EFFICIENT PARTITION-LOCALIZED UNLEARNING FOR GRAPH TRANSFORMER

Anonymous Authors

ABSTRACT

Graph unlearning aims to reduce the influence of designated training nodes from a deployed graph model without retraining from scratch. Most existing approaches are developed for message-passing GNNs, where forgetting is often realized by editing local neighborhoods or fine-tuning on affected neighbors. A Graph Transformer instead relies on shared attention over tokens, so those neighborhood-based deletion recipes do not directly carry over to its computation. We propose PLU-GT, a partition-localized unlearning method for a shared Graph Transformer. The method identifies patches touched by a forget request, remaps forgotten nodes to virtual nodes to block their participation in intra-patch attention, and optimizes a localized objective that combines affected-region supervision, replay, entropy maximization, and cloak regularization. Experiments on four real-world graphs show that PLU-GT provides a favorable utility–efficiency trade-off under a unified unlearning protocol. It remains competitive with full retraining in predictive utility while achieving several-fold speedups, and substantially reduces posterior-based membership leakage under the specified attacker.

1 INTRODUCTION

Graph unlearning is increasingly important for deployed graph predictors that must respond to deletion requests under privacy regulations such as the GDPR European Parliament & Council of the European Union (2016). Social platforms raise such requests when users leave a network Luo et al. (2026). Citation and collaboration graphs likewise require removing papers or entities from trained predictors Fan et al. (2025). Deleting raw records is insufficient when parameters still encode information about forgotten entities Jain et al. (2026). Graph unlearning therefore asks how to update a trained model so that designated training nodes no longer influence predictions, ideally approaching a predictor retrained on the retain set Chen et al. (2022). Most existing graph-unlearning methods are developed for message-passing GNNs Li et al. (2025a). Their neighborhood-based deletion recipes do not directly transfer to a Graph Transformer Zhuo et al. (2025). Shared parameters and patch-level attention create different paths through which forgotten nodes can affect the forward computation Liao et al. (2025).

Partitioned Graph Transformers are an attractive backbone when attention is organized over patch tokens rather than purely local neighborhoods Xing et al. (2025). Relative to deep message passing, such designs are also motivated by the need to capture longer-range interactions and to mitigate oversquashing Dwivedi et al. (2026). Yet deploying such models does not automatically solve deletion Wu & Wang (2025). Shared attention parameters, patch-level tokenization, and attention-path visibility create unlearning difficulties that GNN neighborhood recipes do not fully cover Guo et al. (2026). Even recent scalable unlearning pipelines still largely assume GNN-style computation graphs Yi & Wei (2025).

Concretely, approximate unlearning on a shared Graph Transformer faces three coupled difficulties. Parameter sharing means that updates motivated by a local forget set can change predictions outside that set Zhang et al. (2025). Under-forgetting arises when optimization only reinforces affected retain labels, so forget posteriors may remain confident and membership signals can persist Chen et al. (2025). Attention-path leakage arises when structural edits are not aligned with patch attention, so forgotten nodes may remain effectively visible to the forward pass Zhang et al. (2026). Influence-style and neighborhood-deletion pipelines designed for GNN message passing likewise need Transformer-compatible edits on patch-attention backbones Ding et al. (2025).

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.

This paper addresses the gap with a partition-localized view of Transformer unlearning Fan et al. (2025). The forget set is mapped to the patches it touches; forgotten slots are remapped to virtual nodes so they are excluded from intra-patch attention; and a pretrained Transformer is warm-started with a localized multi-term objective that couples retain utility and forget-side regularization. When the auxiliary weights are zero, the objective reduces to affected-region cross-entropy on retain nodes in the touched patches Li et al. (2024).

Concretely, PLU-GT (**P**artition-**L**ocalized **U**nlearning for a **G**raph **T**ransformer) proceeds in three steps. First, it constructs an affected retain training subset from patches that intersect the forget set, concentrating optimization where deletion is most relevant Li et al. (2025a). Second, it applies a hard virtual-node remap that removes forgotten nodes from patch attention while remaining compatible with common graph-edit protocols Cheng et al. (2023). Third, it optimizes affected-region CE, replay on non-affected retain nodes, and entropy/cloak regularizers on forget-indexed logits after the edit, selecting checkpoints by validation accuracy on the original evaluation graph Jain et al. (2026). The three components are designed to address parameter sharing, under-forgetting, and attention-path leakage, respectively.

The contributions are threefold.

- We study approximate graph unlearning for a Graph Transformer and identify three challenges that are not directly addressed by neighborhood-based GNN unlearning: shared-parameter interference, under-forgetting in the posterior space, and attention-path leakage.
- We propose PLU-GT, which combines affected-patch localization, virtual-node hard edits, and a multi-term localized objective on one shared Transformer.
- We conduct a controlled empirical evaluation against a broad suite of adapted unlearning baselines and analyze the utility, membership leakage, efficiency, and component contributions of PLU-GT.

2 Related Work

2.1 Graph Transformer

Graph Transformers replace or complement fixed neighborhood aggregation with attention over nodes or subgraph units Zhuo et al. (2025). A common motivation is to capture longer-range dependencies and to alleviate oversquashing that deep message passing can suffer from Dwivedi et al. (2026). A practical route to scalable computation is to organize nodes into patch tokens and run Transformer layers on these blocks instead of dense all-pairs attention Liao et al. (2025). Hierarchical masking further refines which tokens interact inside Transformer blocks Xing et al. (2025). In a shared-parameter design, the same encoder is applied across patches, so local token edits and parameter updates can interact through the common weights Li et al. (2025b). Data-centric graph construction pipelines likewise emphasize that learned predictors depend on how graph tokens are organized Du et al. (2025). This paper does not propose a new Transformer architecture; it studies how to unlearn on a shared, partition-based Graph Transformer backbone Fan et al. (2026).

2.2 Graph Unlearning

Graph unlearning seeks to reduce the influence of designated training data without full retain-set retraining Jain et al. (2026). Shard-and-aggregation pipelines such as GraphEraser isolate parameters across shards for exact-style localization Chen et al. (2022). Approximate methods trade guarantees for speed: GIF updates parameters via influence functions Wu et al. (2023). IDEA studies certified approximate unlearning for graph neural networks Dong et al. (2024). ScaleGUN further targets scalable and certifiable deletion under tighter budgets Yi & Wei (2025).

Another line edits local neighborhoods or fine-tunes around affected nodes, as in MEGU Li et al. (2024). GNNDelete introduces dedicated neighbor-deletion modules Cheng et al. (2023). Adaptive Graph Unlearning balances forget and retain signals in local neighborhoods Ding et al. (2025). Closely related objectives include memorization-guided forgetting in MGU Ding et al. (2026). FairGU studies fairness-aware deletion in social networks Luo et al. (2026). SGU scales approximate updates through node-influence maximization Li et al. (2025a). OpenGU organizes evaluation

around utility, efficiency, forgetting, and posterior membership probes Fan et al. (2025). Federated graph unlearning extends deletion requests to multi-client training settings Ai et al. (2026). Multimodal federated graph unlearning further studies cross-modality deletion constraints Lu et al. (2026). Federated graph learning benchmarks highlight related privacy and distribution challenges for graph models Li et al. (2025c). Dynamic graph unlearning investigates post-processing updates when graphs evolve over time Zhang et al. (2025).

