

## A Details of Related Works

### A.1 Graph Backdoor Attacks

Graph backdoor attacks have emerged as a critical security threat in graph representation learning, particularly for GNNs. These attacks operate by injecting malicious triggers in the form of decently designed subgraphs or feature perturbations, into carefully selected training nodes. This creates a hidden shortcut between the perturbed target nodes and predetermined malicious labels during training. During inference, the infected model exhibits two distinct behaviors: it will consistently misclassify any input containing the trigger pattern to the attacker-specified labels, while maintaining normal classification performance on unperturbed nodes. To improve the stealth and effectiveness of such attacks, GTA [9] introduced a trainable trigger generator that produces sample-specific perturbations, substantially improving attack success rates. To improve this, UGBA [10] developed a more sophisticated target selection strategy that reduces the required number of target nodes while incorporating homophily constraints through enhancing the cosine similarity between triggers and target nodes, thereby improving both stealthiness and attack performance. To further boost stealthiness, DPGBA [11] addressed key limitations of existing methods by proposing an adversarial learning approach that generates in-distribution triggers and employs a novel loss function to boost attack success rates. While the attack above all perturb the topological structure of target nodes, SPEAR [12] represents a paradigm shift by exclusively perturbing node attributes while preserving the original graph topology. This approach significantly enhances stealthiness and presents unique detection challenges, causing traditional edge pruning based defense strategies ineffective. Therefore, our work focuses on defending against such highly stealthy attacks, particularly SPEAR where topological modifications are absent. Given the increasing sophistication of modern graph backdoor attacks and their potential security implications, developing robust defense mechanisms capable of detecting and mitigating such threat has become imperative.

### A.2 Graph Backdoor Defense

To mitigate the threat posed by graph backdoor attacks, a variety of defense strategies have been proposed, albeit still limited compared to the image domain. Existing methods primarily leverage structural or feature-based inconsistencies introduced by triggers. For example, Prune [9] removes suspicious edges between nodes with low cosine similarity, based on the observation that the trigger of attacks like GTA and UGBA often behave like out-of-distribution outliers, OD [11] adopts an unsupervised reconstruction-based approach using graph auto-encoders to identify and isolate outlier nodes with high reconstruction loss, which are likely to be injected triggers. To further generalize defense strategies across diverse attack settings. To improve the generalization of defense, RIGBD [13] incorporates robust training with randomized edge dropping based identification strategy, aiming to train a benign model using perturbed training data. Despite their effectiveness in certain scenarios, these methods suffer from several limitations. For, example, these method, are all edge dropping based, which will not be effective for SPEAR, and the heuristic method for splitting target nodes and clean nodes in RIGBD lacks robustness in particular situation. In contrast, our method exploits dynamic training signals, particularly the early-stage loss behavior, which is overlooked in existing graph domain defenses. Our training-time defense strategy provides a more fine-grained and adaptive mechanism for backdoor mitigation.

### A.3 Training-time Defense

Training-time defenses against backdoor attacks aim to detect and neutralize poisoned samples during the learning phase, thereby preventing the model from internalizing malicious correlations. Several recent approaches in the image domain have leveraged early training dynamics to isolate poisoned samples. For instance, ABL [14] observed that poisoned samples tend to exhibit lower training losses than benign ones. Based on this, they proposed a two-stage framework: initially identifying and isolating the samples with the lowest losses with fixed isolation ratio, followed by an unlearning phase aimed at mitigating the backdoor effect on these selected samples. DBD [15] and ASD [16], both using Symmetric Cross Entropy (SCE) Loss to separate poisoned samples with ASD using a more precise meta-splitting method and then adopt hybrid training strategies, combining self-supervised or semi-supervised learning to mitigate the influence of poisoned data. PIPD [17] adopts a n-step progressive isolation strategy to iteratively isolate suspected poisoned

data throughout training, improving model robustness without sacrificing clean accuracy. HARVEY [18] constructs a strongly backdoored reference model by leveraging Reverse Cross-Entropy (RCE) loss to iteratively isolate poisoned samples. It begins with a naive RCE-based split, progressively fine-tunes the model to focus on backdoor patterns, and finally retrain on the identified clean subset to obtain a backdoor-free model. While these methods have shown success in image classification tasks, they often overlook the unique challenges posed by graph-structured data such as homophily, message passing, and node interdependence where target nodes may propagate misleading signals to their neighbors. In the context of GNNs, training-time defense becomes more challenging due to the intertwined nature of the graph topology.

## B Time Complexity Analysis

We analyze the time complexity of LoSplit by decomposing it into three key components, following the GCN complexity computation from [28] and [13]:

**(1) Early-stage Training.** The GNN is trained for  $T_S < 20$  epochs on the poisoned graph  $\mathcal{G}_T$ . For each layer:

- *Feature Transformation:*  $\mathcal{O}(NM^2)$  for  $N$  nodes and  $M$ -dim features
- *Neighborhood Aggregation:*  $\mathcal{O}(|E|M)$  via sparse operations, where  $|E|$  is the number of edges

Over  $L$ -layer GCN and  $T_S$  epochs:

$$\mathcal{O}(T_S \cdot L \cdot (NM^2 + |E|M))$$

**(2) Loss Clustering.** For target-class nodes ( $n_t$  nodes):

- Z-score standardization:  $\mathcal{O}(n_t)$  per epoch
- GMM clustering:  $\mathcal{O}(n_t)$  (with  $K = 2$ , fixed parameter)

Total over  $T_S$  epochs:

$$\mathcal{O}(T_S n_t)$$

**(3) Fine-tuning.** Retraining on the purified graph shares the same structure as (1) but runs  $T > 200$  epochs:

$$\mathcal{O}(T \cdot L \cdot (NM^2 + |E|M))$$

**Total Complexity.** Combining all phases:

$$\mathcal{O}((T_S + T)L(NM^2 + |E|M) + T_S n_t)$$

We compare *LoSplit* with RIGBD as both follow a two-stage defense paradigm: identifying poisoned (target) nodes, followed by retraining a new GNN to eliminate the influence of backdoor triggers. This shared structure places them in the same defense category. Under the SPEAR attack, *LoSplit* achieves substantially higher efficiency - it requires only one full GCN training, one naive split-phase training, and simple clustering, while RIGBD incurs the cost of two full trainings and repeated random edge perturbations (see Table 3). We run on an NVIDIA RTX 4090 GPU (24GB) with an Intel i7-13700K CPU, using Python 3.8.19 and PyTorch 1.12.1.

