

Article

Graph Attributed Unlearning via Propagation Suppression and Knowledge Dissipation

Zhiyu Chen ¹, Jiaquan Liang ¹, Qi Luo ^{2,*} and Zhipeng Cai ³

¹ Department of Computer Technology, Changchun University of Technology, Changchun 130012, China; chenzhiyu@ccut.edu.cn (Z.C.); jqliang@stu.ccut.edu.cn (J.L.)

² School of Computer Science and Technology, Shandong University, Qingdao 266237, China

³ Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA; zcai@gsu.edu

* Correspondence: luqi2018@mail.sdu.edu.cn

Abstract

With the growing global emphasis on data privacy protection, particularly the enforcement of the “right to be forgotten” under the GDPR, effectively deleting private information from models has become an urgent challenge. Graph-structured data presents a particularly challenging unlearning scenario due to its non-Euclidean nature and strong relational dependencies, which are prevalent in real-world applications such as social and recommendation systems. To address this issue, graph unlearning has been introduced to eliminate the influence of deleted data on models while preserving their overall performance. The effectiveness of graph unlearning is typically evaluated by three key metrics: model performance, unlearning efficiency, and robustness against membership inference attacks, which together determine the overall quality of an unlearning method. Existing graph unlearning methods fall into exact and approximate regimes. Most studies focus on edge/node-level unlearning, and existing attempts at feature-level unlearning remain limited. Exact unlearning methods that adopt the SISA partition and retraining paradigm may inadvertently reintroduce the features intended to be unlearned during the aggregation phase, thereby leading to incomplete unlearning. Approximate methods, on the other hand, often incur excessive information loss in feature-level removal, which degrades predictive accuracy. Accordingly, we propose a graph unlearning framework specifically designed for feature-level unlearning, consisting of two main stages. In the first stage, we zero out the features of the unlearned nodes at each layer to block their propagation through the GNN, thereby reducing their influence on neighboring node representations. In the second stage, we induce misclassification of the unlearned nodes to progressively degrade model representations and learned knowledge associated with them, enabling more thorough feature-level unlearning. Experiments on multiple graph datasets and models demonstrate that our method achieves favorable overall unlearning performance in most settings, offering a balanced trade-off between accuracy, unlearning efficiency, and unlearning effectiveness.



Academic Editor: Jonathan Blackledge

Received: 15 June 2026

Revised: 15 July 2026

Accepted: 20 July 2026

Published: 24 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: graph unlearning; feature-level unlearning; graph neural networks

MSC: 68T07

1. Introduction

The rapid advancement of artificial intelligence (AI) has intensified concerns over data privacy. As AI models rely heavily on large-scale user data, ensuring their secure and lawful use has become a critical challenge. To address these risks, many regions and countries have

introduced stringent data protection regulations, including the EU GDPR [1], California CPRA [2], Brazil LGPD [3], and India DPDP [4]. Among these regulations, the “right to be forgotten” under the GDPR introduces a unique requirement: individuals may request the removal of their personal data from already trained models. As retraining models from scratch upon each request is often computationally prohibitive, machine unlearning has emerged as a practical mechanism to remove the influence of specific data from trained models while maintaining their utility [5–7].

Graph data [8–10] has recently attracted increasing attention [11] for its ability to model complex systems. Building on this representation, graph neural networks (GNNs) [12–16] have demonstrated remarkable effectiveness in learning from complex relational structures across diverse domains [17–19]. Consequently, graph data has been widely applied in social networks, recommender systems, and knowledge graphs [13,20–24], as well as in tasks such as node classification [25–27] and link prediction [28–30]. However, unlike Euclidean data such as images or tabular records, graph data is inherently non-Euclidean and exhibits complex structural dependencies among nodes and edges [31,32]. These dependencies introduce unique challenges for unlearning, as information associated with a single node may propagate through multi-hop message passing and influence distant parts of the graph.

To address these challenges, recent studies have proposed graph unlearning methods that aim to remove the influence of designated training data from learned GNN models while preserving their utility. Existing graph unlearning methods can be broadly categorized into two types according to their implementation mechanisms: exact unlearning and approximate unlearning. Exact unlearning [33–35] completely removes the target data from the training set and retrains the model from scratch, typically following the SISA (Sharded, Isolated, Sliced, and Aggregated) framework. This approach partitions the original graph into subgraphs, retrains only those containing the data to be forgotten, and aggregates the results, with different methods mainly varying in subgraph partitioning and aggregation strategies. On the other hand, approximate unlearning [36–44] aims to strike a balance between unlearning quality and computational efficiency, thereby avoiding the expensive retraining cost required by exact unlearning.

Feature-level unlearning is essential for fine-grained privacy control in graph applications. For example, in social networks, edge unlearning corresponds to removing relationships, node unlearning corresponds to deleting accounts, whereas feature unlearning involves removing posts or profile attributes while preserving social connections. Feature-level unlearning may target either specific attributes or the entire feature vector of a node. Due to the strong correlation between node attributes and graph topology in real-world graphs, removing all features of a node is substantially more challenging than removing a single attribute.

In this work, we focus on full-feature unlearning, where the entire feature vector of a target node is removed. Full-feature unlearning poses several challenges for graph representation learning. First, removing all features alters the semantic position of a node in the embedding space, potentially breaking consistency with its neighbors. Second, during message passing, feature-depleted nodes may propagate weak or misleading signals, leading to representation drift in neighboring nodes. Third, node information can propagate through multi-hop aggregation, making residual feature influence difficult to eliminate completely. These factors make full-feature unlearning a more challenging and privacy-sensitive problem than unlearning individual attributes. When applied to feature-level unlearning, existing graph unlearning methods exhibit notable limitations. Exact unlearning approaches rely on subgraph retraining, which may disrupt structural information and allow residual feature influence to re-enter through message passing from

neighboring nodes. Approximate unlearning approaches modify model parameters or representations to suppress specific information, but may inadvertently remove critical knowledge and degrade overall model performance. To contextualize our contributions, we summarize several representative graph unlearning methods in Table 1. As highlighted in the table, most existing methods treat features as an erasable unit equivalent to nodes or edges. In the table, “×” indicates that the corresponding method does not incorporate a feature-level unlearning mechanism. Their original design does not account for the unique issues arising from full-feature deletion, namely the multi-hop propagation and residual effects of feature information through graph topology. This research gap renders existing methods lacking in targeted mechanisms when confronted with full-feature unlearning demands. These limitations motivate the need for unlearning mechanisms that explicitly prevent feature-induced information propagation while preserving retained knowledge.

Table 1. Comparison of representative graph unlearning methods.

Method	Unlearning Type	Task Type	Feature-Level Support
GraphEraser [33]	Exact	All	Partial
GraphRevoker [34]	Exact	Node	×
GIF [36]	Approximate	All	Partial
GNNDelete [37]	Approximate	All	Partial
Node-CUL [38]	Approximate	Node	×
Ours	Approximate	Feature	Full

To address these challenges, we propose a feature-level unlearning framework for graph neural networks, named **FUSED-GNN** (Feature-level Unlearning via Suppression and Dissipation in GNNs). The framework consists of two main components. First, we introduce *Propagation Suppression*, which explicitly blocks feature-induced message passing by setting the features of unlearned nodes to zero at each GNN layer, preventing their information from propagating to neighboring nodes. Second, we propose *Knowledge Dissipation*, which adjusts the post-training output distribution of unlearned nodes to erase their classification-related knowledge and ensure effective unlearning. To mitigate unintended loss of retained knowledge, we further perform a limited number of retraining iterations on the remaining nodes.

The main contributions of this work are summarized as follows:

1. We study full-feature unlearning in graph neural networks and propose a feature-level unlearning framework that enables fine-grained and controllable removal of node feature information while preserving model utility.
2. We introduce two complementary mechanisms, *Propagation Suppression* and *Knowledge Dissipation*, to prevent feature-induced information propagation and to eliminate residual classification knowledge associated with unlearned nodes.
3. Extensive experiments on four benchmark datasets with three representative GNN architectures demonstrate that, compared with retraining as the reference, our method achieves up to a 7% improvement in test accuracy (on Citeseer with GAT), while significantly reducing unlearning time and maintaining AUC values close to the ideal 0.5, indicating effective data unlearning without sacrificing model utility.