Training-free parameter editing offers another cost-oriented deletion route Yang et al. (2025). Corrective unlearning targets removing adverse effects of manipulated graph entities after training Kolipaka et al. (2025). Mass-deletion regimes motivate curriculum-style forgetting objectives Zhang et al. (2026). Influence-aware preference optimization provides an alternative forget-side training view Chen et al. (2025). Adversarial verification probes whether unlearning has only been incompletely realized Wu & Wang (2025). Spatio-temporal settings explore subgraph virtual-edge reconstruction for forgetting Guo et al. (2026). Most of this literature remains GNN-centric, whereas PLUG-GT targets a partition-localized Graph Transformer, where deletion must operate on patch-level token attention rather than local neighborhoods, and uses virtual-node remapping with a retain/forget multi-term objective to handle shared-parameter updates and attention-path leakage.

3 PRELIMINARIES

3.1 NOTATIONS

We use calligraphic letters for sets, bold uppercase letters for matrices, and bold lowercase letters for vectors. An undirected attributed graph is written $\mathcal{G} = \{\mathcal{V}, \mathbf{A}, \mathbf{X}\}$, where $\mathcal{V} = \{1, \ldots, n\}$ is the node set, $\mathbf{A} \in \{0,1\}^{n\times n}$ is the adjacency matrix induced by the edge set, and $\mathbf{X} \in \mathbb{R}^{n\times d}$ collects node features. When convenient we also refer to edges through an index list compatible with common GNN edit protocols, but the partitioned Transformer used in this work primarily consumes patch tokens rather than iterative message passing over $\mathbf{A}$. Labels y_v are observed on a training split $\mathcal{V}_{\text{train}}$, with disjoint validation and test sets. A node feature vector is written $\mathbf{x}_v$, and a layer representation matrix is written $\mathbf{H}$. A Graph Transformer f_θ maps features and a patch (token) organization $\mathcal{P}$ to node logits. Training minimizes cross-entropy on $\mathcal{V}_{\text{train}}$.The graph, patch, forget-set, replay-set, and loss-weight notation used throughout the paper is summarized in Appendix A. Later method equations reuse these symbols without redefining them at each step.

3.2 GRAPH TRANSFORMER

This work considers a partitioned Graph Transformer. Nodes are grouped into patches $\mathcal{P} = \{P_1, \ldots, P_K\}$ (e.g., via METIS-style partitioning). Each patch is padded to a common length L with a virtual node $v_\emptyset$. Multi-head attention is computed inside patches with a mask that forbids attending to virtual positions. All patches share one parameter set θ, so inference is a single forward pass rather than an ensemble vote over independent shard models.

Let $\mathbf{H} \in \mathbb{R}^{L\times d}$ denote the token representation matrix inside a patch (after optional positional encodings are concatenated to features). The standard scaled-dot-product attention uses projections

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_Q, \qquad \mathbf{K} = \mathbf{H}\mathbf{W}_K, \qquad \mathbf{V} = \mathbf{H}\mathbf{W}_V, \tag{1}$$

$$\mathrm{Attn}(\mathbf{H}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_K}} + \mathbf{M}\right)\mathbf{V}, \tag{2}$$

where $\mathbf{M}$ is an additive mask that places $-\infty$ on virtual-node positions (and optionally other forbidden pairs). In the partitioned setting, attention is computed independently inside each padded patch and then reassembled into a shared node representation matrix over $\mathcal{V}$. Forgotten nodes that remain inside a patch continue to contribute keys, queries, and values unless an explicit edit removes them from the token list or the mask. This observation motivates the hard virtual-node remap introduced in Section 4.

Patch-level attention provides a broader intra-patch receptive field than purely local message passing and is useful for reducing oversquashing Zhuo et al. (2025). Patches also act as token blocks that

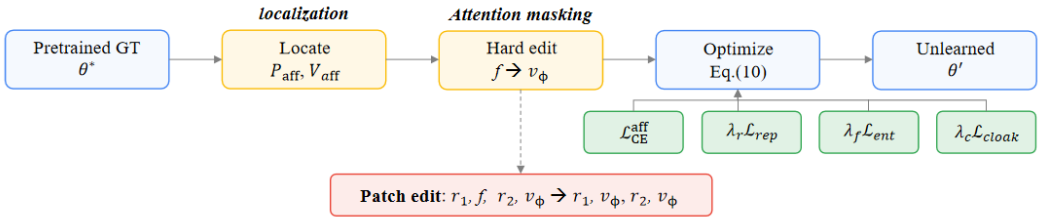

Figure 1: Overview of PLU-GT. The top row shows localization, virtual-node hard editing, and multi-term optimization on a shared Graph Transformer; the inset illustrates remapping a forgotten node f to $v_\emptyset$ inside a patch.

make the computation closer to sequence Transformers Liao et al. (2025). Studying unlearning on this backbone is therefore a step toward deletion-aware Transformer graph models. Optional positional encodings may be concatenated to features before the encoder. Structure enters primarily through the partition layout and such encodings rather than through a message-passing stack over an edge index.

3.3 Graph Unlearning

Graph unlearning asks for an updated model that behaves as if designated training data had not been used, without paying the full cost of retain-set retraining whenever possible Jain et al. (2026). In the node-deletion setting considered here, a forget set $\mathcal{V}_f \subset \mathcal{V}_{\text{train}}$ of size $\lfloor \rho |\mathcal{V}_{\text{train}}| \rfloor$ is sampled, and the retain training set is:

$$\mathcal{V}_{\text{train}}^r = \mathcal{V}_{\text{train}} \setminus \mathcal{V}_f. \tag{3}$$

Starting from a pretrained checkpoint $\theta^\star$, an approximate unlearning algorithm returns parameters θ' that should approach the Retrain solution θ^{rt} trained from scratch on $\mathcal{V}_{\text{train}}^r$ under the same edit protocol. Ideally, $f_{\theta'} \approx f_{\theta^{\text{rt}}}$, both in retain utility and in the empirical membership signal associated with $\mathcal{V}_f$.

Important concepts used throughout the paper are the forget ratio ρ, the affected patches that intersect $\mathcal{V}_f$, and the distinction between the edited graph used during unlearning optimization and the original graph used for reported utility and membership evaluation. Utility is measured by test accuracy of $f_{\theta'}$. Privacy is probed by a posterior membership-inference attack (MIA): an MLP attacker is trained on member versus non-member softmax posteriors and evaluated by attack AUC on forgotten versus test nodes Fan et al. (2025). Values near 0.5 indicate weaker membership leakage, and the paper also reports $|\text{AUC} - 0.5|$ when interpreting privacy. Efficiency is measured by wall-clock unlearning time.