Table 3: Running Time Comparison (seconds) with Complexity Analysis

Method	Complexity	Cora	PubMed	OGB-arXiv
RIGBD	$\mathcal{O}(L(K + 2T)(NM^2 +  E M))$	4.36	18.69	917.12
LoSplit	$\mathcal{O}(L(T_S + T)(NM^2 +  E M) + T_S n_t)$	1.82	6.88	138.65

## C Detailed Proofs

### C.1 Proof of Why RCE Loss Amplifies Early-Stage Dynamics Better than CE Loss

To understand why the Reverse Cross-Entropy (RCE) loss amplifies early-stage loss dynamics better than the standard Cross-Entropy (CE) loss, we analyze and compare their gradient behavior.

469 **Cross-Entropy Loss (CE)** The CE loss is defined as:

$$\mathcal{L}_{\text{CE}} = - \sum_{k=0}^{K-1} q(k|x) \cdot \log p(k|x), \quad (10)$$

470 where  $q(k|x)$  is the one-hot ground-truth label distribution, and  $p(k|x)$  is the predicted probability  
471 over  $K$  classes.

472 **Reverse Cross-Entropy Loss (RCE)** The RCE loss is defined as:

$$\mathcal{L}_{\text{RCE}} = - \sum_{k=0}^{K-1} p(k|x) \cdot \log q(k|x) \quad (11)$$

$$= -p(y|x) \cdot \log 1 + \sum_{k \neq y} p(k|x) \cdot C \quad (12)$$

$$= C \cdot (1 - p(y|x)), \quad (13)$$

473 where  $y$  is the ground-truth class, and  $C = -\log \varepsilon$  with  $\varepsilon = 1^{-10}$ , which prevents  $\log 0$  when  
474  $q(k|x) = 0$ .

475 We now compute the derivative of both loss functions with respect to the predicted probability  $p(y_t|x)$   
476 for the target class  $y_t$ :

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial p(y_t|x)} = -\frac{1}{p(y_t|x)}, \quad (14)$$

$$\frac{\partial \mathcal{L}_{\text{RCE}}}{\partial p(y_t|x)} = -C. \quad (15)$$

478 Note that while CE loss exhibits a diminishing gradient as  $p(y_t|x)$  increases, the RCE loss maintains  
479 a constant gradient magnitude due to the constant coefficient  $C$ . This property ensures stronger  
480 gradient feedback for confident predictions in early training.

481 Assuming a 1-layer GCN, let  $z = \sigma(H_u)$  be the pre-softmax logits where  $\sigma$  is the ReLU activation.  
482 The predicted probability is  $p(y_t|x) = \text{softmax}(z)$ . The embedding  $H_u$  for a poisoned (target) node  
483  $u$  is defined as:

$$H_u = \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_v d_u}} X_v W_B + \frac{1}{d_u} X_u W_B + \frac{1}{\sqrt{d_u d_\delta}} \delta W_B, \quad (16)$$

484 or, in the case of feature perturbation,

$$H_u = \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_v d_u}} X_v W_C + \frac{1}{d_u} (X_u + \delta) W_C, \quad (17)$$

485 where  $\delta$  denotes the injected backdoor feature, and  $W_B, W_C$  are trainable weights in the poisoned  
486 and clean paths, respectively.

487 For a clean node  $v$ , the embedding is:

$$H_v = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} X_u W_C + \frac{1}{d_v} X_v W_C. \quad (18)$$

488 We now compute gradients using the chain rule. First, the derivative of  $p(y_t|x)$  with respect to logits  
489  $z_j$  is:

$$\frac{\partial p(y_t|x)}{\partial z_j} = \begin{cases} p(y_t|x)(1 - p(y_t|x)) & \text{if } j = y_t, \\ -p(y_t|x) \cdot p(j|x) & \text{if } j \neq y_t. \end{cases} \quad (19)$$

490 Next, we have:

$$\frac{\partial z}{\partial H_u} = 1, \quad (20)$$

491 and for backpropagation through the linear layer:

$$\frac{\partial H_u}{\partial W_B} = \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_v d_u}} X_v + \frac{1}{d_u} X_u + \frac{1}{\sqrt{d_u d_\delta}} \delta, \quad (21)$$

492 or

$$\frac{\partial H_u}{\partial W_B} = \sum_{v \in \mathcal{N}(u)} \frac{1}{\sqrt{d_v d_u}} X_v + \frac{1}{d_u} (X_u + \delta). \quad (22)$$

493 Similarly, for a clean node  $v$ ,

$$\frac{\partial H_v}{\partial W_C} = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} X_u + \frac{1}{d_v} X_v. \quad (23)$$

494 Let us define:

$$S_B = \frac{\partial H_u}{\partial W_B}, \quad S_C = \frac{\partial H_v}{\partial W_C}.$$

495 Then the gradients of the losses with respect to weights are:

$$\frac{\partial \mathcal{L}_{CE}}{\partial W_B} = (p(y_t|x) - 1) \cdot S_B, \quad (24)$$

496

$$\frac{\partial \mathcal{L}_{RCE}}{\partial W_B} = C \cdot p(y_t|x)(p(y_t|x) - 1) \cdot S_B. \quad (25)$$

497 For clean nodes:

$$\frac{\partial \mathcal{L}_{CE}}{\partial W_C} = (p(y_c|x) - 1) \cdot S_C, \quad (26)$$

498

$$\frac{\partial \mathcal{L}_{RCE}}{\partial W_C} = C \cdot p(y_c|x)(p(y_c|x) - 1) \cdot S_C. \quad (27)$$

499 **Why Target Nodes Converge Faster During Training** Target nodes are connected to injected  
500 trigger nodes with carefully crafted features  $\delta$ , which amplify their alignment toward the target class  
501  $y_t$ . Consequently:

- 502 • The aggregated embedding  $H_u$  is heavily influenced by  $\delta$ , biasing  $p(y_t|x)$  upward even in  
503 early training.
- 504 • The RCE loss applies a strong gradient penalty proportional to  $1 - p(y_t|x)$ , unlike CE which  
505 weakens as  $p(y_t|x)$  increases.
- 506 • Thus, RCE drives faster reduction of loss for target nodes, amplifying early-stage dynamics  
507 that can be exploited for detection.

508 This explains why RCE is more sensitive to early backdoor convergence than CE, making it more  
509 effective for early-stage poisoned node identification.

## 510 C.2 Proof of Assumption 1

511 We provide a theoretical justification for the existence of an optimal epoch  $t^*$  during training with  
512 Reverse Cross-Entropy (RCE) loss, at which the loss divergence between target and clean nodes  
513 is maximized. This underpins our assumption that early-stage loss dynamics are informative for  
514 separating target nodes from clean nodes.

