A OUTLINE

A.1 Systematic Review of GU baselines

GU remains a burgeoning field with numerous research gaps. To advance its future development and highlight the motivation of our approach, we conduct a systematic review of most existing GU strategies according to our proposed novel taxonomies in Table 6.

Model Agnostic. It is well-known that quantifying the impact of UE on Non-UE is considerably more challenging in graph-based scenarios than in computer vision due to the intricate interactions among graph entities during model training. This complexity makes UE-corresponding knowledge removal difficult, especially in the complex GNN architectures where gradient flows are well hidden. To ensure certified or exact data removal with a strict theoretical guarantee, some existing GU strategies simplify the default model architecture to a linear-based GNN to satisfy necessary assumptions, designing the corresponding unlearning algorithms through exhaustive derivations. Although they achieve satisfactory theoretical results, the inherent generalization limitations of linear models persist, preventing effective deployment in practical applications.

Request Agnostic. The inherent complexity and interconnected nature of graph-structured data lead to diverse data removal requests in GU, as formally characterized in Sec. 2.1. To address specific graph-based data removal scenarios while maintaining optimal unlearning efficacy and predictive performance, several existing approaches have developed specialized GU algorithms tailored to particular types of graph entity deletion requests, such as edge unlearning or node unlearning. While these specialized methods demonstrate effectiveness within their respective domains, the development of a general-purpose GU framework that can handle various types of graph entity removal requests remains an essential research direction. Furthermore, a deeper investigation into the potential connections and dependencies among different graph entities (e.g., nodes, edges, and features) could reveal more effective and unified GU paradigms, potentially leading to significant improvements in both computational efficiency and comprehensive performance metrics across diverse graph learning scenarios.

Efficient & Direct Model Updates. As the most important target of GU, Model Update aims to modify the original model parameterized by $\mathbf{W}$ into a new model parameterized by $\mathbf{W}^\star$. This modification ensures that $\mathbf{W}^\star$ forgets the gradient-driven knowledge contributed by UE during model training and eliminates the influence of UE on Non-UE. To achieve this goal, existing GU strategies have proposed various model update mechanisms. We propose that the following methods, which possess Efficient & Direct characteristics, are more advantageous for real-world deployment: (1) Projection-based methods *directly* project the original model weights into specific sub-spaces through optimized constraints. (2) Gradient-based methods *directly* update trainable parameters based on gradient convergence analysis. (3) Learning-based methods (fine-tuning) *directly* modify the original model using fine-tuning mechanisms. These methods *efficiently* achieve Model Update, allowing the original model to continue training, offering greater flexibility.

In contrast, the following methods have limitations: (1) Partition-based methods require retraining specific partitions and a partition output aggregation module. *Limitations: These methods make them difficult to deploy effectively in practical applications when deletion requests are frequent.* (2) Learning-based methods (deletion module) propose freezing the original model and injecting an additional trainable module to achieve unlearning. *Limitations: These methods do not update the original model, preventing further training.*

Inference Protection. Although the aforementioned projection-based and gradient-based methods achieve Efficient & Direct Model Updates by directly modifying the original model weights, they inadvertently neglect Non-UE during the execution of GU, leading to sub-optimal unlearning and predictive performance. As a result, despite significant theoretical advancements and certifiable, their practical performance is concerning, as highlighted in a recent study [32]. In contrast, while partition-based and learning-based methods still have room for improvement in terms of forgetting, these methods place significant emphasis on the predictive performance of Non-UE, thereby achieving Inference Protection. In a nutshell, considering the deployment of GU in complex real-world applications, their practical unlearning and predictive performance are pivotal in designing efficient GU algorithms.

Billion-level Scalability. Considering the strict optimization constraints of projection-based methods, the fine-grained graph partitioning and learnable partition output aggregation module of partition-based methods, and the gradient-based methods' reliance on bounds on the gradient residual norm and the computation of the inverse Hessian matrix, these approaches inevitably lack scalability when dealing with billion-level industrial graphs. In contrast, learning-based methods, through carefully designed fine-tuning mechanisms, can be easily extended to large-scale graphs. Meanwhile, it ensures the performance of Model Update and Inference Protection while enjoying high running efficiency.

Based on this, we suggest that GU method should be capable of being applicable to any GNN backbone model (Model Agnostic) and handling any unlearning request at any time (Request Agnostic). Hence, it should not only have the ability to adjust the trained model and continue training efficiently and directly (Efficient & Direct Model Update) but also generate predictions that prioritize the performance of Non-UE (Inference Protection). Furthermore, considering real-world deployment requirements, such methods should demonstrate high efficiency in both training and inference, particularly in industry scenarios (Billion-level Scalability). We hope that the above design principles will drive the future development of GU and inspire more practical approaches.

Table 6: A systematic summary of recent GU studies.

Methods	Type	Model Agnostic	Request Agnostic	Efficient & Direct Model Update	Inference Protection	Billion-level Scalability
Projector [13] (AISTATS'23)	Projection	✗	✓	✓	✗	✗
GraphEditor [12] (arXiv'23)	Projection	✗	✓	✓	✗	✗
GraphEraser [7] (CCS'22)	Partition	✓	✓	✗	✓	✗
GUIDE [47] (USENIX'23)	Partition	✓	✓	✗	✓	✗
GraphRevoker [64] (WWW'24)	Partition	✓	✓	✗	✓	✗
CGU [10] (ICLR'23)	Gradient	✗	✓	✓	✗	✗
CEU [55] (KDD'23)	Gradient	✗	✗	✓	✗	✗
GST [35] (WWW'23)	Gradient	✗	✗	✓	✗	✗
GIF [54] (WWW'23)	Gradient	✓	✓	✓	✗	✗
ScaleGUN [60] (ICLR'25)	Gradient	✓	✓	✓	✗	✓
D2DGN [42] (arXiv'24)	Learning	✓	✓	✓	✓	✗
GCU [10] (ICBD'23)	Learning	✓	✗	✗	✓	✗
GNNDelete [8] (ICLR'23)	Learning	✓	✓	✗	✓	✗
UtU [44] (WWW'24)	Learning	✓	✗	✗	✓	✗
MEGU [30] (AAAI'24)	Learning	✓	✓	✓	✓	✗
SGU (This Paper)	Learning	✓	✓	✓	✓	✓

