Subgraph Federated Unlearning

Anonymous Author(s)

ABSTRACT

Subgraph federated learning aims to collaboratively train a global model over distributed subgraphs stored in multiple local clients with strict privacy constraints, which is crucial to a wide range of applications such as healthcare, recommendation systems, and financial crime detection. With the increasing emphasis on the "right to be forgotten," the issue of machine unlearning of subgraph federated models has gained significant importance. However, existing federated unlearning approaches largely focused on unstructured data, overlooking the impact of structural dependency and cross-client interferences in graph-based data. To this end, in this paper, we propose ReGEnUnlearn, a subgraph federated unlearning framework for efficient and comprehensive unlearning of multiple target clients. Specifically, we first propose the *Reinforced Federated Policy Sampler* (RFPS) to learn optimal sampling strategies that minimize the interference among cross-client subgraphs. By interacting with the federated graph sampling environment, the agent learns to selectively forget an optimal subgraph of target clients, thus preserving the global model utility. Moreover, we propose the *Parameter-free Graph Prompt Knowledge Distillation* (PGPKD) module, which retains the unique graph knowledge contributed by the target clients, thereby facilitating comprehensive unlearning via a tailored gradient ascent objective. Extensive experiments in various federated settings demonstrate ReGEnUnlearn's superiority over existing federated unlearning methods, offering a speedup of 3.6× to 9× compared to traditional retraining while maintaining model utility within a range of 100% – 102%. The source code is available at https://anonymous.4open.science/r/Unlearn-F27B/README.md

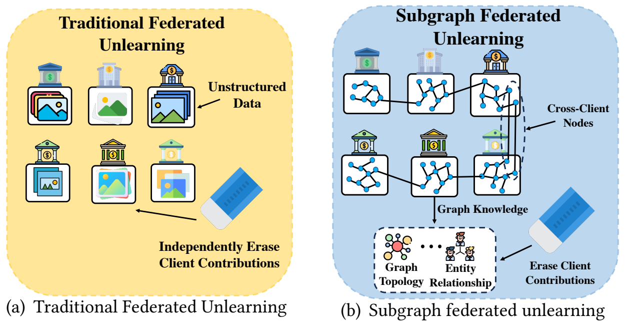
CCS CONCEPTS

• **Security and privacy** → *Mobile and wireless security*; • **Computing methodologies** → **Supervised learning**.

KEYWORDS

machine unlearning; subgrpah federated learning

ACM Reference Format:
Anonymous Author(s). 2018. Subgraph Federated Unlearning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX


Traditional Federated Unlearning
Unstructured Data
Independently Erase Client Contributions
Subgraph Federated Unlearning
Cross-Client Nodes
Graph Knowledge
Graph Topology
Entity Relationship
Erase Client Contributions

(a) Traditional Federated Unlearning (b) Subgraph federated unlearning

Figure 1: (a) Traditional federated unlearning on unstructured data (*e.g.*, image, time series, and text *etc.*). (b) Subgraph federated unlearning raises additional concerns about erasing structural knowledge and cross-client subgraph overlap.

1 INTRODUCTION

Subgraph federated learning aims to address the challenges of collaborative learning across distributed subgraphs stored in multiple local systems while adhering to strict privacy regulations [3]. This approach is critical to diverse applications, including healthcare, recommendation systems, and financial crime detection, especially in cross-silo federated learning scenarios [36, 40, 43]. With increasing emphasis on privacy rights, such as the "right to be forgotten [11] ," subgraph federated unlearning emerges as a crucial task, aiming to selectively remove specific clients' contributions from the global model. For example, in the context of international banks applying federated learning for financial crime detection, changing privacy laws may compel certain banks to withdraw their data, underscoring the need for effective unlearning mechanisms.

Nevertheless, existing federated unlearning approaches, as discussed in [12, 13, 23, 30, 35], predominantly focus on non-graph data, thus failing to address the complexities in subgraph federated unlearning. For example, VERIFI [10] involves the submission of gradient updates by all clients to the server. The server then amplifies gradients from remaining clients while reducing contributions from the target clients. Halimi *et al.* [14] introduce a projected gradient ascent to maximize empirical loss on the target clients. As another example, FedRecover [46] iteratively eliminates contributions from target clients based on historical storage models. However, the above methods primarily focused on erasing the contribution of a target client with unstructured data (*e.g.*, image, time series, and text *etc.*), largely overlook the impact of structural dependency and cross-client interferences in graph-based data, as depicted in Figure 1. To address this gap, we investigate the subgraph unlearning problem in a federated environment with multiple client participants.

However, it is a non-trivial issue to effectively opt out of the contribution of multiple distributed clients while maintaining the model utility. (1) *Cross-client subgraph interference.* In the federated scenario, the subgraphs held by each client may overlap with each other (*e.g.*, common users in multiple banks, patients shared by

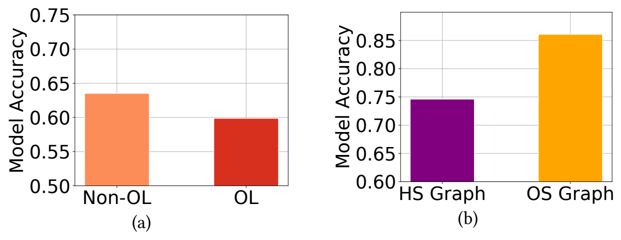


Figure 2: (a) Comparison of unlearning on subgraphs with non-overlapping (Non-OL) and overlapping (OL) on the Photo dataset. The subgraph overlapping significantly reduces the model's utility. (b) Unlearning on optimally sampled subgraph (OS Subgrpah) better retains the model utility compared to heuristically sampled subgraph (HS Subgraph).

different hospitals). Simply eliminating contributions of a subgraph in a target client may forget common knowledge encoded in the overlapped subgraph, therefore leading to unexpected performance degradation of the global model, as illustrated in Figure 2(a). One straightforward solution is removing cross-client subgraphs via graph sampling and applying unlearning on the filtered subgraph, as depicted in Figure 2(b). However, due to privacy constraints, both the server and clients have limited access to the identity of cross-client subgraphs. How to eliminate the cross-client interference to guarantee the global model utility is the first challenge. (2) *Diversified structural contribution of each client.* A key benefit of federated learning is to collaboratively train a better global model by absorbing personalized knowledge from each client, *e.g.*, rare disease cases in a hospital, exclusive user purchase records in an e-commerce platform. However, removing the unique contribution of a target client from a global graph model requires identifying various knowledge, such as entity attributes, node connectivity, and high-order structural patterns. How to comprehensively unlearn subgraph knowledge of a target client is another challenge.

To address the aforementioned challenges, we introduce the *Reinforced Graph Knowledge Enhancement Subgraph Federated Unlearning (ReGEnUnlearn)* framework, comprising two key modules. First, the *Reinforced Federated Policy Sampler* (RFPS) module is proposed to mitigate the interference among cross-client subgraphs. By interacting with the federated graph sampling environment, the agent learns optimal graph sampling strategies that facilitate the removal of cross-client nodes while preserving model utility. Second, a *Parameter-free Graph Prompt Knowledge Distillation* (PGPKD) module is devised to distill specific graph knowledge from the target clients. This process encodes target clients' unique knowledge into the distilled graph prompts, which can be inserted into the sampled subgraph. In this way, the structural knowledge of a target client can be comprehensively unlearned by optimizing a gradient ascent objective over the prompt-enhanced subgraph.

The contributions of our work are summarized as follows: (1) To our knowledge, this is the first study to investigate subgraph federated unlearning with consideration of multiple target clients. (2) We propose the ReGEnUnlearn framework to comprehensively eliminate multiple target clients while preserving utility. ReGEnUnlearn is agnostic to federated algorithms and can be easily integrated into different subgraph federated scenarios. (3) We conduct comprehensive experiments on four real-world datasets under various subgraph federated settings. The proposed framework demonstrates notable speedups ranging from 3.6× to 9× compared to traditional retraining while maintaining model utility within a range of 100%-102%.

2 PRELIMINARIES

In this section, we provide the background on subgraph federated learning and federated unlearning.

