

Towards Effective and General Graph Unlearning via Mutual Evolution

Xunkai Li^{*1}, Yulin Zhao^{*3}, Zhengyu Wu¹,
Wentao Zhang^{4,5}, Rong-Hua Li^{1,2}, Guoren Wang¹

¹Beijing Institute of Technology, Beijing, China

²Shenzhen Institute of Technology, Shenzhen, China

³Shandong University, Shandong, China

⁴Peking University, Beijing, China

⁵National Engineering Laboratory for Big Data Analytics and Applications, Beijing, China

{cs.xunkai.li, yulinzhao233}@gmail.com, Jeremywzy96@outlook.com,

wentao.zhang@pku.edu.cn, lironghuabit@126.com, wanggrbit@gmail.com

Abstract

With the rapid advancement of AI applications, the growing needs for data privacy and model robustness have highlighted the importance of machine unlearning, especially in thriving graph-based scenarios. However, most existing graph unlearning strategies primarily rely on well-designed architectures or manual process, rendering them less user-friendly and posing challenges in terms of deployment efficiency. Furthermore, striking a balance between unlearning performance and framework generalization is also a pivotal concern. To address the above issues, we propose **Mutual Evolution Graph Unlearning (MEGU)**, a new mutual evolution paradigm that simultaneously evolves the predictive and unlearning capacities of graph unlearning. By incorporating aforementioned two components, MEGU ensures complementary optimization in a unified training framework that aligns with the prediction and unlearning requirements. Extensive experiments on 9 graph benchmark datasets demonstrate the superior performance of MEGU in addressing unlearning requirements at the feature, node, and edge levels. Specifically, MEGU achieves average performance improvements of 2.7%, 2.5%, and 3.2% across these three levels of unlearning tasks when compared to state-of-the-art baselines. Furthermore, MEGU exhibits satisfactory training efficiency, reducing time and space overhead by an average of 159.8x and 9.6x, respectively, in comparison to retraining GNN from scratch.

Introduction

Recently, graphs have been a trending AI topic. To enable graph learning with human-like intelligence, graph neural networks (GNNs) have achieved state-of-the-art performance in node- (Chen et al. 2020; Zhang et al. 2022), link- (Cai et al. 2021; Tan et al. 2023), and graph-level (Xu et al. 2019; Yang et al. 2022) scenarios.

As most academic works center on training GNN under experimental settings, its real-world implementation often requires extra modifications to meet practical demands, such as the deletion of graph elements. It is critical in practicing data-driven AI applications, where the presence of irrelevant, inaccurate, or privacy-sensitive data elements can

significantly impact the predictive performance of trained GNNs. Two motivations behind the real-world AI deployment of data deletion can be further illustrated as follows: (i) **Data privacy**: Deletion of elements takes into account the "right to be forgotten" in machine learning, enabling users to request the removal of sensitive elements used for training. As a result, this changes node presence and helps protect data privacy. (ii) **Model robustness**: The presence of industry-related noise and fluctuation compromises data quality. By employing data deletion, the impact of such noise on contaminating node attributes and edge presence can be mitigated, leading to enhanced model robustness.

To achieve data deletion, machine unlearning (MU) is introduced, aiming to enable trained models to forget the influence of unlearning entities (deleted elements). In general, the MU strategy contains two crucial modules for practical demands: (i) **Predictive module**: It maintains predictive performance for non-unlearning entities; (ii) **Unlearning module**: It removes the influence of unlearning entities. Given the distinctive graph-based challenges in real-world deployments, addressing fundamental tasks of graph unlearning (GU) involves designing strategies for feature, node, and edge-level operations. Compared to MU in computer vision, GU poses unique challenges since the extensive entity interactions by GNN training (i.e., message-passing). A naive approach is to retrain the model from scratch but it suffers from the high costs of frequent unlearning requests.

Recently, some approximate-based GU methods are proposed. GIF (Wu et al. 2023) establishes the graph influence function to capture the relationship between data variations and model weights, and certified GU approaches (Chien, Pan, and Milenkovic 2022, 2023) propose a theoretical framework for approximate unlearning in linear GNNs. These methods mainly focus on the unlearning module but overlook the predictive module. As a result, although these methods offer high flexibility, related research (Mitchell et al. 2022) highlights potential compromises in their practical performance due to limited consideration for non-unlearning entities. Meanwhile, seeking a balanced trade-off between generalization boundaries and performance remains challenging in real world deployment. Other GU

^{*}These authors contributed equally.

Methods	Types	Mod. Agn.?	Pre. Per.?	Con. Tra.?	Dep. Eff.?
GIF (2023)	Appro.	✓	✗	✓	✓
CGU (2022)	Appro.	✗	✗	✓	✗
GUIDE (2023)	Learn.	✓	✓	✓	✗
Proj. (2023)	Learn.	✗	✓	✗	✓
Del. (2023)	Learn.	✓	✓	✗	✓
Era. (2022)	Learn.	✓	✓	✓	✗
MEGU (Ours)	Learn.	✓	✓	✓	✓

Table 1: A summary of recent GU studies.

approaches (Chen et al. 2022; Cong and Mahdavi 2023a; Cheng et al. 2023; Wang, Huai, and Wang 2023) introduce learnable mechanisms to adjust the original model or output for non-unlearning entities while eliminating the impact of unlearning entities. However, their predictive and unlearning capabilities often rely on well-designed architectures and handcrafted mechanisms, leaving room for improvement.

Building upon this, we review recent GU methods in Table 1 and suggest that a successful GU method should be capable of both handling unlearning requests at any time and being applicable to any backbone model (Model Agnostic). Hence, it should not only generate predictions that prioritize the performance of non-unlearning entities (Preserve Performance) but also possess the ability to adjust the trained model and continue training (Continue Training). Notably, the focus should be on designing these processes with a priority on mitigating the impact of unlearning entities. Furthermore, considering the real-world deployment requirements, they should demonstrate high efficiency in the both training and inference process (Deploy Efficiency).

