Wu XG, Wu HJ, Zhou X *et al.* Towards defense against adversarial attacks on graph neural networks via calibrated co-training. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(5): 1161–1175 Sept. 2022. DOI 10.1007/s11390-022-2129-2

Towards Defense Against Adversarial Attacks on Graph Neural Networks via Calibrated Co-Training

Xu-Gang Wu[1] (吴旭刚), Hui-Jun Wu[1,*] (邬会军), *Member, CCF*, Xu Zhou[1] (周 旭), *Member, CCF*
Xiang Zhao[2] (赵 翔), *Senior Member, CCF*, and Kai Lu[1] (卢 凯), *Senior Member, CCF*

[1] *College of Computer, National University of Defense Technology, Changsha 410073, China*

[2] *College of Systems Engineering, National University of Defense Technology, Changsha 410073, China*

E-mail: {wuxugang13, wuhuijun, zhouxu, zhaoxiang, kailu}@nudt.edu.cn

Received December 31, 2021; accepted September 27, 2022.

Abstract Graph neural networks (GNNs) have achieved significant success in graph representation learning. Nevertheless, the recent work indicates that current GNNs are vulnerable to adversarial perturbations, in particular structural perturbations. This, therefore, narrows the application of GNN models in real-world scenarios. Such vulnerability can be attributed to the model's excessive reliance on incomplete data views (e.g., graph convolutional networks (GCNs) heavily rely on graph structures to make predictions). By integrating the information from multiple perspectives, this problem can be effectively addressed, and typical views of graphs include the node feature view and the graph structure view. In this paper, we propose C^{2}oG, which combines these two typical views to train sub-models and fuses their knowledge through co-training. Due to the orthogonality of the views, sub-models in the feature view tend to be robust against the perturbations targeted at sub-models in the structure view. C^{2}oG allows sub-models to correct one another mutually and thus enhance the robustness of their ensembles. In our evaluations, C^{2}oG significantly improves the robustness of graph models against adversarial attacks without sacrificing their performance on clean datasets.

Keywords adversarial defense, graph neural network, multi-view, co-training

1 Introduction

Graph neural networks (GNNs) achieved remarkable performance in analyzing graph data, such as citation networks, biological networks, and social networks. The graph convolutional network (GCN) and its variants [1–4] have attracted considerable attention due to their high performance and efficiency. However, recent studies have demonstrated that these message-passing based models are subject to adversarial perturbations [5–9]. When an adversary conducts unnoticeable alterations to the graph data, the accuracy of the learned model drops dramatically. Compared with feature perturbations, structural perturbations are more effective in conducting successful attacks [7–9].

To enhance the robustness of GNNs against structural perturbations, some defense techniques have been proposed. A general principle is to eliminate the detrimental impacts of perturbed edges. For this purpose, some defense methods [9–12] assume that all the connected vertexes should be similar in the feature space. Based on this assumption, they calculate the pairwise similarity scores between connected nodes and then delete or pay less attention to edges that connect dissimilar nodes. However, this heuristic is not applicable to heterophilic graphs. Other methods [13, 14] note that structural perturbations tend to affect the high-rank portion of the graph. As a result, they replace the graph topology with its low-rank approximation to purify the graph. Nevertheless, recent work shows that such a heuristic is sub-optimal [15]. In summary, heuristic approaches are often model-dependent and they may

Regular Paper

This work was partially supported by the National University of Defense Technology Foundation under Grant Nos. ZK20-09 and ZK21-17, and the Natural Science Foundation of Hunan Province of China under Grant No. 2021JJ40692.

*Corresponding Author

also introduce unexpected bias.

In the literature of adversarial learning in computer vision, model ensembles are commonly used as defense methods. Recent work [16–18] points out that output diversity is the key to the success of these ensemble-based defense methods. However, applying ensemble techniques to GNNs straightforwardly can hardly improve the robustness of the ensemble model. To illustrate this phenomenon, we evaluate the classification accuracy of the ensemble of GCNs and GATs under Metattack [8]. Fig.1 shows that Metattack can transfer between GCNs and GATs well, leading to the poor robustness of their ensembles.

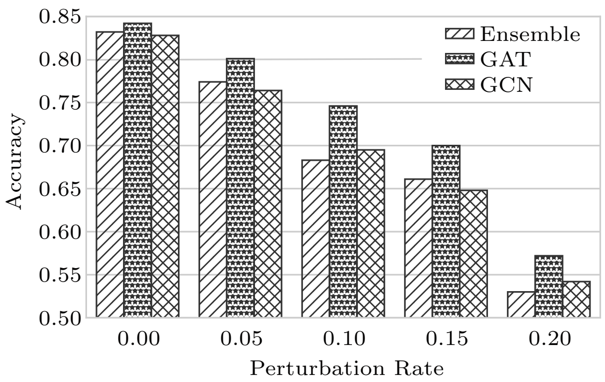


Fig.1. Classification accuracy of ensemble of GCNs and GATs under Metattack on Cora [19].

In this paper, we propose that introducing diversified models that share different views of graph data will help to improve the overall robustness of the ensemble. Compared with the image data, the graph data have two complementary views: the node feature view and the graph structure view. The complementarity of these two views makes it difficult for adversarial attacks to transfer between them. However, we find the vanilla ensemble approach achieves sub-optimal results (see the details in Subsection 5.5.2). This is mainly due to the fact that the vanilla ensemble only aggregates the final results of trained sub-models but misses the opportunities to fully exchange knowledge between the sub-models during training.

