RECONSTRUCT THEN RECTIFY: INFORMATION-CONSERVING NODE INTERACTION ESTIMATION FOR GRAPH UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph unlearning aims to remove the influence of specified nodes or edges from a trained graph neural network (GNN). We study efficient approximate node unlearning, whose target is to reproduce the behavior of retraining without claiming a certified deletion guarantee. Existing approaches can require partial retraining, full-parameter re-estimation, or accurate influence approximation. We propose *Reconstruct Then Rectify* (RTR), a post-hoc framework that leaves the base GNN parameters fixed. RTR first learns initial node-interaction estimates through information-conserving reconstruction and then refines them using a range–null-space decomposition under a linear inverse constraint. The refined interactions are subtracted from retained-node embeddings. Under the stated linear-inverse assumptions, the rectification preserves the constraint and does not increase the squared L_2 estimation error. Experiments on multiple datasets and four GNN backbones show substantial runtime reductions while maintaining utility close to retraining; we further evaluate poisoned-node removal and resistance to MIA.

1 INTRODUCTION

Machine unlearning removes the impact of some training data from the machine learning models upon requests (Cao & Yang, 2015). It is essential in many critical scenarios, such as the enforcement of laws concerning the protection of the right to be forgotten (Kwak et al., 2017; Pardau, 2018; Regulation, 2018), and the demands for model providers to revoke the negative effect of poisoned data (Rubinstein et al., 2009; Zhang et al., 2022), wrongly annotated data (Pang et al., 2021), or out-of-date data (Wang et al., 2022). Existing approaches pursue two distinct objectives: *certified unlearning* provides formal forgetting guarantees under explicit assumptions, but can trade off utility or applicability (Guo et al., 2020; Chien et al.; Wu et al., 2023b), whereas efficient *approximate unlearning* aims to reproduce the behavior of retraining as closely as possible without claiming a certified deletion guarantee.

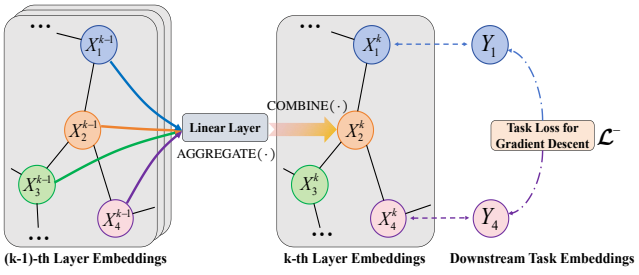


Figure 1: Brief description of graph models.

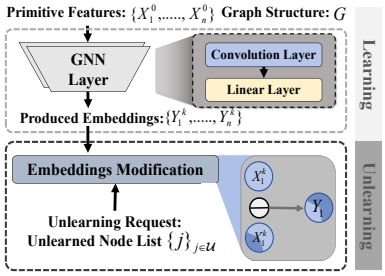


Figure 2: Proposed unlearning diagram.

We study the latter setting for node unlearning in GNNs. This setting arises, for example, when deleting users, items from graph-based recommender systems, or documents and entities from graph-structured retrieval systems. The problem is particularly hard for graph data, as feature entanglement and parameter-efficiency challenges make it difficult to produce representations close to exact.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To be specific, the *first* challenge of graph unlearning is the complicated entanglement of node features woven in the GNN. As shown in Fig. 1, node embeddings are produced by aggregation over neighborhoods, where representations are intertwined across layers. This coupling makes it difficult to faithfully identify the true counterparts of the removal requests. Among *edge unlearning*, *feature unlearning*, and *node unlearning*, this challenge is most severe for the last because removing a node simultaneously removes both its features and adjacent edges, yielding the most complicated feature scope. Many SISA-based (Sharded, Isolated, Sliced, and Aggregated) methods (Bourtoule et al., 2021; Chen et al., 2022a;b) had to retain a partial retraining component as a compromise, which is also the dominant computational cost. Others face their own bottlenecks: influence-function-based methods, such as GIF (Wu et al., 2023a), require expensive inverse-Hessian estimation and suffer accumulated approximation errors under larger perturbations; learning-based methods, such as MEGU (Li et al., 2024), depend on high-impact node selection from the original model predictions, so prediction bias can propagate to unlearning errors; projector-based methods (Cong & Mahdavi, 2023) provide guarantees mainly for linear GNNs, while nonlinear models require additional extensions.

The *second* challenge is the parameter efficiency of unlearning under multiple requests. In practical deployments, requests can arrive continuously, yet many existing methods still formulate unlearning as full-parameter re-estimation. Only a few methods can operate with substantially fewer parameter updates. SISA-based methods (Bourtoule et al., 2021; Chen et al., 2022a;b) propose to conduct a reasonable division of graph data into disjoint shards and retrain its sub-models. Thus, only the parameters of the involved sub-models need to be updated, avoiding full-parameter re-estimation. However, division incurs additional cost, and the destruction of the graph structure leads to performance degradation. Learning-based method GNNdelete (Cheng et al., 2023) updates only deletion operators while freezing other parameters, but its overhead grows linearly with GNN depth. Additionally, it could not be easily generalized.

We propose a method that resolves both challenges without any retraining throughout the unlearning process. For the *first* challenge, we disassemble complicated entanglement into general node-wise interactions and learn them via *information-conserving reconstruction*. With the deepening of GNN layers, embedding is reduced in dimension and compressed in information. We further derive an individual *linear inverse* constraint for each interaction term, which rectifies raw estimation with verifiably restricted L_2-norm error.

For the *second* challenge, instead of altering full GNN parameters, we attach a lightweight unlearning module after the whole pretrained GNN (Fig. 2) to nullify the impact of the unlearned nodes $\mathcal{U}$. Different from layer-wise designs, our module is appended once at the model output and introduces only about 0.5% extra FLOPs relative to the original model. This lightweight design enables efficient handling of various unlearning requests on shared original embeddings while mimicking retraining. We further propose a joint treatment of various unlearning proportions by adaptively reweighting loss terms for different unlearning proportions, and perform gradient-descent unrolling for downstream tasks with a local search loss.

Highlights of our contribution are: **(1)** The proposed method decouples the learning and unlearning by directly modifying shared original embeddings to satisfy diverse unlearning requirements. This approach eliminates the necessity for full-parameter re-estimation and partial retaining, offering a more efficient and conceptually coherent solution. **(2)** The reconstruction process is model-agnostic, with corresponding theoretical support; moreover, the range-null space decomposition provides a tighter bound with a smaller L_2 rectification error. **(3)** Extensive experiments on multiple real-world graph datasets and four state-of-the-art GNN models have illustrated the superior unlearning efficiency of acceleration of $40\times$ to $88\times$ compared with retraining, up to $12.5\times$ compared with baselines. Its model utility is closer to retraining. Its efficacy and privacy could be comprehensively verified.

2 Related Work

2.1 Machine Unlearning

Machine unlearning aims to remove the influence of a subset of the training data from the trained model for privacy, security, or utility concerns. Ever since Cao & Yang (2015) first introduced the concept, several methods have been proposed to address the unlearning tasks, which can be classified into exact (Ginart et al., 2019; Karasuyama & Takeuchi, 2010; Bourtoule et al., 2021) or approximate

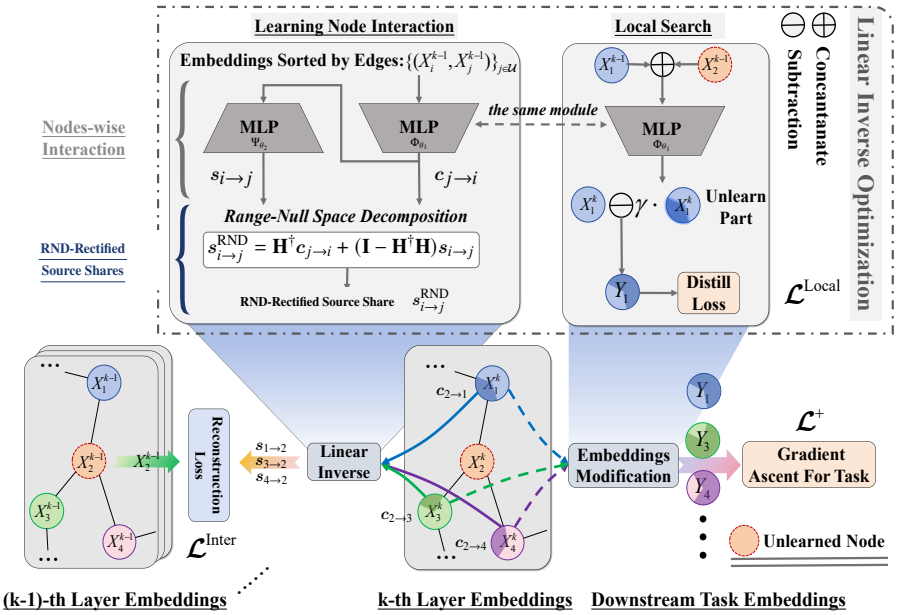


Figure 3: Overview of RTR. Reconstruction learns initial node-interaction estimates from fused hidden representations, after which range–null-space decomposition rectifies them. The refined interactions modify retained-node embeddings; $\mathcal{L}^{+}$ adapts to the downstream task.

unlearning (Koh & Liang, 2017; Guo et al., 2020; Izzo et al., 2021). The former methods typically involve re-training so that the unlearned model is identical to the one trained without the unlearned samples. Such methods include SISA (Bourtoule et al., 2021) that partitions the data, trains a set of constituent submodels, and then aggregates. By model fine-tuning, the approximate unlearning is more efficient than exact. For example, Guo et al. (2020) proposes removing the impact of the unlearned sample from the model by using the influence function (Koh & Liang, 2017). Unrolling SGD (Thudi et al., 2022) introduces a regularizer to reduce the 'verification error', which is an approximation to the distance between the unlearned model and a retrained-from-scratch model. Langevin unlearning (Chien et al., 2024) leverages the Langevin dynamic for unlearning.

