Navigating Graph Robust Learning against All-Intensity Attacks

Xiangchi Yuan [1] Chunhui Zhang [1] Yijun Tian [2] Chuxu Zhang [1]

Abstract

Graph Neural Networks have demonstrated exceptional performance in a variety of graph learning tasks, but their vulnerability to adversarial attacks remains a major concern. Accordingly, many defense methods have been developed to learn robust graph representations and mitigate the impact of adversarial attacks. However, most of the existing methods suffer from two major drawbacks: (i) their robustness degrades under higher-intensity attacks, and (ii) they cannot scale to large graphs. In light of this, we develop a novel graph defense method to address these limitations. Our method first applies a denoising module to recover a cleaner graph by removing edges associated with attacked nodes, then, it utilizes Mixture-of-Experts to select differentially private noises of different magnitudes to counteract the node features attacked at different intensities. In addition, the overall design of our method avoids relying on heavy adjacency matrix computations like SVD, thus enabling the framework's applicability on large graphs.

1 Introduction

Graph Neural Networks (GNNs) are powerful tools for learning graph-structured data and have shown state-of-the-art performance in various graph learning tasks such as node classification (Kipf & Welling, 2017; Veličković et al., 2018) and graph classification (Ying et al., 2018; You et al., 2020). Despite this, GNNs remain vulnerable to adversarial attacks (Zügner & Günnemann, 2019; Zheng et al., 2021). Considering this vulnerability, various defend methods have been proposed to increase the robustness of GNNs against these attacks (Jin et al., 2020; Zhang & Zitnik, 2020).

However, many current defend methods for GNNs on graph datasets still suffer from *(i)* robustness degradation for defending graph attacks with increasing intensity and *(ii)* limited scalability for working on large graph datasets. Specifically, to illustrate the first drawback *(i)*, for example, the Graph Robustness Benchmark (GRB) (Zheng et al., 2021) has shown that the majority of node injection attacks (Goodfellow et al., 2015; Zheng et al., 2021; Madry et al., 2018; Zou et al., 2021) are mild in evaluation, but many defense methods (such as RGCN (Zhu et al., 2019) and GNN-SVD (Entezari et al., 2020)) are only effective under these mild attack settings, while their robustness tends to degrade as the attack intensity increases, making them less practical in real-world severe scenarios. To explain the second drawback *(ii)*, current popular methods such as GNN-SVD (Entezari et al., 2020) and GNNGuard (Zhang & Zitnik, 2020) must compute a dense adjacency matrix, which leads to prohibitive computational cost when applied to large graphs.

To combat the above challenges, we propose a novel framework called **DRAGON** (i.e., **D**ifferentially P**r**ivate M**a**sked **G**raph Aut**o**-E**n**coder for Anti-degraded Robustness). To address (i) the robustness degradation under increasing-intensity graph attacks, DRAGON utilizes a denoise masked auto-encoder to reconstruct the given attacked graph towards cleaner node connections and further uses a differential privacy-based Mixture-of-Experts approach to eliminate the impact of injected nodes hidden in an attacked graph. For dealing with (ii) the limited scalability issue on large graph datasets, DRAGON avoids the heavy computation of large adjacency matrices, which is achieved by making all designs not require large-scale adjacency matrices like GNNSVD (Entezari et al., 2020) and GNNGuard (Zhang & Zitnik, 2020), thereby we avoid out-of-memory problems while scaling our framework to large graph datasets.

Specifically, DRAGON employs a two-step approach as follows: *First*, given an attack graph, a masked graph autoencoder is applied to eliminate malicious edges connected to injected nodes, resulting in a cleaner graph with very few injected nodes connected to clean nodes; *Second*, we ensemble the differential privacy (DP) mechanism in GNN layer, which introduces random noises into the graph features while counteracting the attacked node features and constraining the change of the model's output, thus against attacks in injected nodes. Additionally, the Mixture-of-Experts (MoE) technique is used to manipulate multiple DP

[*]Equal contribution [1]Brandeis University [2]University of Notre Dame. Correspondence to: Chuxu Zhang <chuxuzhang@brandeis.edu>.

2^{nd} *AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

expert networks, each holding Gaussian DP noise of different magnitudes. These expert networks are then assigned to injected node features attacked at different intensities, providing an appropriate level of noise to improve robustness. To the best of our knowledge, this work is a first attempt to address the issues of robustness degradation and limited scalability for adversarial learning on graphs.

2 Preliminaries

Sparse Mixture-of-Experts. The basic idea behind MoE is to divide the input space into numerous partitions and assign different expert models to different partitions. The final output is obtained by assembling the predictions of all the experts, typically using a gating mechanism that determines the weight of each expert based on the input. Formally, let h be the input space to the MoE, and $\mathbf{E} = \{E_i(\cdot)\}_{i=1}^{N}$ denotes that an MoE layer consists of n experts. The output of the MoE is given by: $y = \sum_{i=1}^{N} p_i(h)E_i(h)$, where E_i is the i-th expert model, $p_i(h)$ is the weight assigned to the i-th expert for the input space h, and N is the number of experts. The weights are obtained from a gating network, which is trained to assign higher weights to experts that are more likely to make accurate predictions for a given input space.

Node Injection Attacks. We consider an undirected attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ is the set of N nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{X} \in \mathbb{R}^{N\times D}$ denotes the set of node features with D dimensions. A node classification model $f : \mathcal{G} \to \mathcal{Z}$ maps a graph $\mathcal{G}$ to a probability vector $\mathcal{Z} \in \mathbb{R}^{N\times K}$ with K classes. The objective of node injection attacks on GNN is to maximize the number of incorrect predictions by injecting new malicious nodes into the graph, but cannot modify the existing edges or node features. Formally, the objective of the attack can be formulated as:

$$\max_{\mathcal{G}'} |\arg\max k \in [1, ..., K] f(\mathcal{G}') \neq \arg\max k \in [1, ..., K] f(\mathcal{G}), \tag{1}$$

where $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{X}')$ is the attacked graph, and $\mathcal{V}'$ is the set of $N + M$ nodes, where M is the number of injected malicious nodes. The constraints on the modified graph are represented by the edge distance metric $d_{\mathcal{E}}(\mathcal{E}', \mathcal{E}) \leq \Delta_{\mathcal{E}}$ and node feature distance metric $d_{\mathcal{X}}(\mathcal{X}', \mathcal{X}) \leq \Delta_{\mathcal{X}}$, where $\Delta_{\mathcal{E}}$ represents the limited number of edges associated with the injected nodes, and $\Delta_{\mathcal{X}}$ is the maximum range of values that can be changed in the injected node features.

Differential Privacy. The idea of Differential Privacy (DP) is to add randomness to the computation in order to prevent attackers from leaking any individual information, thereby DP guarantees the output of a function $f(\cdot)$ that needs to be protected over two neighbouring datasets are indistinguishable. Specifically, a randomized mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private if, for all neighboring datasets $D \sim D'$ that differ in one record or are bounded by a certain distance and for all events $\mathcal{O} \subseteq Range(\mathcal{M})$ in the output space O of $\mathcal{M}$, we have the follows:

$$\Pr[\mathcal{M}(D) \in \mathcal{O}] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in \mathcal{O}] + \delta, \tag{2}$$

where ϵ and δ are parameters that control the privacy strength and privacy budget in DP, respectively. Standard (ϵ, δ)-DP is guaranteed by adding statistical noise to the output of the protected function, so that changing a single input in the dataset does not significantly change the distribution of the output from the protected function.

3 Methodology

In this section, we present a novel framework called DRAGON to increase the GNN robustness. Figure 1 illustrates the overall framework which utilizes two key components. First, DRAGON utilizes the DMGAN as an attacked graph preprocessing module, which deletes malicious edges associated with injected nodes to recover a cleaner graph from an attacked graph for the defender GNN. Second, the framework incorporates the DP mechanism and graph convolution together as a DP-graph convolution (DP-GConv) layer to improve the defense module. Since the intensities of attacks are unpredictable, we split the DP-GConv into multiple experts associated with different magnitudes of DP noise in DP-GConv to handle attacks of different intensities.

3.1 Denoise Masked Graph Auto-Encoder

Denoise Masked Graph Auto-Encoder (DMGAN) eliminates the negative influence of injected nodes by removing the malicious edges associated with the injected nodes. And its pipeline includes three steps: masking the input (attacked) graph, encoding and decoding, and self-supervising.