515 **Gradient Behavior Under CE vs. RCE** Recall the gradient of the CE loss with respect to model  
 516 weights  $W_B$  for a target node with prediction  $p(y_t|x)$  is:

$$\frac{\partial \mathcal{L}_{CE}}{\partial W_B} = (p(y_t|x) - 1) \cdot S_B, \quad (30)$$

517 where  $S_B$  denotes the input-dependent component of the gradient.

518 This gradient is largest in magnitude when  $p(y_t|x) \rightarrow 0$ , and vanishes as  $p(y_t|x) \rightarrow 1$ . Thus, CE  
 519 provides diminishing gradient signals for nodes that converge early, such as target nodes influenced  
 520 by backdoor triggers.

521 By contrast, the RCE gradient is:

$$\frac{\partial \mathcal{L}_{RCE}}{\partial W_B} = C \cdot p(y_t|x)(p(y_t|x) - 1) \cdot S_B, \quad (31)$$

522 where  $C = -\log \varepsilon$  is a large positive constant. This expression is a concave quadratic function of  
 523  $p(y_t|x)$ , which reaches its maximum magnitude at  $p(y_t|x) = 0.5$ , and vanishes at both  $p = 0$  and  
 524  $p = 1$ . Therefore, RCE focuses learning on predictions with moderate confidence, which typically  
 525 arise during early training, and suppresses gradients for already confident predictions.

526 The difference in gradient behavior between CE and RCE is illustrated in Figure 9. While CE assigns  
 527 the largest gradient when predictions are highly incorrect, RCE concentrates gradient energy around  
 528  $p = 0.5$ , making it more sensitive to early-stage dynamics.

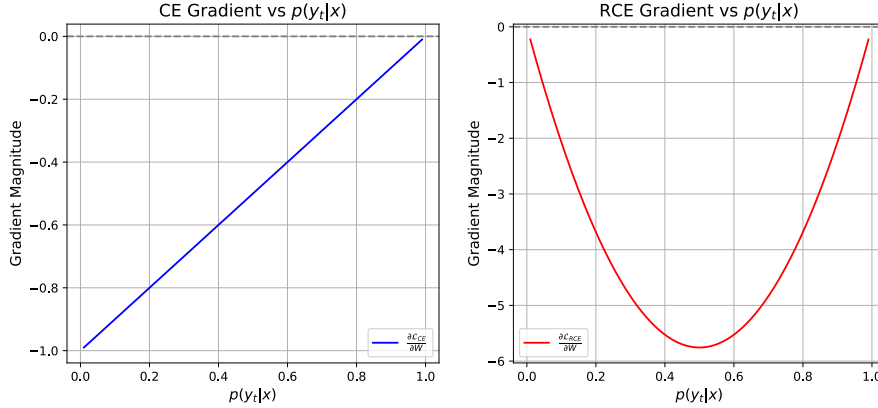


Figure 9: Gradient magnitude (w.r.t. prediction confidence  $p(y_t|x)$ ) for Cross-Entropy (CE) and Reverse Cross-Entropy (RCE). Unlike CE, RCE induces a peak gradient at  $p = 0.5$ , which amplifies early-stage separation between fast-converging target nodes and slower clean nodes.

529 **Why an Optimal Epoch  $t^*$  Exists** During early training, both clean and target nodes start with  
 530 low prediction confidence. However, due to the connection to adversarial triggers, target nodes  
 531 quickly converge toward the target class  $y_t$ , surpassing the  $p(y_t|x) = 0.5$  region and entering the  
 532 gradient-suppressed regime of RCE.

533 In contrast, clean nodes, lacking such shortcut connections, progress more slowly and remain within  
 534 the high-gradient region centered around  $p(y_t|x) = 0.5$ . As a result, the RCE loss of clean nodes  
 535 temporarily increases (or decreases slowly), while the loss of target nodes decreases sharply.

536 These theoretical insights are empirically illustrated in Figure 10, which compares the per-epoch  
 537 gradient magnitudes of clean and target nodes under CE and RCE. As shown, CE quickly flattens the  
 538 gradient difference, while RCE maintains a transient peak in target gradients, creating a temporal  
 539 separation window between the two node types.

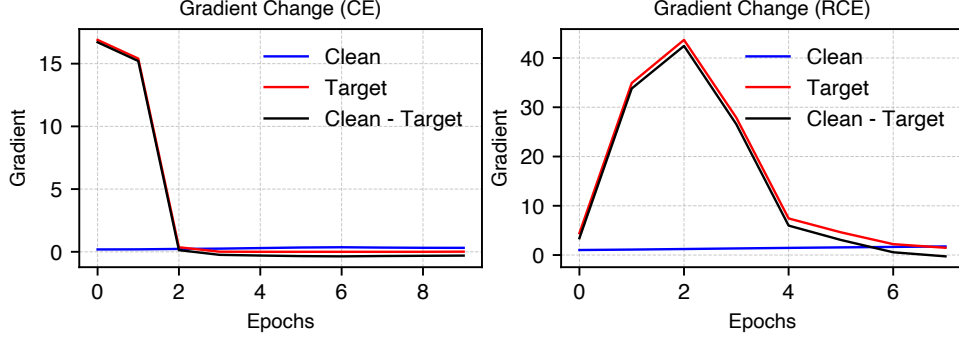


Figure 10: Gradient dynamics over training epochs under Cross-Entropy (left) and Reverse Cross-Entropy (right). Under CE, both clean and target node gradients quickly decay and overlap. Under RCE, target node gradients exhibit a transient peak in early epochs while clean gradients remain low, producing a clear early-stage separation.

540 This mismatch in convergence speed causes the loss gap between the two groups to widen during  
541 early epochs and then shrink as both converge. Formally, let

$$\Delta(t) = \mathbb{E}_{v \in \mathcal{V}_C} [\mathcal{L}_{\text{RCE}}^{(t)}(v)] - \mathbb{E}_{u \in \mathcal{V}_B} [\mathcal{L}_{\text{RCE}}^{(t)}(u)]$$

542 denote the expected RCE loss difference at epoch  $t$  between clean and target node sets. Since this  
543 function increases and then decreases over training, it admits a maximum at some intermediate epoch  
544  $t^*$ , where the two groups are maximally separated in the loss space.

545 **Conclusion** The non-monotonic gradient structure of RCE induces a temporal window during  
546 which the loss values of target and clean nodes diverge maximally. This guarantees the existence of a  
547 well-defined epoch  $t^*$  that maximizes the separation between the two groups. Therefore, Assumption 1  
548 is theoretically justified and supports the design of our early-stage detection mechanism based on  
549 loss dynamics.