2.2 Graph Unlearning

Graph unlearning is tailored for GNNs, which could be divided into (1) The Shards-based method: GraphEraser (Chen et al., 2022b) and GUIDE (Wang et al., 2023) extend the shards-based idea to graph-structured data, which offers partition methods to preserve the structural information and also designs a weighted aggregation for inference. Moreover, GraphRevoker (Zhang, 2024) utilized a property-aware sharding method and contrastive sub-model aggregation for efficient partial retraining and inference. They rely too much on a reasonable division of graph data into disjoint shards and these sub-models' costly retraining. Its inference cost is also higher as it requires results aggregation from all sub-models; (2) The IF-based method: CGU (Chien et al.), GIF (Wu et al., 2023a), CEU (Wu et al., 2023b), IDEA (Dong et al., 2024), and GST (Pan et al., 2023) leverage rigorous mathematical formulations to quantify the impact of data removal on model, allowing for efficient model updates under the Lipschitz continuous condition and loss convergence condition. These methods work under the unlearning requests that minorly change the graph structure, like 10% of graph edges. Its performance drops for node unlearning; (3) Learning-based method: GNNDelete (Cheng et al., 2023) bounding edge prediction through a deletion operator under Granger causality, and MEGU (Li et al., 2024) achieved effective and general graph unlearning through a mutual evolution design. However, these methods are limited to settings where certain assumptions hold.; (4) Others: Projector (Cong & Mahdavi, 2023) provides closed-form solutions with theoretical guarantees but is more case-specific in linear GNN.

3 RECONSTRUCT: LEARNING AGGREGATED INTERACTIONS

We turn the problem of graph unlearning into a post-hoc embedding modification: removing the influence of unlearned nodes from retained-node representations so that the resulting embeddings approximate those obtained by retraining from scratch. To achieve this, we first characterize node-level update and representation dynamics under a unified GNN view.

3.1 UNIFIED GNN MODELING

Update Dynamics. Following the forward process of GNN, the embedding $\boldsymbol{h}_{e_i}^k$ of node e_i in the k-th layer is obtained by the layer $k-1$ node embeddings as:

$$\boldsymbol{a}_{e_i}^k = \mathrm{AGGREGATE}^k\left(\left\{\boldsymbol{h}_{e_j}^{k-1} : e_j \in \mathcal{N}_{e_i}\right\}\right), \quad \boldsymbol{h}_{e_i}^k = \mathrm{COMBINE}^k\left(\boldsymbol{h}_{e_i}^{k-1}, \boldsymbol{a}_{e_i}^k\right). \tag{1}$$

This formulation covers most GNN frameworks, because $\mathrm{AGGREGATE}(\cdot)$ integrates neighborhood information, whereas $\mathrm{COMBINE}(\cdot)$ accounts for self-information and dimension reshape; the latter further subsumes a wide range of information fusion mechanisms, including residual connections.

Unlearned nodes' effect first manifests in $\mathrm{AGGREGATE}(\cdot)$ results of their adjacent nodes, then propagates to $\mathrm{COMBINE}(\cdot)$, and ultimately results in representation shifts across the retained nodes.

Representation Dynamics. We decompose the GNN architecture into three components: an input layer, one or more hidden layers, and an output layer. Accordingly, there are three types of representations: initial features $\mathcal{X}$, k-th hidden representations $\{\mathcal{H}^i\}_{i=1}^k$, and final representations $\mathcal{F}$.

Each transformation along the arrows of $\mathcal{X} \to \{\mathcal{H}^i\}_{i=1}^k \to \mathcal{F}$ corresponds to a dimensionality-reduction (rank-deficient mapping) step in representation space. Meanwhile, the hidden embedding size is often kept constant across all hidden layers (Luo et al., 2024). The optimal number of hidden layers l is usually less than 3, because excessive aggregation of multi-hop inter-class interaction makes representations less distinguishable, thereby degrading task performance (Chen et al., 2020).

3.2 INFORMATION-CONSERVING RECONSTRUCTION FOR AGGREGATED INTERACTION

We fuse hidden-layer representations as $\mathcal{H}^* = g(\{\mathcal{H}^i\}_{i=1}^k)$, to learn the interaction between $\mathcal{H}^*$ and $\mathcal{F}$, thereby circumventing explicit modeling of complex inter-layer interactions among hidden layers. Then, from an information-conserving perspective, node interaction is learned through an operation that is practically equivalent to reconstruction by $\mathcal{L}^{\text{Inter}}$ to ensure the completeness of unlearning.

Modeling Node-wise Interaction. For brevity, we use i and j to index nodes e_i and e_j. Let $e_i \in \mathcal{R}$ be a retained node and $e_j \in \mathcal{U} \cap \mathcal{N}_{e_i}$ one of its adjacent nodes to forget. Their fused hidden representations are denoted by $\boldsymbol{h}_{e_i}^*, \boldsymbol{h}_{e_j}^* \in \mathcal{H}^*$, respectively. An interaction MLP Φ_{θ_1} predicts $\boldsymbol{c}_{j\to i}$, the contribution of e_j to the final representation of e_i. We aggregate the contributions of all adjacent forgotten nodes into the representation shift $\boldsymbol{\Delta}_{e_i}$ and subtract it from the original final representation:

$$\boldsymbol{c}_{j\to i} = \Phi_{\theta_1}(\boldsymbol{h}_{e_j}^*, \boldsymbol{h}_{e_i}^*), \qquad \boldsymbol{\Delta}_{e_i} = \sum_{e_j \in \mathcal{U}\cap\mathcal{N}_{e_i}} \boldsymbol{c}_{j\to i}, \qquad \tilde{\boldsymbol{h}}_{e_i}^k = \boldsymbol{h}_{e_i}^k - \boldsymbol{\Delta}_{e_i}. \tag{2}$$

Here $\tilde{\boldsymbol{h}}_{e_i}^k \in \mathcal{F}$ is the post-unlearning representation of retained node e_i.

Embedding reconstruction loss. To ensure that each learned contribution retains sufficient information about its source node, a reconstruction MLP Ψ_{θ_2} maps $\boldsymbol{c}_{j\to i}$ back to a source share: $\boldsymbol{s}_{i\to j} = \Psi_{\theta_2}(\boldsymbol{c}_{j\to i})$. The share is further rectified as $\boldsymbol{s}_{i\to j}^{\text{RND}}$ in Sec. 4. For each forgotten node e_j, its fused hidden representation is reconstructed by aggregating the shares recovered through its retained neighbors: $\bar{\boldsymbol{h}}_{e_j}^* = \sum_{e_i \in \mathcal{R}\cap\mathcal{N}_{e_j}} \boldsymbol{s}_{i\to j}$. Guided by the intuition that better reconstruction can support more comprehensive unlearning, we align them via distillation:

$$\mathcal{L}^{\text{Inter}} = \frac{1}{|\mathcal{U}|} \sum_{e_m \in \mathcal{U}} \mathrm{KL}\big(\mathrm{Norm}[\boldsymbol{h}_{e_m}^*], \mathrm{Norm}[\bar{\boldsymbol{h}}_{e_m}^*]\big), \tag{3}$$

where $\mathrm{Norm}[\cdot]$ indicates normalization to enable information distillation, $\mathrm{KL}(\cdot)$ as KL divergence. In practical implementation, we get the the fused hidden-layer representations $\mathcal{H}^*$ via averaging. This design is discussed in greater depth in Appendix A.

4 RECTIFY: RANGE–NULL-SPACE DECOMPOSITION FOR PER-INTERACTION

We next constrain each individual interaction, rather than only their aggregation. A *linear inverse* relation makes each reconstructed share $\boldsymbol{s}_{i\to j}$ an approximate preimage of its interaction $\boldsymbol{c}_{j\to i}$.

Establishing the linear inverse constraint. Since $\boldsymbol{c}_{j\to i}$ lies in a lower-dimensional representation space than $\boldsymbol{s}_{i\to j}$, estimating $\boldsymbol{s}_{i\to j}$ from $\boldsymbol{c}_{j\to i}$ is intrinsically *underdetermined*. We employ range–null-space decomposition to rectify the preliminary estimate: the vector is projected onto the null space ($\mathbf{I}-\mathbf{H}^{\dagger}\mathbf{H}$) and the range space ($\mathbf{H}^{\dagger}$), which are combined into a constrained solution.

In the process of GNNs as Eq. 1, multiple features are linearly aggregated, then fused. The linear inverse constraint between $\boldsymbol{c}_{j\to i}$ and $\boldsymbol{s}_{i\to j}$ is written with a rank-deficient operator $\mathbf{H}$ and an error term z for the nonlinear residual: $\boldsymbol{c}_{j\to i}-z=\mathbf{H}\cdot\boldsymbol{s}_{i\to j}$, where $\mathbf{H}$ aligns the last hidden embeddings with the output, which is indicated by GNN weight $\mathbf{W}^{k}$. Consistency holds in z-dropped model.

$$\boldsymbol{s}_{i\to j}^{\mathrm{RND}}=\mathbf{H}^{\dagger}\boldsymbol{c}_{j\to i}+(\mathbf{I}-\mathbf{H}^{\dagger}\mathbf{H})\boldsymbol{s}_{i\to j} \tag{4}$$

where $\boldsymbol{s}_{i\to j}$ is the preliminary estimation and $\mathbf{H}^{\dagger}$ is a generalized inverse of $\mathbf{H}$ satisfying $\mathbf{H}\mathbf{H}^{\dagger}\mathbf{H}\equiv\mathbf{H}$. Lemma 1 verifies **consistency**:

Lemma 1. ***(RND Consistency)*** *Rectified $\boldsymbol{s}_{i\to j}^{\mathrm{RND}}$ satisfies the same linear inverse constraint with $\boldsymbol{s}_{i\to j}$*

$$\begin{aligned}\mathbf{H}\cdot\boldsymbol{s}_{i\to j}^{\mathrm{RND}}&=\mathbf{H}\mathbf{H}^{\dagger}\mathbf{H}\,\boldsymbol{s}_{i\to j}+(\mathbf{H}-\mathbf{H}\mathbf{H}^{\dagger}\mathbf{H})\,\boldsymbol{s}_{i\to j}\\&=\mathbf{H}\cdot\boldsymbol{s}_{i\to j}=\boldsymbol{c}_{j\to i}.\end{aligned}$$

Hence we replace $\boldsymbol{s}_{i\to j}$ in Eq. 3 with $\boldsymbol{s}_{i\to j}^{\mathrm{RND}}$ when optimizing $\mathcal{L}^{\mathrm{Inter}}$. RND is applied to the reconstructed shares during training; inference still subtracts the unrectified interactions $\boldsymbol{c}_{j\to i}=\Phi_{\theta_1}(\cdot)$ as in Eq. 2. Theorem 1 states the corresponding non-increasing error property:

Theorem 1. *Assume that the ideal share $\boldsymbol{s}_{i\to j}^{*}$ satisfies $\boldsymbol{c}_{j\to i}=\mathbf{H}\boldsymbol{s}_{i\to j}^{*}$. Define $e_0=\|\boldsymbol{s}_{i\to j}^{\mathrm{RND}}-\boldsymbol{s}_{i\to j}^{*}\|_2^2$ and $e_1=\|\boldsymbol{s}_{i\to j}-\boldsymbol{s}_{i\to j}^{*}\|_2^2$. Then $e_0\le e_1$, with*

$$e_0=\|(\mathbf{I}-\mathbf{H}^{\dagger}\mathbf{H})[\boldsymbol{s}_{i\to j}-\boldsymbol{s}_{i\to j}^{*}]\|^2.$$

Thus, the RND step does not increase the L_2-norm error; the proof is given in Appendix B.