Mask the Input Graph. Given the input graph $\mathcal{G}$, we adopt the random walk (Perozzi et al., 2014) as a path-masking strategy to mask the graph. We denote the masked subgraph as $\mathcal{G}_{\mathrm{m}} = (\mathcal{V}_{\mathrm{m}}, \mathcal{E}_{\mathrm{m}}, \mathcal{X}_{\mathrm{m}})$, and the visible subgraph as $\mathcal{G}_{\mathrm{v}} = (\mathcal{V}_{\mathrm{v}}, \mathcal{E}_{\mathrm{v}}, \mathcal{X}_{\mathrm{v}})$, where they complement each other, i.e., $\mathcal{G}_{\mathrm{m}} \cup \mathcal{G}_{\mathrm{v}} = \mathcal{G}$. The masking process is formulated as follows: $\mathcal{E}_{\mathrm{m}} \sim \mathrm{RandomWalk}(\mathcal{V}_{\mathrm{r}}, n_{\mathrm{w}}, l_{\mathrm{w}})$, where $\mathcal{V}_{\mathrm{r}} \subseteq \mathcal{V}$ is the set of root nodes. n_{w} and l_{w} represent the number of walks per node and the walk lengths, respectively.

Generate the Clean Graph via Auto-Encoder. We first obtain the encoded representations H obtained from the encoder for nodes $\mathcal{V}$. Then the decoder is trained with two goals: reconstructing the graph structures and predicting the node degrees. The decoder leverages the representations of a pair of nodes as link representations to reconstruct the connections of the original graph. Define the structure decoder as Dec_{ω} with weight parameters ω as follows: $\mathrm{Dec}_{\omega}(h_i, h_j) = \mathrm{Sigmoid}(\mathrm{MLP}(h_i \circ h_j))$, where $\circ$ denotes

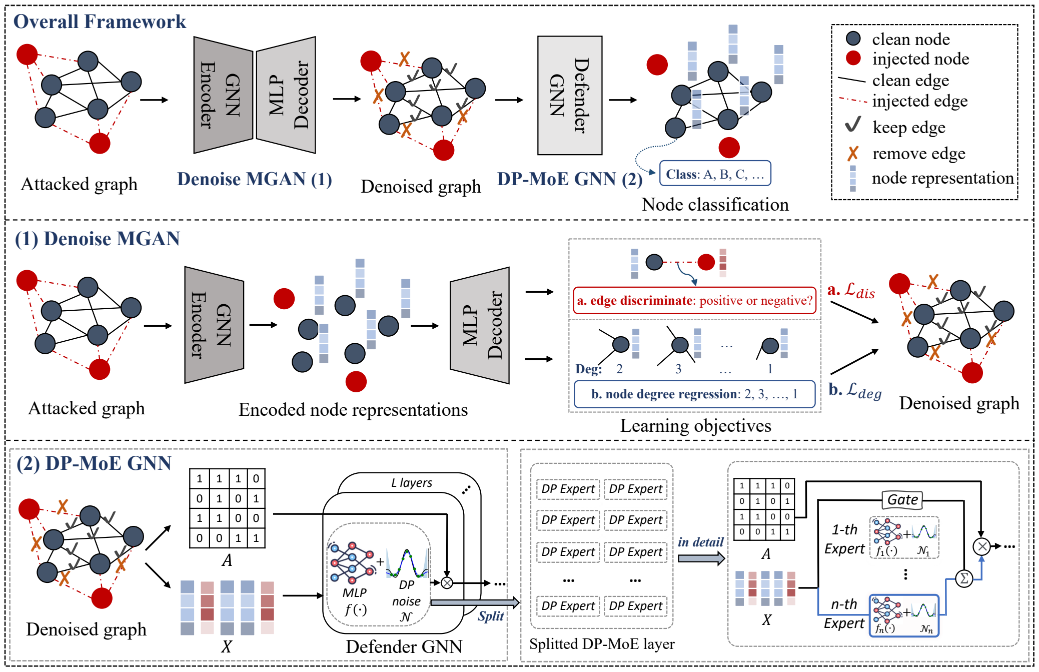

Figure 1: Overall framework. (1) In Denoise MGAN, a cleaner graph is recovered by removing the malicious edges. Then, the cleaner graph is classified using (2) DPMoE GNN, which consists of a DP graph convolutional layer split into multiple DP expert networks.

the element-wise product, $h_i \in H$ and $h_j \in H$ are the representations of node i and node j, respectively. To decode the node degree, we define the degree decoder Dec_ϕ with weight parameters ϕ as follows: $\mathrm{Dec}_\phi(h_v) = \mathrm{MLP}(h_v)$, where h_v is the representation of the target node $v \in \mathcal{V}_\mathrm{m}$.

Self-Supervised Loss for Denoise Auto-Encoder. Since the decoder is supposed to learn two goals, the loss function consists of two terms. First, a binary cross-entropy loss term is used to predict the edge for graph structure decoding. Formally, it is defined as:

$$\mathcal{L}_\mathrm{e} = -\,(1/|\mathcal{E}_\mathrm{m}|) \sum_{(u,v)\in\mathcal{E}_\mathrm{m}} \log(\mathrm{Dec}_\omega(h_u, h_v)) \\ -\,(1/|\mathcal{E}_\mathrm{n}|) \sum_{(u',v')\in\mathcal{E}_\mathrm{n}} \log(1-\mathrm{Dec}_\omega(h_{u'}, h_{v'})), \tag{3}$$

where h is the node representation output from the encoder, $\mathcal{E}_\mathrm{n}$ is a set of unexist negative edges sampled from training graph $\mathcal{G}$. Second, a regression loss term is applied to measure how accurately the node degree prediction matches the original in the masked graph, which is defined as:

$$\mathcal{L}_\mathrm{d} = (1/|\mathcal{V}_\mathrm{m})| \sum_{v\in\mathcal{V}_\mathrm{m}} ||\mathrm{Dec}_\phi(h_v) - \deg(v)||_2^2, \tag{4}$$

where $\deg(\cdot)$ and $\mathcal{V}_\mathrm{m}$ denote the masked node degree and the masked nodes in the graph, respectively. The overall objective $\mathcal{L}$ to be minimized during training is the combination of $\mathcal{L}_\mathrm{d}$ and $\mathcal{L}_\mathrm{e}$.

After using the loss $\mathcal{L}$ to train the auto-encoder DMGAN, during attack evaluation, an attacked graph $\mathcal{G}^{'} = (\mathcal{V}^{'}, \mathcal{E}^{'}, \mathcal{X}^{'})$ is perturbed from $\mathcal{G}$ and used as an input. Then we use the trained DMGAN to decode the reconstructed edges of $\mathcal{G}^{'}$, i.e., $\mathcal{E}_r$. We use $\mathcal{E}_r$ to index the reconstructed subgraph $\mathcal{G}_r$ from $\mathcal{G}^{'}$, and then use $\mathcal{G}_r$ for downstream tasks.

3.2 DP-based Mixture of Experts

After cleaning the attacked graph with the denoising preprocessing module (i.e., DMGAN), we use the DP mechanism in the defender GNN to complete a robust prediction on the graph $\mathcal{G}_r$. To achieve this, we first introduce the DP mechanism into the GNN layer to get DP-GConv, and then split DP-GConv into DPMoE consisting of different DP experts to defend against attacks of varying intensities.

Differentially-Private Graph Convolution. The Differentially Private Graph Convolution (DP-GConv) is designed to help Defender GNN learn more robust representations. DP-GConv updates the representation of a target node, $h_v^{(l-1)}$, which is output by the $(l-1)$-th layer, through three steps: First, a Gaussian differentially-private noise module $\mathcal{M}(\cdot)$ is used to compute the Gaussian noise $\mathcal{N}$, which can be later added to $h_v^{(l-1)}$. The procedure is formulated as $\mathcal{M}(h_v^{(l-1)}) = h_v^{(l-1)} + \mu\mathcal{N}$, where the Gaussian DP noise module $\mathcal{M}(\cdot)$ constrains the change in the final output of the GNN model when given perturbed adversarial input such as an attacked graph with injected nodes, μ represents the scaling coefficient of the Gaussian

(ϵ, δ)-DP noise $\mathcal{N}$ and $\mathcal{N}$ uses Gaussian distribution with mean zero and standard deviation $\sigma = \sqrt{2\ln(1.25/\delta)}/\epsilon$. Second, $\mathcal{M}(h_v^{(l-1)})$ is multiplied with a learnable weight matrix, $W^{(l)}$, which is executed by the $\mathrm{DPLinear}(\cdot)^l$ module: $\mathrm{DPLinear}^{(l)}(h_v^{(l-1)}) = \mathrm{W}^{(l)}\mathcal{M}(h_v^{(l-1)})$, where $\mathrm{DPLinear}(\cdot)^l$ is also used to transform all neighboring nodes linked with the current target node. Third, DP-GConv combines the feature of the target node and neighboring nodes to update the node representation as follows:

$$\begin{aligned} h_v^{(l)} =& \mathrm{COM}^{(l)}\Big(\mathrm{DPLinear}^{(l)}(h_v^{(l-1)}), \\ & \mathrm{AGG}\left(\left\{\mathrm{DPLinear}^{(l)}(h_u^{(l-1)}), \forall u \in N_v\right\}\right)\Big), \end{aligned} \tag{5}$$

where $\mathrm{AGG}(\cdot)$ and $\mathrm{COM}(\cdot)$ represent the neighbor aggregation and combination functions, respectively; N_v denotes the set of node v's all neighboring nodes u.