## 550 D Details of Experimental Settings

### 551 D.1 Datasets Details

552 **Cora, Citeseer, and Pubmed.** These are three widely used citation network benchmarks in graph  
553 learning. In each dataset, nodes represent documents, and edges denote citation relationships.  
554 Cora contains 2,708 nodes and 5,429 edges, with each node described by a 1,433-dimensional  
555 bag-of-words feature vector across 7 classes. CiteSeer includes 3,327 nodes, 4,552 edges, and  
556 3,703-dimensional features across 3 categories. PubMed consists of 19,717 nodes and 44,338 edges,  
557 with 500-dimensional TF/IDF-like features and 3 classes.

558 **OGB-arXiv.** Provided by the Open Graph Benchmark, OGB-arXiv represents a large-scale citation  
559 network from the arXiv paper corpus. It includes 169,343 nodes and 1,166,243 edges. Each node is  
560 described by a 128-dimensional feature derived from title and abstract content, and classified into one  
561 of 40 scientific fields. It uses a temporal split for train/validation/test sets to better simulate real-world  
562 applications.

Table 4: Statistics of datasets used in our experiments.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,552	3,703	3
PubMed	19,717	44,338	500	3
OGB-arXiv	169,343	1,166,243	128	40

## D.2 Attack Method

We briefly describe the four backdoor attack methods evaluated in this paper.

**GTA:** GTA [9] is the first method to leverage a learnable trigger generator that produces sample-specific subgraph triggers for each target node. The generator is optimized to maximize the attack success rate (ASR), but it lacks constraints on stealthiness, which may lead to detectable patterns in the poisoned graphs.

**UGBA:** UGBA [10] enhances GTA by selecting a set of diverse and representative nodes for poisoning, improving attack efficiency. It also incorporates a homophily constraint that forces the generated trigger features to align with those of the target node’s neighbors, improving stealth and making the backdoor more difficult to detect.

**DPGBA:** DPGBA [11] proposes an adversarial training framework to generate in-distribution triggers. A novel loss function ensures that the generated triggers remain close to the data distribution while still being effective. This design helps the attack remain stealthy and improves the overall ASR compared to prior methods.

**SPEAR:** SPEAR [12] targets the feature space instead of the graph structure by learning to perturb a small number of node features in a subtle but malicious way. It selects stealthy features and vulnerable nodes to inject the trigger. SPEAR is highly effective and particularly challenging to defend against, as it leaves the graph structure unchanged and manipulates only a few features.

## D.3 Defense Baseline

We evaluate the effectiveness of *LoSplit* against a range of representative defense baselines, which can be grouped into three categories:

**Detection-Deletion Defenses.** These methods aim to remove or weaken the influence of trigger structures in the graph:

- **Prune** [10]: Prune observes that trigger patterns often violate the homophily property commonly found in real-world graphs. It removes edges between nodes with low similarity, thereby disrupting trigger connections. Prune can be applied during both training and inference.
- **OD** [11]: Outlier Detection (OD) uses a graph auto-encoder to reconstruct node features and prunes nodes with high reconstruction error. This helps eliminate anomalous trigger nodes while retaining clean ones. It is applied during training.

**Robust Training-Based Defenses.** These defenses aim to improve the integrity of the model by first identify and isolate target nodes and then retrain the backdoored model:

- **RIGBD** [13]: RIGBD identifies candidate trigger nodes through random edge perturbations and performs adversarial training to suppress their influence. It enhances model robustness against structure-based backdoor attacks.
- **ABL** [14]: ABL detects target nodes based on abnormal loss increases during early training epochs. It isolates suspected poisoned nodes and fine-tunes the model using a unlearning strategy to mitigate backdoor effects.

**Robust GNN Models.** These methods are originally designed for general adversarial attacks but can be adapted to backdoor scenarios:

- **GNNGuard** [27]: GNNGuard uses node feature similarity to adaptively reweight edges during training, effectively suppressing adversarial perturbations and enhancing model resilience.
- **RobustGCN** [26]: RobustGCN models node embeddings as Gaussian distributions and introduces variance-based attention to reduce the influence of anomalous neighbors, improving robustness against structure attacks.

## E Additional Results of Separation using Early-Loss Dynamics For Different Attacks and Datasets

In this section, we present a further analysis of the target nodes identification via early loss dynamics under different graph backdoor attacks and datasets discussed in Section 4.2. Specifically, Figure 11, Figure 12 and Figure 13 illustrates the loss distribution of target class nodes on the Citeseer, Pubmed and OGB-arXiv dataset for four representative attacks, i.e. GTA, UGBA, DPGBA, and SPEAR.

From these results, we observe that across all attack variants, target nodes tend to exhibit distinct early-stage loss distributions compared to clean nodes in the same class. This separation is particularly clear in GTA and UGBA, where the backdoor introduces a strong shortcut, leading to rapidly decreasing or highly concentrated loss values for target nodes. Even in more stealthy attacks such as DPGBA and SPEAR, which aim to minimize detectable differences in graph structure or features, but the early training dynamics still provides reliable signals for separating poisoned and clean samples.

These findings demonstrate the generalization and robustness of *LoSplit*'s early loss based identification strategy. Regardless of the underlying attack mechanism or trigger form, the training loss in early epochs naturally diverges between clean and target nodes.

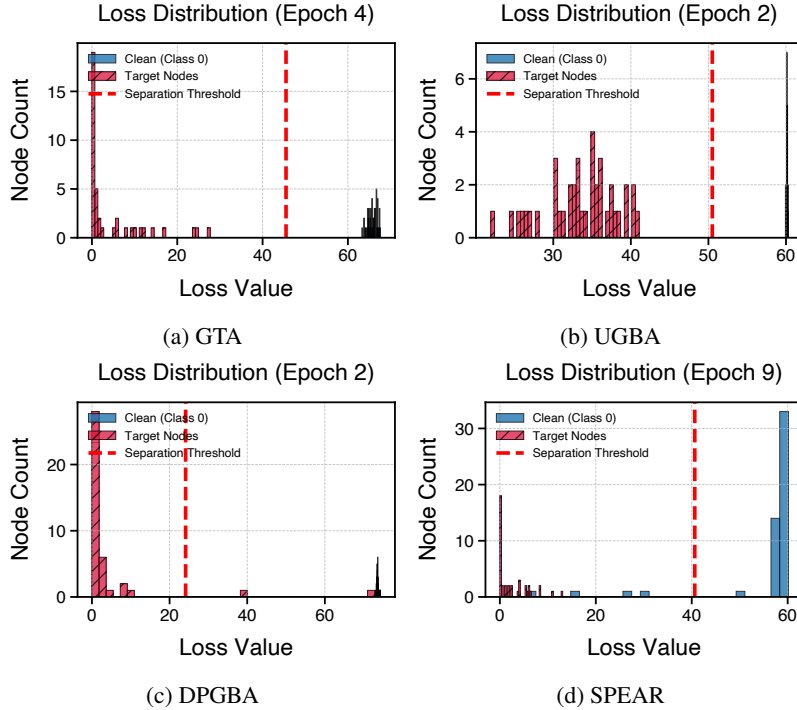


Figure 11: Distribution of the Early-Stage Loss Separation in Target Class on Citeseer.