2.1 Subgraph Federated Learning

2.1.1 Notation. We represent the global graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ denotes the node set, $\mathcal{E}$ represents the edge set, and $\mathcal{X}$ denotes the node feature set. Each node $v \in \mathcal{V}$ is associated with its feature $\mathbf{x}_v \in \mathcal{X}$. In the federated learning system, we consider a central server denoted as S and M distributed local clients $C = \{C_1, \cdots, C_M\}$, each possessing a subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{X}_i)$. These entities collaboratively train a global model $F(\mathbf{w})$, where $\mathbf{w}$ is the model weights and $\mathcal{V} = \cup_{i=1}^{M} \mathcal{V}_i$.

2.1.2 Local Graph Learning. The *Message Passing Neural Network (MPNN)* is a widely used framework for graph data that can accommodate various Graph Neural Network (GNN) architectures. In the subgraph federated learning, each client has a local GNN model and collaborates with others to train a global model. The local graph learning process consists of the following two phases.

Message Passing. For each client C_i, the l-th layer in MPNN is defined as follows,

$$\mathbf{h}_j^{l+1,i} = \mathbf{U}_{w^{l,i}}(\mathbf{h}_j^{l,i}, \mathbf{m}_j^{l+1,i}), \tag{1}$$

where $\mathbf{h}_j^{l+1,i}$ represents the node feature vector at layer $l+1$ for node v_j in client C_i. The term $\mathbf{m}_j^{l+1,i}$ denotes the aggregated message from the neighbors of node v_j, and $\mathbf{U}_{w^{l+1,i}}$ is a function updating the node feature vector, with $w^{l+1,i}$ as the corresponding parameter in the l-th layer. The computation of the message $\mathbf{m}_j^{l+1,i}$ is define as below,

$$\mathbf{m}_j^{l+1,i} = \sum_{v_k \in \mathcal{N}(v_j)} \mathbf{M}_{\theta^{l+1,i}}(\mathbf{h}_j^{l,i}, \mathbf{h}_k^{l,i}, \mathbf{e}_{jk}^{l,i}), \tag{2}$$

where $\mathcal{N}(v_j)$ represents the neighborhood of node v_j, $\mathbf{M}_{\theta^{l+1,i}}$ is the function generating the message from node features, and $\theta^{l+1,i}$ is the corresponding parameter. $\mathbf{e}_{jk}^{l,i}$ denotes message embedding associated with the edge between nodes v_j and v_k in the l-th layer.

Readout. The readout phase computes the final feature for subsequent tasks,

$$\hat{y}_{v_j}^i = P_{\omega^i}(\mathbf{h}_j^{L,i} | v_j \in \mathcal{V}^i), \tag{3}$$

where $\hat{y}_{v_j}^i$ represents the prediction for node v_j. The readout function P_{ω^i} encompasses methods such as mean pooling, where ω^i is the parameter for the readout function.

2.1.3 Subgraph Federated Optimization. The objective of subgraph federated learning is defined as follows,

$$\min_{\mathbf{w}} \sum_{i=1}^{M} p_i \mathcal{R}_i(\mathcal{F}_i(\mathbf{w})), \tag{4}$$

where the term p_i denotes the model aggregation weight, with the constraint that $\sum_{i=1}^{M} p_i = 1$. The function $\mathcal{R}_i(\cdot)$ corresponds to the local empirical risk function and is formally expressed as

$$\mathcal{R}_i(\mathcal{F}_i(\mathbf{w})) := \mathbb{E}_{(\mathcal{G}_i, \mathcal{Y}_i)}[\mathcal{L}(\mathcal{F}_i(\mathbf{w}; \mathcal{G}_i), \mathcal{Y}_i)], \tag{5}$$

where $\mathcal{L}(\mathcal{F}_i(\mathbf{w}; \mathcal{G}_i), \mathcal{Y}_i)) := \frac{1}{|\mathcal{V}_i|} \sum_{v_i \in \mathcal{V}} l(\mathcal{F}_i(\mathbf{w}; \mathcal{G}_i(v_i)), y_{v_i})$ is the local loss function, $\mathcal{F}_i(\cdot)$ is the GNN model for client C_i.

To optimize the objective function, we consider the classic federated algorithm, FedAvg algorithm [27], for illustration. In each round t, the central server sends the global model $\mathbf{w}^t$ to all local clients. Subsequently, each client performs multi-steps of the Stochastic Gradient Descent (SGD) optimization method to refine their local models. Following this local optimization, the clients transmit their updated local models as $\mathbf{w}_i^t$ back to the central server. Finally, the central server aggregates these local models by weights p_i to obtain the next round's global model $\mathbf{w}^{t+1} = \sum_i p_i \mathbf{w}_i^t$. This optimization process iterates over a specified number of T rounds.

2.2 Federated Unlearning

Approximate Federated Unlearning. In this paper, we specifically focus on client-level federated unlearning. In this scenario, multiple clients want to opt out of the federation and eliminate their contribution from the global model. Assume that there are N_t clients who want to opt out of the federation. We refer to these clients as target clients $\mathcal{I} = \{\mathcal{I}_1, \cdots, \mathcal{I}_{N_t}\} \subseteq C$. Naive retraining methods from scratch are computationally expensive and impractical, especially when confronted with frequent removal requests.

Objective. The objective of this paper is to solve approximate federated unlearning, which can eliminate the contribution of the subgraph in target clients while maintaining comparable model utility comparable to full retraining.

Unlearning Verification. The backdoor trigger is one of the most widely adopted verification methods for evaluating the performance of federated unlearning methods [6, 9, 15]. In practical scenarios, target clients utilize their datasets, incorporating a fraction of the data injected with backdoor triggers corresponding to specific target labels. The resulting global model learns the correlation between the trigger patterns and the target labels. By denoting the backdoor trigger as g, a successful trigger would lead the GNN model to classify into the target label,

$$\mathcal{F}_i \oplus g(\mathbf{w}; \mathcal{G}_i(v_i)) = y_\tau, \tag{6}$$

where y_τ represents the target label. A successful unlearning method should disentangle this correlation, resulting in a lower attack success rate.

$$\mathcal{F}_i \oplus g(\mathbf{w}; \mathcal{G}_i(v_i)) = y_{v_i}, \tag{7}$$

Threat Model. This work focuses on semi-honest scenarios, excluding considerations of malicious clients or model replacement attacks. The global model is expected to exhibit robust performance on clean datasets, ensuring that introducing a backdoor trigger does not compromise its overall performance.

3 REINFORCED GRAPH KNOWLEDGE ENHANCEMENT SUBGRAPH FEDERATED UNLEARNING

Overview. Figure 3 illustrates the overall framework of Reinforced Graph Knowledge Enhancement Subgraph Federated Unlearning, which consists of two crucial modules: (1) Reinforced Federated Policy Sampler (RFPS) module aims to mitigate the cross-client interference. (2) Prompt Knowledge Enhancement Graph Distillation (PKEGD) module distills specific graph knowledge from the target clients for comprehensive subgraph unlearning. In particular, the first module RFPS, selectively samples subgraphs for unlearning to guarantee the global model utility. For the second module, PKEGD encodes key graph knowledge to graph prompts, which can be inserted with the sampled subgraph from RFPS. Finally, the global model is optimized via the tailored unlearning loss over the integrated subgraph for multi-client unlearning.

3.1 Reinforced Federated Policy Sampler

Unlearning across multiple clients would significantly decrease the model utility, especially with cross-client node interference. Based on the observation illustrated in Figure 2, the first intuition of our model is unlearning on a sampled subgraph. However, a key challenge is to remove the overlapping subgraph without accessing other clients' data. To address this issue, we propose a Reinforced Federated Policy Sampler, which leverages a reinforcement learning algorithm to enable the agent to determine the optimal policy for sampling a subgraph. The graph sampling process is formulated into a general decision $\mathcal{M} = \{\mathcal{S}^{(i)}, \mathcal{A}^{(i)}, \mathcal{P}^{(i)}, R\}$ for client $C_i \in C$. Here, $\mathcal{S}^{(i)} = \{s_t^{(i)}\}$ represents the set of states comprising all possible intermediate and final sampled graphs. The set $\mathcal{A}^{(i)} = \{a_t^{(i)}\}$ represents the actions characterizing the sampled graph's behavior at each time step. $\mathcal{P}^{(i)}$ is the transition dynamics specifying the possible outcomes of carrying out an action. $R(s_t^{(i)})$ is a reward function specifying the reward after reaching the state $s_t^{(i)}$.

