RECOGU: RETENTION-AWARE COARSENING FOR GRAPH UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph unlearning seeks to remove the influence of requested data while preserving utility on retained data. Reducing the training graph can lower retraining costs, but the resulting changes in features, structure, and supervision weights can introduce additional utility loss relative to full retraining on the retained graph. We propose ReCoGU, Retention-Aware Coarsening for Graph Unlearning, which constructs label-pure training units, merges adjacent small units with the same label, and represents each unit with retained-member mean features and binary inter-unit connections. ReCoGU stores feature sums, unit sizes, and inter-unit edge counts, updates only affected statistics for each request, and retrains from scratch on the updated coarse graph with square-root unit-size loss weights. We define collateral utility loss (CUL) as the difference between the average classification losses of ReCoGU and full retraining on the same retained training graph. For finite GNN outputs, aligned labels, and a Lipschitz loss, we derive a bound separating prediction distortion at the two trained parameter sets, supervision-weight correction, and surrogate optimization error. Across five datasets and three GNN backbones, ReCoGU has the lowest unlearning time in 10 of 15 settings and the second-lowest in the remaining five. Compared with baselines other than Retrain, ReCoGU achieves the highest Micro-F1 in 14 of the 15 settings. Its unlearning time ranges from 0.036 to 4.964 seconds, compared with 2.865 to 384.069 seconds for full retraining. On Cora and Physics, prediction distortion is positively associated with CUL (Spearman $\rho = 0.904$ and 0.629), and ablations show that mean features and inter-unit connections both reduce CUL.

1 INTRODUCTION

Graph data provide a natural representation of entities and their relationships, and graph neural networks (GNNs) exploit this structure through message passing to support learning in applications such as citation analysis, recommendation, social networks, and finance (Kipf & Welling, 2016; Hamilton et al., 2017; Wu et al., 2020). However, graph data can also contain sensitive attributes, interactions, and relationships, while neighborhood aggregation may allow a trained GNN to retain information about individual nodes and their local contexts (Zhang et al., 2024; He et al., 2021). To protect personal data, privacy regulations increasingly emphasize individuals' rights to control its use and request its removal (Protection, 2018; Pardau, 2018). A direct way to satisfy such a request is to remove the requested data and retrain the GNN from scratch, but repeating this process over the entire graph is expensive (Bourtoule et al., 2021; Ginart et al., 2019; Guo et al., 2020). This trade-off between data removal and retraining cost motivates graph unlearning, which seeks to remove the influence of requested graph data efficiently while preserving the model's utility on the retained graph.

Most efficient graph unlearning methods follow two broad routes. A more detailed discussion of these graph unlearning paradigms is provided in Appendix A. Partitioned retraining methods retrain only selected parts of the graph (Chen et al., 2022; Wang et al., 2023), whereas parameter update methods modify a trained model to approximate the effect of an unlearning request (Wu et al., 2023a; Cheng et al., 2023; Yang et al., 2025). Both routes reduce computation by limiting either the graph region that is retrained or the parameters that are updated. Recent methods based on graph coarsening take a complementary approach: they replace the retained graph with fewer training units and retrain on the resulting compact graph (Li et al., 2025; 2026). However, once an unlearning

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.

request changes the retained nodes and edges, it is unclear whether the updated coarse graph still reflects the training problem on the retained graph. Coarsening may change unit labels, member features, and the connections used for message passing, making the additional utility loss relative to full retraining difficult to identify. This calls for a coarsening scheme that keeps unit labels aligned with retained nodes, preserves the structural information needed for message passing, and makes the additional utility loss measurable.

After an unlearning request, full retraining updates its objective to reflect the retained graph, whereas compression additionally changes unit membership, representative features, inter-unit connectivity, and supervision weights. ReCoGU preserves label-pure units so that surviving labels remain aligned, uses member means that minimize within-unit squared reconstruction error for a fixed partition, and stores additive feature and inter-unit edge statistics so that only affected quantities are updated. These invariants isolate the residual discrepancy: the weighted objective optimized on the coarse graph is not generally identical to the empirical risk evaluated over retained nodes. We quantify this discrepancy relative to full retraining as collateral utility loss and decompose it into prediction distortion, supervision-weight correction, and surrogate optimization error. This decomposition clarifies the role of each mechanism and provides a request-level diagnostic of the efficiency–utility trade-off.

ReCoGU addresses this question by separating information that must remain exact from information that can be compressed. It first constructs label-pure training units and merges only small adjacent units that share the same label, recovering compactness without changing the supervision label of a surviving unit. Each unit is represented by the mean feature of its retained members, and binary connections preserve whether two units are linked. ReCoGU stores the feature sums, unit sizes, and inter-unit edge counts. An unlearning request then updates only the affected statistics; the fixed unit partition can be reused for later requests. The updated coarse graph is trained from scratch with square-root unit-size loss weights, and the resulting model is evaluated on the retained node-level graph. This construction combines label consistency, local graph maintenance, and a smaller training graph in one unlearning procedure.

We connect this construction to retained utility by comparing ReCoGU with full retraining on the same retained graph. We define collateral utility loss (CUL) as the difference between the average classification loss of ReCoGU and that of full retraining, both evaluated on the same retained training graph. Under finite outputs, aligned labels, and a Lipschitz loss, the derived bound separates the gap into prediction distortion at the two trained parameter sets, the net correction caused by supervision reweighting, and the optimization error of the surrogate training objective. The fixed-parameter prediction distortion therefore provides an empirical diagnostic for how changes in the coarsened graph contribute to CUL. The main contributions of this paper are summarized as follows:

- **Retention-aware training surrogate.** We propose ReCoGU, which combines label-pure training units, structural merging of units with the same label, local updates of retained statistics, square-root unit-size loss weighting, and retraining from scratch to reduce unlearning cost while keeping supervision aligned with the retained graph.
- **Risk-based analysis.** We formalize collateral utility loss relative to full retraining on the retained graph and derive a bound that separates prediction distortion, supervision-weight correction, and optimization error.
- **Empirical evaluation.** We evaluate utility, unlearning time, and membership inference across five datasets and three GNN backbones, and analyze the relation between prediction distortion and retained utility loss under different unlearning requests through controlled ablations.

2 PRELIMINARIES

We consider node classification on an undirected attributed graph:

$$G = (V, E, X, Y). \tag{1}$$

Here, V and E are the node and edge sets, $X = [x_v]_{v \in V}$ is the node-feature matrix with $x_v \in \mathbb{R}^d$, and $Y = (y_v)_{v \in V}$ contains labels from $[K_y] = \{1, \ldots, K_y\}$. The nodes are partitioned into training, validation, and test sets, $V = V_{\mathrm{tr}} \dot{\cup} V_{\mathrm{val}} \dot{\cup} V_{\mathrm{te}}$. For a node subset $S \subseteq V$, $G[S]$ denotes the induced

subgraph; in particular, the training graph is $G_{\mathrm{tr}} = G[V_{\mathrm{tr}}]$, with feature matrix X_{tr}, binary adjacency matrix A_{tr}, and size $n_{\mathrm{tr}} = |V_{\mathrm{tr}}|$. ReCoGU constructs its training units and processes unlearning requests using the training nodes and their labels. A GNN with parameters $\theta \in \Theta$ maps a graph H and one of its vertices u to a logit vector $f_\theta(H)_u \in \mathbb{R}^{K_y}$, and $\ell(z, y)$ denotes the node-classification loss for a logit vector z and label y. Adjacency matrices record binary inter-vertex connections; self-loops and normalization, when used by a backbone, are applied consistently to the node-level and surrogate graphs.

A node-unlearning request is specified by a set of training nodes $V_u \subseteq V_{\mathrm{tr}}$. The request removes these nodes together with their incident edges from the retained training problem, giving

$$V'_{\mathrm{tr}} = V_{\mathrm{tr}} \setminus V_u, \qquad G'_{\mathrm{tr}} = G_{\mathrm{tr}}[V'_{\mathrm{tr}}]. \tag{2}$$

The corresponding feature, label, and adjacency restrictions are denoted by X'_{tr}, Y'_{tr}, and A'_{tr}, respectively. Let θ'_r denote the parameters obtained by retraining the same GNN backbone from scratch on G'_{tr} under the retained training protocol. This full-retraining model is the reference for retained-data utility after the request. ReCoGU subsequently constructs a compact training surrogate $\widehat{G}'$ and trains a model with parameters $\theta'_{\mathrm{ReCoGU}}$ on that surrogate. The construction of $\widehat{G}'$ is described in Section 3, while the retained empirical-risk comparison between the two models is defined and analyzed in Section 4.

3 RECOGU

3.1 MOTIVATION

Graph coarsening can reduce GNN training cost by replacing the retained graph with a compact training surrogate (Huang et al., 2021; Xia et al., 2025). However, after an unlearning request, changes in retained members and edges may cause the surrogate's labels, features, and message-passing structure to deviate from the training problem that full retraining would solve. An effective surrogate must therefore satisfy three requirements: it should be compact enough to reduce retraining cost, faithful enough to preserve the retained training problem, and locally updatable enough to support repeated requests without reconstructing the partition. ReCoGU addresses these requirements by fixing a label-pure partition, merging only adjacent small units with the same label, representing each surviving unit with the mean features of its retained members and binary inter-unit connections, and locally updating affected statistics before cold-start retraining. This design preserves label correspondence and retained structural information while providing a compact surrogate whose discrepancy from full retraining can be analyzed through collateral utility loss; the overall framework is shown in Fig. 1.