Our contributions. (1) *New Perspective.* In this paper, we first emphasize the constraints of current GU strategies from a new perspective involving two distinct modules in Table 1. (2) *New Method.* Building upon this, we propose **Mutual Evolution Graph Unlearning (MEGU)**, which comprises original model-based predictive module and linear unlearning module to adjust the original model and generate predictions for non-unlearning entities, respectively. From the mutual evolution perspective, the effectiveness of the predictive module in eliminating the influence of unlearning entities relies on the forgetting capability of the unlearning module, and the reasoning capability of the predictive module is essential for the unlearning module to generate reliable predictions. (3) *SOTA Performance.* Extensive experiments on 9 benchmark datasets demonstrate that MEGU achieves not only state-of-the-art performance but also high training efficiency and scalability. Especially, MEGU outperforms GNNDelete (Cheng et al. 2023) by a margin of 2.8%-6.4% in terms of predictive accuracy, while achieving up to 4.5x-7.2x training speedups, respectively.

Preliminaries

Problem Formalization

In this work, we focus on the semi-supervised node classification task based on the topology of labeled set \mathcal{V}_L and

unlabeled set \mathcal{V}_U , and the nodes in \mathcal{V}_U are predicted with the supervised by \mathcal{V}_L . Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ with $|\mathcal{V}| = n$ nodes, $|\mathcal{E}| = m$ edges, and $\mathcal{X} = \mathbf{X}$. The feature matrix is $\mathbf{X} = \{x_1, \dots, x_n\}$ in which $x_v \in \mathbb{R}^f$ represents the feature vector of node v , and f represents the dimension of the node attributes, the adjacency matrix (including self-loops) is $\hat{\mathbf{A}} \in \mathbb{R}^{n \times n}$. Besides, $\mathbf{Y} = \{y_1, \dots, y_n\}$ is the label matrix, where $y_v \in \mathbb{R}^{|\mathcal{Y}|}$ is a one-hot vector and $|\mathcal{Y}|$ represents the number of the classes. In GU, after receiving unlearning request $\Delta\mathcal{G} = \{\Delta\mathcal{V}, \Delta\mathcal{E}, \Delta\mathcal{X}\}$ on original model parameterized by \mathbf{W} , the goal is to output the predictions of non-unlearning entities (i.e. \mathcal{V}_U) and adjusted model parameterized by \mathbf{W}^* , both with minimal impact from the unlearning entities. The typical unlearning requests include feature-level $\Delta\mathcal{G} = \{\emptyset, \emptyset, \Delta\mathcal{X}\}$, node-level $\Delta\mathcal{G} = \{\Delta\mathcal{V}, \emptyset, \emptyset\}$, edge-level $\Delta\mathcal{G} = \{\emptyset, \Delta\mathcal{E}, \emptyset\}$ in \mathcal{V}_L .

Graph Neural Networks

Motivated by spectral graph theory and deep neural networks, the concept of graph convolution is initially introduced in (Bruna et al. 2013). However, the computational complexity associated with eigenvalue decomposition hinders its deployment. To overcome this challenge, the Graph Convolutional Network (GCN) (Kipf and Welling 2017) is proposed, which approximates the convolution operator using the first-order approximation of Chebyshev polynomials. GCN propagates node information iteratively to neighboring nodes for label prediction. Building upon this framework, recent studies (Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018; Chen et al. 2020) have further optimized the model architectures, achieving remarkable performance improvements. Further research advancements on GNNs can be found in recent surveys (Zhou et al. 2022; Bessadok, Mahjoub, and Rekik 2022; Song et al. 2022).

Graph Unlearning

In this part, We provide an overview of recent advancements in GU. GraphEraser (Chen et al. 2022) attempts to partition the graph into multiple shards to handle unlearning requests within each shard. Building upon this, GUIDE (Wang, Huai, and Wang 2023) further optimizes the partitioning and shard aggregation strategies. However, their performance depends heavily on partitioning quality and aggregators. GraphEditor (Cong and Mahdavi 2023b) and Projector (Cong and Mahdavi 2023a) provide closed-form solutions with theoretical guarantees. However, their application is limited due to the linear assumption. Approximate-based methods (Chien, Pan, and Milenkovic 2023, 2022; Wu et al. 2023) have emerged as efficient solutions. However, as highlighted by MEND (Mitchell et al. 2022), the lack of consideration for non-unlearning entities may impact their practical performance. Meanwhile, balancing the trade-off during deployment between generalization and performance remains challenging. GNNDelete (Cheng et al. 2023) proposes layer-based unlearning operators to obtain predictions without adjusting the original trained model, but its deployment efficiency decreases with model depth and cannot handle unlearning requests for continue training.