Therefore, to fully exploit this property, our work uses a calibrated co-training framework on graph (C^2oG) to learn an ensemble of sub-models from both the feature view and the structure view. Specifically, co-training [20, 21] is a simple yet effective technique for learning an ensemble of sub-models in multiple different views under the semi-supervised setting. It trains a separate classifier for each view and adds the most confident predictions of each sub-model to the training dataset. An ensemble obtained from co-training enables the sub-models to correct each other during the training stage and can potentially achieve better results.

Nevertheless, applying the vanilla co-training framework to graph data faces two challenges. 1) Co-training uses the softmax outputs as the indicators of sub-models' confidence. However, this can be inaccurate since neural networks are often miscalibrated, especially when sub-models are heterogeneous. 2) Co-training selects unlabeled data simply based on their confidence. If dominant classes exist, the co-training process will amplify the imbalance of classes and force the sub-models to overfit to the dominant class. To address these issues, we use temperature scaling [22] to calibrate the outputs, and enforce the consistency of class distribution when adding predictions during the co-training process.

Our evaluation results show that C^2oG could incorporate the knowledge of the sub-models trained on the two views to significantly alleviate the impact of adversarial perturbations. Our contributions are summarized as follows.

- We propose C^2oG, which is a calibrated co-training framework, to combine the feature information and the structure information of graphs in a holistic manner. C^2oG is easy-to-implement and model-agnostic.
- We highlight the incomparable confidence and imbalanced training set challenges in the co-training framework and propose to use model calibration and class balancing mechanisms to address these problems, which further improve the performance of C^2oG.
- Experiments show that C^2oG consistently outperforms the state-of-the-art baselines under different perturbation ratios. Moreover, our defense can still work well in adaptive attack settings where the defense internals are exposed to the attackers.

2 Related Work

2.1 Attack and Defense on Graph Data

Despite the great success of GNNs, recent work shows that these graph-based models are vulnerable to unnoticeable modifications [5, 6, 8]. Nettack [6] conducts its attack on a surrogate model and ensures the edge perturbations and the feature perturbations to be unnoticeable via considering the degree distribution and feature co-occurrence. RL-S2V [5] applies reinforcement learning to generate adversarial examples. Metattack [8] addresses the global attack problem. It uses the meta-gradient to generate a perturbed graph that leads to an overall decrease in models' performance. The results of these attacks suggest that structural attacks are more effective than feature attacks when applied to the graph data.

Several techniques have been proposed to defend against these topological attacks[9–13, 23]. GCN-Jaccard[9] assumes that connected nodes have similar features so that edges between dissimilar nodes are more likely to be the perturbed edges. Based on this assumption, GCN-Jaccard calculates the Jaccard similarity scores between connected nodes and drop edges that connect nodes with scores below the preset threshold. Similarly, GNNGuard[10] employs the attention mechanism to assign higher weights to the edges between similar nodes and lower weights to the edges between unrelated nodes. Other work stems from the observation that structural attacks tend to affect the high-rank portion of a graph. GCN-SVD[14] was proposed to purify the perturbed graph via replacing it with its low-rank approximation, while Pro-GNN[13] introduces a regularization term to generate a low-rank and sparse graph during the training process. However, these methods rely on the validity of their heuristic knowledge. SimP-GCN[23] attempts to integrate the structure information and node features by combining the k-NN graph and the original graph. Nevertheless, the scoring function that balances these two graphs merely depends on the hidden representation so that the graph used for learning could be unstable, thus leading to high variance for the results.

The most related defense method to our work is UM-GNN[24]. UM-GNN was proposed to learn a feature-based model via distilling knowledge from the GNN model using an uncertainty matching strategy. However, its knowledge distillation is one-directional: only the feature-based model can distill knowledge from the GNN model. Consequently, the GNN model cannot get enhanced using the information from the feature-based model. As the perturbation rate grows, the knowledge transferred from the GNN model becomes less effective, which impairs the performance of UM-GNN.

2.2 Ensemble Training for Enhanced Robustness

Although ensemble training was initially proposed to improve models' performance[25–28], a recent line of work[16–18] shows that it can be used as the defense against adversarial attacks. The intuition behind the defense methods[16–18] is that a small overlap between adversarial subspaces (Adv-SS) of different sub-models can prevent adversarial attacks from transferring between sub-models. Pang *et al.*[16] employed an adaptive diversity-promoting regularizer to encourage diversity among non-maximal predictions. Kariyappa and Qureshi[17] proposed diversity training to reduce the correlation of loss functions between sub-models. Yang *et al.*[18] distilled the non-robust features from each sub-model and taught the other sub-models to be robust against these non-robust features. Although these defense approaches have been well studied in image recognition tasks, their application in graph-based tasks remains to be explored.

2.3 Co-Training

Co-training was first introduced by Blum and Mitchell[20] as a semi-supervised learning method to utilize the unlabeled data. The intuition behind the co-training approach is to utilize classifiers from different views to enhance each other via pseudo labels. Several following studies aim to provide theoretical support and expand the practical applications[20, 29–32]. A recent line of work applies co-training to tasks such as image recognition, object detection and text classification[33–35]. Self-paced multi-view co-training[33, 34] extends the co-training to multi-view scenarios and formulates the co-training as a self-paced learning process. Deep co-training[35] uses adversarial examples to encourage the diversity between different views. Although being widely used as a semi-supervised learning technique, applying co-training to adversarial defense, especially with graph neural networks, remains unexplored.