3.2 LABEL-PURE PARTITIONING FOR UNLEARNING

To establish training units whose supervision remains valid throughout unlearning, ReCoGU first constructs a label-pure partition on the training graph. Let $G_{\mathrm{tr}} = G[V_{\mathrm{tr}}]$ denote the subgraph induced by the training nodes. Following supervised graph coarsening (Xia et al., 2025), ReCoGU initializes units with class-balanced high-degree centers, refines mixed units with label-aware shortest-path splits, and enforces strict label purity. It then merges only small adjacent units that share the same label, recovering compactness without changing the supervision of any surviving unit. The full center-selection, refinement, purity, and merge-threshold construction is given in Appendix B. Let the resulting partition be $\mathcal{C} = \{C_1, \ldots, C_B\}$, with mapping $\pi(v) = i$ for $v \in C_i$. For an unlearning request $V_u \subseteq V_{\mathrm{tr}}$, the surviving members of C_i are

$$C'_i = C_i \setminus V_u. \tag{3}$$

If $C'_i \neq \emptyset$, then $T'_i = 1$ follows directly from $T_i = 1$. Hence, arbitrary node removal preserves label purity without repartitioning. This property is important for both sides of ReCoGU's objective: it avoids the repeated cost of reconstructing the partition, while keeping the label information of the reduced training problem exactly aligned with that of full retraining on the retained graph. The fixed mapping π can therefore be restricted to the retained nodes and reused for subsequent graph updates.

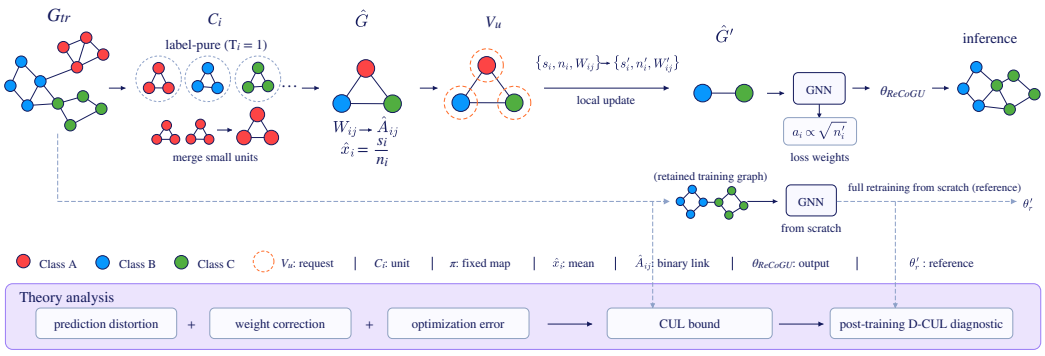

Figure 1: **The framework of ReCoGU.** The training graph is partitioned into label-pure units, and adjacent small units with the same label are merged. For an unlearning request, ReCoGU locally updates the affected unit and constructs an updated surrogate graph. A GNN is then retrained from scratch with square–root unit–size loss weights ($\omega'_i \propto \sqrt{n'_i}$). The gray dashed branch denotes full retraining on the retained graph. The purple strip summarizes the three terms in the CUL bound—prediction distortion, supervision–weight correction, and surrogate optimization error—and their relation to collateral utility loss.

3.3 Coarsened Graph Construction

To incorporate the label-pure partition into a compact training graph, ReCoGU maps each training unit to a supernode, represents its features by the mean of its member features, and constructs superedges from the connections between units. The resulting graph reduces the scale of retraining while preserving label consistency and cross-unit connectivity, supporting model utility preservation on the retained graph.

For the partition $\mathcal{C} = \{C_1, \ldots, C_B\}$, each unit becomes one supernode. Its feature is the mean of its training members, $\widehat{x}_i = n_i^{-1} \sum_{v \in C_i} x_v$, and its label $\widehat{y}_i$ is the common member label. For units C_i and C_j, let $m_{ij} = \sum_{u \in C_i} \sum_{v \in C_j} A_{\mathrm{tr}}(u, v)$ and set $\widehat{A}_{ij} = \mathbb{I}[m_{ij} > 0]$ for $i \neq j$. Thus, the surrogate keeps member-level labels and the existence of cross-unit connections while reducing the number of training vertices. ReCoGU stores the feature sums $S = \widehat{P}^\top X_{\mathrm{tr}}$, unit sizes, and inter-unit edge counts for local updates. The full matrix construction and its feature-error rationale are provided in Appendix C.

3.4 Graph Update and Retraining

After an unlearning request, the coarse graph must reflect the retained nodes and their surviving connections before retraining. ReCoGU achieves this through localized graph updates, followed by retraining with unit-size-weighted loss. This design reduces graph reconstruction cost while accounting for differences in training unit size during retraining.

Given an unlearning request $V_u \subseteq V_{\mathrm{tr}}$, let $V'_{\mathrm{tr}} = V_{\mathrm{tr}} \setminus V_u$ and $G'_{\mathrm{tr}} = G_{\mathrm{tr}}[V'_{\mathrm{tr}}]$. For each training unit C_i, its retained members are $C'_i = C_i \setminus V_u$. Using the mapping π and the statistics stored in Section 3.3, ReCoGU updates the feature sum and unit size as

$$s'_i = s_i - \sum_{v \in C_i \cap V_u} x_v, \qquad n'_i = n_i - |C_i \cap V_u|. \tag{4}$$

For each nonempty unit, the updated feature and label are

$$\widehat{x}'_i = \frac{s'_i}{n'_i}, \qquad \widehat{y}'_i = \widehat{y}_i, \quad n'_i > 0. \tag{5}$$

Thus, the updated features are computed directly from the retained members, while label purity is preserved without repartitioning.

Next, ReCoGU updates the connections between training units. Let $d_{ij}(V_u)$ denote the number of original edges between C_i and C_j with at least one endpoint in V_u, counting each undirected edge once. The updated edge counts and coarse adjacency are

$$m'_{ij} = m_{ij} - d_{ij}(V_u), \qquad \widehat{A}'_{ij} = \mathbb{I}[m'_{ij} > 0], \quad i \neq j. \tag{6}$$

ReCoGU removes empty units and edges with zero counts, obtaining the updated training graph $\widehat{G}' = (\widehat{V}', \widehat{E}', \widehat{X}')$ with labels $\widehat{Y}'$. This update preserves the connections between surviving training units while limiting reconstruction to the affected units and the edges connected to them.

ReCoGU then retrains the GNN from scratch on $\widehat{G}'$. To account for differences in unit size, ReCoGU weights each training unit by the square root of its updated size. This gives larger units greater influence, with their weights growing sublinearly with unit size. Let $\mathbf{C}' = \mathrm{diag}(n'_1, \ldots, n'_{B'})$ denote the updated unit-size matrix, where $B' = |\widehat{V}'|$ and surviving units are reindexed from 1 to B'. The retraining objective is

$$\begin{aligned} \omega'_i &= \frac{B'\sqrt{n'_i}}{\sum_{j=1}^{B'} \sqrt{n'_j}}, \\ \mathcal{L}(\theta) &= \frac{1}{B'} \sum_{i=1}^{B'} \omega'_i \, \ell\Big(f_\theta(\widehat{G}')_i, \widehat{y}'_i \Big). \end{aligned} \tag{7}$$

Here, ℓ denotes the node classification loss and $f_\theta(\widehat{G}')_i$ is the prediction for training unit i. The weights have a mean of one across surviving units, allowing their contributions to reflect unit size without changing the overall loss scale. ReCoGU optimizes this objective from a fresh initialization, and we denote the attained parameters by $\theta'_{\mathrm{ReCoGU}}$. The resulting model is applied to the retained node-level graph for inference.

With indexed unit statistics, the update touches only requested nodes and their incident edges; the explicit complexity expression and implementation boundary are given in Appendix D.

4 Theoretical Analysis

Graph coarsening reduces GNN training cost by providing a compact training graph (Huang et al., 2021). ReCoGU applies this strategy after an unlearning request. We analyze the resulting utility cost relative to full retraining after the same request, following the retraining reference used in machine unlearning (Bourtoule et al., 2021; Sekhari et al., 2021). The central question is how closely the surrogate preserves the training problem that full retraining would solve. Our analysis connects prediction distortion to the retained empirical risk gap, accounting for how square-root weighting determines each unit's contribution to the training objective and how closely training approaches the surrogate optimum.

4.1 The Retained Training Problem and Its Surrogate

We characterize the retained training problem and ReCoGU's surrogate through their prediction outputs and supervision weights. This formulation establishes a basis for analyzing retained-data utility after unlearning. To isolate the cost of compression, we compare both training problems on the same retained data. Let $V'_{\mathrm{tr}} = V_{\mathrm{tr}} \setminus V_u$, $n'_{\mathrm{tr}} = |V'_{\mathrm{tr}}| > 0$, and $G'_{\mathrm{tr}} = G_{\mathrm{tr}}[V'_{\mathrm{tr}}]$. The surviving units $\{C'_i\}_{i=1}^{B'}$ partition V'_{tr}, with $n'_i = |C'_i|$ and $\pi(v) = i$ for $v \in C'_i$. The updated graph $\widehat{G}'$ uses member means, binary superedges, and standard self-loops I. Inter-unit edge counts maintain these binary edges; they are not propagation weights.

Let θ denote the GNN parameters and f_θ its logit map, which outputs class scores before probability normalization. The same parameterized predictor applies to both graphs. Write $\ell(z, y)$ for the classification loss of a score vector z against label y. Here, y_v is node v's label, and $\widehat{y}'_i$ is the common label of unit i.