Splitted Differentially-Private Mixture-of-Experts. For the anti-degraded robustness of the defender GNN while against node injection attacks in different intensities, we propose a novel design called Splitted Differentially-Private Mixture-of-Experts (DPMoE). This design divides each DP-GConv layer's $\mathrm{DPLinear}(\cdot)$ into multiple expert networks, with each expert network equipped with a specific magnitude of Gaussian DP module. This enables the defender GNN to effectively handle node injection attacks of different intensities. The DPMoE module, which is splitted from $\mathrm{DPLinear}(\cdot)$, is formulated as follows:

$$\mathrm{DPMoE}(h) = \sum_{i \in \mathcal{T}} p_i(h) \cdot \mathrm{DPLinear}_i(h), \tag{6}$$

where $\mathcal{T}$ represents the set of activated top-k expert indices. $\mathrm{DPMoE}(\cdot)$ combines the output of multiple DP expert networks, where each expert $\mathrm{DPLinear}_i(\cdot)$ has a different scaling coefficient μ_i for the Gaussian DP noise $\mathcal{N}$ in different magnitudes as mentioned in Equation (2) (μ_i is linearly increased as the i of that expert increases). Specifically, the top-k activated expert indices are determined by the gate-value $p_i(h)$, which can be obtained using a softmax function as described as:

$$p_i(h) = \frac{\exp(t(h)_i + \varepsilon_i)}{\sum_{k=1}^{N} \exp(t(h_i)_k + \varepsilon_k)}, \tag{7}$$

where $t(\cdot)$ is a linear transformation, and N is the number of all experts. The activated expert indices in the DPMoE module are determined by a gate-value $p_i(h)$. $p_i(h)$ takes the logits of all experts, obtained through a linear transformation of the input feature vector h, and computes the activation probability of each expert. The logits are weighted by the i-th value $t(h)_i$ of the linear transformation, and a random noise term ε_i is added to ensure randomness in the expert activating procedure. ε_i is typically chosen to be a sample from a Gaussian distribution.

Therefore, the final updated feature representation of the target node is obtained as follows:

$$\begin{aligned} h_v^{(l)} =& \mathrm{COM}^{(l)}\Big(\mathrm{DPMoE}^{(l)}(h_v^{(l-1)}), \\ & \mathrm{AGG}\left(\left\{\mathrm{DPMoE}^{(l)}(h_u^{(l-1)}), \forall u \in N_v\right\}\right)\Big). \end{aligned} \tag{8}$$

Compared to the previous DP-GConv as formulated in Equation (5), the proposed DPMoE layer uses a gating mechanism to manipulate multiple experts with multi-scale DP Gaussian noises, allowing it to handle different intensities of node injection attacks with greater anti-degraded robustness.

Theoretical Analysis of DPMoE Robustness. If DPMoE satisfies the following Equation (9) in Lemma 3.1, we can theoretically ensure the robustness of the model. The complete analysis and proof are listed in Appendix C.

Lemma 3.1. Robustness for DPMoE. *Suppose a GNN $f(\cdot)$ containing DPMoE satisfies (ϵ, δ)-DP. The model with label probability output vector $p(h_v^{(l)}) = (\mathbb{E}(f_1(h_v^{(l)})), \ldots, \mathbb{E}(f_K(h_v^{(l)})))$ for node v is robust to features $h_v^{(l)}$ after node injection attack on layer l, if some $k \in \mathcal{K}$ and the expected value $\mathbb{E}$ of output satisfies the following property:*

$$\mathbb{E}(f_k(h_v^{(l)})) > e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(f_i(h_v^{(l)})) + (1 + e^{\epsilon})\delta. \tag{9}$$

4 Experiments

The experiments aim to address three key research questions: **RQ-1**: Can DRAGON outperform other state-of-the-art (SOTA) methods in terms of robustness and scalability? **RQ-2**: What is the contribution of each component in the denoise-then-defend framework to robustness? **RQ-3**: How does DRAGON navigate graph data attacked at different intensities via anti-degraded robust design?

4.1 Experiment Settings

Datasets. We evaluate the robustness and scalability of our proposed DRAGON model using the Graph Robustness Benchmark (GRB) dataset (Zheng et al., 2021), which includes graphs of varying scales, such as *grb-cora* (small-scale), *grb-citeseer* (small-scale), *grb-flickr* (medium-scale), *grb-reddit* (large-scale), and *grb-aminer* (large-scale). The statistics of the datasets are listed in Appendix D.

Baselines Methods. We evaluate the robustness of DRAGON against several baseline methods from two perspectives: First, for baselines that have been applied on robustness against injection attacks, we consider GCN-SVD (Entezari et al., 2020), GNNGuard (Zhang & Zitnik, 2020), RGCN (Zhu et al., 2019), EvenNet (Lei et al., 2022). Second, we compare against general GNN models

Table 1: Overall assessments among five GRB datasets with different-intensitiy FGSM attacks. The best result is bolded and the runner-up is underlined. Inten. denotes Attack Intensity and OOM represents Out-of-Memory. *Method*+AT means applying adversarial training.

	Inten.	GCN	GAT	RGCN	GCNSVD	GATGuard	EvenNet	GCN+AT	GAT+AT	RGCN+AT	GAME+AT	Ours	Ours+AT
	nulla	84.2±0.6	83.6±0.5	84.0±0.7	64.5±1.0	81.7±1.0	82.4±0.7	83.5±0.1	79.6±0.9	85.4±0.5	**85.5±0.8**	84.1±0.7	81.6±0.7
	I	65.6±3.2	58.9±4.2	51.8±1.5	44.1±0.4	80.6±1.1	67.7±2.2	82.5±1.1	79.4±0.8	79.1±2.1	81.7±0.4	**84.2±0.6**	82.0±0.7
Cora	II	57.3±3.0	38.9±3.7	23.0±0.7	42.0±2.0	80.7±1.0	53.3±2.1	79.6±0.8	79.2±0.7	72.8±4.6	80.1±0.7	**84.1±0.3**	81.6±1.4
(Small)	III	33.6±3.0	24.3±4.3	13.6±0.3	37.5±2.9	80.7±1.0	39.6±1.7	71.4±4.7	79.5±0.8	49.7±7.2	79.1±0.8	**84.0±0.5**	81.7±0.5
	IV	17.9±0.8	15.8±2.5	12.0±0.1	33.7±1.4	79.4±1.2	28.9±0.8	47.9±3.9	78.3±0.1	25.0±2.6	78.3±0.2	**84.0±0.3**	81.0±0.9
	V	12.4±0.1	14.3±1.6	11.9±0.1	33.0±0.4	79.4±1.5	23.8±0.2	35.6±4.4	79.3±0.6	24.7±8.9	77.3±0.6	**83.3±0.3**	80.4±0.8
	nulla	71.6±0.8	72.0±0.6	73.7±0.7	68.2±1.2	72.8±0.3	69.0±0.5	73.8±0.6	69.6±1.5	73.9±0.3	**76.1±1.3**	75.9±0.6	71.1±0.3
	I	37.9±6.5	19.4±0.6	62.4±2.0	23.4±2.1	72.5±0.6	57.3±2.3	59.7±2.9	69.2±0.7	71.4±1.4	72.4±0.8	**75.6±0.2**	70.6±0.6
Citeseer	II	33.3±8.4	21.1±2.5	45.9±5.5	22.6±0.6	72.4±0.6	49.0±3.0	28.0±1.6	68.8±0.7	64.4±1.4	69.3±0.9	**74.7±0.3**	70.4±0.7
(Small)	III	17.8±1.3	19.7±3.8	35.1±3.9	21.7±2.0	72.5±0.6	36.8±3.1	27.1±3.5	68.7±0.3	58.7±3.6	66.7±1.2	**75.9±0.5**	70.7±1.5
	IV	16.2±1.0	19.6±5.6	31.4±5.1	19.4±1.8	72.4±0.6	28.6±1.9	23.6±6.4	68.7±0.3	51.9±2.8	64.6±0.3	**75.6±0.5**	70.3±1.0
	V	21.0±4.0	13.5±4.7	32.7±2.4	14.6±1.0	72.5±0.6	24.0±3.3	26.7±9.0	68.8±0.8	47.3±4.3	62.8±0.7	**75.3±0.3**	70.6±1.1
	nulla	47.1±0.5	50.0±1.2	50.8±0.7	OOM	OOM	49.0±0.6	45.4±0.3	44.2±1.8	43.1±5.6	52.2±0.9	**52.7±0.1**	51.1±0.0
	I	39.6±0.9	47.8±2.6	48.2±1.0	OOM	OOM	48.2±0.7	46.3±0.8	44.3±2.1	40.3±4.6	45.6±1.1	**52.0±0.2**	51.0±0.1
Flickr	II	30.6±0.1	44.1±3.9	43.7±2.1	OOM	OOM	45.1±1.4	44.3±1.2	44.0±2.4	40.0±2.8	43.0±1.4	**51.8±2.1**	50.3±0.0
(Medium)	III	13.5±1.0	34.2±7.3	18.9±2.5	OOM	OOM	37.2±2.3	27.1±7.6	43.1±3.1	45.9±3.6	41.1±1.1	**51.2±1.5**	49.8±0.0
	IV	9.6±0.3	24.5±9.6	12.6±1.3	OOM	OOM	29.5±3.4	15.2±2.7	42.8±3.5	43.5±1.2	40.8±0.9	**50.3±1.8**	48.7±0.3
	V	9.1±0.1	24.9±9.8	15.7±4.5	OOM	OOM	26.9±3.8	14.4±5.6	42.4±4.1	43.8±1.5	38.4±2.1	**48.8±1.7**	47.5±0.1
	nulla	95.6±0.0	95.8±0.0	95.5±0.1	OOM	OOM	95.1±0.0	95.6±0.0	95.4±0.0	95.7±0.0	96.1±0.0	**96.2±0.0**	95.7±0.0
	I	95.6 ±0.1	95.4±0.1	93.0±0.1	OOM	OOM	95.0±0.1	95.4±0.1	95.4±0.2	93.2±0.0	95.3±0.0	**96.1±0.0**	95.6±0.1
Reddit	II	94.7±0.0	95.2±0.2	85.8±1.4	OOM	OOM	94.9±0.2	95.1±0.0	95.4±0.1	85.0±0.3	95.1±0.1	**96.1±0.0**	95.6±0.0
(Large)	III	93.6±0.1	94.2±0.1	75.4±1.4	OOM	OOM	93.8±0.2	94.4±0.2	95.3±0.0	72.0±0.1	94.8±0.0	**96.0±0.0**	95.5±0.0
	IV	91.7±0.3	93.9±1.0	59.1±1.0	OOM	OOM	93.7±0.1	93.2±0.3	95.2±0.1	51.5±0.1	94.2±0.2	**95.9±0.1**	95.5±0.0
	V	88.6±0.1	93.1±0.5	49.6±1.3	OOM	OOM	93.2±0.1	88.9±0.2	95.2±0.0	42.8±0.0	93.0±0.1	**95.8±0.1**	95.5±0.1
	nulla	63.4±0.1	**66.8±0.7**	63.6±0.2	OOM	OOM	62.7±0.0	64.1±0.0	63.9±0.0	64.4±0.2	64.5±0.2	64.3±0.8	64.3±0.0
	I	61.0±0.5	59.3±0.3	51.2±0.6	OOM	OOM	61.0±0.2	62.7±0.0	63.8±0.2	61.0±1.0	63.0±0.1	61.7±0.5	**64.1±0.0**
AMiner	II	51.2±0.3	46.9±1.0	32.6±0.3	OOM	OOM	52.3±0.1	57.4±0.0	63.4±0.1	50.6±2.2	62.1±0.0	56.1±0.2	**63.7±0.1**
(Large)	III	39.3±0.1	35.6±0.8	21.9±0.3	OOM	OOM	42.7±0.0	48.6±0.0	62.7±0.0	38.3±1.2	60.8±0.1	49.0±0.4	**63.1±0.1**
	IV	32.2±1.2	30.2±2.5	16.51±1.2	OOM	OOM	32.2±0.1	41.3±0.0	62.1±0.1	31.0±0.7	58.6±0.2	42.5±0.3	**62.6±0.3**
	V	24.5±0.7	23.8±2.3	13.0±0.9	OOM	OOM	26.3±0.2	32.4±0.0	61.1±0.3	22.0±2.3	56.7±0.3	49.7±0.2	**62.0±0.5**

