A More Details and Reasons about Our Setting

A.1 Threat Model

We adopt a unified setting which is also used by Graph Robustness Benchmark (Zheng et al., 2021), that is evasion, inductive, and black-box. We give the reasons below.

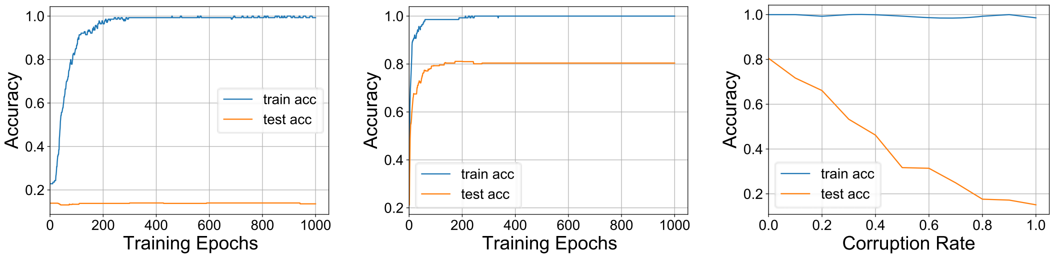
Evasion. The attack only happens at test time, which means that defenders are able to obtain the original clean graph $\mathcal{G}_{\text{train}}$ for training, while testing on a perturbed graph $\mathcal{G}'$. We adopt evasion attacks for two reasons. Firstly, the unnoticeability constraints in previous works, ranging from degree distributions (Zügner et al., 2018) to statistical features (Wu et al., 2019), can not reflect how much the graph is changed indeed, as they are *weakly correlated* to the information required for node classification [4]. Secondly, it is well-known that neural networks have universal approximation power (Hornik et al., 1989), thus can easily overfit the training set (Goodfellow et al., 2016), or even *memorize* the labels appeared during training (Zhang et al., 2017). As a generalization from deep learning models to graphs, GNNs tend to exhibit similar properties, which is shown empirically in our experiments (See Appendix A.2 for details). Even trained on a highly poisoned graph, GNNs may still converge to 100% training accuracy, for which defenders are unable to tell whether it is perturbed hence unlikely to make any effective defenses. Thus, before a better solution appears, the evasion setting might be more appropriate.

Inductive. The training of GNNs is performed in an inductive manner, that is, f_θ is trained on training nodes with their labels and inter-connections while testing with the whole graph. We argue that it is *inappropriate* to conduct evasion attacks for GNNs learned in a transductive manner, which enables the GNN to access all of the nodes and edges (not all the labels) during training. If new nodes or connections are added after training, it will violate the transductive setting. Besides, inductive learning often happens in many applications, as the networks usually grow larger over time.

Black-box. The adversary has no information about the target model, but the adversary may obtain the graph and training labels to train a surrogate model for generating perturbed graph $\mathcal{G}'$.

Combining all of the above, conducting effective attacks raises special challenges to adversaries, since defenders can adopt the information extracted from training graph $\mathcal{G}_{\text{train}}$ to learn more robust hidden representations (Zhu et al., 2019), or learn to drop noisy edges (Wu et al., 2019; Zhang & Zitnik, 2020; Jin et al., 2020), or even perform adversarial training (Jin & Zhang, 2021; Feng et al., 2021) which is known as one of the strongest defense mechanisms in the domain of images (Goodfellow et al., 2015; Madry et al., 2018).

A.2 Memorization Effects of Graph Neural Networks



(a) Original labels (b) Random labels (c) Partial random labels

Figure 5: Training curve of GCN on Cora with random labels

We conduct experiments with GCN (Kipf & Welling, 2017) on Cora (Yang et al., 2016). The architecture we select is a 2-Layer GCN with 16 hidden units, optimized using Adam (Kingma & Ba, 2015) with a learning rate of 0.01 and a L_2 weight decay of 5×10^{-4} for the first layer. We train 1000 epochs and report the training accuracy and test accuracy according to the best validation

---

[4] Given the same degree distribution, we can shuffle to generate multiple graphs with similar degree distribution but completely different semantic meanings.

accuracy. We randomly sample certain percent of nodes from the whole graph and reset their labels. It can be seen from Fig. 5 (b) and (c) that even with all random labels, the training accuracy can reach to nearly 100%, which serves as a strong evidence for the existence of memorization effects in GNNs. In other words, even a GNN is trained on a heavily poisoned graph (changes dramatically in the sense of semantic), it can still achieve good training accuracy while the defender has no way to explicitly find it or do anything about it. That is against to the original setting and purpose of adversarial attacks (Szegedy et al., 2014; Goodfellow et al., 2015; Madry et al., 2018). Thus, it urges the community for a proper solution to the ill-defined unnoticeability in current graph adversarial learning. Till the appearance of silver bullet, evasion attack can serve as a better solution than poisoning attack.

B MORE DEAILS ABOUT GIA AND GMA COMPARISON

B.1 IMPLEMENTATION OF GRAPH MODIFICATION ATTACK

Following Metattack (Zügner & Günnemann, 2019), we implement Graph Modification Attack by taking A as a hyper-parameter. Nevertheless, since we are conducting evasion attack, we do not have meta-gradients but the gradient of A with respect to $\mathcal{L}_{\text{atk}}$, or $\nabla_A\mathcal{L}_{\text{atk}}$. Each step, we take the maximum entry in $\nabla_A\mathcal{L}_{\text{atk}}$, denoted with $\max(\nabla_A\mathcal{L}_{\text{atk}})$, and change the corresponding edge, if it is not contained in the training graph. Then we perform the perturbation as follows:

(a) If $\max(\nabla_A\mathcal{L}_{\text{atk}}) \leq 0$ and the corresponding entry in A is 0, i.e., the edge does not exist before, we will add the edge.

(b) If $\max(\nabla_A\mathcal{L}_{\text{atk}}) \geq 0$ and the corresponding entry in A is 1, i.e., the edge exists before, we will remove the edge.

If the selected entry can not satisfy neither of the above conditions, we will take the left maximum entry to perform the above procedure until we find one that satisfy the conditions. Here we exclude perturbations on node features given limited budgets, since Wu et al. (2019) observed the edge perturbations produce more harm than node perturbations. Besides, as shown in the proof, the damage brought by perturbations on node features is at most the damage brought by a corresponding injection to the targets in GIA, hence when given the same budgets to compare GMA and GIA, we can exclude the perturbations on nodes without loss of generality.

B.2 IMPLEMENTATION OF GRAPH INJECTION ATTACK WITH $\mathcal{M}_2$

GIA with $\mathcal{M}_2$ is implemented based on the GMA above. For each edge appears in the perturbed graph produced by GMA but does not exist in the original graph, in GIA, we will inject a node to connect with the corresponding nodes of the edge. After injecting all of the nodes, then we use PGD (Madry et al., 2018) to optimize the features of the injected nodes.

C MORE HOMOPHILY DISTRIBUTIONS

C.1 EDGE-CENTRIC HOMOPHILY

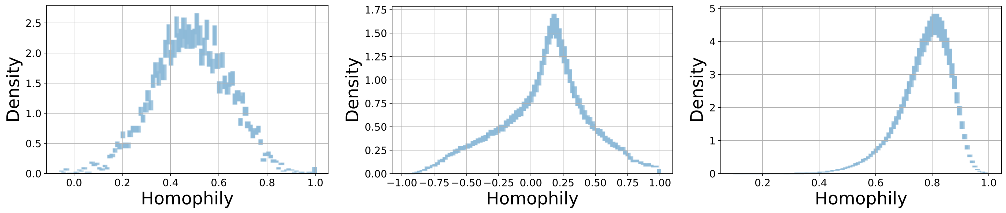
In addition to node-centric homophily (Def. 6), we can also define edge-centric homophily as:

Definition C.1 (Edge-Centric Homophily). *The homophily for an edge (u, v) can be defined as.*

$$h_e = sim(X_u, X_v), \tag{12}$$

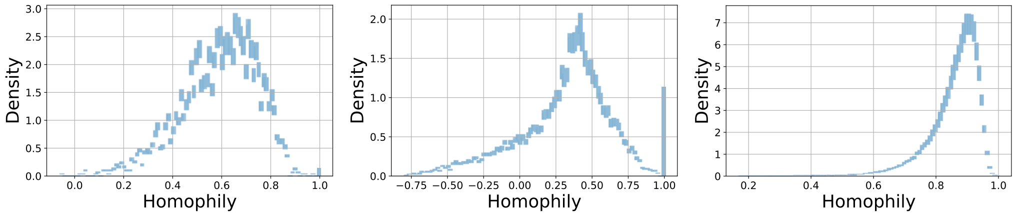
where $sim(\cdot)$ is also a distance metric, e.g., cosine similarity.

With the definition above, we can probe the natural edge-centric homophily distribution of real-world benchmarks, as shown in Fig. 6. It turns out that the edge-centric homophily distributes follows a guassian prior. However, it seems we can not utilize edge-centric homophily to instantiate the homophily unnoticeability for several reasons. On the one hand, edge similarity does not consider the degrees of the neighbors which is misaligned with the popular aggregation scheme of GNN. On the other hand, edge-centric and node-centric homophily basically perform similar functionality to retain the homophily, but if considering the future extension to high-order neighbor relationships, edge similarity might be harder to extend than node-centric homophily. Thus, we utilize the node-centric homophily for most of our discussions.



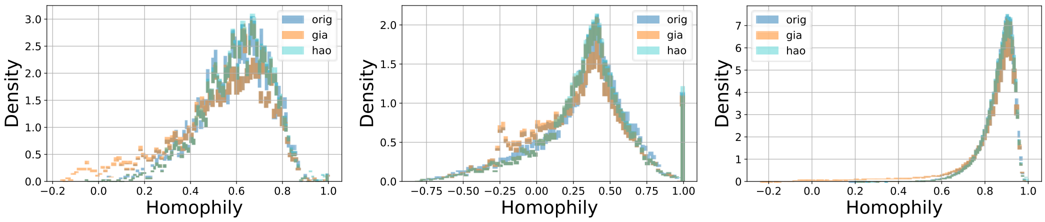
(a) Cora (b) Computers (c) Arxiv

Figure 6: Edge-Centric homophily distributions



(a) Cora (b) Computers (c) Arxiv

Figure 7: Homophily distributions before attack



(a) Cora (b) Computers (c) Arxiv

Figure 8: Homophily distributions after attack

C.2 MORE HOMOPHILY DISTRIBUTIONS CHANGES

We provide more homophily distribution results of the benchmarks we used in the experiments for Cora, Computers and Arxiv, shown as in Fig. 7 and Fig. 8, respectively. GIA is implemented with TDGIA (Zou et al., 2021). Note that the budgets for TDGIA here is different from that in the previous sections, which utilized the budgets resulting in the maximum harm when compared with GMA. Similarly, GIA without HAO would severely break the original homophily distribution hence making GIA can be easily defended by homophily defenders. While incorporated with HAO, GIA would retain the original homophily during attack.

D PROOFS AND DISCUSSIONS OF THEOREMS

D.1 PROOF FOR THEOREM 1

Theorem 1. *Given moderate perturbation budgets* $\triangle_{GIA}$ *for GIA and* $\triangle_{GMA}$ *for GMA, that is, let* $\triangle_{GIA} \leq \triangle_{GMA} \ll |V| \leq |E|$*, for a fixed linearized GNN* f_θ *trained on* $\mathcal{G}$*, assume that* $\mathcal{G}$ *has no isolated nodes, and both GIA and GMA adversaries follow the optimal strategy, then,* $\forall \triangle_{GMA} > 0, \exists \triangle_{GIA} \leq \triangle_{GMA}$*, such that:*

$$\mathcal{L}_{atk}(f_\theta(\mathcal{G}'_{GIA})) - \mathcal{L}_{atk}(f_\theta(\mathcal{G}'_{GMA})) \leq 0,$$

where $\mathcal{G}'_{GIA}$ *and* $\mathcal{G}'_{GMA}$ *are the perturbed graphs generated by GIA and GMA, respectively.*

Proof. The proof sketch is to show that,

(a) There exists a mapping, that when given the same budget, i.e., $\triangle_{\text{GIA}} = \triangle_{\text{GMA}} \ll |V| \leq |E|$, for each perturbation generated by GMA intended to attack node u by perturbing edge (u, v), or node attributes of some node v that connects to u, we can always map it to a corresponding injection attack, that injects node x_w to attack u, and lead to the same effects to the prediction.