A.2 Our Approach and ScaleGUN

Scalable GU has recently gained significant attention, garnering significant attention due to its fundamental importance in addressing real-world challenges. ScaleGUN [60] represents an important advancement, offering a certifiable mechanism to efficiently handle billion-scale graphs through lazy local propagation, ensuring certified removal of nodes, edges, and features. However, our approach—NIM combined with SGU—introduces critical improvements, particularly in inference protection. While ScaleGUN focuses on updating the embedding matrix and emphasizes scalability and certification, SGU addresses both scalability and the challenge of gradient-driven node entanglement. This entanglement complicates complete knowledge removal due to persistent interactions among graph elements. NIM decouples influence propagation, identifying and minimizing the impact of unlearned nodes more effectively. The key distinction between SGU and ScaleGUN is in inference protection. While ScaleGUN ensures certified unlearning, it does not safeguard post-unlearning model integrity. SGU, through fine-tuning, preserves accurate and reliable predictions, crucial in applications where model integrity is as important as unlearning itself. Additionally, NIM is a plug-and-play strategy with more flexibility, seamlessly integrating with existing GU methods for enhanced performance. In summary, SGU offers a more comprehensive solution, addressing inference protection, and ensuring unlearning requests do not degrade model accuracy or reasoning, making it a robust choice for real-world applications.

A.3 Our Approach and Traditional Social IM

The core of our approach in this paper is the NIM module, inspired by the traditional social IM problem. It aims to identify Non-UE significantly influenced by UE during model training as reliable HIE. This guides SGU in constructing entity-specific optimization objectives to achieve complete knowledge removal while preserving predictions. Notably, despite the numerous connections between NIM and social IM, significant differences also exist. Therefore, in this section, we review the traditional social IM problem and the key design of NIM to avoid confusion and further elucidate the intuition behind our approach and potential future improvements.

As described in Sec. 2.3, the traditional social IM problem views the seed node set S as the source of influence, where other nodes that are significantly influenced become θ-based threshold activated node set $\sigma(S)$. Specifically, traditional social IM aims to find the seed set S that maximizes $\sigma(S)$ in the current network topology under the following constraints: (1) Certain influence propagation model, such as the Linear Threshold and Independent Cascade models; (2) Quantification function based on threshold θ to obtain a unified criterion; (3) Budget $\mathcal{B}$ decides the size of the seed node set $|S| = \mathcal{B}$. Although the problem of finding S to maximize $\sigma(S)$ under these constraints is NP-hard, some studies [33] have indicated that if $\sigma(S)$ is non-decreasing and sub-modular with respect to S, a greedy algorithm can provide an approximation guarantee of $(1 - 1/e)$. Consequently, current studies in traditional social IM seek to design more reasonable quantification functions and reduce the theoretical complexity of greedy search processes through numerical linear algebra and algorithmic approaches. In a nutshell, in traditional social IM, obtaining the optimal seed set S is the ultimate goal of the optimization problem, while maximizing the activated node set $\sigma(S)$ is one of the constraints. Current research efforts in the field are predominantly focused on investigating the intricate relationships between quantification criteria and optimization constraints, aiming to achieve tighter error bounds and enhanced computational efficiency. These studies employ advanced numerical computation methods to systematically analyze and optimize the trade-offs between various performance metrics.

Building upon these foundations, we describe social IM in the context of GU. Although concepts such as the seed node set S, activated node set $\sigma(S)$, and influence propagation functions exist in NIM, they have different meanings compared to traditional social IM. Specifically, NIM uses UE to represent the seed node set $S = \Delta\mathcal{V}$, HIE to represent the activated node set $\sigma(S)$, and GNN propagation formulas to represent the influence propagation model (technical details in Sec. 3.1.1). Based on this, the significant differences between NIM and traditional social IM are as follows: (1) NIM has a fixed seed node set, thus, the budget $\mathcal{B}$ in GU determines $\sigma(S)$, independent of S. (2) Unlike the optimization problem

Algorithm 1 Scalable Graph Unlearning

Model training related to downstream tasks;
Receive graph element deletion request $\Delta\mathcal{G}$;
Initialize seed set (HIE) $\mathcal{S} = \emptyset$;
Execute unlearning entity transformation based on Eq. (9);
if f_{prop} not satisfies decoupled-based GNN paradigm **then**
  for $i = 1, 2, \ldots, k$ **do**
    Execute graph propagation to obtain $\tilde{\mathbf{X}}^k$ based on Eq. (4);
  end for
end if
Execute inference to obtain soft label $\tilde{\mathbf{Y}}^k$ based on the $\tilde{\mathbf{X}}^k$;
for $t = 1, 2, \ldots, \mathcal{B}$ **do**
  for $v \in \mathcal{V}/\Delta\mathcal{V}$ **do**
    Execute the influence quantification function based on the $\tilde{\mathbf{X}}^k$, $\tilde{\mathbf{Y}}^k$, and Eq. (5-7);
    Update $\sigma(\mathcal{S} \cup \{v\})$ for each Non-UE based on the $\tilde{I}(v, u, k) = \tilde{I}_t(v, u, k) + \tilde{I}_f(v, u, k)$;
  end for
  $v^\star = \arg\max_{v \in \mathcal{V}/\Delta\mathcal{V}} \tilde{I}(v, u, k)$;
  $\mathcal{S} = \mathcal{S} \cup \{v^\star\}$;
  Obtain UE, HIE, and Non-UE based on the $\mathcal{V}$, $\Delta\mathcal{V}$, and $\mathcal{S}$;
  Execute entity-based fine-tuning based on Eq. (9-12) to obtain the modified backbone model $f^\star_{gnn}$;
  Execute inference to obtain Non-UE prediction $\tilde{\mathbf{Y}}$;
  return $f^\star_{gnn}$ and $\tilde{\mathbf{Y}}$;
end for

addressed in traditional social IM, NIM directly identifies nodes meeting the criteria as $\sigma(S)$ through straightforward calculations based on the unlearning budget $\mathcal{B}$ and threshold θ using the seed set S, the influence propagation model, and the influence quantitative function. Notably, since NIM does not involve complex algorithmic problems and challenging optimization constraints, it focuses more on designing suitable influence propagation models and constructing fine-grained influence quantification criteria, ensuring a high-quality and reliable activated node set $\sigma(S)$ as HIE.