## F Results of Separation using Early-Loss Dynamics Under Different Malicious Labels

We further assess the robustness of *LoSplit* by varying the attacker-specified target class  $y_t$  across all possible categories. Since different classes may exhibit different learning dynamics—particularly those that are easier to learn—we aim to investigate whether *LoSplit* remains effective when the backdoor is injected into such easily learnable classes. This scenario is especially challenging because the clean nodes belonging to the target class may converge faster during training, potentially obscuring the divergence between target and clean nodes. Experimental results presented in Figure 14, 15, 16, 7 demonstrate that *LoSplit* consistently achieves clear separation, highlighting its robustness regardless of the specific malicious label chosen.



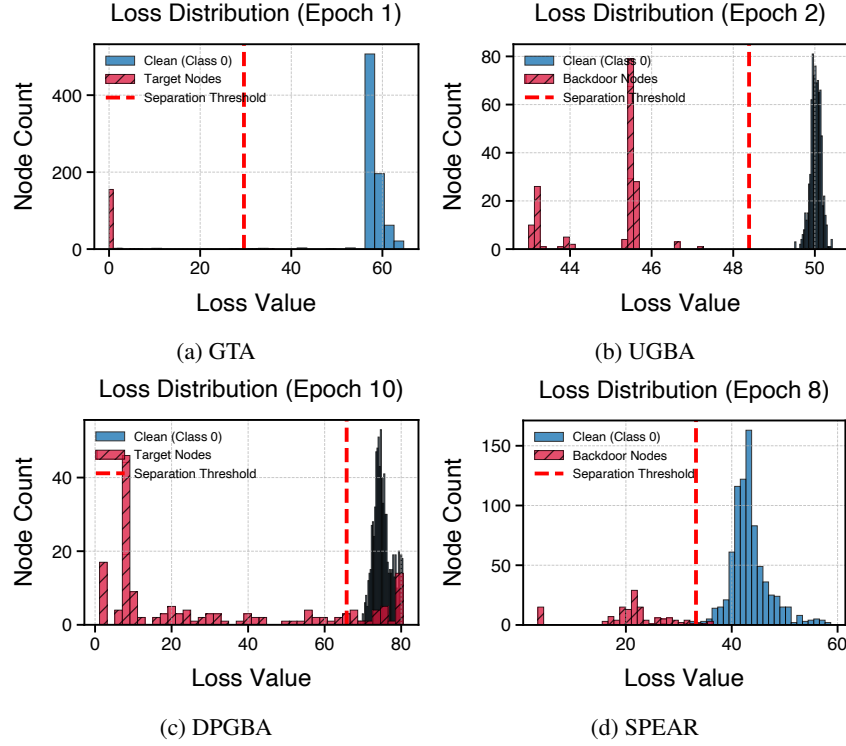


Figure 12: Distribution of the Early-Stage Loss Separation in Target Class on Pubmed.

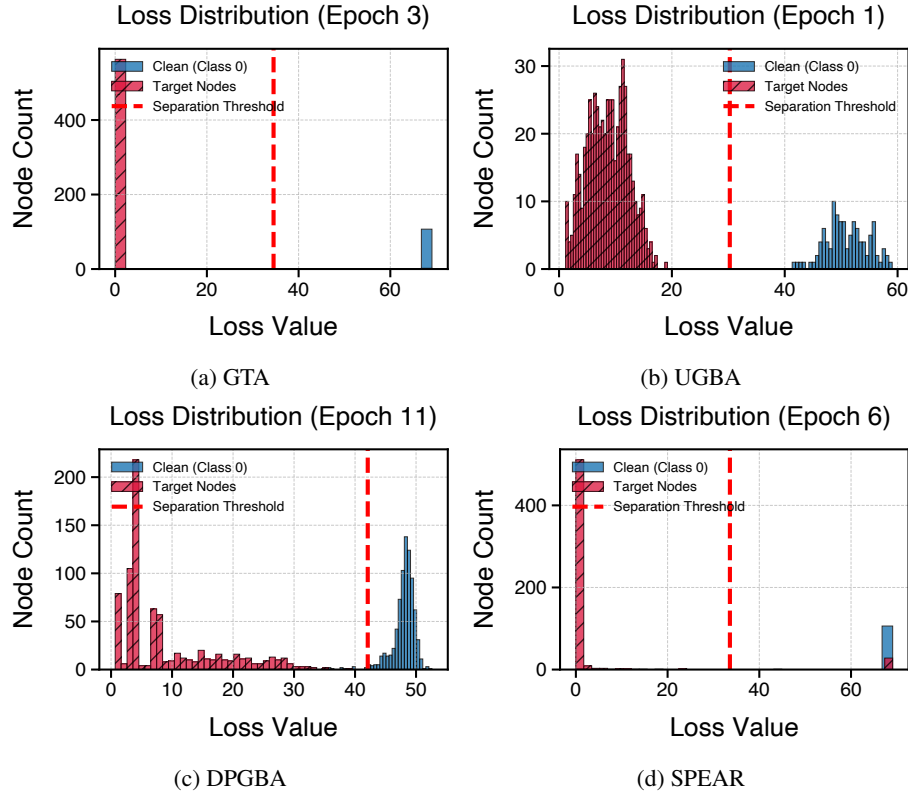


Figure 13: Distribution of the Early-Stage Loss Separation in Target Class on OGB-arXiv.