(b) When the number of perturbation budget increases, the optimal objective values achieved of GIA is monotonically non-increasing with respect to $\triangle_{\text{GIA}}$, that is

$$\mathcal{L}^{k+1}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}})) \leq \mathcal{L}^{k}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}})),$$

where $\mathcal{L}^{k}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}}))$ is the optimal value achieved under the perturbation budget of k, which is obvious.

Once we prove both (a) and (b), the $\mathcal{L}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}}))$ will approach to $\mathcal{L}^{k}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GMA}}))$ from the below as $\triangle_{\text{GIA}}$ approaches to $\triangle_{\text{GMA}}$, hence proving Theorem 1. Furthermore, for the flexibility of the constraints on X_w, we may adopt the gradient information of X_w with respect to $\mathcal{L}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}}))$ to further optimize X_w and make more damages. Hence, we have $\mathcal{L}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}})) \leq \mathcal{L}^{k}_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GMA}}))$.

Next, we will prove (a) above. Following Wu et al. (2019), in GMA, adding new connections between nodes from different classes produces the most benefit to the adversarial objective. Hence, given the limited perturbation budget, we give our primary focus to the action of connecting nodes from different classes and will prove (a) also holds for the remaining two actions, i.e., edge deletion and node attribute perturbation.

We prove (a) by induction on the number of linearized layers. First of all, we will show prove (a) holds for 1-layer and 2-layer linearized GNN as a motivating example. The model is as $f_\theta = \hat{A}^2X\Theta$ with $H = \hat{A}X\Theta$ and $Z = f_\theta$.

Plural Mapping $\mathcal{M}_2$. Here we define the mapping $\mathcal{M}_2$ for edge addition. For each edge perturbation pair (u, v) generated by GMA, we can insert a new node w to connect u and v. The influence of adversaries can be identified as follows, as Θ is fixed, we may exclude it for simplicity:
In layer (1):

- Clean graph:

$$H_i = \sum_{t \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i d_t}} X_t \tag{13}$$

- GMA:

$$H'_i = \begin{cases} \sum_{t \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_t(d_i+1)}} X_t + \frac{1}{\sqrt{d_v(d_i+1)}} X_v, & i \in \{u\} \\ \sum_{t \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_t(d_i+1)}} X_t + \frac{1}{\sqrt{d_u(d_i+1)}} X_u, & i \in \{v\} \\ H_i, & i \notin \{u, v\} \end{cases} \tag{14}$$

- GIA:

$$H_i'' = \begin{cases} \sum_{t \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_t(d_i+1)}} X_t + \frac{1}{\sqrt{3(d_i+1)}} X_w, & i \in \{u, v\} \\ H_i, & u \notin \{u, v, w\} \\ \frac{1}{\sqrt{3}} (\frac{1}{\sqrt{d_u+1}} X_u + \frac{1}{\sqrt{d_v+1}} X_v + \frac{1}{\sqrt{3}} X_w), & i \in \{w\} \end{cases} \tag{15}$$

where d_i refers to the degree of node i with self-loops added for simplicity. Thus, in layer (1), to make the influence from GMA and GIA on node u equal, the following constraint has to be satisfied:

$$\frac{1}{\sqrt{3(d_u+1)}} X_w = \frac{1}{\sqrt{(d_v+1)(d_u+1)}} X_v, \tag{16}$$

which is trivially held by setting

$$X_w = \frac{\sqrt{3}}{\sqrt{d_v+1}} X_v. \tag{17}$$

Normally, GMA does not consider isolated nodes (Zügner et al., 2018; Zügner & Günnemann, 2019) hence we have $d_v \geq 2$ and $X_w \in \mathcal{D}_X$. Note that we can even change X_w to make more affects to node u with gradient information, then we may generate a more powerful perturbation in this way. Then, we go deeper to layer 2. In layer (2):

- Clean graph:

$$Z_i = \sum_{t \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i d_t}} H_t \tag{18}$$

- GMA:

$$Z_i' = \begin{cases} \sum_{t \in \mathcal{N}(i)} \frac{H_t}{\sqrt{d_t(d_i+1)}} + \frac{H_i'}{d_i+1} + \frac{H_v'}{\sqrt{(d_v+1)(d_i+1)}}, & u \in \{u\} \\ \sum_{t \in \mathcal{N}(i)} \frac{H_t}{\sqrt{d_t(d_i+1)}} + \frac{H_i'}{d_i+1} + \frac{H_u'}{\sqrt{(d_u+1)(d_i+1)}}, & u \in \{v\} \\ \sum_{t \in \mathcal{N}(i)} \frac{H_t'}{\sqrt{d_t(d_i+1)}}, & u \in \mathcal{N}(u) \cup \mathcal{N}(v) \\ Z_u, & \text{otherwise} \end{cases} \tag{19}$$

- GIA:

$$Z_i'' = \begin{cases} \sum_{t \in \mathcal{N}(i)} \frac{1}{\sqrt{d_t(d_i+1)}} H_t + \frac{1}{d_i+1} H_i'' + \frac{1}{\sqrt{3(d_i+1)}} H_w'', & i \in \{u, v\} \\ H_u, & i \notin \{u, v, w\} \\ \frac{1}{\sqrt{3}} (\frac{1}{\sqrt{d_u+1}} H_u'' + \frac{1}{\sqrt{d_v+1}} H_v'' + \frac{1}{\sqrt{3}} H_w''), & i \in \{w\} \end{cases} \tag{20}$$

Similarly, to make $Z'_u = Z''_u$, we have to satisfy the following constraint:

$$\begin{gathered}
\frac{1}{d_u+1}H''_u + \frac{1}{\sqrt{3(d_u+1)}}H''_w = \frac{1}{d_u+1}H'_i + \frac{1}{\sqrt{(d_v+1)(d_u+1)}}H'_v, \\
\frac{\sqrt{3}}{d_u+1}\sum_{t\in\mathcal{N}(u)\cup\{u\}}\frac{1}{\sqrt{d_t}}X_t + \frac{4}{3}X_w + \frac{1}{\sqrt{3}}\left(\frac{1}{\sqrt{d_u+1}}X_u + \frac{1}{\sqrt{d_v+1}}X_v\right) \\
= \\
\frac{\sqrt{3}}{d_u+1}\sum_{t\in\mathcal{N}(u)\cup\{u\}}\frac{X_t}{\sqrt{d_t}} + \frac{\sqrt{3X_v}}{\sqrt{d_v+1}} + \\
\frac{\sqrt{3}}{\sqrt{d_v+1}}\left(\sum_{t\in\mathcal{N}(v)\cup\{v\}}\frac{X_t}{\sqrt{d_t(d_v+1)}} + \frac{X_u}{\sqrt{(d_u+1)(d_v+1)}}\right), \\
\frac{4}{3}X_w + \frac{1}{\sqrt{3}}\left(\frac{1}{\sqrt{d_u+1}}X_u + \frac{1}{\sqrt{d_v+1}}X_v\right) \\
= \\
\frac{\sqrt{3X_v}}{\sqrt{d_v+1}} + \frac{\sqrt{3}}{\sqrt{d_v+1}}\left(\sum_{t\in\mathcal{N}(v)\cup\{v\}}\frac{X_t}{\sqrt{d_t(d_v+1)}} + \frac{X_u}{\sqrt{(d_u+1)(d_v+1)}}\right),
\end{gathered} \tag{21}$$

then we let $X_w = \frac{3}{4}(\text{RHS} - \frac{1}{\sqrt{3}}(\frac{1}{\sqrt{d_u+1}}X_u + \frac{1}{\sqrt{d_v+1}}X_v))$ to get the solution of X_w that makes the same perturbation. Similarly, we can infer $X_w \in \mathcal{D}_X$. The following proof also applies to layer 2.

Next, we will prove that, for a linearized GNN with k layers ($k \geq 1$), i.e., $H^{(k)} = \hat{A}^k X\Theta$, once $\exists X_w$, such that the predictions for node u is the same to that perturbed by GMA, i.e., $H_u^{(k-1)} = E_u^{(k-1)}$, then $\exists X'_w$, such that $H_u^{(k)} = E_u^{(k)}$. Here we use H to denote the prediction of GNN attacked by GMA and E for that of GIA. Note that, once the theorem holds, as we have already proven the existence for such X_w, it naturally generalizes to an arbitrary number of layers.

To be more specific, when $H_u^{(k-1)} = E_u^{(k-1)}$, we need to show that, $\exists X_w$, s.t.,

$$\begin{aligned}
H_u^{(k)} &= \sum_{j\in\mathcal{N}(u)}\frac{1}{\sqrt{d_u+1}\sqrt{d_j}}H_j^{(k-1)} + \frac{1}{d_u+1}H_u^{(k-1)} + \frac{1}{\sqrt{d_u+1}\sqrt{d_v+1}}H_v^{(k-1)}, \\
E_u^{(k)} &= \sum_{j\in\mathcal{N}(u)}\frac{1}{\sqrt{d_u+1}\sqrt{d_j}}E_j^{(k-1)} + \frac{1}{d_u+1}E_u^{(k-1)} + \frac{1}{\sqrt{d_u+1}\sqrt{3}}E_w^{(k-1)}, \\
H_u^{(k)} &= E_u^{(k)}.
\end{aligned} \tag{22}$$

Here we make a simplification to re-write Eq. 22 by defining the influence score.

Definition D.1 (Influence Score)**.** *The influence score from node v to u after k neighbor aggregations with a fixed GNN following Eq. 1, is the weight for X_v contributing to $H_u^{(k)}$:*

$$H_u^{(k)} = \sum_{j\in\mathcal{N}(u)\cup\{u\}} I_{uj}^k \cdot X_j, \tag{23}$$

which can be calculated recursively through:

$$I_{uw}^k = \sum_{j\in\mathcal{N}(u)\cup\{u\}}(I_{uj}\cdot I_{jw}^{(k-1)}) + I_{uw}^{(k-1)}. \tag{24}$$

As Θ is fixed here, we can simply regard $I_{uv}^k = \hat{A}_{uv}^k$. Compared to the predictions after k-th propagation onto the clean graph, in GMA, $H_u^{(k)}$ is additionally influenced by node v, while in GIA, $H_u^{(k)}$ is additionally influenced by node v and node w. Without loss of generality, we may absorb the influence from neighbors of node v into that of node v. Hence we can rewrite Eq. 22 as the following:

$$\begin{aligned} \Delta H_u^{(k)} &= I^k_{\text{GMA}_{uv}} X_v, \\ \Delta E_u^{(k)} &= I^k_{\text{GIA}_{uv}} X_v + I^k_{\text{GIA}_{uw}} X_w, \\ \Delta H_u^{(k)} &= \Delta E_u^{(k)}, \end{aligned} \tag{25}$$

where

$$I^k_{\text{GIA}_{uv}} = \sum_{j \in \mathcal{N}(u)\cup\{u\}} I_{\text{GIA}_{uj}} \cdot I^{(k-1)}_{\text{GIA}_{jv}} + I_{\text{GIA}_{uw}} \cdot I^{(k-1)}_{\text{GIA}_{wv}}.$$

Then we can further simplify it as,

$$(I^k_{\text{GMA}_{uv}} - I^k_{\text{GIA}_{uv}}) X_v = I^k_{\text{GIA}_{uw}} X_w. \tag{26}$$

To show the existence of X_w that solves the above equation, it suffices to show $I^k_{\text{GIA}_{uw}} \neq 0$ and $X_w \in \mathcal{D}_X$. Note that $\exists X_w$ s.t.,

$$(I^{(k-1)}_{\text{GMA}_{uv}} - I^{(k-1)}_{\text{GIA}_{uv}}) X_v = I^{(k-1)}_{\text{GIA}_{uw}} X_w. \tag{27}$$

Since $\hat{A}^k \geq \mathbf{0}, \forall k \geq 0$, so we have $I^{(k-1)}_{\text{GIA}_{uw}} > 0$. Moreover,

$$I^k_{uw} = \sum_{j \in \mathcal{N}(u)\cup\{u\}} (\hat{A}_{uj} \cdot \hat{A}^{(k-1)}_{jw}) + I^{(k-1)}_{uw},$$

then it is obvious that the $I^k_{uw} > 0$. Moreover, with the definition of $I^k_{uv} = \hat{A}^k_{uv}$, it is obvious that $I^{(k-1)}_{\text{GIA}_{uw}} \geq I^{(k-1)}_{\text{GMA}_{uv}}$ for v with a degree not less than 1 (i.e., v is not an isolated node). Hence, we have $(I^{(k-1)}_{\text{GMA}_{uv}} - I^{(k-1)}_{\text{GIA}_{uv}})/I^{(k-1)}_{\text{GIA}_{uw}} \leq 1$ and $X_w \in \mathcal{D}_X$.