State. For a client $C_i \in C$, we define the state as all possible sampled graphs. Specifically, each state s_t is represented by a graph $\mathcal{G}_i' = (\mathcal{V}_i', \mathcal{E}_i', \mathcal{X}_i')$, where $\mathcal{E}' = \{(u, w) \mid u, w \in \mathcal{V}' \text{ and } (u, w) \in \mathcal{E}\}$, and $\mathcal{X}_i'$ represents the corresponding node features.

Action. The action space encompasses all possible combinations of selected k nodes, ranging from the original graph $\mathcal{G}^{(i)}$. The $a_t^{(i)} = \{0, 1\}^n$ represents an action, where the i-th of value a_t indicates whether node node v_i is selected (1) or not (0). Specifically, the action is generated by a policy function $\pi_\theta(a_t^{(i)} | s_t^{(i)})$, which takes the state and the original graph as input, where θ represents the parameters. In particular, the policy network $\pi_\theta(a_t^{(i)} | s_t^{(i)})$ consists of node embedding and action prediction functions.

Node Embedding. To predict actions, the policy network first computes the node embedding of the input graph using the node embedding function $g_\varphi(\cdot)$, parameterized with φ. Specifically, we employ a graph neural network to calculate the node embedding:

$$H^{(l+1)} = \text{AGG}\left(\sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{E} \tilde{D}^{\frac{1}{2}} H^l W\right)\right), \tag{8}$$

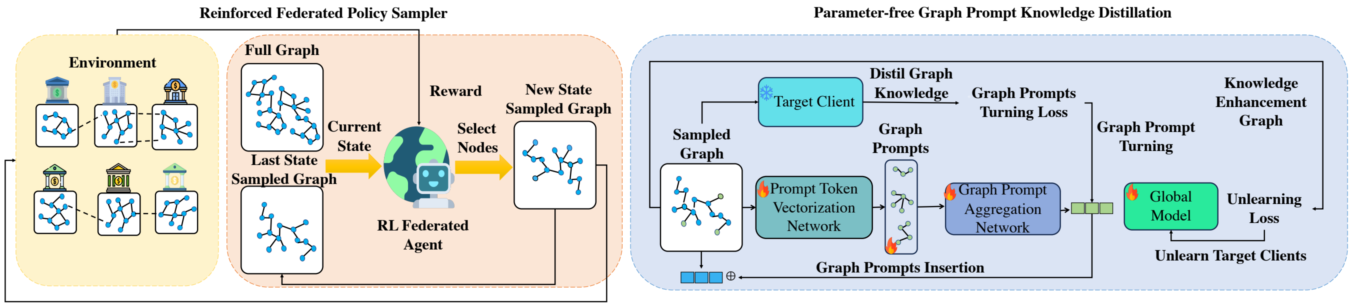


Figure 3: Framework overview.

where $H^{(l+1)}$ represents the node features matrix at layer $l+1$. AGG$(\cdot)$ denotes the aggregation function, σ is the activation function, $\tilde{D}$ is the normalized degree matrix, $\tilde{E}$ is the adjacency matrix, and W is the learnable weight matrix.

Action Prediction. The action $a_t^{(i)}$ is predicted based on two components: the probability-sampling-based action $a_{P_t}^{(i)}$ and the learnable action $a_{L_t}^{(i)}$,

$$a_t^{(i)} = \mathrm{CONCAT}(a_{P_t}^{(i)}, a_{L_t}^{(i)}), \tag{9}$$

where CONCAT represents the concatenation function, and $a_{P_t}^{(i)} \sim P_\phi$ is a stochastic probability distribution with parameter ϕ. The learnable action $a_{L_t}^{(i)}$ is defined by the following equations,

$$a_{L_t}^{(i)} = \mathrm{SOFTMAX}(\mathrm{MLP}(\mathrm{CONCAT}(g_\varphi(s_{t-1}^{(i)}), g_\varphi(\mathcal{G}_i)))), \tag{10}$$

where SOFTMAX$(\cdot)$ denotes the softmax function. The final sampled graph is denoted as $\mathcal{G}_i' = (\mathcal{V}_i', \mathcal{E}_i', \mathcal{X}_i')$, where $\mathcal{E}' = \{(u,w) \mid u,w \in \mathcal{V}' \text{ and } (u,w) \in \mathcal{E}\}$, and $\mathcal{X}_i'$ signifies the corresponding node features. The sampled nodes in $\mathcal{V}_i'$ are derived from the action a_t and maintain their K-hop neighborhood $\mathcal{N}_K(v)$.

$$\mathcal{V}_i' = \{v \mid v \in \mathcal{V}_i, v \sim a_t^{(i)}\} \cup \textstyle\bigcup_{v \in \mathcal{V}_i} \mathcal{N}_K(v)\,, \tag{11}$$

where $\mathcal{N}_K(v)$ represents the set of K-hop neighboring nodes in the original graph $\mathcal{G}_i$. Here, the sampling rate is denoted by $s = \frac{k}{|\mathcal{V}_i|}$ to control the number of sampled nodes.

Transition. After taking action $a_t^{(i)}$, the environment undergoes a transition, and the state changes from $s_t^{(i)}$ to $s_{t+1}^{(i)}$, governed by the transition probability $P(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)})$.

Reward Design. The environment yields a reward $R(r_t^{(i)})$ to assess the action $a_t^{(i)}$ in the state $s_t^{(i)}$ for client C_i. The environment aims to assign a positive reward when the sampled graph does not significantly impact the model's utility while preserving unlearning performance. The reward function is designed as follows,

$$R(s_t^{(i)}) = \begin{cases} \frac{1}{\mathrm{acc}_0 - \mathrm{acc}_1 + 1} & \text{if } \mathrm{acc}_1 < \mathrm{acc}_0 \\ -1 & \text{otherwise}\,, \end{cases} \tag{12}$$

where acc_0 denotes the client's accuracy on the pre-unlearning model. The acc_1 represents the accuracy post the client's unlearning process. To conduct the unlearning procedure, a gradient ascent is executed on the presently sampled graph $\mathcal{G}'_i$ at time t.

Federated Policy Gradient Training. The objective of the agent is to train an optimal policy network capable of maximizing the expected reward. The training of the policy network involves defining the overall loss as follows.

$$\mathcal{L}(\theta) = \sum_i \mathbb{E}_{\pi_\theta(a_t^{(i)}|s_t^{(i)})}[(r_t^{(i)} \nabla \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})], \tag{13}$$

The optimization of the policy network utilizes Adam optimizer, which is detailed in Appendix A.

3.2 Parameter-free Graph Prompt Knowledge Distillation

Recent research [31, 39] demonstrates the ability of graph prompts to enhance performance on downstream tasks by leveraging the knowledge from pre-trained models and increasing the expressiveness of downstream graph representations. Graph prompts can serve as a medium for distilling knowledge from pre-trained models [32]. However, existing graph prompt methods are parameter-extensive, which are computationally expensive to improve downstream tasks [31]. In this module, we introduce the parameter-free graph prompt to distill subgraph knowledge from the target clients.

3.2.1 *Graph Prompt Generation.*

We first learn a graph prompt generator $f_\phi(\cdot)$ to derive graph prompts, which comprises three major components: graph prompt token vectorization, graph prompt aggregation, and graph prompts insertion.