4 Method: PLU-GT

PLU-GT is a partition-localized unlearning procedure for a shared Graph Transformer. Given a pretrained Transformer $\theta^\star$ and a forget request $\mathcal{V}_f$, the method first locates the affected patches and the affected retain training nodes, then hard-edits each touched patch by remapping forgotten slots to the virtual node so that forgotten nodes are excluded from intra-patch attention, and finally warm-starts from $\theta^\star$ to optimize a localized objective that mixes affected-region CE, replay, and forget-side entropy/cloak terms before returning the best validation checkpoint θ'. The design responds to three coupled difficulties: shared parameters can propagate local updates globally, forget posteriors may remain confident if the loss never regularizes $\mathcal{V}_f$, and leaving forgotten nodes inside attention can preserve structural leakage even after label removal. This sequence separates structural editing from parameter updating while preserving replay supervision outside the affected region. Figure 1 provides a visual summary of the complete pipeline.

4.1 Affected-Patch Localization

Before updating parameters, the method must first identify how the forget request intersects the Transformer computation. Fine-tuning on the full retain set would overlook the patch structure that

organizes attention. Optimizing only a forget-set CE objective would also be insufficient, because retain-training nodes in the same patches remain coupled to the edited tokens through the shared parameters. The localization step therefore identifies the patches that contain forgotten nodes and collects the retain-training nodes located in those patches. A patch is called affected ($\mathcal{P}_{\text{aff}}$) if it contains at least one forgotten node, and the affected retain-training set ($\mathcal{V}_{\text{aff}}$) consists of the retain-training nodes in the affected patches.

$$\mathcal{P}_{\text{aff}}(\mathcal{V}_f) = \{P_k \in \mathcal{P} \mid P_k \cap \mathcal{V}_f \neq \emptyset\}, \tag{4}$$

$$\mathcal{V}_{\text{aff}} = \big\{v \in \mathcal{V}^r_{\text{train}} \;\big|\; \text{patch}(v) \in \mathcal{P}_{\text{aff}}(\mathcal{V}_f)\big\}. \tag{5}$$

When partitions are fine-grained, $|\mathcal{P}_{\text{aff}}|/K$ can be much smaller than one, which concentrates supervision and helps explain empirical speedups relative to full retain retraining. If almost all patches are touched, localization remains well-defined but the efficiency margin shrinks toward ordinary fine-tuning. In either case, $\mathcal{V}_{\text{aff}}$ provides the primary retain-side supervision region used by the later objective.

4.2 Hard Editing via Virtual-Node Remapping

Localization alone does not remove forgotten nodes from attention. If a forgotten node remains as a concrete token inside its patch, it can still produce keys and values that influence retain neighbors, and its own representation can remain informative to a membership attacker. Soft objective-only unlearning without edits is supported for analysis, but it does not address attention-path leakage. We therefore apply a hard edit that is compatible with the existing virtual-node mask already used for padding. For a padded patch $P_k = [u_{k,1}, \ldots, u_{k,L}]$, hard editing replaces every forgotten slot by the virtual node:

$$u'_{k,j} = \begin{cases} v_\emptyset, & u_{k,j} \in \mathcal{V}_f, \\ u_{k,j}, & \text{otherwise.} \end{cases} \tag{6}$$

Because the Transformer mask already zeros attention involving $v_\emptyset$, the remap excludes forgotten nodes from intra-patch attention. In our experiments we also drop all incident edges of $\mathcal{V}_f$ for protocol compatibility with GNN-style edits; for the partitioned Graph Transformer, however, the operational deletion mechanism remains the slot remap rather than message-passing neighborhood updates. After editing, the forward pass uses the modified patch organization $\mathcal{P}'$ during unlearning optimization, while reported utility and MIA metrics are computed on the original evaluation graph with updated parameters (Appendix B). Forgotten nodes remain on the encoder-to-classifier pathway for forget-side regularization, although they no longer participate in intra-patch attention. Their indexed logits are still used in $\mathcal{L}_{\text{ent}}$ and $\mathcal{L}_{\text{cloak}}$.

4.3 Localized Multi-term Unlearning Objective

After the edit, a forward pass on the edited patches produces logits $z = f_\theta(\mathbf{X}, \mathcal{P}')$. Retain nodes that remain in $\mathcal{P}'$ receive ordinary patch-Transformer updates, while forget-indexed coordinates z_v ($v \in \mathcal{V}_f$) stay on the encoder-to-classifier pathway of that same forward and are not refreshed by intra-patch attention. If the loss only reinforces retain labels in $\mathcal{V}_{\text{aff}}$, two difficulties remain. First, non-affected retain regions can drift because shared parameters are updated without any replay signal. Second, forget posteriors can remain confident if $\mathcal{V}_f$ never appears in the objective after the edit. The training objective therefore needs both retain-side utility terms and forget-side regularizers. Affected CE and replay CE are:

$$\mathcal{L}^{\text{aff}}_{\text{CE}} = \frac{1}{|\mathcal{V}_{\text{aff}}|} \sum_{v \in \mathcal{V}_{\text{aff}}} \ell_{\text{CE}}(z_v, y_v), \tag{7}$$

$$\mathcal{L}^{\text{replay}}_{\text{CE}} = \frac{1}{|\mathcal{V}_{\text{rep}}|} \sum_{v \in \mathcal{V}_{\text{rep}}} \ell_{\text{CE}}(z_v, y_v), \tag{8}$$

where $\mathcal{V}_{\text{rep}}$ is a random subset of non-affected retain-train nodes with size at most N_{cap}. Letting $p_v = \text{softmax}(z_v)$ and $H(p) = -\sum_c p_c \log p_c$, entropy maximization and an optional cloak term are:

$$\mathcal{L}_{\text{ent}} = -\frac{1}{|\mathcal{V}_f|} \sum_{v \in \mathcal{V}_f} H(p_v), \qquad \mathcal{L}_{\text{cloak}} = \frac{1}{|\mathcal{V}_f|} \sum_{v \in \mathcal{V}_f} \text{KL}(q \,\|\, p_v). \tag{9}$$

The cloak reference q is a single stop-gradient distribution shared by all forget nodes. At each step we sample up to N_{ref}=256 nodes from the retain pool $\mathcal{V}_{\text{aff}} \cup \mathcal{V}_{\text{rep}}$, average their logits to $\bar{z}_{\text{ref}}$, and set $q = \text{softmax}(\bar{z}_{\text{ref}})$ with $\bar{z}_{\text{ref}}$ detached so that gradients flow only through the forget posteriors p_v. The KL is written as $\text{KL}(q\|p_v)$, matching the implementation that pulls each p_v toward this re-sampled retain reference. The full objective used by PLU-GT is:

$$\mathcal{L} = \mathcal{L}_{\text{CE}}^{\text{aff}} + \lambda_r \mathcal{L}_{\text{CE}}^{\text{replay}} + \lambda_f \mathcal{L}_{\text{ent}} + \lambda_c \mathcal{L}_{\text{cloak}}. \tag{10}$$

Setting $\lambda_r = \lambda_f = \lambda_c = 0$ yields an affected-region cross-entropy baseline that supervises only retain-train nodes in $\mathcal{V}_{\text{aff}}$. Enabling replay stabilizes non-affected regions. Entropy and cloak target under-forgetting in posteriors.

Optimization warm-starts from $\theta^\star$ and runs Adam for T epochs. At each epoch, validation accuracy is computed on the original graph tensors $(\mathbf{X}, \mathcal{P}, \mathbf{A})$, and the best checkpoint is kept. Final test utility and posterior MIA likewise use the original evaluation graph with updated parameters. Structural edits therefore participate in the unlearning optimization graph, while main reported metrics follow a fixed original-graph protocol shared with the baseline transplants (Sec. 5; Appendix B). The complete procedure is written out in Algorithm 1.