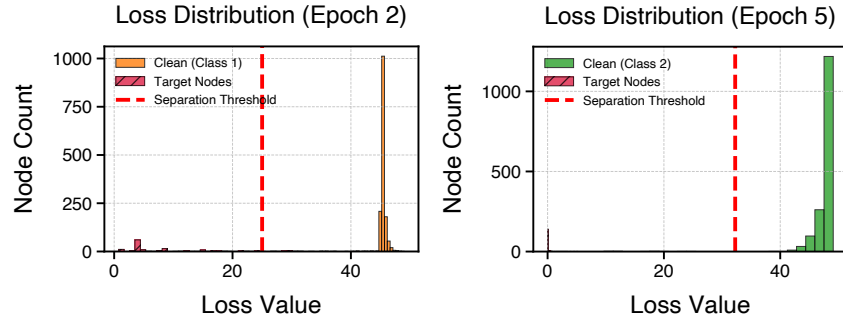


Figure 14: GTA

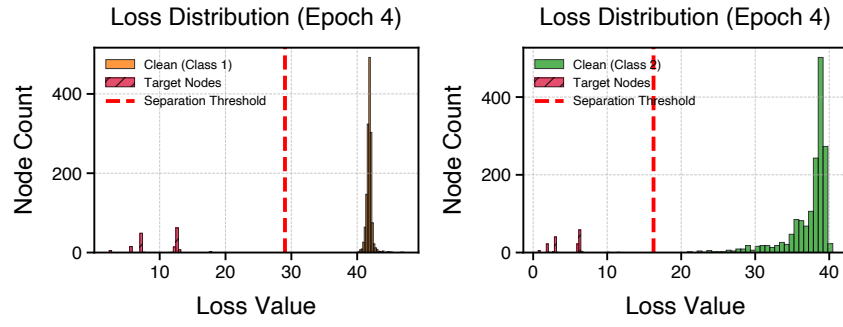


Figure 15: UGBA

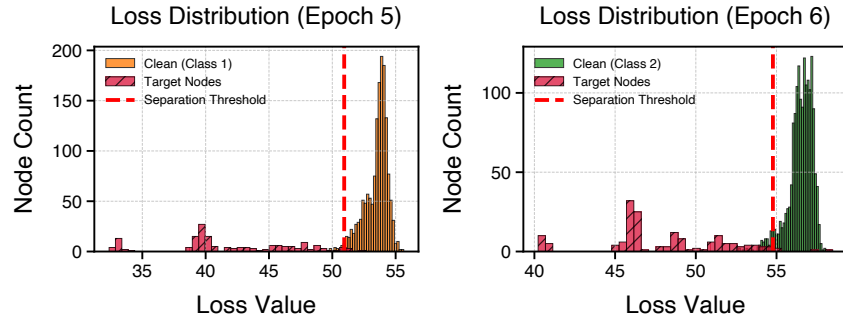


Figure 16: DPGBA

## G Results of Defense Performance and Separation Ability using Early-Loss Dynamics under Different Trigger Numbers

To assess the robustness of *LoSplit* under different attack budgets, we evaluate its performance against varying numbers of backdoor triggers:  $\{10, 20, 40, 100, 160\}$  for Cora and  $\{40, 80, 160, 200, 320\}$  for PubMed under the SPEAR attack. We use split training parameters  $T_S = 20, \eta_S = 0.005$  (Cora) and  $T_S = 25, \eta_S = 0.002$  (PubMed). Results are summarized in Table 5.

Table 5: ASR and ACC before and after applying *LoSplit* under different trigger numbers.

Dataset	Trigger Number	Before		After LoSplit				
		ASR ↓	ACC ↑	ASR ↓	ACC ↑	Rec ↑	Prec ↑	FPR ↓
Cora	10	100.0	83.33	0.00	83.70	100.0	100.0	0.00
	20	100.0	83.33	0.00	84.07	100.0	100.0	0.00
	40	97.78	84.07	0.00	83.70	100.0	100.0	0.00
	100	92.25	82.96	0.00	82.59	97.1	100.0	0.55
	160	94.83	80.37	0.00	82.59	97.6	100.0	0.74
PubMed	40	93.57	84.83	1.92	85.19	97.0	80.0	0.03
	80	90.31	84.88	1.67	85.28	97.1	82.5	0.05
	160	92.49	84.17	0.75	85.13	92.7	87.5	0.28
	200	91.94	84.98	0.58	85.39	92.4	90.5	0.38
	320	76.37	84.93	5.08	85.19	100	64.1	0.00

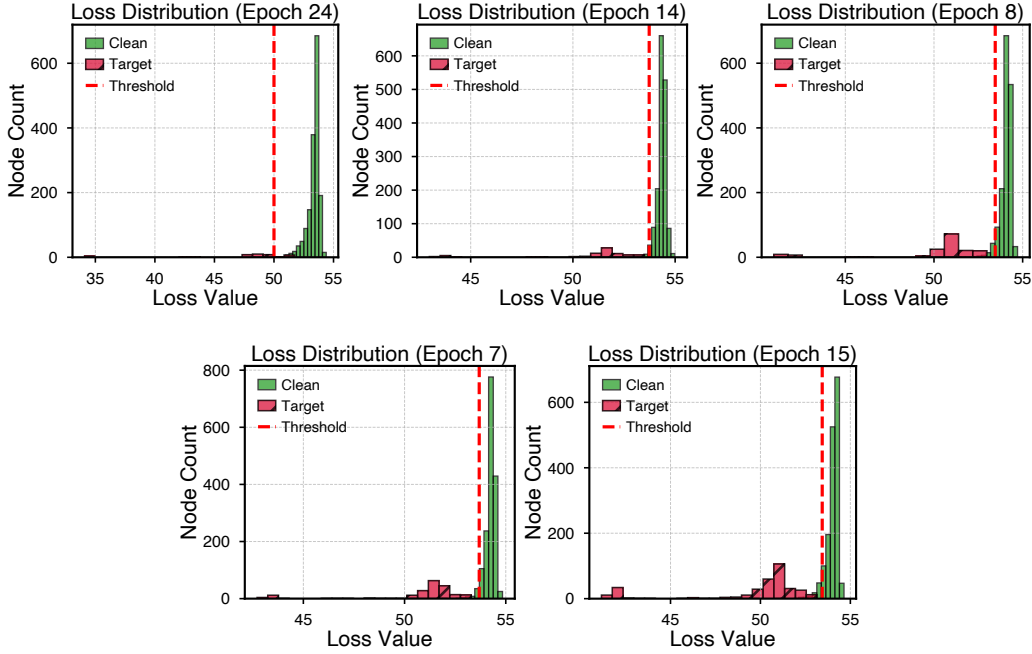


Figure 17: SPEAR on PubMed under different attack budget

Overall, *LoSplit* consistently achieves near-zero ASR and clean accuracy on par with the baseline across all trigger counts. Its target node detection remains effective, with high recall and precision, and low false positive rate, even at high trigger numbers (e.g., 160 on Cora or 320 on PubMed). While detection metrics exhibit minor variation, defense performance remains stable, validating the robustness of our Decoupling-Unlearning strategy. Notably, in weak attacks (e.g., PubMed with 320 triggers whose ASR is below 80%), separation becomes less distinct due to weaker loss divergence, slightly impacting detection ability.