Graph Prompt Token Vectorization. To derive the graph prompt, we first generate vectorized graph prompt tokens, which can be done via a Multi-Layer Perceptron (MLP),

$$MLP(\mathcal{G}_i') = \mathcal{G}_{\mathbf{v}_q'}^{(i)}, \tag{14}$$

$$\mathcal{G}_{\mathbf{v}_q'}^{(i)} = (\mathcal{P}, \mathcal{S}), \tag{15}$$

$$\mathbf{s}_{ij} = \begin{cases} \sigma(\mathbf{p}_i \cdot \mathbf{p}_j) & \sigma(\mathbf{p}_i \cdot \mathbf{p}_j) > \eta \\ 0 & otherwise \end{cases}, \tag{16}$$

where $\mathcal{G}_{\mathbf{v}_q'}^{(i)}$ is the graph prompt for node $\mathbf{v}_q'$, $\mathcal{P} = \{\mathbf{p}_1, \cdots, \mathbf{p}_q\}$ denotes the vectorized tokens. The superscript indicates the index for subgraph $\mathcal{G}_i'$. $\mathcal{S} = \{(\mathbf{p}_i, \mathbf{p}_j) | \mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}\}$ denotes the pairwise connected relation among the tokens. $\mathbf{s}_{ij}$ is the connection relation between token $\mathbf{p}_i$ and $\mathbf{p}_j$. σ is the sigmoid function and η is the hyperparameter serving as a control threshold.

Graph Prompt Aggregation. To integrate the distilled graph prompts with the sampled graph, we initiate the process by aggregating the graph tokens. Let $f_\varphi(\mathcal{G}^{(i)}_{\mathbf{v}'_q})$ denote a universal aggregation function with parameter φ. The initial step involves aggregating the graph prompt tokens by taking into account the similarity among tokens,

$$\mathbf{h}_i^l = \sigma\Big(\sum_{\mathbf{p}_j \in \mathcal{N}(\mathbf{p}_j)} \mathbf{s}_{ij}\mathbf{W}_l \mathbf{h}_j^{k-1}\Big), \tag{17}$$

where $\mathbf{h}_i^l$ is the embedding of token $\mathbf{p}_i$ at the l-th layer, and $\mathbf{W}_l$ is the function weights at layer l. The the universal aggregation function paramater φ is defined as $\{\mathbf{W}_k\}_{k=1}^L$. $\mathcal{N}(\mathbf{p}_j)$ represents the connected relation set of token $\mathbf{p}_j$. Finally, mean global pooling is applied to obtain the final embedding,

$$\mathbf{p}'_i = \frac{1}{q}\sum_{j=1}^{q} \mathbf{h}_j^{(L)}, \tag{18}$$

where $\mathbf{h}_j^{(L)}$ is the feature vector of the token $\mathbf{p}_j$ at final layer L.

Graph Prompts Insertion. To incorporate the aggregated graph prompts into the sampled graph $\mathcal{G}'_i$, we firstly generate N_q graph prompts $\{\mathcal{G}^{(i)}_{\mathbf{v}'_j}\}_{j=1}^{N_q}$. Then, we randomly select N_q nodes from the sampled graph $\mathcal{G}'_i$ for insertion. The embedding $\{\mathbf{p}'_i\}_{i=1}^{N_q}$ of aggregated graph prompt tokens are added to the sampled graph $\mathcal{G}'_i$.

$$\hat{\mathbf{x}}'_i = \begin{cases} \mathbf{x}'_i + \mathbf{p}'_i & \mathbf{x}'_i \in \mathcal{T}_x \\ \mathbf{x}'_i & otherwise, \end{cases} \tag{19}$$

where $\mathcal{T}_x = \{\mathbf{x}'_1, \cdots, \mathbf{x}'_q\}$ denote the sampled nodes feature set.

The final sampled graph inserted with graph prompts is termed knowledge enhancement graph $\hat{\mathcal{G}}'_i = (\mathcal{V}'_i, \mathcal{E}'_i, \hat{\mathcal{X}}'_i)$, where $\hat{\mathcal{X}}'_i = \{\hat{\mathbf{x}}'_1, \cdots, \hat{\mathbf{x}}'_{n'}\}$ is augmented feature set, where $\hat{\mathbf{x}}'_q$ is the enhanced target node feature.

3.2.2 *Graph Prompt Turning.*

The goal of graph prompts turning is dist the specific graph knowledge learned from the target client by fine-tuning the task parameters, denoted as $\omega = \{\{\mathcal{G}^{(i)}_{\mathbf{v}'_j}\}_{j=1}^{N_q}, \{\phi\}\}$. The graph prompts turning loss is defined as follows,

$$\mathcal{L}_{\hat{\mathcal{G}}'_i} = l(F_i(\mathbf{w}; \hat{\mathcal{G}}'_i), \mathcal{Y}'_i), \tag{20}$$

where $l(\cdot)$ is the cross-entropy node classification loss, $\mathcal{Y}'_i$ is the label set for graph $\mathcal{G}'_i$. The model weight of the local model $F_i(\mathbf{w})$ in the target client is fixed.

3.3 Gradient Descent Unlearning

Based on the knowledge enhancement graph, the target clients are ready for unlearning. To provide guidance for the unlearning process, we employ the average weights of the remaining models as a constraint on the global model. The final unlearning loss is defined as follows,

$$\begin{aligned}\mathcal{L}_u(\mathcal{F}(\mathbf{w})) = \sum_{C_i \in \mathcal{I}} [-\frac{1}{|\mathcal{V}_i|} \sum_{u' \in \mathcal{V}'} [l(\mathcal{F}_i(\mathbf{w}; \hat{\mathcal{G}}_i(u')), y_{u'})] \\ + \lambda_u \cdot \|\mathbf{w}_i - \mathbf{w}^*\|^2], \end{aligned}\tag{21}$$

Algorithm 1: Reinforced Graph Knowledge Enhancement Subgraph Federated Unlearner

```
Input: Client data $\{\mathcal{G}_i\}_{i=1}^M$, pretrained policy sampler
       $\pi_\theta(\cdot)$, global model $F(\mathbf{w})$, fine-tuning epoch $E$.
Result: Enhanced compact subgraph $\hat{\mathcal{G}}'_i = (\mathcal{V}'_i, \mathcal{E}'_i, \hat{\mathcal{X}}'_i)$
1  Use pre-trained policy sampler $\pi_\theta(\cdot)$ to sample the graphs
     $\{\mathcal{G}'{}_i\}_{i=1}^{N_t}$ based on Equation 11.;
2  for each target client $C_i \in \mathcal{I}$ in parallel do
3      for $k = 1$ to $E$ do
4          Generate the graph prompts based on Equation 14;
5          Aggregate the tokens embedding based on
             Equations 17 and 18;
6          Insert the graph tokens based on Equation 19;
7          Compute the prompt enhancement loss by
             Equation 20;
8          Update $\omega = \{\{\mathcal{G}^{(i)}_{\mathbf{v}'_j}\}_{j=1}^{N_q}, \{\varphi\}\}$ using Adam
             optimization with the gradients of $\mathcal{L}_{\hat{\mathcal{G}}'_i}$) ;
9      end
10 end
11 Use Adam optimizer to optimize the unlearning loss based
     on Equation 21;
12 Return unlearned model $\mathcal{F}(\mathbf{w}_{C\backslash\mathcal{I}})$.
```

where $\mathbf{w}^* = \frac{1}{|C\backslash\mathcal{I}|}\sum_{i \in C\backslash\mathcal{I}} \mathbf{w}_i$ is the guided model constraint, and λ_u is the constrain parameter. $l(\cdot)$ is the corss-entropy loss. The final unlearned global model is denoted as $\mathcal{F}\mathbf{w}_{C\backslash\mathcal{I}}(\cdot)$.

We also employ the Adam optimizer to minimize the unlearning loss. After unlearning, the performance of the updated global model may decrease for other clients. To mitigate this issue, the server conducts a few rounds of federated learning involving the remaining clients [6, 8, 15]. Empirical studies demonstrate that, in practice, only a few rounds are sufficient to keep the new global model $\mathbf{w}_{C\backslash\mathcal{I}}$ up to date. The complete procedure for subgraph federated unlearning is reported in Algorithm 1.