Algorithm 1 PLU-GT for partition-localized Graph Transformer unlearning

Require: Graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}, \mathbf{X}\}$, patches $\mathcal{P}$, pretrained $\theta^\star$, forget set $\mathcal{V}_f$, weights $(\lambda_r, \lambda_f, \lambda_c)$, epochs T, replay cap N_{cap}
Ensure: Unlearned parameters θ'
1: Build node→patch map; compute $\mathcal{P}_{\text{aff}}$ and $\mathcal{V}_{\text{aff}}$
2: Construct edited patches $\mathcal{P}'$ by remapping $\mathcal{V}_f$ to $v_\emptyset$; drop incident edges of $\mathcal{V}_f$ (as in our experiments)
3: Sample replay set $\mathcal{V}_{\text{rep}}$ from non-affected retain-train nodes ($\leq N_{\text{cap}}$)
4: Initialize $\theta' \leftarrow \theta^\star$ and track the best validation checkpoint on the original graph
5: **for** $t = 1$ to T **do**
6: Forward on $(\mathbf{X}, \mathcal{P}')$, evaluate Eq. (10), and update θ' with Adam
7: Score validation accuracy on original $(\mathbf{X}, \mathcal{P}, \mathbf{A})$ and keep the best checkpoint
8: **end for**
9: **return** the best checkpoint

5 Experiments

5.1 Experimental Setup

Experiments use four graphs: Pokec-z and Pokec-n (social, ρ=0.05), Cora (citation, ρ=0.10), and Coauthor-CS (coauthor, ρ=0.10). Cora uses Planetoid splits. CS uses a stratified $80/10/10$ split. All methods load the same pretrained partitioned Graph Transformer checkpoint per dataset, so the unlearning algorithm is the only experimental factor. Forget sets are sampled from $\mathcal{V}_{\text{train}}$ with seeds $\{20, 22, 23, 25, 42\}$, and mean±std is reported over these five requests.

The main tables compare PLU-GT against adapted unlearning baselines under a shared protocol. GraphEraser represents shard-localized unlearning Chen et al. (2022). IDEA provides certified approximate updates Dong et al. (2024). MEGU couples predictive and unlearning objectives Li et al. (2024). MGU uses memorization-guided forgetting Ding et al. (2026). GIF applies influence-function updates Wu et al. (2023). FairGU is evaluated in its FIM-damping stage Luo et al. (2026). SGU pursues scalable influence-maximization unlearning Li et al. (2025a). GNNDelete provides neighbor-deletion modules Cheng et al. (2023). All methods share the same pretrained checkpoint, data split, forget requests, original evaluation graph, and posterior MIA attacker. Structural edits and update rules remain method-specific and are documented in Appendix D. We adapt each baseline to the shared Graph Transformer backbone under a unified protocol. For PLU-GT, the experiments apply both virtual-node remapping and incident-edge removal, but the operational Graph Transformer deletion mechanism is the remap; edge removal aligns the edited training graph with GNN-style forget protocols used by several transplants.

Unlearning optimization uses the edited graph (virtual-node remap and, when applicable, incident-edge removal). Main-table ACC and posterior MIA are computed on the original evaluation graph

with the updated parameters; Appendix B reports the edited-graph ACC comparison. Retrain is evaluated under the same original-graph protocol and serves as a retain-supervised reference.

Reported main-table metrics are test ACC, raw MIA AUC, and wall-clock unlearning time. Privacy discussions additionally use $|\mathrm{AUC} - 0.5|$. Pokec-n uses a tuned replay/forget setting under the same localized objective. Unlearning hyperparameters and backbone settings are collected in Appendix E, where the effect of the forget ratio is studied in a separate section.

5.2 Main Utility Results

After unlearning, PLU-GT attains the best approximate ACC on Pokec-z (67.45) and Pokec-n (66.67), and remains competitive on Cora (69.76) and Coauthor-CS (94.33). On Cora, GraphEraser reaches 62.86 ACC under the same epoch budget, while PLU-GT reaches 69.76. GIF and SGU lose more accuracy on the social graphs, while GNNDelete preserves Cora ACC but drops sharply on CS. Methods are ordered consistently across tables, with PLU-GT in the last row; best and second-best marks follow the caption protocol. The full numbers are reported in Table 1.

Table 1: Test accuracy (%) after unlearning (mean±std over five forget seeds $\{20, 22, 23, 25, 42\}$). Best is **bold**; second best is underlined; PLU-GT is the last row.

Method	Pokec-z	Pokec-n	Cora	CS
GraphEraser	66.91±0.88	66.13±0.62	62.86±4.69	**94.44±0.49**
IDEA	65.59±2.56	64.97±2.82	64.68±5.54	93.74±0.53
MEGU	66.70±1.61	63.05±2.49	68.80±1.61	93.80±0.36
MGU	66.16±1.37	65.14±1.07	68.78±2.55	93.28±0.90
GIF	59.68±6.38	61.50±5.92	64.22±2.43	87.73±2.88
FairGU	66.94±1.00	65.07±2.06	**70.06±1.09**	92.37±1.84
SGU	63.05±5.26	61.15±0.50	66.44±2.16	92.13±2.90
GNNDelete	66.39±0.31	62.83±0.84	69.66±0.77	86.41±1.10
PLU-GT	**67.45±0.67**	**66.67±0.50**	69.76±0.84	94.33±0.25

On Pokec-n, PLU-GT achieves the best approximate-method accuracy (66.67) while maintaining an MIA AUC close to chance (0.497) and requiring substantially less time than full retraining (3.2 s vs. 30.6 s). The corresponding accuracy and privacy values are shown in Figure 2. Complementary readings follow from the same measurements. MEGU is close on Cora ACC (68.80) but leaves raw MIA at 0.863. MGU is among the fastest methods under the matched epoch budget, with Cora ACC 68.78. FairGU is strong on Cora ACC (70.06) yet weaker on CS ACC (92.37). GNNDelete preserves Cora ACC (69.66) but drops on CS ACC (86.41).

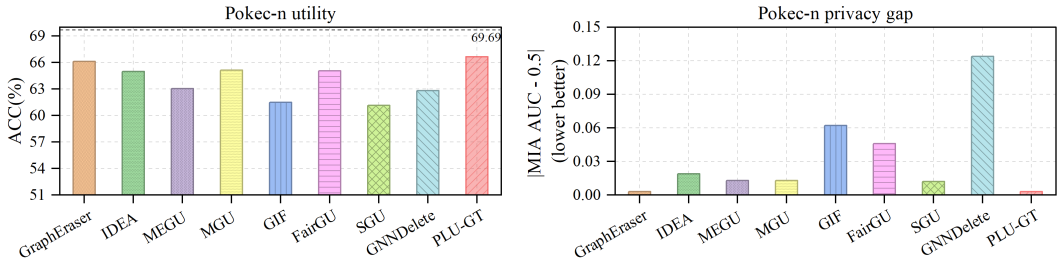

Figure 2: Pokec-n motivating comparison. Left: ACC (%) for approximate methods; the dashed line marks Retrain ACC 69.69 as a reference (not a ranked baseline); PLU-GT leads at 66.67. Right: privacy gap $|\mathrm{MIA\ AUC} - 0.5|$ (lower better); PLU-GT stays near chance at 0.497.