Now we have proved (a) holds for linearized GNN. For the remaining actions of GMA, we can also use similar mappings to prove (a). For an edge deletion of (u, v), one may rewrite Eq. 22 for the left nodes other than v. Given a new mapping $\mathcal{M}_1$ that injects one node w to node u, we may also write the equation involving I^k_{uw} and derive the same conclusions similarly. Intuitively, for edge deletion, considering the classification probability, removing an edge is equivalent to enlarge the predicted classification probability for other classes, hence it fictionalizes likewise the edge addition and we can use a similar proof for this action. Besides, $\mathcal{M}_1$ can also apply to the perturbation of features to node u that we inject one node w to make the same effect. Thus, we complete the whole proof. $\square$

Theorem 1 for other GNNs. We can extend Theorem 1 to other GNNs such as GCN, GraphSage, etc. Recall the theorem 1 in Xu et al. (2018):

Lemma 1. *Given a k-layer GNN following the neighbor aggregation scheme via Eq. 1, assume that all paths in the computation graph of the model are activated with the same probability of success p. Then the influence distribution I_x for any node $x \in V$ is equivalent, in expectation, to the k-step random walk distribution on $\tilde{\mathcal{G}}$ starting at node x.*

To apply Lemma 1, we observe that the definition of I^k_{uw} is analogous to random walk starting from node u. Thus, one may replace the definition of I^k_{uw} here to the influence score defined by Xu et al. (2018), conduct a similar proof above with random walk score and obtain the same conclusions, given the mapping $\mathcal{M}_2$, for each edge addition (u, v), $\exists X_w$, such that

$$\mathbb{E}(\mathcal{L}^k_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}}))) = \mathbb{E}(\mathcal{L}^k_{\text{atk}}(f_\theta(\mathcal{G}'_{\text{GIA}}))). \tag{28}$$

Though the original theorem only proves Lemma 1 for GCN and GraphSage, it is obvious one can easily extend the proof in Xu et al. (2018) for aggregation scheme as Eq. 1.

Cases for Less GIA Budget. We can reduce GIA budgets in two ways.

(a) For GMA that performs both node feature perturbation and edge addition, considering a edge perturbation (u, v), $\mathcal{M}_2$ essentially also applies for node feature perturbations on u or v without additional budgets.

(b) It is very likely that with the mapping above, GIA will produce many similar nodes. Hence, with one post-processing step to merge similar nodes together and re-optimize them again, GIA tends to require less budgets to make the same or more harm than GMA. That is also reflected in our experiments as shown in Fig. 1b.

D.2 $\mathcal{M}_2$ FOR MORE GMA OPERATIONS

Here we explain how our theoretical results also apply to the remaining actions, i.e., edge deletion and node feature perturbation, of GMA with $\mathcal{M}_2$ (Def. 3.2). In the proof for Theorem 1, we have proved the existence of mappings for edge removal and node feature perturbation. Once the injected node features are set to have the same influence to the predictions on the targets, they can be further optimized for amplifying the damage, thus all of our theoretical results can be derived similarly like that for edge addition operation.

D.3 PROOF FOR THEOREM 2

Theorem 2. *Given conditions in Theorem 1, consider a GIA attack, which* (i) *is mapped by* $\mathcal{M}_2$ *(Def. 3.2) from a GMA attack that only performs edge addition perturbations, and* (ii) *uses a linearized GNN trained with at least one node from each class in* $\mathcal{G}$ *as the surrogate model, and* (iii) *optimizes the malicious node features with PGD. Assume that* $\mathcal{G}$ *has no isolated node, and has node features as* $X_u = \frac{C}{C-1}e_{Y_u} - \frac{1}{C-1}\mathbf{1} \in \mathbb{R}^d$*, where* Y_u *is the label of node* u *and* $e_{Y_u} \in \mathbb{R}^d$ *is a one-hot vector with the* Y_u*-th entry being* 1 *and others being* 0*. Let the minimum similarity for any pair of nodes connected in* $\mathcal{G}$ *be* $s_{\mathcal{G}} = \min_{(u,v)\in E} sim(X_u, X_v)$ *with* $sim(X_u, X_v) = \frac{X_u \cdot X_v}{\|X_u\|_2 \|X_v\|_2}$*. For a homophily defender* g_θ *that prunes edges* (u, v) *if* $sim(X_u, X_v) \leq s_{\mathcal{G}}$*, we have:*

$$\mathcal{L}_{atk}(g_\theta(\mathcal{M}_2(\mathcal{G}'_{GMA}))) - \mathcal{L}_{atk}(g_\theta(\mathcal{G}'_{GMA})) \geq 0.$$

Proof. We prove Theorem 2 by firstly show the following lemma.

Lemma 2. *Given conditions in Theorem 2, as the optimization on* X_w *with respect to* $\mathcal{L}_{atk}$ *by PGD approaches, we have:*

$$sim(X_u, X_w)^{(t+1)} \leq sim(X_u, X_w)^{(t)},$$

where t *is the number of optimization steps.*

We prove Lemma 2 in the follow-up section, i.e., Appendix D.4. With Lemma 2, known that GIA is mapped from GMA with $\mathcal{M}_2$, X_w will be optimized to have the same effects as GMA at first and continue being optimized to a more harmful state, hence for the unit perturbation case as Fig. 2a, we know:

$$\text{sim}(X_u, X_w) \leq \text{sim}(X_u, X_v), \tag{29}$$

as the optimization on X_w approaches. Furthermore, it follows:

$$h_u^{\text{GIA}} \leq h_u^{\text{GMA}}, \tag{30}$$

where h_u^{GIA} and h_u^{GMA} denote the homophily of node u after GIA and GMA attack, respectively. Now if we go back to the homophily defender g_θ, for any threshold specified to prune the edge (u, v), as Lemma 2 and Eq. 29 indicates, direct malicious edges in GIA are more likely to be pruned by g_θ. Let τ_{GIA} and τ_{GMA} denote the corresponding similarity between (u, w) in GIA and (u, v) in GMA, we have several possibilities compared with $s_{\mathcal{G}} = \min_{(u,v)\in E} \text{sim}(X_u, X_v)$:

(a) $\tau_{\text{GIA}} \leq \tau_{\text{GMA}} \leq s_{\mathcal{G}}$: all the malicious edges will be pruned, Theorem 2 holds;

(b) $\tau_{\text{GIA}} \leq s_{\mathcal{G}} \leq \tau_{\text{GMA}}$: all the GIA edges will be pruned, Theorem 2 holds;

(c) $s_{\mathcal{G}} \leq \tau_{\text{GIA}} \leq \tau_{\text{GMA}}$: this is unlikely to happen, otherwise τ_{GIA} can be optimized to even worse case, Theorem 2 holds;

Thus, we complete our proof. □

Interestingly, we can also set a specific threshold τ_h for homophily defender s.t., $\tau_h - s_{\mathcal{G}} \leq \epsilon \geq 0$, where some of the original edges will be pruned, too. However, some of previous works indicate promoting the smoothness or slightly dropping some edges will bring better performance (Rong et al., 2020; Yang et al., 2021a; Zhao et al., 2021; Yang et al., 2021b). The similar discussion can also be applied to this case and obtain the same conclusions.

D.4 PROOF FOR LEMMA 2

Proof. To begin with, without loss of generality, we may assume the number of classes is 2 and $Y_u = 0$, which can be similarly extended to the case of multi-class. With the feature assignment in the premise, let the label of node u is Y_u, we have:

$$X_u = \begin{cases} [1, -1]^T, & Y_u = 0, \\ [-1, 1]^T, & Y_u = 1. \end{cases} \tag{31}$$

After setting it to having the same influence as that in GMA following Eq. 26, we have:

$$X_w = \frac{(I^k_{\text{GMA}_{uv}} - I^k_{\text{GIA}_{uv}})}{I^k_{\text{GIA}_{uw}}} X_v. \tag{32}$$

Then, let $\mathcal{L}_u$ denote the training loss $\mathcal{L}_{\text{train}}$ on node u, we can calculate the gradient of X_w:

$$\frac{\partial \mathcal{L}_u}{\partial X_u} = \frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} \cdot \frac{\partial H_u^{(k)}}{\partial X_w} = \frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} \cdot I^k_{\text{GIA}_{uw}} \cdot \Theta. \tag{33}$$

With Cross-Entropy loss, we further have:

$$\frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} = [-1, 1]^T. \tag{34}$$

Then, we can induce the update step of optimizing X_w with respect to $\mathcal{L}_{\text{atk}} = -\mathcal{L}_{\text{train}}$ by PGD:

$$X_w^{(t+1)} = X_w^{(t)} + \epsilon\, \text{sign}(I^k_{\text{GIA}_{uw}} \cdot [-1, 1]^T \cdot \Theta), \tag{35}$$

where t is the number of update steps. As the model is trained on at least nodes with indicator features following Eq. 31 from each class, without loss of generality, here we may assume $\Theta \geq \mathbf{0}$, the optimal Θ would converge to $\Theta \geq \mathbf{0}$. Thus,

$$\text{sign}(I^k_{\text{GIA}_{uw}} \cdot [-1, 1]^T \cdot \Theta) = \text{sign}(I^k_{\text{GIA}_{uw}} \cdot [-1, 1]^T).$$

Let us look into the change of cosine similarity between node u and node v as:

$$\Delta\text{sim}(X_u, X_w) = \alpha(X_u \cdot X_w^{(t+1)} - X_u \cdot X_w^{(t)}), \tag{36}$$

where $\alpha \geq 0$ is the normalized factor. To determine the sign of $\Delta\text{sim}(X_u, X_w)$, we may compare $X_u \cdot X^{(t+1)}$ with $X_u \cdot X_w^{(t)}$. Here we expand $X_u \cdot X_w^{(t+1)}$. Let X_{u0}, X_{u1} to denote the first and second element in X_u respectively, we have:

$$\begin{aligned} X_u \cdot X_w^{(t+1)} &= \frac{X_u \cdot X_w + \epsilon\, \text{sign}(I^k_{\text{GIA}_{uw}} \cdot [-1, 1]^T) X_u}{\|X_u\|_2 \cdot \left\| X_w^{(t+1)} \right\|_2}, \\ &= \frac{X_u \cdot X_w + \epsilon(X_{u1} - X_{u0})}{\|X_u\|_2 \sqrt{X_{w0}^2 + X_{w1}^2 + \epsilon^2 + 2\epsilon(X_{w1} - X_{w0})}}, \end{aligned} \tag{37}$$

where we omit the sign of $I^k_{\text{GIA}_{uw}}$ for $I^k_{\text{GIA}_{uw}} \geq 0$ according to the definition. Recall that we let $Y_u = 0$, hence we have $(X_{u1} - X_{u0}) \leq 0$. Besides, following Eq. 26, we have $\text{sign}(X_{w1} - X_{w0}) = \text{sign}(X_{v1} - X_{v0})$. As GMA tend to connect nodes from different classes, we further have $\text{sign}(X_{w1} - X_{w0}) \geq 0$. Comparing to $X_u \cdot X_w^{(t)}$, we know in Eq. 37, the numerator decreases and the denominator increases, as $\epsilon \geq 0$, so the overall scale decreases. In other words, we have:

$$\Delta\text{sim}(X_u, X_w) = \alpha(X_u \cdot X_w^{(t+1)} - X_u \cdot X_w^{(t)}) \leq 0, \tag{38}$$

which means that the cosine similarity between node u and node v decreases as the optimization of X_w with respect to $\mathcal{L}_{\text{atk}}$ processes. Thus, we complete our proof for Lemma 2. ☐

D.5 PROOF FOR THEOREM 3

Theorem 3. *Given conditions as Theorem 2, when $\lambda > 0$, we have $m(\mathcal{H}_{\mathcal{G}}, \mathcal{H}_{\mathcal{G}'_{HAO}}) \leq m(\mathcal{H}_{\mathcal{G}}, \mathcal{H}_{\mathcal{G}'_{GIA}})$, hence:*

$$\mathcal{L}_{atk}(g_\theta(\mathcal{G}'_{HAO})) - \mathcal{L}_{atk}(g_\theta(\mathcal{G}'_{GIA})) \leq 0,$$

where $\mathcal{G}'_{HAO}$ is generated by GIA with HAO, and $\mathcal{G}'_{GIA}$ is generated by GIA without HAO.