4 EXPERIMENTS

In this section, we empirically evaluate the proposed framework and compare it with the state-of-the-art federated unlearning methods. We report the unlearning performance on the real-world dataset. To be more specific, we aim to answer the following research questions. **RQ 1**: Does the proposed method effectively eliminate contributions from multiple clients compared to state-of-the-art unlearning methods? **RQ 2**: How effective are the proposed components within the unlearning framework? **RQ 3**: How robust is the proposed method concerning changes in hyperparameter values? **RQ 4**: Does the unlearning process compromise the privacy of the clients?

4.1 Environmental Setup

Datasets. The distributed subgraph is constructed by partitioning the datasets into a predetermined number of participants. In subgraph federated learning (FL), each client possesses a subgraph as a subset of the original graph. Specifically, we utilize a set of real-world graph datasets, including Cora [2], Pubmed [29], Photo [26], and Cs [2]. In the default setting, the overlapping nodes are set to 0.2. Various overlapping nodes setting experiments is presented in Appendix B.1.

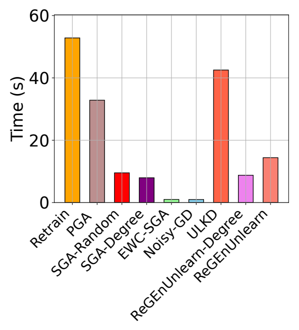
Evaluation Metrics. For evaluating the effectiveness of federated unlearning methods, we use the widely acknowledged metrics [6, 8, 15]. (1) Attack Success Rate (ASR): The backdoor trigger is employed to evaluate the efficacy of unlearning methods. ASR quantifies the successful classification of manipulated data into the target label [38]. A lower ASR signifies heightened proficiency in unlearning. (2) Model Accuracy (MA): We assess the accuracy of the global model after the unlearning process as the model utility.

Baselines. We consider the following federated unlearning methods: *Retraining from Scratch*: This approach involves retraining on an initialized model using the data from the remaining clients. *Projected Gradient Ascent (PGA)* [15]: PGA is an unlearning method designed to maximize the empirical loss on the local clients. *EWC-SGA* [38]: EWC-SGA combines elastic weight consolidation and stochastic gradient ascent to enable the removal of client's contribution without the need for full model retraining. *Noisy-GD* [7]: Noisy-GD is a robust data-deletion method that ensures differential privacy constraints are met. *ULKD* [37]: ULKD is a server storage history method used to eliminate target client sharing and improve the model's utility through distillation. In evaluating sampled-based methods, we utilize random sampling and node degree for graph selection. Stochastic Gradient Ascent (SGA) is then applied to eradicate target client information in the sampled graph, denoted as *SGA-Random* and *SGA-Degree*. *ReGEnUnlearn-Degree* is the variant of our proposed method by using the vanilla heuristics degree-based graph sampler.

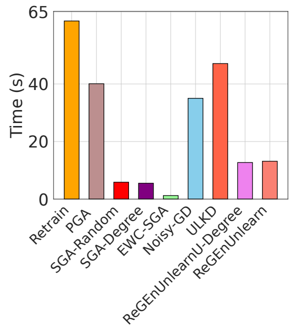
Implementations Details. All code is executed within the PyTorch framework. The experiments are carried out on two servers: a Linux CentOS Server equipped with 4 RTX 3090 GPUs and a Linux Ubuntu Server with 1 A800 GPU. The unlearning scenario considered involves the server conducting federated learning training and subsequently receiving $N_t = 4$ target clients' requests to opt out of the federation. In the context of federated subgraph learning, there are $M = 10$ clients. The different numbers of target clients are discussed in Sections 4.4. Due to the page limit, the variations in the number of clients are presented in Appendix B.1. For the graph backdoor trigger, we employ the Erdos-Rényi (ER) model to generate the graph trigger, with a trigger size of five, and use the Gaussian distribution for the trigger features. We consider FedAvg as the default federated algorithm. Additionally, we evaluate the effectiveness of the proposed ReGEnUnlearn under more advanced federated algorithms in Appendix B.2. Each experiment is iterated five times to derive average results.

4.2 RQ 1: Main Results

Table 1 presents comprehensive results for the proposed methods and respected baseline models across two metrics. We can make the following observations: the proposed framework's superior performance in ASR across four datasets when compared to all baseline models. In particular, ReGEnUnlearn achieves nearly 100% elimination of multi-client contributions on datasets Cora, Pubmed, and Cs. Furthermore, our proposed framework demonstrates superiority model utility when compared to retraining methods across all four datasets. In specific, ReGEnUnlearn achieves performance levels of 100.66%, 100.69%, 103.47%, and 102.42%, which are directly comparable to retraining methods. We further report the running time of federated unlearning methods in Figure 4 and 5. Specifically, our proposed framework achieves 3.66×, 4.7×, 16.07×, and 9.08× speedup when compared to retraining methods on four datasets.

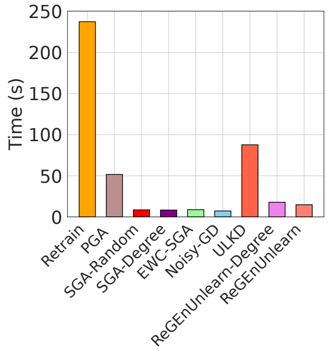


(a) Cora

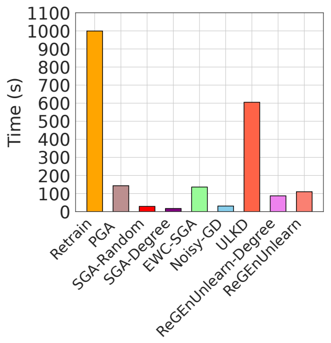


(b) Pubmed

Figure 4: Running time on Cora and Pubmed.



(a) Photo



(b) Cs

Figure 5: Running time on datasets Photo and Cs.

4.3 RQ 2: Ablation Study

To evaluate the effectiveness of each module within ReGEnUnlearn, we executed an ablation study across four datasets, employing two metrics. Specifically, we investigated the following variants: (1) *ReGEnUnlearn-WoPolicy* excludes the policy sampler module using the random sampler to replace it. (2) *ReGEnUnlearn-WoPrompt* excludes the graph prompt empowerment module.

As delineated in Table 2, the following observations across four datasets. Each of the two modules significantly contributes to the overall performance. Notably, the removal of any single module results in performance degradation. Furthermore, our investigation reveals a performance decline when substituting the policy sampler with the vanilla heuristics sampler method, underscoring the tangible benefits of employing a learnable sampling strategy to navigate the graph. Additionally, a performance improvement

Table 1: Overall Performance of Federated Unlearning

	Cora		Pubmed	
Methods	Attack Success Rate	Model Accuracy	Attack Success Rate	Model Accuracy
Retrain	0	0.8720 (0.0244)	**0**	0.8548 (0.0032)
PGA	0.2 (0.4472)	0.8594 (0.0532)	0.2 (0.4)	0.6621 (0.0203)
SGA-Random	0.2 (0.4472)	0.8628 (0.0277)	0.4 (0.5477)	0.746 (0.0416)
SGA-Degree	0.2 (0.4472)	0.8696 (0.0382)	0.4343 (0.5211)	0.7429 (0.0353)
EWC-SGA	0	0.7174 (0.0505)	0.2 (0.4)	0.7694 (0.0260)
Noisy-GD	0.9993 (0.0014)	0.7430 (0.0578)	0.9999 (0.0001)	0.8383 (0.0236)
ULKD	0	0.4029 (0.1808)	0.3998 (0.4897)	0.7178 (0.2179)
ReGEnUnlearn-Degree (Ours)	0	0.8763 (0.0227)	0.0159 (0.0283)	**0.8639 (0.0169)**
ReGEnUnlearn (Ours)	**0**	**0.8778 (0.0202)**	0.0027 (0.0048)	0.8607 (0.014)
	Photo		Cs	
Methods	Attack Success Rate	Model Accuracy	Attack Success Rate	Model Accuracy
Retrain	**0**	0.7 (0.0751)	0	0.8437 (0.0128)
PGA	0	0.5008 (0.0537)	0.2 (0.4472)	0.8379 (0.0680)
SGA-Random	0.3285 (0.3505)	0.6287 (0.0600)	0.2 (0.4472)	0.8415 (0.0371)
SGA-Degree	0.2364 (0.1388)	0.6327 (0.2364)	0.2 (0.4472)	0.8494 (0.0191)
EWC-SGA	0.6914 (0.3908)	0.7128 (0.0287)	0.4007 (0.5471)	0.8505 (0.0287)
Noisy-GD	1	0.6991 (0.0300)	0.1995 (0.3985)	0.8650 (0.0045)
ULKD	0.2 (0.4)	0.4994 (0.0420)	0	0.3569 (0.1294)
ReGEnUnlearn-Degree (Ours)	0.2629 (0.31)	0.7229 (0.0431)	0	**0.8730 (0.0068)**
ReGEnUnlearn (Ours)	0.1703 (0.1202)	**0.7243 (0.0725)**	**0**	0.8642 (0.0137)

is observed when omitting the graph prompts modules, providing empirical validation for the efficacy of unlearning the contributions of target clients. Overall, the above results verify the effectiveness of the proposed ReGEnUnlearn framework.