5.3 Ablation Study

The ablation study evaluates how the auxiliary terms complement affected-region cross-entropy under identical budgets on all four datasets. Full PLU-GT uses affected-region cross-entropy, replay,

entropy, and cloak when enabled; Pokec-z and Pokec-n keep cloak at zero by design, so their component mixtures differ from Cora/CS. The nested removals first disable cloak, then entropy, and finally replay (CE only); there is no separate single-term w/o-replay row beyond this last step. On Cora, the representative setting with the full component set, removing entropy and cloak raises MIA from 0.522 to 0.811 while ACC remains 68.30; further removing replay to obtain CE only then drops ACC from 68.30 to 58.02, with MIA staying high (0.788). Figure 3 visualizes these representative Cora effects, and the complete results on all datasets are reported in Appendix C.

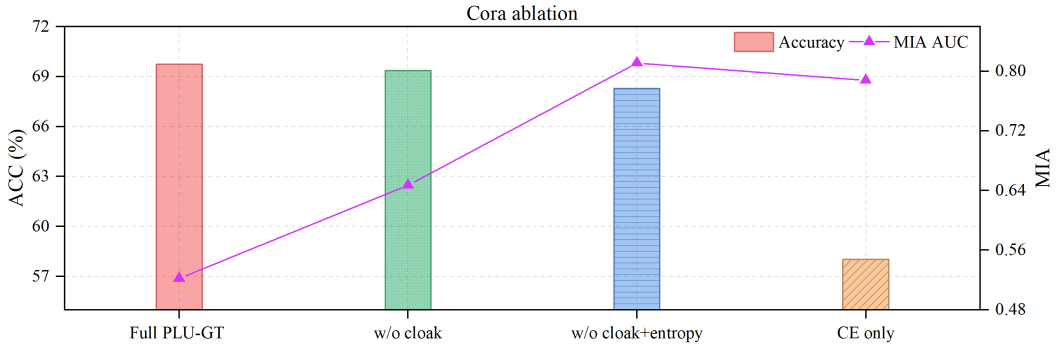

Figure 3: Cora ablation under identical budgets (paired ablation tables in Appendix C). Bars show ACC (%) on the left axis (CE-only 58.02; Full PLU-GT 69.76); the line shows raw MIA AUC on the right axis (removing entropy and cloak raises MIA; Full PLU-GT returns to 0.522).

5.4 Efficiency Analysis

PLU-GT consistently achieves the lowest wall-clock time across all four datasets, requiring only 2.0, 3.2, 2.1, and 3.3 seconds on Pokec-z, Pokec-n, Cora, and CS, respectively. This consistently low runtime demonstrates the efficiency of PLU-GT across datasets with different graph characteristics. Compared with the second-fastest method, MGU, PLU-GT further reduces the unlearning time by approximately 37.5%, 38.5%, and 32.7% on Pokec-z, Pokec-n, and CS, respectively, while achieving a comparable runtime to the second-best MEGU on Cora. Notably, PLU-GT also exhibits relatively small runtime variance across repeated runs, suggesting stable computational behavior. In contrast, several competing methods incur substantially higher or more variable costs; for example, IDEA and FairGU consistently require considerably longer runtimes, while GIF and SGU show pronounced variability on some datasets. Overall, these results indicate that PLU-GT provides an efficient and stable unlearning procedure without introducing substantial computational overhead. Table 2 reports the wall-clock time of all approximate unlearning methods, while additional multi-metric efficiency comparisons are provided in Appendix E.

Table 2: Unlearning wall-clock time in seconds (mean±std). Best (lowest) is **bold**; second best is underlined.

Method	Pokec-z	Pokec-n	Cora	CS
GraphEraser	5.4±0.1	7.3±0.1	2.3±0.4	5.2±0.1
IDEA	19.7±0.2	20.9±0.7	9.9±0.5	18.9±1.0
MEGU	16.2±7.8	10.4±7.4	2.3±0.8	6.1±0.3
MGU	3.2±0.2	5.2±0.2	2.6±0.5	4.9±0.2
GIF	8.7±3.0	9.7±0.2	14.9±1.8	22.6±26.2
FairGU	20.7±0.6	23.3±0.9	8.5±0.6	28.7±0.1
SGU	10.1±0.2	6.5±0.2	2.3±0.3	12.5±11.5
GNNDelete	10.0±1.7	14.0±2.3	5.0±0.7	15.7±0.3
PLU-GT	**2.0±0.2**	**3.2±0.1**	**2.1±0.4**	**3.3±0.2**

5.5 Privacy Analysis

Privacy is ranked by closeness of raw MIA AUC to 0.5 in absolute deviation, so a raw AUC far below 0.5 is not treated as better privacy by itself Fan et al. (2025). On Pokec-z, GIF (0.491)

is closest to chance and SGU (0.515) is second, with PLU-GT at 0.517. On Pokec-n, PLU-GT and GraphEraser tie for closest to chance (0.497). On Cora, PLU-GT is closest to 0.5 (0.522); GraphEraser remains at 0.777. GNNDelete's Cora AUC of 0.082 yields a large $|\mathrm{AUC} - 0.5|$ and is read as inverted membership separability under the fixed attacker, not as stronger privacy. On CS, PLU-GT is closest to chance (0.495) while remaining competitive in ACC (94.33). The full MIA numbers are reported in Table 3. Raw AUC values slightly below 0.5 are reported as-is and ranked by absolute deviation from chance Wu & Wang (2025). Posterior MIA is an empirical probe under a fixed attacker Jain et al. (2026).

Table 3: Membership-inference AUC after unlearning (mean±std). Interpret privacy via $|\mathrm{AUC} - 0.5|$; a raw AUC below 0.5 does not by itself mean better privacy. Best (closest to 0.5) is **bold**; second best is underlined.

Method	Pokec-z	Pokec-n	Cora	CS
GraphEraser	0.530±0.042	**0.497±0.045**	0.777±0.043	0.528±0.006
IDEA	0.531±0.060	0.519±0.073	0.842±0.122	0.546±0.010
MEGU	0.533±0.060	0.487±0.064	0.863±0.104	0.540±0.008
MGU	0.522±0.083	0.513±0.047	0.604±0.116	0.508±0.006
GIF	**0.491±0.025**	0.562±0.053	0.730±0.083	0.521±0.014
FairGU	0.537±0.048	0.546±0.049	0.780±0.052	0.537±0.009
SGU	0.515±0.049	0.512±0.064	0.806±0.083	0.539±0.008
GNNDelete	0.357±0.033	0.624±0.158	0.082±0.035	0.480±0.003
PLU-GT	0.517±0.059	**0.497±0.057**	**0.522±0.097**	**0.495±0.008**

We report the remaining Retrain numbers here. Under the same forget seeds and the original-graph evaluation protocol (Appendix B), the ACC, MIA, and wall-clock times of Retrain and PLU-GT are compared in Table 4. Retrain is a retain-supervised parameter reference under that shared protocol. On Cora, GraphEraser leaves an accuracy gap of about -7.6 relative to Retrain, while PLU-GT is about -0.6; on CS, the gap of PLU-GT stays near zero.