Proof. Similar with the proof for Theorem 2, we begin with binary classification, without loss of generality. With the feature assignment in the premise, let the label of node u is Y_u, we have:

$$X_u = \begin{cases} [1,-1]^T, & Y_u = 0, \\ [-1,1]^T, & Y_u = 1. \end{cases} \tag{39}$$

Let $\mathcal{L}_u$ denote the training loss $\mathcal{L}_{\text{train}}$ on node u, we look into the gradient of X_w with respect to $\mathcal{L}_u$:

$$\frac{\partial \mathcal{L}_u}{\partial X_u} = \frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} \cdot \frac{\partial H_u^{(k)}}{\partial X_w} = \frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} \cdot I^k_{\text{GIA}_{uw}} \cdot \Theta. \tag{40}$$

With Cross-Entropy loss, we further have:

$$\frac{\partial \mathcal{L}_u}{\partial H_u^{(k)}} = [-1,1]^T. \tag{41}$$

Together with HAO, we can infer the update step of optimizing X_w with respect to $\mathcal{L}_{\text{atk}} = -\mathcal{L}_{\text{train}} + \lambda C(\mathcal{G}, \mathcal{G}')$ by PGD:

$$X_w^{(t+1)} = X_w^{(t)} + \epsilon\, \text{sign}((I^k_{\text{GIA}_{uw}} \cdot [-1,1]^T + \lambda[1,-1]^T) \cdot \Theta), \tag{42}$$

where t is the number of update steps. Similarly, without loss of generality, we may assume $\Theta \geq \mathbf{0}$. As the optimization approaches, given $\lambda > 0$, GIA with HAO will early stop to some stage that $(I^k_{\text{GIA}_{uw}} \cdot [-1,1]^T + \lambda[1,-1]^T) = \mathbf{0}$, hence similar to the proof of Theorem 2, it follows:

$$h_u^{\text{GIA}} \leq h_u^{\text{HAO}}, \tag{43}$$

where h_u^{GIA} and h_u^{HAO} denote the homophily of node u after GIA and GIA with HAO attack, respectively. Likewise, we can infer that:

$$\mathcal{L}_{\text{atk}}(g_\theta(\mathcal{G}'_{\text{HAO}})) - \mathcal{L}_{\text{atk}}(g_\theta(\mathcal{G}'_{\text{GIA}})) \leq 0.$$

Thus, we complete our proof. ☐

D.6 CERTIFIED ROBUSTNESS OF HOMOPHILY DEFENDER

Here we prove the certified robustness of homophily for a concrete GIA case. We prove via the decision margin as follows:

Definition D.2 (Decision Margin). *Given a k-layer GNN, let $H^{(k)}_{[u,c]}$ denote the corresponding entry in $H^{(k)}_u$ for the class c, the decision margin on node u with class label Y_u can be denoted by:*

$$m_u = H^{(k)}_{[u,y_u]} - \max_{c \in \{0,..,C-1\}} H^{(k)}_{[u,c]}.$$

A Multi-Layer Perceptron (MLP) can be taken as a 0-layer GNN which the definition also applies. Then, we specify the certified robustness as follows:

Proposition D.1 (Certified Robustness of Homophily Defender). *Consider a direct GIA attack uses a linearized GNN trained with at least one node from each class in $\mathcal{G}$, that targets at node u by injecting a node w connecting to u, let node features $x_u = \frac{C}{C-1}onehot(Y_u) - \frac{1}{C-1}\mathbf{1}$, the homophily of u be τ, the decision margin of a MLP on u be γ, the minimum similarity for any pair of nodes connected in the original graph be $s_{\mathcal{G}} = \min_{(u,v)\in E} sim(X_u, X_v)$, homophily defender g_θ can defend such attacks, if $-\alpha\frac{1}{\sqrt{1+1/d_u}}(\tau + \beta\gamma) \leq s_{\mathcal{G}}$, and g_θ prunes edges (u,v) s.t.,*

$$sim(X_u, X_w) \leq -\alpha\sqrt{\frac{1}{1+1/d_u}}(\tau + \beta\gamma),$$

where $\alpha, \beta \geq 0$ are corresponding normalization factors.

Intuitively, effective attacks on a node with higher degrees, homophily or decision margin require a lower similarity between node w and u hence more destruction to the homophily of node u. GIA without any constraints tends to optimize $\text{sim}(X_u, X_w)$ to a even lower value. Thus, it becomes easier to find a suitable condition for g_θ, with which it can painlessly prune all vicious edges while keeping all original edges.

Proof. Analogous to the proof for Lemma 2, without loss of generality, we begin with binary classification, normalized indicator features and $Y_u = 0$ as follows:

$$X_u = \begin{cases} [1, -1]^T, & Y_u = 0, \\ [-1, 1]^T, & Y_u = 1. \end{cases} \tag{44}$$

The decision margin based on k-th layer representation can be denoted by

$$m = H^{(k)}_{[u,y_u]} - \max_{c \in \{0,..,C-1\}} H^{(k)}_{[u,c]}, \tag{45}$$

follows the Definition D.2. In our binary classification case, we have

$$\gamma = H^{(0)}_{[u,0]} - H^{(0)}_{[u,1]}, \tag{46}$$

where $H^{(0)}$ is the output of a 0-layer GNN, or MLP (Multi-Layer Perceptron). A k-layer GNN can be regarded as generating new hidden representation for node u by aggregating its neighbors, hence, we may induce the decision margin for a k-layer GNN at node u as

$$m = H^{(k)}_{[u,0]} - H^{(k)}_{[u,1]} = ([\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} - [\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[1]}) + I^{(k)}_{uu} \gamma, \tag{47}$$

where we can replace the influence from neighbors with homophily of node u. Observe that h_u essentially indicates how much neighbors of node u contribute to $H^{(k)}_{[u,0]}$, for example, in binary case, let $\zeta > 0$ be the corresponding normalization factor,

$$h_u = \frac{1}{\zeta}([\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} [X_u]_{[0]} + [\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[1]} [X_u]_{[1]}),$$

which means,

$$[\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[1]} = \frac{1}{[X_u]_{[1]}} (\zeta h_u - [\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} [X_u]_{[0]}),$$

replaced with $X_u = [1, -1]^T$,

$$\begin{aligned} m &= H^{(k)}_{[u,0]} - H^{(k)}_{[u,1]} \\ &= ([\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} - [\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[1]}) + \\ &= ([\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} - \frac{1}{[X_u]_{[1]}} (\zeta h_u - [\sum_{j \in \mathcal{N}(u)} I_{uj} X_j]_{[0]} [X_u]_{[0]})) + I^{(k)}_{uu} \gamma \\ &= \zeta h_u + I^{(k)}_{uu} \gamma. \end{aligned} \tag{48}$$

Hence, we have:

$$m = H^{(k)}_{[u,0]} - H^{(k)}_{[u,1]} = \zeta h_u + I^{(k)}_{uu} \gamma,$$

where $\zeta \geq 0$ is the factor of h_u. With node w injected, the margin can be rewritten as:

$$m' = \sqrt{\frac{d_u}{d_u + 1}} m + I^{(k)}_{uw} (X_{[w,0]} - X_{[w,1]}). \tag{49}$$

To perturb the prediction of node u, we make $m \leq 0$, hence, we have

$$\begin{aligned} m' = \sqrt{\frac{d_u}{d_u+1}} m + I_{uw}^{(k)}(X_{[w,0]} - X_{[w,1]}) \leq 0, \\ I_{uw}^{(k)}(X_{[w,1]} - X_{[w,0]}) \geq \sqrt{\frac{d_u}{d_u+1}} m, \\ (X_{[w,1]} - X_{[w,0]}) \geq \frac{1}{I_{uw}^{(k)}} \sqrt{\frac{d_u}{d_u+1}} (\zeta h_u + I_{uu}^{(k)} \gamma). \end{aligned} \tag{50}$$

Observe that $\text{sim}(X_u, X_w) = (X_{[w,0]} - X_{[w,1]})$ and $h_u = \tau$, hence, we can write Eq. 50 in a clean form as

$$\text{sim}(X_u, X_w) \leq -\alpha \sqrt{\frac{d_u}{d_u+1}} (\tau + \beta\gamma), \tag{51}$$

where α, β are corresponding normalization factors whose signs are determined by signs of I_{uw}^k and I_{uu}^k respectively. In other words, GIA has to optimize X_w satisfying the above requirement to make the attack *effective*, however, given the premise that all $s_{\mathcal{G}} = \min_{(u,v)\in E} \text{sim}(X_u, X_v) \geq -\alpha\sqrt{\frac{d_u}{d_u+1}}(\tau + \beta\gamma)$, a defense model g_θ will directly prune all of the vicious edges satisfying the above requirement and make the attack *ineffective*, which is exactly what we want to prove. ☐

E MORE IMPLEMENTATIONS OF HOMOPHILY DEFENDER

There are many ways to design homophily defenders, inheriting the spirit of recovering the original homophily. In addition to edge pruning, one could leverage variational inference to learn the homophily distribution or the similarity distribution among neighbors. Then we use adversarial training to train the model to denoise. Similarly, learning to promote the smoothness of the graph can also be leveraged to build homophily defenders (Zhao et al., 2021; Yang et al., 2021a;b). Besides, outlier detection can also be adopted to remove or reduce the aggregation weights of malicious edges or nodes. In the following two subsections, we will present two variants that perform better than GNNGuard (Zhang & Zitnik, 2020).

E.1 DETAILS OF EFFICIENT GNNGUARD

The originally released GNNGuard requires $O(n^2)$ computation for node-node similarity, making it prohibitive to run on large graphs. To this end, we implement an efficient alternative of GNNGuard adopting a similar message passing scheme, let τ be the threshold to prune an edge:

$$H_u^{(k)} = \sigma(W_k \cdot \sum_{j\in\mathcal{N}(u)\cup\{u\}} \alpha_{uj} H_j^{(k-1)}), \tag{52}$$

where

$$\alpha_{uj} = \text{softmax}(\frac{z_{uj}}{\sum_{v\in\mathcal{N}(u)\cup\{u\}} z_{uv}}),$$

and

$$z_{uj} = \begin{cases} \dfrac{\mathbf{1}\{\text{sim}(H_j^{(k-1)}, H_u^{(k-1)}) > \tau\} \cdot \text{sim}(H_j^{(k-1)} \cdot H_u^{(k-1)})}{\sum_{v\in\mathcal{N}(u)} \mathbf{1}\{\text{sim}(H_v^{(k-1)}, H_u^{(k-1)}) > \tau\} \cdot \text{sim}(H_v^{(k-1)}, H_u^{(k-1)})}, & u \neq j, \\ \dfrac{1}{\sum_{v\in\mathcal{N}(u)} \mathbf{1}\{\text{sim}(H_v^{(k-1)} \cdot H_u^{(k-1)}) > \tau\} + 1} & u = j. \end{cases}$$

Essentially, it only requires $O(E)$ complexity. We will present the performance of Efficient GNNGuard (EGNNGuard) in table 3.

E.2 DETAILS OF ROBUST GRAPH ATTENTION NETWORK (RGAT)

We introduce another implementation of Robust Graph Attention Network (RGAT). We adopt the same spirit of GCNGuard (Zhang & Zitnik, 2020), that eliminates unlike neighbors during message passing based on neighbor similarity. Specifically, we change the standard GAT (Veličković et al., 2018) attention mechanism as

$$\alpha_{i,j} = \frac{\mathbb{1}\{\text{sim}(x_i, x_j) \geq \tau\}}{\sum_{k\in\mathcal{N}(i)\cup\{i\}} \mathbb{1}\{\text{sim}(x_i, x_k) \geq \tau\}},$$

Additionally, we also adopt the idea of RobustGCN (Zhu et al., 2019) that stabilize the hidden representations between layers, so we add Layer Normalization (Ba et al., 2016) among layers of RGAT. Empirical experiments show that RGAT is a more robust model with or without GIA attacks. For more details, we refer readers to Table 3.

E.3 PERFORMANCE OF HOMOPHILY DEFENDERS

Table 3: Performance of homophily defenders used in experiments

Model	**Natural Accuracy**	**Test Robustness**	**Running Time**
GNNGuard	83.58	64.96	1.76×10^{-3}
EGNNGuard	84.45	64.27	5.39×10^{-5}
RGAT	85.75	66.57	6.03×10^{-5}
GCN	84.99	36.62	5.87×10^{-5}

We test the performance of different homophily defenders on Cora. Natural Accuracy refers to the test accuracy on clean graph. Test Robustness refers to their averaged performance against all the attacks. Running time refers to their averaged running time for one training epoch. We repeat the evaluation 10 times to obtain the average accuracy. We can see that EGNNGuard has competitive performance with GNNGuard while $20\times$ faster. RGAT performs slightly better and $10\times$ faster. Hence, for large graphs and adversarial training of GNNGuard, we will use EGNNGuard instead.

F MORE DETAILS ABOUT ALGORITHMS USED

Here we provide detailed descriptions of algorithms mentioned in Section. 4.2.