5 UNLEARNING IN DOWNSTREAM TASK

The reconstruction and rectification remove the unlearned nodes interactions at the representation. An unlearning request should also be reflected in downstream task while preserving utility for retained nodes. Under varying unlearning ratios, we reweight distribution-shift and interaction terms by $\mathcal{L}$.

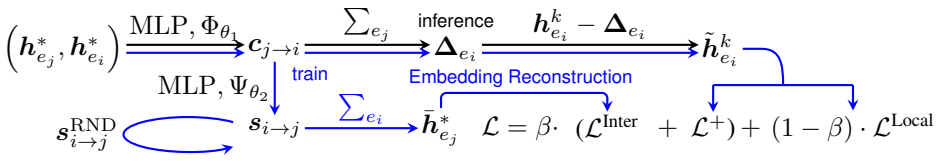
Unlearning in a task. To remove task-specific factors, an ascent loss $\mathcal{L}^{+}$ is adopted, e.g., maximizing classification loss on the unlearned nodes embeddings $\mathcal{L}^{+}=-\sum_{j\in\mathcal{U}}\mathrm{CE}(\mathrm{softmax}(\hat{\boldsymbol{h}}_{e_j}^{k}),y_{e_j})$.

A joint treatment of various unlearning proportions. Since the target distribution could be affected by *unlearning proportion*, which determines the retained *graph structure* and node interactions. We designed a generalized optimization target for better scalability.

For $\mathcal{L}^{\mathrm{Local}}$, we sample retained high-degree nodes p and distill local structure from teacher embeddings obtained by running the frozen original GNN once on the pruned graph after removing $\mathcal{U}$. Denote these teacher targets by $\hat{\boldsymbol{h}}_{e_p}^{k}$. Given a fixed unlearning ratio β, the final loss is unified as $\mathcal{L}$.

$$\mathcal{L}^{\mathrm{Local}}=\sum_{p}\mathrm{KL}(\mathrm{Norm}[\hat{\boldsymbol{h}}_{e_p}^{k}],\mathrm{Norm}[\tilde{\boldsymbol{h}}_{e_p}^{k}]),\quad \mathcal{L}=\beta\cdot(\mathcal{L}^{+}+\mathcal{L}^{\mathrm{Inter}})+(1-\beta)\cdot\mathcal{L}^{\mathrm{Local}} \tag{5}$$

The two-branch process of training and inference can therefore be summarized as



The weight β increases with the unlearning ratio and simultaneously scales the interaction loss $\mathcal{L}^{\text{Inter}}$ in Eq. 5: as more nodes are removed, local search on the original graph becomes less reliable and the interaction term is emphasized. Algorithm 1 summarizes the two-phase procedure.

6 EXPERIMENTS

We conduct experiments on five public graph datasets with different sizes, including Cora[1] (Kipf & Welling, 2017), Citeseer[2] (Kipf & Welling, 2017), CS [3] (Zhang et al., 2021) and ogbn-arxiv & -products Hu et al. (2020) in Tab. 4. In the main paper we first report a comprehensive comparison of existing methods in Tab. 1, and then present the four metrics results below from Sec. 6.2 to Sec. 6.4 compared with GIF and MEGU. These two methods are relatively efficient, and they are the most related and most comparable typical counterparts of our approach. We further ablate range–null-space decomposition and the deletion ratio in Sec. 6.5, and visualize the embedding-distribution similarity in Sec. 6.6. The overall experimental setup is given in Appendix C; additional experiments and supporting material appear in Appendices D.

Algorithm 1: Reconstruct Then Rectify (RTR)

Require: Frozen GNN, $G = (\mathcal{V}, \mathcal{E})$, request $\mathcal{U}$, ratio β, final embeddings $\boldsymbol{h}^k$, and fused embeddings $\boldsymbol{h}^*$.
Ensure: Retained-node embeddings $\tilde{\boldsymbol{h}}^k_{e_i}$.

$\mathbf{H} \leftarrow \mathbf{W}^k$.
while not converged **do**
  for each unlearned node $m \in \mathcal{U}$ **do**
    for each retained $i \in \mathcal{N}_m \setminus \mathcal{U}$ **do**
      Obtain $\boldsymbol{c}_{m\to i}$ and $\boldsymbol{s}_{i\to m}$ by Ψ_{θ_2} and Φ_{θ_1}.
      Rectify $\boldsymbol{s}^{\text{RND}}_{i\to m}$ by Eq. 4.
    end for
    $\tilde{\boldsymbol{h}}^*_{e_m} \leftarrow \sum_{i\in\mathcal{N}_m\setminus\mathcal{U}} \boldsymbol{s}^{\text{RND}}_{i\to m}$
  end for
  Compute $\tilde{\boldsymbol{h}}^k$ by Eq. 2 and $\mathcal{L}$ by Eq. 5.
  Update θ_1 and θ_2 by backpropagation.
end while
for each retained node $i \in \mathcal{V} \setminus \mathcal{U}$ **do**
  $\tilde{\boldsymbol{h}}^k_{e_i} \leftarrow \boldsymbol{h}^k_{e_i} - \sum_{j\in\mathcal{U}\cap\mathcal{N}_{e_i}} \boldsymbol{c}_{j\to i}$
end for

Metrics. We comprehensively evaluate the performance of our method in terms of the following: (i) ***Unlearning Efficiency***: We record the running time (RT) to reflect the unlearning efficiency across different unlearning algorithms for efficiency comparison; (ii) ***Model Utility***: We use F1-score — the harmonic average of precision and recall — to measure the utility of unlearned models. This performance should be consistent with training from scratch, indicating an equivalent utility; (iii) ***Unlearning Efficacy***: We adopt an indirect evaluation strategy that measures model utility in the task of removing adversarial or poisoned data; (iv) ***Unlearning Privacy***: We employ the membership inference attack designed for unlearning to evaluate the privacy leakage of the unlearned samples by AUC values. Ideally, the attacker infers the unlearned samples close to random guessing.

Table 1: Comprehensive SGC node-unlearning comparison under the OpenGU 80/20 protocol with a 10% deletion request (mean F1 ± std and RT in seconds).

Strategy	Cora		Citeseer		CS	
	F1-score	RT	F1-score	RT	F1-score	RT
Retrain	0.8357±0.0161	5.96	0.6756±0.0132	7.69	0.8942±0.0147	102.46
GraphEraser	0.8114±0.0100	105.51	0.7357±0.0125	135.65	0.9124±0.0008	1323.26
GUIDE	0.7389±0.0218	44.80	0.6350±0.0071	57.60	0.8696±0.0015	2128.36
GraphRevoker	0.8109±0.0109	30.54	0.7345±0.0061	39.27	0.9126±0.0010	793.96
GIF	0.8175±0.0109	0.15	0.6258±0.0067	0.18	0.9187±0.0022	11.12
CGU	0.8637±0.0078	82.85	0.7562±0.0041	106.53	OOT	OOT
ScaleGUN	0.7882±0.0014	1.09	0.7342±0.0013	1.31	0.9144±0.0009	55.60
IDEA	0.8771±0.0025	**0.14**	0.6366±0.0049	0.40	0.9169±0.0008	33.35
GNNDelete	0.7478±0.0549	0.95	0.6426±0.0382	1.14	0.7626±0.0273	122.32
MEGU	**0.8268±0.0156**	0.95	0.6360±0.0111	1.15	**0.8947±0.0022**	123.35
Projector	0.8679±0.0237	5.86	0.7700±0.0065	7.03	0.8840±0.0063	88.97
RTR	0.8518±0.0143	**0.14**	**0.6645±0.0068**	**0.17**	0.9013±0.0089	**0.96**

[1] https://paperswithcode.com/dataset/cora
[2] https://paperswithcode.com/dataset/citeseer
[3] https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.Coauthor.html

6.1 Comprehensive Comparison

Tab. 1 gives a comprehensive comparison of existing methods. RTR ties IDEA for the fastest RT on Cora and is fastest on Citeseer and CS; its F1 is closest to Retrain on Citeseer and second closest on Cora and CS. GAT, GCN, and GIN results are reported in Appendix D.1.

6.2 Unlearning Efficiency and Model Utility

Both the GIF method and our method are far more efficient than Retraining as shown in Tab. 2. Obviously, retraining consumes exorbitant time and computational expenses. With the increase in graph scale, retraining method suffers from $O(n^2)$ complexity with n being the number of nodes.

Table 2: Comparison of F1 scores and running time (RT) for different graph unlearning methods with 10% nodes deleted from the original graph. The bold indicates the best result.