2. Related Work

2.1. Machine Unlearning

Machine unlearning has emerged from the growing need to enable data deletion in machine learning systems, particularly those trained on Euclidean data such as images, tabular records, and textual features. Based on the paradigms of exact and approximate

unlearning [45], various methods have been proposed for data unlearning in machine learning models. Within the scope of exact unlearning, several notable techniques [46–48] have been proposed, among which the SISA framework by Bourtole et al. [46] stands as a representative and influential example. SISA improves unlearning efficiency by partitioning the training data into independent subsets and enabling localized retraining through staged learning, thereby reducing computational cost and supporting verifiable data removal. Research on approximate unlearning primarily focuses on enhancing computational efficiency and improving applicability to modern, large-scale models. Representative approaches in this line include [49–53]. Tarun et al. [51] proposed UNSIR, a rapid unlearning framework that combines error-amplifying noise with an “impair–repair” weight-adjustment process to achieve efficient data removal. Several recent surveys provide comprehensive overviews of machine unlearning, outlining its theoretical foundations, methodological categorizations, and practical challenges across diverse application settings [54–57]. As summarized above, exact unlearning methods are well suited for settings that demand strict regulatory guarantees, while approximate unlearning methods offer greater practicality in latency-sensitive or resource-constrained environments.

2.2. Graph Unlearning

Similarly, graph unlearning methods have experienced rapid development in recent years and can be categorized into two paradigms: exact unlearning and approximate unlearning. Recent research on exact unlearning methods includes [33–35,39,40]. Chen et al. [33] proposed GraphEraser, a pioneering GNN unlearning framework that employs balanced partitioning and shard-specific retraining to reduce deletion cost. Wang et al. [35] introduced GUIDE, which achieves efficient inductive unlearning through guided graph partitioning and subgraph repairing. Zhang et al. [34] proposed GraphRevoker, which performs property-aware sharding and contrastive submodel aggregation to enable efficient partial retraining. For approximate unlearning, a variety of approaches have been explored in recent studies [36–38,41–44]. Wu et al. [36] proposed Graph Influence Function (GIF), which leverages structural neighbor dependencies to approximate parameter updates and enable efficient unlearning. Lee et al. [38] developed Node-CUL, a node-level unlearning method that adjusts embeddings of forgotten nodes and reconstructs neighborhoods to maintain model utility. Ding et al. [44] introduced AGU, a task-adaptive framework employing GNN-specific unlearning modules and neighbor selection for targeted element removal. Guan et al. [58] proposed three fine-tuning-based node unlearning methods—CLR, TNMPP, and CNNF—that leverage label replacement and graph topology to efficiently remove sensitive nodes while preserving model utility. Moreover, Zhang et al. [59] and Guo et al. [60] extended graph unlearning to dynamic and spatio-temporal graphs, respectively. Following these developments, comprehensive surveys have been conducted to summarize recent progress, among which Said et al. [61] provided an extensive overview of graph unlearning techniques. While prior studies have explored feature-level unlearning to a limited extent, our work distinguishes itself by focusing on fine-grained and reliable feature-level unlearning in graph neural networks.

3. Preliminaries

3.1. Graph Neural Networks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ denote a graph, where $\mathcal{V} = \{v_1, v_2, \dots, v_{|\mathcal{V}|}\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the node feature matrix, with each node v_i associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$. Graph neural networks (GNNs) learn

node representations by iteratively aggregating information from neighboring nodes and updating each node's embedding. At the l -th layer, the aggregation step can be expressed as

$$\mathbf{m}_i^{(l)} = \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(l)} + \mathbf{x}_i, \quad (1)$$

where $\mathbf{m}_i^{(l)}$ denotes the aggregated message for node v_i , and $\mathbf{h}_j^{(l)}$ represents the hidden feature of node v_j at layer l . The subsequent update step refines the node representation through a learnable transformation:

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{W}^{(l)} \mathbf{m}_i^{(l)}), \quad (2)$$

where $\mathbf{W}^{(l)}$ is the trainable weight matrix at layer l , and $\sigma(\cdot)$ is a nonlinear activation function such as ReLU or sigmoid. By stacking multiple layers, GNNs enable nodes to capture higher-order dependencies and structural information from the graph.

3.2. Feature-Level Graph Unlearning

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denotes the node feature matrix. We define two types of feature unlearning operations on node features: attribute specific and full-feature unlearning.

3.2.1. Attribute-Specific Feature Unlearning

In this setting, only part of the feature dimensions of a target node v_u are removed. Let $\mathcal{D}_u \subseteq \{1, \dots, d\}$ denote the indices of the feature dimensions to be unlearned. The modified graph after attribute-specific unlearning can be expressed as

$$\mathcal{G}'_{\text{attr}} = (\mathcal{V}, \mathcal{E}, \mathbf{X} \setminus \mathbf{X}_{u[:, \mathcal{D}_u]}), \quad (3)$$

where $\mathbf{X}_{u[:, \mathcal{D}_u]}$ represents the removed feature dimensions of node v_u .

3.2.2. Full-Feature Unlearning

In this setting, the entire feature vector of the target node v_u is erased. The modified graph is defined as

$$\mathcal{G}'_{\text{full}} = (\mathcal{V}, \mathcal{E}, \mathbf{X} \setminus \mathbf{X}_u), \quad (4)$$

where $\mathbf{X}_u$ denotes the complete feature vector of node v_u .

Let $\mathcal{M}_{\text{re}}$ denote the retrained model on the modified graph $\mathcal{G}'$, and $\mathcal{M}_{\text{un}}$ represent the model obtained by applying a feature unlearning algorithm that approximates the behavior of $\mathcal{M}_{\text{re}}$. We expect the accuracy of the unlearned model to approach that of the retrained model; i.e.,

$$\text{Acc}(\mathcal{M}_{\text{un}}) \approx \text{Acc}(\mathcal{M}_{\text{re}}), \quad (5)$$

Meanwhile, the unlearning model should require less runtime than retraining:

$$\text{Time}(\mathcal{M}_{\text{un}}) < \text{Time}(\mathcal{M}_{\text{re}}). \quad (6)$$

Moreover, the AUC value of membership inference attacks against $\mathcal{M}_{\text{un}}$ should be as close as possible to 0.5, indicating effective unlearning.

4. Method

The proposed FUSED-GNN framework comprises two main stages, as shown in Figure 1. First, during message propagation at each graph neural network layer, we

nullify the feature vectors of target forgotten nodes to effectively block their information flow. This mechanism suppresses the feature influence of unlearned nodes layer by layer, ensuring that the model does not retain their informational traces during the unlearning process. Second, upon completion of the forward propagation, we swap the maximum ground-truth probability with the minimum misclassification probability in the predicted probability distribution of the unlearned nodes. The loss is then computed and model parameters are updated based on this perturbed distribution, thereby compelling the knowledge of these unlearned nodes to dissipate and further enhancing the elimination of their characteristic information.

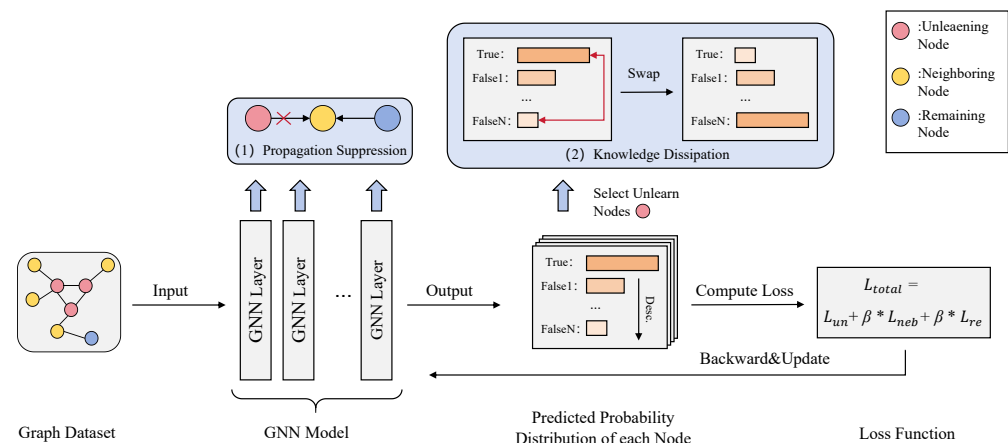


Figure 1. Illustration of the overall training pipeline of Fused-GNN, where “×” indicates the absence of message passing along the corresponding path between the nodes. The Propagation Suppression process is performed within each GNN layer during forward propagation, whereas the Knowledge Dissipation process applies probability swapping to the predicted probability distribution of the unlearned nodes. The total loss is subsequently computed and back-propagated through the network to update the model parameters.