In our implementation, to seamlessly integrate NIM with any reasonable GNN backbone and reduce application complexity, we default to using the propagation formulas in the GNN backbone as the influence propagation model and directly measure the differences in smooth features and soft labels before and after propagation and learnable neural architecture (e.g., MLP) as the influence quantification criterion. While experimental results demonstrate that this strategy is simple and effective, a promising direction involves designing a unified influence propagation model specifically suited for GU by considering the common characteristics of graph propagation equations. Additionally, exploring more diverse quantification criteria, such as incorporating neighborhood information when calculating influence, is worth further investigation. In a nutshell, although NIM is inspired by traditional social IM, there are significant differences in their definitions and research focuses. However, essentially, they share many commonalities. Extracting key insights from traditional social IM to enhance NIM is valuable. Such cross-domain knowledge transfer could potentially lead to breakthroughs in areas such as influence propagation and seed selection strategies while maintaining the unique GU characteristics.

A.4 Algorithm and Complexity Analysis

For a more comprehensive overview, we present the complete SGU algorithm in Algorithm 1. To illustrate its complexity, we use SGC [53] as the backbone for k-step graph propagation. Our analysis can be easily extended to any other backbone. For a k-layer SGC with batch size b, the propagated feature $\mathbf{X}^{(k)}$ has a time and space complexity of $O(kmf)$ and $O((b+k)f)$. For a more detailed analysis of propagation mechanisms shown in Sec. 2.2, we recommend referring to related studies [28] that provide comprehensive insights on this topic. At this stage, we have captured influence propagation features from a topological perspective and generated soft label predictions from a feature perspective using forward inference, with negligible computational overhead. Next, we focus on selecting activated nodes as HIEs based on influence measurement, involving matrix computations for distance metrics. We optimize this process using locality-sensitive hashing, an approximate nearest neighbors algorithm, alongside parallelized CPU and GPU computations implemented with NumPy. The time and space complexity of this step are bounded by $O(n \log f/p)$ and $O(nf)$, respectively. Finally, we perform fine-tuning by accurately identifying UE, HIE, and Non-UE entities. Due to entity-specific optimizations implemented for this step, the training overhead is minimal compared to the earlier NIM process and can be disregarded.

A.5 Dataset Description

Cora, **CiteSeer**, and **PubMed** [59] are three citation network datasets, where nodes and edges represent papers and citation relationships. The node features are word vectors, where each element indicates the presence or absence of each word in the paper.
Amazon Photo and **Amazon Computers** [40] are segments of the Amazon co-purchase graph, where nodes represent items and edges represent that two goods are frequently bought together. Given product reviews as bag-of-words node features.
ogbn-arxiv and **ogbn-papers100M** [22] are two citation graphs indexed by MAG [49]. Each paper involves averaging the embeddings of words in its title and abstract. The embeddings of individual words are computed by running the skip-gram model.
ogbn-products [22] is a co-purchasing network, where nodes represent products and edges represent that two products are frequently bought together. Node features are generated by extracting bag-of-words features from the product descriptions.
PPI [63] stands for protein-protein interaction network, where nodes represent protein. If two proteins participate in a life process or perform a certain function together, it is regarded as an interaction between these two proteins.
Flickr [63] originates from SNAP. In this graph, each node represents an image. An edge exists between two nodes if the corresponding images share common properties. Node features are represented by a 500-dimensional bag-of-words model of the images. The labels consist of 81 tags for each image, manually merged into 7 classes, with each image belonging to one of these classes.
Reddit [20] dataset collected from Reddit, where 50 large communities have been sampled to build a post-to-post graph, connecting posts if the same user comments on both. For features, off-the-shelf 300-dimensional GloVe vectors are used.

Table 7: The statistical information of the experimental datasets.

Dataset	#Nodes	#Features	#Edges	#Classes	#Task Type	Description
Cora	2,708	1,433	5,429	7	Transductive	Citation Network
CiteSeer	3,327	3,703	4,732	6	Transductive	Citation Network
PubMed	19,717	500	44,338	3	Transductive	Citation Network
Amazon Photo	7,487	745	119,043	8	Transductive	Co-purchase Graph
Amazon Computer	13,381	767	245,778	10	Transductive	Co-purchase Graph
ogbn-arxiv	169,343	128	2,315,598	40	Transductive	Citation Network
ogbn-products	2,449,029	100	61,859,140	47	Transductive	Co-purchase Graph
ogbn-papers100M	111,059,956	128	1,615,685,872	172	Transductive	Citation Network
PPI	56,944	50	818,716	121	Inductive	Protein Network
Flickr	89,250	500	899,756	7	Inductive	Image Network
Reddit	232,965	602	11,606,919	41	Inductive	Social Network
ogbn-collab	235,868	128	1,285,465	-	Link Prediction	Collaboration network
ogbn-ppa	576,289	58	30,326,273	-	Link Prediction	Protein Network
ogbn-citation2	2,927,963	128	30,561,187	-	Link Prediction	Citation Network

ogbn-collab [49] is an undirected graph, representing a subset of the collaboration network between authors. Each node represents an author and edges indicate the collaboration between authors. All edges are associated with the year (meta-information), representing the number of co-authored papers published in that year.
ogbn-ppa [43] is an undirected, unweighted graph where nodes represent proteins from 58 different species, and edges indicate biologically meaningful associations between proteins. Each node is associated with a 58-dimensional one-hot feature vector that indicates the species of the corresponding protein.
ogbl-citation2 [49] represents the citation network. Each node is a paper with 128-dimensional word2vec features that summarize its title and abstract, and each directed edge indicates that one paper cites another. All nodes also come with meta-information indicating the year the corresponding paper was published.