Table 4: Retrain versus PLU-GT after unlearning (mean±std over five forget seeds). ACC (%); MIA AUC; wall-clock time (s).

Metric	Method	Pokec-z	Pokec-n	Cora	CS
ACC (%)	Retrain	69.07±0.40	69.69±0.52	70.44±2.10	94.37±0.36
	PLU-GT	67.45±0.67	66.67±0.50	69.76±0.84	94.33±0.25
MIA	Retrain	0.492±0.046	0.533±0.061	0.552±0.174	0.519±0.006
	PLU-GT	0.517±0.059	0.497±0.057	0.522±0.097	0.495±0.008
Time (s)	Retrain	30.2±0.7	30.6±1.3	16.4±1.2	24.7±0.6
	PLU-GT	2.0±0.2	3.2±0.1	2.1±0.4	3.3±0.2

6 Conclusion

We studied approximate unlearning for a shared Graph Transformer and proposed PLU-GT, which combines affected-patch localization, virtual-node hard editing, and a localized multi-term objective. Under a unified evaluation protocol, PLU-GT remains competitive with full retraining in predictive utility while substantially reducing unlearning time. On the representative Cora ablation, the replay and forget-side regularizers improve the utility–membership-leakage trade-off over affected-region cross-entropy alone. These results are empirical under a fixed posterior membership probe: entropy and cloak regularize forget posteriors and provide empirical rather than formal deletion guarantees. Partition-aware structural editing remains a useful direction for Graph Transformer unlearning, although stronger deletion guarantees, broader attacker evaluations, and validation on additional Transformer backbones remain important future work.

ETHICS STATEMENT

This work evaluates approximate graph unlearning on publicly available graph datasets under a model-level protocol. Membership inference is used only as an empirical probe of residual membership signal after unlearning, not as a claim of re-identification of individuals. The reported results should not be interpreted as a legal deletion guarantee under regulations such as the GDPR. We do not release personally identifying information, and any accompanying code or supplementary material for submission will be prepared in anonymized form.

AI USE STATEMENT

We used generative AI tools for language editing, reference-format checking, assistance with experiment scripts, ablation-study organization, multi-seed experimental runs used to select hyperparameters, and preliminary analysis of experimental results. The method outline was proposed and finalized by the authors; AI tools assisted with implementation details, ablation variants, and hyperparameter search rather than independently inventing the core algorithm. These tools were not used to generate synthetic datasets or to write proofs. AI-assisted code and text were reviewed and verified by the authors, and all experimental protocols, numerical results, citations, and technical claims were independently checked. The authors take full responsibility for the final content of the paper and all associated artifacts.

REFERENCES

Yuming Ai, Xunkai Li, Jiaqi Chao, Bowen Fan, Zhengyu Wu, Yinlin Zhu, Rong-Hua Li, and Guoren Wang. PAGE: A unified approach for federated graph unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19589–19597, 2026.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 499–513, 2022.

Qiang Chen, Zhongze Wu, Ang He, Xi Lin, Shuo Jiang, Shan You, Chang Xu, Yi Chen, and Xiu Su. Graph unlearning meets influence-aware negative preference optimization. arXiv preprint arXiv:2510.19479, 2025. Preprint.

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

Pengfei Ding, Yan Wang, Guanfeng Liu, and Jiajie Zhu. Adaptive graph unlearning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 2767–2774, 2025.

Pengfei Ding, Yan Wang, and Guanfeng Liu. Re-understanding graph unlearning through memorization. In *Proceedings of the ACM Web Conference*, pp. 2940–2951, 2026.

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.

Enjun Du, Xunkai Li, Tian Jin, Zhihan Zhang, Rong-Hua Li, and Guoren Wang. GraphMaster: Automated graph synthesis via LLM agents in data-limited environments. In *Advances in Neural Information Processing Systems*, 2025.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico Lopez, Charilaos Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. In *International Conference on Learning Representations*, 2026.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with

regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union, L119, 1–88, 2016. Regulation.

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. In *Advances in Neural Information Processing Systems*, 2025.

Bowen Fan, Zhilin Guo, Xunkai Li, Yihan Zhou, Bing Zhou, Zhenjun Li, Rong-Hua Li, and Guoren Wang. Unveiling the vulnerability of graph-LLMs: An interpretable multi-dimensional adversarial attack on TAGs. In *Proceedings of the ACM Web Conference*, 2026.

Qiming Guo, Wenbo Sun, Chen Pan, Ye Wang, and Wenlu Wang. Unlearning on spatio-temporal graphs through subgraph virtual edge reconstruction. arXiv preprint arXiv:2608.29369, 2026. Preprint.

Samyak Jain, Ronak Kalvani, Sainyam Galhotra, and Sayan Ranu. Is graph unlearning ready for practice? a benchmark on efficiency, utility, and forgetting. In *International Conference on Learning Representations*, 2026.

Varshita Kolipaka, Akshit Sinha, Debangan Mishra, Sumit Kumar, Arvindh Arun, Shashwat Goel, and Ponnurangam Kumaraguru. A cognac shot to forget bad memories: Corrective unlearning for graph neural networks. In *International Conference on Machine Learning*, 2025.

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.

Xunkai Li, Bowen Fan, Zhengyu Wu, Zhiyu Li, Rong-Hua Li, and Guoren Wang. Toward scalable graph unlearning: A node influence maximization based approach. arXiv preprint arXiv:2501.11823, 2025a. Preprint.

Xunkai Li, Daohan Su, Zhengyu Wu, Guang Zeng, Hongchao Qin, Rong-Hua Li, and Guoren Wang. Toward effective digraph representation learning: A magnetic adaptive propagation based approach. In *Proceedings of the ACM Web Conference*, 2025b.

Xunkai Li, Yinlin Zhu, Boyang Pang, Guochen Yan, Yeyu Yan, Zening Li, Zhengyu Wu, Wentao Zhang, Rong-Hua Li, and Guoren Wang. OpenFGL: A comprehensive benchmark for federated graph learning. In *Proceedings of the VLDB Endowment*, 2025c.

Ningyi Liao, Zihao Yu, Siqiang Luo, and Gao Cong. HubGT: Fast graph transformer with decoupled hierarchy labeling. In *Advances in Neural Information Processing Systems*, 2025.

Haodong Lu, Zekai Chen, Weiwei Ji, Shihao Li, Xunkai Li, Xun Wu, Yinlin Zhu, and Rong-Hua Li. MMFGU: Multimodal federated graph unlearning. arXiv preprint arXiv:2607.28708, 2026. Preprint.

Renqiang Luo, Yongshuai Yang, Huafei Huang, Qing Qing, Mingliang Hou, Ziqi Xu, Yi Yu, Jingjing Zhou, and Feng Xia. FairGU: Fairness-aware graph unlearning in social networks. In *Proceedings of the ACM Web Conference*, pp. 9442–9451, 2026.

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*, pp. 651–661, 2023.

Kun Wu and Wendy Hui Wang. Verification of incomplete graph unlearning through adversarial perturbations. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3204–3215, 2025.