Model		Dataset					
		Cora		Citeseer		CS	
Backbone	Strategy	F1-score (%)	RT (second)	F1-score (%)	RT (second)	F1-score (%)	RT (second)
GCN	Retrain	80.81±1.11	6.33	69.73±1.37	8.53	90.39±0.32	108.25
	GIF	74.51±0.67	0.20	58.76±0.85	0.43	82.89±0.43	11.64
	MEGU	77.32±1.63	0.19	63.67±1.28	**0.22**	86.08±0.47	9.19
	Ours	**82.73±1.00**	**0.16**	**67.75±0.90**	0.47	**90.91±0.40**	**1.23**
GAT	Retrain	87.45±0.96	15.72	77.05±0.49	35.47	91.24±0.21	128.37
	GIF	81.74±0.75	1.39	75.58±0.46	1.49	89.21±0.01	6.74
	MEGU	85.81±1.03	0.94	79.05±0.79	1.23	88.43±0.58	4.51
	Ours	**86.86±1.78**	**0.76**	**78.13±0.74**	**0.94**	**91.86±0.65**	**1.23**
SGC	Retrain	80.29±1.80	6.63	69.49±1.49	8.29	89.29±1.77	110.48
	GIF	74.56±0.71	0.20	58.67±0.88	**0.22**	83.34±0.35	12.04
	Ours	**81.84±0.43**	**0.14**	**65.89±0.56**	0.40	**90.26±0.49**	**0.96**
GIN	Retrain	81.84±1.02	8.48	73.75±1.65	25.97	88.71±0.31	115.34
	GIF	75.17±1.99	0.79	68.49±1.73	1.67	87.68±1.19	2.33
	Ours	**82.29±0.73**	**0.30**	**74.65±0.56**	**0.53**	**89.62±0.82**	**2.02**

The efficiency of both our method and GIF is comparable on Cora and Citeseer. On larger graphs, GIF requires more iterations for convergence due to a more complicated Hessian matrix, thus leading to the performance gap in CS. In contrast, our method merely includes a low-cost training in embedding reconstruction and a 'subtraction' in the inference, and hence the computational cost increases almost linearly with the number of nodes. The efficiency of MEGU is between our method and GIF. Compared with retraining, our method achieves almost an equivalent F1 score. The GIF method is less effective since the local structure surrounding the unlearned nodes is entirely changed. Both our method and GIF are well adapted to multiple backbones of GNNs. The MEGU's F1 scores of GCN and GAT are also included, which are better than GIF in the mass, but still inferior to our method.

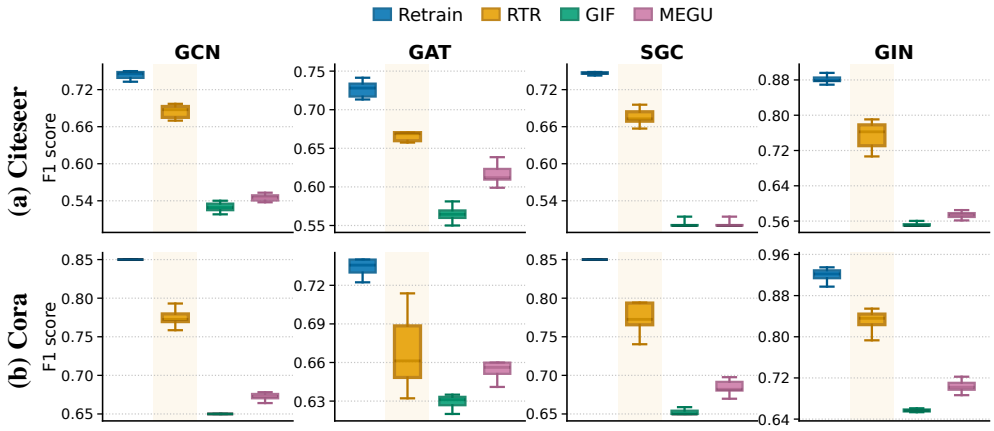


Figure 4: Unlearning efficacy comparison of the our method with GIF and MEGU in box diagram.

6.3 Unlearning Efficacy

We test the unlearning efficacy by removing adversarial or poisoned nodes from the graph and evaluating the performance improvement. To create adversarial data, we randomly shift the labels of 30% training nodes to their groundtruth label plus 1, e.g. $y_{e_i} \rightarrow (y_{e_i} + 1) \bmod n_c$, where n_c denotes the number of categories. As shown in Fig. 4, we find that our proposed method can effectively improve the performance of the original model output representation on downstream tasks for different graph models. In the figure, models denoted as 'Noisy' are trained on all data, including the adversarial data, while those in 'Retrain' are trained from scratch with the adversarial data removed. Models in the 'Unlearn' category apply our method to remove the adversarial data.

Table 3: Attack AUC of membership inference against different unlearning methods.

Models	Cora				Citeseer				CS			
	GCN	GAT	SGC	GIN	GCN	GAT	SGC	GIN	GCN	GAT	SGC	GIN
GIF	0.5952	0.5428	0.5834	0.6010	0.6052	0.5381	0.5996	0.5657	0.5744	0.5977	0.5591	0.5930
MEGU	0.4856	**0.5008**	0.5081	0.4768	0.4532	0.4483	0.4735	0.4750	0.5155	0.5190	0.5186	0.4964
CGU	0.5019	0.5084	**0.5015**	**0.5026**	0.4637	0.4366	0.4856	0.5216	0.5048	0.5124	0.5208	0.4962
GraphEraser	0.7601	0.5358	0.7684	0.4694	0.7991	0.6574	0.7521	0.5950	0.4982	0.5192	0.6582	0.5736
IDEA	0.5371	0.4754	0.5328	0.5344	0.5672	0.5279	0.5659	0.5075	0.5021	0.4827	0.4970	0.5193
Ours	**0.5014**	0.4857	0.4825	0.5129	**0.5010**	**0.4916**	**0.4963**	**0.4987**	**0.4988**	**0.5015**	**0.4994**	**0.5002**

6.4 Unlearning Privacy

Prior work has shown that unlearning may incur additional privacy leakage (Chen et al., 2021), as attackers can exploit embedding shifts between pre- and post-unlearning models. In particular, nodes exhibiting unusually larger embedding deviations are more vulnerable to MIA.

Note that our method modifies original embeddings, which may expose unlearned nodes through the distribution of embedding shifts. Following Chen et al. (2021), we conduct MIA by collecting node embeddings before and after unlearning 10% of nodes and computing their L_2 distances. We then vary distance thresholds to distinguish unlearned nodes and report the resulting attack AUC.

As shown in Tab. 3, the proposed method is resilient to MIA. Under various backbones and datasets, the AUC values are close to 0.5 consistently. It shows that the attack is close to a random guess. Hence, the unlearned nodes are indistinguishable from the retained. In fact, even in the retraining method, AUCs deviate from 0.5 by some small margins, showing that even the ideal unlearning would leak the unlearned sample to some extent. More detailed analysis of privacy baseline comparisons are illustrated in Tab. 3. Our method demonstrates strong performance among a wide range of methods.

6.5 Ablation Studies

Range-null space decomposition (RND). We show the effect of RND by removing the component from the framework at an unlearning ratio of 30%. The results on Cora and Citeseer illustrate that our method with RND achieves a closer F1 score with retraining for the multi-label node classification task. The F1 scores reduce when RND is removed. This shows that the linear inverse constraints lead to less loss of useful information in the unlearning process.

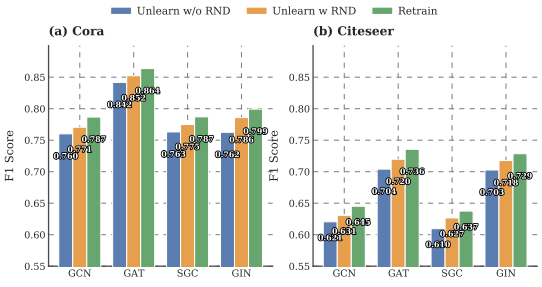


Figure 6: Ablation study of RND on Cora and Citeseer.

Varying unlearning ratios. We conduct experiments on Cora and Citeseer under unlearning ratios of 10%, 20%, 30%, and 40%, with results provided in Fig. 5. It is noted that the model utility of retraining decreases slowly and evenly with the increase of the unlearned proportion. The F1 score of GIF drops rapidly with the growing unlearning ratio. Our method is consistent with that of the retraining, both in mean and

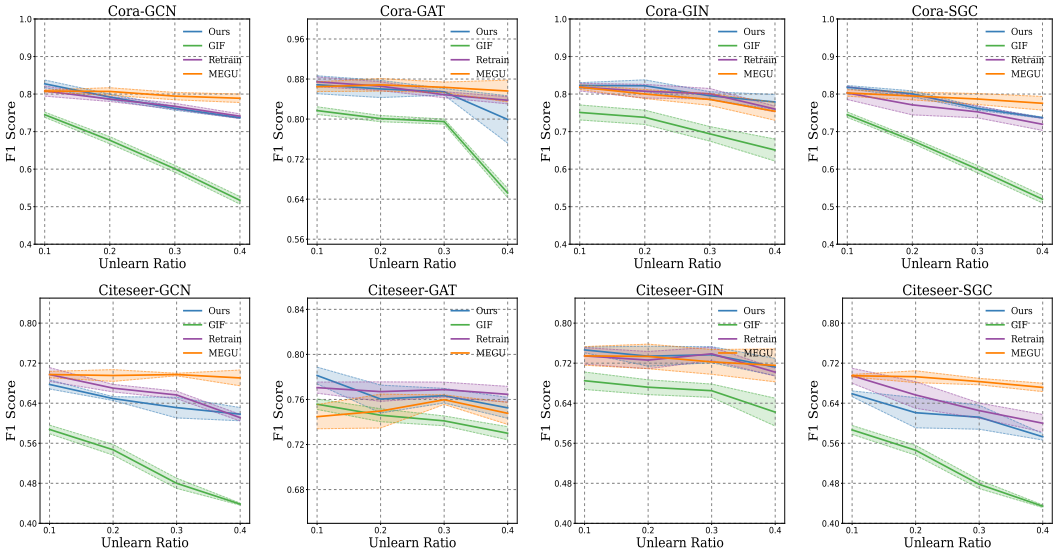


Figure 5: Varying unlearn ratios comparison with GIF and MEGU. Dotted line indicates variances.

variance, and thus it can be concluded that our method is more robust than GIF at different ratios. And these experiments were actually implemented the same way as *sequential unlearning*, demonstrating our method's capability for multiple sequential requests.

6.6 Unlearning Visualization

In each domain(training from all, retraining, and our unlearning method), we independently derive the PDFs with bandwidth $h = 1$ as illustrated in Fig. 7. We verify that the embedding distribution of retraining overlaps with that of training from all but significantly differs in density, as shown in Fig. 7(a). Our method produces the embedding distributions consistent with retraining as Fig. 7(b).