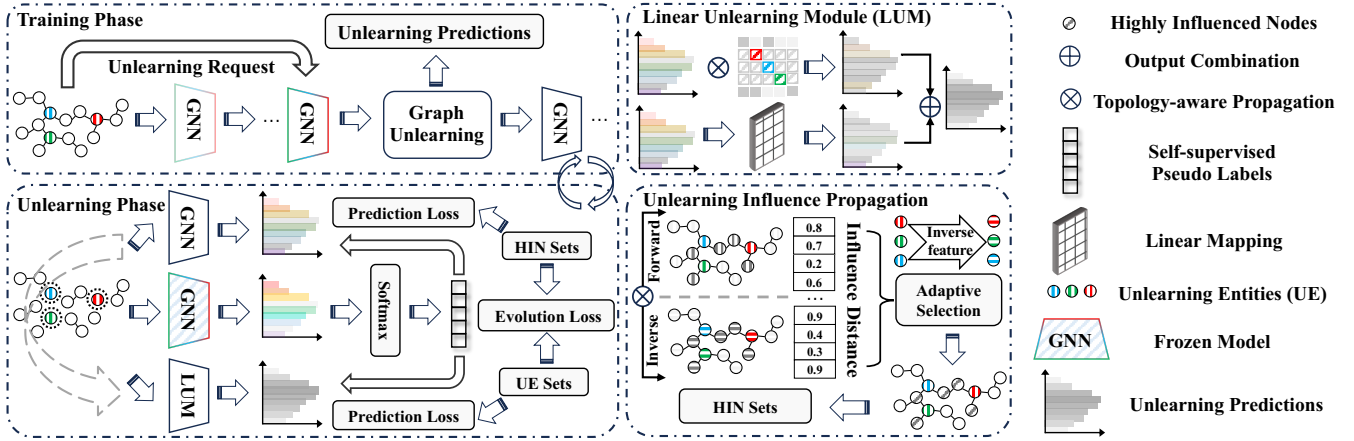


Figure 1: Overview of our proposed MEGU. Unlearning Prediction represents the prediction of non-unlearning entities.

Model Framework

In this section, we introduce MEGU, which provides a new paradigm for GU by deconstructing the MU targets. To begin with, we provide an overview of the MEGU pipeline and its intuitions. Then, considering the unique challenges posed by GNNs and aiming to achieve graph-based mutual evolution, we introduce adaptive high-influence neighborhood selection and topology-aware unlearning propagation. Building upon these technologies, the predictive module and unlearning module are trained in a topology-guided mutually boosting manner by a well-designed optimization objective.

Architecture Overview

As illustrated in Fig. 1, we initialize the predictive module with the original trained model. Throughout the unlearning process, its target is to adjust the original model under unlearning requirements while retaining the reasoning capability. This design preserves the original model’s predictive accuracy while efficiently achieving unlearning through an end-to-end learnable mechanism with minimal cost. Moreover, the adjusted original model can be further utilized for continued training, offering deployment flexibility. As for the unlearning module, its target is to generate predictions for non-unlearning entities based on the predictive module while offering forgetting capacity for model adjustment. This strategy minimizes the computational overhead associated with unlearning. From the mutual evolution perspective, the predictive module relies on the unlearning module’s forgetting ability, guiding the modification of the original trained model. Similarly, the unlearning module depends on the predictive module’s reasoning capability to generate reliable predictions. Consequently, these two modules mutually optimize each other within the unified MEGU framework.

For the three downstream unlearning tasks, our processing details are as follows: (1) Feature-level: we treat nodes as unlearning entities while preserving their topology; (2) Node-level: we consider nodes as unlearning entities and remove their related topological connections; (3) Edge-level: we consider connected nodes as unlearning entities but preserve their topology and remove the unlearning edge.

Adaptive High-influence Neighborhood Selection

Due to the rich interactions in the GNNs, we need to identify the nodes that are highly influenced by unlearning entities. This is pivotal in forming an optimization objective that preserves predictive accuracy while reducing unlearning entity impacts. Existing methods consider nodes within a fixed neighborhood of unlearning entities as highly influenced nodes (HIN). Unfortunately, they neglect the distinct roles of graph elements in topology-based propagation.

To address this issue, we propose adaptive high-influence neighborhood selection, which leverages the forward and inverse feature propagation based on the original topology to obtain smoothed features from two perspectives. Formally, the above process in l -layer original GNN can be defined as

$$\tilde{\mathbf{X}} = \hat{\mathbf{A}}^l \mathbf{X}, \quad \tilde{\mathbf{X}}' = \hat{\mathbf{A}}^l \mathbf{X}', \quad (1)$$

$$\mathbf{X}'_i = \mathbf{X}_i, \quad \mathbf{X}'_j = \mathbf{1} - \mathbf{X}_j, \quad \forall i \in \mathcal{V}/\Delta\mathcal{V}, \quad \forall j \in \Delta\mathcal{V},$$

where $\mathbf{1}$ is the 1-vector of size f and \mathbf{X}' is the inverse feature for unlearning entities. Meanwhile, taking into account that the original l -layer GNN aggregates information from the l -hop neighborhoods, we employ l -step feature smoothing by default. Notably, in the case of edge unlearning, we treat the two nodes connected by $\Delta\mathcal{E}$ as unlearning entities to perform inverse features. Intuitively, when we reverse the features of unlearning entities, it leads to significant changes in the smoothed features of HIN from two topology-based propagation perspectives. To quantify this difference, we introduce the following concept of *influence distance*, which serves as a measure to adaptively select HIN (see Alg. 1). By considering the unique structural properties of different entities, our approach effectively mitigates the bias that arises from treating all nodes within a fixed neighborhood equally.

Definition 1 (Influence Distance). The influence distance \mathcal{D}_k parameterized by node k and forward and inverse feature propagation results $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{X}}'$ is formally defined as

$$\forall k \in \mathcal{V}/\Delta\mathcal{V}, \quad \mathcal{D}_k = \text{Dis}(\tilde{\mathbf{X}}_k, \tilde{\mathbf{X}}'_k), \quad (2)$$

where $\tilde{\mathbf{X}}_k$ denotes the k^{th} row of $\tilde{\mathbf{X}}$, $\text{Dis}(\cdot)$ is a function positively relative with the difference, which can be implemented using Euclidean distance, cosine similarity, etc.

