006	0.566 ±0.015
	GNNDELETE	**0.961 ±0.004**	**0.973 ±0.005**	**0.926 ±0.006**	**0.976 ±0.005**	**0.940 ±0.005**	**0.963 ±0.010**
2.5	RETRAIN	0.967 ±0.001	0.696 ±0.011	0.930 ±0.003	0.736 ±0.011	0.942 ±0.005	0.875 ±0.008
	GRADASCENT	0.446 ±0.130	0.500 ±0.067	0.582 ±0.006	0.758 ±0.033	0.406 ±0.054	0.454 ±0.037
	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
	GRAPHERASER	0.505 ±0.024	0.500 ±0.000	0.538 ±0.009	0.500 ±0.000	0.544 ±0.014	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.570 ±0.011	0.657 ±0.006	0.570 ±0.011	0.657 ±0.006	0.570 ±0.011
	GNNDELETE	**0.954 ±0.003**	**0.909 ±0.004**	**0.920 ±0.004**	**0.916 ±0.006**	**0.943 ±0.005**	**0.938 ±0.009**
5.0	RETRAIN	0.966 ±0.001	0.707 ±0.004	0.929 ±0.002	0.744 ±0.008	0.942 ±0.004	0.885 ±0.010
	GRADASCENT	0.446 ±0.126	0.492 ±0.064	0.581 ±0.010	0.704 ±0.022	0.388 ±0.056	0.455 ±0.028
	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
	GRAPHERASER	0.532 ±0.001	0.500 ±0.000	0.527 ±0.022	0.500 ±0.000	0.524 ±0.015	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.572 ±0.006	0.657 ±0.006	0.572 ±0.006	0.657 ±0.006	0.572 ±0.006
	GNNDELETE	**0.950 ±0.003**	**0.859 ±0.005**	**0.921 ±0.005**	**0.863 ±0.006**	**0.941 ±0.002**	**0.930 ±0.009**

Table 9: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using PubMed dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{IN}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.968 ±0.001	0.493 ±0.040	0.931 ±0.003	0.533 ±0.037	0.940 ±0.002	0.626 ±0.041
	GRADASCENT	0.469 ±0.095	0.496 ±0.058	0.436 ±0.028	0.553 ±0.029	0.687 ±0.060	0.556 ±0.042
	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
	GRAPHERASER	0.547 ±0.004	0.500 ±0.000	0.536 ±0.000	0.500 ±0.000	0.524 ±0.002	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.469 ±0.021	0.657 ±0.006	0.469 ±0.021	0.657 ±0.006	0.469 ±0.021
	GNNDELETE	**0.951 ±0.005**	**0.838 ±0.014**	**0.909 ±0.003**	**0.888 ±0.016**	**0.929 ±0.006**	**0.835 ±0.006**
2.5	RETRAIN	0.968 ±0.001	0.499 ±0.019	0.931 ±0.002	0.541 ±0.013	0.937 ±0.004	0.614 ±0.015
	GRADASCENT	0.470 ±0.087	0.474 ±0.039	0.522 ±0.066	0.704 ±0.086	0.631 ±0.050	0.499 ±0.018
	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
	GRAPHERASER	0.538 ±0.003	0.500 ±0.000	0.521 ±0.003	0.500 ±0.000	0.533 ±0.010	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.467 ±0.006	0.657 ±0.006	0.467 ±0.006	0.657 ±0.006	0.467 ±0.006
	GNNDELETE	**0.920 ±0.014**	**0.739 ±0.010**	**0.891 ±0.005**	**0.759 ±0.012**	**0.909 ±0.005**	**0.782 ±0.013**
5.0	RETRAIN	0.967 ±0.001	0.503 ±0.009	0.929 ±0.003	0.545 ±0.005	0.936 ±0.005	0.621 ±0.003
	GRADASCENT	0.473 ±0.090	0.473 ±0.038	0.525 ±0.069	0.686 ±0.090	0.635 ±0.073	0.493 ±0.018
	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
	GRAPHERASER	0.551 ±0.004	0.500 ±0.000	0.524 ±0.020	0.500 ±0.000	0.531 ±0.000	0.500 ±0.000
	GRAPHEDITOR	0.657 ±0.006	0.468 ±0.002	0.657 ±0.006	0.468 ±0.002	0.657 ±0.006	0.468 ±0.002
	GNNDELETE	**0.916 ±0.006**	**0.691 ±0.012**	**0.887 ±0.009**	**0.713 ±0.005**	**0.895 ±0.004**	**0.761 ±0.005**