4.4 RQ 3 Parameter Analysis

Next, we investigate the parameter sensitivity of our proposed approach. We present findings related to the number of target clients N_t, the sampling rate s, and the number of graph tokens q on the Cora dataset across three metrics. Similar trends are observed across the other three datasets.

First, we vary the value of N_t from 1 to 5, as depicted in Figure 6(a). The observations are summarized as follows. The proposed ReGEnUnlearn framework successfully eliminates all client contributions across the range of target clients. Additionally, an increase in K initially leads to a rise in subtle MA.

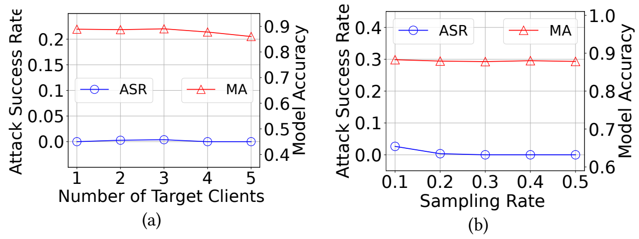


Figure 6: (a) Effect of the number of target clients. (b) Effect of the number of target clients.

Secondly, we adjust the sampling rate s within the range of 0.1 to 0.5, as depicted in Figure 6(b). Regarding ASR, a reduction is observed as the sampling rate increases, underscoring the robustness of our method. A parallel trend is also discernible in terms of MA.

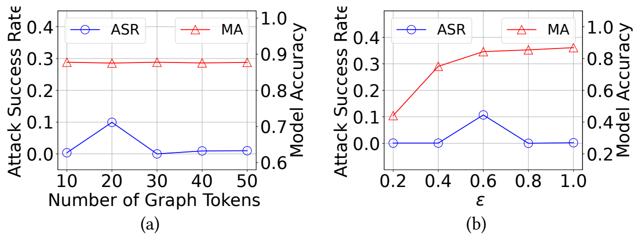


Figure 7: (a) Effect of the number of tokens. (b) Effect of the number of ϵ.

Finally, we vary the number of graph tokens q within the range of 10 to 50, as depicted in Figure 7(a). Our observations yield the following synthesis: with an increase in q, the ASR exhibits an initial ascent followed by a subsequent descent. Simultaneously, MA maintains a relatively stable profile devoid of pronounced variations.

Overall, adjusting the above hyperparameters induces performance variations within a reasonable range across three metrics, thereby demonstrating the robustness of our approach.

4.5 RQ 4: Privacy Analysis

Privacy Protection. In the phase of federated unlearning, the direct uploading of the GCN model by target clients poses potential privacy concerns. The gradients in the model update may inadvertently expose private information to the target clients, as the GCN model gradients inherently encode graph information preferences. To protect the privacy of the target clients, Differential Privacy (DP) can be employed during the unlearning stage.

$$\mathbf{w}_i + \mathcal{N}(0, \sigma^2\mathbf{I}) \tag{22}$$

Table 2: Ablation Study of Federated Unlearning

	Cora		Pubmed	
Methods	Attack Success Rate	Model Accuracy	Attack Success Rate	Model Accuracy
ReGEnUnlearn-WoPolicy	0	0.8744 (0.0264)	0.1293 (0.2574)	0.7882 (0.0321)
ReGEnUnlearn-WoPrompt	0.6 (0.5477)	0.8633 (0.0188)	0.3890 (0.5273)	0.8264 (0.03324)
ReGEnUnlearn (Ours)	**0**	**0.8778 (0.0202)**	**0.0027 (0.0048)**	**0.8607 (0.014)**
	Photo		Cs	
Methods	Attack Success Rate	Model Accuracy	Attack Success Rate	Model Accuracy
ReGEnUnlearn-WoPolicy	0.282 (0.3123)	0.7002 (0.0528)	0.2 (0.4)	0.8605 (0.01467)
ReGEnUnlearn-WoPrompt	0.3918 (0.4803)	0.6212 (0.0960)	0	0.8617 (0.0099)
ReGEnUnlearn (Ours)	**0.1703 (0.1202)**	**0.7243 (0.0725)**	**0**	**0.8642 (0.0137)**

where $\mathcal{N}(0, \sigma^2\mathbf{I})$ is the Gaussian noise with a mean of 0 and standard deviation $\sigma^2\mathbf{I}$. Let $\sigma = \sqrt{2\log\frac{1.25}{\delta}}/\epsilon$, as established by [1], ensuring that ReGEnUnlearn adheres to (ϵ, δ)-differential privacy.

Effect of ϵ. We set δ to 1×10^{-5} and adjust ϵ from 0.2 to 1. Figure 7(b) presents the comprehensive results. Notably, as ϵ increases, there are subtle changes, yet the overall impact of unlearning remains consistent. Additionally, a trade-off exists between privacy protection and model utility. As the model's privacy protection increases, its utility decreases.

5 RELATED WORK

5.1 Subgraph Federated Learning

Recently, researchers have made substantial progress in federated subgraph learning [36, 40, 43]. Various FL frameworks have been designed for graph learning tasks, encompassing recommendation [37], graph classification [42], and node classification [45], among others [17, 33]. For instance, Wu *et al.* [37] introduced a federated framework for privacy-preserving Graph Neural Network (GNN)-based recommendation systems. This framework enables collective training of GNN models from decentralized user data. He *et al.* [16] proposed FedGNN, a unified framework applicable to graphs from diverse domains. In federated subgraph learning, a key challenge involves addressing missing link problems. Zhang *et al.* [45] introduced FedeSage+, a subgraph federated unlearning framework that trains neighborhood generators along with FedSage to handle missing links across local subgraphs. Baek *et al.* [4] presented FedPub, a federated subgraph framework utilizing functional embeddings to construct client relations based on similarity. Another challenge lies in defending against poisoning attacks. Recent studies indicate that federated subgraph systems are vulnerable to backdoor attacks, making them susceptible to graph triggers. For example, Liu *et al.* [21] formally proposed a federated graph backdoor framework capable of attacking federated graph systems. However, there is a notable absence of attention to privacy issues within the context of machine unlearning. To address this gap, to the best of our knowledge, we are the first to explore subgraph federated unlearning. We introduce ReGEnUnlearn framework for efficient and comprehensive unlearning of multiple target clients.

5.2 Federated Unlearning

Recently, federated unlearning [6, 18, 25, 28, 44, 46] has garnered significant research attention. Two scenarios exist in federated unlearning: sample-level federated unlearning [8, 12, 20, 22, 47] and client-level federated unlearning [5, 37, 38, 41, 46]. Sample-level unlearning is a natural extension of the centralized setup. In such settings, FL systems are tasked with requesting the forgetting of a particular category or subset. For example, Wang *et al.* [34] proposed a federated unlearning framework capable of scrubbing specific categories. Liu *et al.* [24] proposed a rapid training approach to completely erase data samples from a trained FL model. In client-level federated unlearning, the FL system is tasked with requesting the forgetting of the client's entire contribution in a cross-silo scenario. For instance, Wu *et al.* [37] propose that the server stores historical local client information. When clients opt out of the federation, the server eliminates their contributions and uses knowledge distillation to maintain model utility. Halimi *et al.* [15] proposed a gradient descent method to unlearn the client's entire contribution. Subgraph federated unlearning primarily occurs in cross-silo scenarios where multiple institutes (e.g., banks and hospitals, *et al.*) hold a subgraph and train FL models under strict privacy regulations. Therefore, our primary focus is on considering how to eliminate the entire contributions of multiple clients.