(i.e., GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018)), RGCN (Zhu et al., 2019) and GAME (Zhang et al., 2023) that are integrated with a generic defense approach, Adversarial Training (AT) (Madry et al., 2018). Training and evaluation configuration is included in Appendix D.

Attack Strategies. In this section, we examine five effective and diverse graph injection attack methods (FGSM (Goodfellow et al., 2015), SPEIT (Qinkai et al., 2020), PGD (Madry et al., 2018), TDGIA (Zou et al., 2021), HAO (Chen et al., 2022)) that can impair the performance of victim GNNs, including our proposed DRAGON framework and other baselines. Notice that we mainly focus on injection attacks and a discussion of GNNs under modification attack is included in Appendix E.

In configuring these attack strategies, we maintain consistency with the default configuration in the GRB benchmark by setting the number of edges per injected node as per the GRB benchmark. Then, to evaluate the robustness under varying attack intensities, we first establish a base attack intensity for each graph dataset, and then **multiply** the number of injected nodes to create five different attack intensities from I to V as shown in Appendix D. For hyperparameters to generate attacks and surrogate model, we keep it same with GRB benchmark and details are listed in Appendix D.

4.2 Overall Robustness and Scalability on GRB

To answer the first key question **RQ-1**, we evaluate the robustness and scalability of DRAGON and compare it with other state-of-the-art (SOTA) baseline methods. The results are reported in Table 1 and detailed results including SPEIT, PGD, TDGIA, and HAO attack evaluation are included in Appendix B. The results in Table 1 show that DRAGON outperforms the other baselines in terms of robust accuracy under attacks of five intensities without encountering out-of-memory problems on the same hardware platform. For instance of anti-degraded robustness, on the small-scale grb-citeseer dataset, DRAGON outperforms the most competitive baseline GAME+AT by 3.2% when the attack intensity is I and by 12.5% when the attack intensity increases to V; On the medium grb-flickr dataset, compared to the most competitive baseline GAME+AT, DRAGON outperforms this baseline by 6.4% when the attack intensity is I and by 10.4% when the attack intensity increases to V; On the large grb-reddit dataset, DRAGON outperforms the most competitive baseline, GAT+AT, by 0.7% when the attack intensity is I and by 0.6% when the attack intensity increases to V; In addition, our DRAGON also demonstrates strong representation ability on clean graphs (i.e., nulla) setting.

Table 2: Ablation studies for DRAGON on graphs of varying scales and under FGSM attack of varying intensities.

	Int.	Base model	w.o.denoise	w.o.DPMoE	DRAGON
	nulla	**66.8±0.1**	66.3±0.1	64.8±0.0	64.3±0.8
	I	59.3±0.5	61.7±0.4	60.9±0.8	**61.7±0.5**
AMiner	II	46.9±0.3	52.4±0.7	51.9±0.2	**56.1±0.2**
(large)	III	35.6±0.1	42.4±1.7	40.9±0.4	**49.0±0.4**
	IV	30.1±1.2	34.0±0.5	34.6±0.3	**42.5±0.3**
	V	23.8±0.7	26.5±2.2	28.1±0.9	**39.7±0.2**

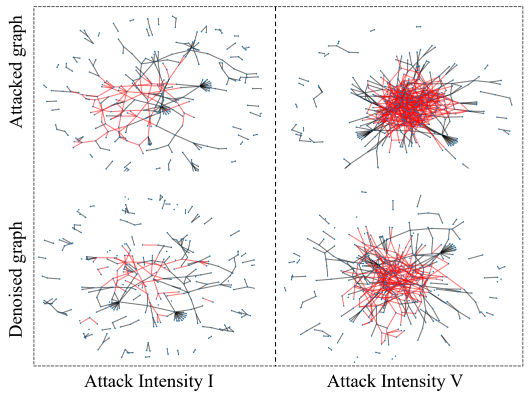

Figure 2: The visualization of sampled *Cora* graph at attack intensity I and V and their DMGAN-denoised version. The black edges denote normal edges and the red edges represent malicious edges.

4.3 Ablation Study

To answer **RQ-2**, we remove each of the components in DRAGON and perform experiments to observe performance on the AMiner dataset. We denote the model ablated by DRAGON as (a) without denoise (DMGAN), (b) without DPMoE, and (c) the GAT backbone, i.e., the base model without denoise and DPMoE, as shown in Table 10. For (a) w.o. denoise, the model experienced a 3.7% loss in accuracy when the attack intensity is II, and a 6.8% loss in accuracy when the attack intensity increases to V. For (b) w.o. DPMoE, the removal of DPMoE led to a 0.8% loss in accuracy when the attack intensity is I, and a 11.6% loss in accuracy when the attack intensity increases to V. Finally, for (c) w.o. denoise and DPMoE, removing both components resulted in a 2.5% increase in accuracy when the attack intensity is I, and a 15.9% loss in accuracy when the attack intensity increases to V. To summarize, each DRAGON component shows improvement in overall robustness.

4.4 What Wins the Anti-degraded Robustness?

To address the key research question **RQ-3**, we investigate the denoising module (DGMGAN) in recovering attacked graphs and the capability of the defender component (DPMoE) in manipulating the magnitude of the Gaussian Differential Privacy mechanisms and analyze how they improve DRAGON's anti-degradation robustness.