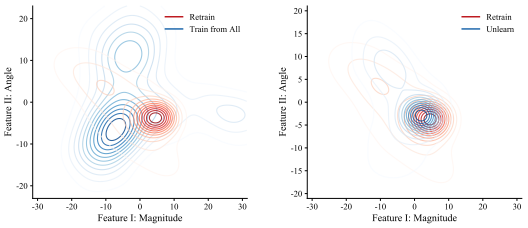


Figure 7: Embedding distribution when unlearning 30% nodes under the SGC backbone.

We employ 2D kernel density estimation to calculate the probability density function (PDF) f_V for embedding set $V = (\text{Mag}_i, \text{Ang}_i)_{i=1}^n$. We use a Gaussian kernel for PDF estimation with $\alpha = 2\pi n h^2$, where $d_{\text{Mag}_i} = (\text{Mag} - \text{Mag}_i)^2$, $d_{\text{Ang}_i} = (\text{Ang} - \text{Ang}_i)^2$.

$$f_V(\text{Mag}, \text{Ang}) = \frac{1}{\alpha} \sum_{i=1}^{n} \exp[-\frac{1}{2h^2}(d_{\text{Mag}_i} - d_{\text{Ang}_i})]$$

7 Conclusion

We propose an efficient yet effective graph unlearning method in this paper. Our core idea is to reverse the aggregation process in GNN training by modeling the interaction between the unlearned nodes and their neighbors. Such an interaction is learned through fused hidden-layer embeddings reconstruction. The range-null space decomposition method is adopted to rectify the raw estimation of the interaction. We theoretically show that our estimation has an absolutely lower error with consistency and realness. Experimental results on multiple representative datasets demonstrate the effectiveness and efficiency of our proposed approach, compared to the state-of-the-art.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability and LaTeX formatting. We have not used generative AI tools to generate datasets, experimental numbers, or figures. We reviewed the AI-assisted text and checked the mathematical claims against the appendix. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies efficient approximate node unlearning on public graph benchmarks. It does not involve human subjects, private user data, or a new dataset release. We have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The RTR training and inference procedure is given in Algorithm 1. Datasets, the two evaluation protocols, implementation details, and hyperparameters are collected in Appendix C. Anonymous source code will be submitted as supplementary material.

REFERENCES

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM web conference 2022*, pp. 2768–2777, 2022a.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3438–3445, 2020.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pp. 896–911, 2021.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pp. 499–513, 2022b.

Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. Gnndelete: A general strategy for unlearning in graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.

Eli Chien, Haoyu Peter Wang, Ziang Chen, and Pan Li. Langevin unlearning: A new perspective of noisy gradient descent for machine unlearning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Weilin Cong and Mehrdad Mahdavi. Efficiently forgetting what you have learned in graph representation learning via projection. In *International Conference on Artificial Intelligence and Statistics*, pp. 6674–6703. PMLR, 2023.

Yushun Dong, Binchi Zhang, Zhenyu Lei, Na Zou, and Jundong Li. Idea: A flexible framework of certified unlearning for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 621–630, 2024.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3832–3842, 2020.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2008–2016. PMLR, 2021.

Masayuki Karasuyama and Ichiro Takeuchi. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks*, 21(7):1048–1059, 2010.

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

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Chanhee Kwak, Junyeong Lee, Kyuhong Park, and Heeseok Lee. Let machines unlearn–machine unlearning and the right to be forgotten. 2017.

Xunkai Li, Yulin Zhao, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. Towards effective and general graph unlearning via mutual evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13682–13690, 2024.

Yi Li, Shichao Zhang, Guixian Zhang, and Debo Cheng. Community-centric graph unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18548–18556, 2025.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. *Advances in Neural Information Processing Systems*, 37:97650–97669, 2024.

Chao Pan, Eli Chien, and Olgica Milenkovic. Unlearning graph classifiers with limited data resources. In *Proceedings of the ACM Web Conference 2023*, pp. 716–726, 2023.

Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. Recorrupted-to-recorrupted: Unsupervised deep learning for image denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2043–2052, 2021.

Stuart L Pardau. The california consumer privacy act: Towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.

Protection Regulation. General data protection regulation. *Intouch*, 25:1–5, 2018.

Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, and Nina Taft. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, pp. 1–14, 2009.

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pp. 303–319. IEEE, 2022.

Cheng-Long Wang, Mengdi Huai, and Di Wang. Inductive graph unlearning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 3205–3222, 2023.

Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, Min Lin, and Tat-Seng Chua. Causal representation learning for out-of-distribution recommendation. In *Proceedings of the ACM Web Conference 2022*, pp. 3562–3571, 2022.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, pp. 651–661, 2023a.

Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. Certified edge unlearning for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2606–2617, 2023b.

Zhe-Rui Yang, Jindong Han, Chang-Dong Wang, and Hao Liu. Erase then rectify: A training-free parameter editing approach for cost-effective graph unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13044–13051, 2025.

Jiahao Zhang. Graph unlearning with efficient partial retraining. In *Companion Proceedings of the ACM Web Conference 2024*, pp. 1218–1221, 2024.

Jiayi Zhang, Zhi Cui, Xiaoqiang Xia, Yalong Guo, Yanran Li, Chen Wei, and Jianwei Cui. Writing polishment with simile: Task, dataset and a neural approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14383–14392, 2021.

Jie Zhang, Chen Dongdong, Qidong Huang, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu. Poison ink: Robust and invisible backdoor attack. *IEEE Transactions on Image Processing*, 31:5691–5705, 2022.

Appendix

This appendix first motivates the reconstruction objective and proves the RND bound, then records dataset statistics, extra experiments, method extensions, related work, and limitations. Appendix A relates reconstruction to an information-theoretic motivation. Appendix B proves the RND error bound. Appendix C states datasets, protocols, and implementation. Appendix D reports additional experiments, including backbone comparisons, memory and FLOPs, large-scale graphs, extreme-ratio, and recent-baseline results. Appendix E describes low- and high-ratio adaptations. Appendix F extends the related-work discussion.

CONTENTS

A Information-Theoretic Motivation for Reconstruction . . . 13
B Proof of the RND Error Bound . . . 15
C Dataset Statistics . . . 15
C.1 Protocol consistency . . . 16
C.2 Implementation . . . 16
D Additional Experiments . . . 16
D.1 Additional backbone comparisons . . . 16
D.2 Memory and FLOPs . . . 16
D.3 Large-scale graphs . . . 18
D.4 Extreme deletion ratio . . . 18
D.5 Recent baselines . . . 19
E Method Extensions . . . 19
E.1 Low-ratio removal . . . 19
E.2 High-ratio removal . . . 20
F Extended Related Work . . . 20
F.1 Machine unlearning . . . 20
F.2 Graph unlearning . . . 21

A INFORMATION-THEORETIC MOTIVATION FOR RECONSTRUCTION

This section is a motivation for the reconstruction objective, not a certified deletion guarantee. The starting observation is that unlearning concerns an *already converged* network: if the source model is an arbitrary intermediate iterate, both the original state and the retraining target depend on the optimization trajectory, so the inverse problem of inferring $h_{\theta(\mathcal{D}\setminus\mathcal{U})}$ from $\left(h_{\theta(\mathcal{D})}, \mathcal{U}\right)$ is ill-posed: the solution is not unique.

Definition 1 (Approximate graph unlearning in representation space)**.** *Let $\mathcal{D}$ be a graph training dataset and $\ell(\theta;\mathcal{D})$ the value of the loss function of a model with parameters θ on $\mathcal{D}$. A* converged *training state is a stationary point*

$$\theta^*(\mathcal{D}) \in \left\{\theta : \nabla_\theta \ell(\theta;\mathcal{D}) = \mathbf{0}\right\}. \tag{6}$$

Write h_θ for the representation map induced by θ. Given a forget set $\mathcal{U} \subset \mathcal{D}$, approximate unlearning in representation space is the problem of inferring the converged representations of $h_{\theta^*(\mathcal{D}\setminus\mathcal{U})}$, from the observed $h_{\theta^*(\mathcal{D})}$ and unlearning set $\mathcal{U}$, without retraining on $\mathcal{D}\setminus\mathcal{U}$.

That awareness comes from the influence-function derivation later adopted by GIF, which obtains a Newton correction $\Delta\theta \approx -H_{\theta^*}^{-1}\nabla_\theta \Delta\ell$ only after the training gradient vanishes. We therefore first define the problem at a stationary representation (Definition 1). Because unlearning typically acts on an already converged network, we then show that if the reconstructed interaction contains as much task-relevant information as the original hidden representation, the learned interaction decomposition is complete enough to support unlearning. The remainder of the section develops this argument and relates it to $\mathcal{L}^{\text{Inter}}$.

Write $Z = \boldsymbol{h}^*$ for the source hidden representation, $C = \{\boldsymbol{c}_{j\to i}\}$ for the learned contributions of node j to the final representation of node i, $S = \{\boldsymbol{s}_{i\to j}\}$ for the reconstruction shares derived from these contributions, and $\bar{Z} = \bar{\boldsymbol{h}}^*$ for the reconstruction obtained by aggregating the corresponding shares. Since the ground-truth label Y provides the supervision under which the task-relevant hidden representation Z is learned, while C, S, and $\bar{Z}$ are successively derived from Z, we model their dependence by the Markov chain

$$Y \to Z \to C \to \bar{Z}.$$

Lemma 2 (Entropy risk and mutual-information equivalence)**.** *Let Z be a representation variable and Y the prediction target. For a predictor $q_\theta(y \mid z) \in \mathcal{Q}$, define the population log-loss risk*

$$\mathcal{R}_{\mathcal{Q}}(Z;\theta) = \mathbb{E}_{(z,y)\sim p(z,y)}\big[-\log q_\theta(y \mid z)\big].$$

If a converged predictor q_{θ^*} matches the conditional $p(y \mid z)$, then $\mathcal{R}_{\mathcal{Q}}(Z;\theta^*) = H(Y \mid Z)$ and $I(Z;Y) = H(Y) - \mathcal{R}_{\mathcal{Q}}(Z;\theta^*)$. Consequently, under a fixed data distribution, for any two representations Z_1, Z_2,

$$\mathcal{R}_{\mathcal{Q}}(Z_1;\theta^*) = \mathcal{R}_{\mathcal{Q}}(Z_2;\theta^*) \quad\Longleftrightarrow\quad I(Z_1;Y) = I(Z_2;Y).$$

Lemma 2 provides a bridge from the relationship between prediction risks to the corresponding relationship between mutual informations. Under the following task-sufficiency assumption for a converged network, we can therefore use this lemma to characterize the mutual-information relationship between Z and C as $I(C;Y) = I(Z;Y)$.

