Erase Then Rectify: A Training-Free Parameter Editing Approach for Cost-Effective Graph Unlearning

Zhe-Rui Yang[1,2,3], **Jindong Han**[4], **Chang-Dong Wang**[1,3,*], **Hao Liu**[2,4,*]
[1]Sun Yat-sen University, Guangzhou, China
[2]The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
[3]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[4]The Hong Kong University of Science and Technology, Hong Kong, China
yangzhr9@mail2.sysu.edu.cn, jhanao@connect.ust.hk, wangchd3@mail.sysu.edu.cn, liuh@ust.hk

Abstract

Graph unlearning, which aims to eliminate the influence of specific nodes, edges, or attributes from a trained Graph Neural Network (GNN), is essential in applications where privacy, bias, or data obsolescence is a concern. However, existing graph unlearning techniques often necessitate additional training on the remaining data, leading to significant computational costs, particularly with large-scale graphs. To address these challenges, we propose a two-stage training-free approach, *Erase then Rectify* (ETR), designed for efficient and scalable graph unlearning while preserving the model utility. Specifically, we first build a theoretical foundation showing that masking parameters critical for unlearned samples enables effective unlearning. Building on this insight, the Erase stage strategically edits model parameters to eliminate the impact of unlearned samples and their propagated influence on intercorrelated nodes. To further ensure the GNN's utility, the Rectify stage devises a gradient approximation method to estimate the model's gradient on the remaining dataset, which is then used to enhance model performance. Overall, ETR achieves graph unlearning without additional training or full training data access, significantly reducing computational overhead and preserving data privacy. Extensive experiments on seven public datasets demonstrate the consistent superiority of ETR in model utility, unlearning efficiency, and unlearning effectiveness, establishing it as a promising solution for real-world graph unlearning challenges.

Code — https://github.com/AllminerLab/ETR
Extended version — https://arxiv.org/abs/2409.16684

Introduction

Machine unlearning has garnered attention for its ability to remove the influence of specific data subsets from a well-trained machine learning model (Nguyen et al. 2022), which is crucial for forgetting sensitive, mislabeled, or outdated information (Liu and Tsang 2017). Typically, the subset of data to be unlearned is much smaller than the entire training dataset. As a result, one primary goal of machine unlearning is to efficiently eliminate the impact of these unlearned samples while preserving the model's predictive power, thereby avoiding the costly process of retraining the model from scratch on the remaining data (Xu et al. 2024).

In the graph domain, there is also a strong demand to forget specific information, such as users requesting the removal of personal data in social networks. In particular, Graph Neural Networks (GNNs) are widely used to process graph-structured data through message propagation and neighborhood aggregation (Zhou et al. 2020). However, due to the interconnected nature of graphs, unlearned samples can significantly affect their neighbors, making conventional machine unlearning methods unapplicable (Cheng et al. 2023). As a result, graph unlearning has emerged as a specialized task aimed at removing unlearned samples and mitigating their impact on other nodes (Said et al. 2023).

Recently, several graph unlearning approaches have been proposed. For instance, GIF (Wu et al. 2023a) and CGU (Chien, Pan, and Milenkovic 2023) approximate parameter changes caused by data removal and update the affected parameters. However, these methods primarily focus on forgetting unlearned samples while overlooking the predictive capability on the remaining data, which can significantly compromise model utility (Li et al. 2024). To address this, some recent studies (Li et al. 2024; Wang, Huai, and Wang 2023) consider both predictive and unlearning objectives when an unlearning request is received. However, they typically require additional training on the remaining data, leading to high computational overhead, particularly with large graphs, even when only a few samples are unlearned. This substantial resource demand limits the efficiency and scalability of these methods in real-world applications.

In this work, we focus on developing an efficient and scalable graph unlearning method that preserves model utility. However, achieving this goal is non-trivial. First, the interconnected nature of graph data means that unlearned samples can significantly affect their neighbors through message propagation, complicating the removal of both the unlearned samples and their influence. Second, the remaining dataset is typically much larger than the unlearned subset, leading to substantial computational overhead when accessing the remaining data. Balancing model utility with low computational cost presents another significant challenge.

To address these challenges, we propose **E**rase **t**hen **R**ectify (ETR), a training-free, two-stage approach for cost-effective graph unlearning. First, we establish a theoretical

[*]Corresponding author.

foundation showing that masking parameters critical to unlearned samples enables effective unlearning. Based on this, in the Erase stage, we propose a neighborhood-aware parameter editing strategy to remove the impact of unlearned samples and their cascading effects, while minimizing the impact on model utility. In the Rectify stage, we introduce a subgraph-based gradient approximation method to estimate the unlearned model's gradient on the remaining data, further enhancing model performance. Notably, ETR achieves graph unlearning without requiring explicit model training or access to the entire training dataset, avoiding computational overhead and ensuring data privacy. Overall, the main contributions are as follows:

- We theoretically prove that masking parameters crucial for unlearned samples enable effective graph unlearning.
- We propose a training-free neighborhood-aware parameter editing method for graph unlearning that effectively forgets unlearned samples and their influence on intercorrelated samples.
- We propose a gradient approximation method to enhance GNN performance on the remaining graph without requiring access to the entire training dataset, thereby ensuring efficiency and scalability on large-scale graphs.
- Extensive experiments demonstrate the model utility, unlearning efficiency, and unlearning efficacy of ETR. Notably, ETR achieves an average of 4583.9x less time and 4.2x less memory usage than retraining from scratch.

Related work

Graph Unlearning

Graph unlearning methods are divided into exact and approximate approaches, aiming to obtain a model that is exactly or approximately equivalent to retraining from scratch (Said et al. 2023). Current efforts primarily focus on approximate methods. For example, GraphEraser (Chen et al. 2022) and GUIDE (Wang, Huai, and Wang 2023) follow the SISA (Bourtoule et al. 2021) paradigm, partitioning the training set into multiple shards, with a separate model trained for each shard. Other approaches (Wu et al. 2023a,b; Chen et al. 2023; Chien, Pan, and Milenkovic 2023) use influence functions to approximate the impact of unlearning on model parameters. GNNDelete (Cheng et al. 2023) freezes model parameters while training additional parameters for unlearning. MEGU (Li et al. 2024) simultaneously trains the model's prediction and unlearning objectives. Despite their effectiveness, these methods still face scalability issues with large graphs.