Topology-aware Unlearning Propagation

To achieve mutual evolution for two individual modules and improve final predictions in graph scenarios, we propose the topology-aware unlearning propagation based on the predictive module and non-unlearning entities \mathbf{A}^* , \mathbf{X}^* , where we remove the unlearning entities in \mathbf{A} , \mathbf{X} . This strategy considers both the topological structure and the self-supervised information \mathbf{L} from the predictive module, which effectively integrates the predictive and unlearning modules while upholding the homophily assumption to improve predictions. Specifically, its foundation lies in the expectation that connected nodes in a graph exhibit similar labels, aligning with the network’s inherent homophily or assortative characteristics. Thus, we can encourage smoothness over the distribution over labels by another label propagation (i.e. \mathbf{L}). Meanwhile, it introduces a novel paradigm for two module interaction in the GU process, which is formally expressed as

$$\begin{aligned} \mathbf{Y}(\hat{\mathbf{Y}}, \mathbf{E}(\mathbf{L})) &:= \mathbf{Y}_u = \hat{\mathbf{Y}}_u, \mathbf{Y}_v = \mathbf{G}(\hat{\mathbf{Y}}_v + \mathbf{G}(\mathbf{E}_v)), \\ \mathbf{E}(\mathbf{L}) &:= \mathbf{E}_u^{(0)} = \vec{0}, \mathbf{E}_v^{(0)} = \mathbf{L} - \hat{\mathbf{Y}}_v, \forall u \in \mathcal{V}_L, \forall v \in \mathcal{V}_U, \\ \mathbf{G}(\mathbf{T}) &:= \mathbf{T}_i^{(l)} = \alpha \mathbf{T}_i^{(0)} + (1-\alpha) \sum_{j \in \mathcal{N}_i^{(l)}} \frac{1}{\sqrt{\tilde{d}_i \tilde{d}_j}} \mathbf{T}_j^{(l-1)}, \end{aligned} \quad (3)$$

where \mathbf{E} denotes the error correction matrix. Building upon this, we adopt the approximate calculation for the personalized PageRank (Chien et al. 2021), where $\mathcal{N}_i^{(1)}$ denotes the one-hop neighbors of node i . Meanwhile, we set α according to datasets and backbone-based propagation step l by default to capture structural information. The aforementioned process can be regarded as the materialization of the unlearning module leveraging the reasoning capacity of the predictive module to generate reliable predictions. As depicted in Fig. 1, this thoughtful technology forms the unlearning module in MEGU, which generates final predictions \mathbf{Y}^* for non-unlearning entities. It is formally represented as

$$\begin{aligned} \mathbf{Y}^* &:= \mathbf{Y}^*(\hat{\mathbf{Y}}^*, \mathbf{E}(\hat{\mathbf{Y}}^*)), \\ \hat{\mathbf{P}} &= \text{Encoder}(\mathbf{A}^*, \mathbf{X}^*, \mathbf{W}^*), \hat{\mathbf{P}}^* = \mathbf{W}_u \hat{\mathbf{P}}, \\ \hat{\mathbf{Y}} &= \text{Softmax}(\hat{\mathbf{P}}), \hat{\mathbf{Y}}^* = \text{Softmax}(\hat{\mathbf{P}}^*), \end{aligned} \quad (4)$$

where $\text{Encoder}(\cdot)$ parameterized by \mathbf{W}^* is any adjusted original trained model in the predictive module, \mathbf{W}_u is the trainable linear unlearning operator.

Optimization Objective

Since the unlearning request occurs within the training set, we exclusively utilize self-supervised information during the unlearning process to prevent potential label leakage concerns. As illustrated in Fig 1, we freeze the original model at the time of receiving the unlearning request to provide self-supervised information $\tilde{\mathbf{Y}}$, preserving the reasoning and forgetting capacity of the predictive and unlearning module.

Specifically, the predictive module utilizes the cross-entropy (CE) loss based on the output of the frozen model to preserve its reasoning capability. Simultaneously, it leverages Kullback-Leibler divergence (KL) loss and the output

Algorithm 1: Adaptive HIN Selection

```

1: Initialize: HIN =  $\emptyset$ ,  $\omega = 0$ ,  $\epsilon = 0.1$ ,  $\mu = \text{True}$ ;
2: Execute forward and inverse feature propagation based
   on the Eq. (1) to obtain  $\tilde{\mathbf{X}}, \tilde{\mathbf{X}}'$ ;
3: Calculate cosine similarity-based influence distance  $\mathcal{D}$ 
   according to the Eq. (2);
4: while  $\mu$  do
5:   for node  $u$  in  $\Delta\mathcal{V}$ , node  $v$  in  $\mathcal{V}/\Delta\mathcal{V}$  do
6:     if  $\mathcal{D}_v \leq \epsilon$  and  $v \in \mathcal{N}_u^{(l)}$  then
7:       HIN = HIN  $\cup$   $v$ ;
8:     end if
9:   end for
10: Calculate the maximum  $\mathcal{D}_{\max}$  in HIN,  $\mu = \text{False}$ ;
11: if  $\mathcal{D}_{\max} \neq \omega$  then
12:    $\omega = \mathcal{D}_{\max}$ ,  $\epsilon = \epsilon + 0.1$ ,  $\mu = \text{True}$ ;
13: end if
14: end while

```

of the unlearning module to eliminate the impact of unlearning entities on the original model. Remarkably, benefiting from the initialization of the original model, the predictive module already possesses commendable predictive performance for non-unlearning entities. However, to mitigate the impact of unlearning entities, it is crucial to remove the related knowledge in HIN (KL loss) while maintaining their predictive accuracy (CE loss). Hence, we narrow down the optimization scope of the predictive module from all non-unlearning entities to HIN, which aligns with our dual objectives of unlearning and efficiency improvement.

$$\mathcal{L}_p = \sum_{u \in \text{HIN}} \mathcal{L}_{CE}(\hat{\mathbf{Y}}_u, \tilde{\mathbf{Y}}_u) + \sum_{v \in \text{HIN}} \mathcal{L}_{KL}(\hat{\mathbf{Y}}_v^*, \hat{\mathbf{Y}}_v). \quad (5)$$