The retained empirical risk $\mathcal{R}'_{\mathrm{tr}}(\theta)$ is the average classification loss over retained training nodes. The surrogate objective $\mathcal{L}(\theta)$ is the weighted average classification loss over updated units. Each normalized weight a_i specifies unit i's share of this objective. It equals ω'_i / B', where ω'_i is the unit

weight used in Section 3 before averaging over B' units. These objectives are defined as follows.

$$\mathcal{R}'_{\mathrm{tr}}(\theta) = \frac{1}{n'_{\mathrm{tr}}} \sum_{v \in V'_{\mathrm{tr}}} \ell\big(f_\theta(G'_{\mathrm{tr}})_v, y_v\big), \tag{8}$$

$$\mathcal{L}(\theta) = \sum_{i=1}^{B'} a_i \ell\big(f_\theta(\widehat{G}')_i, \widehat{y}'_i\big), \qquad a_i = \frac{\omega'_i}{B'} = \frac{\sqrt{n'_i}}{\sum_{j=1}^{B'} \sqrt{n'_j}}. \tag{9}$$

Thus $\sum_i a_i = 1$, matching the loss normalization in Section 3. Larger units receive greater training weight, with relative contributions determined by the square roots of their retained sizes. The exponent $1/2$ is Section 3's chosen weighting rule; the analysis quantifies its effect without assuming that this exponent is optimal. Let $\theta'_{\mathrm{ReCoGU}}$ and θ'_r be the parameters obtained by ReCoGU and full retraining, respectively, in the same parameter space Θ. We define their *collateral utility loss (CUL)* as the difference between their empirical risks on the same retained training graph:

$$\mathrm{CUL}(V_u) = \mathcal{R}'_{\mathrm{tr}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{R}'_{\mathrm{tr}}(\theta'_r). \tag{10}$$

This measured gap can be negative; it is nonnegative when θ'_r globally minimizes $\mathcal{R}'_{\mathrm{tr}}$ over Θ.

For fixed parameters, we define the surrogate distortion $\Delta(\theta)$ as the mean squared distance between corresponding node and unit logits. We use the Euclidean norm $\|\cdot\|_2$ to measure each distance:

$$\Delta(\theta) = \frac{1}{n'_{\mathrm{tr}}} \sum_{v \in V'_{\mathrm{tr}}} \Big\| f_\theta(G'_{\mathrm{tr}})_v - f_\theta(\widehat{G}')_{\pi(v)} \Big\|_2^2. \tag{11}$$

This compares the two training representations through the actual nonlinear GNN, holding its parameters fixed. Evaluating this quantity through the nonlinear GNN captures how graph compression affects node-level predictions at fixed parameters. Under node-uniform supervision, each retained node contributes equally to the empirical risk. Unit i therefore receives total weight $p_i = n'_i / n'_{\mathrm{tr}}$, its proportion of retained training nodes. Square-root training assigns weight a_i to this unit. We define the weight correction $\mathcal{C}_{\mathrm{w}}(\theta)$ as the difference between these two weighted losses computed from the same surrogate predictions:

$$\mathcal{C}_{\mathrm{w}}(\theta) = \sum_{i=1}^{B'} (p_i - a_i) \ell\big(f_\theta(\widehat{G}')_i, \widehat{y}'_i\big). \tag{12}$$

The weight correction accounts exactly for supervision reweighting, while surrogate distortion measures prediction changes associated with graph compression. Together, these quantities provide the basis for bounding collateral utility loss relative to full retraining in the following analysis.

4.2 From Surrogate Distortion to Collateral Utility Loss

To relate the two training problems to retained-data utility, we use a standard comparison argument for empirical risk minimization (Shalev-Shwartz & Ben-David, 2014). ReCoGU's label-pure units ensure that each retained node and its representative unit share the same target label, $\widehat{y}_{\pi(v)} = y_v$, and this correspondence is preserved after local updates. We then assume a β-Lipschitz loss with respect to the logits, so prediction differences across the retained and surrogate graphs can be translated into loss differences. Standard multiclass cross-entropy satisfies this property for finite logits with $\beta = \sqrt{2}$ (Zhang et al., 2023), as shown in Appendix F. Combining this relation with the supervision-weight correction and optimization error yields the following bound on collateral utility loss relative to full retraining.

Theorem 1 (Collateral utility bound)**.** *For finite GNN outputs and the label correspondence above, define the surrogate optimization error*

$$\varepsilon_{\mathrm{opt}} = \mathcal{L}(\theta'_{\mathrm{ReCoGU}}) - \inf_{\theta \in \Theta} \mathcal{L}(\theta) \geq 0,$$

assuming the infimum is finite. Then

$$\begin{aligned} \mathrm{CUL}(V_u) \leq \beta \Big(\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)} \Big) \\ + \mathcal{C}_{\mathrm{w}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{C}_{\mathrm{w}}(\theta'_r) + \varepsilon_{\mathrm{opt}}. \end{aligned} \tag{13}$$

The complete proof, the cross-entropy Lipschitz verification, and the interpretation of each diagnostic term are given in Appendix F.

Remark 1 (Tightness and practical use)**.** *The right-hand side is an assumption-level worst-case envelope, rather than a claim that every request attains equality. The fixed-parameter transfer step is sharp under the stated assumptions: equality is approached when retained nodes have equal logit distances and their loss changes align with the maximal-slope direction. For cross-entropy, $\beta = \sqrt{2}$ is the sharp global Lipschitz supremum, approached as the logits tend to a wrong one-hot prediction. The correction $\mathcal{C}_{\mathrm{w}}$ is exact by Eq. equation 30, and $\varepsilon_{\mathrm{opt}}$ is exact by definition. Thus no uniformly smaller bound based only on $(\beta, \Delta, \mathcal{C}_{\mathrm{w}}, \varepsilon_{\mathrm{opt}})$ follows without additional assumptions. In practice, D, W, and η identify actionable sources of CUL: reduce D through mean features and retained inter-unit connections, reduce W through better supervision-weight alignment, and reduce η through surrogate optimization.*

For empirical evaluation, we collect the corresponding terms as

$$\begin{aligned} D &:= \beta\left[\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)}\right], \\ W &:= \mathcal{C}_{\mathrm{w}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{C}_{\mathrm{w}}(\theta'_r), \\ \eta &:= \mathcal{L}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{L}(\theta'_r). \end{aligned} \tag{14}$$

Here, D is the combined prediction-distortion term in the theorem, W is the net supervision-weight correction, and η compares the attained surrogate losses of the two trained models. In particular, η is distinct from $\varepsilon_{\mathrm{opt}}$, which measures the gap to the infimum of the surrogate objective.

Applying the same comparison argument directly to the two attained models gives the evaluable relation

$$\mathrm{CUL}(V_u) \le D + W + \eta. \tag{15}$$

5 Experiments

5.1 Experimental Setup

Experimental setup. We evaluate ReCoGU on five node-classification datasets (Cora (Yang et al., 2016), CiteSeer (Yang et al., 2016), PubMed (Yang et al., 2016), CS (Shchur et al., 2018), and Physics (Shchur et al., 2018)) with GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), and GCN (Kipf & Welling, 2016) backbones. The baselines are GUIDE (Wang et al., 2023), CGE (Li et al., 2025), ETR (Yang et al., 2025), TCGU (Li et al., 2026), GNNDelete (Cheng et al., 2023), and full Retrain. We report Micro-F1, unlearning time, membership-inference AUC, CUL, prediction distortion, and input-feature distortion. Unless stated otherwise, we use 80/10/10 splits, random 1% node requests, the same requests for all methods, and 20 independent runs. The complete metric definitions, implementation settings, request-ratio protocols, and timing procedure are given in Appendix E.

5.2 Experimental Results

Utility and unlearning efficiency. Table 1 reports Micro-F1 and unlearning time under node unlearning. Excluding the Retrain baseline, ReCoGU attains the highest Micro-F1 across 14 out of the 15 settings. Its Micro-F1 exceeds those of TCGU and GNNDelete in every configuration. Compared with retraining from scratch, ReCoGU improves Micro-F1 by 0.045–0.079 on Cora and by 0.005–0.012 on CiteSeer. On PubMed, CS, and Physics, the difference from retraining from scratch ranges from -0.026 to 0.013; the higher F1 on Cora, under the same tuning protocol, may reflect a regularization or smoothing effect from coarsened training. These results show that the coarsened training graph preserves node classification utility across different graph scales and GNN backbones.

ReCoGU obtains the lowest unlearning time in 10 of the 15 configurations and the second-lowest time in the remaining five. The five exceptions are PubMed–GAT, PubMed–GCN, CS–GraphSAGE, CS–GAT, and CS–GCN, where ETR has a lower unlearning time. Across all configurations, ReCoGU requires 0.036–4.964 seconds, whereas retraining from scratch requires 2.865–384.069

Table 1: Model utility and unlearning efficiency under node unlearning.