Training-Free Machine Unlearning

Several training-free methods have been proposed for efficient machine unlearning. Fisher Forgetting (Golatkar, Achille, and Soatto 2020a) and NTK (Golatkar, Achille, and Soatto 2020b) are weight scrubbing methods that add noise to parameters informative for unlearned samples. However, Fisher Forgetting is computationally expensive and harms predictive performance (Tarun et al. 2023), while NTK requires additional models for unlearning. The closest work to ours is SSD (Foster, Schoepf, and Brintrup 2024), which dampens parameters crucial for the unlearned dataset while preserving others. Unlike SSD, we account for the impact of unlearned samples on other samples through message propagation. We also propose adaptive hyperparameter selection to improve versatility across datasets and introduce the Rectify strategy to enhance performance on remaining data.

Preliminaries

Notations and Background

In this paper, we employ $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ to denote a graph comprising $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges. Each node $\mathbf{v}_i \in \mathcal{V}$ has a d-dimensional feature vector $\mathbf{x}_i \in \mathcal{X}$. A major category of GNNs is message-passing neural networks (MPNNs) (Gilmer et al. 2017), such as GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018). MPNNs propagate and aggregate features from neighboring nodes, then transform these aggregated features to update node representations (Gilmer et al. 2017). Following previous works (Chen et al. 2022; Said et al. 2023), we focus on node classification in this study. We denote the training dataset as D, the unlearned dataset as D_f, the remaining dataset as D_r (i.e., $D \setminus D_f$), and the k-hop neighborhood of D_f as D_k.

Fisher Information Matrix (FIM)

The FIM uses the second-order derivative of the loss function to quantify parameter sensitivity, providing a measure of importance with respect to input samples (Guo et al. 2020). For a distribution $p(y|x, w)$, the FIM and its first-order derivative property (Kay 1993) are given as follows:

$$F_D = \mathbb{E}_{x,y}\left[-\nabla_w^2 \log p\left(y|x,w\right)\right] \tag{1}$$

$$= \mathbb{E}_{x,y}\left[\nabla_w \log p\left(y|x,w\right) \nabla_w \log p\left(y|x,w\right)^T\right] \tag{2}$$

Notably, obtaining the FIM is computationally expensive. A common approach is to use its diagonal $diag(F)$ (Kirkpatrick et al. 2017), which can be computed efficiently using the first-order derivative. The i-th diagonal element of the FIM over dataset D is denoted as $F_{D,ii}$, representing the importance of the i-th parameter with respect to D.

Problem Statement

Given a graph $\mathcal{G}$, the optimal GNN model $f_{\mathcal{G}}$ trained on $\mathcal{G}$, and a graph unlearning request $\Delta\mathcal{G} = (\Delta\mathcal{V}, \Delta\mathcal{E}, \Delta\mathcal{X})$, the objective of graph unlearning is to derive a new GNN model $\widehat{f}$ that minimizes the discrepancy between $\widehat{f}$ and $f_{\mathcal{G}\setminus\Delta\mathcal{G}}$. Here, $f_{\mathcal{G}\setminus\Delta\mathcal{G}}$ represents the GNN model retrained from scratch on the remaining graph $\mathcal{G} \setminus \Delta\mathcal{G}$.

Graph unlearning can be categorized into three tasks: **Node Unlearning**: $\Delta\mathcal{G} = (\Delta\mathcal{V}, \Delta\mathcal{E}, \Delta\mathcal{X})$, where $\Delta\mathcal{V}$ represents unlearned nodes, $\Delta\mathcal{E}$ denotes edges connecting to $\Delta\mathcal{V}$, and $\Delta\mathcal{X}$ represents the features of nodes in $\Delta\mathcal{V}$. **Edge Unlearning**: $\Delta\mathcal{G} = (\emptyset, \Delta\mathcal{E}, \emptyset)$, where $\Delta\mathcal{E}$ represents the set of edges to be unlearned. **Feature Unlearning**: $\Delta\mathcal{G} = (\emptyset, \emptyset, \Delta\mathcal{X})$, where $\Delta\mathcal{X}$ represents the unlearned features, and will be padded with zeros. We will use the node unlearning task to illustrate the proposed method, and describe how

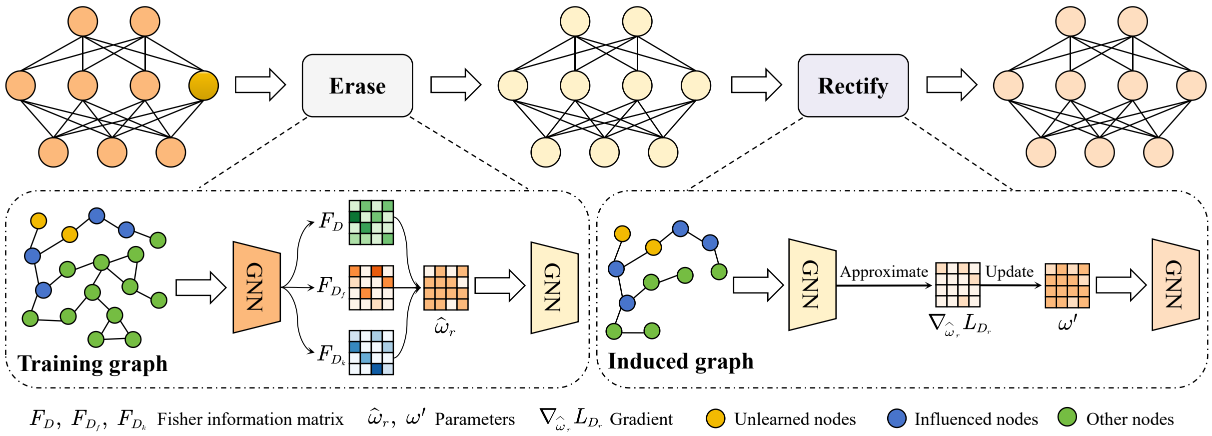

Figure 1: The framework of ETR: Forgetting the influence of unlearned samples through Erase, followed by enhancing the model performance on the remaining dataset via Rectify.

the proposed method applies to edge unlearning and feature unlearning tasks in the Appendix (Yang et al. 2024).