Yujie Xing, Xiao Wang, Bin Wu, Hai Huang, and Chuan Shi. Unifying and enhancing graph transformers via a hierarchical mask framework. In *Advances in Neural Information Processing Systems*, 2025.

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.

Lu Yi and Zhewei Wei. Scalable and certifiable graph unlearning: Overcoming the approximation error barrier. In *International Conference on Learning Representations*, 2025.

Chenhan Zhang, Ali Braytee, Madhushi Bandara, Xin Hao, Paul J. Kennedy, Massimo Piccardi, and Raymond Owen. CUNO: Curriculum and preference optimization for stable graph unlearning under mass deletion. arXiv preprint arXiv:2609.08244, 2026. Preprint.

He Zhang, Bang Wu, Xiangwen Yang, Xingliang Yuan, Xiaoning Liu, and Xun Yi. Dynamic graph unlearning: A general and efficient post-processing method via gradient transformation. In *Proceedings of the ACM Web Conference*, 2025.

Jiaming Zhuo, Ziyi Ma, Yintong Lu, Yuwei Liu, Kun Fu, Di Jin, Chuan Wang, Wenning Wu, Zhen Wang, Xiaochun Cao, and Liang Yang. A closer look at graph transformers: Cross-aggregation and beyond. In *Advances in Neural Information Processing Systems*, 2025.

A NOTATION

Table A.1 lists the main symbols used in the method and experiments.

Table A.1: Notation used throughout the paper.

Symbol	Meaning
$\mathcal{G} = \{\mathcal{V}, \mathbf{A}, \mathbf{X}\}$	Graph, adjacency, features
$\mathbf{H}, \mathbf{Q}, \mathbf{K}, \mathbf{V}$	Representation / attention matrices
K, P_k, L	#patches, k-th patch, padded length
$v_\emptyset$	Virtual node used for padding / hard delete
$\mathcal{V}_f$, ρ	Forget set, forget ratio
$\mathcal{P}_{\text{aff}}$, $\mathcal{V}_{\text{aff}}$	Affected patches / affected retain-train nodes
$\mathcal{V}_{\text{rep}}$, N_{cap}	Replay set / max replay size
q, N_{ref}	Cloak retain reference / retain samples for q
$\lambda_r, \lambda_f, \lambda_c$	Replay / entropy / cloak weights
$\theta^\star, \theta', \theta^{\text{rt}}$	Pretrained / unlearned / retrain parameters

B EVALUATION-GRAPH PROTOCOL

Main-table ACC and posterior MIA are reported on the original evaluation graph after unlearning updates that may use an edited optimization graph (Sec. 5). This appendix checks whether locking the test graph changes the Cora utility reading for representative methods. Table A.2 compares test ACC on the original graph (main protocol) with ACC on the edited graph used at optimization time, under the same forget seeds $\{20, 22, 23, 25, 42\}$. For hard-edit methods, the edited graph applies the same virtual-node remap and incident-edge removal as in training; MEGU and MGU use their transplant edge edits. Across these methods, Cora ACC under the two graphs remains close, so the main-table ranking is not an artifact of evaluating only on the original graph. We do not rank methods by edited-graph posterior MIA: after hard remapping, forget-indexed posteriors are no longer produced by the same attention pathway as on the original graph, and the fixed posterior attacker becomes poorly calibrated for cross-method comparison. Original-graph MIA therefore remains the shared privacy probe in the main tables.

Table A.2: Cora test ACC (%) under the original evaluation graph (main protocol) versus the edited optimization graph (mean±std over five forget seeds). Original-graph ACC for Retrain/PLU-GT matches Table 4; approximate-method original ACC matches Table 1.

Method	ACC @ original	ACC @ edited
Retrain	70.44±2.10	70.40±2.06
GraphEraser	62.86±4.69	62.88±4.75
MEGU	68.80±1.61	69.92±0.85
MGU	68.78±2.55	68.32±2.44
PLU-GT	69.76±0.84	69.72±0.92

C Ablation tables

Each ablation row removes terms from full PLU-GT under identical budgets on all four datasets. Component mixtures differ slightly across datasets: Pokec-z and Pokec-n keep cloak at zero, so those tables omit a separate w/o-cloak row, whereas Cora and CS enable the full auxiliary set and serve as the main illustration of complete component contributions. Main-text Section 5.3 already states the Cora finding; the complete ACC / MIA numbers are archived below in two paired tables (Cora with Coauthor-CS; Pokec-z with Pokec-n). Wall-clock time is reported only in the main efficiency table, because every ablation row uses the same epoch budget.

Table A.3: Cora / Coauthor-CS ablation by removing terms from full PLU-GT (ACC / MIA).

	Cora		Coauthor-CS	
Variant	ACC (%)	MIA	ACC (%)	MIA
Full PLU-GT	69.76±0.84	0.522±0.097	94.33±0.25	0.495±0.008
w/o cloak	69.38±1.84	0.647±0.076	94.10±0.19	0.507±0.006
w/o cloak & entropy	68.30±1.77	0.811±0.130	94.56±0.12	0.525±0.006
CE only	58.02±7.80	0.788±0.095	94.56±0.12	0.525±0.006

Table A.4: Pokec-z / Pokec-n ablation by removing terms from full PLU-GT (ACC / MIA).

	Pokec-z		Pokec-n	
Variant	ACC (%)	MIA	ACC (%)	MIA
Full PLU-GT	67.45±0.67	0.517±0.059	66.67±0.50	0.497±0.057
w/o cloak & entropy	66.34±2.28	0.525±0.061	66.75±0.43	0.500±0.058
CE only	66.48±0.87	0.522±0.060	66.06±0.64	0.539±0.060

D Baseline adaptations on the shared Graph Transformer

Table A.5: How each baseline is adapted to the shared partitioned Graph Transformer. Utility and MIA are reported on the original evaluation graph with updated parameters.

Method	Adaptation on the shared Graph Transformer	Notes
GraphEraser	Hard edit (virtual-node remap + drop incident edges of $\mathcal{V}_f$); warm-start shared θ; CE on retain-train nodes in affected patches only; matched T as PLU-GT.	Not classical per-shard models or shard aggregation.
IDEA	Influence-style LiSSA update on shared θ using FairGP forwards on edited vs. original graphs; k-hop train influence.	One-shot parameter update; scale re-tuned for Transformer gradients.
MEGU	Remove forget edges for the training graph; keep patches intact; SGD on shared θ (+ GATE) with mutual-evolution losses on forget/neighbor sets; matched T as PLU-GT.	Neighbor construction follows the MEGU recipe on this backbone.
MGU	Remove forget edges; keep patches intact; Adam on shared θ with prototype-KL and distillation on influenced neighbors; matched T as PLU-GT.	Influence radius follows layer depth.
GIF	Influence-function / Neumann update on shared θ with FairGP forwards; optional utility-guard rollback.	Patch remapping is not the core GIF step.
FairGU	Diagonal-FIM damping of shared parameters (forget vs. retain importance).	FIM-damping stage only; adversarial FGNN Stage-1 is not transplanted.
SGU	No structural edit; activate a budgeted retain subset and fine-tune shared θ with SGU multi-term losses.	Dataset-tuned budgets and collapse guards.
GNNDelete	Freeze FairGP; train DeletionLayers after encoder/ICA blocks with DEC+NI on k-hop deletion edges.	Deletion operators act on Transformer representations, not MP layers.