3 Preliminary Study

In this section, we present why two-view co-training is a desirable defense technique for the graph data. As discussed in Subsection 2.2, output diversity plays a crucial role in the robustness of the ensemble model. Recent work[36] shows that adversarial examples are more likely to transfer between models that have a large adversarial subspace (Adv-SS) overlap. Sub-models with diverse outputs have a smaller Adv-SS overlap, making it harder for attackers to craft adversarial examples that fool all sub-models[16–18].

To minimize the Adv-SS overlap, using different aspects of the input data is a promising approach. Graph data naturally contains two complementary views: a node feature view and a graph structure view. The models trained by these two views share little adversarial subspace by definition. An attack on node features would barely affect a structure-based model and vice versa, which implies the potential for improving robustness via an ensemble of sub-models from these two views. Furthermore, from an attacker's perspective, conducting attacks on the feature view is more difficult due to the following reasons: 1) in most graph data, node features are sparse and high-dimensional, which makes the perturbations on node features detectable; 2) in most cases, node features are discrete and practically interpretable, which causes restrictions on feature modifications. If an attacker attempts to attack a social network via modifying a person's birthday, his/her age should also be changed to ensure the validity of the data. Such restrictions are difficult to model in a differentiable manner, which constitutes a challenge for gradient-based attacks.

Although existing GNN models are designed to integrate the feature information and the structure information of the graph data via message passing, they do not pay enough attention to the feature information. In an empirical study, Jin *et al.*[13] pointed out that GCNs prefer to preserve structure information rather than feature information during the message passing process. This behavior is consistent with the empirical conclusions in previous work[9]: 1) topological attacks are more effective than feature attacks when attacking models like GCNs; 2) attackers tend to connect nodes with dissimilar node features to achieve a successful attack. In this paper, such models whose predictions are more dependent on the graph structure are called structure-dominant models. Correspondingly, models that rely mainly on node features for predictions are called feature-dominant models.

Therefore, to fully exploit the feature information and achieve better robustness, we propose a two-view co-training framework, named calibrated co-training on graph (C^{2}oG), to learn a robust ensemble of the feature-dominant models and the structure-dominant models.

4 Proposed Approach

In this section, we first formulate the problem and elaborate on the overall co-training framework. To further explain how we implement the framework, we then describe the specific models, which can act as the structure-dominant and feature-dominant components. Last but not least, we introduce the model calibration and the class balancing mechanisms, which play critical roles in ensuring the effectiveness of the co-training framework.

4.1 Problem Formulation

In this paper, we focus on the semi-supervised node classification problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be a graph with n nodes, where $\mathcal{V}$ is the set of nodes $\{v_1, \cdots, v_n\}$ with $|\mathcal{V}| = n$, $\mathcal{E}$ is the set of edges, and $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n\times m}$ is a feature matrix. Then we can separate $\mathcal{G}$ into two views. In the node feature view, for each node $v \in \mathcal{V}$, its feature $\boldsymbol{x}_v \in \mathbb{R}^m$ is an m-dimensional row vector. In the graph structure view, the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ can be formulated by setting $A_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $A_{ij} = 0$, otherwise. In the semi-supervised node classification problem, nodes are separated into two sets $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, where nodes in $\mathcal{S}$ are labeled and nodes in $\mathcal{U}$ are not. Our goal is to learn a high-performance ensemble of a feature-dominant model f_{feat} and a structure-dominant model f_{struct}, using both the labeled and the unlabeled data from $\mathcal{V}$.

4.2 Overall Framework

Graph data can be separated into two views[23], i.e., the feature view and the structure view. These two views provide different information about the nodes in the graph. The feature view describes nodes' intrinsic properties, while the structure view tells the relationship between the nodes. The neural networks trained in either of these two views can classify the nodes effectively. In addition, the information provided by these two views is complementary so that the knowledge distilled from the model in one view can promote the performance of the model in the other view. Therefore, we apply the two-view co-training framework on graph data (C^{2}oG) and learn an ensemble of sub-models in these two views.

The overall framework of C^{2}oG is shown in Fig.2 and the corresponding algorithm is demonstrated in Algorithm 1. For each node v, we separate its information into a feature view and a structure view. After that, two classifiers are trained separately for each view. The feature-dominant model f_{feat} primarily uses node features as its input to classify the nodes, while the structure-dominant model f_{struct} usually uses the