Dataset	GNN	Retrain		GUIDE		GNNDelete		CGE		ETR		TCGU		ReCoGU	
		F1	T (s)	F1	T (s)	F1	T (s)	F1	T (s)	F1	T (s)	F1	T (s)	F1	T (s)
Cora	GraphSAGE	0.806 ± 0.009	3.784	0.736 ± 0.010	1.432	0.752 ± 0.014	0.894	0.844 ± 0.006	1.156	0.777 ± 0.008	0.123	0.803 ± 0.007	1.384	$\mathbf{0.857 \pm 0.006}$	**0.036**
	GAT	0.821 ± 0.009	2.865	0.694 ± 0.011	1.734	0.764 ± 0.016	1.137	0.800 ± 0.007	1.392	0.782 ± 0.008	0.117	0.816 ± 0.008	1.863	$\mathbf{0.866 \pm 0.008}$	**0.093**
	GCN	0.795 ± 0.008	4.362	0.801 ± 0.009	1.375	0.741 ± 0.013	0.812	0.640 ± 0.012	1.103	0.788 ± 0.008	0.094	0.801 ± 0.006	1.214	$\mathbf{0.874 \pm 0.007}$	**0.058**
CiteSeer	GraphSAGE	0.731 ± 0.009	5.476	$\mathbf{0.741 \pm 0.009}$	1.178	0.706 ± 0.013	0.957	0.713 ± 0.010	1.029	0.718 ± 0.009	0.163	0.728 ± 0.008	1.437	0.736 ± 0.008	**0.055**
	GAT	0.715 ± 0.010	8.207	0.716 ± 0.010	0.968	0.674 ± 0.018	1.246	0.635 ± 0.012	0.834	0.688 ± 0.009	0.135	0.722 ± 0.009	2.047	$\mathbf{0.727 \pm 0.010}$	**0.108**
	GCN	0.737 ± 0.009	5.903	0.732 ± 0.010	1.473	0.713 ± 0.012	0.873	0.615 ± 0.012	1.278	0.730 ± 0.009	0.116	0.739 ± 0.007	1.268	$\mathbf{0.742 \pm 0.009}$	**0.077**
PubMed	GraphSAGE	$\mathbf{0.887 \pm 0.004}$	27.264	0.749 ± 0.007	74.437	0.838 ± 0.009	2.183	0.815 ± 0.006	9.706	0.832 ± 0.004	0.207	0.851 ± 0.005	2.671	0.861 ± 0.003	**0.112**
	GAT	$\mathbf{0.863 \pm 0.004}$	38.665	0.644 ± 0.010	69.508	0.812 ± 0.014	2.715	0.825 ± 0.006	9.063	0.807 ± 0.006	**0.236**	0.827 ± 0.006	3.214	0.854 ± 0.004	0.368
	GCN	$\mathbf{0.884 \pm 0.004}$	30.217	0.859 ± 0.005	120.913	0.836 ± 0.011	1.968	0.769 ± 0.007	8.776	0.850 ± 0.005	**0.158**	0.857 ± 0.005	2.436	0.866 ± 0.003	0.172
CS	GraphSAGE	$\mathbf{0.951 \pm 0.002}$	252.926	0.826 ± 0.005	149.134	0.915 ± 0.007	3.981	0.809 ± 0.006	19.435	0.919 ± 0.002	**1.224**	0.918 ± 0.004	3.392	0.931 ± 0.003	1.579
	GAT	0.923 ± 0.002	292.183	0.714 ± 0.008	150.937	0.906 ± 0.009	4.587	0.791 ± 0.007	19.674	0.925 ± 0.003	**1.485**	0.911 ± 0.004	3.746	$\mathbf{0.936 \pm 0.003}$	1.649
	GCN	0.923 ± 0.002	241.336	0.722 ± 0.008	147.924	0.897 ± 0.008	3.642	0.798 ± 0.006	19.273	0.917 ± 0.002	**1.018**	0.914 ± 0.003	3.183	$\mathbf{0.930 \pm 0.002}$	1.397
Physics	GraphSAGE	$\mathbf{0.964 \pm 0.001}$	281.065	0.860 ± 0.004	329.387	0.927 ± 0.008	8.071	0.851 ± 0.005	42.924	0.932 ± 0.002	4.236	0.941 ± 0.003	4.963	0.963 ± 0.001	**3.415**
	GAT	0.955 ± 0.001	384.069	0.752 ± 0.006	330.554	0.934 ± 0.007	9.524	0.830 ± 0.005	43.078	0.955 ± 0.002	5.117	0.938 ± 0.003	5.472	$\mathbf{0.960 \pm 0.001}$	**4.964**
	GCN	$\mathbf{0.965 \pm 0.001}$	319.367	0.959 ± 0.002	71.368	0.881 ± 0.021	7.386	0.826 ± 0.005	42.876	0.947 ± 0.002	3.645	0.943 ± 0.002	4.618	0.963 ± 0.001	**2.912**

Table 2: Membership inference attack AUC (%) under node unlearning.

Dataset	GNN	Original	Retrain	GUIDE	GNNDelete	CGE	ETR	TCGU	ReCoGU
Cora	GCN	73.3	51.0	51.2	56.4	50.6	54.6	52.6	51.0
	GAT	75.2	50.8	49.7	57.4	50.4	55.2	52.6	50.2
	SAGE	74.1	51.0	50.7	56.3	51.6	54.5	52.5	50.7
CiteSeer	GCN	78.3	51.0	49.3	57.8	51.6	55.6	52.9	51.0
	GAT	79.1	51.0	50.4	59.1	50.9	56.3	53.1	49.6
	SAGE	79.7	51.0	49.9	57.9	50.2	55.6	52.9	51.2
PubMed	GCN	70.5	50.9	49.6	55.4	50.3	53.8	52.3	50.5
	GAT	71.8	51.0	50.1	56.8	51.0	54.5	52.5	50.3
	SAGE	69.3	51.1	50.8	55.1	51.0	53.6	52.2	49.8
CS	GCN	73.1	51.1	50.6	55.7	49.8	54.2	52.4	50.6
	GAT	72.3	50.9	50.5	56.2	50.6	54.3	52.3	49.4
	SAGE	70.8	51.3	49.9	55.6	50.7	54.0	52.5	50.0
Physics	GCN	68.7	51.2	50.0	55.1	51.0	53.7	52.4	50.7
	GAT	70.2	51.5	51.4	56.4	49.6	54.5	52.8	49.9
	SAGE	67.9	51.7	49.3	55.1	51.0	54.0	52.8	51.1

seconds. On Physics, for example, ReCoGU requires 3.415, 4.964, and 2.912 seconds with GraphSAGE, GAT, and GCN, respectively, compared with 281.065, 384.069, and 319.367 seconds for retraining from scratch. The efficiency gain results from updating retained unit statistics and retraining on the updated coarsened training graph instead of retraining on all retained nodes. These results address the gap between efficient graph unlearning and utility preservation: ReCoGU substantially reduces request-processing time while maintaining competitive node classification performance.

Unlearning efficacy. We evaluate unlearning efficacy using the node membership inference attack AUC reported in Table 2 (Shokri et al., 2017). The original models exhibit substantial membership leakage, with AUC values ranging from 67.9% to 79.7%. Retrain, GUIDE, CGE, and ReCoGU yield AUC values close to random guessing, ranging from 49.3% to 51.7%, because the requested nodes are excluded before retraining. The parameter-update methods GNNDelete and ETR remain farther from 50%, ranging from 53.6% to 59.1%, indicating that approximate updates retain more membership information, while TCGU, which retrains on transferred condensed data, lies between them at 52.2%–53.1%. ReCoGU retrains from scratch on updated coarsened graph; the requested nodes therefore contribute no features, labels, or topology to the unlearned model.

Prediction distortion and collateral utility loss. Figure 2 shows a positive association between the combined prediction-distortion term D and CUL. On Cora, the mean D increases from 0.866 to 1.326 as the request ratio grows from 5% to 20%, while the mean CUL increases from 0.029 to 0.059, yielding a Spearman correlation of 0.904. Physics shows the same overall tendency, with mean D increasing from 1.108 to 1.558 and a Spearman correlation of 0.629, although CUL remains nearly unchanged between the 10% and 20% requests. These results support using prediction distortion as an empirical diagnostic of retained utility loss, while the non-monotone behavior on Physics is consistent with additional contributions from supervision reweighting and optimization.

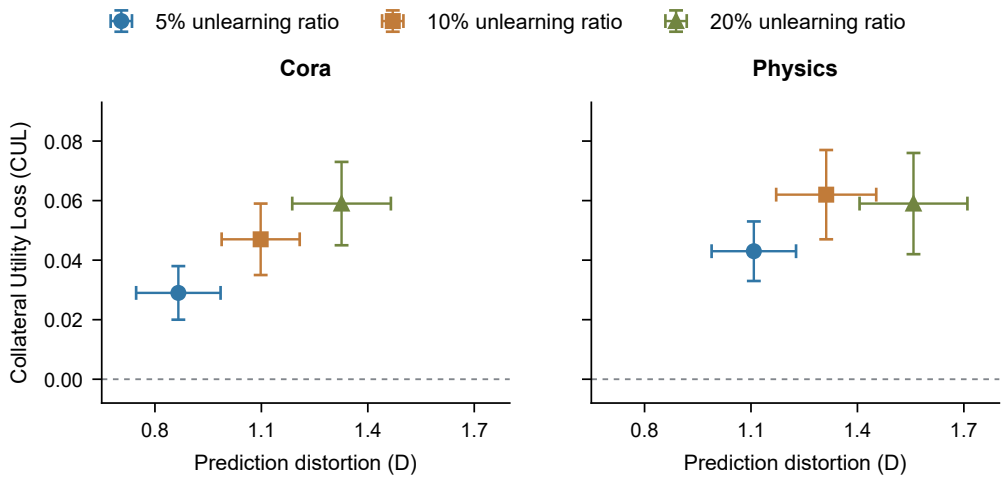

Figure 2: **Combined prediction distortion and collateral utility loss on Cora and Physics.** Markers denote means over five paired runs at unlearning ratios of 5%, 10%, and 20%. Horizontal and vertical bars denote the sample standard deviations of the combined prediction-distortion term $D = \beta[\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)}]$ and CUL, respectively.

Table 3: Ablations of mean-feature and cross-unit-connection.