For the unlearning module, it utilizes the reverse CE loss to enhance its forgetting capability for unlearning entities. Meanwhile, it leverages the KL loss and the output of the predictive module to ensure the predictive performance

$$\mathcal{L}_u = - \sum_{u \in \Delta\mathcal{V}(\mathcal{X}, \mathcal{E})} \mathcal{L}_{CE}(\hat{\mathbf{Y}}_u^*, \tilde{\mathbf{Y}}_u) + \sum_{v \in \Delta\mathcal{V}(\mathcal{X}, \mathcal{E})} \mathcal{L}_{KL}(\hat{\mathbf{Y}}_v, \hat{\mathbf{Y}}_v^*). \quad (6)$$

Based on Eq. (5) and Eq. (6), in the perspective of mutual evolution, we formulate the overall optimization objective in MEGU to achieve κ -based flexible unlearning

$$\mathcal{L} = \mathcal{L}_p + \kappa \mathcal{L}_u. \quad (7)$$

Experiments

In this section, we conduct a thorough evaluation of MEGU. We commence by introducing 9 benchmark datasets and baselines. Then, we present the methodology used to evaluate the effectiveness of GU. Details about the experimental setup can be found in (Li et al. 2023)A.1-A.4. In general, we aim to address following questions: **Q1:** Compared to existing GU strategies, can MEGU achieve state-of-the-art performance? **Q2:** If MEGU is effective, where do its reasoning and forgetting capabilities come from? **Q3:** Does MEGU really achieve mutual evolution between the predictive module and unlearning module? For more extended experiments and discussions please refer to (Li et al. 2023)A.5-A.6.

Bone	Strategy	Cora		CiteSeer		PubMed		Photo		Computer		CS		Physics	
		F1	T.	F1	T.	F1	T.	F1	T.	F1	T.	F1	T.	F1	T.
GCN	Retrain	85.6±.3	15	75.6±.2	41	86.5±.1	71	91.2±.1	39	83.1±.2	63	91.4±.1	44	95.2±.1	169
	LPA	42.1±.0	15	48.0±.0	16	63.7±.0	33	45.2±.0	18	38.2±.0	18	58.3±.0	24	65.3±.0	36
	KMeans	48.0±.0	15	39.6±.0	16	64.4±.0	32	54.4±.0	18	40.4±.0	18	67.0±.0	22	73.5±.0	34
	SR	79.2±.5	10	74.0±.1	12	85.2±.0	28	80.5±.1	5	74.8±.1	9.2	85.6±.1	11	91.6±.1	23
	Fast	79.0±.2	9	73.6±.0	12	85.1±.0	28	80.7±.0	5	75.9±.0	9	85.4±.0	11	91.4±.0	23
	GIF	83.8±.3	.3	73.9±.2	.4	85.4±.6	.5	89.8±.3	.3	83.2±.3	.3	90.5±.2	.4	93.8±.1	.5
	Delete	81.7±.6	2	72.8±.4	2	85.0±.4	4	88.6±.4	2	83.4±.2	2	90.7±.5	2	93.0±.6	3
	MEGU	85.2±.9	.2	75.8±.0	.2	86.9±.0	.3	92.2±.1	.2	85.6±.0	.2	92.0±.0	.3	95.9±.0	.6
GAT	Retrain	86.3±.5	17	77.3±.4	43	86.8±.2	81	91.8±.3	39	83.5±.3	63	91.5±.2	50	95.4±.2	199
	LPA	44.6±.0	23	48.5±.0	23	62.5±.0	55	48.7±.0	29	40.7±.0	28	61.3±.0	37	67.2±.0	63
	KMeans	48.3±.0	23	39.3±.0	24	64.8±.0	52	66.0±.0	28	43.0±.0	28	70.0±.0	36	74.0±.0	60
	SR	76.5±.5	15	74.1±.2	19	83.2±.0	38	81.6±.1	8	76.5±.2	10	84.9±.0	13	89.7±.1	30
	Fast	78.2±.3	16	74.2±.2	18	83.4±.1	38	80.7±.1	6	76.3±.2	9	84.8±.1	14	89.6±.0	28
	GIF	82.8±.6	.9	73.6±.2	.8	84.5±.1	.9	88.3±.2	.9	82.6±.3	2	88.3±.1	.9	92.2±.1	1.8
	Delete	83.0±.8	2	73.0±.5	2	84.7±.2	3	88.5±.4	2	82.0±.3	2	88.5±.4	2	92.4±.2	3
	MEGU	86.4±.1	.3	77.8±.1	.3	86.2±.0	.4	91.5±.1	.3	83.8±.1	.5	91.7±.1	.7	95.6±.1	1.5

Table 2: Transductive performance and training efficiency on the node unlearning. The best result is bold.

Bone	Strategy	PPI		Flickr	
		F1	Time	F1	Time
SAGE	Retrain	56.7±0.2	249	50.6±0.3	479
	GIF	54.2±0.2	1.4	48.7±0.4	1.7
	Delete	54.8±0.2	9.2	48.5±0.6	12.6
	MEGU	57.5±0.2	1.3	50.3±0.4	1.0
SAINT	Retrain	55.3±0.1	212	49.6±0.2	403
	GIF	53.3±0.0	0.5	48.1±0.5	1.0
	Delete	52.9±0.1	5.7	47.8±0.4	4.8
	MEGU	55.6±0.4	1.0	50.0±0.5	0.5
CI-GCN	Retrain	56.4±0.8	221	51.2±0.05	426
	GIF	53.2±1.7	0.6	48.7±0.7	1.0
	Delete	54.2±1.0	5.7	48.6±0.5	4.9
	MEGU	57.4±1.0	0.2	50.2±1.0	0.4

Table 3: Inductive performance on the node unlearning.