Methodology

Framework Overview

The overall framework of ETR is depicted in Figure 1. In the Erase stage, we evaluate parameter importance with respect to input samples based on the FIM. Subsequently, we identify critical parameters concerning the unlearned samples and their influence on other samples. We then modify these parameters to erase the effect of the unlearned samples. In the Rectify stage, we utilize the induced graph of unlearned samples to approximate model gradients on the remaining dataset, which are then used to enhance the model performance on the remaining dataset.

Graph Unlearning via Parameter Masking

Previous research indicates that certain neurons or parameters are critical in memorizing specific samples (Maini et al. 2023), and that gradient manipulation can prevent the memorization of noisy information (Chen et al. 2021). Motivated by these findings, we propose to achieve efficient unlearning by directly masking the parameters responsible for memorizing the data to be unlearned, as described below:

$$\widehat{\omega}_{r,j} = \begin{cases} 0, & j \in M \\ \omega_j^*, & j \notin M \end{cases} \tag{3}$$

where ω^* represents the optimal parameters on D, and M denotes the set of parameters responsible for memorizing the unlearned dataset. To explore the effectiveness of the masking strategy, we provide theoretical justification for its ability to achieve graph unlearning as follows:

Theorem 1. *For a GNN model, if we approximate the FIM F with its diagonal $diag(F)$, and assume all elements of $diag(F)$ are strictly positive, the mean squared distance between the optimal model parameters trained on D_r and the parameters obtained from (3), denoted as $Q = \frac{1}{|\omega|}\sum_j \left(\omega_{r,j}^* - \widehat{\omega}_{r,j}\right)^2$, has the following upper bound:*

$$Q \leqslant \frac{1}{|\omega|}\left(c_1 \sum_{j\in M} \frac{1}{F_{D_r,jj}^2} + c_2 \sum_{j\notin M} \frac{F_{D_f,jj}^2}{F_{D,jj}^2 F_{D_r,jj}^2} + c_3\right) \tag{4}$$

where c_1, c_2, c_3 are constants, and $|\omega|$ denotes the number of parameters.

The proof of Theorem 1 is provided in the Appendix. According to Theorem 1, adding the j-th parameter to M will increase the upper bound of Q by $\frac{1}{|\omega|}\left(\frac{c_1}{F_{D_r,jj}^2} - \frac{c_2 F_{D_f,jj}^2}{F_{D,jj}^2 F_{D_r,jj}^2}\right)$. This increase is less than 0 when $\frac{F_{D_f,jj}}{F_{D,jj}} > \sqrt{\frac{c_1}{c_2}} = c$, indicating that the parameter w_j is much more important for D_f than for D. Therefore, masking parameters that are crucial for unlearned samples but not for others can reduce the upper bound of Q, facilitating the forgetting of the unlearned dataset.

Erase: Neighborhood-Aware Parameter Editing

Despite the above analysis theoretically guaranteeing the effectiveness of the masking strategy in achieving unlearning, it has two major limitations First, the masking strategy overlook the extent to which $\frac{F_{D_f,jj}}{F_{D,jj}}$ exceeds c. In particular, parameters for which $\frac{F_{D_f,jj}}{F_{D,jj}}$ significantly exceeds c are much more critical for the unlearned dataset compared to those where $\frac{F_{D_f,jj}}{F_{D,jj}}$ only slightly exceeds c. Second, the masking strategy overlooks the impact of unlearned samples on their neighbors through message propagation.

To address these, we propose the Erase strategy as:

$$\widehat{\omega}_{r,j} = \begin{cases} a\frac{F_{D,jj}}{F_{D_f,jj}}\omega_j^*, & F_{D_f,jj} > \gamma F_{D,jj}, \\ b\frac{{F_{D,jj}}^2}{F_{D_f,jj}F_{D_k,jj}}\omega_j^*, & F_{D_f,jj}F_{D_k,jj} > \eta {F_{D,jj}}^2 \\ & \text{and } F_{D_f,jj} \leq \gamma F_{D,jj}, \\ \omega_j^*, & otherwise. \end{cases} \tag{5}$$

where a, b, γ, and η are hyperparameters.

The advantages of the above design are threefold. First, it considers the impact of unlearned samples on their neighbors. According to the properties of the FIM, a large ratio of $\frac{F_{D_f,jj}}{F_{D,jj}}$ or $\frac{F_{D_k,jj}}{F_{D,jj}}$ indicates that the parameter is crucial for D_f or D_k, but not for D. Drawing inspiration from Theorem 1, if a parameter is crucial for both D_f and D_k, but not for other samples, i.e., satisfying $F_{D_f,jj}F_{D_k,jj} > \eta {F_{D,jj}}^2$, we consider that the parameter has been influenced by message propagation from D_f. In such cases, we modify these parameters to forget the influence of unlearned samples.

Second, when modifying parameters, we account for the extent to which they are influenced by unlearned samples. Specifically, rather than directly masking the parameters, we use the coefficients $\frac{F_{D,jj}}{F_{D_f,jj}}$ and $\frac{{F_{D,jj}}^2}{F_{D_f,jj}F_{D_k,jj}}$ to determine the degree of modification. This implies that a large $\frac{F_{D_f,jj}}{F_{D,jj}}$ or $\frac{F_{D_f,jj}F_{D_k,jj}}{{F_{D,jj}}^2}$ necessitates substantial modifications to ω_j^* , while a small value requires only minor modifications.