Dataset	Variant	e_0	$\sqrt{\Delta(\theta'_r)}$	CUL
Cora	ReCoGU	$\mathbf{0.191 \pm 0.004}$	$\mathbf{0.326 \pm 0.031}$	$\mathbf{0.047 \pm 0.012}$
	Member representative	0.235 ± 0.006	0.392 ± 0.034	0.054 ± 0.013
	No cross-unit edges	$\mathbf{0.191 \pm 0.004}$	0.717 ± 0.077	0.096 ± 0.020
Physics	ReCoGU	$\mathbf{0.238 \pm 0.007}$	$\mathbf{0.439 \pm 0.043}$	$\mathbf{0.062 \pm 0.015}$
	Member representative	0.286 ± 0.008	0.486 ± 0.035	0.066 ± 0.009
	No cross-unit edges	$\mathbf{0.238 \pm 0.007}$	0.814 ± 0.078	0.127 ± 0.031

All request-level evaluations also satisfy $\mathrm{CUL} \leq D + W + \eta$ within numerical precision, consistent with the derived loss comparison. This inequality serves as a consistency check, whereas the diagnostic evidence comes from the observed association between D and CUL.

Roles of mean features and connections between training units. Table 3 demonstrates that mean features and cross-unit connections preserve different aspects of the retained training problem. Replacing the mean feature with a single member representative increases CUL from 0.047 to 0.054 on Cora and from 0.062 to 0.066 on Physics, together with higher input and prediction distortion. Removing cross-unit edges leaves the input distortion unchanged but increases CUL from 0.047 to 0.096 on Cora and from 0.062 to 0.127 on Physics. These results indicate that mean features preserve node-level information, whereas cross-unit connections preserve information transmitted during graph propagation; both contribute to the smaller retained utility gap of ReCoGU.

6 Conclusion

In this paper, we propose ReCoGU, a retention-aware graph unlearning framework that updates a compact coarsened training graph and retrains from scratch after node removal. We derive a retained-risk bound that separates prediction distortion, supervision-weight correction, and optimization error. Experiments show that ReCoGU reduces unlearning cost while preserving retained-data utility, and that distortion provides a useful diagnostic of collateral utility loss.

AI USE STATEMENT

In this work, we used generative AI tools only for language-related tasks for which disclosure is recommended. Specifically, these tools were used to improve the grammar, wording, and readability of text originally drafted by the authors. All AI-assisted revisions were carefully reviewed by the authors to ensure that they accurately reflect our methods, claims, and results. The authors take full responsibility for the final content of this work, including any text or other artifacts produced with the assistance of generative AI.

REFERENCES

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022. doi: 10.1145/3548606.3559352.

Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. Gnndelete: A general strategy for unlearning in graph neural networks. *arXiv preprint arXiv:2302.13406*, 2023.

Weilin Cong and Mehrdad Mahdavi. Efficiently forgetting what you have learned in graph representation learning via projection. In *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 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 *Proc. 30th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, pp. 621–630, 2024.

Bowen Fan, Yuming Ai, Xunkai Li, Zhilin Guo, LEI ZHU, Guang Zeng, Rong-Hua Li, and Guoren Wang. Opengu: A comprehensive benchmark for graph unlearning. *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 38, 2026.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 70, pp. 1263–1272. PMLR, 2017.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y. Zou. Making AI forget you: Data deletion in machine learning. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 32, 2019.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proc. Int. Conf. Mach. Learn. (ICML)*, volume 119, pp. 3832–3842, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 30:1024–1034, 2017.

Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B. Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *arXiv preprint arXiv:2402.03358*, 2024.

Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021.

Antonin Joly and Nicolas Keriven. Graph coarsening with message-passing guarantees. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-3648.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. Amer. Math. Soc., Providence, RI, 2 edition, 2017. ISBN 9781470429621. doi: 10.1090/mbk/107. With contributions by Elizabeth L. Wilmer.

Fan Li, Xiaoyang Wang, Dawei Cheng, Wenjie Zhang, Chen Chen, Ying Zhang, and Xuemin Lin. TCGU: Data-centric graph unlearning based on transferable condensation. *IEEE Trans. Knowl. Data Eng.*, 38(2):1334–1348, 2026.

Xunkai Li, Yulin Zhao, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. Towards effective and general graph unlearning via mutual evolution. In *Proc. AAAI Conf. Artif. Intell.*, volume 38, pp. 13682–13690, 2024.

Yi Li, Shichao Zhang, Guixian Zhang, and Debo Cheng. Community-centric graph unlearning. In *Proc. AAAI Conf. Artif. Intell.*, volume 39, pp. 18548–18556, 2025. doi: 10.1609/aaai.v39i17.34041.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *J. Mach. Learn. Res.*, 20(116): 1–42, 2019.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 32, 2019.

Data Protection. General data protection regulation. *Intersoft Consulting, Accessed in October*, 24 (1), 2018.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 34, pp. 18075–18086, 2021.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge Univ. Press, 2014. ISBN 9781107057135. doi: 10.1017/CBO9781107298019.

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

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proc. IEEE Symp. Secur. Privacy (SP)*, pp. 3–18, 2017.

Yash Sinha, Murari Mandal, and Mohan Kankanhalli. Distill to delete: Unlearning in graph networks with knowledge distillation. *IEEE Trans. Neural Netw. Learn. Syst.*, 2025.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 31, 2018.

Cheng-Long Wang, Mengdi Huai, and Di Wang. Inductive graph unlearning. In *Proc. 32nd USENIX Secur. Symp.*, pp. 3205–3222, 2023.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. GIF: A general graph unlearning strategy via influence function. In *Proc. ACM Web Conf. (WWW)*, pp. 651–661, 2023a.

Kun Wu, Jie Shen, Yue Ning, Ting Wang, and Wendy Hui Wang. Certified edge unlearning for graph neural networks. In *Proc. 29th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, pp. 2606–2617, 2023b.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1): 4–24, 2020.

Shuyin Xia, Xinjun Ma, Zhiyuan Liu, Cheng Liu, Sen Zhao, and Guoyin Wang. Graph coarsening via supervised granular-ball for scalable graph neural network training. In *Proc. AAAI Conf. Artif. Intell.*, volume 39, pp. 12872–12880, 2025. doi: 10.1609/aaai.v39i12.33404.

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 *Proc. AAAI Conf. Artif. Intell.*, volume 39, pp. 13044–13051, 2025. doi: 10.1609/aaai.v39i12.33423.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 40–48. PMLR, 2016.

Lu Yi and Zhewei Wei. Scalable and certifiable graph unlearning: Overcoming the approximation error barrier. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, pp. 98452–98489, 2025.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge Univ. Press, 2023.

Yi Zhang, Yuying Zhao, Zhaoqing Li, Xueqi Cheng, Yu Wang, Olivera Kotevska, Philip S Yu, and Tyler Derr. A survey on privacy in graph neural networks: Attacks, preservation, and applications. *IEEE Trans. Knowl. Data Eng.*, 36(12):7497–7515, 2024.

A Related Work

A.1 Graph Unlearning

Graph unlearning aims to remove the influence of requested graph elements while preserving model utility on the retained graph (Fan et al., 2026). Existing methods reduce unlearning cost via various paradigms. Shard-based methods, such as GraphEraser (Chen et al., 2022) and GUIDE (Wang et al., 2023), retrain only affected subgraphs; however, their efficiency relies on cutting cross-shard edges, which removes the original message-passing dependencies, while GUIDE can only recover them through reconstructed neighbors. Parameter update methods, including Projector (Cong & Mahdavi, 2023), GIF (Wu et al., 2023a), GNNDelete (Cheng et al., 2023), MEGU (Li et al., 2024), ETR (Yang et al., 2025), and D2DGN (Sinha et al., 2025), estimate or edit the effect of unlearning directly in a model trained on the requested data; these updates may leave residual influence from the requested data or alter parameters needed by retained nodes. A complementary line studies certified graph unlearning under explicit model or approximation assumptions, including CEU (Wu et al., 2023b), IDEA (Dong et al., 2024), and ScaleGUN (Yi & Wei, 2025); however, their certification guarantees rely on restrictive optimization assumptions, limiting their applicability to general nonlinear GNN training. Graph reduction takes a different route: it reduces the size of the training problem itself. A compact surrogate graph can preserve interactions between training units while supporting cold-start retraining of a single GNN after an unlearning request. This makes the deviation from full retraining an explicit property of the surrogate training graph, rather than of shard aggregation or parameter editing. Existing graph-reduction methods, such as CGE (Li et al., 2025) and TCGU (Li et al., 2026), demonstrate that a smaller training graph can substantially reduce retraining cost. However, the resulting surrogate inevitably changes the features, structural dependencies, and training weights available to the model. How these changes translate into additional utility loss relative to full retraining on the retained graph remains insufficiently characterized. ReCoGU addresses this issue by constructing a label-consistent coarsened training graph that supports efficient request updates and by explicitly analyzing the utility gap induced by this surrogate relative to full retraining.

A.2 Graph Coarsening

Graph coarsening provides a natural way to reduce the retraining cost of graph unlearning by replacing the retained graph with a smaller training surrogate. Existing studies mainly seek compact graph representations while preserving important information (Hashemi et al., 2024). Loukas (2019) establishes spectral and cut guarantees through restricted spectral approximation; Huang et al. (2021) applies coarsening to scalable GNN training and analyzes its learning effects; Xia et al. (2025) further incorporates label information to construct supervised supernodes. More recent work studies how coarsening preserves message-passing behavior on the reduced graph (Joly & Keriven, 2024). However, although graph coarsening provides useful inspiration for efficient graph unlearning, the two tasks pursue different objectives. Graph coarsening mainly seeks a compact graph that preserves selected graph properties, whereas graph unlearning requires the model trained on the reduced retained graph to remain as close as possible to the model obtained by retraining from scratch on the retained graph. ReCoGU therefore tailors graph coarsening to the objective of graph unlearning to reduce this discrepancy while retaining the efficiency benefit of graph coarsening.