Assumption 1 (Task-sufficiency at convergence)**.** *At a stationary representation $Z = h_{\theta^*(\mathcal{D})}$, the predictor has largely filtered task-irrelevant variation and retained task-relevant information. We therefore assume that the contribution representation C, although rank-deficient with respect to Z, preserves the predictive risk of the converged source representation:*

$$\mathcal{R}_{\mathcal{Q}}(C;\theta^*) = \mathcal{R}_{\mathcal{Q}}(Z;\theta^*). \tag{7}$$

Under the conditions of Lemma 2, this risk equivalence directly implies

$$I(C;Y) = I(Z;Y). \tag{8}$$

Ideal unlearning does not remove all task-relevant information of unlearned nodes from the model. We have shown that $I(C;Y) = I(Z;Y)$. When faithful reconstruction is further achieved, the learned contributions can therefore be interpreted as carriers of the corresponding unlearned nodes' task-relevant information, providing the information to be targeted by the subsequent subtraction.

The node-wise reconstruction loss further provides an upper bound on the distribution-level discrepancy over the entire unlearned set. Let $p_m = \mathrm{Norm}[\boldsymbol{h}^*_{e_m}]$ and $q_m = \mathrm{Norm}[\bar{\boldsymbol{h}}^*_{e_m}]$ denote the normalized source and reconstructed representations of each forgotten node $e_m \in \mathcal{U}$, and define the corresponding empirical mixture distributions as

$$P_{\mathcal{U}} = \frac{1}{|\mathcal{U}|}\sum_{m\in\mathcal{U}} p_m, \qquad Q_{\mathcal{U}} = \frac{1}{|\mathcal{U}|}\sum_{m\in\mathcal{U}} q_m.$$

By the joint convexity of KL divergence, we have

$$D_{\mathrm{KL}}(P_{\mathcal{U}}\|Q_{\mathcal{U}}) \le \frac{1}{|\mathcal{U}|}\sum_{m\in\mathcal{U}} D_{\mathrm{KL}}(p_m\|q_m) = \mathcal{L}^{\mathrm{Inter}}. \tag{9}$$

Hence, $\mathcal{L}^{\mathrm{Inter}}$ serves as an explicit upper bound on the KL divergence between the aggregate source and reconstructed distributions of the unlearned nodes. Minimizing the node-wise reconstruction loss therefore simultaneously constrains the distribution-level reconstruction discrepancy over the entire unlearned set.

Together with the task-sufficiency result, this distributional reconstruction further supports C as carrying the task-relevant information associated with the unlearned nodes. Consequently, subtracting their corresponding contributions can be viewed as removing these identified information components from the retained representations, thereby realizing the intended forgetting operation at the representation level.

B Proof of the RND Error Bound (Theorem 1)

Proof. We show that rectifying the preliminary estimate $\boldsymbol{s}_{i\to j}$ does not increase its squared L_2 error under the assumptions of Theorem 1.

Let $\boldsymbol{s}^*_{i\to j}$ be an ideal share satisfying $\boldsymbol{c}_{j\to i} = \mathbf{H}\boldsymbol{s}^*_{i\to j}$. Because $\mathbf{H}^\dagger$ is the Moore–Penrose pseudoinverse, $(\mathbf{H}^\dagger\mathbf{H})^\top = \mathbf{H}^\dagger\mathbf{H}$. Substituting the RND update (Eq. 4) and the ideal-share constraint yields

$$\begin{aligned}\|\boldsymbol{s}^{\mathrm{RND}}_{i\to j} - \boldsymbol{s}^*_{i\to j}\|_2^2 &= \|\mathbf{H}^\dagger\boldsymbol{c}_{j\to i} + (\mathbf{I} - \mathbf{H}^\dagger\mathbf{H})\boldsymbol{s}_{i\to j} - \boldsymbol{s}^*_{i\to j}\|_2^2 \\ &= \|(\mathbf{H}^\dagger\mathbf{H} - \mathbf{I})\boldsymbol{s}^*_{i\to j} + (\mathbf{I} - \mathbf{H}^\dagger\mathbf{H})\boldsymbol{s}_{i\to j}\|_2^2 \\ &= \|(\mathbf{I} - \mathbf{H}^\dagger\mathbf{H})(\boldsymbol{s}_{i\to j} - \boldsymbol{s}^*_{i\to j})\|_2^2.\end{aligned}$$

It remains to show that $\mathbf{A} = \mathbf{I} - \mathbf{H}^\dagger\mathbf{H}$ is an orthogonal projection, so that applying it cannot increase an L_2 norm. We verify that $\mathbf{A}$ is idempotent and symmetric.

(1) ***Idempotent Matrix***. It holds that $\mathbf{AA} = \mathbf{A}$ thus $\mathbf{A}$ is *idempotent matrix*:

$$\begin{aligned}\mathbf{AA} &= (\mathbf{I} - \mathbf{H}^\dagger\mathbf{H})(\mathbf{I} - \mathbf{H}^\dagger\mathbf{H}) \\ &= \mathbf{I} - \mathbf{H}^\dagger\mathbf{H} - \mathbf{H}^\dagger\mathbf{H} + \mathbf{H}^\dagger\mathbf{H}\mathbf{H}^\dagger\mathbf{H} \\ &= \mathbf{I} - \mathbf{H}^\dagger\mathbf{H} - \mathbf{H}^\dagger\mathbf{H} + \mathbf{H}^\dagger\mathbf{H} \\ &= \mathbf{I} - \mathbf{H}^\dagger\mathbf{H} = \mathbf{A}\end{aligned}$$

(2) ***Symmetric Matrix***. It holds that $\mathbf{A} = \mathbf{A}^\top$ thus $\mathbf{A}$ is *symmetric matrix*:

$$\begin{aligned}\mathbf{A}^\top &= (\mathbf{I} - \mathbf{H}^\dagger\mathbf{H})^\top \\ &= \mathbf{I} - (\mathbf{H}^\dagger\mathbf{H})^\top \\ &= \mathbf{I} - \mathbf{H}^\dagger\mathbf{H} = \mathbf{A}\end{aligned}$$

Thus $\mathbf{A}$ is an orthogonal projection, and $\|\boldsymbol{s}^{\mathrm{RND}}_{i\to j} - \boldsymbol{s}^*_{i\to j}\|_2^2 \le \|\boldsymbol{s}_{i\to j} - \boldsymbol{s}^*_{i\to j}\|_2^2$. Equality is possible, so the result is non-increasing rather than strictly improving. The argument uses the linear-inverse assumption $\boldsymbol{c}_{j\to i} = \mathbf{H}\boldsymbol{s}^*_{i\to j}$ and the Moore–Penrose pseudoinverse; it does not claim exact equivalence to retraining. □

C Dataset Statistics

We conduct the main comparisons on Cora, Citeseer, and CS, and further evaluate RTR on the larger ogbn-arxiv and ogbn-products datasets (Hu et al., 2020). Tab. 4 lists the corresponding statistics. These datasets are the benchmark datasets for evaluating the performance of GNN models for node

Table 4: Statistics of the datasets used in this paper.

Dataset	#Type	#Nodes	#Edges	#Features	#Classes
Cora	Citation	2,708	5,429	1,433	7
Citeseer	Citation	3,327	4,732	3,703	6
CS	Coauthor	18,333	163,788	6,805	15
ogbn-arxiv	Citation	169,343	1,166,243	128	40
ogbn-products	Co-purchase	2,449,029	61,859,140	100	47

classification task. Cora and Citeseer are citation datasets, where nodes represent the publications and edges indicate citations between two publications. CS is a coauthor dataset, where nodes are authors connected by an edge if they have collaborations; the features are keywords of the paper. ogbn-arxiv & -products are substantially larger-scale datasets.

C.1 Protocol consistency

The main paper reports two explicitly separated protocols.

- **OpenGU 80/20.** Tab. 1 and Tabs. 5–7 use an 80/20 train/test split and delete 10% of the training.
- **GIF-style 90/10.** Tab. 2 uses a 90/10 split and the same 10% deletion. Protocol-matched MEGU numbers are available only for GCN and GAT in that table.

Large-scale numbers in Tab. 12 follow the protocol of Yang et al. (2025) and form a third, separate comparison. Implementation choices that are shared across protocols are stated next; metrics and baseline roles are those of Sec. 6.

C.2 Implementation

Experiments are implemented in PyTorch and run on an NVIDIA GeForce RTX 4090 with 24 GB of memory. For a node-unlearning request we randomly delete a fraction β of the training nodes together with their incident edges. Backbone learning rates are $\{0.05, 0.01, 0.05, 0.01\}$ and weight decays are $\{1\mathrm{e}{-4}, 1\mathrm{e}{-3}, 1\mathrm{e}{-4}, 1\mathrm{e}{-4}\}$ for GCN, GAT, SGC, and GIN, respectively. The constraint matrix $\mathbf{H}$ is the final weight that maps the last hidden embeddings to the output: the convolutional or linear weight for GCN, GAT, and SGC, and the trailing linear weight of the GIN convolutional block. The pseudoinverse $\mathbf{H}^{\dagger}$ is computed with NumPy. The interaction and reconstruction maps Φ_{θ_1} and Ψ_{θ_2} are independent three-layer MLPs.

D Additional Experiments

The experiments below are complementary to the two main-paper protocols. This section reports additional backbones, memory and FLOPs, large-scale graphs, extreme deletion ratios, and recent baselines. Each table is ranked only inside its own protocol.