Table 10: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using DBLP dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{OUT}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.965 ±0.002	0.783 ±0.018	0.956 ±0.002	0.744 ±0.021	0.934 ±0.003	0.861 ±0.019
	GRADASCENT	0.567 ±0.008	0.696 ±0.017	0.501 ±0.030	0.667 ±0.052	0.753 ±0.055	0.789 ±0.091
	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
	GRAPHERASER	0.518 ±0.002	0.500 ±0.000	0.523 ±0.013	0.500 ±0.000	0.517 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.769 ±0.040	0.616 ±0.019	0.769 ±0.040	0.616 ±0.019	0.769 ±0.040	0.616 ±0.019
	GNNDELETE	**0.959 ±0.002**	**0.964 ±0.005**	**0.950 ±0.002**	**0.980 ±0.003**	**0.924 ±0.006**	**0.894 ±0.020**
2.5	RETRAIN	0.965 ±0.002	0.777 ±0.009	0.955 ±0.003	0.739 ±0.005	0.934 ±0.003	0.858 ±0.002
	GRADASCENT	0.528 ±0.015	0.583 ±0.016	0.501 ±0.026	0.576 ±0.017	0.717 ±0.022	0.766 ±0.019
	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
	GRAPHERASER	0.515 ±0.002	0.500 ±0.000	0.563 ±0.013	0.500 ±0.000	0.552 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.769 ±0.040	0.607 ±0.017	0.769 ±0.040	0.607 ±0.017	0.769 ±0.040	0.607 ±0.017
	GNNDELETE	**0.957 ±0.003**	**0.892 ±0.004**	**0.949 ±0.003**	**0.905 ±0.002**	**0.926 ±0.007**	**0.898 ±0.017**
5.0	RETRAIN	0.964 ±0.003	0.788 ±0.006	0.955 ±0.003	0.748 ±0.008	0.936 ±0.004	0.868 ±0.005
	GRADASCENT	0.555 ±0.099	0.591 ±0.065	0.501 ±0.023	0.559 ±0.024	0.672 ±0.032	0.728 ±0.022
	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
	GRAPHERASER	0.541 ±0.002	0.500 ±0.000	0.523 ±0.013	0.500 ±0.000	0.522 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.769 ±0.040	0.611 ±0.018	0.769 ±0.040	0.611 ±0.018	0.769 ±0.040	0.611 ±0.018
	GNNDELETE	**0.956 ±0.004**	**0.859 ±0.002**	**0.949 ±0.003**	**0.859 ±0.005**	**0.924 ±0.007**	**0.898 ±0.019**

Table 11: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using DBLP dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{IN}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.965 ±0.003	0.496 ±0.028	0.957 ±0.002	0.513 ±0.021	0.934 ±0.005	0.571 ±0.035
	GRADASCENT	0.556 ±0.018	0.657 ±0.008	0.511 ±0.023	0.612 ±0.107	0.678 ±0.084	0.573 ±0.045
	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
	GRAPHERASER	0.515 ±0.001	0.500 ±0.000	0.523 ±0.000	0.500 ±0.000	0.507 ±0.003	0.500 ±0.000
	GRAPHEDITOR	0.776 ±0.025	0.431 ±0.014	0.776 ±0.025	0.431 ±0.014	0.776 ±0.025	0.431 ±0.014
	GNNDELETE	**0.951 ±0.002**	**0.829 ±0.006**	**0.928 ±0.004**	**0.889 ±0.011**	**0.906 ±0.009**	**0.736 ±0.012**
2.5	RETRAIN	0.964 ±0.003	0.506 ±0.013	0.956 ±0.002	0.525 ±0.012	0.931 ±0.005	0.581 ±0.014
	GRADASCENT	0.555 ±0.066	0.594 ±0.063	0.501 ±0.020	0.592 ±0.017	0.700 ±0.025	0.524 ±0.017
	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
	GRAPHERASER	0.527 ±0.002	0.500 ±0.000	0.538 ±0.013	0.500 ±0.000	0.517 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.776 ±0.025	0.432 ±0.009	0.776 ±0.025	0.432 ±0.009	0.776 ±0.025	0.432 ±0.009
	GNNDELETE	**0.934 ±0.002**	**0.748 ±0.006**	**0.914 ±0.007**	**0.774 ±0.015**	**0.897 ±0.006**	**0.740 ±0.015**
5.0	RETRAIN	0.963 ±0.003	0.504 ±0.006	0.955 ±0.002	0.528 ±0.007	0.931 ±0.006	0.578 ±0.009
	GRADASCENT	0.555 ±0.060	0.581 ±0.073	0.490 ±0.022	0.551 ±0.030	0.723 ±0.032	0.516 ±0.042
	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
	GRAPHERASER	0.509 ±0.011	0.500 ±0.000	0.511 ±0.006	0.500 ±0.000	0.503 ±0.000	0.500 ±0.000
	GRAPHEDITOR	0.776 ±0.025	0.430 ±0.011	0.776 ±0.025	0.430 ±0.011	0.776 ±0.025	0.430 ±0.011
	GNNDELETE	**0.917 ±0.005**	**0.713 ±0.007**	**0.912 ±0.007**	**0.733 ±0.018**	**0.864 ±0.005**	**0.732 ±0.008**