F.1 DETAILS OF METAGIA AND AGIA

F.1.1 INDUCTION OF META GRADIENTS FOR METAGIA

With the bi-level optimization formulation of GIA, similar to meta-attack, we can infer the meta-gradients as follows:

$$\nabla_{A_{\text{atk}}}^{meta} = \nabla_{A_{\text{atk}}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X^*_{\text{atk}})), \quad X^*_{\text{atk}} = \text{opt}_{X_{\text{atk}}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}})). \tag{53}$$

Consider the opt process, we have

$$X_{\text{atk}}^{(t+1)} = X_{\text{atk}}^{(t)} - \alpha \nabla_{X_{\text{atk}}^{(t)}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}}^{(t)})). \tag{54}$$

With that, we can derive the meta-gradient for A_{atk}:

$$\begin{aligned} \nabla_{A_{\text{atk}}}^{\text{meta}} &= \nabla_{A_{\text{atk}}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X^*_{\text{atk}})) \\ &= \nabla_{X_{\text{atk}}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}}^{(t)})) \cdot [\nabla_{A_{\text{atk}}} f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}}^{(t)}) + \nabla_{X_{\text{atk}}^{(t)}} f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}}^{(t)}) \cdot \nabla_{A_{\text{atk}}} X_{\text{atk}}^{(t)}], \end{aligned} \tag{55}$$

where

$$\nabla_{A_{\text{atk}}} X_{\text{atk}}^{(t+1)} = \nabla_{A_{\text{atk}}} X_{\text{atk}}^{(t)} - \alpha \nabla_{A_{\text{atk}}} \nabla_{X_{\text{atk}}^{(t)}} \mathcal{L}_{\text{atk}}(f_{\theta^*}(A_{\text{atk}}, X_{\text{atk}}^{(t)})). \tag{56}$$

Note that $X_{\text{atk}}^{(t)}$ depends on A_{atk} according to Eq. 54, so the derivative w.r.t. A_{atk} need to be traced back. Finally, the update schema for A_{atk} is as follows:

$$A_{\text{atk}}^{(t+1)} = A_{\text{atk}}^{(t)} - \beta \nabla_{A_{\text{atk}}^{(t)}}^{\text{meta}}. \tag{57}$$

Directly computing the meta gradients is expensive, following Metattack, we adopt approximations like MAML (Finn et al., 2017) for efficiency consideration. We refer readers to the paper of Metattack for the detailed algorithms by replacing the corresponding variables with those above.

F.2 DETAILS OF AGIA

For optimizing weights of edge entries in A_{atk}, we can use either Adam (Kingma & Ba, 2015), PGD (Madry et al., 2018) or other optimization methods leveraging gradients. For simplicity, we use PGD to illustrate the algorithm description of AGIA as follows:

```
Algorithm 1: AGIA: Adaptive Graph Injection Attack with Gradient
Input: A graph $\mathcal{G}=(A,X)$, a trained GNN model $f_{\theta^*}$, number of injected nodes $c$, degree
       budget $b$, outer attack epochs $e_{\text{outer}}$, inner attack epochs for node features and adjacency
       matrix $e^X_{\text{inner}}$, $e^A_{\text{inner}}$, learning rate $\eta$, weight for sparsity penalty $\beta$, weight for homophily
       penalty $\lambda$ ;
Output: Perturbed graph $\mathcal{G}'=(A',X')$;
Random initialize injection parameters ($A_{\text{atk}}, X_{\text{atk}}$);
$\boldsymbol{Y}_{\text{orig}} \leftarrow f_{\theta^*}(A,X)$ ;   /* Obtain original predictions on clean graph */
for $epoch \leftarrow 0$ to $e_{outer}$ do
    Random initialize $X_{\text{atk}}$;
    for $epoch \leftarrow 0$ to $e^X_{inner}$ do
        $A' \leftarrow A \,\|\, A_{\text{atk}}, X' \leftarrow X \,\|\, X_{\text{atk}}$ ;
        $X_{\text{atk}} \leftarrow \text{Clip}_{(x_{\min},x_{\max})}(X_{\text{atk}} - \eta \cdot \nabla_{X_{\text{atk}}}(\mathcal{L}^h_{\text{atk}}))$ ;
    for $epoch \leftarrow 0$ to $e^A_{inner}$ do
        $A' \leftarrow A \,\|\, A_{\text{atk}}, X' \leftarrow X \,\|\, X_{\text{atk}}$ ;
        $A_{\text{atk}} \leftarrow \text{Clip}_{(0,1)}(A_{\text{atk}} - \eta \cdot \nabla_{A_{\text{atk}}}(\mathcal{L}^A_{\text{atk}}))$ ;
    $A_{\text{atk}} \leftarrow \|_{i=1}^{k} \arg\max_{\text{top } b}(A_{\text{atk}[i,:]})$ ;
```

Here, $\mathcal{L}^h_{\text{atk}}$ refers to the objective of GIA with HAO for the optimization of X_{atk}. For the optimization of A_{atk}, we empirically find the λ_A would degenerate the performance, which we hypothesize that is because of the noises as A_{atk} is a discrete variable. Hence, we set $\lambda_A = 0$ in our experiments. Additionally, we introduce a sparsity regularization term for the optimization of A_{atk}:

$$\mathcal{L}^A_{\text{atk}} = \mathcal{L}_{\text{atk}} + \beta \frac{1}{|V_{\text{atk}}|} \sum_{u \in V_{\text{atk}}} |b - \|A_{\text{atk}_{u,:}}\|_1|. \tag{58}$$

Besides, we empirically observe that Adam performs better than PGD. Hence, we would use Adam for AGIA in our experiments, and leave other methods for future work. Adopting Adam additionally brings the benefits to utilize momentum and history information to accelerate the optimization escape from the local optimum, which PGD fails to achieve.

F.3 DETAILS OF SEQGIA

Since gradient methods require huge computation overhead, we propose a novel divide-and-conquer strategy to iteratively select some of the most vulnerable targets with Eq. 11 to attack. Note that it is different from traditional sequential injection methods which still connect the targets in full batch.

For simplicity, we also illustrate the algorithm with PGD, and one may switch to other optimizer such as Adam to optimize A_{atk}. The detailed algorithm is as follows:

Algorithm 2: SeqGIA: Sequential Adaptive Graph Injection Attack

```
Input: A graph $\mathcal{G} = (A, X)$, a trained GNN model $f_{\theta^*}$, number of injected nodes $k$, degree
    budget $b$, outer attack epochs $e_{\text{outer}}$, inner attack epochs for node features and adjacency
    matrix $e^X_{\text{inner}}, e^A_{\text{inner}}$, learning rate $\eta$, weight for sparsity penalty $\beta$, weight for homophily
    penalty $\lambda$, sequential step for vicious nodes $\gamma_{\text{atk}}$, sequential step for target nodes $\gamma_c$ ;
Output: Perturbed graph $\mathcal{G}' = (A', X')$;
Initialize injection parameters $(A_{\text{atk}}, X_{\text{atk}})$;
$\boldsymbol{Y}_{\text{orig}} \leftarrow f_{\theta^*}(A, X)$ ;   /* Obtain original predictions on clean graph */
while Not Injecting All Nodes do
    $n_{\text{atk}} \leftarrow \gamma_{\text{atk}} * |V_{\text{atk}}|$; $n_c \leftarrow \gamma_c * |V_c|$ ;
    Ranking and selecting $n_c$ targets with Eq. 11;
    Random initialize $A^{(\text{cur})}_{\text{atk}} \in \mathbb{R}^{n_c \times n_{\text{atk}}}, X^{(\text{cur})}_{\text{atk}} \in \mathbb{R}^{n_{\text{atk}} \times d}$ ;
    for epoch $\leftarrow 0$ to $e_{outer}$ do
        for epoch $\leftarrow 0$ to $e^X_{inner}$ do
            $A' \leftarrow A \parallel A_{\text{atk}} \parallel A^{(\text{cur})}_{\text{atk}}, X' \leftarrow X \parallel X_{\text{atk}} \parallel X^{(\text{cur})}_{\text{atk}}$;
            $X^{(\text{cur})}_{\text{atk}} \leftarrow \text{Clip}_{(x_{\min}, x_{\max})}(X^{(\text{cur})}_{\text{atk}} - \eta \cdot \nabla_{X^{(\text{cur})}_{\text{atk}}}(\mathcal{L}^h_{\text{atk}}))$ ;
        for epoch $\leftarrow 0$ to $e^A_{inner}$ do
            $A' \leftarrow A \parallel A_{\text{atk}} \parallel A^{(\text{cur})}_{\text{atk}}, X' \leftarrow X \parallel X_{\text{atk}} \parallel X^{(\text{cur})}_{\text{atk}}$;
            $A^{(\text{cur})}_{\text{atk}} \leftarrow \text{Clip}_{(0,1)}(A^{(\text{cur})}_{\text{atk}} - \eta \cdot \nabla_{A^{(\text{cur})}_{\text{atk}}}(\mathcal{L}^A_{\text{atk}}))$ ;
        $A^{(\text{cur})}_{\text{atk}} \leftarrow \|_{i=1}^{n_{\text{atk}}} \arg\max_{\text{top } b}(A^{(\text{cur})}_{\text{atk}[i,:]})$ ;
    $A_{\text{atk}}$=$A_{\text{atk}} \parallel A^{(\text{cur})}_{\text{atk}}$; $X_{\text{atk}}$=$X_{\text{atk}} \parallel X^{(\text{cur})}_{\text{atk}}$;
```

Actually, one may also inject few nodes via heuristic based algorithms first, then inject the left nodes with gradients sequentially. Assume that α nodes are injected by heuristic, we may further optimize the complexity from

$$O(\frac{1}{\gamma_{\text{atk}}}(|V_c|\log|V_c| + e_{\text{outer}}(e^A_{\text{inner}}|V_c|\gamma_c|V_{\text{atk}}| + e^X_{\text{inner}}|V_{\text{atk}}|d))N_{V_c})$$

to

$$\begin{aligned} O(\alpha\frac{1}{\gamma_{\text{atk}}}(|V_c|\log|V_c| &+ |V_{\text{atk}}|b + e^X_{\text{inner}}|V_{\text{atk}}|d)N_{V_c} + \\ (1-\alpha)\frac{1}{\gamma_{\text{atk}}}(|V_c|\log|V_c| &+ e_{\text{outer}}(e^A_{\text{inner}}|V_c|\gamma_c|V_{\text{atk}}| + e^X_{\text{inner}}|V_{\text{atk}}|d))N_{V_c}) \end{aligned}$$

in Table 7.

G MORE DETAILS ABOUT THE EXPERIMENTS

G.1 STATISTICS AND BUDGETS OF DATASETS

Here we provide statistics of datasets used in the experiments as Sec. 5.1. The label homophily utilizes the previous homophily definition (Zhu et al., 2020), while the avg. homophily utilizes the node-centric homophily based on node similarity.

Table 4: Statistics of datasets

Datasets	**Nodes**	**Edges**	**Classes**	**Avg. Degree**	**Label Homophily**	**Avg. Homophily**
Cora	2680	5148	7	3.84	0.81	0.59
Citeseer	3191	4172	6	2.61	0.74	0.90
Computers	$13,752$	$245,861$	10	35.76	0.77	0.31
Arxiv	$169,343$	$1,166,243$	40	13.77	0.65	0.86
Aminer	$659,574$	$2,878,577$	18	8.73	0.65	0.38
Reddit	$232,965$	$11,606,919$	41	99.65	0.78	0.31

Following previous works (Zou et al., 2021; Zheng et al., 2021), we heuristically specify the budgets for each dataset acoording the the number of target nodes and average degrees.

Table 5: Budgets for non-targeted attacks on different datasets

Datasets	**Nodes**	**Degree**	**Node Per.(%)**	**Edge Per.(%)**
Cora	60	20	2.24%	23.31%
Citeseer	90	10	2.82%	21.57%
Computers	300	150	2.18%	18.30%
Arxiv	1500	100	0.71%	10.29%

For targeted attack, as we select 800 nodes as targets, where 200 of highest classification margin, 200 of lowest classification margin and 400 randomly following previous works (Zügner et al., 2018). We also reduce the budgets accordingly.