These results demonstrate that *LoSplit* is both scalable and reliable across a wide range of perturbation rate. Visualizations of separation under different trigger numbers are shown in Figure 8 and 17.

## H Additional Results of Defense Performance under Different Backbone Model

To further evaluate the generalizability of our method across different GNN architectures, we conduct experiments using both GAT and GraphSAGE as backbones. The results in Table 6 summarize the defense performance of *LoSplit* and RIGBD under various backdoor attacks (GTA, UGBA, DPGBA, and SPEAR) on three datasets (Cora, PubMed, and OGB-arxiv).

Table 6: Comparison with backdoor defenses under different attacks.

Attacks	Defense	Cora		PubMed		OGB-arxiv	
		ASR(%) ↓	ACC(%) ↑	ASR(%) ↓	ACC(%) ↑	ASR(%) ↓	ACC(%) ↑
GTA	GAT	83.76	83.33	97.61	81.23	92.67	64.92
	GraphSage	99.63	84.07	98.73	84.68	90.83	64.75
	RIGBD-GAT	93.96	85.56	1.02	84.63	1.73	64.51
	RIGBD-GraphSage	99.63	78.89	98.97	75.14	80.24	64.47
	LoSplit-GAT	1.33	83.70	1.16	85.08	0.81	64.70
	LoSplit-GraphSage	2.58	80.74	0.58	84.98	0.59	64.30
UGBA	GAT	100.00	76.67	91.28	84.22	92.53	65.01
	GraphSage	97.33	84.07	95.38	86.25	96.16	65.66
	RIGBD-GAT	4.33	84.81	90.42	84.27	2.04	65.02
	RIGBD-GraphSage	98.16	77.04	90.11	82.30	2.06	65.75
	LoSplit-GAT	2.66	80.00	0.67	85.12	1.13	65.83
	LoSplit-GraphSage	4.00	82.96	1.67	83.51	0.74	65.95
DPGBA	GAT	100.00	83.70	91.53	84.07	92.78	66.13
	GraphSage	98.89	82.22	89.30	85.84	91.61	67.53
	RIGBD-GAT	6.22	83.33	0.84	83.51	1.28	64.86
	RIGBD-GraphSage	1.78	80.74	1.42	85.49	2.03	65.61
	LoSplit-GAT	0.89	84.07	0.51	83.90	1.02	65.31
	LoSplit-GraphSage	0.44	82.59	0.63	85.44	1.38	65.90
SPEAR	GAT	95.94	83.33	97.98	83.51	100.00	67.55
	GraphSage	100.00	84.44	83.00	85.59	95.00	67.00
	RIGBD-GAT	84.89	82.96	84.22	83.82	93.42	65.87
	RIGBD-GraphSage	100.00	82.96	100.00	85.24	91.56	65.99
	LoSplit-GAT	0.44	83.33	0.26	85.45	0.21	65.87
	LoSplit-GraphSage	0.37	83.33	0.00	85.64	0.00	66.09

Across all settings, *LoSplit* consistently achieves significantly lower ASR compared to both the original GAT and GraphSAGE, as well as the RIGBD defense. Moreover, the clean accuracy (ACC) of *LoSplit* remains comparable to or even higher than that of RIGBD, indicating that our decoupling-unlearning training does not sacrifice model utility. Notably, even under challenging attacks like SPEAR, which perturb only node features, *LoSplit* maintains ASR close to 0% and substantially outperforms RIGBD, especially on the PubMed and OGB-arxiv datasets.

These results demonstrate that *LoSplit* generalizes robustly across different GNN model architectures. Overall, *LoSplit* is not tailored to any specific GNN design and can serve as a general-purpose training-time defense framework across various architectures.

## 664 I Sample-wise Loss Distribution under RCE Loss in image

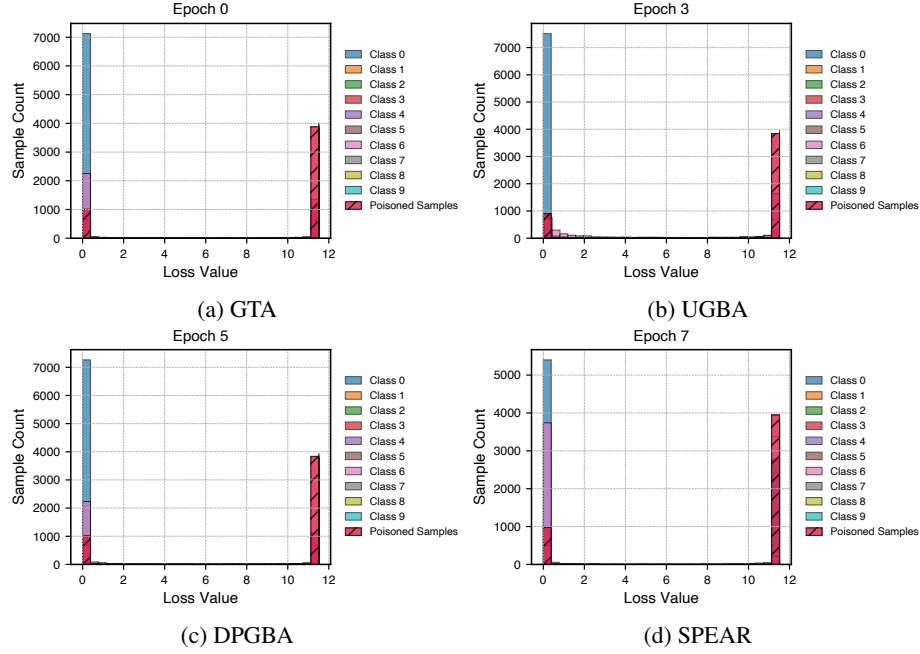


Figure 18: Distribution of the Early-Stage RCE Loss in Image Domain under Cifar-10 Dataset.