Strategy	PubMed		Flickr	
	Feature	Edge	Feature	Edge
Retrain	86.85±0.1	87.13±0.1	48.29±0.2	48.14±0.2
LPA	64.28±0.0	64.26±0.0	43.51±0.0	42.63±0.0
Kmeans	67.63±0.0	65.97±0.1	43.18±0.1	42.45±0.0
SR	83.73±0.1	82.25±0.0	46.90±0.0	47.02±0.0
Fast	83.54±0.0	82.32±0.1	46.78±0.1	46.93±0.1
CGU	79.70±0.1	78.31±0.0	OOT	OOT
GIF	83.05±0.0	82.10±0.1	47.09±0.1	47.04±0.2
Projector	80.79±0.1	81.64±0.1	47.06±0.1	47.13±0.1
Delete	83.86±0.1	82.17±0.1	47.12±0.1	47.22±0.0
MEGU	86.95±0.0	86.80±0.0	48.35±0.2	48.10±0.2

Table 4: Predictive performance with SGC backbone.

Experimental Setup

Datasets. We split all datasets following the guidelines of recent GU approaches (Cheng et al. 2023; Wu et al. 2023),

which randomly split nodes into 80% for training and 20% for testing. For a comprehensive overview of datasets and baselines, please refer to (Li et al. 2023)A.1.

Baselines. We compare MEGU with the following baselines: (1) GraphEraser and GUIDE; (2) CGU and GIF; (3) Projector and GNNDelete. For details regarding the baselines, please refer to (Li et al. 2023)A.2. Unless otherwise stated, we adopt GCN as the backbone and the node unlearning by default to present results. Notably, we experiment with multiple backbone GNNs in separate modules to validate the generalizability of MEGU and avoid complex charts. To alleviate the randomness and ensure a fair comparison, we repeat each experiment 10 times to present performance. We customize the training epochs for each GU strategy to their respective optimal values, ensuring convergence and reporting the average training time (second).

Unlearning Targets. In our experiments, GU requests are categorized as follows: (1) Feature-level: We randomly select 10% of nodes from the training set and mask the full-dimensional features. (2) Node-level: We randomly select 10% of nodes from the training set and remove related edges. (3) Edge-level: We randomly select 10% of edges from the training graph. Then, the two nodes connected by the unlearning edges are considered unlearning entities. After that, we evaluate the performance of the predictive module using the Micro-F1 score for the semi-supervised node classification, being a harmonic mean of precision and recall, which places greater emphasis on each individual sample. As a result, it effectively captures instances of classification errors, making it well-suited for evaluating such unlearning cases. Additionally, to verify the forgetting ability of GU strategies, we adopt the Edge Attack. In this strategy, we randomly select two nodes with different labels as targets for adding noisy edges, which are treated as unlearning entities. Intuitively, as a method achieves better unlearning, it tends to effectively mitigate the negative impact of noisy edges on predictive performance, thus ensuring robustness.

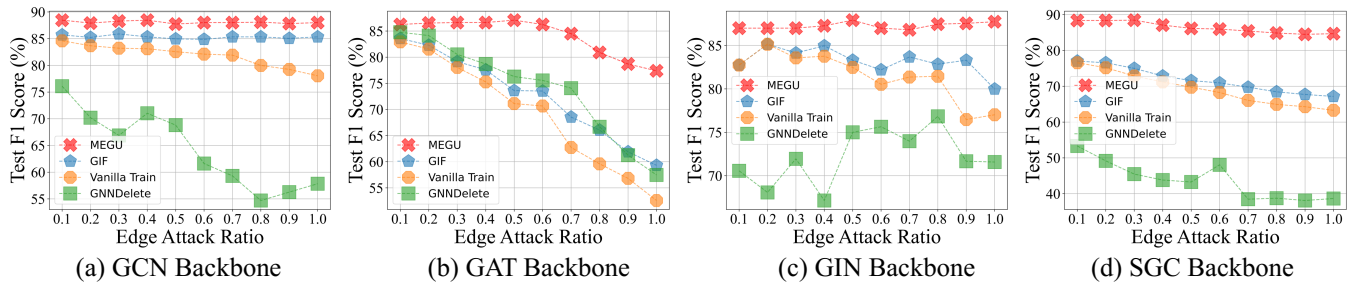


Figure 2: Edge Attack performance on Cora. The x-axis is the ratio of noisy edges to the existing edges.

Bone	Module	Cora		CiteSeer	
		Feat	Edge	Feat	Edge
GCN	w/o HIN	87.8±.6	87.3±.4	74.6±.2	76.7±.3
	w/o TUP	87.3±.4	87.6±.3	74.5±.2	76.0±.1
	MEGU	88.5±.3	78.5±.3	75.8±.3	77.8±.1
GAT	w/o HIN	83.2±.3	84.8±.5	72.8±.3	73.8±.2
	w/o TUP	82.8±.2	83.4±.1	73.1±.1	73.6±.1
	MEGU	84.0±.3	85.3±.2	74.3±.2	74.8±.1
GIN	w/o HIN	85.1±.3	84.3±.4	74.4±.2	74.9±.2
	w/o TUP	85.7±.1	83.4±.2	74.3±.1	74.7±.1
	MEGU	86.5±.2	75.1±.1	75.5±.2	75.6±.1

Table 5: Ablation study on three representative backbones.

Performance Comparison

To answer **Q1** from the perspective of the predictive module, we report the transductive performance in Table 2, which validates that MEGU consistently outperforms baselines. For instance, MEGU exhibits an average improvement of 2.4% over the SOTA approach in Cora. Notably, the underperforming results of GraphEraser and GUIDE align with their original papers and are likely attributed to the heavy reliance on the partition quality, making them less suitable for scenarios involving substantial element forgetting. Besides, the results presented in Table 3 consistently demonstrate the superior performance of MEGU over all baselines in the inductive setting, underscoring MEGU’s remarkable ability to predict unseen nodes. Furthermore, we include GU baselines relying on linear model assumptions in Table 4. Experimental outcomes demonstrate that MEGU outperforms the most competitive methods, achieving average performance gains of 1.7% and 2.3% for feature and edge unlearning. This observation demonstrates that MEGU can achieve satisfactory performance without relying on a powerful backbone. Its potential for widespread application in linear GNNs is evident, showcasing its generalizability.