D.1 Additional Backbone Comparisons

The main paper reports the broad SGC comparison under the OpenGU 80/20 protocol in Tab. 1. This section reports GAT, GCN, and GIN under the same protocol. Values are mean $\pm$ standard deviation over 10 runs. For F1, dark and light gray mark the non-retraining methods closest and second closest to Retrain; for RT, they mark the fastest and second-fastest methods. These tables complement, rather than pool with, the GIF-style 90/10 comparison in Tab. 2. Tab. 5 is the GAT counterpart of the main-text SGC table. RTR is closest to Retrain on Citeseer and CS, and second closest on Cora, while remaining among the fastest methods. Tab. 6 reports GCN. RTR is closest to Retrain on Cora and CS and second closest on Citeseer; it is the fastest method in every column of this table. Tab. 7 reports GIN. RTR is closest to Retrain on Cora and CS. On Citeseer its F1 exceeds Retrain, so we do not treat that column as a closeness ranking; the runtime remains among the lowest.

Table 5: GAT comparison under the OpenGU 80/20 split with a 10% node-unlearning request (mean F1 $\pm$ standard deviation and RT in seconds).

GAT	Cora		Citeseer		CS	
	F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)
Retrain	0.8745±0.0112	8.13	0.7854±0.0108	11.73	0.9425±0.0147	152.47
MEGU	0.8690±0.0143	0.48	0.7624±0.0145	0.51	0.9121±0.0132	64.32
CGU	0.8782±0.0117	0.43	0.7462±0.0126	0.71	0.9329±0.0100	41.30
GNNDelete	0.8480±0.0229	1.20	0.7310±0.0339	1.28	0.5554±0.0418	13.75
GraphEraser	0.8395±0.0101	11.60	0.7087±0.0103	11.12	0.9148±0.0141	37.69
IDEA	0.8413±0.0150	0.76	0.7312±0.0014	0.75	0.9177±0.0035	28.43
RTR	0.8695±0.0134	0.41	0.7895±0.0121	0.52	0.9400±0.0109	1.41

D.2 Memory And FLOPs

RTR adds two small MLPs to a frozen base GNN. Tab. 8 reports the resulting peak memory and FLOPs; the measured extra FLOPs stay below 0.5% in these settings. FLOPs and wall-clock runtime

Table 6: GCN comparison under the OpenGU 80/20 split with a 10% node-unlearning request (mean F1 ± standard deviation and RT in seconds).

GCN	Cora		Citeseer		CS	
	F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)
Retrain	0.8394±0.0104	5.234	0.7655±0.0109	7.654	0.9205±0.0148	99.64
MEGU	0.8727±0.0127	5.959	0.7486±0.0124	8.213	0.9046±0.0130	109.43
CGU	0.8875±0.0150	0.47	0.7450±0.0145	0.31	0.9190±0.0121	21.49
GNNDelete	0.8251±0.0316	0.86	0.7102±0.0218	0.91	0.4184±0.0351	36.73
GraphEraser	0.8376±0.0138	6.62	0.7042±0.0132	6.32	0.9010±0.0113	128.84
IDEA	0.8616±0.0011	0.35	0.6982±0.0041	0.33	0.9237±0.0042	32.43
RTR	0.8402±0.0143	0.32	0.7483±0.0100	0.28	0.9198±0.0106	1.13

Table 7: GIN comparison under the OpenGU 80/20 split with a 10% node-unlearning request (mean F1 ± standard deviation and RT in seconds).

GIN	Cora		Citeseer		CS	
	F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)
Retrain	0.8358±0.0134	9.25	0.7502±0.0144	7.52	0.8998±0.0122	130.84
MEGU	0.8007±0.0107	0.21	0.7568±0.0129	0.33	0.8416±0.0139	72.15
CGU	0.8690±0.0149	0.31	0.7492±0.0137	0.32	0.8939±0.0102	18.47
GNNDelete	0.7343±0.0320	0.82	0.6922±0.0405	0.77	0.3627±0.0471	2.20
GraphEraser	0.8542±0.0111	7.17	0.7072±0.0112	5.77	0.0425±0.0126	21.03
IDEA	0.8137±0.0046	0.42	0.6997±0.0033	3.40	0.8391±0.0110	32.04
RTR	0.8429±0.0125	0.28	0.7652±0.0115	0.22	0.9022±0.0136	1.03

need not vary proportionally, so we report both. Tab. 9 lists Cora FLOPs after unlearning and the relative overhead over each backbone. Tab. 10 reports the same comparison on Citeseer. Tab. 11 reports CS. Across the three datasets, RTR remains the lowest-overhead method in every backbone column.

Table 8: Peak memory and FLOPs of RTR. Percentages are extra FLOPs relative to the backbone.

Models	Cora		Citeseer		CS	
	MEM	FLOPs	MEM	FLOPs	MEM	FLOPs
GCN	0.73 GB	62,938,717(+0.45%)	1.02 GB	197,933,004(+0.12%)	4.76 GB	2,010,758,706(+0.31%)
GAT	0.78 GB	502,307,745(+0.06%)	1.09 GB	1,582,470,428(+0.01%)	15.94 GB	16,050,937,830(+0.04%)
SGC	0.73 GB	62,852,061(+0.45%)	1.02 GB	197,826,540(+0.12%)	5.00 GB	2,010,172,050(+0.31%)
GIN	0.69 GB	65,845,867(+0.43%)	0.90 GB	201,491,836(+0.11%)	4.12 GB	2,030,200,248(+0.31%)

Table 9: Cora: backbone FLOPs and extra FLOPs incurred by unlearning.

Models	Cora			
	GCN	GAT	SGC	GIN
Base FLOPs	6.27e7	6.56e7	6.26e7	5.02e8
MEGU	1.01e8(+6.1e1%)	5.40e8(+7.6e0%)	1.01e8(+6.1e1%)	1.04e8(+5.9e1%)
CGU	7.19e7(+1.5e1%)	5.44e8(+8.3e0%)	6.92e7(+1.1e1%)	7.48e7(+1.4e1%)
GNNDelete	7.84e7(+2.5e1%)	5.18e8(+3.1e0%)	7.83e7(+2.5e1%)	8.13e7(+2.4e1%)
GraphEraser	4.13e8(+5.6e2%)	8.53e8(+7.0e1%)	4.13e8(+5.6e2%)	4.16e8(+5.4e2%)
IDEA	5.33e9(+8.4e3%)	3.41e12(+6.8e5%)	1.01e10(+1.6e4%)	5.48e10(+8.3e4%)
RTR	**6.29e7(+4.5e-1%)**	**5.02e8(+6.0e-2%)**	**6.29e7(+4.5e-1%)**	**6.58e7(+4.3e-1%)**

Table 10: Citeseer: backbone FLOPs and extra FLOPs incurred by unlearning.

Models	Citeseer			
	GCN	GAT	SGC	GIN
Base FLOPs	1.98e8	2.01e8	1.98e8	1.58e9
MEGU	2.30e8(+1.7e1%)	1.61e9(+2.1e0%)	3.03e7(+1.7e1%)	2.34e8(+1.6e1%)
CGU	2.15e8(+8.8e0%)	1.68e9(+6.4e0%)	2.07e8(+5.0e0%)	2.19e8(+8.6e0%)
GNNDelete	2.12e8(+7.3e0%)	1.60e9(+9.2e-1%)	2.12e8(+7.3e0%)	2.16e8(+7.2e0%)
GraphEraser	9.39e8(+3.7e2%)	2.32e9(+4.7e1%)	9.38e8(+3.7e2%)	9.42e8(+3.7e2%)
IDEA	3.53e11(+1.8e5%)	2.26e13(+1.4e6%)	4.96e10(+2.5e4%)	3.56e11(+1.8e5%)
RTR	**1.98e8(+1.2e-1%)**	**1.58e9(+1.0e-2%)**	**1.98e8(+1.2e-1%)**	**2.01e8(+1.1e-1%)**

Table 11: CS: backbone FLOPs and extra FLOPs incurred by unlearning.

Models	CS			
	GCN	GAT	SGC	GIN
Base FLOPs	2.00e9	2.02e9	2.00e9	1.60e10
MEGU	3.22e9(+6.1e1%)	1.73e10(+7.6e0%)	3.22e9(+6.1e1%)	3.24e9(+6.0e1%)
CGU	2.07e9(+3.1e0%)	1.63e10(+1.3e0%)	2.06e9(+3.0e0%)	2.09e9(+3.1e0%)
GNNDelete	2.26e9(+1.3e1%)	1.63e10(+1.6e0%)	2.26e9(+1.3e1%)	2.28e9(+1.3e1%)
GraphEraser	3.20e10(+1.5e3%)	4.60e10(+1.9e2%)	3.20e10(+1.5e3%)	3.20e10(+1.5e3%)
IDEA	1.19e12(+5.9e4%)	7.63e13(+4.8e5%)	1.04e12(+5.2e4%)	1.20e12(+5.9e4%)
RTR	**2.01e9(+3.1e-1%)**	**1.61e10(+4.0e-2%)**	**2.01e9(+3.1e-1%)**	**2.03e9(+3.1e-1%)**

D.3 LARGE-SCALE GRAPHS

Table 12: Node unlearning on `ogbn-arxiv` and `ogbn-products` under the setting of Yang et al. (2025). Within each backbone–dataset block, dark and light gray mark the best and second-best non-retraining F1 or RT; Retrain is the yellow reference. This ranking is local to the table. OOM denotes out-of-memory.

Backbone	Method	ogbn-arxiv		ogbn-products	
		F1-score	RT (second)	F1-score	RT (second)
GCN	Retrain	67.5	152	73.6	519
	GIF	62.1	7	OOM	OOM
	GNNDelete	51.7	34	OOM	OOM
	MEGU	65.6	2	71.3	7
	RTR	**67.4**	**1**	**74.1**	**5**
GAT	Retrain	67.3	217.9	73.9	709
	GIF	66.1	12	OOM	OOM
	GNNDelete	49.6	44	OOM	OOM
	MEGU	58.5	3	63.7	8
	RTR	**68.1**	**2**	**73.6**	**5**

Tab. 12 reports node unlearning on `ogbn-arxiv` and `ogbn-products` under the setting of Yang et al. (2025). RTR remains close to Retrain in F1 and is the fastest method in every backbone–dataset block of this table. GIF and GNNDelete run out of memory on ogbn-products. These numbers are not ranked against Tab. 1 or Tab. 2.