6 CONCLUSIONS

In this paper, we presented ReGEnUnlearn, a subgraph federated unlearning framework for efficient and comprehensive unlearning of multiple target clients. By sampling the graphs and distilling graph knowledge, the proposed framework improves both model utility and unlearning performance. Specifically, we introduce the *Reinforced Federated Policy Sampler* (RFPS) to learn optimal sampling strategies that minimize the interference among cross-client subgraphs. By interacting with the federated graph sampling environment, the agent learns to selectively forget an optimal subgraph of target clients, thus preserving the global model utility. Moreover, we propose the *Parameter-free Graph Prompt Knowledge Distillation* (PGPKD) module, which retains the unique graph knowledge contributed by the target clients, thereby facilitating comprehensive unlearning via a tailored gradient ascent objective. We conduct extensive experiments under diverse federated settings to demonstrate the superiority of the proposed framework over state-of-the-art federated unlearning approaches. It is also worth mentioning that the framework is federated algorithm-agnostic, which means it can be easily adopted in other subgraph federated scenarios. In the future, we plan to apply ReGEnUnlearnon other tasks such as federated graph classification and link prediction tasks.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318.

[2] Mehdi Azabou, Venkataramana Ganesh, Shantanu Thakoor, Chi-Heng Lin, Lakshmi Sathidevi, Ran Liu, Michal Valko, Petar Veličković, and Eva L Dyer. 2023. Half-Hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning*. PMLR, 1341–1360.

[3] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized Subgraph Federated Learning. (2023).

[4] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized subgraph federated learning. In *International Conference on Machine Learning*. PMLR, 1396–1415.

[5] Xiaoyu Cao, Jinyuan Jia, Zaixi Zhang, and Neil Zhenqiang Gong. 2023. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1366–1383.

[6] Tianshi Che, Yang Zhou, Zijie Zhang, Lingjuan Lyu, Ji Liu, Da Yan, Dejing Dou, and Jun Huan. 2023. Fast Federated Machine Unlearning with Nonlinear Functional Theory. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 169, 28 pages.

[7] Rishav Chourasia and Neil Shah. 2023. Forget Unlearning: Towards True Data-Deletion in Machine Learning *(ICML'23)*. JMLR.org, Article 240, 46 pages.

[8] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2022. Sequential Informed Federated Unlearning: Efficient and Provable Client Unlearning in Federated Optimization. *arXiv preprint arXiv:2211.11656* (2022).

[9] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2022. Federated graph machine learning: A survey of concepts, techniques, and applications. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 32–47.

[10] Xiangshan Gao, Xingjun Ma, Jingyi Wang, Youcheng Sun, Bo Li, Shouling Ji, Peng Cheng, and Jiming Chen. 2022. Verifi: Towards verifiable federated unlearning. *arXiv preprint arXiv:2205.12709* (2022).

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

[12] Jinu Gong, Osvaldo Simeone, and Joonhyuk Kang. 2021. Bayesian variational federated learning and unlearning in decentralized networks. In *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 216–220.

[13] Jinu Gong, Osvaldo Simeone, and Joonhyuk Kang. 2022. Compressed particle-based federated bayesian learning and unlearning. *IEEE Communications Letters* 27, 2 (2022), 556–560.

[14] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated Unlearning: How to Efficiently Erase a Client in FL? *CoRR* abs/2207.05521 (2022). arXiv:2207.05521

[15] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated unlearning: How to efficiently erase a client in fl? *arXiv preprint arXiv:2207.05521* (2022).

[16] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. 2021. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145* (2021).

[17] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. 2021. SpreadGNN: Decentralized Multi-Task Federated Learning for Graph Neural Networks on Molecular Data. arXiv:2106.02743 [cs.LG]

[18] Guanghao Li, Li Shen, Yan Sun, Yue Hu, Han Hu, and Dacheng Tao. 2023. Subspace based Federated Unlearning. *arXiv preprint arXiv:2302.12448* (2023).

[19] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[20] Yuyuan Li, Chaochao Chen, Xiaolin Zheng, and Jiaming Zhang. 2023. Federated unlearning via active forgetting. *arXiv preprint arXiv:2307.03363* (2023).

[21] Fan Liu, Siqi Lai, Yansong Ning, and Hao Liu. 2023. Bkd-FedGNN: A Benchmark for Classification Backdoor Attacks on Federated Graph Neural Network. *arXiv preprint arXiv:2306.10351* (2023).

[22] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.

[23] Yang Liu, Zhuo Ma, Ximeng Liu, and Jianfeng Ma. 2020. Learn to forget: User-level memorization elimination in federated learning. *arXiv preprint arXiv:2003.10933* 1 (2020).

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

[25] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, and Xingliang Yuan. 2023. A Survey on Federated Unlearning: Challenges, Methods, and Future Directions. *arXiv preprint arXiv:2310.20448* (2023).

[26] Yi Luo, Guangchun Luo, Ke Yan, and Aiguo Chen. 2022. Inferring from References with Differences for Semi-Supervised Node Classification on Graphs. *Mathematics* 10, 8 (2022), 1262.

[27] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[28] Chao Pan, Jin Sima, Saurav Prakash, Vishal Rana, and Olgica Milenkovic. 2023. Machine Unlearning of Federated Clusters. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=VzwfoFyYDga

[29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[30] Ningxin Su and Baochun Li. 2023. Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[31] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (, Long Beach, CA, USA,) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 2120–2131.

[32] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. 2023. Graph Prompt Learning: A Comprehensive Survey and Beyond. *arXiv preprint arXiv:2311.16534* (2023).

[33] Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. 2023. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 9953–9961.

[34] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. 2022. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*. 622–632.

[35] Weiqi Wang, Zhiyi Tian, Chenhan Zhang, An Liu, and Shui Yu. 2023. BFU: Bayesian Federated Unlearning with Parameter Self-Sharing. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 567–578.

[36] Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. 2022. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4110–4120.

[37] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications* 13, 1 (2022), 3091.

[38] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. 2022. Federated unlearning: Guarantee the right of clients to forget. *IEEE Network* 36, 5 (2022), 129–135.

[39] Xuansheng Wu, Kaixiong Zhou, Mingchen Sun, Xin Wang, and Ninghao Liu. 2023. A survey of graph prompting methods: techniques, applications, and challenges. *arXiv preprint arXiv:2303.07275* (2023).

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[41] Hui Xia, Shuo Xu, Jiaming Pei, Rui Zhang, Zhi Yu, Weitao Zou, Lukun Wang, and Chao Liu. 2023. Fedme 2: Memory evaluation & erase promoting federated unlearning in dtmn. *IEEE Journal on Selected Areas in Communications* (2023).

[42] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems* 34 (2021), 18839–18852.

[43] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. 2022. Fedgcn: Convergence and communication tradeoffs in federated training of graph convolutional networks. *arXiv preprint arXiv:2201.12433* (2022).

[44] Wei Yuan, Hongzhi Yin, Fangzhao Wu, Shijie Zhang, Tieke He, and Hao Wang. 2023. Federated unlearning for on-device recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 393–401.

[45] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *Advances in Neural Information Processing Systems* 34 (2021), 6671–6682.

[46] Lefeng Zhang, Tianqing Zhu, Haibin Zhang, Ping Xiong, and Wanlei Zhou. 2023. Fedrecovery: Differentially private machine unlearning for federated learning frameworks. *IEEE Transactions on Information Forensics and Security* (2023).