Finally, we balance the two forgetting objectives: forgetting D_f and forgetting the influence of D_f on D_k. One of these objectives may dominate the other due to the potential difference in magnitude between $\frac{F_{D,jj}}{F_{D_f,jj}}$ and $\frac{{F_{D,jj}}^2}{F_{D_f,jj}F_{D_k,jj}}$. To mitigate this, we introduce two balancing coefficients a and b to balance these two forgetting objectives.

Note the choice of hyperparameters may vary depending on the dataset. To facilitate hyperparameter selection, we adaptively select them for different datasets. Specifically, γ is set as the top $m\%$ value of $\frac{F_{D_f}}{F_D}$, and η is set as the top $m\%$ value of $\frac{F_{D_f}F_{D_k}}{{F_D}^2}$. This ensures that $m\%$ of the parameters satisfy $F_{D_f,jj} > \gamma F_{D,jj}$ and $m\%$ satisfy $F_{D_f,jj}F_{D_k,jj} > \eta {F_{D,jj}}^2$. Furthermore, we set $a = \gamma$ and $b = \eta$ to balance the magnitudes of $\frac{F_{D,jj}}{F_{D_f,jj}}$ and $\frac{{F_{D_{jj}}}^2}{F_{D_f,jj}F_{D_k,jj}}$.

Rectify: Model Utility Enhancement

Although the Erase strategy mitigates the impact on model performance by considering the extent to which unlearned samples influence parameters, parameter editing approach may still negatively affect the GNN's performance on D_r to some degree. In this part, we propose the Rectify strategy to enhance GNN performance on D_r without requiring access to the entire training dataset, which minimizes computational overhead while ensuring data privacy and security.

Firstly, we estimate the gradients of the edited model on D_r using the induced graph of D_f Specifically, we assume that we can store the gradients from the model's final iteration of the training phase. After removing D_f, the neighbor structure of $D_r \setminus D_k$ remains unchanged, while only the neighbor structures of D_f and D_k undergo changes. Additionally, during the Erase stage, the modified parameters are crucial only for D_f and D_k, but not for $D_r \setminus D_k$. Therefore, for nodes in $D_r \setminus D_k$, their gradients can be considered nearly unchanged, i.e.,

$$\nabla_{\widehat{\omega}_r} l_j \approx \nabla_{\omega^*} l_j, \forall j \in D_r \setminus D_k \tag{6}$$

where l_j denotes the loss of the GNN on the j-th sample. Through the aforementioned approximation, we can easily

---

Algorithm 1: ETR: Erase then Rectify

Input: Unlearned dataset D_f; Dataset D_k; GNN $f_{\mathcal{G}}$; Optimal parameters ω^*; Gradient $\nabla_{\omega^*} L_D$; Hyperparameters m and λ.

Erase

Calculate the gradients $\nabla_{\omega^*} L_{D_f}$ and $\nabla_{\omega^*} L_{D_k}$.
Calculate the FIM F_D, F_{D_f}, F_{D_k} via (2).
Obtain $\gamma = top\text{-}m\%(\frac{F_{D_f}}{F_D})$ and $\eta = top\text{-}m\%(\frac{F_{D_f}F_{D_k}}{{F_D}^2})$
for j $in\ range$ $|\omega|$ **do**
    Obtain $\alpha_j = \frac{F_{D,jj}}{F_{D_f,jj}}$ and $\beta_i = \frac{{F_{D,jj}}^2}{F_{D_f,jj}F_{D_k,jj}}$
    if $F_{D_f,jj} > \gamma F_{D,jj}$ **then**
        $\widehat{\omega}_{r,j} = \alpha_j \gamma \omega_j^*$
    else if $F_{D_f,jj}F_{D_k,jj} > \eta {F_{D,jj}}^2$ **then**
        $\widehat{\omega}_{r,j} = \beta_j \eta \omega_j^*$
    else
        $\widehat{\omega}_{r,j} = \omega_j^*$
    end if
end for

Rectify

Calculate the gradient $\nabla_{\widehat{\omega}_r} L_{D_k}$.
Obtain the gradient $\nabla_{\widehat{\omega}_r} L_{D_r}$ via (9).
Obtain the parameters ω' via (10).

Output: ω'.

---

calculate the gradient of the edited model on D_r as follows:

$$\nabla_{\widehat{\omega}_r} L_{D_r} = \frac{1}{|D_r|} \sum_{j \in D_r} \nabla_{\widehat{\omega}_r} l_j \tag{7}$$

$$\approx \frac{1}{|D_r|} \left(\sum_{j \in D_r \setminus D_k} \nabla_{\omega^*} l_j + \sum_{j \in D_k} \nabla_{\widehat{\omega}_r} l_j \right) \tag{8}$$

where L_D denotes the loss function of the GNN on dataset D. Further, leveraging the gradient of the GNN on D, we obtain $\sum_{j \in D_r \setminus D_k} \nabla_{\omega^*} l_j = |D| \nabla_{\omega^*} L_D - \sum_{j \in D_f} \nabla_{\omega^*} l_j - \sum_{j \in D_k} \nabla_{\omega^*} l_j$. Finally, we estimate the gradient of the edited model on D_r as follows:

$$\nabla_{\widehat{\omega}_r} L_{D_r} = \frac{1}{|D_r|} (|D| \nabla_{\omega^*} L_D - |D_f| \nabla_{\omega^*} L_{D_f} - |D_k| \nabla_{\omega^*} L_{D_k} + |D_k| \nabla_{\widehat{\omega}_r} L_{D_k}) \tag{9}$$

Notably, $\nabla_{\omega^*} L_D$ can be stored during GNN training, while the sizes of D_f and D_k are typically small. Therefore, we can efficiently approximate $\nabla_{\widehat{\omega}_r} L_{D_r}$ through (9). After obtaining the gradient $\nabla_{\widehat{\omega}_r} L_{D_r}$, we rectify the parameters through one-step gradient descent, defined as follows:

$$\omega' = \widehat{\omega}_r - \lambda \nabla_{\widehat{\omega}_r} L_{D_r} \tag{10}$$

where λ is the hyperparameter. The pseudo-code for the ETR approach is presented in Algorithm 1.