4.1. Propagation Suppression

In the first stage of the proposed feature unlearning framework, we explicitly intervene in the message-passing mechanism of the graph neural network (GNN) to block the propagation of information from the nodes designated to be unlearned, as shown in Figure 2. This operation ensures that the representations of all non-unlearning nodes in the graph remain unaffected by the designated unlearning nodes during subsequent feature propagation. In a conventional GNN, the hidden representation of node v_i at the $(l + 1)$ -th layer is updated by aggregating information from its neighboring nodes $\mathcal{N}(i)$ as

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} a_{ij}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)} \right), \quad (7)$$

where $\mathbf{h}_i^{(l)}$ denotes the feature vector of node v_i at layer l , $a_{ij}^{(l)}$ represents the attention or normalization coefficient indicating the importance of message flow from node v_j to v_i , $\mathbf{W}^{(l)}$ is the learnable weight matrix at layer l , and $\sigma(\cdot)$ is a nonlinear activation function. To realize feature unlearning, we suppress the contribution of unlearned nodes by setting their feature representations to zero during the message-passing step of every layer. Let $\mathcal{V}_u$ denote the set of nodes to be unlearned. For any node $v_j \in \mathcal{V}_u$, we enforce the zero-feature constraint:

$$\mathbf{h}_j^{(l)} = \mathbf{0}, \quad \forall l \in \{0, 1, \dots, L\}, \quad (8)$$

where L is the total number of network layers. Substituting Equation (8) into Equation (7), the modified propagation rule becomes:

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \setminus \mathcal{V}_u} a_{ij}^{(l)} \mathbf{w}^{(l)} \mathbf{h}_j^{(l)} \right), \quad (9)$$

Through the propagation rule defined in Equation (9), the dissemination of information originating from the unlearned nodes is effectively blocked.

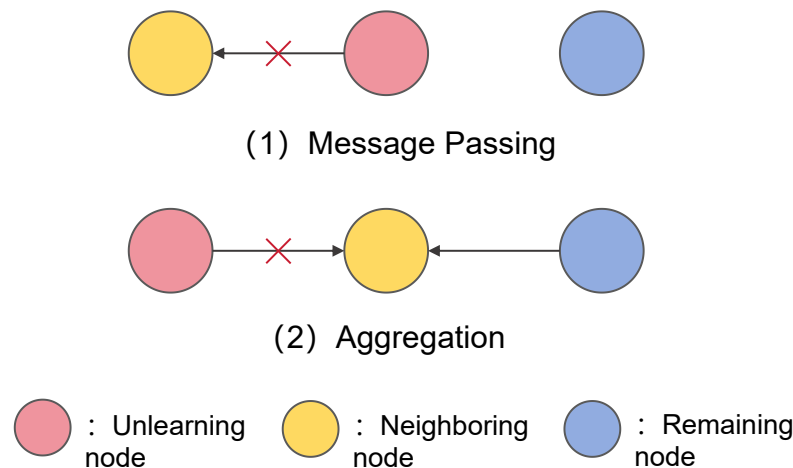


Figure 2. Illustration of the Propagation Suppression mechanism is provided, where “x” indicates the absence of message passing along the corresponding path between the nodes. By setting the features of the nodes to be unlearned to zero at every GNN layer, these nodes are prevented from transmitting messages to their neighbors during the message-passing phase. Consequently, during the aggregation phase, each neighboring node only incorporates information from other neighbors or remaining nodes that are not connected to the unlearning nodes, effectively isolating the influence of the unlearned nodes from the rest of the graph.

To guarantee a complete and deterministic suppression of information flow, we maintain an ID list that records all nodes designated for feature unlearning. During both the unlearning and inference stages, a masking operation is applied based on this ID list to nullify the corresponding node representations. The masking process is implemented as an $\mathcal{O}(1)$ vector-wise multiplication, which introduces negligible computational overhead. After the masking multiplication, the features of the forgotten nodes at each layer during training are set to zero. This ensures that when a neighbor node aggregates information from a forgotten node, the contribution of that node is completely nullified. Therefore, information isolation is achieved by construction rather than through approximation. Since the aggregation function (e.g., sum, mean, or max) operates directly on the feature vectors, setting the feature vector of the unlearned node to zero completely removes the influence of that node from the aggregation result. This lightweight design ensures that the proposed Propagation Suppression mechanism achieves complete information isolation with minimal efficiency loss, making it entirely practical for large-scale GNNs.

By performing this suppression at every layer, the proposed method guarantees layer-wise isolation of unlearned nodes, ensuring that neither their feature information nor their topological dependencies are preserved in subsequent node embeddings. This mechanism in this first stage establishes a necessary foundation for complete and consistent feature unlearning across the entire GNN.

4.2. Knowledge Dissipation

The second stage of our feature unlearning framework aims to dissipate the knowledge retained by the model on the unlearned nodes by reshaping their output probability distributions. Instead of enforcing explicit misclassification, we encourage the model to

increase the prediction uncertainty for these nodes, thereby diluting their influence on the learned decision boundary. Following the forward pass, let $\hat{y}_i$ denote the predicted probability distribution of node v_i at the final output layer, which is computed via the softmax function applied to the output logits $\mathbf{z}_i$ of the same layer:

$$\hat{y}_i = \text{softmax}(\mathbf{z}_i), \quad (10)$$

where $\mathbf{z}_i$ is the logits vector of node v_i , and $\text{softmax}(\cdot)$ transforms it into a normalized probability distribution. For each unlearned node $v_i \in \mathcal{V}_u$, we modify its probability distribution by redistributing the confidence mass between the true class and the least likely incorrect class:

$$\hat{y}_{i,\text{swap}} = \text{swap}(\hat{y}_i[y_{\text{true}}], \hat{y}_i[y_{\text{min}}]), \quad (11)$$

where y_{true} denotes the ground-truth label of node v_i , and y_{min} corresponds to the label with the lowest predicted probability. This adjustment increases the prediction entropy of unlearned nodes, weakening their class certainty and thereby promoting Knowledge Dissipation. By applying this redistribution to all unlearned nodes during optimization, the model gradually loses the discriminative information related to them, resulting in higher uncertainty and reduced representational dependency in the final predictions. Consequently, this stage complements the feature suppression step by erasing residual task-specific knowledge associated with unlearned nodes. The loss function in the framework is as follows:

(1) Unlearning Loss. For the unlearned nodes $\mathcal{V}_u$, the model intentionally misclassifies them using the swapped probability distribution $\hat{y}_{i,\text{swap}}$. The corresponding loss is defined as

$$\mathcal{L}_{\text{un}} = -\frac{1}{|\mathcal{V}_u|} \sum_{v_i \in \mathcal{V}_u} \sum_{c=1}^C \hat{y}_{i,c} \log(\hat{y}_{i,\text{swap}}). \quad (12)$$

where C denotes the total number of classes in the classification task, and c is the class index ranging from 1 to C . For each unlearned node v_i , the summation iterates over all classes to compute the loss associated with the predicted probability of each class. Here, $\hat{y}_{i,c}$ denotes the original predicted probability of the model for class c , and $\hat{y}_{i,\text{swap}}$ denotes the corresponding probability in the swapped target distribution, which is constructed by exchanging the maximum ground-truth class probability with the minimum misclassification probability in $\hat{y}_i$. It is important to note that $\hat{y}_{i,\text{swap}}$ is treated as a fixed target distribution during loss computation, and gradients do not flow through the swapping operation. The cross-entropy loss is computed between the swapped distribution $\hat{y}_{i,\text{swap}}$ and the ground-truth labels, and the gradient backpropagates through the model parameters, encouraging the model to adjust its predictions toward the swapped target distribution.