Table 6: Budgets of targeted attacks on different datasets

Datasets	**Nodes**	**Degree**	**Node Per.(%)**	**Edge Per.(%)**
Computers	100	150	0.73%	6.1%
Arxiv	120	100	0.07%	1.03%
Aminer	150	50	0.02%	0.26%
Reddit	300	100	0.13%	0.26%

G.2 COMPLEXITY OF ALGORITHMS

Here we provide complexity analysis of GIA algorithms used in the experiments as Sec. 5.1. As defined in previous sections, e^X_{inner} is the number of epochs optimized for node features, b is the number of maximum degree of vicious nodes, d is the number of feature dimension, N_{V_c} is the number of k-hop neighbors of the victim nodes for perform one forwarding of a k-layer GNN, e_{outer} is the number of epochs for optimizing A_{atk}, γ_c is the ratio of target nodes to attack in one batch, γ_{atk} is the ratio of vicious nodes to inject in one batch.

G.3 DETAILS OF BASELINES

Here we provide the categories of defense models used in the experiments as Sec. 5.1. We categorize all models into Vanilla, Robust and Extreme Robust (Combo). Basically, popular GNNs are belong to vanilla category, robust GNNs are belong to robust categorty, and a robust trick will enhance the

Table 7: Complexity of various attacks

Type	Algorithm	Time Complexity	Space Complexity
Gradient	MetaGIA	$O(\|V_{\text{atk}}\|b(\|V_c\|\|V_{\text{atk}}\|\log(\|V_c\|\|V_{\text{atk}}\|)+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c})))$	$O(\|V_c\|\|V_{\text{atk}}\|+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$
	AGIA	$O(e_{\text{outer}}(e_{\text{inner}}^{A}\|V_c\|\|V_{\text{atk}}\|+(e_{\text{inner}}^{A}+e_{\text{inner}}^{X})d(N_{V_c}+\|V_{\text{atk}}\|)))$	$O(\|V_c\|\|V_{\text{atk}}\|+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$
	AGIA-SeqGIA	$O(e_{\text{outer}}(\|V_c\|\log(\|V_c\|)+e_{\text{inner}}^{A}\gamma_c\|V_c\|\|V_{\text{atk}}\|+(e_{\text{inner}}^{A}+e_{\text{inner}}^{X})d(N_{V_c}+\|V_{\text{atk}}\|)))$	$O(\gamma_c\|V_c\|\gamma_{\text{atk}}\|V_{\text{atk}}\|+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$
Heuristic	PGD	$O(\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$	$O(\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$
	TDGIA	$O((\|V_c\|\log\|V_c\|+\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$	$O(\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$
	ATDGIA	$O(\|V_c\|\log\|V_c\|+\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$	$O(\|V_{\text{atk}}\|b+e_{\text{inner}}^{X}d(\|V_{\text{atk}}\|+N_{V_c}))$

robust level by one to the next Category. Consistenly to the observation in GRB (Zheng et al., 2021), we find adding Layer Normalization (Ba et al., 2016) before or between convulotion layers can enhance the model robustness. We use LN to denote adding layer norm before the first convulotion layer and LNi to denote adding layer norm between convulotion layers.

Table 8: Defense model categories

Model	Category	Model	Category	Model	Category	Model	Category
GCN	Vanilla	GCN+LN	Robust	GCN+LNi	Robust	GCN+FLAG	Robust
GCN+LN+LNi	Combo	GCN+FLAG+LN	Combo	GCN+FLAG+LNi	Combo	GCN+FLAG+LN+LNi	Combo
Sage	Vanilla	Sage+LN	Robust	Sage+LNi	Robust	Sage+FLAG	Robust
Sage+LN+LNi	Combo	Sage+FLAG+LN	Combo	Sage+FLAG+LNi	Combo	Sage+FLAG+LN+LNi	Combo
GAT	Vanilla	GAT+LN	Robust	GAT+LNi	Robust	GAT+FLAG	Robust
GAT+LN+LNi	Combo	GAT+FLAG+LN	Combo	GAT+FLAG+LNi	Combo	GAT+FLAG+LN+LNi	Combo
Guard	Robust	Guard+LN	Combo	Guard+LNi	Combo	EGuard+FLAG	Combo
Guard+LN+LNi	Combo	EGuard+FLAG+LN	Combo	EGuard+FLAG+LNi	Combo	EGuard+FLAG+LN+LNi	Combo
RGAT	Robust	RGAT+LN	Combo	RGAT+FLAG	Combo	RGAT+FLAG+LN	Combo
RobustGCN	Robust	RobustGCN+FLAG	Combo				

G.4 DETAILS OF EVALUATION AND MODEL SETTINGS

G.4.1 MODEL SETTING

By default, all GNNs used in our experiments have 3 layers, a hidden dimension of 64 for Cora, Citeseer, and Computers, a hidden dimension of 128 for the rest medium to large scale graphs. We also adopt dropout (Srivastava et al., 2014) with dropout rate of 0.5 between each layer. The optimizer we used is Adam (Kingma & Ba, 2015) with a learning rate of 0.01. By default, we set total training epochs as 400 and employ the early stop of 100 epochs according to the validation accuracy. For the set of threshold in homophily defenders, we use PGD (Madry et al., 2018) to find the threshold which performs well on both the clean data and perturbed data. By default, we set the threshold as 0.1, while for Computers and Reddit, we use 0.15 for Guard and EGuard, and for Citeseer and Arxiv we use 0.2 for RGAT.

For adversarial training with FLAG (Kong et al., 2020), we set the step size be 1×10^{-1}, and train 100 steps for Cora, 50 steps for Citeseer, 10 steps for the rest datasets. We empirically observe that FLAG can enhance both the natural accuracy and robustness of GNNs. We refer readers to the results for more details in Sec. H.2 and Sec. H.3.

G.4.2 EVALUATION SETTING

For final model selection, we select the final model with best validation accuracy. For data splits, we follow the split methods in GRB (Zheng et al., 2021) which splits the datasets according to the node degrees. For non-targeted attack, following previous works (Zou et al., 2021; Zheng et al., 2021), we select all test nodes as targets. While for targeted attacks, we follow previous works (Zügner et al., 2018) to select 200 nodes with highest classification margin and lowest classification margin of the surrogate model. Then we randomly select 400 nodes as targets. In other words, there are 800 target nodes in total for targeted attack. Note for targeted attack, the natural accuracy on the target nodes might be different from normal test accuracy. We also follow previous works to specify the attack budgets as Table. 5 for non-targeted attack and Table. 6 for targeted attack.

During evaluation, we firstly use the surrogate model to generate the perturbed graph then let the target models which has trained on the clean graph to test on the perturbed graph. We repeat the evaluation for 10 times on Cora, Citeseer, Computers, and Arxiv, and 5 times for Aminer and Reddit since model performs more stably on large graphs. Then we report mean test accuracy of the target models on the target nodes due to the space limit.

G.4.3 ATTACKS SETTING

By default, we use PGD (Madry et al., 2018) to generate malicious node features. The learning step is 0.01 and the default training epoch is 500. We also employ the early stop of 100 epochs according to the accuracy of the surrogate model on the target nodes. As Table. 7 shows, MetaGIA requires huge amount of time originally. To scale up, we use a batch update which updates the injected edges by a step size of b, i.e., the maximum degree of injected nodes, and limit the overall update epochs by $|V_{\text{atk}}|/6$, where we empirically observe this setting performs best in Cora and stick it onto other datasets.

For the setting of λ for HAO, we search the parameters within 1 to 8 by a step size of 0.5 such that the setting of λ will not degenerate the performance of the attacks on surrogate model. Besides heuristic approaches, we additionally use a hinge loss to stabilize the gradient information from $\mathcal{L}_{\text{atk}}$ and $C(\mathcal{G}, \mathcal{G}')$, where the former can be too large that blurs the optimization direction of the latter. Take Cross Entropy with $\log$_softmax as an example, we adopt the following to constrict the magnitude of $\mathcal{L}_{\text{atk}}$:

$$\begin{aligned}\mathcal{L}_{\text{atk}[u]} = (-H^{(k)}_{[u,Y_u]}) \cdot \mathbf{1}\{\frac{\exp(H^{(k)}_{[u,Y_u]})}{\sum_i \exp(H^{(k)}_{[u,i]})} \geq \tau\} \\ + \log(\sum_i \exp(H^{(k)}_{[u,i]} \cdot \mathbf{1}\{\frac{\exp(H^{(k)}_{[u,i]})}{\sum_j \exp(H^{(k)}_{[u,j]})} \geq \tau\})),\end{aligned} \tag{59}$$

where $\mathbf{1}\{\frac{\exp(H^{(k)}_{[u,Y_u]})}{\sum_i \exp(H^{(k)}_{[u,i]})} \geq \tau\}$ can be taken as the predicted probability for $Y_u = u$ and τ is the corresponding threshold for hinge loss that we set as 1×10^{-8}.

For the hyper-parameter setting of our proposed strategies in Sec. 4.2, we find directly adopting λ in PGD for λ_X and setting $\lambda_A = 0$ performs empirically better. Hence we stick to the setting for λ_A and λ_X. For the weight of sparsity regularization term in AGIA, we directly adopt $1/b$. For the hyper-parameters in heuristic methods, we directly follow TDGIA (Zou et al., 2021). For SeqGIA, we set γ_{atk} be $\min(0.2, \lfloor |V_c|/2b \rfloor)$ and $\gamma_c = \min(|V_c|, \gamma_{\text{atk}}|V_{\text{atk}}|b)$ by default.

G.5 SOFTWARE AND HARDWARE

We implement our methods with PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). We ran our experiments on Linux Servers with 40 cores Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 256 GB Memory, and Ubuntu 18.04 LTS installed. One has 4 NVIDIA RTX 2080Ti graphics cards with CUDA 10.2 and the other has 2 NVIDIA RTX 2080Ti and 2 NVIDIA RTX 3090Ti graphics cards with CUDA 11.3.

H DETAILED RESULTS OF ATTACK PERFORMANCE

H.1 FULL RESULTS OF AVERAGED ATTACK PERFORMANCE

In this section, we provide full results of averaged attack performance across all defense models, as a supplementary for Table 3b.

Table 9: Full Averaged performance across all defense models

Model	**Cora**†	**Citeseer**†	**Computers**†	**Arxiv**†	**Arxiv**‡	**Computers**‡	**Aminer**‡	**Reddit**‡
Clean	84.74	74.10	92.25	70.44	70.44	91.68	62.39	95.51
PGD	61.09	54.08	61.75	54.23	36.70	62.41	26.13	62.72
+HAO	56.63	48.12	59.16	45.05	28.48	59.09	22.15	56.99
MetaGIA	60.56	53.72	61.75	53.69	28.78	62.08	32.78	60.14
+HAO	58.51	47.44	60.29	48.48	24.61	58.63	29.91	**54.14**
AGIA	60.10	54.55	60.66	48.86	32.68	61.98	31.06	59.96
+HAO	**53.79**	48.30	**58.71**	48.86	29.52	**58.37**	26.51	56.36
TDGIA	66.86	52.45	66.79	49.73	31.68	62.47	32.37	57.97
+HAO	65.22	46.61	65.46	49.54	**22.04**	59.67	22.32	54.32
ATDGIA	61.14	49.46	65.07	46.53	32.08	64.66	24.72	61.25
+HAO	58.13	**43.41**	63.31	**44.40**	29.24	59.27	**17.62**	56.90

The lower is better. †Non-targeted attack. ‡Targeted attack.

Table 10: Detailed results of non-targeted attacks on Cora (1)

	EGuard+LNi+FLAG+LN	EGuard+FLAG+LN	EGuard+LNi+FLAG	Guard+LNi+LN	RGAT+FLAG+LN	GCN+LNi+FLAG+LN	RobustGCN+FLAG	RGAT+LN	Guard+LN	EGuard+FLAG
Clean	83.48	84.17	85.9	79.56	87.29	86.37	86.21	85.29	81.72	85.56
PGD	82.53	83.94	85.74	79.74	76.78	71.10	69.57	79.56	81.44	85.35
+HAO	77.99	73.04	66.25	74.21	68.09	71.06	70.3	67.92	68.12	53.99
MetaGIA	82.68	83.96	85.86	79.51	75.18	69.72	69.4	78.04	81.59	85.48
+HAO	69.49	65.92	66.83	63.02	66.38	71.86	76.8	57.75	55.35	56.77
AGIA	82.75	83.69	85.78	79.56	75.77	69.25	69.10	79.10	81.43	85.34
+HAO	75.25	69.10	61.00	70.12	65.48	69.86	71.08	62.76	60.96	48.54
TDGIA	83.13	83.65	85.72	79.13	82.37	79.31	76.11	82.2	81.37	85.39
+HAO	77.93	73.58	75.47	73.67	75.18	79.45	78.63	69.58	64.66	65.31
ATDIGA	82.57	83.54	85.39	79.38	78.76	76.09	73.08	79.8	81.47	84.88
+HAO	74.43	71.88	71.21	66.97	72.51	76.87	76.17	60.61	62.38	63.53
AVG	79.29	77.86	77.74	74.99	74.89	74.63	74.22	72.96	72.77	72.74

H.2 Detailed Results of Non-Targeted Attacks

In this section, we present the detailed non-targeted attack results of the methods and datasets used in our experiments for Table 1. For simplicity, we only give the results of top 20 robust models according to the averaged test accuracy against all attacks.