For edge and feature unlearning, we can proceed in a similar manner. The pseudo-code for edge and feature unlearning is provided in the Appendix.

Complexity Analysis

In this section, we analyze the complexity of ETR. Calculating $\nabla_{\omega^*} L_{D_f}$ with a time complexity of $\mathcal{O}(|D_f|)$. Similarly,

Bone	Method	PubMed		CiteSeer		Cora		CS		Physics	
		F1	T	F1	T	F1	T	F1	T	F1	T
GCN	Retrain	89.74±0.27	39.1	76.76±0.72	14.1	84.35±1.14	7.5	92.63±0.30	285.8	95.90±0.10	856.7
	BEKM	80.03±0.72	68.2	61.11±2.27	12.7	66.42±3.33	11.0	89.03±0.49	71.7	94.53±0.19	152.7
	BLPA	81.63±0.83	66.1	57.69±2.31	13.0	61.51±3.48	11.5	88.88±0.56	68.2	94.45±0.24	142.8
	GIF	73.66±0.26	0.4	62.88±1.34	0.2	75.87±1.86	0.2	85.19±0.56	1.3	91.37±0.23	3.3
	SR	88.04±0.25	29.0	75.35±0.70	8.6	81.25±0.95	8.6	91.77±0.28	49.0	94.72±0.14	166.4
	Fast	88.11±0.24	29.1	75.83±0.88	8.8	80.70±1.70	8.4	91.64±0.29	47.8	94.74±0.16	138.3
	GNNDelete	84.38±1.27	2.3	74.77±0.94	1.8	78.82±1.50	1.7	90.09±0.59	3.8	85.84±3.07	6.5
	MEGU	88.26±0.47	0.3	76.04±0.93	0.2	82.62±1.05	0.2	92.39±0.51	0.5	95.40±0.20	0.9
	ETR	**89.48±0.34**	**0.02**	**77.09±0.60**	**0.02**	**83.54±0.92**	**0.02**	**92.72±0.37**	**0.04**	**95.71±0.15**	**0.07**
GAT	Retrain	88.42±0.36	50.2	76.52±1.00	16.3	82.29±1.00	9.0	92.57±0.43	295.47	96.26±0.18	875.2
	BEKM	70.30±0.77	102.2	49.64±1.36	23.5	56.79±4.83	20.7	81.10±1.39	114.3	92.04±0.43	214.5
	BLPA	71.65±0.78	98.2	49.52±2.44	20.1	49.70±3.17	18.7	81.35±0.58	94.2	91.82±0.43	194.5
	GIF	88.07±0.22	1.1	75.80±1.06	0.7	81.70±1.38	0.6	**92.79±0.36**	2.1	95.84±0.16	5.1
	SR	86.67±0.41	33.6	75.92±0.88	12.0	76.05±1.78	12.3	89.49±0.64	54.8	94.76±0.41	149.3
	Fast	86.60±0.33	35.4	75.89±0.93	12.4	75.46±0.93	12.3	89.70±0.95	55.3	94.86±0.45	145.5
	GNNDelete	83.09±3.67	3.1	67.21±4.36	2.6	76.31±2.65	2.5	91.08±1.22	4.8	94.43±0.95	8.1
	MEGU	85.23±0.47	0.3	76.07±0.79	0.2	**82.84±0.72**	0.2	91.74±0.32	0.6	95.14±0.22	1.2
	ETR	**88.32±0.22**	**0.03**	**76.31±1.59**	**0.03**	79.70±0.85	**0.03**	92.45±0.13	**0.05**	**96.19±0.18**	**0.09**

Table 1: Performance comparison in terms of F1 score and running time, where F1 indicates the F1 score, T indicates the running time (in seconds). The best experimental results are highlighted in bold, while the second-best results are underscored.

calculating $\nabla_{\omega^*} L_{D_k}$ and $\nabla_{\widehat{\omega}_r} L_{D_k}$ with a time complexity of $\mathcal{O}(|D_k|)$. Computing the FIM has a time complexity of $\mathcal{O}(|\omega|)$. The Erase strategy can be executed in parallel, with a time complexity of $\mathcal{O}(|\omega|)$. Thus, the overall time complexity of ETR is $\mathcal{O}(|D_f| + |D_k| + |\omega|)$. Regarding space complexity, the GNN parameters have a space complexity of $\mathcal{O}(|\omega|)$. The node features have a space complexity of $\mathcal{O}(|D_f|d + |D_k|d)$. The edges have a space complexity of $\mathcal{O}(|\mathcal{E}_f| + |\mathcal{E}_k|)$, where $|\mathcal{E}_f|$ and $|\mathcal{E}_k|$ denote the number of edges in D_f and D_k, respectively. Computing the FIM has a space complexity of $\mathcal{O}(|\omega|)$. Therefore, the overall space complexity of ETR is $\mathcal{O}(|D_f|d + |D_k|d + |\omega| + |\mathcal{E}_f| + |\mathcal{E}_k|)$. Notably, D_f and D_k are much smaller than D, resulting in low computational overhead for large-scale graphs.

Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of ETR. The experiments aim to answer the following research questions: **RQ1**: How does ETR perform in terms of model utility, unlearning efficiency, and unlearning efficacy? **RQ2**: How does ETR perform on large-scale graphs? **RQ3**: How do hyperparameters influence ETR's performance? **RQ4**: How do different strategies in ETR contribute to its effectiveness?

Experimental Setup

Datasets We conduct experiments on PubMed, CiteSeer, Cora (Yang, Cohen, and Salakhutdinov 2016), CS, Physics (Shchur et al. 2018), ogbn-arxiv, and ogbn-products (Hu et al. 2020). The statistics and detailed descriptions of these datasets are provided in the Appendix.

Baselines We compare ETR with various baselines, including Retrain, GraphEraser-BEKM, GraphEraser-BLPA (Chen et al. 2022), GIF (Wu et al. 2023a), GUIDE-SR, GUIDE-Fast (Wang, Huai, and Wang 2023), GNNDelete (Cheng et al. 2023), and MEGU (Li et al. 2024). Detailed descriptions of these baselines are provided in the Appendix.