Denoising Injected Nodes by DMGAN. We visualize a

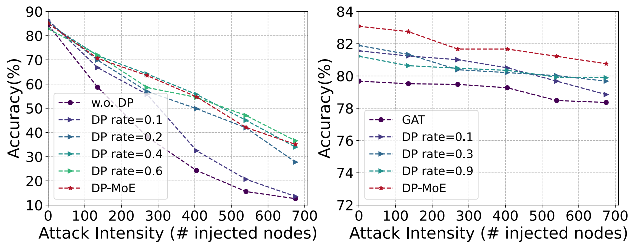

Figure 3: Different DP rates (scaling coefficient) on DRAGON w. single DP rate and w. multiple DP rates via DPMoE using standard training (left) and adversarial training (right) on *Cora* dataset.

comparison between the attacked graph input and the denoised version produced by DMGAN in Figure 2. The results show that as the attack intensity increases, thanks to the link pattern representation ability learned by DMGAN, it continues to effectively remove a large number of malicious edges that are associated with the injected nodes, thereby reducing the negative impact of the attack.

Effectiveness of DPMoE. It is essential to note that relying solely on the DP mechanism in a normal GNN backbone, without MoE, presents challenges in achieving a balance between performance in attack and non-attack scenarios. As shown in Figure 3, models with higher DP scaling coefficients exhibit flatter performance curves, indicating that they are robust in high-intensity attack scenarios but less accurate in non-attack scenarios. Therefore, to achieve a sweeter trade-off in both non-attack scenarios and attack scenarios of varying intensity, DPMoE is introduced to dynamically balance between the two scenarios. This is achieved by using multiple experts to integrate DP scaling coefficients of different magnitudes for different scenarios. This allows the activation of the appropriate expert network with the appropriate DP scaling coefficient, providing a sweet trade-off solution for both non-attack and attack scenarios of varying intensities: on the one hand, in non-attack scenarios, DPMoE activates the expert with minimal DP noise; on the other hand, as the attack intensity increases, it activates the expert with larger magnitudes of DP noise.

5 Conclusion

In this paper, we first identify two practical issues: degraded robustness and limited scalability in current adversarial graph learning. To address them simultaneously, we propose a novel framework named **DRAGON** by utilizing a denoising masked graph auto-encoder and a differential privacy mechanism. Our experimental results show the effectiveness of DRAGON in denoising malicious edges, counteracting the injected attack node features through differential privacy, and navigating graph data of varying scales and attacks of different intensities. Overall, DRAGON is a robust and scalable solution for improving the robustness of GNNs in real-world applications.

References

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., and Xuan, Q. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

Chen, Y., Yang, H., Zhang, Y., Ma, K., Liu, T., Han, B., and Cheng, J. Understanding and improving graph injection attack by promoting unnoticeability. In *ICLR*, 2022.

Du, J., Zhang, S., Wu, G., Moura, J. M., and Kar, S. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*, 2020.

Gao, H., Wang, Z., and Ji, S. Large-scale learnable graph convolutional networks. In *KDD*, 2018.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. Graph structure learning for robust graph neural networks. In *KDD*, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.

Lei, R., Zhen, W., Li, Y., Ding, B., and Wei, Z. Evennet: Ignoring odd-hop neighbors improves robustness of graph neural networks. In *NeurIPS*, 2022.

Li, J., Wu, R., Sun, W., Chen, L., Tian, S., Zhu, L., Meng, C., Zheng, Z., and Wang, W. Maskgae: Masked graph modeling meets graph autoencoders. *arXiv preprint arXiv:2205.10053*, 2022.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD*, 2014.

Qinkai, Z., Yixiao, F., Yanhao, L., Liu Qingmin, H. M., and Qibo, S. Kdd cup 2020 ml track 2 adversarial attacks and defense on academic graph 1st place solution. https://github.com/Stanislas0/KDD_CUP_2020_MLTrack2_SPEIT, 2020.

Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *WWW*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*, 2018.

Wang, J., Luo, M., Suya, F., Li, J., Yang, Z., and Zheng, Q. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery*, 34(5): 1363–1389, 2020.

Wang, X., He, X., Cao, Y., Liu, M., and Chua, T.-S. Kgat: Knowledge graph attention network for recommendation. In *KDD*, 2019.

Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, 2018.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *ICML*, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

Zhang, C., Tian, Y., Ju, M., Liu, Z., Ye, Y., Chawla, N., and Zhang, C. Chasing all-round graph representation robustness: Model, training, and optimization. In *ICLR*, 2023.

Zhang, X. and Zitnik, M. Gnnguard: Defending graph neural networks against adversarial attacks. In *NeurIPS*, 2020.

Zheng, Q., Zou, X., Dong, Y., Cen, Y., Yin, D., Xu, J., Yang, Y., and Tang, J. Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning. In *NeurIPS on Datasets and Benchmarks*, 2021.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. Robust graph convolutional networks against adversarial attacks. In *KDD*, 2019.

Zou, X., Zheng, Q., Dong, Y., Guan, X., Kharlamov, E., Lu, J., and Tang, J. Tdgia: Effective injection attacks on graph neural networks. In *KDD*, 2021.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *KDD*, 2018.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

A Related Work

Graph Neural Networks (GNNs). GNNs have achieved outstanding performance in various graph mining tasks (Hamilton et al., 2017; Battaglia et al., 2018; Wu et al., 2020) due to their ability to effectively learn non-Euclidean data. Early works related to GNN proposed such as graph convolutional networks (GCN) are proposed to apply convolutional concepts to graph data (Kipf & Welling, 2017; Gao et al., 2018; Wu et al., 2019). Subsequently, graph attention networks (Veličković et al., 2018; Wang et al., 2019) were introduced by incorporating attention mechanisms. In addition, masked autoencoder (He et al., 2022) has been introduced into the graph domain in a self-supervised manner(Li et al., 2022), where we extend its philosophy to construct clean graphs from attacked graphs.

Adversarial Learning on Graphs. A variety of graph attack methods have been proposed to disrupt the structural or semantic properties of the graph, including inserting or removing connections (Du et al., 2017; Chen et al., 2018; Waniek et al., 2018), perturbing node features (Zügner et al., 2018; Zügner & Günnemann, 2019; Sun et al., 2020), or adding malicious nodes (Wang et al., 2020; Zou et al., 2021). In response to attacks, researchers have proposed a range of defense methods for GNNs including inserting or removing connections (Du et al., 2017; Chen et al., 2018; Waniek et al., 2018), perturbing node features (Zügner et al., 2018; Zügner & Günnemann, 2019; Sun et al., 2020), or adding malicious nodes (Wang et al., 2020; Zou et al., 2021) However, two fundamental challenges, namely degraded robustness caused by extremely intense attacks and limited scalability on large graph datasets, remain unresolved. To the best of our knowledge, our framework DRAGON, which combines masked graph auto-encoder design and MoE mechanism with differential privacy noises, is the first work that addresses these challenges simultaneously.

Therefore, it is crucial to develop more robust graph defense methods and to promote responsible deployment and management of GNNs. From this perspective, our proposed new graph defense framework, DRAGON, facilitates future research on trustworthy graph neural networks. Future research and applications need to consider comprehensive and appropriate strategies to deal with possible impacts and threats.

B Additional Performance Comparisons

We also test the robustness of DRAGON under the PGD attack and the SPEIT attack. The results are shown in Figure 6 and Figure 4. The figures show our method outperforms all baselines under representative attacks as a scalable and robust framework for GNNs.

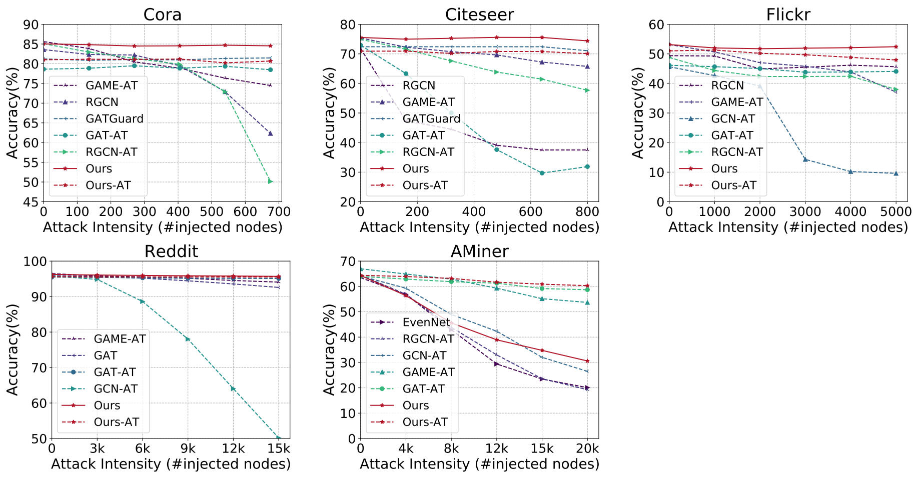

Figure 4: The performance of top-5 baselines and our method under the SPEIT Attack.