A.6 Compared Baselines

To evaluate the effectiveness of various GU strategies, we have selected commonly used GNNs as the backbone models to simulate scenarios where unlearning requests are received during training. These compared models represent successful recent designs in scalable graph learning widely applicable in both transductive and inductive settings and link-specific GNNs for link-level downstream tasks. Furthermore, various backbone GNNs can be employed to assess the generalization capability of diverse GU approaches. The salient characteristics of all baseline models are outlined below:
GCN [26] introduces a novel approach to graphs that is based on a first-order approximation of spectral convolutions on graphs. This approach learns hidden layer representations that encode both local graph structure and features of nodes.
GAT [46] utilizes attention mechanisms to quantify the importance of neighbors for message aggregation. This strategy enables implicitly specifying different weights to different nodes in a neighborhood, without depending on the graph structure upfront.
GIN [57] presents a theoretical framework analyzing GNNs' expressive power to capture graph structures. They develop a simple architecture that is probably the most expressive among GNNs and as powerful as the Weisfeiler-Lehman graph isomorphism test.
GraphSAGE [20] leverages neighbor node attribute information to efficiently generate representations. This method introduces a general inductive framework that leverages node feature information to generate node embeddings for previously unseen data.
GraphSAINT [63] is an inductive framework that enhances training efficiency through graph sampling. In each iteration, a complete GCN is built from the properly sampled subgraph, which decouples the sampling from the forward and backward propagation.
Cluster-GCN [9] is designed for training with stochastic gradient descent by leveraging the graph clustering structure. At each step, it samples a block of nodes that associate with a dense subgraph identified by a graph clustering algorithm and restricts the neighborhood search within this subgraph.
SGC [53] simplifies GCN by removing non-linearities and collapsing weight matrices between consecutive layers. Theoretical analysis shows that the simplified model corresponds to a fixed low-pass filter followed by a linear classifier.
SIGN [16] introduces a novel, efficient, and scalable graph deep learning architecture that eliminates the need for graph sampling. This method sidesteps the need for graph sampling by using graph convolutional filters of different size.
GBP [6] introduces a scalable GNN that employs a localized bidirectional propagation process involving both feature vectors and the nodes involved in training and testing.
S^2GC [73] introduces a modified Markov Diffusion Kernel for GCN, which strikes a balance between low- and high-pass filters to capture the global and local contexts of each node.
AGP [48] proposes a unified randomized algorithm capable of computing various proximity queries and facilitating propagation. This method provides a theoretical bounded error guarantee and runs in almost optimal time complexity.

GAMLP [70] is designed to capture the inherent correlations between different scales of graph knowledge to break the limitations of the enormous size and high sparsity level of graphs hinder their applications under industrial scenarios.
SEAL [65] is learning heuristics from the network. By extracting local subgraphs around target links, SEAL aims to map subgraph patterns to link existence, thus learning suitable heuristics. Based on the heuristic theory, SEAL learns heuristics using a GNN.
NeoGNN [62] learns structural features from the adjacency matrix and estimates overlapped neighborhoods for link prediction. This method generalizes neighborhood overlap-based heuristics and manages multi-hop neighborhoods, enhancing link prediction.
BUDDY [4] propose efficient link prediction with hashing, a novel full-graph GNN that uses subgraph sketches to approximate key components of subgraph GNNs without explicit subgraph construction. For scalability, BUDDY uses feature pre-computation to maintain predictive performance without GPU memory constraints.
NCNC [51] introduces MPNN-then-SF, an architecture using structural features to guide MPNN's representation pooling, implemented as a neural common neighbor. To mitigate graph incompleteness, NCNC uses a link prediction model to complete the common neighbor structure, achieving neural common neighbor with completion.
MPLP [14] harnesses quasi-orthogonal vectors to estimate link-level structural features while preserving node-level complexities. This technology breaks the limitation of node-level representation and struggles with joint structural features essential for link prediction, like a common neighbor.
GraphEditor [12] is an efficient approach for unlearning that supports graph deletion for linear GNN. It doesn't require retraining from scratch or access to all training data and ensures exact unlearning, guaranteeing the removal of all corresponding information.
GUIDE [47] improves GraphEraser by the graph partitioning with fairness and balance, efficient subgraph repair, and similarity-based aggregation. Notably, GUIDE can be efficiently implemented on the inductive graph learning tasks for its low graph partition cost, no matter on computation or structure information.
GraphRevoker [64] is a novel GU framework that maintains model utility better than traditional retraining-based methods. Unlike conventional approaches that partition the training graph into subgraphs, leading to information loss, it employs graph property-aware sharding to preserve graph properties and uses graph contrastive sub-model aggregation for effective prediction.
CGU [10] introduces the first approach for approximate GU with provable guarantees. Their method addresses diverse unlearning requests and evaluates feature mixing during propagation. Notably, CGU only focus on SGC and generalized PageRank extensions.
CEU [55] focuses on edge unlearning in GNNs, training a new GNN as if certain edges never existed. CEU updates the pre-trained GNN model parameters in a single step, efficiently removing the influence of specific edges. The authors provide rigorous theoretical guarantees under convex loss function assumptions.
GIF [54] incorporates an additional loss term by graph-based influence function, considering structural dependencies, and provides a closed-form solution by efficient estimation of the inverse Hessian matrix for better understanding the unlearning mechanism.
D2DGN [42] is a novel knowledge distillation approach for learning-based GU. This framework divides graph knowledge for retention and deletion, using response-based soft targets and feature-based node embeddings while minimizing KL divergence.
GNNDelete [8] is a novel model-agnostic layer-wise operator designed to optimize topology influence in the GU requests. This method supervises the unlearning operator by designing a regularization term based on the interactions between graph entities.
UtU [44] targets edge unlearning and claims that current methods effectively remove specific edges but suffer from over-forgetting, leading to performance decline. Based on this, UtU simplifies the process by unlinking the forgotten edges from the graph structure, improving efficiency and preserving performance.
MEGU [30] is a novel paradigm that simultaneously evolves predictive and unlearning capacities. Specifically, it ensures complementary optimization within a unified training framework, balancing performance and generalization to meet forgetting and prediction.
ScaleGUN [60] is a scalable and certifiable GU mechanism by lazy local propagation, ensuring certified unlearning in three scenarios.