Settings Following GIF, we partition each graph into a training subgraph comprising training nodes and a test subgraph containing the remaining nodes. For PubMed, CiteSeer, Cora, CS, and Physics, we randomly allocate 90% of the nodes to the training set. For ogbn-arxiv and ogbn-products, we use the splits provided in (Hu et al. 2020). The unlearning ratio is set to 5% for all tasks. We use two-layer GCN and GAT models with a hidden state dimension of 256, training for 100 epochs. We tune the learning rates within the range of [1e-4, 1e-1] and the weight decay within [1e-7, 1e-2] for different datasets. All experiments are conducted on a single Nvidia A40 GPU. Each experiment is run ten times, and we report the average value and standard deviation.

Evaluation of Model Utility (RQ1)

We evaluate the F1 score for node classification on the remaining dataset. The results for node unlearning are shown in Table 1, while the results for edge and feature unlearning are provided in the Appendix. It can be observed that ETR consistently performs best in most cases, which can be attributed to the Rectify strategy that enhances model performance on the remaining dataset. Additionally, BEKM and BLPA exhibit suboptimal performance, consistent with the results reported in their original papers, likely due to disruptions in graph structure resulting from graph partitioning. GUIDE mitigates this problem, leading to improved performance. Among the baselines, MEGU achieves the best performance, which can be attributed to its training of the model's predictive objective during unlearning.

Bone	Method	PubMed	CiteSeer	Cora	CS	Physics
GCN	Retrain	1.57	1.52	1.30	6.13	14.62
	BEKM	0.80	0.60	0.45	5.87	14.87
	BLPA	**0.45**	**0.48**	**0.38**	**1.44**	2.89
	GIF	1.70	1.00	0.61	11.18	37.63
	SR	1.48	1.23	1.20	2.09	4.97
	Fast	1.44	1.23	1.20	2.36	5.10
	GNNDelete	1.64	0.87	1.22	2.76	5.25
	MEGU	0.68	0.52	0.41	2.07	4.52
	ETR	1.18	1.21	1.17	1.61	**2.43**
GAT	Retrain	1.77	1.52	1.30	6.17	15.28
	BEKM	0.78	0.60	0.45	5.87	14.86
	BLPA	**0.48**	**0.48**	**0.40**	1.60	3.53
	GIF	2.88	0.80	0.72	5.89	14.3
	SR	1.52	1.23	1.20	2.20	5.21
	Fast	1.52	1.23	1.20	2.41	5.32
	GNNDelete	1.90	1.31	1.24	3.05	6.10
	MEGU	0.94	0.57	0.45	3.62	7.64
	ETR	1.20	1.21	1.17	**1.54**	**2.30**

Table 2: The memory overhead (GB) of different methods.

Backbone	Method	PubMed	CiteSeer	Cora	CS	Physics
GCN	BEKM	5e-2	2e-3	8e-3	2e-2	2e-2
	BLPA	5e-2	2e-3	8e-3	2e-2	1e-2
	GIF	7e-2	4e-3	2e-2	3e-2	1e-2
	SR	9e-2	8e-3	2e-2	8e-3	2e-3
	Fast	8e-2	9e-3	2e-2	8e-3	2e-3
	MEGU	**1e-2**	3e-3	3e-2	1e-3	7e-4
	ETR	3e-2	**5e-4**	**9e-4**	**4e-4**	**2e-4**
GAT	BEKM	**9e-3**	1e-2	2e-2	9e-3	5e-4
	BLPA	**9e-3**	1e-2	2e-2	8e-3	5e-4
	GIF	6e-2	7e-3	2e-2	2e-3	3e-2
	SR	7e-2	1e-2	1e-2	4e-3	7e-2
	Fast	7e-2	1e-2	1e-2	4e-3	7e-2
	MEGU	9e-2	4e-2	3e-3	2e-2	1e-2
	ETR	3e-2	**1e-3**	**2e-3**	**4e-4**	**2e-4**

Table 3: The root mean square distance between the parameters and those of Retrain.

Evaluation of Unlearning Efficiency (RQ1)

The runtime results are presented in Table 1. ETR consistently outperforms all baselines, which can be attributed to its training-free nature. ETR reduces runtime by thousands of times compared to Retrain, demonstrating its superior efficiency. While GraphEraser and GUIDE are also faster than Retrain in most cases, they still require considerable time for unlearning due to the need for retraining multiple models. Although GIF and MEGU improve runtime efficiency compared to other baselines, they still fall short of ETR. Specifically, ETR reduces runtime by 38 times compared to GIF and 12 times compared to MEGU.

The memory overhead results are shown in Table 2. ETR performs comparably to the baselines on small datasets and outperforms them on larger datasets (Photo and Computer). This is because memory overhead on small datasets is primarily driven by model parameters, limiting the advantages of ETR. However, ETR reduces memory overhead on large datasets by not requiring the entire training dataset, decreasing it by 3.4 times compared to Retrain. Among the baselines, BLPA performs best due to its use of graph partitioning, which reduces memory overhead but significantly degrades model utility. Overall, ETR achieves an excellent trade-off between unlearning efficiency and model utility.

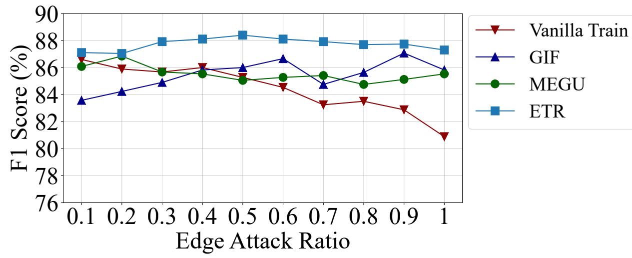

Figure 2: Adversarial edge unlearning on the Cora dataset.