All approximate methods load the same pretrained partitioned Graph Transformer checkpoint per dataset and use the same forget seeds and posterior MIA protocol. Structural edits and parameter-update rules are method-specific rather than identical across rows. For GraphEraser, MEGU, and MGU, the main-table results use the same unlearning epoch budget as PLU-GT (Pokec-z $T{=}30$; Pokec-n/Cora/CS $T{=}50$). Unless noted otherwise, iterative methods select checkpoints by validation accuracy on the original evaluation graph, and reported wall-clock time covers the unlearning update after the shared pretrained checkpoint is loaded, including periodic validation used for checkpoint selection, but excluding pretraining, graph preprocessing, and MIA attacker training. All timing numbers in this paper were measured on a single NVIDIA RTX 3060 Laptop GPU under the same software stack. Baselines are Transformer transplants of the official algorithms rather than classical GNN-backbone runs; method-specific hyperparameters follow the corresponding transplant scripts. Table A.5 records the concrete adaptations used in our experiments.

E Additional efficiency and hyperparameter analyses

This appendix collects multi-metric and runtime visualizations that compare all reported baselines, together with the forget-ratio sensitivity study and implementation hyperparameters. They support the main efficiency and privacy tables rather than replacing them.

The main ACC, MIA, and time tables already state the operating points of all approximate methods. On Cora, converting ACC, privacy proximity to chance, and relative speed into comparable scores shows that PLU-GT balances the three axes more evenly than most adapted baselines (Figure A.1).The four-dataset wall-clock ranking remains consistent with Table 2, as visualized in Figure A.2.

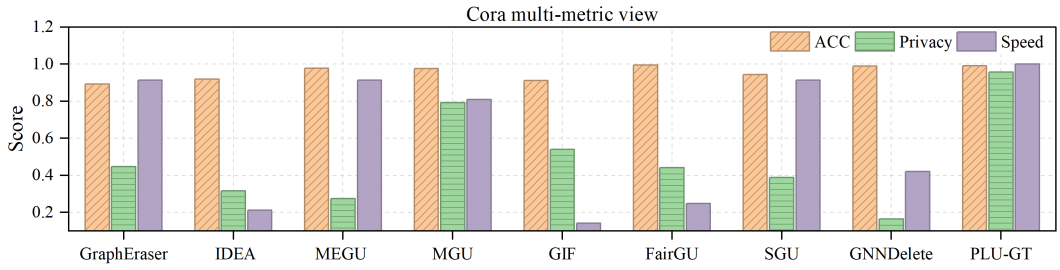

Figure A.1: Cora multi-metric scores for the full baseline suite (higher better). ACC is the ratio to Retrain; Privacy is $1 - \min(|\mathrm{AUC} - 0.5|/0.5, 1)$; Speed is max-normalized inverse wall-clock time among the plotted methods (fastest scores 1).

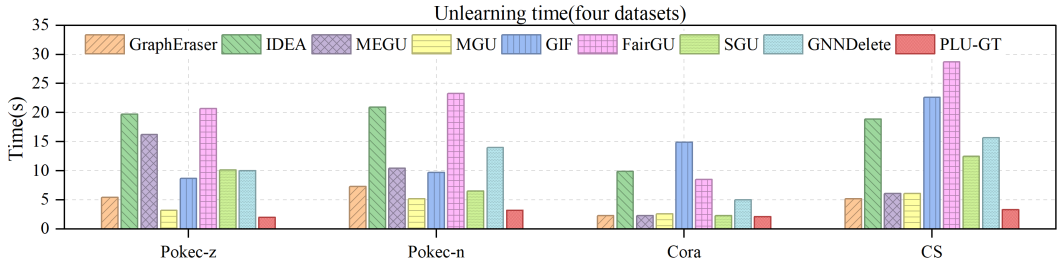

Figure A.2: Wall-clock unlearning time (s) across Pokec-z, Pokec-n, Cora, and CS for the full baseline suite.

We also vary the forget ratio $\rho \in \{0.05, 0.10, 0.15, 0.20\}$ on Cora for PLU-GT under seeds $\{20, 22, 23\}$. This is a request-size sensitivity study rather than a loss-weight sweep: larger ρ means more training nodes are requested for deletion under the same localized objective. Under the fixed posterior-based MIA protocol, the observed leakage metric varies with the forget ratio, while accuracy remains within a relatively narrow range (about 68.7–70.5%). Wall-clock time remains comparatively flat because the current implementation still materializes all patches. These ACC, MIA, and time trends are shown in Figure A.3.

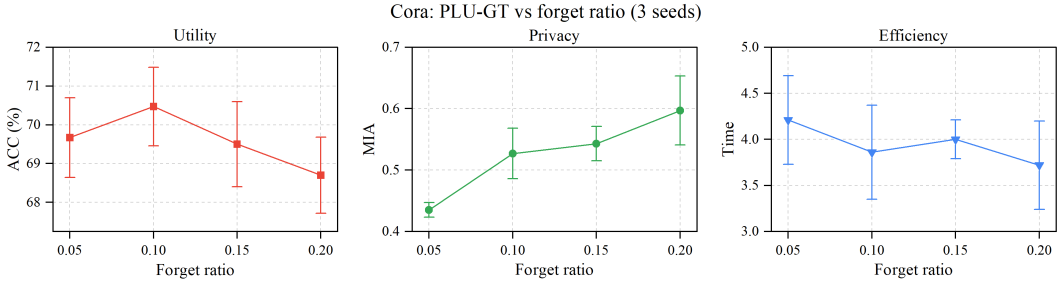

Figure A.3: Cora forget-ratio sensitivity for PLU-GT (seeds $20, 22, 23$; $\rho \in \{0.05, 0.10, 0.15, 0.20\}$). Left: ACC; middle: raw MIA AUC; right: wall-clock time.

Pokec-n uses a larger replay weight and a smaller forget weight than the default recipe. Cora and CS enable cloak, with cloak reference q built from up to $N_{\mathrm{ref}}{=}256$ retain-pool samples as in Sec. 4. The exact unlearning hyperparameters used in the main experiments are listed in Table A.6.

Table A.6: Unlearning hyperparameters for PLU-GT.

Dataset	λ_r	λ_f	λ_c	T	η	N_{cap}
Pokec-z	0.75	0.05	0	30	5e−3	2000
Pokec-n	1.0	0.005	0	50	5e−3	2000
Cora	0.75	0.13	0.09	50	5e−3	2000
CS	0.75	0.13	0.09	50	5e−3	2000

All compared unlearning methods share the same Graph Transformer backbone on each dataset. The corresponding patch, width, depth, and positional-encoding settings are listed in Table A.7.

Table A.7: Shared Graph Transformer backbone settings per dataset.

Dataset	patches	hidden	layers	heads	PE dim	feat norm
Pokec-z	300	64	2	1	8	row
Pokec-n	250	64	2	1	16	row
Cora	30	128	2	4	8	row
CS	100	128	2	4	8	row