A.7 Hyperparameter settings

The hyperparameters in the backbones and GU are set according to the original paper if available. Otherwise, we perform a hyperparameter search via the Optuna [2]. Specifically, we explore the optimal shards within the ranges of 20 to 100. The weight coefficients of the loss function and other hyperparameters are obtained by means of an interval search from $\{0, 1\}$ or the interval suggested in the original paper. For our proposed SGU, the detailed hyperparameter settings are as follows. To begin with, in the NIM process, we explore the optimal number of graph propagation steps within the range of 3 to 10, while adhering to the original GNN backbone propagation steps (k in Eq. (4)). Subsequently, we set the default unlearning budget $\mathcal{B}$ to be three times of $|\Delta\mathcal{V}|$ for selecting HIE, and the optimal threshold θ is searched within the range of 0.5 to 1. After that, we construct λ-flexible entity-specific optimization objectives for efficient fine-tuning. The overall loss function is represented as $\mathcal{L} = \lambda\mathcal{L}_f + (1-\lambda)\mathcal{L}_p$, where λ is searched within the range of 0 to 1. In addition, we choose Adam as the default optimizer. As for the learning rate, weight decay, and dropout, they exhibit significant variations across GNN backbones, GU strategies, and unlearning scenarios. Therefore, we perform separate hyperparameter searches to report the best performance. Notably, for learning-based GU approaches, the training epoch is set to 50, benefiting from the fine-tuning. For both node- and link-level scenarios, we split all datasets following the guidelines of recent GU approaches [8, 30, 44, 55], which randomly allocate 80% of nodes for training and 20% for testing, and 90% of edges for training and 10% for testing. To alleviate the randomness and ensure a fair comparison, we repeat each experiment 10 times for unbiased performance. Notably, we experiment with multiple GNN backbones in separate experimental modules to validate the generalizability of SGU and avoid complex charts, making the results more reader-friendly.

A.8 Experiment Environment

Our experiments are conducted on the machine with Intel(R) Xeon(R) Platinum 8468V, and NVIDIA H800 PCIe and CUDA 12.2. The operating system is Ubuntu 20.04.6. As for the software, we use Python 3.8 and Pytorch 2.2.1 for implementation.

Table 8: Edge-level predictive performance under feature and node unlearning.

GU (↓)	Backbone (↓)	collab		ppa		citation2	
		Feature	Node	Feature	Node	Feature	Node
ScaleGUN	SEAL	63.46±0.32	64.12±0.44	49.86±0.31	50.28±0.28	85.92±0.40	86.71±0.39
	NeoGNN	58.15±0.46	59.33±0.53	49.50±0.44	49.87±0.45	86.38±0.51	86.52±0.48
	BUDDY	66.34±0.65	66.85±0.70	50.34±0.27	50.76±0.36	87.87±0.36	87.23±0.44
	NCNC	66.87±0.72	67.29±0.65	58.79±0.36	59.18±0.47	88.04±0.39	88.40±0.37
	MPLP	67.32±0.57	67.51±0.53	61.55±0.52	61.72±0.41	89.83±0.32	90.16±0.30
UtU	SEAL	64.92±0.75	64.83±0.84	50.23±0.62	50.75±0.57	86.63±0.62	87.06±0.57
	NeoGNN	58.43±0.37	59.10±0.52	50.42±0.58	50.20±0.65	86.94±0.84	86.83±0.73
	BUDDY	66.90±0.68	67.41±0.61	50.56±0.51	51.13±0.70	88.05±0.50	87.71±0.64
	NCNC	67.28±0.57	66.82±0.49	59.82±0.75	60.26±0.63	88.76±0.62	89.24±0.58
	MPLP	67.65±0.83	67.56±0.75	62.17±0.83	62.34±0.72	90.21±0.45	90.45±0.52
SGU	SEAL	66.84±0.63	66.52±0.72	51.85±0.56	52.17±0.62	88.72±0.48	88.93±0.62
	NeoGNN	60.11±0.54	60.94±0.60	52.49±0.60	52.32±0.51	88.31±0.67	88.64±0.58
	BUDDY	67.76±0.89	68.22±0.81	52.13±0.47	52.68±0.39	89.02±0.73	89.50±0.69
	NCNC	68.65±0.52	68.54±0.64	62.50±0.51	62.39±0.50	89.85±0.64	90.17±0.54
	MPLP	68.97±0.73	69.27±0.78	63.58±0.82	63.70±0.77	90.79±0.55	90.86±0.77

Table 9: Edge-level performance within StealLink.