Table 12: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using OGB-Collab dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{OUT}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.986 ±0.001	0.553 ±0.004	0.983 ±0.001	0.541 ±0.002	0.859 ±0.011	0.523 ±0.003
	GRADASCENT	0.606 ±0.012	0.509 ±0.002	0.885 ±0.206	0.535 ±0.020	0.665 ±0.097	0.511 ±0.009
	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
	GRAPHERASER	0.544 ±0.000	0.500 ±0.000	0.551 ±0.004	0.500 ±0.000	0.513 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.839 ±0.001	0.468 ±0.005	0.839 ±0.001	0.468 ±0.005	**0.839 ±0.001**	0.468 ±0.005
	GNNDELETE	**0.985 ±0.002**	**0.723 ±0.002**	**0.983 ±0.001**	**0.728 ±0.007**	0.580 ±0.269	**0.545 ±0.065**
2.5	RETRAIN	0.986 ±0.001	0.553 ±0.001	0.983 ±0.001	0.541 ±0.002	0.858 ±0.006	0.525 ±0.002
	GRADASCENT	0.963 ±0.009	0.552 ±0.006	0.976 ±0.003	0.543 ±0.003	0.645 ±0.094	0.513 ±0.008
	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
	GRAPHERASER	0.517 ±0.000	0.500 ±0.000	0.524 ±0.009	0.500 ±0.000	0.542 ±0.001	0.500 ±0.000
	GRAPHEDITOR	0.839 ±0.002	0.465 ±0.002	0.839 ±0.002	0.465 ±0.002	**0.839 ±0.002**	0.465 ±0.002
	GNNDELETE	**0.983 ±0.002**	**0.642 ±0.002**	**0.983 ±0.000**	**0.639 ±0.003**	0.579 ±0.271	**0.531 ±0.047**

Table 13: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using OGB-Collab dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{IN}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	GCN		GAT		GIN	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.986 ±0.001	0.521 ±0.002	0.983 ±0.001	0.644 ±0.002	0.855 ±0.007	0.599 ±0.003
	GRADASCENT	0.552 ±0.093	0.505 ±0.003	0.921 ±0.121	0.569 ±0.034	0.658 ±0.088	0.524 ±0.014
	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
	GRAPHERASER	0.554 ±0.002	0.500 ±0.000	0.512 ±0.003	0.500 ±0.000	0.532 ±0.002	0.500 ±0.000
	GRAPHEDITOR	0.839 ±0.002	0.571 ±0.003	0.839 ±0.002	0.571 ±0.003	**0.839 ±0.002**	**0.571 ±0.003**
	GNNDELETE	**0.976 ±0.002**	**0.715 ±0.002**	**0.974 ±0.001**	**0.814 ±0.011**	0.559 ±0.253	0.566 ±0.094
2.5	RETRAIN	0.986 ±0.001	0.532 ±0.002	0.982 ±0.001	0.650 ±0.004	0.845 ±0.010	0.612 ±0.003
	GRADASCENT	0.962 ±0.008	0.534 ±0.006	0.697 ±0.040	0.541 ±0.018	0.974 ±0.003	**0.585 ±0.004**
	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
	GRAPHERASER	0.539 ±0.001	0.500 ±0.000	0.542 ±0.013	0.500 ±0.000	0.528 ±0.009	0.500 ±0.000
	GRAPHEDITOR	0.839 ±0.002	0.575 ±0.001	0.839 ±0.002	0.575 ±0.001	**0.839 ±0.002**	0.575 ±0.001
	GNNDELETE	**0.972 ±0.003**	**0.665 ±0.002**	**0.966 ±0.005**	**0.772 ±0.015**	0.543 ±0.225	0.568 ±0.097