Remarkably, in some cases, MEGU outperforms training GNN from scratch (Retrain). This is because unlearning requests involve the removal of existing graph entities, which could have a negative impact on the Retrain. Fortunately, MEGU’s mutual evolution mechanism has the capability to capture such data variations and can mitigate the performance limitations through its optimization framework.

Unlearning Capability

To answer **Q1** from the perspective of the unlearning module, we visualize the forgetting capability of various GU strategies under the Edge Attack setting through Fig. 2. Intuitively, as the number of noisy edges increases, the accuracy of the unlearning predictions tends to decline. Therefore, for a clear comparison, we introduce vanilla train, a baseline retrained directly on the noisy graph. In the context of Edge Attack, we treat noisy edges as unlearning entities. If a GU approach possesses robust unlearning capabilities, it can mitigate the adverse effects caused by noisy edges, thereby ensuring consistent and satisfactory performance. From the experimental results, we observe that GNNDelete and GIF do not consistently achieve optimal unlearning performance, whereas MEGU consistently outperforms other baselines in terms of unlearning abilities. This advantage is particularly prominent in scenarios where GCN, GIN, and SGC are employed as backbones. Notably, GAT, which heavily relies on edge-based attention mechanisms for information aggregation, is more susceptible to the negative impact of edge attacks, resulting in performance degradation.

Training Efficiency

Since GU strategies allow for efficient inference through quick forward computation post-training. Thus, we report the average training time in Table 2 and Table 3. In this regard, the pre-training time of the backbone is not included in the report but we incorporate the time required for shard partitioning. Notably, to ensure a fair comparison, we customize the training epochs for each GU strategy, guaranteeing model convergence and optimal performance. According to the results, our findings are as follows: (1) Shard-based GU methods and retraining GNN from scratch incur significantly long training time; (2) Benefiting from the mutual evolution, MEGU achieves model convergence and superior performance within a much shorter time frame (30 - 50 epochs) compared to other strategies. (e.g., GNNDelete requires over 200 epochs) This observation is also validated by the experimental results presented in Fig. 3. Additionally, Table 4 demonstrates that CGU encounters the OOT (Out of Time) error when dealing with relatively larger-scale graphs (i.e. Flickr), with instances of runtime exceeding 3600 seconds. This arises due to the substantial computational overhead inherent in the process of performing original model corrections based on the gradient Hessian matrix.

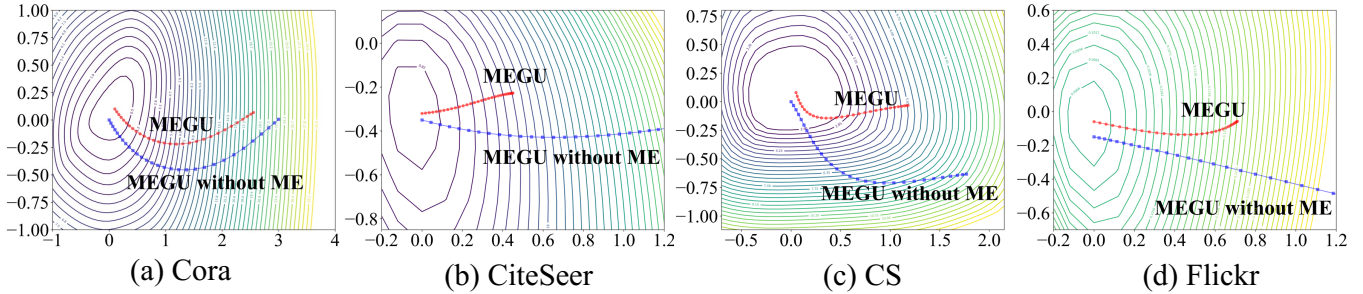


Figure 3: The training trajectories of MEGU and its variants without the mutual evolution design on the same loss landscape.

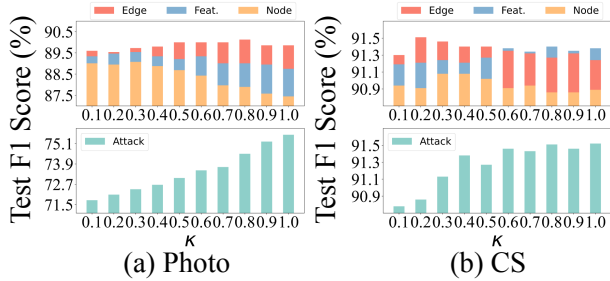


Figure 4: Sensitivity analysis on GAT backbone.

Ablation Study and Sensitivity Analysis

To answer **Q2**, we investigate the contributions of Adaptive HIN Selection (Ada. HIN) and Topology-aware Unlearning Propagation (Topo. UP) in MEGU. For Ada. HIN, it constructs a tailored loss function for the predictive module. This ensures the preservation of the predictive module’s reasoning ability. Results in Table 5 show significant performance improvement with Ada. HIN. For instance, in the CiteSeer feature unlearning case using GAT as the backbone, F1 Score increases from 72.8% to 74.3%. For Topo. UP, it integrates both modules to generate predictions and strengthens their interaction through the mutual evolution loss. Experimental results in Table 5 highlight Topo. UP’s effectiveness in enhancing prediction quality for non-unlearning entities, especially in the GAT backbone. This aligns with our intent to leverage the topology for GU. Topo. UP’s propagated features excel in capturing interactions across receptive field sizes, aided by self-supervision.

In this part, we present the sensitivity analysis in Fig. 4 to further answer **Q2** from the perspective of the hyperparameter settings (see Eq. (7)). Based on the experimental results shown in Fig. 4, we notice that MEGU’s predictive performance on non-unlearning entities in feature, node, and edge-level downstream tasks tends to decrease or exhibit unstable fluctuations with increasing κ . This outcome is attributed to the dilution effect on \mathcal{L}_p , which aims to uphold the predictive strength of the predictive module. Conversely, as the emphasis on MEGU’s unlearning ability represented by \mathcal{L}_u grows, its performance against Edge Attack progressively improves. These findings offer practical intuition for selecting an appropriate κ in real-world scenarios.