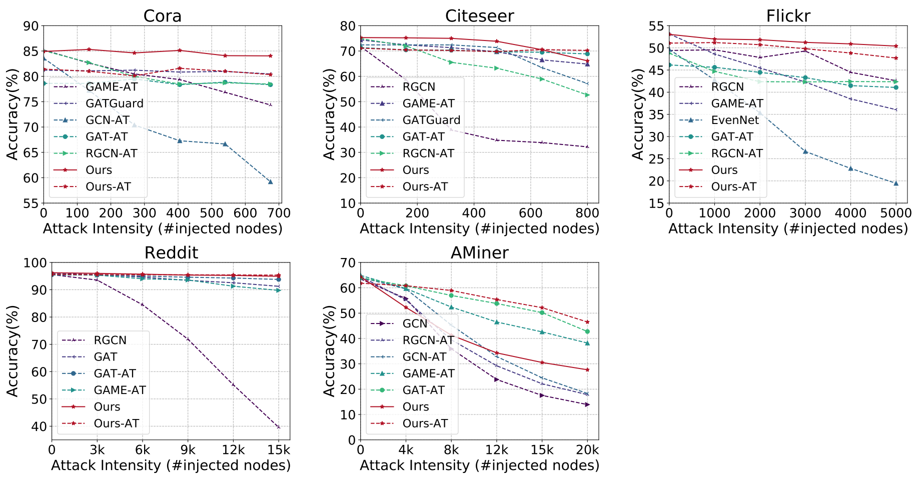

Figure 5: The performance of top-5 baselines and our method under the PGD Attack.

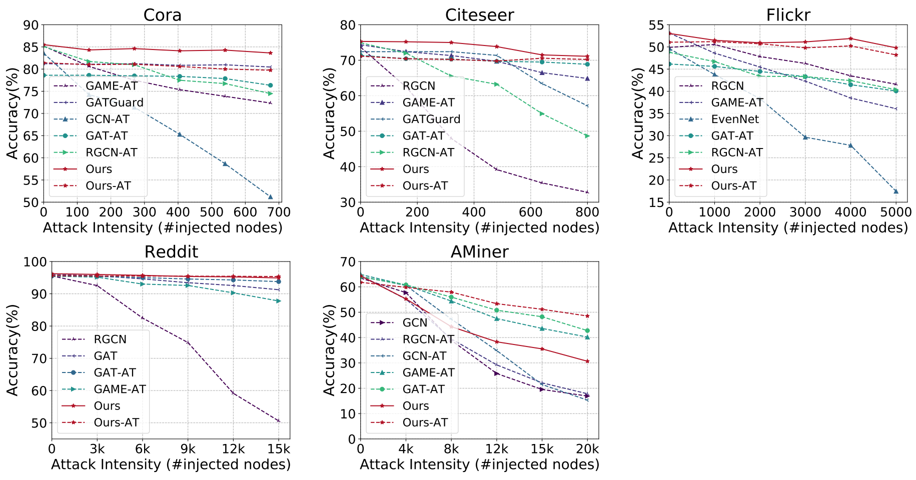

Figure 6: The performance of top-5 baselines and our method under the TDGIA Attack.

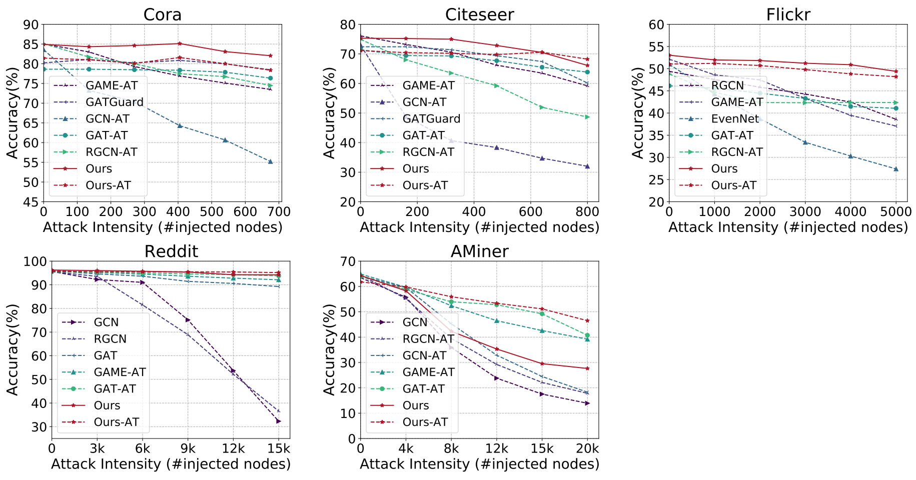

Figure 7: The performance of top-5 baselines and our method under the HAO Attack.

C Proof

In this section, we will deliver a comprehensive proof and analysis outlining the robustness of the DPMoE module. Please note that the proof of Proposition C.1 can be adapted from Lemma 1 in (Lecuyer et al., 2019), with notations unified for the purpose of this paper.

Let $B_p(r)$ represents a p-norm ball with radius r, such that $B_p(r) = \{\Delta h_v^{(l)} \in \mathbb{R}^N : \|\Delta h_v^{(l)}\|_p \leq r\}$, where $\Delta h_v^{(l)}$ is the noise that modify $h_v^{(l)}$ to $h_v^{'(l)}$ at layer l ($l > 0$) after message passing for node v. If the node classification model $f(\cdot)$, where $f(h_v^{(l)}) = k$, remains robust to this attack, we have

$$f_k(h_v^{'(l)}) > \max_{i:i\neq k} f_i(h_v^{'(l)}), \forall \Delta h_v^{(l)} \in B_p(r). \tag{10}$$

The subsequent Proposition C.1 and Lemma C.2 present how Equation (10) is satisfied, thereby ensuring model robustness.

Proposition C.1. (Expected Output Bound) *Suppose a GNN $f(\cdot)$ that contains DP-GConv and has bounded output $f(h_v^{(l)}) \in [0, b],\ b \in \mathbb{R}^+$, satisfies (ϵ, δ)-DP. Then, the expected output $\mathbb{E}$ of its output has the following property:*

$$\mathbb{E}(f(h_v^{(l)})) \leq e^\epsilon \mathbb{E}(f(h_v^{'(l)}) + b\delta, \forall \Delta h_v^{(l)} \in B_p(b). \tag{11}$$

Proof. Consider $\triangle h_v^{(l)} \in B_p(b)$ that modify $h_v^{(l)}$ to $h_v^{'(l)}$, the expected output of $f(\cdot)$ is:

$$\mathbb{E}(f(h_v^{(l)})) = \int_0^b Pr(f(h_v^{(l)}) > t)dt. \tag{12}$$

The post-processing property points out that, any computation applied to the output of an (ϵ, δ)-DP algorithm remains (ϵ, δ)-DP. According to this property, we can apply Equation (2) with the (ϵ, δ)-DP property to $f(\cdot)$ and obtain:

$$\begin{aligned}\mathbb{E}(f(h_v^{(l)})) &\leq e^\epsilon \Big(\int_0^b Pr(f(h_v^{'(l)}) > t)dt\Big) + \int_0^b \delta dt \\ &= e^\epsilon \mathbb{E}(f(h_v^{'(l)})) + \int_0^b \delta dt \\ &= e^\epsilon \mathbb{E}(f(h_v^{'(l)})) + b\delta.\end{aligned} \tag{13}$$

□

Assuming that $f(\cdot)$ with DP-GConv satisfies (ϵ, δ)-DP with respect to a p-norm metric, where $f(h_v^{(l)}) = (f_1(h_v^{(l)}), \ldots, f_K(h_v^{(l)})),\ f_k(h_v^{(l)}) \in [0, 1]$, we apply Proposition C.1 with $b = 1$ to k:

$$\mathbb{E}(f_k(h_v^{(l)})) \leq e^\epsilon \mathbb{E}(f_k(h_v^{'(l)})) + \delta, \forall k, \forall \Delta h_v^{(l)} \in B_p(1). \tag{14}$$

Following Proposition C.1 and Equation (14), we next prove that DPMoE module containing DP-GConv provides robustness. The proof of **Lemma 3.1** in **Section 4.2** is provided:

Lemma C.2. Robustness of DPMoE. *Suppose a GNN $f(\cdot)$ containing DPMoE satisfies (ϵ, δ)-DP. The model with label probability output vector $p(h_v^{'(l)}) = (\mathbb{E}(f_1(h_v^{'(l)})), \ldots, \mathbb{E}(f_K(h_v^{'(l)})))$ for node v is robust to features $h_v^{'(l)}$ after node injection attack on layer l, if some $k \in \mathcal{K}$ and the expected value $\mathbb{E}$ of output satisfies the following property:*

$$\mathbb{E}(f_k(h_v^{'(l)})) > e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(f_i(h_v^{'(l)})) + (1 + e^\epsilon)\delta. \tag{15}$$