(2) Neighbor and Remaining Node Losses. To preserve both local and global predictive performance, we apply standard cross-entropy losses on the neighboring nodes $\mathcal{V}_n$ and remaining nodes $\mathcal{V}_r$:

$$\begin{aligned} \mathcal{L}_{\text{neb}} &= -\frac{1}{|\mathcal{V}_n|} \sum_{v_i \in \mathcal{V}_n} \sum_{c=1}^C y_{i,c} \log \hat{y}_{i,c}, \\ \mathcal{L}_{\text{re}} &= -\frac{1}{|\mathcal{V}_r|} \sum_{v_i \in \mathcal{V}_r} \sum_{c=1}^C y_{i,c} \log \hat{y}_{i,c}. \end{aligned} \quad (13)$$

The composite objective Equation (14) thus enforces effective unlearning for the unlearning node set $\mathcal{V}_u$ while maintaining consistent and reliable predictions across the neighbor node set $\mathcal{V}_n$ and the remaining node set $\mathcal{V}_r$.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{un}} + \beta * \mathcal{L}_{\text{neb}} + \beta * \mathcal{L}_{\text{re}}. \quad (14)$$

5. Experiments

5.1. Experimental Setup

5.1.1. Datasets and Models

We conduct experiments on four commonly used citation network datasets: Cora-ML, CiteSeer, PubMed, and CS. Cora-ML contains machine-learning related publications categorized into seven classes, while CiteSeer comprises scientific papers from six fields with relatively sparse citation links. PubMed consists of biomedical articles on diabetes with a large number of nodes and three classes, and CS is extracted from the Microsoft Academic Graph and includes computer science papers labeled by their research areas. The statistical details of these datasets are summarized in Table 2. To evaluate the performance of our method on these datasets, we employ Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Graph Isomorphism Networks (GINs) as baseline models.

Table 2. Summary of datasets used in the experiments.

Dataset	Nodes	Edges	Features	Classes
Cora-ML	2485	5278	1433	7
CiteSeer	3312	4660	3703	6
PubMed	19,717	44,338	500	3
CS	18,333	81,894	3703	6

5.1.2. Comparison Methods

We include two unlearning methods as baseline approaches to assess the effectiveness of our proposed unlearning framework. (1) **Retrain** performs complete retraining of the GNN model on the remaining nodes only, functioning as the ideal reference of a perfectly unlearned model for assessing both unlearning effectiveness and model utility. (2) **Gradient Ascent** approximates unlearning by performing a small number of gradient-ascent updates on the loss of the to-be-unlearned samples (nodes/features), deliberately increasing their loss to weaken their influence on the model parameters. (3) **GIF** estimates model parameter updates through graph-aware influence functions that account for both the target nodes and their structurally affected neighbors, enabling efficient and effective approximate unlearning. (4) **Node-CUL** (Node-level Contrastive Unlearning) removes the influence of target nodes by directly optimizing their embeddings through a contrastive learning objective. It pushes the embeddings of unlearning nodes away from those of their same-class neighbors while pulling them toward remaining nodes of different classes, making them indistinguishable from unseen nodes. Furthermore, a neighborhood reconstruction mechanism is introduced to adjust the embeddings of neighboring nodes and preserve model utility during unlearning. Since the frameworks of exact and approximate unlearning methods differ fundamentally in their mechanisms and implementation, we only adopt approximate unlearning methods for comparison. Specifically, exact methods such as SISA are designed for node-level unlearning, and their data sharding or model ensemble strategies incur substantial storage and computational overhead, whereas our work focuses on efficiency in approximate feature unlearning. We therefore do not include such methods in our main experiments.

5.1.3. Evaluation Metrics

(1) **Model performance.** We evaluate the test accuracy of $\mathcal{V}_{ts}$ from the unlearned models to assess how well the model preserves its predictive capability after unlearning. (2) **Efficiency.** We measure the runtime of each unlearning framework to evaluate the computational efficiency of the unlearning process. (3) **AUC.** To further verify the unlearning effectiveness, we perform a membership inference attack (MIA) using the Likelihood

Ratio Attack (LiRA) [62]. We label all unlearning nodes as members and randomly select the same number of test nodes as non-members, then train multiple shadow models to estimate the membership likelihood. We report the AUC value, where a value close to 0.5 indicates that the model can hardly distinguish unlearned nodes from unseen ones, implying successful unlearning.

5.1.4. Parameter Settings

All models are trained with the same general hyperparameter configuration for fair comparison. The base GNN models are trained for 30 epochs, while the retraining-based method is trained for 300 epochs to ensure complete convergence. For the Gradient Ascent method, we perform 10 ascent steps on Cora and Citeseer, and 5 ascent steps on PubMed and CS. The GIF method is set with 30 iterations, a damping coefficient of 0.1, a scaling factor of 50.0, and a learning rate α of 0.05. The Node-CUL framework is trained for 30 epochs. Our proposed method is trained for 10 epochs on the four datasets, respectively.

5.1.5. Implementation Details

All experiments are conducted on a server equipped with two NVIDIA RTX 3090 GPUs, an Intel(R) Xeon(R) Silver 4314 CPU @ 2.40 GHz, and 128 GB RAM. The operating system is Ubuntu 22.04. All models are implemented using Python 3.9 and PyTorch 1.12.

5.2. Experimental Results of Feature Unlearning

In this section, we present the experimental results of feature unlearning to comprehensively evaluate the effectiveness of the proposed framework. The experiments are analyzed from three key perspectives of feature unlearning: (1) **unlearning model performance**: the classification accuracy of the model on the test set after the unlearning operation, which reflects its ability to retain useful knowledge; (2) **unlearning efficiency**: the comparison of unlearning time between the proposed method and baseline unlearning frameworks, which reflects the time overhead of the unlearning operation; and (3) **unlearning effectiveness**: based on the results of membership inference attacks conducted on the unlearned model, we evaluate its unlearning performance and privacy protection capability. As an approximate unlearning method, our approach is empirically validated through experiments on model utility, efficiency, and privacy, rather than through certified theoretical guarantees. Taking these three aspects into comprehensive consideration, we first provide an overview of the experimental outcomes to highlight the overall effectiveness of the proposed framework. The experiments are conducted on four benchmark graph datasets using three representative graph neural network architectures. The results show that our method consistently achieves strong performance, high efficiency, and robust privacy protection in feature unlearning scenarios. The results are summarized in Table 3. Bold values indicate the best performance under the current settings. Arrows indicate the direction of preference for each metric: $\uparrow$ means that higher values are better, $\downarrow$ means that lower values are better, and $\rightarrow 50$ means that values closer to 50 are preferred. The same conventions for bold and arrows apply to all other tables throughout the paper.

Table 3. Result of different unlearning method experiments.