[47] Yanci Zhang and Han Yu. 2022. Towards Verifiable Federated Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5686–5693.

A REINFORCED FEDERATED POLICY SAMPLER

Unlearning across multiple clients could significantly diminish the model's utility, particularly in the presence of interference from cross-client nodes. The key challenge is how to remove the overlapping subgraph without accessing data from other clients. To tackle this issue, we introduce the Reinforced Federated Policy Sampler. This approach employs a reinforcement learning algorithm, empowering the agent to discern the optimal policy for subgraph sampling. During the federated training stage, the server can execute the Reinforced Federated Policy Sampler Pre-training, facilitating its application in the subsequent unlearning stage. Further details about the algorithm are elucidated in Algorithm 2.

Algorithm 2: Reinforced Federated Policy Sampler Pre-training

```
Input: Target clients $\mathcal{I}$, subgraphs $\{\hat{\mathcal{G}}_i\}_i^M$, local unlearn
       epoch $n$.
Result: Pre-trained Policy Sampler $\pi_\theta(a_t|s_t)$
1  Server initializes $\theta$.;
2  for each round $t = 1, 2, \cdots$ do
3      for each target client $C_i \in C$ in parallel do
4          for $j = 1$ to $n$ do
5              Generate the action based on Equation 3.1 ;
6              Obtain the sampled graph based on Equation 11 ;
7              Compute the reward by Equation 12 ;
8              Update the local client
               $\mathbb{E}_{\pi_\theta(a_t^{(i)}|s_t^{(i)})}[(r_t^{(i)} \nabla \log \pi_\theta(a_t^{(i)} \mid s_t^{(i)})]$ by
               Adam optimizer. ;
9          end
10     end
11     $\theta_{t+1} \longleftarrow \sum_i \theta_{t+1}^{(i)}$ ;
12 end
13 Return $\theta$ to the server.
```

B FURTHER EXPERIMENTS

The statistical analysis of graph data is presented in Table 3. We extend our investigations by conducting additional experiments across diverse federated learning scenarios, including further parameter analysis and exploration of experiments under more advanced scenarios.

Table 3: Statistics analysis of the graph datasets.

Datasets	# of Nodes	# of Edges	# of Classes
Cora	2,708	5,278	7
Pubmed	19,717	44,324	3
Photo	7,650	238,163	8
Cs	18,333	163,788	15

B.1 Further Experiments Parameter Analysis

We further conduct the additional experiments under different parameter settings to evaluate the effect of parameters, including scenarios with different numbers of clients, regularization coefficient, and overlapping rates on dataset Cora.

Effect of number of clients M. We fix the the number of unlearned clients is 4, vary the number of clients from 10 to 30. Figure 8(a) reports the overall results. We have made the following observations. First, it becomes evident that the number of clients exerts negligible influence on unlearning performance. Notably, the ASR remains consistent across different configurations, showing a deviation of 0. Additionally, we increase the value of M, leading to a corresponding rise in the MA with the augmentation of supervised signals.

Effect of parameter λ_u. We vary the parameter λ_u from 0.2 from 1.0. Figure 8(b) reports the results across two metrics. First, we observe that with the increase of λ_u, the ASR first increases and then decreases. Second, with the increase of parameter λ_u

Effect of overlapping rate. We systematically vary the overlapping rate within the range of 0.1 to 0.5. The outcomes are illustrated in Figure 9(a), showcasing results across two key metrics. Initially, as the overlapping rate escalates, the ASR consistently maintains a value of 0, underscoring the effectiveness of our proposed method.

Table 4: Unlearning Experiments under the FexProx

Methods	Attack Success Rate	Model Accuracy
Retrain	0	0.8802 (0.0142)
PGA	0.4 (0.5477)	0.8652 (0.0193)
SGA-Random	0.2007 (0.4448)	0.8715 (0.01659)
SGA-Degree	0.4021 (0.4702)	0.8720 (0.0214)
EWC-SGA	0.4 (0.5477)	0.6560 (0.1541)
Noisy-GD	1	0.4116 (0.1418)
ULKD	1	0.8297 (0.0405)
ReGEnUnlearn-Degree (Ours)	0	0.8807 (0.0146)
ReGEnUnlearn (Ours)	**0**	**0.8807 (0.0132)**

Table 5: Unlearning Experiments under the FedPub

Methods	Attack Success Rate	Model Accuracy
Retrain	0	0.8778 (0.0142)
PGA	0.4036 (0.5445)	0.8643 (0.0160)
SGA-Random	0.4036 (0.5455)	0.8667 (0.0190)
SGA-Degree	0.4 (0.5477)	0.8700 (0.0189)
EWC-SGA	0.4 (0.5477)	0.6696 (0.1746)
Noisy-GD	1	0.4671 (0.1738)
ULKD	0	0.1891 (0.0747)
ReGEnUnlearn-Degree (Ours)	0	0.8773 (0.0140)
ReGEnUnlearn (Ours)	**0**	**0.8845 (0.0152)**

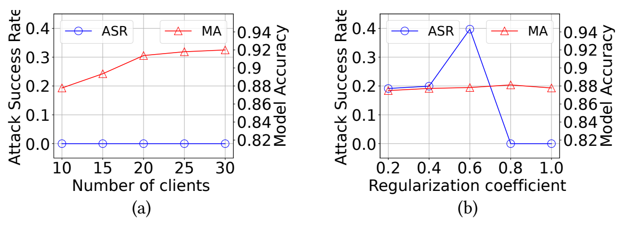


Figure 8: (a) Effect of number of clients M. (b) Effect of parameter λ_u

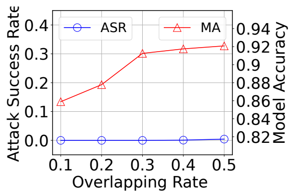
Furthermore, a discernible trend emerges where an uptick in the overlapping rate correlates with an increase in the MA, attributable to the concurrent rise in the number of samples.

B.2 Unlearning Experiments under Other Federated Scenarios

We further conduct the experiments on more advanced scenarios on datasets Cora. More specifically, we will conduct further experiments on more advanced federated algorithms and Non-IID Settings.

Advanced Federated Algorithms. We investigate federated unlearning scenarios, encompassing both a general federated algorithm (e.g., FedProx [19]) and one tailored for graph scenarios (e.g., FedPub [4]). Table 4-5 reports the overall experimental results across three metrics. We make the following observations. Firstly, our proposed framework successfully eliminates contributions from all clients for both FedProx and FedPub, in stark contrast to retraining methods, where traditional approaches fail to eliminate contributions from the target client. Additionally, we observe a reduction in model utility under more advanced federated algorithms compared to the FedAvg algorithm. For example, our proposed method achieves performance metrics of (100%, 100.76%) in comparison to retraining methods under FedProx and FedPub.

Non-IID Setting. To evaluate the effectiveness of subgraph federated unlearning in the Non-IID setting, we conduct additional experiments. Specifically, we utilize the Dirichlet function to partition the participants with a parameter set to 0.3. The results are presented in Table 6, and the following observations can be made. Firstly, unlearning multiple target clients is more challenging under the Non-IID setting, with most baselines struggling to eliminate all client contributions. Additionally, our methods consistently maintain approximately 100.43%. compare with the retraining methods.



(a)

Figure 9: (a) Effect of overlapping rate.

Table 6: Unlearning Experiments under Non-IID Setting

Methods	Attack Success Rate	Model Accuracy
Retrain	0	0.8908 (0.0119)
PGA	0.2 (0.4472)	0.8823 (0.0255)
SGA-Random	0.2 (0.4472)	0.8810 (0.0220)
SGA-Degree	0.2 (0.4472)	0.8871 (0.01684)
EWC-SGA	0.2 (0.4472)	0.6828 (0.1357)
Noisy-GD	0.9976 (0.0035)	0.5733 (0.2095)
ULKD	0.6 (0.5477)	0.4211 (0.1313)
ReGEnUnlearn-Degree (Ours)	0.1687 (0.3772)	0.8989 (0.0181)
ReGEnUnlearn (Ours)	**0**	**0.8947 (0.0209)**