Bone	Method	ogbn-arxiv				ogbn-products			
		F1	T	S	D	F1	T	S	D
GCN	Retrain	67.5	152	3.7	-	73.6	519	11.2	-
	BEKM	64.8	1290	2.9	8e-1	OM	OM	OM	OM
	BLPA	63.9	1323	**0.9**	8e-1	OM	OM	OM	OM
	GIF	62.1	7	19.7	1	OM	OM	OM	OM
	GNNDelete	51.7	34	8.2	-	OM	OM	OM	OM
	MEGU	65.6	2	5.9	4e-2	71.3	7	26.8	6e-2
	ETR	**66.4**	**0.08**	1.2	**2e-2**	**73.8**	**0.5**	**1.2**	**9e-3**
GAT	Retrain	67.3	217.9	6	-	73.9	709	11.2	-
	BEKM	65.3	1804	3.0	6e-1	OM	OM	OM	OM
	BLPA	64.7	1658	2.5	6e-1	OM	OM	OM	OM
	GIF	66.1	12	34.5	5e-1	OM	OM	OM	OM
	GNNDelete	49.6	44	10.1	-	OM	OM	OM	OM
	MEGU	58.5	3	11.2	**4e-3**	63.7	8	26.9	5e-2
	ETR	**67.8**	**0.1**	**1**	2e-2	**73.9**	**0.7**	**1.3**	**2e-2**

Table 4: Performance comparison on large-scale graphs, where S denotes memory overhead, D indicates parameter distance, and "OM" represents "out of memory".

Evaluation of Unlearning Efficacy (RQ1)

We assess unlearning efficacy using both direct and indirect evaluation methods. For direct evaluation, we measure the root mean square distance between the parameters of the unlearned model and those of the retrained model. As shown in Table 3, ETR consistently performs best in most cases, indicating that its parameters are closest to those obtained by retraining from scratch, thereby demonstrating ETR's effectiveness in achieving graph unlearning.

For indirect evaluation, we follow the approach used by GIF and MEGU, adding adversarial edges between nodes of different categories in the training graph. These edges serve as unlearning targets, and we then assess the utility of the unlearned model. As shown in Figure 2, the performance of the vanilla trained model declines with increasing attack ratios, while GIF, MEGU, and ETR maintain or even improve performance. This demonstrates the effectiveness of these methods. Notably, ETR consistently outperforms the other baselines, highlighting its strong unlearning efficacy.

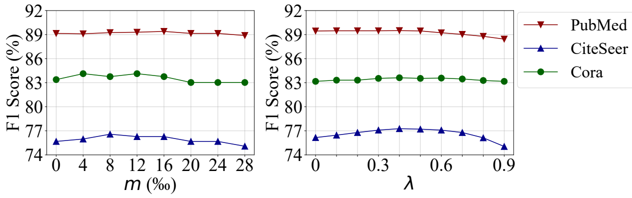

Figure 3: Performance with varying hyperparameters.

Evaluation on Large Scale Graphs (RQ2)

We conduct experiments on the ogbn-arxiv and ogbn-products datasets to assess ETR's effectiveness and efficiency on large-scale graphs, with the results shown in Table 4. GUIDE encounters out-of-memory issues on both datasets, while GraphEraser and GIF also experience out-of-memory issues on the ogbn-products dataset. ETR demonstrates superior performance in terms of model utility, unlearning efficiency, and unlearning efficacy on large-scale graphs. Specifically, ETR shows an average utility improvement of 9.13% over MEGU, reduces runtime by thousands of times compared to Retrain, and decreases memory overhead by 7 times compared to Retrain and 15 times compared to MEGU. In terms of unlearning efficacy, ETR performs best in most cases. Overall, ETR achieves better performance and efficiency compared to other methods.

Hyperparameter Analysis (RQ3)

The hyperparameters m and λ are crucial for ETR's performance. The value of m determines how much the parameters are modified, thereby influencing the extent of model forgetting. The value of λ controls the extent of model rectification. In this section, we investigate their impact on ETR's performance using the PubMed, CiteSeer, and Cora datasets.

We investigate the effect of different m values on the Erase strategy, with results shown in Figure 3. It can be observed that increasing m generally improves model performance, but performance declines if m becomes too large. This is because larger m values help the model forget unlearned samples effectively, while excessively large values can also lead to the loss of essential knowledge. Typically, m values between 8‰ and 16‰ provide the best results for the Erase strategy. Notably, the model performs better compared to $m = 0$, confirming that the Erase strategy effectively forgets unlearned samples and supports Theorem 1.

We tune λ within the range of [0, 0.9] to investigate how it affects ETR's performance, with results shown in Figure 3. It can be observed that ETR's performance generally improves as λ increases. However, performance declines if λ becomes too large. This is because larger λ values aid in model rectification, while excessively large values can also lead to excessive rectification. Optimal performance for ETR is typically achieved with λ values between 0.3 and 0.5.

Ablation Studies (RQ4)

This section investigates the effectiveness of different strategies in ETR. The Erase strategy is crucial for model forgetting, while Rectify is critical for maintaining model utility.

Method	PubMed	CiteSeer	Cora	CS	Physics
w/ mask	88.91	77.00	82.51	92.44	**95.80**
w/o mes	89.30	76.91	82.44	92.53	95.70
ETR	**89.48**	**77.09**	**83.54**	**92.72**	95.71

Table 5: Performance comparison between model variants.

Metric	PubMed	CiteSeer	Cora	CS	Physics
AD	5.7E-06	8.1E-07	1.7E-06	5.3E-06	1.6E-04
RD	0.23	0.09	0.07	0.36	0.22

Table 6: The difference between the approximate gradient and the true gradient.

Results in Figure 3 indicate that increasing m improves performance compared to $m = 0$, demonstrating the effectiveness of the Erase strategy in forgetting unlearned samples. Similarly, increasing λ improves performance compared to $\lambda = 0$, indicating that the Rectify strategy effectively enhances the model performance on the remaining dataset.

To further investigate the effectiveness of the Erase strategy, we compare ETR with two model variants, as shown in Table 5. Specifically, "w/ mask" refers to using the masking strategy, while "w/o mes" indicates not considering the impact of message propagation. The results show that ETR outperforms both variants in most cases, except for a slight decrease on the Physics dataset compared to "w/ mask". This demonstrates that parameter editing is more effective than just applying a mask and emphasizes the importance of considering message propagation in graph unlearning.