Dataset	Method	Test Accuracy (%) $\uparrow$			Unlearn Time (s) $\downarrow$			AUC (%) $\rightarrow$ 50		
		GCN	GAT	GIN	GCN	GAT	GIN	GCN	GAT	GIN
Cora-ML	Retrain (Reference)	84.44 $\pm$ 0.37	82.96 $\pm$ 0.58	83.48 $\pm$ 1.12	50.54 $\pm$ 2.33	63.14 $\pm$ 1.90	41.03 $\pm$ 1.65	66.05	65.50	66.08
	Gradient Ascent	84.96 $\pm$ 0.42	86.51 $\pm$ 0.67	85.48 $\pm$ 0.60	9.08 $\pm$ 0.48	9.47 $\pm$ 0.50	8.99 $\pm$ 0.33	62.86	54.43	58.77
	GIF	84.81 $\pm$ 0.90	84.96 $\pm$ 0.20	86.18 $\pm$ 0.24	17.62 $\pm$ 0.57	37.05 $\pm$ 1.12	15.44 $\pm$ 0.60	69.39	66.82	67.77
	Node-CUL	86.81 $\pm$ 1.18	86.37 $\pm$ 0.60	85.48 $\pm$ 0.99	33.31 $\pm$ 2.21	38.04 $\pm$ 3.73	32.38 $\pm$ 1.76	48.98	47.42	47.30
	Ours	87.18 $\pm$ 1.21	87.03 $\pm$ 0.97	86.07 $\pm$ 0.61	12.81 $\pm$ 0.13	13.28 $\pm$ 0.11	12.51 $\pm$ 0.45	53.83	52.55	51.48
PubMed	Retrain (Reference)	87.62 $\pm$ 0.26	87.53 $\pm$ 0.15	87.01 $\pm$ 0.27	334.16 $\pm$ 19.06	415.94 $\pm$ 14.82	269.13 $\pm$ 16.37	59.20	61.27	63.10
	Gradient Ascent	86.35 $\pm$ 0.90	86.59 $\pm$ 0.46	66.98 $\pm$ 19.99	75.28 $\pm$ 1.84	74.60 $\pm$ 1.95	74.66 $\pm$ 1.55	51.63	48.48	50.84
	GIF	85.53 $\pm$ 0.19	84.32 $\pm$ 0.05	87.58 $\pm$ 0.05	98.42 $\pm$ 4.03	205.37 $\pm$ 11.45	85.08 $\pm$ 6.61	61.75	62.66	64.83
	Node-CUL	86.51 $\pm$ 0.54	87.23 $\pm$ 0.36	74.81 $\pm$ 6.74	497.51 $\pm$ 8.94	506.39 $\pm$ 9.03	490.81 $\pm$ 14.43	49.88	47.30	50.19
	Ours	87.18 $\pm$ 0.41	87.42 $\pm$ 0.53	85.33 $\pm$ 0.52	173.21 $\pm$ 6.39	168.40 $\pm$ 5.61	167.33 $\pm$ 5.36	49.91	48.68	50.77
Citeseer	Retrain (Reference)	73.91 $\pm$ 1.01	70.66 $\pm$ 0.86	74.03 $\pm$ 0.49	65.67 $\pm$ 3.75	83.30 $\pm$ 4.61	55.12 $\pm$ 3.01	70.84	66.37	68.31
	Gradient Ascent	77.22 $\pm$ 0.78	77.59 $\pm$ 0.54	76.44 $\pm$ 0.25	9.15 $\pm$ 0.83	9.64 $\pm$ 0.81	9.08 $\pm$ 0.82	51.16	51.12	59.15
	GIF	77.10 $\pm$ 0.30	76.32 $\pm$ 0.50	75.55 $\pm$ 1.96	21.06 $\pm$ 2.18	45.58 $\pm$ 1.34	18.83 $\pm$ 0.48	68.93	66.85	74.00
	Node-CUL	76.74 $\pm$ 1.15	75.48 $\pm$ 1.17	75.84 $\pm$ 0.83	35.57 $\pm$ 0.60	43.80 $\pm$ 4.67	33.84 $\pm$ 2.24	51.35	52.22	51.94
	Ours	78.79 $\pm$ 0.26	78.19 $\pm$ 0.34	76.80 $\pm$ 0.99	13.08 $\pm$ 1.29	14.01 $\pm$ 1.10	13.25 $\pm$ 0.14	50.02	50.57	50.77
CS	Retrain (Reference)	91.10 $\pm$ 0.29	91.39 $\pm$ 0.37	90.84 $\pm$ 0.38	2187.61 $\pm$ 63.95	2380.33 $\pm$ 98.30	2193.53 $\pm$ 83.52	52.72	54.44	52.99
	Gradient Ascent	89.73 $\pm$ 0.25	93.33 $\pm$ 0.27	77.18 $\pm$ 1.86	178.22 $\pm$ 3.56	179.26 $\pm$ 5.95	176.13 $\pm$ 4.37	50.01	51.34	49.36
	GIF	90.81 $\pm$ 0.20	93.62 $\pm$ 0.07	84.17 $\pm$ 0.03	97.36 $\pm$ 1.48	126.08 $\pm$ 20.87	128.89 $\pm$ 2.87	51.60	55.49	51.83
	Node-CUL	90.23 $\pm$ 0.20	94.82 $\pm$ 0.16	77.95 $\pm$ 5.55	1480.66 $\pm$ 27.44	1499.38 $\pm$ 37.61	1482.03 $\pm$ 29.19	48.73	48.13	49.78
	Ours	89.17 $\pm$ 0.32	93.46 $\pm$ 0.31	79.89 $\pm$ 0.86	433.03 $\pm$ 9.14	429.23 $\pm$ 6.09	430.38 $\pm$ 11.69	51.41	52.32	52.32

5.2.1. Unlearning Model Performance

This experiment aims to evaluate how feature unlearning affects the generalization ability of the model, with a particular focus on the overall classification performance after unlearning. We report the test accuracy of different unlearning methods across multiple datasets and network architectures. As a reference baseline, the Retrain method consistently achieves strong and stable performance across all settings. For example, it attains 84.44% on Cora-ML, 87.62% on PubMed, and 91.39% on CS (GAT), confirming that retraining from scratch can effectively restore model accuracy even after partial data removal. The Gradient Ascent method maintains reasonable test accuracy in most cases. On Cora-ML with GCN, it reaches 84.96%, but the performance drops substantially in more challenging scenarios such as PubMed with GIN, where the accuracy declines to 66.98%. This indicates that unlearning by directly maximizing the target loss may introduce overly strong perturbations and consequently harm generalization. The GIF method achieves competitive performance and even attains the highest accuracy in several settings, such as 87.58% on PubMed with GIN and 90.81% on CS with GCN. However, its performance fluctuates across different datasets and models, suggesting sensitivity to graph structure. The Node-CUL method shows stable performance in some cases. On Cora-ML with GCN, it achieves 86.81%, but its performance becomes unstable on PubMed, especially when using GIN, where the accuracy drops to 74.81%. This indicates that the method has limited robustness to certain network architectures. Overall, the proposed method demonstrates favorable performance across most datasets and architectures. Our approach achieves competitive accuracy in most settings, such as 87.18% on Cora-ML (GCN), 87.42% on PubMed (GAT), and 78.79% on CiteSeer (GCN), while maintaining close performance to the best baseline elsewhere. We note that in several cases the observed performance gaps are small and may not be statistically significant. In most cases where our method does not achieve the best performance, the accuracy gap compared to the strongest baseline remains within 2%. However, the proposed method exhibits relatively limited overall performance on the CS dataset. This is mainly attributed to the large-scale and highly connected citation graph of CS, where dense message passing amplifies feature dependencies among nodes, making the model more reliant on global structural information. Under such conditions, the proposed feature-nullification mechanism effectively achieves the unlearning objective. However, it also weakens the representational capability of the model in highly coupled graph structures, which leads to a slight reduction in overall classification accuracy. Notably, on CS and CiteSeer, the performance gap between several unlearning methods and Retrain becomes more pronounced when using GIN. This can be attributed to the high sensitivity

of GIN to feature perturbations and local neighborhood changes, which are especially evident on large, densely connected graphs such as CS and sparsely connected graphs like CiteSeer.

5.2.2. Unlearning Efficiency

This experiment aims to evaluate the efficiency of different feature unlearning methods in terms of processing time. Based on the experimental results, the Retrain method consistently requires the most time across all datasets, as it involves retraining the model from scratch with multiple full training iterations. The Gradient Ascent method tends to exhibit a sharp decline in model performance when the number of ascent steps increases. Therefore, a smaller number of training iterations is adopted in practice, which makes the method relatively time-efficient. The GIF method is the most efficient on large-scale datasets, since it updates model parameters through influence function computation without additional training iterations. The Node-CUL method follows, as it relies on iterative optimization and the computation of contrastive losses, resulting in a relatively high time cost that remains lower than that of Retrain across most settings. Our proposed method incurs the lowest time cost on smaller datasets, benefiting from a limited number of unlearning optimization steps. On larger datasets, it remains significantly more efficient than Retrain and Node-CUL, while achieving competitive performance, although its time cost is generally higher than that of GIF. These results highlight the time efficiency characteristics of different unlearning strategies.

5.2.3. Membership Inference Attack Results

This experiment aims to evaluate the resistance of different unlearning methods against membership inference attacks. In this paper, we adopt the LiRA attack [62] to compute the AUC of each unlearning method, and the standard deviation is omitted. A successful unlearned model should exhibit an attack AUC value close to 0.5, indicating that the attacker cannot reliably distinguish whether a node has participated in training. Based on the experimental results, we observe that in most feature unlearning experiments, the Retrain method yields relatively higher AUC values. This can be attributed to the fact that retraining requires multiple full training iterations, during which the features of unlearned nodes may be partially recovered through the neighborhood aggregation process in GNN layers. Similarly, the GIF method also exhibits higher AUC values in some experiments due to incomplete removal of node features. The Gradient Ascent method achieves attack AUC values close to 0.5 in most settings, indicating strong resistance to membership inference. Nevertheless, it produces overly high AUC values in a few cases, implying incomplete unlearning and potential unlearning failure under those configurations. In contrast, both the Node-CUL method and our proposed approach consistently achieve AUC values close to 0.5 across all experiments, indicating effective resistance to membership inference attacks. These results suggest that thorough feature removal further reduces node identifiability and enhances privacy protection.