Proof. First we consider GNN $f(\cdot)$ containing only DP-GConv. According to Equation (14), we obtain:

$$\mathbb{E}(f_k(h_v^{(l)}) \leq e^\epsilon \mathbb{E}(f_k(h_v^{'(l)}) + \delta, \tag{16}$$

$$\mathbb{E}(f_i(h_v^{'(l)}) \leq e^\epsilon \mathbb{E}(f_i(h_v^{(l)}) + \delta. \tag{17}$$

Equation (16) gives a lower-bound on $\mathbb{E}(f_k(h_v^{'(l)})$ and Equation (17) gives an upper-bound on $\max_{i \neq k} \mathbb{E}(f_i(h_v^{'(l)})$, thus we further obtain:

$$\begin{aligned}
\mathbb{E}(f_k(h_v^{'(l)})) &\overset{\text{Eq(16)}}{\geq} \frac{\mathbb{E}(f_k(h_v^{(l)})) - \delta}{e^{\epsilon}} \\
&\overset{\text{Eq(15)}}{>} \frac{e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(f_i(h_v^{(l)})) + (1+e^{\epsilon})\delta - \delta}{e^{\epsilon}} \\
&= e^{\epsilon} \max_{i:i\neq k} \mathbb{E}(f_i(h_v^{(l)})) + \delta \\
&\overset{\text{Eq(17)}}{\geq} \max_{i:i\neq k} \mathbb{E}(f_i(h_v^{'(l)})),
\end{aligned} \tag{18}$$

which is in line with Equation (10), implying that model $f(\cdot)$ containing DP-GConv is robust. Then we prove $f(\cdot)$ containing DPMoE is still robust. Simplifying Equation (8) to $h_v^{(l+1)} = \text{Update}(h_v^{(l)})$ and substitute it into Equation (15), we obtain:

$$\mathbb{E}(\text{Update}(f_k(h_v^{(l)}))) > e^{2\epsilon} \max_{i:i\neq k} \mathbb{E}(\text{Update}(f_i(h_v^{(l)})) + (1+e^{\epsilon}))\delta. \tag{19}$$

Similar to the previous proof, if $f(\cdot)$ containing DPMoE satisfies Equation (19), $f(\cdot)$ is robust. $\square$

Lemma C.2 elucidates the relationship between robustness and perturbations. This correlation facilitates the existence of a maximum solution $\Delta h_{v(max)}^{(l)}$ which guarantees the robustness of the model. After injection attacks, denote $h_v^{(l)}$ as follows:

$$h_v^{(l)} = \text{COM}^{(l)} \left(\text{DPMoE}^{(l)} h_v^{(l-1)},\ \text{AGG} \left(\left\{ \text{DPMoE}^{(l)} h_u^{(l-1)}, \text{DPMoE}^{(l)} h_w^{(l-1)} \right\} \right) \right), \tag{20}$$

where $\forall u \in N_v \wedge u \in G, \forall w \in N_v \wedge w \in G^{'} \wedge w \notin G$. Then denote attack method as $\text{Att}(\cdot)$ and the attacked graph as follows:

$$G^{'} = \text{Att}(\Delta E, \Delta N, G), \tag{21}$$

where ΔE is the budget of edges per injected node,ΔN is the budget of injected nodes, and G is input graph. According to the Lemma C.2, exists a maximum solution $\Delta h_{v(max)}^{(l)}$ to ensure the model is certified robust when Equation (15) and (ϵ, δ)-DP in DP-GConv hold, where $\Delta h_{v(max)}^{(l)} =$ (8) $-$ (20). And solve Equation (15), Equation (20) and $\Delta h_{v(max)}^{(l)}$ there also exists a maximum ΔN to ensure the model is robust when ΔE is fixed. Note we don't need to inversely solve explicit maximum ΔN.

D Implementation Details

D.1 Reproducibility Settings

Training and Evaluation Configuration. In our experiments, we maintain consistency with the default hyperparameters of the GRB benchmark for the baseline methods. To ensure reliable results, we conduct 5 runs for each experimental result and report the mean value and standard deviation. Additionally, we adhere to the GRB benchmark's data splitting protocol, with 60% of the graph data as the training set, 10% as the validation set, and 30% as the test set for each benchmark dataset. Statistics of GRB data covering small to large graphs are listed in Table 4. All training and evaluation are performed on an NVIDIA V100 GPU with 32 GB of memory. The code can be accessed through this anonymous link[1].

To ensure reproducibility, we follow the hyperparameter settings of baselines used in GRB (Zheng et al., 2021) and other original papers (Chen et al., 2022) (Zhang et al., 2023). The hyperparameters of DPMoE are given in Table 5, including hyperparameters of adversarial training listed in Table 3 Specifically, for *Cora*, we set the total number of experts to $N = 10$ and the number of activated experts to $k = 2$. For other datasets, by default, we set the total number of experts to $N = 4$ and the number of activated experts to $k = 1$. Note that the DP scaling coefficient μ_i for each individual expert is *linearly* increased via multiplying the minimum coefficient by i as the index i of that expert increases. Additionally, hyperparameters of DMGAN are shown in Table 8. The mask rate is 0.7, the walks per node is 1, and the walk length is 3.

[1] https://www.dropbox.com/sh/l3ekm0epdw1mal4/AABnQsbqxJ1dVzPuzCJSkkz1a?dl=0

Table 3: Hyperparameters for adversarial training. The settings follow the GRB benchmark.

Datasets	*Cora*	*Citeseer*	*Flickr*	*Reddit*	*AMiner*
Injections	20	30	200	500	500
Edges	20	20	100	200	100
Step size	0.01	0.01	0.01	0.01	0.01
Iteration	10	10	10	10	10
Attack	FGSM	FGSM	FGSM	FGSM	FGSM

Table 4: Statistics of GRB data covering small to large graphs.

GRB Datasets	Scale	Nodes	Edges	Feat.	Classes	Feat. Range (normalized)
Cora	Small	2,680	5,148	302	7	[-0.94, 0.94]
Citeseer	Small	3,191	4,172	768	6	[-0.96, 0.89]
Flickr	Medium	89,250	449,878	500	7	[-0.47, 1.00]
Reddit	Large	232,965	11,606,919	602	41	[-0.98, 0.99]
AMiner	Large	659,574	2,878,577	100	18	[-0.93, 0.93]

Table 5: Hyperparameters of the DPMoE defender GNN.

Datasets	Layer num	Hidden size	Heads	Dropout	LR		Backbone layer		Minimal DP rate	
					w.o. AT	w. AT	w.o. AT	w. AT	w.o. AT	w. AT
Cora	3	128	4	0.5	0.001	0.01	GATGuard	GAT	0.3	0.1
Citeseer	3	64	6	0.5	0.001	0.01	GATGuard	GAT	0.3	0.1
Flickr	3	64	8	0.5	0.0001	0.0001	GAT	GAT	0.3	0.1
Reddit	2	64	8	0.5	0.01	0.01	GAT	GAT	0.1	0.1
AMiner	3	64	4	0.5	0.01	0.01	GAT	GAT	0.1	0.1

D.2 More Details about Attack Strategies

We examine three effective and diverse graph attack methods that can impair the performance of victim GNNs, including our proposed DRAGON framework and other baselines. The details of these attack strategies are listed as follows:

- **FGSM**: The Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) computes the optimal max-norm constrained perturbation as attacks by linearizing the loss function around the current value of the parameters.
- **SPEIT**: SPEIT (Qinkai et al., 2020) emerges as the first-place winner in the KDD-CUP 2020 Graph Adversarial Attack & Defense competition. It generates perturbed adjacent matrix and feature gradient attacks for a global black-box node injection attack.
- **PGD**: The Projected Gradient Descent (PGD) (Madry et al., 2018) is an adversary method that leverages local first-order gradient information about the network to generate the strongest attack inputs.
- **TDGIA**: Topological Defective Graph Injection Attack (Zou et al., 2021) introduces the topological defective edge selection strategy and the smooth feature optimization objective to generate the features for the injected nodes.
- **HAO**: Harmonious Adversarial Objective (HAO)(Chen et al., 2022) introduces homophily unnoticeability that enforces graph injection attack to preserve the homophily, thereby enabling stronger attacks.

In order to generate gradient-based attacks, we follow previous research (Zheng et al., 2021) and use GCN as the surrogate model. To enhance the strength, stability, and transferability of the attacks, we also add layer normalization (LN) layers provided by the GRB benchmark to the GCN surrogate model. The hyperparameters of the surrogate model are presented in Table 6. The step size is 0.01 and iteration is 1000 when attacks are generated.

And we provide details of node injection attack intensities from $nulla$ to V in Table 7.

Table 6: Hyperparameters of the surrogate model. The settings follow the GRB benchmark.

Datasets	*Cora*	*Citeseer*	*Flickr*	*Reddit*	*AMiner*
Hidden Size	64	64	128	128	128
Layer Number	3	3	3	3	3
Learning Rate	0.01	0.01	0.01	0.01	0.01
Dropout	0.5	0.5	0.5	0.5	0.5

Table 7: The configurations about attacks of five intensities and without attacks: # node represent the number of injected nodes, # edge/n. represent the max number of edges per injected node, and nulla represents *without* any attack injections into the graph input.