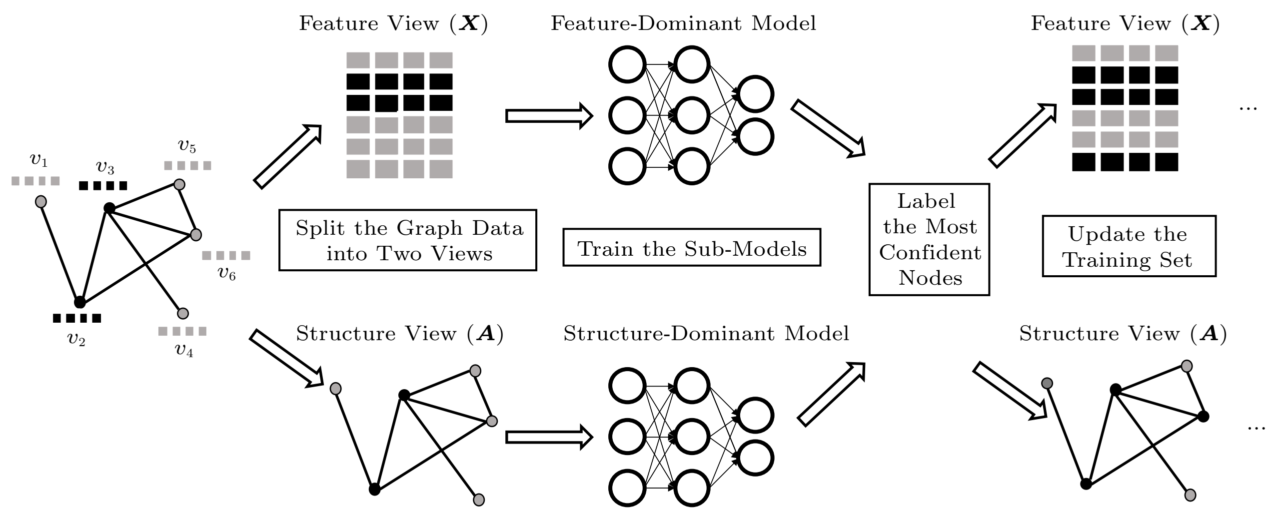


Fig.2. Overall framework of C^{2}oG. Graph data is separated into the feature view and the structure view. During the co-training process, the feature-dominant model and the structure-dominant model label their most confident nodes and add the selected nodes into the training set.

Algorithm 1. Two-View Co-Training for Graph Data

Input: graph data $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, nodes to add per iteration for class c N_c^{add}, labeled dataset $\mathcal{S}$, unlabel dataset $\mathcal{U}$, the number of classes C, structure-dominant model f_{struct}, feature-dominant model f_{feat}, the maximum number of iterations I

Output: f_{struct}, f_{feat}

1 **while** true **do**
2   Train f_{struct} under the structure view of $\mathcal{S}$;
3   Train f_{feat} under the feature view of $\mathcal{S}$;
4   Calculate confidence scores of the structure-dominant model s_{struct} on $\mathcal{U}$ using f_{struct};
5   Calculate confidence scores of the feature-dominant model s_{feat} on $\mathcal{U}$ using f_{feat};
6   **for** $c = 1$ to C **do**
7   Use f_{struct} to label N_c^{add} most confident nodes in $\mathcal{U}$;
8   Use f_{feat} to label N_c^{add} most confident nodes in $\mathcal{U}$;
9   **if** the same node is chosen **then**
10   Comparing s_{struct} and s_{feat} and using the pseudo-label with a higher confidence score;
11   **endif**
12   **endfor**
13   Update $\mathcal{S}$&$\mathcal{U}$;
14   **if** there is no test data left or achieving maximum iteration **then**
15   break
16   **endif**
17 **end**

---

graph structure as its clue for node labels. In each iteration, we first train f_{struct} and f_{feat} (lines 2 and 3 in Algorithm 1 respectively). After that, the confidence scores of each model on all unlabeled nodes s_{struct} and s_{feat} are calculated (lines 4 and 5 respectively), and the most confident unlabeled nodes from each view are added to the training set (lines 6–13). After the preset number of iterations is achieved or there is no test data left (lines 14–16), we obtain two well-trained models, f_{feat} and f_{struct}, and generate an ensemble of them by averaging their predictions. In the remaining part of this subsection, we introduce more details about the framework, including 1) the structure-dominant models and the feature-dominant models we use to learn representation for nodes; 2) the techniques we use to improve the performance of the co-training framework, including model calibration and class balancing.

4.3 Structure-Dominant Models

Structure-dominant models are the models which either only use the structure or do not fully utilize the node feature information of the graph by design.

4.3.1 Structure-Dominant GNNs

Graph convolutional networks (GCNs) [1] achieve remarkable success in learning representation for graph data. Given a graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, the updates of node embeddings can be derived by the following formulation:

$$\boldsymbol{X}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}\right),$$

where $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix. $\boldsymbol{X}^{(l)}$ is the layer-wise node embedding, $\boldsymbol{W}^{(l)}$ is the layer-wise parameter and $\sigma(\cdot)$ denotes the activation function. Although a GCN takes both features and graph structure as its input, the representation it learns for each node largely depends on the local graph structure of the node. The reasons are the follows. On the one hand, features are just simply aggregated from neighboring nodes. On the other hand, no expressive functions are used to model the features of certain classes.

We also design a more feature-independent GCN that uses a one-hot feature to replace the original node features, named GCN1h. Since no meaningful feature information is provided, the predictions of GCN1h is determined only by the graph structure.

Besides GCNs, it is worth noting that some other state-of-the-art GNNs can also be adopted as structure-dominant models since they share the same message passing mechanism as GCNs. In this paper, we consider two representative GNNs, APPNP [37] and GCNII [38]. APPNP enhances the structure information with the personalized page rank while GCNII uses initial residual and identity mapping to improve the model's performance.

4.3.2 *Structure-Based Multilayer Perceptron (S-MLP)*

The spectral method is another effective method to distill the structure information for each node [39, 40]. Given the graph topology $\boldsymbol{A}$, its eigenvalues and eigenvectors of Laplacian are computed by solving:

$$\boldsymbol{D}^{-1}\boldsymbol{L}\boldsymbol{y} = \lambda\boldsymbol{y},$$

where $\boldsymbol{D}^{-1}\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$ is the normalized laplacian matrix, $\boldsymbol{y}$ denotes the eigenvector, and λ denotes the eigenvalue. Let $\lambda_0, \lambda_1, \cdots, \lambda_k$ be the k smallest eigenvalues of $\boldsymbol{D}^{-1}\boldsymbol{L}$ and $\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_k$ be the respective eigenvectors. Each node u can be embedded into a k-dimensional space as $\boldsymbol{a}_u = (\boldsymbol{y}_{1u}, \boldsymbol{y}_{2u}, \cdots, \boldsymbol{y}_{ku})$.