Overall, the results across model performance, unlearning efficiency, and membership inference robustness demonstrate that our method achieves a superior balance between utility, efficiency, and privacy compared with all baseline approaches. It consistently maintains strong and stable accuracy while keeping the unlearning cost low, and achieves attack AUC values close to 0.5 across all settings, indicating effective privacy protection. Notably, in a few graph-oriented experiments, our approach does not push the AUC as close to 0.5 as node-level unlearning methods such as Node-CUL. This deviation, however, is intentional rather than a sign of inferior unlearning. In graph data, the so-called “Neighborhood Effect” implies structural dependency among adjacent nodes. Since their representations are

entangled through message passing and shared embeddings, enforcing perfect unlearning on a target node ($AUC = 0.5$) can induce severe collateral performance degradation on its neighbors. Consequently, aggressively enforcing an AUC of 0.5 tends to harm the utility of neighboring regions in the graph. Our method is therefore designed to approach the ideal privacy protection target as closely as possible while preserving overall model utility. It achieves a favorable trade-off between model performance, unlearning efficiency, and unlearning effectiveness, rather than blindly forcing $AUC = 0.5$ at the cost of model usability. We also observe that GIN, which integrates neighborhood information via sum-based aggregation, is most sensitive to changes in neighborhood structure. On large-scale graphs such as CS, where neighborhood sizes vary significantly, the aggregation shift caused by feature zeroing becomes more pronounced, resulting in slightly lower model accuracy compared to the baseline methods. Compared with retraining-based, influence-based, and optimization-based alternatives (e.g., GA and Node-CUL), our approach avoids the high computational overhead of full retraining and mitigates the instability or incomplete unlearning observed in certain baselines. These advantages collectively lead to consistently strong overall performance across all experiments.

5.3. Experimental Results of the Ablation Study

In this subsection, we present the results of the ablation study to evaluate the contribution of each stage in our proposed unlearning framework. In addition, we also experiment with an alternative approach that modifies the model output distribution, denoted as “Ratio” in the table. Specifically, we divide the predicted probability of the ground-truth label of the forgetting nodes by an integer and then renormalize the overall distribution, thereby reducing the probability mass assigned to the true label. The results are summarized in Table 4. Experiments are conducted on the Cora-ML and Citeseer datasets using three backbone architectures, namely GCN, GAT, and GIN. As shown in Table 4, the complete two-stage framework consistently delivers the highest test accuracy across all settings, demonstrating the effectiveness of the joint design. When either stage is removed, the performance drops notably, revealing that both components contribute complementary benefits to the overall unlearning process. Meanwhile, the unlearning time and privacy-related AUC values remain comparable among all variants, indicating that the utility improvement of the full model is achieved without increasing computational cost or compromising the unlearning effect. In addition, for the alternative “Ratio” approach, we evaluate two reduction factors, namely Ratio = 2 and Ratio = 4. The results show that although this method achieves test accuracy comparable to FUSED-GNN after unlearning, the membership inference attack results are generally higher, indicating that simply reducing the predicted probability of the true label fails to effectively unlearn the target data. These results confirm that the two stages work synergistically to achieve a better trade-off among model utility, efficiency, and privacy protection.

Table 4. Results of the ablation study for the proposed unlearning framework.

Dataset	Method	Test Accuracy (%) $\uparrow$			Unlearn Time (s) $\downarrow$			AUC (%) $\rightarrow 50$		
		GCN	GAT	GIN	GCN	GAT	GIN	GCN	GAT	GIN
Cora-ML	Ours	87.18 $\pm$ 1.21	87.03 $\pm$ 0.97	86.07 $\pm$ 0.61	12.81 $\pm$ 0.13	13.28 $\pm$ 0.11	12.51 $\pm$ 0.45	53.88	52.55	51.48
	w/o Stage 1	85.77 $\pm$ 1.06	86.22 $\pm$ 0.88	85.11 $\pm$ 1.02	12.75 $\pm$ 0.45	12.58 $\pm$ 0.66	12.70 $\pm$ 0.17	54.04	55.30	53.90
	w/o Stage 2	82.25 $\pm$ 1.78	86.44 $\pm$ 0.61	84.22 $\pm$ 1.97	10.04 $\pm$ 0.52	11.19 $\pm$ 0.67	9.29 $\pm$ 0.77	53.85	54.40	54.87
	Ratio = 2	86.07 $\pm$ 0.77	86.37 $\pm$ 0.55	84.52 $\pm$ 1.06	13.19 $\pm$ 0.21	13.97 $\pm$ 0.71	13.58 $\pm$ 0.58	60.36	62.91	67.51
	Ratio = 4	86.22 $\pm$ 0.92	86.51 $\pm$ 0.62	82.89 $\pm$ 1.71	13.31 $\pm$ 0.23	14.51 $\pm$ 1.21	13.58 $\pm$ 0.37	60.18	63.11	67.90
Citeseer	Ours	78.79 $\pm$ 0.26	78.19 $\pm$ 0.34	76.80 $\pm$ 0.99	13.08 $\pm$ 1.29	14.01 $\pm$ 1.10	13.25 $\pm$ 0.14	50.02	50.57	50.77
	w/o Stage 1	77.47 $\pm$ 0.33	77.28 $\pm$ 0.34	75.78 $\pm$ 1.78	13.64 $\pm$ 0.15	14.38 $\pm$ 0.44	11.96 $\pm$ 1.55	50.52	50.15	51.93
	w/o Stage 2	78.49 $\pm$ 0.45	77.40 $\pm$ 0.76	75.84 $\pm$ 1.58	10.19 $\pm$ 0.41	10.59 $\pm$ 1.00	10.05 $\pm$ 0.52	49.93	50.59	51.55
	Ratio = 2	78.49 $\pm$ 0.34	76.63 $\pm$ 0.59	76.39 $\pm$ 0.81	13.96 $\pm$ 0.23	14.49 $\pm$ 0.42	13.91 $\pm$ 0.34	67.74	62.02	71.30
	Ratio = 4	78.86 $\pm$ 0.25	77.41 $\pm$ 0.64	76.45 $\pm$ 0.65	14.18 $\pm$ 0.73	15.04 $\pm$ 0.45	13.81 $\pm$ 0.50	67.60	61.19	71.18

5.4. Experimental Results of the Hyperparameter Analysis

In this subsection, we analyze the influence of the hyperparameter β on the performance of our proposed unlearning framework, using the Cora and Citeseer datasets. The corresponding results are illustrated in Figure 3. As shown in the figure, the classification accuracy of the unlearned models changes with different β values, reaching its peak when $\beta = 4$. For smaller β values, the model tends to maintain relatively higher accuracy, while larger β values lead to a slight degradation in utility. On the other hand, the AUC of membership inference attacks (MIA) decreases as β increases, suggesting stronger privacy protection at higher β levels. Overall, $\beta = 4$ provides a favorable trade-off between model utility and privacy preservation across both datasets. Therefore, we adopt $\beta = 4$ as the default hyperparameter setting in all experiments.

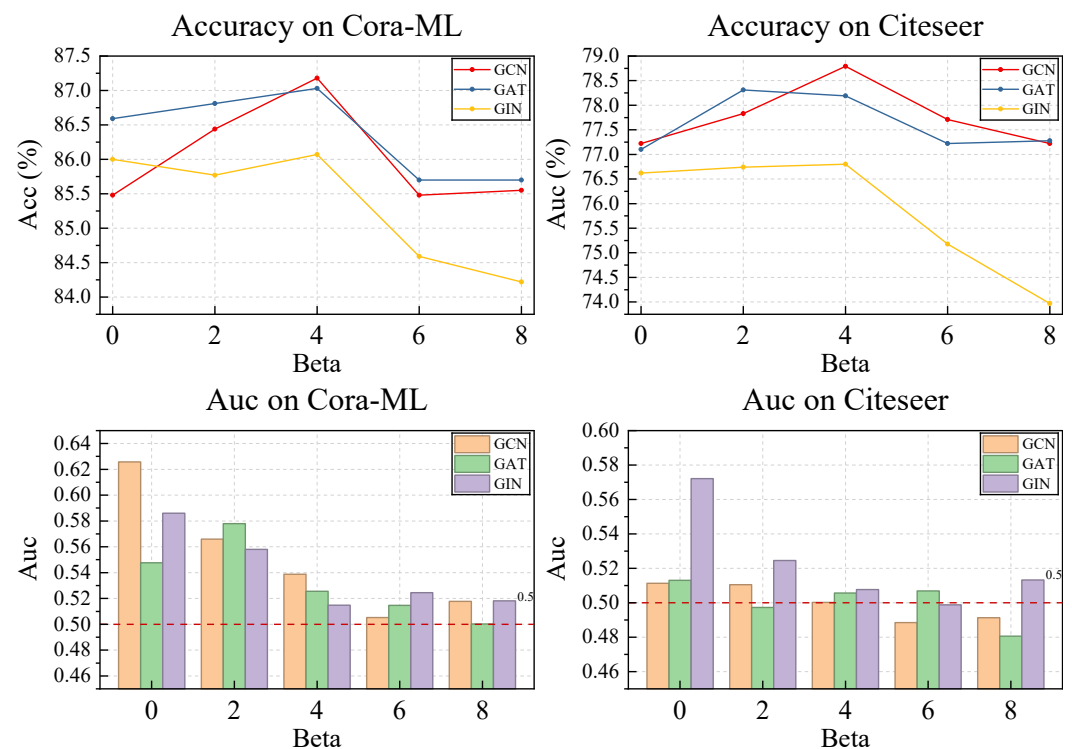


Figure 3. Comparison of the unlearned model accuracy and membership inference attack AUC under different β values on the Cora and Citeseer datasets.