GU (↓)	Backbone (↓)	collab	ppa	citation2
SEAL	GraphRevoker	0.508	OOT	OOM
	CEU	0.534	OOT	OOM
	UtU	0.532	0.544	0.549
	SGU	0.519	0.532	0.528
NCNC	GraphRevoker	0.513	OOT	OOM
	CEU	0.535	OOT	OOM
	UtU	0.538	0.553	0.547
	SGU	0.515	0.525	0.536
MPLP	GraphRevoker	0.517	OOT	OOM
	CEU	0.531	OOT	OOM
	UtU	0.526	0.539	0.551
	SGU	0.520	0.531	0.534

A.9 Link-specific Evaluation

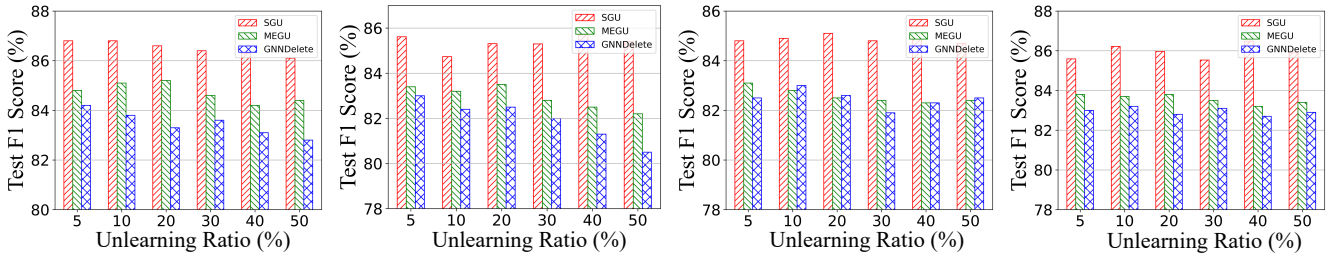
For link-specific scenarios, the evaluation methodologies are as follows: (1) **Model Update**: Inspired by prevalent methods [55], we employ StealLink [21], an edge MIA, to empirically evaluate the extent to which the model has forgotten the removed edges. A higher AUC (>0.5) indicates lower unlearning performance, whereas an AUC close to 0.5 implies complete forgetting. (2) **Inference Protection**: We directly evaluate the reasoning capability of the modified model by reporting the Non-UE predictions (i.e., link prediction) via the metric suggested by the corresponding benchmark datasets. Specifically, according to the suggestion of OGB official documents, we utilize HR@50, HR@100, and MRR to evaluate the predictive performance of ogb-collab, ogb-ppa, and ogb-citation2, respectively.

To extend SGU to link-specific scenarios, we summarize the following key points: (1) We use the transformation function proposed in Eq. (9) to obtain node-level UEs, which are then processed by NIM to obtain HIE for subsequent entity-based optimization objectives; (2) In prediction-level optimization, we need to replace the node-level cross-entropy loss with a link-specific loss function; (3) In embedding-level optimization, since link-specific GNNs still generate node embeddings for edge representation, we can directly apply the loss formulation of SGU as presented in Sec.3.2.1.

In addition to Table 4 presented in Sec. 4.2, we include Table 8 to provide a comprehensive evaluation. Our experimental results indicate that SGU achieves the best performance across all datasets and unlearning requests, demonstrating its robust generalization. Notably, CEU lacks scalability because it needs to consider all graph elements contributing to the gradients for optimization. Meanwhile, to evaluate the unlearning performance of link-specific GU methods, we report their unlearning performance in Table 9. We observe that, although partition-based GraphRevoker can achieve sufficient forgetting by retraining specific partitions, the high partition cost results in OOT and OOM errors. Compared to other baselines, SGU excels in forgetting capability, thanks to the reliable HIE identified by NIM and the well-designed optimization objectives.

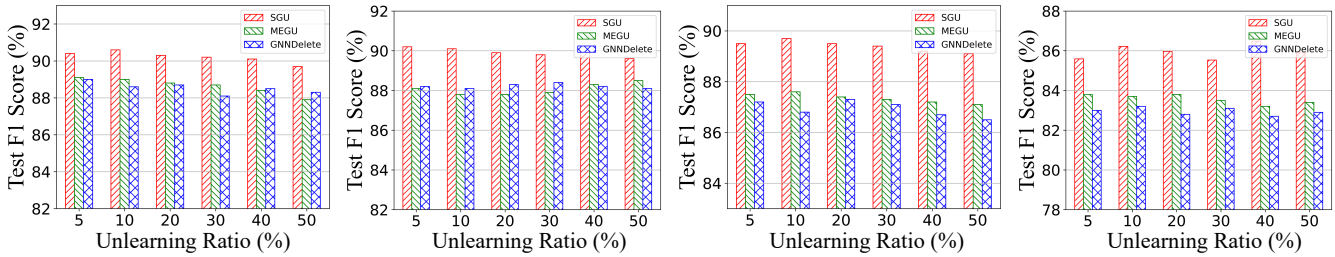
A.10 Unlearning Challenges at Different Scales

To comprehensively evaluate the efficacy of SGU across varying unlearning scales, in addition to the results provided in Fig. 4 in Sec. 4.4, we present additional experimental results in Fig. 7 and Fig. 8. Our experiments reveal that edge unlearning is less impacted than feature and node unlearning on predictive performance. In edge unlearning scenarios, we remove the unlearning edges and treat the connected nodes as UE, resulting in more significant unlearning costs. Such discrepancy arises more from the nuanced process associated with node unlearning, where the edge removal disrupts the original topology and causes performance deterioration. In summary, edge unlearning induces a comparatively milder impact on predictive performance for Non-UE compared to other unlearning scenarios. Inspired by these findings, we introduce three additional large-scale link-level benchmark datasets and link-specific GNNs as baselines to provide a comprehensive evaluation.



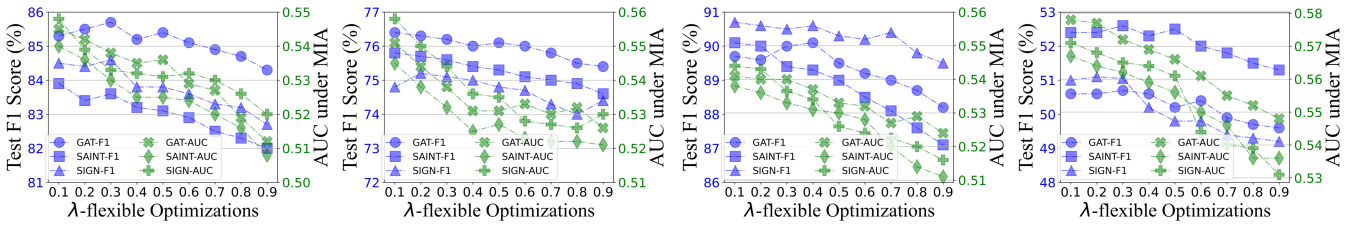
(a) GAT Backbone (b) GIN Backbone (c) GraphSAINT Backbone (d) ClusterGCN Backbone

Figure 7: Predictive performance of feature unlearning within different ratios on Cora.