To enrich the information of $\boldsymbol{a}_u$, we also compute the k-dimension Laplacian eigenmaps of $\boldsymbol{A}^2$, which reflects the two-hop neighboring information of each node. By concatenating the spectral embedding generated from $\boldsymbol{A}$ and $\boldsymbol{A}^2$ together, we get the enhanced $\tilde{\boldsymbol{a}}_u \in \mathbb{R}^{2\times k}$. After that, $\tilde{\boldsymbol{a}}_u$ is sent as the input to a multilayer perceptron model for classification.

4.4 Feature-Dominant Models

Feature-dominant models are the models which either only use the node feature information or do not fully utilize the graph structures by design.

4.4.1 *Feature-Based Multilayer Perceptron*

The multilayer perceptron (MLP) model is the simplest but effective method when we consider the node features alone. The layerwise forwarding process of MLP can be formulated as,

$$\boldsymbol{x}_v^{(l+1)} = \sigma(\boldsymbol{x}_v^l \Theta^{(l)} + \boldsymbol{b}_l),$$

where $\boldsymbol{x}_v \in \mathbb{R}^m$ is an m-dimensional row vector denoting the features of node v, $\Theta^{(l)}$ and $\boldsymbol{b}_l$ are the layerwise parameters, and $\sigma(\cdot)$ denotes the activation function. Given the fact that no structure information is provided, MLP is structure-independent.

4.4.2 *k-NN Based GCN (k-NN-GCN)*

The k-nearest-neighbor based (k-NN based) model first constructs a graph from feature matrix $\boldsymbol{X}$ by using a k-nearest-neighbor algorithm based on the cosine similarity. For each node pair (v_i, v_j), we calculate its feature similarity as:

$$\boldsymbol{s}_{ij} = \frac{\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j}{\|\boldsymbol{x}_i\|\,\|\boldsymbol{x}_j\|}.$$

After that, the k-NN graph $\boldsymbol{A}_k = k\,\mathrm{NN}(\boldsymbol{X})$ can be constructed by connecting the top-k similar node pairs. Finally, the whole graph data $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A}_k)$ is sent to a GCN model to classify the nodes. The structure information here is generated from the node features; therefore the original graph structure is not used in this model.

4.5 Model Calibration

Models' confidence is the most critical indicator during the co-training process. On the one hand, the co-training framework picks up the most confident predictions in each sub-model. When sub-models attempt to add the same node to the training set, the co-training framework decides which pseudo label to use based on sub-models' confidence. On the other hand, in the inference stage, $\mathrm{C^2oG}$ averages each sub-model's confidence to obtain the predictions of their ensembles.

Generally, the softmax output of each sub-model is used to measure the confidence. However, modern neural networks, including GNN, can be miscalibrated [22, 41]. Since the sub-models are heterogeneous in $\mathrm{C^2oG}$, this miscalibration can impair the performance of the co-training method. To alleviate this problem, we use the temperature scaling method [22] to calibrate the output of each sub-model. Given the logits $\boldsymbol{z}$, the calibrated prediction is obtained as:

$$\boldsymbol{q} = \mathrm{softmax}(\boldsymbol{z}/T),$$

where the temperature T is learned by optimization with respect to the negative log likelihood on the validation set.

4.6 Class Balancing

In C^{2}oG, the most confident unlabelled nodes from each view are added to the training set during the co-training process. The confidence of each node is measured by the softmax outputs of the sub-models. However, this strategy can lead to class imbalance if sub-models perform better in one particular class. Furthermore, an imbalanced class distribution will cause the overfitting problem as the co-training process moves on and finally impair the sub-models' performance.

To keep a balanced training dataset, the added data should follow the class distribution of the initial training dataset. Formally, supposing we have N inputs with an initial distribution $(N_1, N_2, \cdots, N_C)$, the number of added proposals for each class c is:

$$N_c^{\text{add}} = \frac{N_c}{N} \times N^{\text{add}},$$

where N^{add} denotes the number of nodes we add to the training data in each iteration.

The co-training process will stop when reaching the preset number of iterations or there is no more data in the test set. In the inference phase, we average sub-models' output to get the predictions of their ensembles. Compared with other defense methods, the advantages of C^{2}oG can be summarized as follows.

• Instead of relying on prior knowledge to purify the perturbed graph, C^{2}oG enhances the robustness of GNNs via knowledge distillation. Compared with the human-designed prior knowledge, the distilled knowledge is more adaptable to different types of the graph data. As a result, C^{2}oG could be applied to more scenarios.

• Compared with existing ensemble approaches, where the knowledge of the graph only flows from GNN to MLP in a one-direction manner or model diversity is not considered [24], C^{2}oG enables the bi-directional knowledge distillation between GNN and MLP, and takes the complementarity of different types of models into consideration. This enhances the robustness of C^{2}oG when the graph structure is heavily perturbed. Furthermore, the distillation process of C^{2}oG is dynamic and non-differentiable, making it more difficult for attackers to conduct adaptive attacks.