D.4 EXTREME DELETION RATIO

Tab. 13 reports an 80% node-deletion setting on Cora and Citeseer. RTR keeps a large runtime advantage over Retrain. Predictive utility remains in the same range as Retrain; we do not interpret

occasional F1 values above Retrain as superiority, because both models are operating far from the original training distribution.

Table 13: Results under an extreme unlearning setting with 80% node removal.

Dataset	Method	GCN		GAT		SGC		GIN	
		F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)
Cora	Retrain	42.97	16.32	71.65	20.54	54.96	14.28	69.37	16.20
	RTR	46.54	0.39	72.47	1.21	56.73	0.38	61.99	0.62
Citeseer	Retrain	36.41	19.21	65.21	42.66	36.56	16.48	62.46	30.23
	RTR	40.71	0.53	68.89	1.32	44.11	0.46	61.26	0.67

D.5 Recent baselines

Table 14: Two protocol-specific snapshots, not one ranking. GCN rows follow the OpenGU 80/20 protocol of Tab. 6; GAT rows follow the GIF-style 90/10 protocol of Tab. 2. CGE and ETR were run under the protocol of the corresponding block.

Backbone	Method	Cora		Citeseer		CS	
		F1-score	RT (second)	F1-score	RT (second)	F1-score	RT (second)
GCN	Retrain	83.94	5.23	76.55	7.65	92.05	99.64
	CGE(Li et al., 2025)	75.86	1.89	70.82	2.94	78.06	7.12
	ETR(Yang et al., 2025)	83.54	0.03	77.09	0.03	92.72	0.07
	RTR	84.02	0.32	74.83	0.28	91.98	1.13
GAT	Retrain	87.45	15.72	77.05	35.47	91.24	128.37
	CGE	74.63	4.18	74.73	6.58	76.64	9.33
	ETR	79.28	0.03	75.92	0.04	90.96	0.08
	RTR	86.86	0.76	78.13	0.94	91.86	1.23

We additionally compare against CGE (Li et al., 2025) and ETR (Yang et al., 2025). CGE maps the graph to community-level units and updates only the affected regions. ETR is a training-free parameter editor with an erase-then-rectify pipeline. Tab. 14 is a complementary snapshot, not a pooled ranking. GCN rows reuse the OpenGU 80/20 numbers of Tab. 6; GAT rows reuse the GIF-style 90/10 numbers of Tab. 2. CGE and ETR were run under the protocol of the corresponding block. CGE is faster than retraining but often farther from Retrain. ETR is the fastest method in this table, with mixed utility relative to Retrain. RTR remains much faster than retraining and closer to Retrain than CGE in these settings; we do not claim dominance over ETR.

E Method Extensions

The main paper already reweights $\mathcal{L}^{\text{Inter}}$, $\mathcal{L}^{+}$, and $\mathcal{L}^{\text{Local}}$ by the deletion ratio β (Eq. 5). The two adaptations below handle the extremes of that spectrum: too few forgotten nodes to supervise reconstruction, and a deletion large enough that the embedding distribution itself shifts. They do not change the reconstruct-then-rectify pipeline.

E.1 Low-ratio Removal

When $|\mathcal{U}|$ is small, Eq. 3 provides little reconstruction supervision. Because RTR decouples training from inference, retained nodes $\mathcal{R}$ can also serve as interaction targets: each retained node still participates in neighborhood aggregation, so its fused hidden representation is a valid reconstruction destination. The extended objective is

$$\mathcal{L}^{\text{Inter+}} = \frac{1}{|\mathcal{U} \cup \mathcal{R}|} \sum_{m \in \mathcal{U} \cup \mathcal{R}} \text{KL}\left(\text{Norm}[\boldsymbol{h}^{*}_{e_m}], \text{Norm}\Big[\sum_{i \in \mathcal{N}_{e_m}} \boldsymbol{s}^{\text{RND}}_{i \to m} \Big] \right). \tag{10}$$

Inference is unchanged: only interactions from forgotten neighbors are subtracted, as in Eq. 2.

E.2 High-ratio Removal

When the forgotten-node ratio is large—in our experiments, above 30%—graph structure dominates the embedding distribution, and a small correction applied to the original-graph embeddings need not land near the retrained distribution. Fig. 8(a) shows that retraining can deviate in angle from training-from-all, while a naive unlearning distribution can remain aligned with that (Fig. 8(b)).

To adapt, we first infer on the pruned graph $G' = G/\Delta G$ with the frozen original model, $\hat{\boldsymbol{h}}^k_{e_i} = \mathrm{GNN}_0(G')$, and then subtract only the remaining neighboring forgotten interactions:

$$\tilde{\boldsymbol{h}}^k_{e_i} = \hat{\boldsymbol{h}}^k_{e_i} - \sum_{j \in \mathcal{U} \cap \mathcal{N}_{e_i}} \boldsymbol{c}_{j \to i}. \tag{11}$$

The local-search loss is the main-text $\mathcal{L}^{\text{Local}}$ evaluated on these pruned-graph teacher embeddings.

This is the same teacher construction already used in Eq. 5; the high-ratio change is the inference initialization in Eq. 11, not a new training principle.

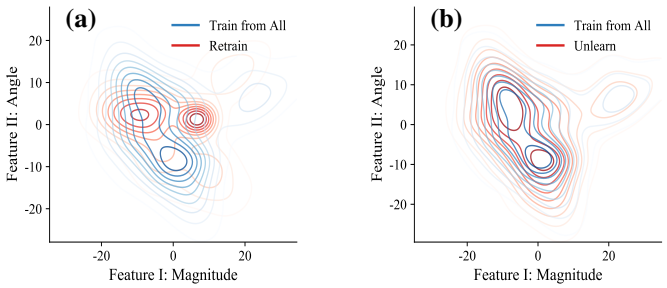


Figure 8: Embeddings distribution of SGC unlearning 40% nodes.

F Extended Related Work

F.1 Machine Unlearning

Machine unlearning removes the influence of a specified training subset from an already trained model, typically for privacy, security, or utility reasons—for example, to honor deletion requests or to revoke the effect of noisy or poisoned data. Since the formulation of Cao & Yang (2015), existing methods are commonly grouped into *exact* unlearning (Ginart et al., 2019; Karasuyama & Takeuchi, 2010; Bourtoule et al., 2021), *approximate* unlearning (Koh & Liang, 2017; Izzo et al., 2021), and *certified* variants that give formal deletion guarantees under explicit assumptions (Guo et al., 2020; Chien et al.; Wu et al., 2023b).

Exact methods seek a model that is identical to one trained without the deleted data. Retraining from scratch meets this requirement but is computationally demanding. *SISA* (Sharded, Isolated, Sliced, and Aggregated) (Bourtoule et al., 2021) partitions the data, trains constituent models on disjoint shards, and aggregates them at inference. A deletion request then retrains only the affected shard, which reduces the retraining scope but still requires a partial retrain and an aggregation step.

Approximate methods update an existing model instead of rebuilding it. Influence-function updates (Koh & Liang, 2017; Guo et al., 2020) estimate the parameter change induced by deleted points, typically through a Newton-style correction; certified variants further add noise or restrict the function class so that a deletion guarantee can be stated. Unrolling SGD (Thudi et al., 2022) regularizes a verification error that approximates the distance to a retrained model. Langevin unlearning (Chien et al., 2024) analyzes noisy gradient descent as an unlearning mechanism. These approaches avoid full retraining, but their accuracy depends on the quality of the local approximation and on how large the deletion is relative to the original training set.

This paper studies efficient *approximate* node unlearning on graphs. We do not claim a certified deletion guarantee; the goal is to approach the behavior of retraining while leaving the base GNN parameters fixed.

Table 15: Graph unlearning method families across three design dimensions. Check marks describe the typical family behavior, not every individual paper.

Method family	Retrain-free	Full-estimation-free	High-ratio setting evaluated
Shards-based	✗	✔	✗
IF-based	✔	✔	✗
Learning-based	✔	✗	✔
Others	✔	✔	✗
RTR	✔	✔	✔

F.2 Graph Unlearning

Graph unlearning specializes the problem to GNNs, where a deleted node removes both its features and its incident edges, and neighborhood aggregation mixes those effects across layers. We group prior methods by their operational mechanism; Tab. 15 summarizes the resulting design trade-offs.

Shards-based methods. GraphEraser (Chen et al., 2022b) and GUIDE (Wang et al., 2023) adapt SISA to graphs by partitioning nodes so that shard-wise models retain as much structural information as possible, then aggregating their predictions. GraphRevoker (Zhang, 2024) adds property-aware sharding and contrastive sub-model aggregation. These methods avoid estimating a full-graph influence, but they still retrain the affected shard, depend on a reasonable partition, and incur an aggregation cost at inference.

Influence-function methods. CGU (Chien et al.), GIF (Wu et al., 2023a), CEU (Wu et al., 2023b), IDEA (Dong et al., 2024), and GST (Pan et al., 2023) extend influence-function updates to edge and node edits and analyze approximation error under Lipschitz and convergence conditions. ScaleGUN is included as an additional influence-style baseline in Tab. 1. The underlying local expansion is most reliable for moderate structural edits. Node deletion is a harder regime: it changes both features and adjacency, and the inverse-Hessian cost grows with graph size. We therefore compare these methods under the node-unlearning protocols of Sec. 6 rather than under an edge-only protocol.

Learning-based methods. GNNDelete (Cheng et al., 2023) inserts a deletion operator that bounds edge prediction, with less emphasis on matching retained-node embeddings. MEGU (Li et al., 2024) couples a mutual-evolution objective with an adaptive high-impact-node set. These methods can handle larger deletion ratios than a first-order influence update, but they typically train additional modules from the original model's predictions, so selection bias in that teacher can propagate into the unlearned model.

Closed-form and recent editors. Projector (Cong & Mahdavi, 2023) gives a closed-form update with theoretical guarantees for linear GNNs; nonlinear backbones require additional extensions. Two more recent baselines appear in the extra experiments of Appendix D.5: CGE (Li et al., 2025) performs community-level localized updates, and ETR (Yang et al., 2025) is a training-free erase-then-rectify parameter editor. We treat them as complementary points on the utility–efficiency spectrum, not as members of a pooled ranking with the main-paper tables.