(a) GAT Backbone (b) GIN Backbone (c) GraphSAINT Backbone (d) ClusterGCN Backbone

Figure 8: Predictive performance of edge unlearning within different ratios on Photo.



(a) Cora (b) CiteSeer (c) Photo (d) Flickr

Figure 9: Sensitivity analysis on entity-specify optimization objectives.

Table 10: Running efficiency experiments.

	ogbn-products		Reddit	
	Pre.(s)	Train.(s)	Pre.(s)	Train.(s)
SGU (S^2GC)	143.7±1.8	38.54±0.8	36.46±0.7	8.72±0.7
ScaleGUN	-	126.81±2.3	-	32.18±1.2

A.11 λ-flexible Entity-specific Optimization

In our proposed NIM and SGU, besides the critical $\mathcal{B}$ and θ in selecting reliable HIE, the hyperparameter λ used to balance forgetting and reasoning loss is also pivotal, as it directly affects unlearning and predictive performance via the fine-tuning. For more details on how λ links our proposed optimization objectives, please refer to Appendix A.7. Therefore, in addition to Fig. 4 provided in Sec. 4.4, we present an additional sensitivity analysis of SGU to λ in Fig. 9.

Notably, in our implementation of linking forgetting and reasoning loss, a larger and smaller λ emphasizes SGU's unlearning and predictive performance. Based on this, according to the experimental results, we find that in most cases, SGU achieves the best unlearning performance at $\lambda = 0.9$, indicated by an AUC closer to 0.5. However, $\lambda = 0.9$ means that SGU neglects reasoning ability, leading to sub-optimal predictions. Therefore, it is important to balance the capabilities of forgetting and reasoning according by adjusting λ to actual needs.

A.12 Efficiency Experiments with ScaleGUN

Our proposed SGU demonstrates competitive efficiency compared to ScaleGUN, as evidenced by Table 10. It reveals that while SGU exhibits slightly lower efficiency than ScaleGUN when employing S^2GC, considering both the one-step pre-processing and training, it offers superior deployment flexibility due to its independence from specific graph propagation. Notably, the one-step preprocessing demonstrates excellent scalability in practical applications, and in most scenarios, we primarily need to account for the training time overhead, where SGU shows significant advantages. Furthermore, SGU exhibits substantial potential for reducing pre-processing time through optimization of its graph propagation equation. It is worth emphasizing that although SGU and ScaleGUN achieve comparable complexity, SGU demonstrates broader application potential, as supported by its superior performance in practical deployment scenarios, as illustrated in Table 2 and Table 3.

REFERENCES

[1] 2025. SGU Technical Report. In *https://github.com/xkLi-Allen/SGU.*

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.*

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *Computer Science* (2013).

[4] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. 2023. Graph neural networks for link prediction with subgraph sketching. *International Conference on Learning Representations, ICLR* (2023).

[5] Deli Chen, Yankai Lin, Guangxiang Zhao, Xuancheng Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Topology-imbalance learning for semi-supervised node classification. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[6] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. *Advances in Neural Information Processing Systems, NeurIPS* (2020).

[7] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security, CCS.*

[8] Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. 2023. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *International Conference on Learning Representations, ICLR.*

[9] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.*

[10] Eli Chien, Chao Pan, and Olgica Milenkovic. 2022. Certified Graph Unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning.*

[11] Seungyoon Choi, Wonjoong Kim, Sungwon Kim, Yeonjun In, Sein Kim, and Chanyoung Park. 2024. DSLR: Diversity Enhancement and Structure Learning for Rehearsal-based Graph Continual Learning. In *Companion Proceedings of the ACM on Web Conference, WWW.*

[12] Weilin Cong and Mehrdad Mahdavi. 2022. GraphEditor: An Efficient Graph Representation Learning and Unlearning Approach. (2022).

[13] Weilin Cong and Mehrdad Mahdavi. 2023. Efficiently Forgetting What You Have Learned in Graph Representation Learning via Projection. In *International Conference on Artificial Intelligence and Statistics, AISTATS.*

[14] Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. 2023. Pure Message Passing Can Estimate Common Neighbor for Link Prediction. *arXiv preprint arXiv:2309.00976* (2023).

[15] Wenzheng Feng, Yuxiao Dong, Tinglin Huang, Ziqi Yin, Xu Cheng, Evgeny Kharlamov, and Jie Tang. 2022. Grand+: Scalable graph random neural networks. In *Proceedings of the ACM Web Conference, WWW.*

[16] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198* (2020).

[17] Johannes Gasteiger, Chendi Qian, and Stephan Günnemann. 2022. Influence-based mini-batching for graph neural networks. In *Learning on Graphs Conference, LoG.* PMLR.

[18] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2020. Certified data removal from machine learning models. *International Conference on Machine Learning, ICML* (2020).

[19] Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. 2020. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proceedings of the ACM on Management of Data, SIGMOD.*

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems, NeurIPS* (2017).

[21] Xinlei He, Jinyuan Jia, Michael Backes, and et al Gong. 2021. Stealing links from graph neural networks. In *USENIX Security Symposium, USENIX.*

[22] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems, NeurIPS* (2020).

[23] Youpeng Hu, Xunkai Li, Yujie Wang, Yixuan Wu, Yining Zhao, Chenggang Yan, Jian Yin, and Yue Gao. 2021. Adaptive hypergraph auto-encoder for relational data clustering. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[24] Peter Kabal and Ravi Prakash Ramachandran. 1986. The computation of line spectral frequencies using Chebyshev polynomials. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34, 6 (1986), 1419–1426.