B Partition and Surrogate Construction Details

This appendix gives the complete construction that is summarized in Section 3.2. It specifies how the initial units are refined to strict label purity and how small units are merged while preserving the invariant used by the unlearning update.

B.1 Label-guided refinement and same-label merging

Following supervised graph coarsening, we set the initial number of centers to

$$\alpha = \lfloor \sqrt{|V_{\mathrm{tr}}|} \rfloor$$

and distribute these centers approximately evenly among the K_y training classes, selecting high-degree nodes within each class. Nodes are assigned to the nearest center according to shortest-path distance in G_{tr}.

For a mixed unit C, let ℓ_1 and ℓ_2 be its two most frequent labels. We select the highest-degree node from each class as splitting seeds:

$$c_r = \arg\max_{v \in C:\, y_v = \ell_r} \deg_{G_{\mathrm{tr}}[C]}(v), \qquad r \in \{1, 2\}. \tag{16}$$

Each remaining node is assigned to the nearer seed according to the shortest path distance in $G_{\mathrm{tr}}[C]$:

$$C_1 = \{v \in C : d_C(v, c_1) \le d_C(v, c_2)\}, \qquad C_2 = C \setminus C_1, \tag{17}$$

where $d_C(\cdot, \cdot)$ denotes the shortest-path distance in the induced subgraph. The refinement is repeated until every unit is strictly label-pure. For a unit C_i, its purity is

$$T_i = \frac{\max_{k \in \{1,\dots,K_y\}} \sum_{v \in C_i} \mathbb{I}[y_v = k]}{|C_i|}, \qquad T_i = 1. \tag{18}$$

Thus, every member of a unit has the same supervision label.

Strict purity can create small units. To recover compactness, assume that each node is independently included in an unlearning request with probability $q \in (0, 1)$. A unit of size n_i is completely removed with probability q^{n_i}. Requiring this probability to be at most $\delta \in (0, 1)$ gives

$$q^{n_i} \le \delta \quad \Longrightarrow \quad n_i \ge \left\lceil \frac{\log \delta}{\log q} \right\rceil \equiv \tau. \tag{19}$$

For each unit satisfying $|C_i| \le \tau$, ReCoGU selects, when available, the same-label neighboring unit with the largest number of original edges:

$$j^* = \arg\max_{\substack{j:\, y(C_j) = y(C_i), \\ E(C_i, C_j) \neq \emptyset}} |E(C_i, C_j)|, \tag{20}$$

where $E(C_i, C_j) = \{(u, v) \in E_{\mathrm{tr}} : u \in C_i, v \in C_j\}$. If no eligible neighbor exists, C_i is retained. Because every merge joins units with the same label, $T_i = 1$ remains valid. After a request, $C_i' = C_i \setminus V_u$ is therefore either empty or still pure, which is the closure property used by the fixed mapping in the main text.

C Matrix-level Surrogate Construction

This section supplies the matrix definitions behind the compact description in Section 3.3. Let $n_{\mathrm{tr}} = |V_{\mathrm{tr}}|$, $\mathcal{C} = \{C_1, \dots, C_B\}$, and $G_{\mathrm{tr}} = (V_{\mathrm{tr}}, E_{\mathrm{tr}}, X_{\mathrm{tr}}, Y_{\mathrm{tr}})$. We use the fixed unit map $\pi(v) = i$ for $v \in C_i$ and define the assignment matrix $\widehat{P} \in \{0, 1\}^{n_{\mathrm{tr}} \times B}$ by

$$\widehat{P}_{vi} = \begin{cases} 1, & \pi(v) = i, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Each row has one nonzero entry. With $n_i = |C_i|$, the size matrix is

$$\mathbf{C} = \mathrm{diag}(n_1, \dots, n_B) = \widehat{P}^\top \widehat{P}. \tag{22}$$

For $X_{\mathrm{tr}} \in \mathbb{R}^{n_{\mathrm{tr}} \times d}$, mean pooling gives

$$\widehat{X} = \mathbf{C}^{-1} \widehat{P}^\top X_{\mathrm{tr}}, \qquad \widehat{x}_i = \frac{1}{n_i} \sum_{v \in C_i} x_v. \tag{23}$$

The least-squares property of the mean gives the smallest within-unit squared feature error for the fixed partition. Since each unit is pure,

$$\widehat{y}_i = y_v, \qquad \forall v \in C_i. \tag{24}$$

Let A_{tr} be the binary training adjacency. Edge aggregation gives

$$M = \widehat{P}^\top A_{\mathrm{tr}} \widehat{P}, \qquad m_{ij} = \sum_{u \in C_i} \sum_{v \in C_j} A_{\mathrm{tr}}(u, v), \quad i \neq j, \tag{25}$$

and the surrogate adjacency is

$$\widehat{A}_{ij} = \begin{cases} 1, & i \neq j \text{ and } m_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

The resulting graph $\widehat{G} = (\widehat{V}, \widehat{E}, \widehat{X})$ retains the unit labels and binary cross-unit connectivity. ReCoGU caches $S = \widehat{P}^\top X_{\mathrm{tr}}$, the sizes n_i, and the counts m_{ij}. These cached quantities are sufficient for the local update in the next section and reproduce the same means and binary adjacency as recomputation from the retained nodes under the fixed partition.

D Update Complexity and Reindexing Details

This section records the implementation boundary for the local update summarized in Section 3.4. For an unlearning request V_u, an affected unit has

$$s_i' = s_i - \sum_{v \in C_i \cap V_u} x_v, \qquad n_i' = n_i - |C_i \cap V_u|,$$

and an inter-unit edge count is updated by

$$m_{ij}' = m_{ij} - d_{ij}(V_u), \qquad \widehat{A}_{ij}' = \mathbb{I}[m_{ij}' > 0].$$

Here, $d_{ij}(V_u)$ counts original undirected edges between C_i and C_j with at least one endpoint in V_u, counting each edge once. Empty units and zero-count edges are removed, and surviving units are reindexed before the square-root weighting is applied. With adjacency lists and indexed unit statistics, the expected cost of updating the stored statistics is

$$T_{\mathrm{update}} = O\left(|V_u| d + \sum_{v \in V_u} \deg_{G_{\mathrm{tr}}}(v) \right), \tag{27}$$

where d is the feature dimension and degrees are measured before the request. The total unlearning time also includes updated graph-tensor construction and cold-start retraining on $\widehat{G}'$. The bound explains why the update phase does not scan all retained members of an affected unit.

E Detailed Experimental Protocol

This appendix gives the settings needed to reproduce the experiments summarized in Section 5.1. Cora, CiteSeer, and PubMed are citation networks, while CS and Physics are coauthorship networks. We use GCN, GAT, and GraphSAGE backbones and compare GUIDE, CGE, ETR, TCGU, GNNDelete, and Retrain. The reported metrics are Micro-F1, unlearning time, membership-inference AUC, CUL, prediction distortion, and input-feature distortion. Micro-F1 is computed on the test nodes. Unlearning time includes processing the request and obtaining the updated model. AUC measures the attack's ability to distinguish requested training nodes from nonmembers, with 0.5 corresponding to random guessing. CUL is the difference between ReCoGU and Retrain in mean cross-entropy on the same retained training graph, excluding regularization. Prediction distortion $\Delta(\theta)$ is the mean squared logit distance between each retained node and its corresponding unit at fixed parameters. The quantity plotted in Fig. 2 is the combined term $D = \beta[\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)}]$ defined in Eq. equation 14. The input-feature distortion e_0 is the corresponding root-mean-square feature distance.

The implementation uses PyTorch and PyTorch Geometric (Paszke et al., 2019; Fey & Lenssen, 2019) on a machine equipped with two NVIDIA RTX 5000 GPUs. Each backbone has two layers and hidden dimension 64. We use Adam with learning rate 0.01. Nodes are randomly divided into training, validation, and test sets in proportions 80%, 10%, and 10%. The default request removes 1% of training nodes, sampled without replacement, and all methods use the same splits and requested nodes. Test utility is evaluated after the requested nodes and their incident edges are removed from the node-level graph. Utility, efficiency, and membership-inference comparisons use 20 independent runs with different requests and random seeds.

For each request ratio, ReCoGU enforces $T_i = 1$ and sets $\tau = \lceil \log \delta / \log q \rceil$ with $\delta = 0.01$. This gives $\tau = 1$ for the default 1% request and $\tau = 2, 2, 3$ for 5%, 10%, and 20% requests, respectively. Assignments remain unchanged for retained nodes at a given ratio; each unit uses the mean feature of its retained members, binary superedges, and a standard self-loop. The normalized loss weight is $a_i = \sqrt{n'_i} / \sum_j \sqrt{n'_j}$. ReCoGU retrains from scratch on the updated surrogate, whereas Retrain uses the retained training graph.

Unlearning time includes data updates, graph-tensor construction, and model updates. For TCGU, graph transfer is included. GPU operations are synchronized before and after timing, while initial preprocessing is recorded separately. The prediction-distortion experiment uses GCN on Cora and Physics, with ratios 5%, 10%, and 20%; each ratio uses five independently sampled node sets and a corresponding initialization seed. ReCoGU and Retrain use identical initial parameters, and requests are applied independently to the original training graph. The feature and superedge ablations reuse the five 10% requests and their paired models; unit assignments, labels, weights, architecture, and initialization seeds remain fixed across variants.