Dataset	Injection	Attack Intensity					
		nulla	I	II	III	IV	V
Cora	# node	0	1*135	2*135	3*135	4*135	5*135
	# edge/n.	0	20	20	20	20	20
Citeseer	# node	0	1*160	1*160	1*160	1*160	1*160
	# edge/n.	0	20	20	20	20	20
Flickr	# node	0	1*1000	2*1000	3*1000	4*1000	5*1000
	# edge/n.	0	100	100	100	100	100
Reddit	# node	0	1*3000	2*3000	3*3000	4*3000	5*3000
	# edge/n.	0	200	200	200	200	200
AMiner	# node	0	1*4000	2*4000	3*4000	4*4000	5*4000
	# edge/n.	0	100	100	100	100	100

D.3 Hyperparameters of DMGAN

According to Figure 8, the trade-off associated with using DMGAN as a denoising module needs to be considered. There is a risk of removing the original edges along with the malicious edges, which may affect the performance of the model when it is used in a non-attack setting. After careful study, we find that a reconstruction rate of 0.3 provides a good balance between these trade-offs. For example, compared with DRAGON integrated with a vanilla graph autoencoder, our DRAGON integrated with DMGAN sacrifices only little accuracy under non-attack settings, but shows a larger accuracy improvement when the graph is under attack. The improvement of our DRAGON increases as the attack intensity increases, demonstrating the robustness of our DPMoE design from several perspectives. And we also provide statistics of reconstruct error rates for Figure 2 in Table 9.

Table 8: Hyperparameters of DMGAN. We use a simple graph auto-encoder model design to ensure scalability.

Auto-encoder	Model	Layers Num	Hidden Size	LR	Dropout
Encoder	GCN	1	128	0.01	0.8
Decoder	MLP	2	64	0.01	0.2

Table 9: Edge error rates of DMGAN reconstructed graph

Intens.	nulla	I	II	III	IV	V
Normal edges	0.046	0.059	0.065	0.069	0.071	0.074
Malicious edges	0.000	0.638	0.674	0.691	0.724	0.749

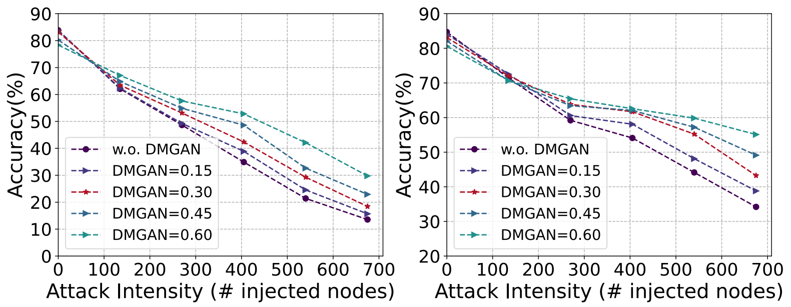

Figure 8: The effect of DMGAN with different reconstruction rates on the performance of GAT (left) and DPMoE (right) as defender GNNs in the Cora dataset under the FGSM attack. (The DPMoE module uses the GAT layer as the backbone layer of each expert network.)

E Defense against Graph Modification Attacks

In the main body of the paper, we mainly present the performance of DRAGON under graph injection attacks. To evaluate the effectiveness of DRAGON under graph modification attacks, we apply GR-BCD and BR-BCD attacks on our method and baselines, including SOTAs (Soft-Medoid/Soft-Median GDCs and GAME). The results are summarizedin Table 9.

Figure 9: The robust accuracy of Soft-Median GDC and Soft-Medoid GDC without or with our GAME framework on the Cora dataset with the global attacks (GR-BCD PR-BCD, $\epsilon = 0.1$) proposed by Soft-Medoid GDC. We set the number of experts to 10 and the hidden units of each expert to 32. We run them on three random splits and report the mean and standard error results.

	GR-BCD	PR-BCD
GCN	0.622 ± 0.003	0.645 ± 0.002
GDC	0.677 ± 0.005	0.674 ± 0.004
PPRGo	0.726 ± 0.002	0.700 ± 0.002
SVD GCN	0.755 ± 0.006	0.724 ± 0.006
Jaccard GCN	0.664 ± 0.001	0.667 ± 0.003
RGCN	0.665 ± 0.005	0.664 ± 0.004
Soft-Median GDC	0.765 ± 0.001	0.752 ± 0.002
Soft-Medoid GDC	0.775 ± 0.003	0.761 ± 0.003
Soft-Median GDC (+GAME)	0.772 ± 0.005	0.759 ± 0.005
Soft-Medoid GDC (+GAME)	0.780 ± 0.007	0.772 ± 0.006
DRAGON	0.788 ± 0.012	0.784 ± 0.007

F More Detailed Ablation Study

Here we provide a more detailed ablation study on Cora, Citeseer and AMiner datasets. DRAGON integrates the denoising preprocessing component (DMGAN) and the defender component (DPMoE) into a scalable and anti-degraded robust graph learning framework. Thus, to answer **RQ-2**, we remove each of the components in DRAGON (to simplify the analysis for reliable conclusions, we uniformly standardize the GAT as the expert backbone in ablation studies, which is slightly different from the expert backbone used in the main results) and perform experiments to observe performance. We denote the model ablated by DRAGON as (a) without denoise, (b) without DPMoE, and (c) the GAT backbone, i.e., the base model without both denoise and DPMoE, as shown in Table 10.

Our experiments showed that removing any component of the DRAGON decreases its performance in node injection attacks, particularly as the attack intensity increases. This highlights the critical role each component plays for robustness of the model and its ability to provide anti-degraded robustness. For (a) w.o. denoise, removing the DMGAN component significantly decreases DRAGON's ability to recover clean graph information from an attacked graph input. For example, on the small grb-cora dataset, the model experienced a 0.3% loss in accuracy when the attack intensity is I, and a 6.8% loss in accuracy when the attack intensity increases to V. This illustrates that a cleaner graph reconstructed by DMGAN is easier for defense. For (b) w.o. DPMoE, disabling the DPMoE component entirely impairs DRAGON's ability to manipulate Gaussian DP noises of varying magnitudes to counteract attacks of varying intensities. On the grb-citeseer dataset, the removal of this component led to a 23.2% loss in accuracy when the attack intensity is I, and a 29.5% loss in accuracy when the attack intensity increases to V. This demonstrates that DPMoE improves the robustness of GNNs against adversarial node injections of varying strengths by providing diverse yet effective Gaussian DP noises. Finally, for (c) w.o. denoise and DPMoE, removing both components degenerate DRAGON into a vanilla GAT model. On the large grb-aminer dataset, this resulted in a 2.5% increase in accuracy when the attack intensity is I, and a 15.9% loss in accuracy when the attack intensity increases to V. This shows the effectiveness of the DRAGON framework in denoising and enhancing a base model's capacity to learn robust representations against attacks of different intensities.

Table 10: Ablation studies for DRAGON on graphs of varying scales and under FGSM attack of varying intensities. Base model denotes backbone w.o. DRAGON.

	Int.	Base model	w.o.denoise	w.o.DPMoE	DRAGON
	nulla	83.6±0.5	83.7±0.2	83.6±0.6	83.2±0.3
	I	58.9±4.2	70.7±1.3	63.4±1.4	**71.0±0.4**
Cora	II	38.9±3.7	63.4±1.9	53.9±0.8	**65.1±1.6**
(small)	III	24.3±4.3	54.1±3.7	42.8±0.7	**58.6±2.0**
	IV	15.8±2.5	41.6±1.3	29.5±1.6	**48.4±1.3**
	V	14.3±1.6	34.0±2.1	18.7±2.3	**40.8±2.9**
	nulla	72.1±0.6	**74.9±0.5**	70.5±0.3	73.4±0.5
	I	45.4±0.6	67.4±3.1	44.3±1.1	**67.5±1.8**
Citeseer	II	21.1±2.5	50.7±0.7	23.3±2.0	**55.5±2.1**
(small)	III	20.9±3.8	47.4±3.5	22.4±1.2	**51.2±1.8**
	IV	19.6±5.6	35.7±2.5	21.5±2.7	**47.4±0.7**
	V	13.5±4.7	37.0±2.4	15.3±3.2	**44.8±1.9**
	nulla	**66.8±0.1**	66.3±0.1	64.8±0.0	64.3±0.8
	I	59.3±0.5	61.7±0.4	60.9±0.8	**61.7±0.5**
AMiner	II	46.9±0.3	52.4±0.7	51.9±0.2	**56.1±0.2**
(large)	III	35.6±0.1	42.4±1.7	40.9±0.4	**49.0±0.4**
	IV	30.1±1.2	34.0±0.5	34.6±0.3	**42.5±0.3**
	V	23.8±0.7	26.5±2.2	28.1±0.9	**39.7±0.2**