[25] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.*

[26] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR.*

[27] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning, ICML.*

[28] Xunkai Li, Meihao Liao, Zhengyu Wu, Daohan Su, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. LightDiC: A Simple Yet Effective Approach for Large-Scale Digraph Representation Learning. *Proceedings of the VLDB Endowment* (2024).

[29] Xunkai Li, Jingyuan Ma, Zhengyu Wu, Daohan Su, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. Rethinking Node-wise Propagation for Large-scale Graph Learning. In *Proceedings of the ACM Web Conference, WWW.*

[30] Xunkai Li, Yulin Zhao, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. Towards Effective and General Graph Unlearning via Mutual Evolution. In *Proceedings of the Association for the Advancement of Artificial Intelligence, AAAI.*

[31] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Jiongran Wu, Peng Zhang, Li Shang, and Ning Gu. 2023. Recommendation unlearning via matrix correction. *arXiv preprint arXiv:2307.15960* (2023).

[32] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations, ICLR.*

[33] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14 (1978), 265–294.

[34] Naoto Ohsaka. 2020. The solution distribution of influence maximization: A high-level experimental study on three algorithmic approaches. In *Proceedings of the ACM on Management of Data, SIGMOD.*

[35] Chao Pan, Eli Chien, and Olgica Milenkovic. 2023. Unlearning Graph Classifiers with Limited Data Resources. In *Proceedings of the ACM Web Conference, WWW.*

[36] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. 2023. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *Advances in Neural Information Processing Systems, NeurIPS* (2023).

[37] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: are we really making progress? *International Conference on Learning Representations, ICLR* (2023).

[38] Anwar Said, Tyler Derr, Mudassir Shabbir, Waseem Abbas, and Xenofon Koutsoukos. 2023. A survey of graph unlearning. *arXiv preprint arXiv:2310.02164* (2023).

[39] Thanveer Basha Shaik. 2023. Revolutionizing healthcare with federated reinforcement learning: from machine learning to machine unlearning. (2023).

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

[41] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy, SP.*

[42] Yash Sinha, Murari Mandal, and Mohan Kankanhalli. 2023. Distill to Delete: Unlearning in Graph Networks with Knowledge Distillation. *arXiv preprint arXiv:2309.16173* (2023).

[43] Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. 2019. STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research* 47, D1 (2019), D607–D613.

[44] Jiajun Tan, Fei Sun, Ruichen Qiu, Du Su, and Huawei Shen. 2024. Unlink to unlearn: Simplifying edge unlearning in gnns. In *Companion Proceedings of the ACM on Web Conference, WWW.*

[45] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the ACM on Management of Data, SIGMOD.*

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations, ICLR.*

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

[48] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate graph propagation. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD.*

[49] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[50] Song Wang, Yushun Dong, Binchi Zhang, Zihan Chen, Xingbo Fu, Yinhan He, Cong Shen, Chuxu Zhang, Nitesh V Chawla, and Jundong Li. 2024. Safety in Graph Machine Learning: Threats and Safeguards. *arXiv preprint arXiv:2405.11034* (2024).

[51] Xiyuan Wang, Haotong Yang, and Muhan Zhang. 2024. Neural common neighbor with completion for link prediction. *International Conference on Learning Representations, ICLR* (2024).

[52] Yexin Wang, Zhi Yang, Junqi Liu, Wentao Zhang, and Bin Cui. 2023. Scapin: Scalable Graph Structure Perturbation by Augmented Influence Maximization. *Proceedings of the ACM on Management of Data, SIGMOD* (2023).

[53] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International Conference on Machine Learning, ICML*.

[54] Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. 2023. GIF: A General Graph Unlearning Strategy via Influence Function. In *Proceedings of the ACM Web Conference, WWW*.

[55] Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. 2023. Certified edge unlearning for graph neural networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD*.

[56] Xin Xin, Liu Yang, Ziqi Zhao, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. On the effectiveness of unlearning in session-based recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM*.

[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? (2019).

[58] Yu Yang, Enhong Chen, Qi Liu, Biao Xiang, Tong Xu, and Shafqat Ali Shad. 2012. On approximation of real-world influence spread. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*. Springer, 548–564.

[59] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *International Conference on Machine Learning, ICML*.

[60] Lu Yi and Zhewei Wei. 2024. Scalable and Certifiable Graph Unlearning via Lazy Local Propagation. *arXiv preprint arXiv:2408.09212* (2024).

[61] Wei Yuan, Hongzhi Yin, Fangzhao Wu, Shijie Zhang, Tieke He, and Hao Wang. 2023. Federated unlearning for on-device recommendation. In *Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM*.

[62] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. 2021. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[63] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *International conference on learning representations, ICLR*.

[64] Jiahao Zhang. 2024. Graph unlearning with efficient partial retraining. In *Companion Proceedings of the ACM on Web Conference, WWW*.

[65] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems, NeurIPS* (2018).

[66] Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. PaSca: A Graph Neural Architecture Search System under the Scalable Paradigm. In *Proceedings of the ACM Web Conference, WWW*.

[67] Wentao Zhang, Yexin Wang, Zhenbang You, Meng Cao, Ping Huang, Jiulong Shan, Zhi Yang, and Bin Cui. 2021. Rim: Reliable influence-based active learning on graphs. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[68] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Node dependent local smoothing for scalable graph learning. *Advances in Neural Information Processing Systems, NeurIPS* (2021).

[69] Wentao Zhang, Zhi Yang, Yexin Wang, Yu Shen, Yang Li, Liang Wang, and Bin Cui. 2021. Grain: Improving data efficiency of graph neural networks via diversified influence maximization. *Proceedings of the VLDB Endowment* (2021).

[70] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. Graph Attention Multi-Layer Perceptron. *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD* (2022).

[71] Yuying Zhao, Yunfei Hu, Pingpeng Yuan, and Hai Jin. 2021. Maximizing influence over streaming graphs with query sequence. *Data Science and Engineering* 6, 3 (2021), 339–357.

[72] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[73] Hao Zhu and Piotr Koniusz. 2021. Simple spectral graph convolution. In *International Conference on Learning Representations, ICLR*.