F Proof and Mechanistic Interpretation of the Collateral Utility Bound

This appendix connects the two training objectives to the retained empirical risk comparison in Theorem 1. We first establish the comparison at fixed parameters, then explain how feature compression, graph propagation, and supervision weighting enter the bound.

F.1 Comparison of Training Objectives and Proof of Theorem 1

To compare the objectives on a common supervision basis, we assign each surrogate prediction the total weight of its represented nodes. Write $\ell_i(\theta) = \ell(f_\theta(\widehat{G}')_i, \widehat{y}'_i)$ for the classification loss of unit i. The auxiliary empirical risk $\mathcal{R}'_{\mathrm{sw}}(\theta)$ averages these losses using the node-proportional weights $p_i = n'_i / n'_{\mathrm{tr}}$:

$$\mathcal{R}'_{\mathrm{sw}}(\theta) = \sum_{i=1}^{B'} p_i \ell_i(\theta) = \frac{1}{n'_{\mathrm{tr}}} \sum_{v \in V'_{\mathrm{tr}}} \ell\big(f_\theta(\widehat{G}')_{\pi(v)}, y_v\big). \tag{28}$$

The second equality follows from the partition and the shared label $\widehat{y}'_{\pi(v)} = y_v$. Each unit prediction is evaluated once for every retained node it represents. Thus, $\mathcal{R}'_{\mathrm{sw}}$ and $\mathcal{R}'_{\mathrm{tr}}$ use the same node weights and labels, allowing their difference to be controlled by the change in predictions.

For fixed parameters θ, the β-Lipschitz property of the loss bounds each node's loss difference by its logit distance. Averaging over retained nodes and applying the triangle inequality and Cauchy–Schwarz gives

$$\begin{aligned} |\mathcal{R}'_{\mathrm{tr}}(\theta) - \mathcal{R}'_{\mathrm{sw}}(\theta)| &\le \frac{\beta}{n'_{\mathrm{tr}}} \sum_{v \in V'_{\mathrm{tr}}} \Big\| f_\theta(G'_{\mathrm{tr}})_v - f_\theta(\widehat{G}')_{\pi(v)} \Big\|_2 \\ &\le \beta \sqrt{\Delta(\theta)}. \end{aligned} \tag{29}$$

The last step uses the fact that the average distance is bounded by its root mean square. This inequality connects prediction distortion to the difference in average classification loss at the same parameters.

We next express the auxiliary risk through the actual surrogate objective. The definition of the weight correction gives the exact identity

$$\mathcal{R}'_{\mathrm{sw}}(\theta) = \mathcal{L}(\theta) + \mathcal{C}_{\mathrm{w}}(\theta). \tag{30}$$

Here, $a_i = \omega'_i / B'$ already includes averaging over the B' units, so $\mathcal{L}$ retains the normalization used in Section 3. Together, these two relations compare the retained risk with the surrogate objective through prediction distortion and supervision reweighting.

Proof of Theorem 1. We apply the preceding relations at each trained model and connect the resulting surrogate losses through the attained optimization quality. At the ReCoGU parameters, Eqs. equation 29 and equation 30 give

$$\mathcal{R}'_{\mathrm{tr}}(\theta'_{\mathrm{ReCoGU}}) \le \mathcal{L}(\theta'_{\mathrm{ReCoGU}}) + \mathcal{C}_{\mathrm{w}}(\theta'_{\mathrm{ReCoGU}}) + \beta\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})}.$$

The parameter vector θ'_r belongs to the shared space Θ and is therefore an admissible candidate for $\mathcal{L}$. Its surrogate loss is at least $\inf_{\theta\in\Theta}\mathcal{L}(\theta)$. The definition of $\varepsilon_{\mathrm{opt}}$ consequently yields

$$\mathcal{L}(\theta'_{\mathrm{ReCoGU}}) \le \mathcal{L}(\theta'_r) + \varepsilon_{\mathrm{opt}}.$$

At the full-retraining parameters, the same fixed-parameter comparison gives

$$\mathcal{L}(\theta'_r) \le \mathcal{R}'_{\mathrm{tr}}(\theta'_r) - \mathcal{C}_{\mathrm{w}}(\theta'_r) + \beta\sqrt{\Delta(\theta'_r)}.$$

Combining these inequalities and subtracting $\mathcal{R}'_{\mathrm{tr}}(\theta'_r)$ gives Eq. equation 13. The two distortion terms arise from evaluating the graph comparison at the two parameter vectors; the remaining terms account for supervision weighting and the attained surrogate loss. □

To verify the loss condition used above, consider standard multiclass cross-entropy on logits $z \in \mathbb{R}^{K_y}$, where K_y is the number of classes. Its loss and gradient are (Zhang et al., 2023)

$$\ell(z,y) = \log\left(\sum_{k=1}^{K_y} e^{z_k}\right) - z_y, \qquad \nabla_z \ell(z,y) = \mathrm{softmax}(z) - e_y,$$

where e_y has entry one at class y and zero elsewhere. The probability vector $s = \mathrm{softmax}(z)$ has nonnegative entries summing to one, so $\sum_{k\neq y} s_k = 1 - s_y$. It follows that

$$\|s - e_y\|_2^2 = (1-s_y)^2 + \sum_{k\neq y} s_k^2 \le 2(1-s_y)^2 \le 2.$$

The gradient norm is therefore at most $\sqrt{2}$. Integrating along the segment between any two finite logit vectors establishes the $\sqrt{2}$-Lipschitz property for a fixed label. Thus, standard cross-entropy supplies the loss condition for any GNN producing finite logits. Its nonnegativity also makes the surrogate infimum finite whenever $\mathcal{L}$ has a finite value at some admissible parameter vector.

F.2 Feature Compression and Nonlinear Message Passing

The prediction distortion in the main bound reflects both the compression of node features and the change in graph propagation. To identify these contributions, we compare the two forward computations layer by layer. The analysis follows the message-passing perspective on graph coarsening (Joly & Keriven, 2024), using ReCoGU's binary adjacency and standard self-loops in the actual nonlinear layer maps.

Let X'_{tr} and A'_{tr} be the feature and binary adjacency matrices of G'_{tr}. Write $\widehat{X}'$ and $\widehat{A}'$ for the corresponding matrices of the updated surrogate. The assignment matrix $\widehat{P}' \in \{0,1\}^{n'_{\mathrm{tr}}\times B'}$, with $\widehat{P}'_{vi} = \mathbb{I}[\pi(v) = i]$, records each retained node's unit; $\mathbb{I}$ is the indicator function. Multiplication by $\widehat{P}'$ copies each unit representation to all its member nodes. The size matrix $\mathbf{C}'$ and member-mean feature matrix satisfy

$$\mathbf{C}' = \widehat{P}'^{\top}\widehat{P}' = \mathrm{diag}(n'_1,\dots,n'_{B'}), \qquad \widehat{X}' = (\mathbf{C}')^{-1}\widehat{P}'^{\top}X'_{\mathrm{tr}}.$$

Thus, $\mathbf{C}'$ provides the member counts used for mean pooling. Propagation on both graphs uses the respective binary adjacency with standard self-loops I.

Following the message and update functions of the message-passing framework (Gilmer et al., 2017), fix θ and write the two forward computations using L layer maps. Here, $H^{(k)}$ and $\widehat{H}^{(k)}$ collect the representations after layer k on the retained and surrogate graphs, respectively:

$$\begin{aligned} H^{(0)} &= X'_{\mathrm{tr}}, & H^{(k)} &= \Phi^{(k)}_{\theta,A'_{\mathrm{tr}}}(H^{(k-1)}),\\ \widehat{H}^{(0)} &= \widehat{X}', & \widehat{H}^{(k)} &= \Phi^{(k)}_{\theta,\widehat{A}'}(\widehat{H}^{(k-1)}). \end{aligned} \tag{31}$$

Each $\Phi^{(k)}$ includes the layer's aggregation, normalization, activation, and self-loop rule, including aggregation that depends on hidden features. The final map produces nodewise logits, giving $H^{(L)} = f_\theta(G'_{\mathrm{tr}})$ and $\widehat{H}^{(L)} = f_\theta(\widehat{G}')$.

To bound how an existing representation difference passes through a layer, suppose its retained-graph map is $L_k(\theta)$-Lipschitz on the relevant input region. For any matrices U, V in this region,

$$\left\| \Phi^{(k)}_{\theta, A'_{\mathrm{tr}}}(U) - \Phi^{(k)}_{\theta, A'_{\mathrm{tr}}}(V) \right\|_F \le L_k(\theta) \|U - V\|_F. \tag{32}$$

The Frobenius norm $\|\cdot\|_F$ is the square root of the sum of squared matrix entries. The region contains $H^{(k-1)}$ and $\widehat{P}'\widehat{H}^{(k-1)}$, and $L_k(\theta)$ measures the layer's sensitivity to changes in these inputs. Such finite constants exist when the maps are locally Lipschitz on a neighborhood of a compact convex region containing the relevant inputs and contained in their domain. Each constant may depend on the graph, parameters, and backbone. These layer conditions support the mechanistic interpretation below; the main theorem uses the directly evaluated $\Delta(\theta)$ and the loss condition established above (Virmaux & Scaman, 2018).