5.5. Experimental Results of Different Unlearning Methods Under Different Unlearning Ratios

In this subsection, we investigate the effect of different unlearning ratios on the overall performance of various unlearning methods on the Citeseer dataset using the GCN model. As reported in Table 5, our method consistently preserves model utility across ratios, exhibiting more stable and generally stronger classification performance, which suggests a smaller disturbance to the primary task while meeting the unlearning objective. Regarding efficiency, although some baselines may show advantages at small unlearning ratios, their runtime tends to increase more noticeably as the unlearning ratio grows, indicating limited scalability. In contrast, our approach maintains a more balanced and controlled time overhead across different settings. Moreover, in terms of the AUC values of the MIA results, our method achieves closer-to-desired suppression results with smaller variations across ratios, demonstrating more consistent and robust unlearning effectiveness. Overall, these results verify that our approach achieves a more favorable trade-off between utility, efficiency, and privacy under varying unlearning ratios.

Table 5. Results of different unlearning methods under different unlearning ratios.

Method	Unlearning Ratio = 5%			Unlearning Ratio = 10%			Unlearning Ratio = 20%		
	Acc (%)	Time (s)	AUC (%)	Acc (%)	Time (s)	AUC (%)	Acc (%)	Time (s)	AUC (%)
Retrain	73.13 ± 0.96	69.38 ± 1.13	69.80	73.91 ± 1.01	65.67 ± 3.75	70.84	73.85 ± 0.65	77.07 ± 1.91	68.82
Gradient Ascent	76.26 ± 0.53	5.91 ± 0.65	50.87	77.22 ± 0.78	9.15 ± 0.83	51.16	76.68 ± 0.16	15.76 ± 0.77	53.55
GIF	76.86 ± 0.13	17.93 ± 2.42	68.68	77.10 ± 0.30	21.06 ± 2.18	68.93	76.74 ± 0.49	20.89 ± 3.83	68.45
Node-CUL	76.32 ± 0.86	20.70 ± 1.79	46.38	76.74 ± 1.15	35.57 ± 0.60	51.53	76.80 ± 0.99	62.10 ± 2.64	51.93
Ours	77.16 ± 0.68	7.49 ± 0.65	51.40	78.79 ± 0.26	13.08 ± 1.29	50.02	79.03 ± 0.81	24.32 ± 1.51	51.47

6. Conclusions

In this work, we addressed the critical challenge of feature-level graph unlearning, a problem largely overlooked by existing node- and edge-level unlearning methods. To achieve fine-grained and effective forgetting while maintaining model utility, we proposed a two-stage unlearning framework composed of Propagation Suppression and Knowledge Dissipation. The first stage suppresses the propagation of residual information from unlearned nodes to their neighbors by modifying the message-passing mechanism in each GNN layer. Additionally, the second stage forces the model to discard previously learned representations through guided misclassification. Extensive experiments on four benchmark datasets, namely Cora-ML, CiteSeer, PubMed, and CS, validate the effectiveness of our approach. The results show that the proposed framework achieves a superior balance among model utility, unlearning efficiency, and privacy protection. In particular, our method maintains accuracy comparable to that of retrained models while significantly reducing computational overhead. Moreover, it achieves AUC values close to 0.5 under membership inference attacks, confirming its robustness in feature-level graph unlearning scenarios.

Author Contributions: Conceptualization, Z.C. (Zhiyu Chen) and Q.L.; Methodology, Z.C. (Zhiyu Chen) and Q.L.; Software, J.L.; Validation, J.L.; Writing—original draft, J.L. and Q.L.; Writing—review and editing, Z.C. (Zhiyu Chen), Q.L. and Z.C. (Zhipeng Cai); Supervision, Z.C. (Zhipeng Cai); Funding acquisition, Z.C. (Zhiyu Chen). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Science and Technology Development Program of Jilin Province under Grant No. 20250203124SF.

Data Availability Statement: The data presented in this study are openly available in Graph-Attribute-Unlearning at <https://github.com/Alchemistqqq/Graph-Attribute-Unlearning> (accessed on 16 March 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

- Voigt, P.; Von dem Bussche, A. *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2017; 339p.
- Pardau, S.L. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. Pol'y* **2018**, *23*, 68.
- Tepedino, G. Desafios da Lei Geral de Proteção de Dados (LGPD). *Rev. Bras. Direito Civ.* **2020**, *26*, 11–15. [[CrossRef](#)]
- Naithani, P. Analysis of India's Digital Personal Data Protection Act, 2023. *Int. J. Law Manag.* **2024**, *67*, 543–553. [[CrossRef](#)]
- Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; Zhang, Y. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confid. Comput.* **2024**, *4*, 100211. [[CrossRef](#)]
- Alzu'Bi, A.; Alomar, A.; Alkhaza'Leh, S.; Abuarqoub, A.; Hammoudeh, M. A review of privacy and security of edge computing in smart healthcare systems: Issues, challenges, and research directions. *Tsinghua Sci. Technol.* **2024**, *29*, 1152–1180. [[CrossRef](#)]
- Yan, B.; Li, K.; Xu, M.; Dong, Y.; Zhang, Y.; Ren, Z.; Cheng, X. On protecting the data privacy of Large Language Models (LLMs) and LLM agents: A literature review. *High-Confid. Comput.* **2025**, *5*, 100300. [[CrossRef](#)]
- Bondy, J.A.; Murty, U.S.R. *Graph Theory with Applications*; Macmillan: London, UK, 1976; Volume 290.