Table 14: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using WordNet18 dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{OUT}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	R-GCN		R-GAT	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.801 ±0.007	0.601 ±0.014	0.898 ±0.003	0.808 ±0.015
	GRADASCENT	0.499 ±0.002	0.501 ±0.003	0.495 ±0.001	0.411 ±0.012
	D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
	GRAPHERASER	0.517 ±0.001	0.500 ±0.000	0.533 ±0.004	0.500 ±0.000
	GNNDELETE	**0.757 ±0.005**	**0.901 ±0.008**	**0.899 ±0.002**	**0.828 ±0.010**
2.5	RETRAIN	0.804 ±0.005	0.639 ±0.004	0.897 ±0.004	0.831 ±0.005
	GRADASCENT	0.493 ±0.002	0.500 ±0.000	0.491 ±0.001	0.424 ±0.015
	D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
	GRAPHERASER	0.515 ±0.004	0.500 ±0.000	0.533 ±0.003	0.500 ±0.000
	GNNDELETE	**0.758 ±0.005**	**0.902 ±0.006**	**0.898 ±0.002**	**0.836 ±0.005**
5.0	RETRAIN	0.801 ±0.004	0.661 ±0.011	0.896 ±0.001	0.861 ±0.006
	GRADASCENT	0.493 ±0.001	0.500 ±0.000	0.491 ±0.001	0.425 ±0.025
	D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
	GRAPHERASER	0.525 ±0.002	0.500 ±0.000	0.531 ±0.004	0.500 ±0.000
	GNNDELETE	**0.756 ±0.004**	**0.910 ±0.006**	**0.897 ±0.002**	**0.852 ±0.004**

Table 15: AUROC (↑) performance of Graph Unlearning techniques on $\mathcal{E}_t$ and $\mathcal{E}_d$ using WordNet18 dataset, when $\mathcal{E}_d = \mathcal{E}_{d,\text{IN}}$. The best performances are marked in **bold** and the second best are underlined. GNNDELETE retains the predictive power of GNNs on $\mathcal{E}_t$ while improving their performance on $\mathcal{E}_d$ in comparison to all baselines.

Ratio (%)	Model	R-GCN		R-GAT	
		$\mathcal{E}_t$	$\mathcal{E}_d$	$\mathcal{E}_t$	$\mathcal{E}_d$
0.5	RETRAIN	0.802 ±0.007	0.584 ±0.005	0.898 ±0.002	0.771 ±0.011
	GRADASCENT	0.496 ±0.002	0.500 ±0.000	0.493 ±0.001	0.487 ±0.008
	D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
	GRAPHERASER	0.505 ±0.002	0.510 ±0.001	0.502 ±0.000	0.502 ±0.000
	GNNDELETE	**0.756 ±0.005**	**0.850 ±0.005**	**0.897 ±0.002**	**0.819 ±0.014**
2.5	RETRAIN	0.800 ±0.005	0.580 ±0.006	0.891 ±0.005	0.783 ±0.009
	GRADASCENT	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
	GRAPHERASER	0.512 ±0.003	0.500 ±0.000	0.545 ±0.015	0.500 ±0.000
	GNNDELETE	**0.751 ±0.006**	**0.845 ±0.007**	**0.893 ±0.002**	**0.786 ±0.004**
5.0	RETRAIN	0.797 ±0.003	0.588 ±0.005	0.883 ±0.002	0.786 ±0.005
	GRADASCENT	0.491 ±0.001	0.500 ±0.000	0.490 ±0.002	0.494 ±0.002
	D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
	GRAPHERASER	0.507 ±0.003	0.500 ±0.000	0.518 ±0.002	0.500 ±0.000
	GNNDELETE	**0.749 ±0.005**	**0.850 ±0.008**	**0.889 ±0.002**	**0.779 ±0.007**

Table 16: Comparison of layer-wise GNNDELETE layer-wise and final-layer-only GNNDELETE . Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: link prediction. Metric: AUC (↑).