In the Rectify stage, we use the induced graph of unlearned samples to approximate the gradient of the model on the remaining dataset. We measure the difference of this approximation using Absolute Difference (AD) and Relative Difference (RD), as shown in Table 6. The results show that both differences are minimal, indicating that the gradient approximation ensures both efficiency and accuracy.

Conclusion

In this paper, we investigate cost-effective graph unlearning and proposes the ETR method, which removes the influence of unlearned samples while preserving model performance on the remaining data. In the Erase stage, we use the Fisher Information Matrix to identify parameters crucial for unlearned samples and their impact on connected samples, then modify them to forget both the unlearned samples and their influence. In the Rectify stage, we leverage the induced graph of unlearned samples to approximate the model's gradient on the remaining data, which is then used to enhance model performance. Extensive experiments demonstrate that ETR achieves strong performance in both efficiency and effectiveness. Future work will explore unlearning partial attributes in multi-attribute samples and partial objectives in multi-task settings (Liu and Tsang 2015; Liu, Tsang, and Müller 2017; Liu et al. 2019), where strong dependencies between different attributes and different tasks pose challenges, especially in graph-based scenarios, where there are also strong dependencies between samples.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62276277, 92370204), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2022B1515120059), the National Key R&D Program of China (Grant No. 2023YFF0725004), the Guangzhou-HKUST(GZ) Joint Funding Program (Grant No. 2023A03J0008), and the Education Bureau of Guangzhou Municipality.

References

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine Unlearning. In *SP*, 141–159.

Chen, C.; Zheng, S.; Chen, X.; Dong, E.; Liu, X. S.; Liu, H.; and Dou, D. 2021. Generalized Data Weighting via Class-Level Gradient Manipulation. In *NeurIPS*, 14097–14109.

Chen, M.; Zhang, Z.; Wang, T.; Backes, M.; Humbert, M.; and Zhang, Y. 2022. Graph Unlearning. In *CCS*, 499–513.

Chen, Z.; Li, P.; Liu, H.; and Hong, P. 2023. Characterizing the Influence of Graph Elements. In *ICLR*.

Cheng, J.; Dasoulas, G.; He, H.; Agarwal, C.; and Zitnik, M. 2023. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *ICLR*.

Chien, E.; Pan, C.; and Milenkovic, O. 2023. Efficient Model Updates for Approximate Unlearning of Graph-Structured Data. In *ICLR*.

Foster, J.; Schoepf, S.; and Brintrup, A. 2024. Fast Machine Unlearning without Retraining through Selective Synaptic Dampening. In *AAAI*, 12043–12051.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, 1263–1272.

Golatkar, A.; Achille, A.; and Soatto, S. 2020a. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *CVPR*, 9301–9309.

Golatkar, A.; Achille, A.; and Soatto, S. 2020b. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In *ECCV (29)*, 383–398.

Guo, C.; Goldstein, T.; Hannun, A. Y.; and van der Maaten, L. 2020. Certified Data Removal from Machine Learning Models. In *ICML*, 3832–3842.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

Kay, S. M. 1993. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

Li, X.; Zhao, Y.; Wu, Z.; Zhang, W.; Li, R.; and Wang, G. 2024. Towards Effective and General Graph Unlearning via Mutual Evolution. In *AAAI*, 13682–13690.

Liu, W.; and Tsang, I. W. 2015. On the Optimality of Classifier Chain for Multi-label Classification. In *NeurIPS*, 712–720.

Liu, W.; and Tsang, I. W. 2017. Making Decision Trees Feasible in Ultrahigh Feature and Label Dimensions. *J. Mach. Learn. Res.*, 18: 81:1–81:36.

Liu, W.; Tsang, I. W.; and Müller, K. 2017. An Easy-to-hard Learning Paradigm for Multiple Classes and Multiple Labels. *J. Mach. Learn. Res.*, 18: 94:1–94:38.

Liu, W.; Xu, D.; Tsang, I. W.; and Zhang, W. 2019. Metric Learning for Multi-Output Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2): 408–422.

Maini, P.; Mozer, M. C.; Sedghi, H.; Lipton, Z. C.; Kolter, J. Z.; and Zhang, C. 2023. Can Neural Network Memorization Be Localized? In *ICML*, 23536–23557.

Nguyen, T. T.; Huynh, T. T.; Nguyen, P. L.; Liew, A. W.; Yin, H.; and Nguyen, Q. V. H. 2022. A Survey of Machine Unlearning. *CoRR*, abs/2209.02299.

Said, A.; Derr, T.; Shabbir, M.; Abbas, W.; and Koutsoukos, X. D. 2023. A Survey of Graph Unlearning. *CoRR*, abs/2310.02164.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR*, abs/1811.05868.

Tarun, A. K.; Chundawat, V. S.; Mandal, M.; and Kankanhalli, M. 2023. Fast Yet Effective Machine Unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–10.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR (Poster)*.

Wang, C.; Huai, M.; and Wang, D. 2023. Inductive Graph Unlearning. In *USENIX Security Symposium*, 3205–3222.

Wu, J.; Yang, Y.; Qian, Y.; Sui, Y.; Wang, X.; and He, X. 2023a. GIF: A General Graph Unlearning Strategy via Influence Function. In *WWW*, 651–661.

Wu, K.; Shen, J.; Ning, Y.; Wang, T.; and Wang, W. H. 2023b. Certified Edge Unlearning for Graph Neural Networks. In *KDD*, 2606–2617.

Xu, H.; Zhu, T.; Zhang, L.; Zhou, W.; and Yu, P. S. 2024. Machine Unlearning: A Survey. *ACM Comput. Surv.*, 56(1): 9:1–9:36.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*, 40–48.

Yang, Z.; Han, J.; Wang, C.; and Liu, H. 2024. Erase then Rectify: A Training-Free Parameter Editing Approach for Cost-Effective Graph Unlearning. *CoRR*, abs/2409.16684.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.