Mutual Evolution in Graph Unlearning

To answer **Q3**, we visualize the convergence insights into MEGU and its non-mutual evolution variant (MEGU without ME) in the same loss landscape (Li et al. 2018). Fig. 3 displays the training trajectories of these two variants, illustrating the convergence states under different training frameworks. In our experimental setup, MEGU without ME implies that the predictive module and unlearning module are independent. Specifically, we remove the additional supervision signal $\hat{\mathbf{P}}$ provided by the predictive module in topology-aware unlearning propagation, as well as the KL Loss that encourages interaction between these two modules in the optimization objective. At this time, $\hat{\mathbf{P}}^*$ in Eq. (3) is generated by the original frozen model. Building upon this, we observe that the design of mutual evolution significantly reduces the convergence difficulty and accelerates the convergence speed. This can be validated by the distance between the initial training trajectory point and the global optimal center point, as well as the trajectory itself. Moreover, this observation further elucidates the reason behind MEGU’s high training efficiency shown in Table 2 and Table 3, as it achieves optimal performance with a minimal number of training epochs. In a nutshell, the mutual evolution-based GU framework not only mitigates the impact of unlearning entities while improving predictions for non-unlearning entities but also maintains efficient computational performance and flexibility.

Conclusion

In this paper, we first address the data removal requirements in graph-based AI applications and provide a new perspective of two crucial modules to analyze the existing GU approaches. Building upon this, we provide reasonable analysis for the essential conditions that GU should satisfy, as illustrated in Table 1. Then, we propose a new framework to achieve effective and general GU via a mutual evolution design. The key insight of our approach lies in leveraging the predictive module’s inference capability and the unlearning module’s forgetting ability within a unified optimization framework, enabling mutual benefits between the two modules. A promising direction for future GU studies is to explore traceable message-passing mechanisms to further mitigate the impact of unlearning entities and improve predictive performance, allowing both modules to benefit from it.

Acknowledgments

This work was partially supported by (I) the National Key Research and Development Program of China 2021YFB3301301, (II) NSFC Grants U2241211, 62072034, and (III) High-performance Computing Platform of Peking University. Rong-Hua Li is the corresponding author of this paper.

References

- Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, KDD*, 2623–2631.
- Bessadok, A.; Mahjoub, M. A.; and Rekik, I. 2022. Graph neural networks in network neuroscience. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5): 5833–5848.
- Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2013. Spectral Networks and Locally Connected Networks on Graphs. *Computer Science*.
- Cai, L.; Li, J.; Wang, J.; and Ji, S. 2021. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning, ICML*.
- Chen, M.; Zhang, Z.; Wang, T.; Backes, M.; Humbert, M.; and Zhang, Y. 2022. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS*, 499–513.
- Cheng, J.; Dasoulas, G.; He, H.; Agarwal, C.; and Zitnik, M. 2023. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *The Eleventh International Conference on Learning Representations, ICLR*.
- Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C.-J. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, KDD*, 257–266.
- Chien, E.; Pan, C.; and Milenkovic, O. 2022. Certified Graph Unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.
- Chien, E.; Pan, C.; and Milenkovic, O. 2023. Efficient model updates for approximate unlearning of graph-structured data. In *The Eleventh International Conference on Learning Representations, ICLR*.
- Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations, ICLR*.
- Cong, W.; and Mahdavi, M. 2023a. Efficiently Forgetting What You Have Learned in Graph Representation Learning via Projection. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 6674–6703. PMLR.
- Cong, W.; and Mahdavi, M. 2023b. GraphEditor: An Efficient Graph Representation Learning and Unlearning Approach. <https://openreview.net/forum?id=tyvshLxFUtP>.
- Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems, NeurIPS*.
- Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR*.
- Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems, NeurIPS*, 31.
- Li, X.; Zhao, Y.; Wu, Z.; Zhang, W.; Li, R.-H.; and Wang, G. 2023. MEGU Technical Report. In <https://github.com/xkLi-Allen/MEGU>.
- Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2022. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR*.
- Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.
- Song, Z.; Yang, X.; Xu, Z.; and King, I. 2022. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*.
- Tan, Q.; Zhang, X.; Liu, N.; Zha, D.; Li, L.; Chen, R.; Choi, S.-H.; and Hu, X. 2023. Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM*, 625–633.
- Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *International Conference on Learning Representations, ICLR*.
- Wang, C.-L.; Huai, M.; and Wang, D. 2023. Inductive Graph Unlearning. *arXiv preprint arXiv:2304.03093*.
- Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning, ICML*.
- Wu, J.; Yang, Y.; Qian, Y.; Sui, Y.; Wang, X.; and He, X. 2023. GIF: A General Graph Unlearning Strategy via Influence Function. In *Proceedings of the ACM Web Conference, WWW*, 651–661.
- Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations, ICLR*.
- Yang, M.; Shen, Y.; Li, R.; Qi, H.; Zhang, Q.; and Yin, B. 2022. A new perspective on the effects of spectrum in graph neural networks. In *International Conference on Machine Learning, ICML*, 25261–25279. PMLR.
- Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML*, 40–48.
- Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2020. Graphsaint: Graph sampling based inductive learning method. In *International conference on learning representations, ICLR*.

Zhang, W.; Yin, Z.; Sheng, Z.; Li, Y.; Ouyang, W.; Li, X.; Tao, Y.; Yang, Z.; and Cui, B. 2022. Graph Attention Multi-Layer Perceptron. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD*.

Zhou, Y.; Zheng, H.; Huang, X.; Hao, S.; Li, D.; and Zhao, J. 2022. Graph Neural Networks: Taxonomy, Advances, and Trends. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(1): 1–54.