## 665 J Training Algorithm

---

### Algorithm 1 Algorithm of *LoSplit*

---

**Require:** Backdoored graph  $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{A}_T, \mathcal{E}_T, \mathbf{X}_T, \mathbf{Y}_T)$ ; split epoch  $T_S = 20$ ; split learning rate  $\eta_S = 0.05$ ; RCE Constant  $C = \log(1^{-10})$

**Output:** Backdoor-free GNN node classifier

1: Randomly initialize  $\theta_{LoSplit}$  for an L-layer GNN  $f_{LoSplit}$  using Eq. [1]

2: **for**  $t = 1, 2, \dots, T_S$  **do**

3:   Compute RCE loss  $\ell_i^{(t)}$  for all  $i \in \mathcal{V}_T$ ;

4:   For each class  $y \in \mathbf{Y}_T$ , compute intra-class variance of  $\ell_i^{(t)}$ ;

5:   Identify target label  $y_t^{(t)}$  via Eq. (4);

6:   Extract nodes with label  $y_t^{(t)}$  as  $\mathcal{V}_{y_t^{(t)}}$ ;

7:   Compute  $\mu, \sigma$  of  $\ell_i^{(t)}, \forall i \in \mathcal{V}_{y_t^{(t)}}$ ;

8:   Standardize loss into z-scores using Eq. (5);

9:   Fit GMM on z-scores to get clusters  $\mathcal{C}_{low}^{(t)}, \mathcal{C}_{high}^{(t)}$ ;

10:   Compute threshold  $\tau^{(t)}$  via Eq. (6);

11:   Compute  $D^{(t)} = \mathbb{E}_{i \in \mathcal{C}_{high}^{(t)}}[\ell_i^{(t)}] - \mathbb{E}_{j \in \mathcal{C}_{low}^{(t)}}[\ell_j^{(t)}]$ ;

12: **end for**

13: Select best split epoch  $t^* = \arg \max_t D^{(t)}$ ;

14: Identify target and clean nodes via Eq. [8];

15: Randomly initialize  $\theta$  for an L-layer GNN node classifier  $f$ ;

16: Fine-tune  $f$  via Eq. [9];

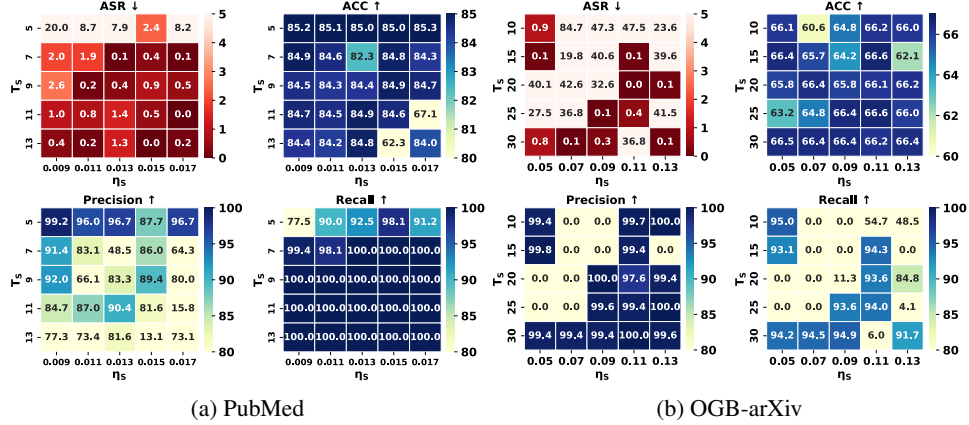
17: **Return** backdoor-free GNN classifier  $f$ ;

---

666 *LoSplit* follows a two-stage strategy to separate and neutralize backdoor triggers. In the first stage, it  
 667 performs early training using Reverse Cross-Entropy (RCE) loss to track loss dynamics across epochs.

By analyzing intra-class variance and standardizing target class losses into z-scores, it identifies suspicious target nodes via GMM-based clustering and selects the epoch with the most pronounced loss separation. In the second stage, it first unlearn the target label and then reassigns new labels to the suspected target nodes and retrains to obtain a backdoor-free classifier.

## K Additional Results of Hyperparameter Analysis



To further evaluate the robustness and generalizability of our method under the SPEAR attack, we conduct a comprehensive hyperparameter analysis on the PubMed and OGB-arxiv datasets. The main hyperparameters under consideration are the early-stage split epoch  $T_S$ , the selected split threshold  $\eta_S$ .

We evaluate the performance using four metrics: attack success rate (ASR), clean accuracy (ACC), backdoor precision, and recall. For the PubMed dataset, we vary  $T_S \in \{5, 7, 9, 11, 13\}$  and  $\eta_S \in \{0.009, 0.011, 0.013, 0.015, 0.017\}$ . The results show that the lowest ASR (as low as 0.0%) occurs when  $T_S = 13$  and  $\eta_S = 0.013$  or  $0.017$ , suggesting effective suppression of backdoor influence at later training stages and higher thresholds. Meanwhile, ACC remains relatively stable around 84–85% with only a slight drop under aggressive ASR minimization, indicating a trade-off between robustness and utility. Both precision and recall remain high (often reaching 100%), with precision peaking when ASR is low, meaning most identified target nodes are true positives. These results highlight that on PubMed, later split epochs (e.g.,  $T_S = 13$ ) combined with moderate-to-high threshold values (e.g.,  $\eta_S = 0.015$ – $0.017$ ) yield the best trade-off between low ASR and high clean accuracy.

For the OGB-arxiv dataset, we explore differently by  $T_S \in \{10, 15, 20, 25, 30\}$  and  $\eta_S \in \{0.05, 0.07, 0.09, 0.11, 0.13\}$ . While some configurations lead to high ASR (e.g., over 40%), optimal combinations such as  $T_S = 20$  and  $\eta_S = 0.11$  reduce ASR to below 1%, demonstrating the method’s ability to filter target-class poisoned nodes. ACC remains consistently around 66%, aligned with the dataset’s classification difficulty. Similarly, both precision and recall reach 100% under favorable settings (e.g.,  $T_S = 20$ ,  $\eta_S = 0.11$  or  $0.13$ ), confirming effective and stable detection. Overall, across both datasets, careful tuning of  $T_S$  and  $\eta_S$  is crucial to balancing low ASR with high ACC. The method consistently achieves high precision and recall, validating its reliability in target nodes detection and offering practical guidance for hyperparameter selection in real-world deployments.

## L Reproducibility

Code will be publicly available upon acceptance.

## M Limitations and Future Works

While *LoSplit* demonstrates strong performance in separating and mitigating backdoor effects, it remains sensitive to the training configurations, and specific attack parameters, which may limit its

robustness in highly heterogeneous or real-world deployment scenarios. Moreover, our decoupling-unlearning strategy focuses on removing the association between target nodes and the target class, but does not explicitly restore target nodes to their original class, potentially limiting full semantic recovery. In future work, we plan to explore inference-time defenses that dynamically detect and neutralize backdoor behaviors without access to retraining. Additionally, developing strategies to recover the original labels of target nodes could further enhance model performance. We also envision extending our framework to other domains beyond graph-structured data, paving the way for more generalized and transferable training-time defense methods.

## **N Broader Impact**

This paper presents work whose goal is improve the robustness of GNNs and thus advance the field of machine learning. While there may be potential societal consequences, we do not identify any that must be specifically highlighted here.