5 Experimental Results

In this section, we evaluate the performance of C^{2}oG on the clean data and its robustness against adversarial attacks. In particular, we focus on answering the following questions.

*Q*1. How does C^{2}oG perform on clean data?

*Q*2. How does C^{2}oG perform under adversarial attacks compared with other state-of-the-art defense methods?

*Q*3. How do model calibration, the class balancing technique and the hyper-parameters affect C^{2}oG's performance?

*Q*4. How does C^{2}oG perform against adaptive attacks?

5.1 Experimental Setup

5.1.1 Datasets

To obtain comparable results, we use three popular citation graphs, i.e., Cora [19], Citeseer [42] and Pubmed [42] to evaluate our model. The basic information of these three graphs is shown in Table 1. In these three datasets, each node represents a document and edges are the citations between documents. In terms of data splits, we randomly pick 10% of nodes for training, 10% for validation and 80% for testing, following [6, 8, 13].

5.1.2 Baselines

We compare our model with the following baselines.

• *GCN-SVD* [14]. GCN-SVD is a defense method based on low-rank approximation.

• *GCN-Jaccard* [9]. GCN-Jaccard is also a preprocessing-based defense, which removes the edges between most dissimilar nodes to purify the graph.

• *SimP-GCN* [23]. SimP-GCN adaptively combines the original graph and the k-NN graph to capture the similarity between nodes.

Table 1. Statistics of the Datasets

Dataset	Number of Nodes	Number of Edges	Number of Classes	Number of Features
Cora [19]	2 485	5 069	7	1 433
Citeseer [42]	2 110	3 668	6	3 703
Pubmed [42]	19 717	44 338	3	500

• *Pro-GNN*[13]. Pro-GNN is a defense method that exploits three properties of real-world graphs: low-rank, sparsity, and feature smoothness.

• *UM-GNN*[24]. UM-GNN trains a feature-based model based on knowledge transferred from GNN models.

• *GNNGuard*[10]. GNNGuard reweighs each edge based on the similarity of connected node pairs.

• *Soft Median*[43]. Soft Median uses the distance between nodes and the median of the neighbours to reweigh each edge.

5.1.3 Parameter Settings

We implement C^{2}oG under the framework of DeepRobust[44], which is a well-known adversarial learning framework for GNNs. Similar to previous work[11,13], we run each experiment 10 times and report the average performance. For GCN, we use the settings of the original GCN[1], i.e., a two-layer structure with 16 hidden units. We use a two-layer structure with 32 hidden units for the MLP model. The number of nearest neighbors k we set in k-NN-GCN is 50. As for S-MLP, we use the eigenvectors corresponding to the lowest 50 eigenvalues to distill the structure information. In the co-training process, each model adds 250 pieces of most confident unlabeled data with their pseudo-labels in one iteration. For the baseline models, we take the same experimental settings as in [13]. We set the learning rate for all sub-models to 0.01, the weight decay to 5×10^{-4}, and the dropout rate to 0.5, and train for 200 epochs.

As for adversarial attacks, we use Metattack[8] which is an effective poisoning attack method on graphs. It treats the graph as a hyperparameter and modifies the graph to increase learning loss via meta-gradient. The edge perturbation rate is set as $\{5\%, 10\%, 15\%, 20\%\}$. We use the same random seed as [13] to make fair comparisons with their reported results.

5.2 Node Classification Accuracy on Clean Graphs

The performance of the ensemble models on clean graphs is shown in the 3rd column of Table 2. From the results, we have the following observations.

• Our ensemble models outperform the baselines on clean graphs on the three datasets. Specifically, the classification accuracy of the GCN+F-MLP model is 0.59%, 3.18%, and 0.43% higher than the vanilla GCN on the three datasets respectively. As a comparison, in most cases, the classification accuracy values of the best-performed baselines are just marginally better than that of the GCN model.

• Our ensemble models perform better than any single sub-model from the ensemble. For instance, the ensemble of GCN and k-NN-GCN (GCN+k-NN-GCN) achieves an accuracy of 84.27% on Cora, while the accuracy is 83.50% for GCN and 71.06% for k-NN-GCN.

• It is worth noting that although GCN1h has relatively weak performance on clean data, it achieves comparable performance after being co-trained with MLP, indicating that C^{2}oG enables GCN1h to effectively distill the knowledge from the feature view.

Results show that our method achieves competitive performance on clean data, thus making it applicable in realistic settings where we have no idea if the graphs are perturbed.

5.3 Node Classification Accuracy Under Attacks

Results under attacks are shown in the 4th–7th columns in Table 2 and Table 3. As we can see, the co-training framework effectively enhances the model's robustness against adversarial attacks. For example, the accuracy of the GCN model decreases drastically from 83.5% to 59.56% as the perturbation rate increases from 0% to 20% on Cora. In comparison, GCN+k-NN-GCN still achieves the accuracy of 76.86% even in the worst-perturbed case. Similar results are obtained on the other two datasets.