9. Cai, D.; Shao, Z.; He, X.; Yan, X.; Han, J. Community mining from multi-relational networks. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 445–452.
10. Aggarwal, C.C.; Wang, H. *Managing and Mining Graph Data*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 40.
11. Stamile, C.; Marzullo, A.; Deusebio, E. *Graph Machine Learning: Take Graph Data to the Next Level by Applying Machine Learning Techniques and Algorithms*; Packt Publishing Ltd.: Birmingham, UK, 2021.
12. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [[CrossRef](#)] [[PubMed](#)]
13. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [[CrossRef](#)]
14. Kou, H.; Xu, J. KAN-GNN: Kolmogorov-Arnold Network Inspired Graph Neural Networks for Personalized API Recommendation. *Tsinghua Sci. Technol.* **2025**. [[CrossRef](#)]
15. Dong, B.; Wang, W.; Sun, L.; Liu, X.; Mai, Z. Anomaly Detection Using Graph Anomaly Rules. *Big Data Min. Anal.* **2025**, *8*, 1075–1091. [[CrossRef](#)]
16. Gong, Z.; Chen, S.; Dai, Q.; Feng, Y.; Wang, J.; Zhang, J. SCoAMPS: Semi-Supervised Graph Contrastive Learning Based on Associative Memory Network and Pseudo-Label Similarity. *Big Data Min. Anal.* **2025**, *8*, 273–291. [[CrossRef](#)]
17. Li, X.; Li, J.; Hao, J.; Liao, X.; Li, M.; Shang, X. Multiplex networks and pan-cancer multiomics-based driver gene identification using graph neural networks. *Big Data Min. Anal.* **2024**, *7*, 1262–1272. [[CrossRef](#)]
18. Wang, D.; Chen, L.; Gong, F.; Zhu, Q. Maximizing Depth of Graph-Structured Convolutional Neural Networks with Efficient Pathway Usage for Remote Sensing. *Tsinghua Sci. Technol.* **2025**, *30*, 1940–1953. [[CrossRef](#)]
19. Meng, X.; Wen, X.; Sun, X.; Xiang, J.; Long, Y.; Lin, X.; Li, M. SAGAN: A Subgraph-Aware Graph Attention Network for Drug repositioning. *Tsinghua Sci. Technol.* **2025**. [[CrossRef](#)]
20. Bhagat, S.; Cormode, G.; Muthukrishnan, S. Node classification in social networks. In *Social Network Data Analytics*; Springer: Boston, MA, USA, 2011; pp. 115–148.
21. Lopes, G.R.; Moro, M.M.; Wives, L.K.; De Oliveira, J.P.M. Collaboration recommendation on academic social networks. In *Advances in Conceptual Modeling—Applications and Challenges: ER 2010 Workshops ACM-L, CMLSA, CMS, DE@ER, FP-UML, SeCoGIS, WISM, Vancouver, BC, Canada, 1–4 November 2010*; Proceedings 29; Springer: Berlin/Heidelberg, Germany, 2010; pp. 190–199.
22. Manessi, F.; Rozza, A.; Manzo, M. Dynamic graph convolutional networks. *Pattern Recognit.* **2020**, *97*, 107000. [[CrossRef](#)]
23. Zhang, X.M.; Liang, L.; Liu, L.; Tang, M.J. Graph neural networks and their current applications in bioinformatics. *Front. Genet.* **2021**, *12*, 690049. [[CrossRef](#)] [[PubMed](#)]
24. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 97. [[CrossRef](#)]
25. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1024–1034.
26. Pal, A.; Eksombatchai, C.; Zhou, Y.; Zhao, B.; Rosenberg, C.; Leskovec, J. Pinnersage: Multi-modal user embedding framework for recommendations at pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual, 6–10 July 2020; pp. 2311–2320.
27. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
28. Zhang, M.; Chen, Y. Link prediction based on graph neural networks. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 5165–5175.
29. Martínez, V.; Berzal, F.; Cubero, J.C. A survey of link prediction in complex networks. *ACM Comput. Surv. (CSUR)* **2016**, *49*, 69. [[CrossRef](#)]
30. Daud, N.N.; Ab Hamid, S.H.; Saadoon, M.; Sahran, F.; Anuar, N.B. Applications of link prediction in social networks: A review. *J. Netw. Comput. Appl.* **2020**, *166*, 102716. [[CrossRef](#)]
31. Asif, N.A.; Sarker, Y.; Chakraborty, R.K.; Ryan, M.J.; Ahamed, M.H.; Saha, D.K.; Badal, F.R.; Das, S.K.; Ali, M.F.; Moyeen, S.I.; et al. Graph neural network: A comprehensive review on non-euclidean space. *IEEE Access* **2021**, *9*, 60588–60606. [[CrossRef](#)]
32. Li, X.; Sun, L.; Ling, M.; Peng, Y. A survey of graph neural network based recommendation in social networks. *Neurocomputing* **2023**, *549*, 126441. [[CrossRef](#)]
33. Chen, M.; Zhang, Z.; Wang, T.; Backes, M.; Humbert, M.; Zhang, Y. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, Los Angeles, CA, USA, 7–11 November 2022; pp. 499–513.
34. Zhang, J. Graph unlearning with efficient partial retraining. In *Proceedings of the Companion Proceedings of the ACM on Web Conference 2024*, Virtual, 13–17 May 2024; pp. 1218–1221.
35. Wang, C.L.; Huai, M.; Wang, D. Inductive graph unlearning. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*, Anaheim, CA, USA, 9–11 August 2023; pp. 3205–3222.
36. Wu, J.; Yang, Y.; Qian, Y.; Sui, Y.; Wang, X.; He, X. Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, Austin, TX, USA, 1–5 May 2023; pp. 651–661.

37. Cheng, J.; Dasoulas, G.; He, H.; Agarwal, C.; Zitnik, M. Gnndelete: A general strategy for unlearning in graph neural networks. *arXiv* **2023**, arXiv:2302.13406.
38. Zhang, Q.; Yang, C.; Xiong, L.; Lee, H.K. Node-level contrastive unlearning on graph neural networks. *arXiv* **2025**, arXiv:2503.02959.
39. Chien, E.; Pan, C.; Milenkovic, O. Certified graph unlearning. *arXiv* **2022**, arXiv:2206.09140.
40. Wu, K.; Shen, J.; Ning, Y.; Wang, T.; Wang, W.H. Certified edge unlearning for graph neural networks. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 2606–2617.
41. Tan, J.; Sun, F.; Qiu, R.; Su, D.; Shen, H. Unlink to unlearn: Simplifying edge unlearning in gnn. In Proceedings of the Companion Proceedings of the ACM on Web Conference 2024, Singapore, 13–17 May 2024; pp. 489–492.
42. Pan, C.; Chien, E.; Milenkovic, O. Unlearning graph classifiers with limited data resources. In Proceedings of the the ACM Web Conference 2023, Austin, TX, USA, 1–5 May 2023; pp. 716–726.
43. Yin, H.; Liu, Z.; Peng, X.; Xiang, L. Graph Unlearning via Embedding Reconstruction—A Range-Null Space Decomposition Approach. *arXiv* **2025**, arXiv:2508.02044.
44. Ding, P.; Wang, Y.; Liu, G.; Zhu, J. Adaptive Graph Unlearning. *arXiv* **2025**, arXiv:2505.12614.
45. Qu, Y.; Yuan, X.; Ding, M.; Ni, W.; Rakotoarivelo, T.; Smith, D. Learn to unlearn: A survey on machine unlearning. *arXiv* **2023**, arXiv:2305.07512.
46. Bourtole, L.; Chandrasekaran, V.; Choquette-Choo, C.A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; Papernot, N. Machine unlearning. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2021; pp. 141–159.
47. Brophy, J.; Lowd, D. Machine unlearning for random forests. In *Proceedings of the International Conference on Machine Learning*; PMLR: New York, NY, USA, 2021; pp. 1092–1104.
48. Aldaghri, N.; Mahdavi, H.; Beirami, A. Coded machine unlearning. *IEEE Access* **2021**, *9*, 88137–88150. [[CrossRef](#)]
49. Gupta, V.; Jung, C.; Neel, S.; Roth, A.; Sharifi-Malvajerdi, S.; Waites, C. Adaptive machine unlearning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 16319–16330.
50. Neel, S.; Roth, A.; Sharifi-Malvajerdi, S. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the Algorithmic Learning Theory*; PMLR: New York, NY, USA, 2021; pp. 931–962.
51. Tarun, A.K.; Chundawat, V.S.; Mandal, M.; Kankanhalli, M. Fast yet effective machine unlearning. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 13046–13055. [[CrossRef](#)]
52. Chundawat, V.S.; Tarun, A.K.; Mandal, M.; Kankanhalli, M. Zero-shot machine unlearning. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2345–2354. [[CrossRef](#)]
53. Kurmanji, M.; Triantafillou, P.; Hayes, J.; Triantafillou, E. Towards unbounded machine unlearning. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 1957–1987.
54. Liu, Z.; Ye, H.; Chen, C.; Zheng, Y.; Lam, K.Y. Threats, attacks, and defenses in machine unlearning: A survey. *IEEE Open J. Comput. Soc.* **2025**, *6*, 413–425. [[CrossRef](#)]
55. Xu, J.; Wu, Z.; Wang, C.; Jia, X. Machine unlearning: Solutions and challenges. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 2150–2168. [[CrossRef](#)]
56. Li, N.; Zhou, C.; Gao, Y.; Chen, H.; Zhang, Z.; Kuang, B.; Fu, A. Machine unlearning: Taxonomy, metrics, applications, challenges, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 13709–13729. [[CrossRef](#)] [[PubMed](#)]
57. Li, C.; Jiang, H.; Chen, J.; Zhao, Y.; Fu, S.; Jing, F.; Guo, Y. An overview of machine unlearning. *High-Confid. Comput.* **2025**, *5*, 100254. [[CrossRef](#)]
58. Guan, F.; Zhu, T.; Wang, Z.; Ren, W.; Zhou, W. Graph unlearning: Efficient node removal in graph neural networks. *Knowl.-Based Syst.* **2026**, *334*, 116024. [[CrossRef](#)]
59. Zhang, H.; Wu, B.; Yang, X.; Yuan, X.; Liu, X.; Yi, X. Dynamic graph unlearning: A general and efficient post-processing method via gradient transformation. In Proceedings of the ACM on Web Conference 2025, Sydney, NSW, Australia, 28 April–2 May 2025; pp. 931–944.
60. Guo, Q.; Sun, W.; Wang, W. Spatio-Temporal Graph Unlearning. *arXiv* **2025**, arXiv:2511.09404.
61. Said, A.; Derr, T.; Shabbir, M.; Abbas, W.; Koutsoukos, X. A survey of graph unlearning. *arXiv* **2023**, arXiv:2310.02164.
62. Carlini, N.; Chien, S.; Nasr, M.; Song, S.; Terzis, A.; Tramer, F. Membership inference attacks from first principles. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2022; pp. 1897–1914.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.