We distinguish the total representation difference from the contribution of the propagation rule. The layer error $e_k(\theta)$ compares the two forward computations after copying unit representations to their members. The propagation residual $r_k(\theta)$ compares one layer on the two graphs with inputs aligned by that same assignment:

$$\begin{aligned} e_k(\theta) &= \frac{\|H^{(k)} - \widehat{P}'\widehat{H}^{(k)}\|_F}{\sqrt{n'_{\mathrm{tr}}}}, \\ r_k(\theta) &= \frac{1}{\sqrt{n'_{\mathrm{tr}}}} \left\| \Phi^{(k)}_{\theta, A'_{\mathrm{tr}}}(\widehat{P}'\widehat{H}^{(k-1)}) - \widehat{P}'\Phi^{(k)}_{\theta, \widehat{A}'}(\widehat{H}^{(k-1)}) \right\|_F. \end{aligned} \tag{33}$$

The normalization expresses each quantity as a root mean square of nodewise vector distances. The residual includes the effects of collapsed within-unit edges, binary cross-unit edges, and the resulting aggregation coefficients.

To separate these contributions, insert the intermediate output $\Phi^{(k)}_{\theta, A'_{\mathrm{tr}}}(\widehat{P}'\widehat{H}^{(k-1)})$ into the difference defining e_k. The triangle inequality and Eq. equation 32 yield

$$e_k(\theta) \le L_k(\theta) e_{k-1}(\theta) + r_k(\theta).$$

The first term carries the previous representation difference through the retained-graph layer; the second adds the discrepancy from propagation on the surrogate. Starting from the initial feature difference e_0 and applying this relation over all L layers gives

$$\sqrt{\Delta(\theta)} \le \left(\prod_{k=1}^{L} L_k(\theta) \right) e_0 + \sum_{k=1}^{L} \left(\prod_{j=k+1}^{L} L_j(\theta) \right) r_k(\theta), \tag{34}$$

where $e_L(\theta) = \sqrt{\Delta(\theta)}$, an empty product equals one, and

$$e_0^2 = \frac{1}{n'_{\mathrm{tr}}} \sum_{i=1}^{B'} \sum_{v \in C'_i} \|x_v - \widehat{x}'_i\|_2^2. \tag{35}$$

Here, x_v is node v's feature vector and $\widehat{x}'_i$ is its unit's member mean. The products of layer constants describe how each contribution passes through subsequent nonlinear layers.

Member means have a precise role in the initial term. For each fixed surviving unit, the least-squares property of the mean (Bishop, 2006) gives, for any representative vector b_i,

$$\sum_{v \in C'_i} \|x_v - b_i\|_2^2 = \sum_{v \in C'_i} \|x_v - \widehat{x}'_i\|_2^2 + n'_i \|b_i - \widehat{x}'_i\|_2^2. \tag{36}$$

The cross term vanishes because $\sum_{v \in C'_i}(x_v - \widehat{x}'_i) = 0$, and the final term is nonnegative. Thus, the member means minimize e_0 for the fixed partition. Equation equation 34 connects this initial

optimality to the prediction bound while retaining the propagation residuals and their subsequent layer dependence.

Local updates recover the sums $s'_i = \sum_{v \in C'_i} x_v$, the counts $n'_i = |C'_i|$, and the surviving inter-unit edge counts. They therefore produce the same means and binary adjacency as recomputation from the retained nodes under the fixed partition. Together, these results identify the feature and propagation contributions to $\Delta(\theta)$ and keep them aligned with the updated training problem.

F.3 Supervision Weights and Evaluation of the Bound

The preceding analysis connects the surrogate's predictions to retained empirical risk. We now characterize the supervision-weight contribution and express the comparison using the losses of the two trained models. Let $p = (p_i)_{i=1}^{B'}$ and $a = (a_i)_{i=1}^{B'}$ be the normalized node-proportional and square-root weights. Their total variation, $\mathrm{TV}(p, a) = \frac{1}{2}\sum_i |p_i - a_i|$, measures the difference in supervision allocation. Its bounded-function characterization (Levin & Peres, 2017) yields

$$|\mathcal{C}_{\mathrm{w}}(\theta)| \le \mathrm{TV}(p, a)\left(\max_i \ell_i(\theta) - \min_i \ell_i(\theta)\right). \tag{37}$$

To see this, subtract $\min_i \ell_i(\theta)$ from each loss in Eq. equation 12; the total correction is unchanged because $\sum_i (p_i - a_i) = 0$. The positive and negative parts of $p - a$ each sum to $\mathrm{TV}(p, a)$, and each centered loss lies between zero and the displayed loss range. This bounds the correction in both directions.

The correction is therefore controlled jointly by the weight allocation and the variation in unit losses at the evaluated parameters. Equal unit sizes, or weights $a_i = p_i$, make the correction zero. For the chosen square-root weights, its signed value records the exact change from the surrogate objective to the node-proportional average loss.

The main theorem expresses optimization quality through $\varepsilon_{\mathrm{opt}}$, which compares the attained ReCoGU surrogate loss with the lower limit of $\mathcal{L}$ over Θ. For evaluation, recall the directly computable quantities D, W, and η from Eq. equation 14. Applying the same comparison argument directly to the two attained parameter sets gives

$$\begin{aligned}\mathrm{CUL}(V_u) \le \beta\left(\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)}\right) \\ + \mathcal{C}_{\mathrm{w}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{C}_{\mathrm{w}}(\theta'_r) + \eta,\end{aligned} \tag{38}$$

which is Eq. equation 15 written in expanded form.

A positive η means that ReCoGU has the higher attained surrogate loss, whereas a negative value means that its surrogate loss is lower. Moreover, $\eta \le \varepsilon_{\mathrm{opt}}$ because $\mathcal{L}(\theta'_r) \ge \inf_{\theta \in \Theta} \mathcal{L}(\theta)$. Thus, η compares two attained surrogate losses, whereas $\varepsilon_{\mathrm{opt}}$ measures the gap between ReCoGU's attained surrogate loss and the lower limit over all admissible parameters.

By Eq. equation 30, the difference between the two weight corrections, together with η, equals $\mathcal{R}'_{\mathrm{sw}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{R}'_{\mathrm{sw}}(\theta'_r)$. Equation equation 38 consequently expresses a comparison of the two trained models using their directly evaluated predictions and losses.

For each request, evaluate both parameter sets on G'_{tr} and $\widehat{G}'$ with deterministic forward passes and the same unregularized classification loss. These four evaluations provide both distortions, both weight corrections, η, and the measured CUL. When training includes regularization or early stopping, $\varepsilon_{\mathrm{opt}}$ remains defined for the classification objective $\mathcal{L}$; it is determined by the attained parameters.

These evaluations make every term in Eq. equation 38 directly computable. The comparison in Eq. equation 38 checks the derived loss relation at the trained parameters. Mechanism-level evaluation can then examine the association of feature distortion, propagation residuals, and supervision reweighting with CUL under controlled requests and compression budgets. These empirical comparisons are conducted within each backbone, with Micro-F1 reported. The theoretical conclusion remains the bound on additional retained training loss established in Theorem 1.

F.4 TIGHTNESS, SCOPE, AND PRACTICAL INTERPRETATION

The main theorem follows from the fixed-parameter comparison

$$|\mathcal{R}'_{\mathrm{tr}}(\theta) - \mathcal{R}'_{\mathrm{sw}}(\theta)| \le \beta\sqrt{\Delta(\theta)}. \tag{39}$$

The inequality uses only the β-Lipschitz property of the loss and the root-mean-square bound on the nodewise logit distances. Its constant is worst-case sharp under these assumptions. In particular, the bound is approached when all retained nodes have the same logit distance and the corresponding loss changes point in the same maximal-slope direction. For multiclass cross-entropy, the gradient norm is bounded by $\sqrt{2}$, and this value is approached when the predicted probability concentrates on an incorrect class. Therefore, a smaller universal coefficient cannot be guaranteed without imposing additional restrictions on the logits or the loss.

The supervision-weight term is an exact correction rather than an additional upper-bound approximation. By Eq. equation 30,

$$\mathcal{R}'_{\mathrm{sw}}(\theta) = \mathcal{L}(\theta) + \mathcal{C}_{\mathrm{w}}(\theta). \tag{40}$$

Likewise,

$$\varepsilon_{\mathrm{opt}} = \mathcal{L}(\theta'_{\mathrm{ReCoGU}}) - \inf_{\theta\in\Theta} \mathcal{L}(\theta) \tag{41}$$

is exact by definition and vanishes when ReCoGU reaches the surrogate infimum. The only remaining slack in Theorem 1 comes from the two fixed-parameter transfer inequalities and from replacing the surrogate optimum by the attained optimization error. Hence the theorem should be interpreted as an assumption-level worst-case envelope, rather than as an equality expected for every request.

For practical diagnosis, define

$$D = \beta\left[\sqrt{\Delta(\theta'_{\mathrm{ReCoGU}})} + \sqrt{\Delta(\theta'_r)}\right], \qquad W = \mathcal{C}_{\mathrm{w}}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{C}_{\mathrm{w}}(\theta'_r),$$

and

$$\eta = \mathcal{L}(\theta'_{\mathrm{ReCoGU}}) - \mathcal{L}(\theta'_r).$$

The observable relation

$$\mathrm{CUL}(V_u) \le D + W + \eta$$

separates three intervention points. A dominant D indicates distortion from the compressed features or graph propagation, a dominant W indicates a mismatch between unit-level and node-level supervision weights, and a dominant η indicates insufficient surrogate optimization. Thus, the bound guides the design and debugging of the surrogate instead of serving only as a post-hoc theoretical statement. It does not claim that the diagnostic alone determines test-set utility or that every method is ordered by its bound.