Our method outperforms the other state-of-the-art defenses, especially on the Cora and the Citeseer dataset. Pro-GNN[13] is the most robust model among the baselines. Therefore, we mainly compare our results with those of Pro-GNN. Our ensemble models outperform Pro-GNN by a large margin on Cora and Citeseer. Results in Table 2 show that our method outperforms ProGNN when adopting GCN as the structure-dominant model, while the results in Table 3 demonstrate the performance improvement when using F-MLP as the feature-dominant model. In addition, other instantiations of our C^{2}oG framework can also achieve good performance. Specifically, the combination of S-MLP and k-NN-GCN gains 0.61%, 2.01%, 4.49%, and 7.43% improvements as the perturbation rate increases from 5% to 20% on Cora respectively. Correspondingly, the improvements on Citeseer are 1.86%, 2.65%, 2.02% and 2.78%. On Pubmed, we also achieve

Table 2. Node Classification Accuracy (%) on Clean Graphs and Perturbed Graphs

Dataset	Model	Perturbation Rate (%)				
		0	5	10	15	20
Cora	GCN	83.50 ± 0.44	76.55 ± 0.79	70.39 ± 1.28	65.10 ± 0.71	59.56 ± 2.72
	GCN-SVD	80.63 ± 0.45	78.39 ± 0.54	71.47 ± 0.83	66.69 ± 1.18	58.94 ± 1.13
	GCN-Jaccard	82.05 ± 0.51	79.13 ± 0.59	75.16 ± 0.76	71.03 ± 0.64	65.71 ± 0.89
	SimP-GCN	81.81 ± 0.62	76.43 ± 1.98	73.27 ± 1.93	70.75 ± 3.98	66.63 ± 6.87
	Pro-GNN	82.98 ± 0.23	82.27 ± 0.45	79.03 ± 0.59	76.40 ± 1.27	73.32 ± 1.56
	GNNGuard	77.33 ± 1.01	75.78 ± 1.23	72.59 ± 1.46	72.56 ± 1.53	72.22 ± 0.99
	Soft Median	84.02 ± 0.50	79.88 ± 0.75	73.41 ± 2.34	70.50 ± 1.13	60.50 ± 0.36
	GCN+F-MLP (ours)	84.09 ± 0.59	**83.48 ± 0.43**	**82.88 ± 0.83**	**81.01 ± 0.57**	76.70 ± 0.63
	GCN+k-NN-GCN (ours)	**84.27 ± 0.31**	83.16 ± 0.28	82.54 ± 0.29	80.57 ± 0.40	**76.86 ± 0.75**
	GCN1h	69.68 ± 0.55	65.31 ± 0.50	59.02 ± 0.37	52.60 ± 0.72	45.66 ± 0.34
	GCN1h+F-MLP (ours)	83.05 ± 0.71	**81.78 ± 0.47**	**81.50 ± 0.35**	**79.91 ± 0.42**	**78.92 ± 0.54**
	GCN1h+k-NN-GCN (ours)	**83.13 ± 0.38**	80.89 ± 0.36	80.06 ± 0.56	79.38 ± 0.44	78.63 ± 0.84
Citeseer	GCN	71.96 ± 0.55	70.88 ± 0.62	67.55 ± 0.89	64.52 ± 1.11	62.03 ± 3.49
	GCN-SVD	70.65 ± 0.32	68.84 ± 0.72	68.87 ± 0.63	63.26 ± 0.96	58.55 ± 1.09
	GCN-Jaccard	72.10 ± 0.63	70.51 ± 0.97	69.54 ± 0.56	65.95 ± 0.94	59.30 ± 1.40
	SimP-GCN	73.76 ± 0.78	73.12 ± 0.85	72.38 ± 0.67	71.75 ± 1.54	69.37 ± 1.50
	Pro-GNN	73.28 ± 0.69	72.93 ± 0.57	72.51 ± 0.75	72.03 ± 1.11	70.02 ± 2.28
	GNNGuard	68.73 ± 1.75	69.15 ± 1.25	69.95 ± 1.00	65.86 ± 1.16	68.21 ± 1.47
	Soft Median	71.33 ± 0.75	69.57 ± 2.22	67.89 ± 1.91	66.03 ± 2.94	56.08 ± 1.34
	GCN+F-MLP (ours)	**75.14 ± 0.54**	**74.83 ± 0.58**	73.70 ± 0.60	73.68 ± 0.83	**71.91 ± 0.93**
	GCN+k-NN-GCN (ours)	74.80 ± 0.58	74.76 ± 0.56	**73.79 ± 0.48**	**73.86 ± 0.79**	71.74 ± 1.34
	GCN1h	69.31 ± 0.34	68.90 ± 0.40	62.41 ± 0.65	62.11 ± 0.38	54.99 ± 0.31
	GCN1h+F-MLP (ours)	74.31 ± 0.47	74.12 ± 0.38	**74.82 ± 0.22**	74.60 ± 0.58	69.30 ± 0.69
	GCN1h+k-NN-GCN (ours)	**74.94 ± 0.41**	**75.04 ± 0.23**	74.53 ± 0.24	**75.01 ± 0.22**	**72.80 ± 0.74**
Pubmed	GCN	87.19 ± 0.09	83.09 ± 0.13	81.21 ± 0.09	78.66 ± 0.12	77.35 ± 0.19
	GCN-SVD	83.44 ± 0.21	83.41 ± 0.15	83.27 ± 0.21	83.10 ± 0.18	83.01 ± 0.22
	GCN-Jaccard	87.06 ± 0.06	86.39 ± 0.06	85.70 ± 0.07	84.76 ± 0.08	83.88 ± 0.05
	SimP-GCN	87.59 ± 0.10	86.79 ± 0.12	86.01 ± 0.10	85.49 ± 0.11	85.37 ± 0.12
	Pro-GNN	87.26 ± 0.23	87.23 ± 0.13	87.21 ± 0.13	**87.20 ± 0.15**	**87.15 ± 0.15**
	GNNGuard	85.25 ± 0.14	85.13 ± 0.15	84.65 ± 0.25	84.51 ± 0.17	84.12 ± 0.24
	Soft Median	**87.70 ± 0.07**	86.34 ± 0.05	85.50 ± 0.08	84.43 ± 0.05	83.67 ± 0.03
	GCN+F-MLP (ours)	87.62 ± 0.05	**87.25 ± 0.09**	**87.20 ± 0.09**	87.05 ± 0.08	87.04 ± 0.04
	GCN+k-NN-CCN (ours)	84.92 ± 0.14	83.74 ± 0.15	82.97 ± 0.15	81.79 ± 0.16	81.35 ± 0.08
	GCN1h	74.58 ± 0.81	70.70 ± 0.33	67.11 ± 0.35	63.82 ± 0.33	61.54 ± 0.46
	GCN1h+F-MLP (ours)	**87.19 ± 0.04**	**86.95 ± 0.11**	**86.85 ± 0.09**	**86.44 ± 0.15**	**86.42 ± 0.10**
	GCN1h+k-NN-GCN (ours)	85.94 ± 0.36	85.32 ± 0.31	84.78 ± 0.79	82.38 ± 0.65	81.88 ± 0.73