H.3 Detailed Results of Targeted Attacks

In this section, we present the detailed targeted attack results of the methods and datasets used in our experiments for Table 2. For simiplicity, we only give the results of top 20 robust models according to the averaged test accuracy against all attacks.

Table 11: Detailed results of non-targeted attacks on Cora (2)

	RGAT+FLAG	Guard+LNi	RobustGCN	GCN+FLAG+LN	GCN+LNi+FLAG	RGAT	GAT+LNi+FLAG+LN	Sage+LNi+FLAG+LN	Guard	GCN+LNi+LN
Clean	87.21	83.18	84.63	85.86	86.36	85.74	86.55	84.95	83.61	84.47
PGD	76.93	83.11	63.20	62.55	60.68	79.28	61.29	61.84	83.08	58.46
+HAO	62.35	53.68	62.60	63.60	61.69	52.60	62.81	62.34	44.02	58.78
MetaGIA	75.14	83.08	63.53	59.18	60.36	77.97	57.88	61.01	83.61	58.10
+HAO	61.53	57.31	69.83	67.00	66.64	49.25	65.82	65.69	45.41	61.94
AGIA	76.04	83.08	62.67	61.26	59.09	78.95	57.84	58.61	83.44	57.05
+HAO	57.17	49.12	61.59	62.65	59.25	47.24	59.80	59.56	39.87	55.62
TDGIA	82.02	83.04	71.34	71.35	73.47	81.79	71.52	70.30	83.44	70.69
+HAO	70.52	67.04	73.38	73.52	75.00	56.95	71.96	71.56	50.79	72.90
ATDIGA	79.06	82.85	66.96	69.61	65.89	79.91	65.57	63.81	83.07	62.95
+HAO	64.50	55.13	70.30	72.46	70.94	42.18	69.26	67.59	40.46	65.53
AVG	72.04	70.97	68.18	68.09	67.22	66.53	66.39	66.11	65.53	64.23

Table 12: Detailed results of non-targeted attacks on Citeseer (1)

	RGAT+LN	RGAT+FLAG+LN	EGuard+LNi+FLAG+LN	Guard+LNi+LN	RGAT	EGuard+FLAG+LN	RGAT+FLAG	EGuard+LNi+FLAG	EGuard+FLAG	GCN+LNi+FLAG+LN
Clean	74.82	75.72	75.44	74.25	74.85	73.64	75.56	74.75	73.57	75.67
PGD	71.00	71.32	75.19	74.21	69.33	73.55	69.84	74.83	73.57	57.97
+HAO	71.00	70.82	66.07	73.04	69.05	61.55	65.78	50.01	47.54	58.77
MetaGIA	70.32	70.21	75.15	74.21	68.42	73.55	68.90	74.83	73.57	56.36
+HAO	70.37	69.77	64.00	71.25	68.04	59.94	63.10	49.70	46.95	57.17
AGIA	71.45	70.51	75.29	74.21	70.31	73.60	69.40	74.83	73.61	56.50
+HAO	71.80	70.70	64.54	70.58	70.24	59.32	62.31	50.33	46.77	58.02
TDGIA	72.29	73.81	75.26	74.21	70.99	73.55	73.34	74.85	73.57	63.01
+HAO	72.51	70.18	68.04	56.69	60.91	65.70	53.99	56.73	52.86	66.52
ATDIGA	72.23	72.82	75.12	74.21	70.61	73.55	72.37	74.82	73.54	61.55
+HAO	71.22	69.63	65.82	52.97	61.08	64.51	53.76	52.94	51.20	64.04
AVG	71.73	71.41	70.90	69.98	68.53	68.41	66.21	64.42	62.43	61.42

Table 13: Detailed results of non-targeted attacks on Citeseer (2)

	Guard+LN	Guard+LNi	RobustGCN+FLAG	Guard	GCN+LNi+FLAG	Sage+LNi+FLAG+LN	GAT+LNi+FLAG+LN	RobustGCN	GCN+LNi+LN	Sage+LNi+FLAG
Clean	73.97	74.41	75.87	74.78	75.45	73.89	75.60	75.46	74.65	73.70
PGD	74.07	74.28	53.81	74.70	47.56	46.82	45.00	39.77	40.69	40.11
+HAO	48.48	38.91	51.10	33.83	49.19	46.93	44.06	39.72	40.79	40.88
MetaGIA	74.07	74.28	53.11	74.70	47.14	46.13	44.76	39.84	40.87	40.13
+HAO	45.32	38.98	50.85	33.95	49.03	46.42	44.08	39.79	41.02	40.90
AGIA	74.07	74.29	53.12	74.72	47.30	46.29	44.07	40.16	41.76	40.73
+HAO	43.47	41.04	50.88	36.51	49.61	47.28	45.66	41.53	42.32	42.82
TDGIA	74.07	74.28	55.01	74.76	49.47	47.06	41.08	37.94	40.68	36.21
+HAO	36.83	36.50	60.37	26.45	57.45	49.82	49.74	47.44	43.85	40.83
ATDIGA	74.07	74.21	54.95	74.72	45.09	41.89	36.24	34.65	32.10	31.17
+HAO	30.21	28.74	55.40	21.70	52.22	45.66	45.19	40.35	35.05	38.81
AVG	58.97	57.27	55.86	54.62	51.77	48.93	46.86	43.33	43.07	42.39

Table 14: Detailed results of non-targeted attacks on Computers (1)

	EGuard+LNi+FLAG+LN	Guard+LNi+LN	EGuard+FLAG+LN	Guard+LN	RGAT+FLAG+LN	RGAT+FLAG	EGuard+LNi+FLAG	Guard+LNi	RGAT+LN	RGAT
Clean	91.04	90.88	91.40	91.23	93.21	93.32	92.16	91.95	93.20	93.17
PGD	90.94	90.87	91.41	91.24	81.59	80.19	88.24	87.93	79.68	79.05
+HAO	87.83	87.59	80.41	75.94	81.80	82.26	64.18	62.69	79.29	79.33
MetaGIA	90.94	90.87	91.41	91.24	81.58	80.18	88.23	87.91	79.68	79.06
+HAO	90.25	90.21	90.11	88.32	81.64	81.72	78.11	76.58	79.29	78.96
AGIA	90.98	90.90	91.40	91.22	78.09	76.59	88.25	87.86	76.62	75.56
+HAO	86.02	85.77	75.97	71.49	77.55	78.17	63.96	62.74	75.23	75.14
TDGIA	90.97	90.91	91.40	91.24	77.07	75.40	90.26	89.94	75.94	74.66
+HAO	90.42	90.34	90.35	89.00	77.12	76.61	74.58	74.22	75.71	74.77
ATDIGA	90.97	90.90	91.41	91.24	82.42	81.77	89.24	88.84	81.29	80.76
+HAO	84.60	83.93	74.38	69.33	82.97	83.50	69.92	68.50	80.92	80.86
AVG	89.54	89.38	87.24	85.59	81.37	80.88	80.65	79.92	79.71	79.21

Table 15: Detailed results of non-targeted attacks on Computers (2)

	GAT+FLAG+LN	EGuard+FLAG	Guard	RobustGCN+FLAG	GAT+LNi+FLAG+LN	RobustGCN	Sage+LNi+FLAG+LN	GAT+LNi+FLAG	GCN+LNi+FLAG+LN	GAT+LNi+LN
Clean	92.17	91.68	91.55	92.46	92.40	92.24	91.71	92.45	93.22	92.05
PGD	82.31	85.82	84.91	73.27	77.91	67.14	63.83	67.61	54.96	52.20
+HAO	69.83	55.62	54.31	72.73	65.08	68.80	62.55	54.93	63.28	69.19
MetaGIA	82.31	85.81	84.91	73.28	77.91	67.14	63.83	67.62	54.96	52.21
+HAO	77.39	69.73	67.90	70.42	69.52	64.76	62.45	58.24	59.31	63.69
AGIA	79.60	86.08	85.21	71.95	75.01	66.01	60.72	64.25	52.34	50.69
+HAO	63.02	56.48	55.35	72.18	61.22	68.84	60.68	53.95	62.78	67.54
TDGIA	80.42	88.64	88.32	72.23	75.27	69.45	63.87	68.58	64.96	58.98
+HAO	79.19	69.75	68.76	71.39	70.84	69.11	63.72	63.45	66.56	65.81
ATDIGA	82.42	87.11	86.03	76.96	79.13	71.92	68.42	71.15	66.01	53.34
+HAO	60.74	61.46	58.81	76.79	64.38	74.26	68.33	57.90	72.34	73.82
AVG	77.22	76.20	75.10	74.88	73.52	70.88	66.37	65.47	64.61	63.59

Table 16: Detailed results of non-targeted attacks on Arxiv (1)

	Guard+LNi+LN	RGAT+LN	RGAT+FLAG+LN	EGuard+LNi+FLAG+LN	EGuard+FLAG+LN	Guard+LN	RobustGCN+FLAG	RobustGCN	GCN+LNi+FLAG+LN	Guard+LNi
	71.15	70.95	70.84	69.50	69.46	69.76	67.85	67.50	71.40	70.99
PGD	71.11	66.57	66.61	69.28	69.24	69.62	60.60	60.81	55.99	70.26
	68.68	66.68	66.60	61.05	61.02	58.92	62.99	62.89	60.02	47.84
MetaGIA	71.09	67.87	67.67	69.23	69.22	69.59	64.10	64.10	63.58	70.40
	69.97	66.81	66.52	66.14	66.13	65.70	63.2	63.30	64.13	58.58
AGIA	70.97	65.22	64.46	68.23	68.17	68.57	59.26	59.23	57.26	64.60
	63.57	57.02	56.60	58.27	58.20	57.73	60.77	60.72	61.50	58.08
TDIGA	71.02	67.54	67.28	68.37	68.33	68.72	63.70	63.56	61.01	65.63
	64.31	61.61	60.99	59.73	59.74	58.33	63.08	63.30	62.81	53.04
ATDGIA	71.01	68.49	68.45	68.18	68.14	68.49	64.95	64.88	63.95	66.39
	69.92	68.67	68.58	66.34	66.35	65.47	65.56	65.62	65.83	55.42
AVG	69.34	66.13	65.87273	65.85	65.82	65.54	63.28	63.26	62.50	61.93

Table 17: Detailed results of non-targeted attacks on Arxiv (2)

	RGAT+FLAG	GCN+LNi+LN	RGAT	GCN+FLAG+LN	GAT+FLAG+LN	EGuard+LNi+FLAG	GCN+LN	EGuard+FLAG	GCN+LNi+FLAG	GAT+LNi+FLAG+LN
	70.63	71.38	70.77	70.00	70.28	69.37	70.42	69.34	71.31	71.00
PGD	66.49	54.46	66.26	54.21	57.44	68.04	51.97	68.03	48.00	57.65
	57.18	58.40	55.38	55.51	59.16	37.02	52.45	36.80	52.75	53.97
MetaGIA	67.42	62.88	67.68	58.54	61.92	68.48	57.04	68.40	55.73	61.56
	58.21	63.35	57.05	59.65	51.65	50.32	57.39	50.23	57.72	54.63
AGIA	63.75	57.12	64.49	49.55	45.96	59.35	48.54	59.25	54.55	49.14
	50.31	61.29	49.36	58.25	49.71	49.24	57.24	49.20	58.10	48.78
TDIGA	66.74	58.91	66.95	55.47	56.30	62.18	52.39	62.10	48.86	52.58
	47.88	61.90	45.59	59.20	49.44	45.08	56.42	44.91	54.68	47.80
ATDGIA	67.97	62.21	68.07	58.61	63.36	62.73	55.26	62.67	54.19	58.50
	60.82	64.82	59.32	62.69	57.51	46.94	59.50	46.83	57.90	56.58
AVG	61.58	61.52	60.99	58.33	56.61	56.25	56.24	56.16	55.80	55.65

Table 18: Detailed results of targeted attacks on Computers (1)