Model	AUC on $\mathcal{E}_t$	AUC on $\mathcal{E}_d$
RETRAIN (reference only)	0.964 ±0.003	0.506 ±0.013
GNNDELETE - layer-wise	0.934 ±0.002	0.748 ±0.006
GNNDELETE - final layer only	0.927 ±0.005	0.739 ±0.005

Table 17: Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: node classification. Metric: Accuracy (↑)

Model	Acc on GCN	Acc on GAT	Acc on GIN
RETRAIN (reference only)	0.821 ±0.012	0.830 ±0.012	0.817 ±0.012
GRADASCENT	0.532 ±0.055	0.547 ±0.024	0.550 ±0.041
D2D	0.519 ±0.008	0.511 ±0.004	0.513 ±0.007
GRAPHERASER	0.681 ±0.037	0.694 ±0.022	0.673 ±0.033
GRAPHEDITOR	0.667 ±0.024	0.673 ±0.019	0.680 ±0.031
GNNDELETE	0.802 ±0.018	0.799 ±0.015	0.812 ±0.017

Table 18: Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: link prediction. Metric: AUC (↑)

Model	Acc on GCN	Acc on GAT	Acc on GIN
RETRAIN (reference only)	0.964 ±0.003	0.956 ±0.002	0.931 ±0.005
GRADASCENT	0.555 ±0.066	0.501 ±0.020	0.700 ±0.025
D2D	0.500 ±0.000	0.500 ±0.000	0.500 ±0.000
GRAPHERASER	0.527 ±0.002	0.538 ±0.013	0.517 ±0.009
GRAPHEDITOR	0.776 ±0.025	0.776 ±0.025	0.776 ±0.025
GNNDELETE	0.934 ±0.002	0.914 ±0.007	0.897 ±0.006

Table 19: Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: graph classification. Metric: AUC (↑)

Model	Acc on GCN	Acc on GAT	Acc on GIN
RETRAIN (reference only)	0.758 ±0.068	0.763 ±0.049	0.769 ±0.047
GRADASCENT	0.601 ±0.063	0.612 ±0.053	0.613 ±0.057
D2D	0.572 ±0.013	0.546 ±0.010	0.591 ±0.021
GRAPHERASER	0.596 ±0.032	0.611 ±0.029	0.607 ±0.027
GRAPHEDITOR	0.613 ±0.035	0.624 ±0.028	0.620 ±0.027
GNNDELETE	0.710 ±0.041	0.723 ±0.038	0.733 ±0.030

Table 20: Deletion task: 100 node deletion on DBLP. Evaluated downstream task: link prediction. Metric: AUC (↑). GNN architecture: GCN.

Model	Acc
RETRAIN (reference only)	0.973 ±0.002
GRADASCENT	0.571 ±0.032
D2D	0.507 ±0.002
GRAPHERASER	0.513 ±0.004
GRAPHEDITOR	0.697 ±0.031
GNNDELETE	0.938 ±0.004

Table 21: Deletion task: 2.5% node feature update on DBLP. Evaluated downstream task: node classification, link prediction, and graph classification. GNN architecture: GCN

Model	Acc of node classification	AUC of link prediction	AUC of graph classification
RETRAIN (reference only)	0.810 ±0.021	0.897 ±0.025	0.753 ±0.042
GRADASCENT	0.614 ±0.042	0.655 ±0.030	0.623 ±0.046
D2D	0.525 ±0.009	0.504 ±0.003	0.586 ±0.017
GRAPHERASER	0.682 ±0.044	0.579 ±0.021	0.592 ±0.032
GRAPHEDITOR	0.711 ±0.039	0.683 ±0.031	0.630 ±0.026
GNNDELETE	0.782 ±0.027	0.850 ±0.027	0.724 ±0.027

Table 22: Deletion task: 2.5% sequential edge deletion, with a batch size of 0.5%, on DBLP. Evaluated downstream task: link prediction. Evaluation metric: AUC (↑). GNN architecture: GCN.

Ratio (%)	AUC on $\mathcal{E}_t$	AUC on $\mathcal{E}_d$
0.5	0.951	0.829
1.0	0.949	0.808
1.5	0.943	0.791
2.0	0.938	0.776
2.5	0.934	0.748

Table 23: Comparison of Graph Unlearning and Dynamic Network Embedding (DNE). Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: link prediction. Evaluation metric: AUC (↑).

Model	AUC on $\mathcal{E}_t$	AUC on $\mathcal{E}_d$
RETRAIN (reference only)	0.964 ±0.003	0.506 ±0.013
JODIE (DNE)	0.801 ±0.042	0.613 ±0.026
GNNDELETE (Unlearning)	0.934 ±0.002	0.748 ±0.006

Table 24: Deletion task: 2.5% edge deletion on DBLP. Evaluated downstream task: outlier detection. GNN architecture: GCN.

Model	Percentage of edges that are classified as regular edges, i.e., non-outliers (↑)
RETRAIN (reference only)	0.746
GRADASCENT	0.503
D2D	0.517
GRAPHERASER	0.563
GRAPHEDITOR	0.642
GNNDELETE	0.710