	EGuard+LNi+FLAG+LN	Guard+LNi+LN	EGuard+FLAG+LN	Guard+LN	Guard+LNi	EGuard+LNi+FLAG	RobustGCN+FLAG	RGAT+FLAG	RGAT+FLAG+LN	EGuard+FLAG
Clean	90.96	90.76	91.56	91.11	91.12	91.29	91.85	92.83	92.78	90.75
PGD	90.96	90.76	91.56	91.11	89.38	89.54	72.36	72.17	74.28	88.36
+HAO	85.81	85.75	79.51	73.71	65.01	64.15	72.58	74.40	74.08	56.50
MetaGIA	90.96	90.76	91.56	91.11	88.93	89.10	73.81	70.58	72.24	88.10
+HAO	85.83	85.69	78.46	72.61	65.62	65.53	73.50	72.10	72.00	56.12
AGIA	91.00	90.82	91.58	91.06	89.11	89.33	72.96	68.85	69.64	88.00
+HAO	85.72	85.71	79.50	74.28	64.71	63.90	73.12	72.61	72.22	56.18
TDGIA	90.96	90.76	91.56	91.11	89.15	89.36	72.06	72.42	72.58	87.75
+HAO	77.15	75.64	65.21	62.97	69.78	70.43	73.08	74.33	74.00	64.31
ATDIGA	90.96	90.76	91.56	91.11	88.99	89.22	75.15	75.68	73.32	88.43
+HAO	78.35	77.67	62.87	59.65	63.75	63.15	74.06	75.78	74.14	56.51
AVG	87.15	86.83	83.18	80.89	78.69	78.64	74.96	74.70	74.66	74.64

Table 19: Detailed results of targeted attacks on Computers (2)

	Guard	RobustGCN	RGAT	RGAT+LN	GAT+FLAG+LN	GAT+LNi+FLAG+LN	GAT+LNi+FLAG	GCN+LNi+FLAG+LN	GAT+LNi+LN	GCN+LNi+FLAG
Clean	90.50	92.07	92.68	92.76	91.07	91.90	91.92	92.25	91.56	92.35
PGD	88.13	70.40	71.85	72.65	77.69	75.25	72.57	63.08	58.46	60.79
+HAO	54.96	70.76	71.78	71.40	71.03	66.01	62.46	66.01	70.49	64.17
MetaGIA	87.67	71.78	70.44	71.33	74.93	73.12	70.89	62.54	57.40	60.71
+HAO	55.00	71.61	70.21	70.35	69.56	64.82	62.58	64.81	67.57	63.04
AGIA	87.57	70.92	68.36	68.58	73.00	71.03	68.50	61.08	56.62	59.26
+HAO	54.89	71.58	69.96	69.99	68.44	64.81	61.00	64.68	69.39	62.28
TDGIA	87.21	69.86	71.54	71.28	74.24	72.86	70.60	62.74	57.54	60.35
+HAO	61.62	71.62	71.39	71.92	54.19	60.51	66.69	66.79	66.74	63.97
ATDIGA	87.85	73.33	74.39	72.19	73.36	75.24	74.06	65.14	56.22	62.67
+HAO	54.93	72.53	72.00	71.49	62.03	63.19	62.14	68.50	73.15	66.06
AVG	73.67	73.31	73.15	73.09	71.78	70.79	69.40	67.06	65.92	65.06

Table 20: Detailed results of targeted attacks on Arxiv (1)

	Guard+LNi+LN	EGuard+LNi+FLAG+LN	Guard+LNi	EGuard+FLAG+LN	EGuard+LNi+FLAG	Guard+LN	EGuard+FLAG	Guard	RobustGCN+FLAG	RGAT
	71.34	71.22	71.22	69.59	70.59	69.78	68.88	69.41	67.28	67.03
PGD	71.31	71.16	71.16	69.47	70.47	69.69	68.69	69.19	39.91	39.13
	69.38	65.69	33.78	47.41	29.12	38.00	14.31	13.94	36.12	36.06
MetaGIA	71.03	71.22	70.53	69.59	70.59	69.78	68.84	69.28	42.56	41.81
	42.56	48.06	33.94	31.84	34.94	26.75	20.34	18.28	38.66	38.44
AGIA	71.06	70.94	70.19	69.25	67.72	69.38	64.38	63.66	39.94	39.47
	38.56	37.22	35.06	24.63	35.31	22.09	16.19	14.09	42.53	42.56
TDIGA	71.00	71.16	69.78	68.97	68.22	69.41	66.09	66.12	41.25	41.31
	38.72	34.19	38.78	23.41	33.94	20.78	17.66	16.06	38.38	38.28
ATDGIA	71.06	70.88	70.56	69.19	69.03	69.56	66.09	66.19	44.06	43.75
	68.97	61.03	37.88	41.69	33.69	34.25	19.16	17.28	39.03	38.84
AVG	62.27	61.16	54.81	53.19	53.06	50.86	44.60	43.95	42.70	42.43

Table 21: Detailed results of targeted attacks on Arxiv (2)

	RobustGCN	RGAT+LN	RGAT+FLAG+LN	GCN+LNi+FLAG	RGAT+FLAG	GCN+LNi+LN	GCN+LNi	GCN+LNi+FLAG+LN	GAT+LN	GAT+FLAG+LN
	67.69	72.06	71.41	71.34	71.16	71.97	71.59	71.75	69.94	69.94
PGD	38.66	40.31	38.06	32.19	37.78	29.09	29.97	29.72	36.34	38.84
	37.22	37.06	34.28	32.75	23.69	28.91	29.56	29.28	28.88	30.47
MetaGIA	35.00	42.56	41.28	30.28	41.03	28.91	28.59	28.50	16.00	14.84
	33.22	34.09	32.53	30.03	27.81	27.50	27.97	27.47	19.44	21.50
AGIA	41.06	42.12	42.06	32.53	39.75	33.09	32.56	31.84	23.84	21.12
	41.97	23.84	23.66	35.19	23.03	34.03	34.47	34.25	16.97	14.94
TDIGA	44.28	43.84	43.91	36.31	42.12	36.34	35.12	36.16	27.38	24.50
	40.81	32.38	31.50	39.47	28.31	38.50	38.62	37.91	27.56	29.28
ATDGIA	43.12	44.34	44.22	34.47	41.91	33.53	33.44	33.28	31.06	24.19
	37.97	39.00	37.84	33.84	30.19	30.53	30.47	30.59	30.28	33.69
AVG	41.91	41.05	40.07	37.13	36.98	35.67	35.67	35.52	29.79	29.39

Table 22: Detailed results of targeted attacks on Aminer (1)

	EGuard+LNi+FLAG	EGuard+LNi+FLAG+LN	Guard+LNi	Guard+LNi+LN	EGuard+FLAG	Guard	RGAT+FLAG	Guard+LN	EGuard+FLAG+LN	RGAT
	59.03	58.06	60.72	60.85	57.06	57.25	61.75	58.50	58.81	62.78
PGD	55.25	48.47	56.31	49.40	53.03	53.16	41.84	49.72	48.31	40.72
	39.06	39.47	37.03	39.40	35.16	34.62	33.53	29.69	29.97	31.75
MetaGIA	52.09	50.66	52.35	49.81	49.03	48.97	46.19	48.34	47.59	45.81
	42.09	45.16	40.26	43.42	37.00	37.09	41.47	36.88	36.62	41.12
AGIA	54.06	48.00	54.82	48.17	51.28	51.34	48.72	48.78	47.59	48.25
	26.44	29.94	23.25	28.08	19.84	18.97	26.50	23.19	24.06	25.78
TDIGA	52.75	46.72	53.68	46.92	50.75	50.87	42.50	47.66	46.28	40.81
	24.31	28.91	18.54	26.07	16.12	15.06	24.00	19.69	20.66	22.5
ATDGIA	53.44	51.00	53.69	49.32	50.34	50.50	45.44	49.97	49.59	45.25
	38.19	42.66	35.93	41.07	33.72	33.72	36.72	31.91	31.69	35.94
AVG	45.16	44.46	44.23	43.86	41.21	41.05	40.79	40.39	40.11	40.06

Table 23: Detailed results of targeted attacks on Aminer (2)

	RGAT+FLAG+LN	GCN+LNi+FLAG+LN	RGAT+LN	Sage+LNi+FLAG+LN	GCN+LNi+FLAG	GCN+LNi+LN	Sage+LNi+LN	GAT+LNi+LN	GAT+LNi+FLAG+LN	Sage+LNi+FLAG
	62.66	64.41	63.78	65.56	63.91	66.88	65.44	66.97	65.78	64.34
PGD	31.97	28.03	29.75	26.22	26.81	22.65	23.78	17.00	16.66	22.03
	29.06	28.16	27.06	26.44	26.81	23.17	23.88	17.58	16.53	22.06
MetaGIA	41.38	41.12	40.78	37.56	36.72	38.17	36.56	38.40	37.31	31.25
	39.62	42.16	38.03	37.38	36.03	37.89	36.03	37.60	37.31	31.12
AGIA	38.34	34.62	37.47	31.94	33.97	31.21	31.31	29.96	29.62	29.50
	28.19	29.03	27.06	27.19	28.00	27.14	26.31	22.00	21.09	25.25
TDIGA	30.47	28.44	28.25	24.41	24.97	20.85	22.19	15.39	15.16	20.56
	27.12	27.53	24.97	24.56	24.84	22.19	22.22	15.75	14.03	20.16
ATDGIA	39.28	36.62	38.03	33.38	32.09	32.44	32.47	34.83	35.12	27.62
	32.66	37.72	31.50	31.87	31.78	32.00	30.72	33.97	33.06	26.12
AVG	36.43	36.17	35.15	33.32	33.27	32.24	31.90	29.95	29.24	29.09

Table 24: Detailed results of targeted attacks on Reddit (1)

	Guard+LNi+LN	RobustGCN	RobustGCN+FLAG	Guard+LNi	Guard+LN	EGuard+LNi+FLAG+LN	EGuard+FLAG+LN	Sage+LNi+FLAG+LN	Guard	EGuard+FLAG
	94.47	95.08	95.30	94.42	94.61	94.61	94.60	97.10	94.05	94.08
PGD	92.91	84.81	83.84	93.03	92.69	92.69	92.53	76.25	92.44	92.72
	80.03	86.12	84.94	75.53	68.53	69.31	69.34	75.25	56.44	58.03
MetaGIA	93.53	88.25	87.22	93.28	93.38	93.66	93.59	80.72	92.40	92.88
	77.47	90.06	90.44	69.91	65.28	68.00	68.34	83.62	46.75	48.59
AGIA	93.62	86.09	87.84	92.84	93.16	92.78	92.69	81.59	92.19	91.31
	88.66	85.06	87.84	85.34	83.09	77.06	77.31	72.19	78.16	67.06
TDIGA	93.03	90.19	89.91	92.25	92.59	92.91	92.53	80.94	91.25	91.59
	86.03	89.06	88.91	80.38	78.69	81.56	81.25	79.78	64.09	66.62
ATDGIA	93.34	87.34	84.91	92.38	92.69	93.97	93.81	76.53	91.62	93.00
	90.78	88.84	88.38	87.94	88.06	79.44	79.25	78.66	80.69	63.22
AVG	89.44	88.26	88.14	87.03	85.71	85.09	85.02	80.24	80.01	78.10

Table 25: Detailed results of targeted attacks on Reddit (2)

	EGuard+LNi+FLAG	Sage+LNi+LN	GAT+LNi+FLAG+LN	Sage+FLAG+LN	Sage+LN	GAT+LNi+LN	GAT+FLAG+LN	GAT+LN	Sage+LNi+FLAG	GCN+LNi+FLAG+LN
	94.07	97.10	95.19	97.13	97.11	95.37	94.49	94.77	97.09	95.84
PGD	92.69	74.94	75.91	67.47	63.75	70.38	73.53	78.12	57.16	71.28
	58.12	73.91	79.59	67.72	64.72	72.91	78.16	76.47	56.62	70.91
MetaGIA	92.84	78.63	68.16	84.03	82.53	67.28	59.91	62.94	65.69	62.13
	48.69	80.56	78.84	80.06	76.59	75.22	74.34	66.12	69.59	59.66
AGIA	91.31	69.50	59.53	62.66	57.19	51.75	43.22	50.28	67.88	59.28
	66.97	58.47	74.19	51.16	49.19	51.12	72.91	46.19	55.75	52.53
TDIGA	91.59	78.09	73.12	74.00	68.62	70.34	64.81	73.00	65.28	58.84
	66.41	77.22	73.91	75.72	71.75	69.72	64.75	67.97	65.44	58.09
ATDGIA	93.03	73.31	64.25	68.44	63.59	64.53	53.66	57.91	65.12	62.34
	63.34	73.78	72.53	69.62	65.00	65.22	62.97	65.50	63.66	57.97
AVG	78.10	75.96	74.11	72.55	69.09	68.53	67.52	67.21	66.30	64.44