Note: Since all the compared baselines are GCN-based, for fair comparison, in this table we demonstrate the performance of C^{2}oG with GCNs (GCN+k-NN-CCN (ours)) as the structure-dominant model. The results of C^{2}oG with other models are displayed in Table 3. GCN1h denotes the model in which we replace the nodes' feature matrix with an identity matrix. Bold fonts highlight the highest accuracy.

comparable results. Moreover, it is worth noting that Pro-GNN trains slowly (over 100x longer than GCN), and requires lots of the GPU memory (10x larger than GCN). Compared with Pro-GNN, our approach is much faster (about 25x faster than Pro-GNN) and less memory-demanding (10x less than Pro-GNN), making it more feasible in practical use. We also compare C^{2}oG with SimP-GCN [23], which also focuses on exploiting the feature information. Results show that C^{2}oG achieves better performance on all three graphs under different perturbation rates. Furthermore, the performance of SimP-GCN possesses high variance, especially on Cora. One possible reason is that SimP-GCN has an unstable graph structure in its training stage.

UM-GNN [24] uses different data splits from the above methods. It follows the original split in [1] and uses the perturbation rate from 0% to 10%. Results in Fig.3 show the accuracy improvement of C^{2}oG and UM-GNN over GCN. It shows that although UM-GNN performs slightly better than C^{2}oG when the perturbation rate is small, it is outperformed by a large margin as the perturbation rate increases. This is caused by the one-directional knowledge transferring scheme in UM-GNN. When the predictions of GNN is highly inaccurate, it will mislead the MLP model. This defect is avoided in C^{2}oG since C^{2}oG can transfer useful information in both directions. The performance of both

Table 3. Node Classification Accuracy (%) on Clean Graphs and Perturbed Graphs with More Structure-Dominant Models

Dataset	Model	Perturbation Rate (%)				
		0	5	10	15	20
Cora	S-MLP	78.30 ± 0.17	76.25 ± 0.23	72.95 ± 0.36	67.31 ± 0.40	54.56 ± 0.25
	S-MLP+F-MLP (ours)	**83.59 ± 0.30**	**84.25 ± 0.33**	**83.52 ± 0.31**	**82.95 ± 0.59**	**80.75 ± 0.59**
	APPNP	**86.00 ± 0.34**	81.44 ± 0.66	76.55 ± 0.36	72.87 ± 0.83	61.43 ± 0.90
	APPNP+F-MLP (ours)	85.61 ± 0.31	**83.26 ± 0.30**	**80.39 ± 0.31**	**77.65 ± 0.35**	**70.10 ± 0.47**
	GCNII	**85.59 ± 0.32**	80.95 ± 0.56	75.36 ± 1.67	70.76 ± 0.93	58.77 ± 1.11
	GCNII+F-MLP (ours)	85.27 ± 0.27	**82.37 ± 0.40**	**79.08 ± 0.37**	**76.82 ± 0.34**	**68.46 ± 0.28**
Citeseer	S-MLP	69.31 ± 0.34	68.90 ± 0.40	62.41 ± 0.65	62.11 ± 0.38	54.99 ± 0.31
	S-MLP+F-MLP (ours)	**74.31 ± 0.47**	**74.12 ± 0.38**	**74.82 ± 0.22**	**74.60 ± 0.58**	**69.30 ± 0.69**
	APPNP	73.36 ± 0.45	72.71 ± 0.54	72.04 ± 0.42	69.44 ± 0.43	60.15 ± 0.73
	APPNP+F-MLP (ours)	**74.93 ± 0.16**	**74.62 ± 0.71**	**73.64 ± 0.65**	**72.57 ± 0.31**	**68.26 ± 0.60**
	GCNII	73.76 ± 0.38	73.85 ± 0.24	71.64 ± 0.65	70.50 ± 0.37	61.56 ± 1.20
	GCNII+F-MLP (ours)	**75.05 ± 0.19**	**74.92 ± 0.44**	**73.84 ± 0.30**	**72.78 ± 0.43**	**68.61 ± 0.45**
Pubmed	S-MLP	77.23 ± 0.17	73.29 ± 0.19	70.36 ± 0.42	66.99 ± 0.75	64.68 ± 0.66
	S-MLP+F-MLP (ours)	**86.57 ± 0.11**	**86.58 ± 0.10**	**86.51 ± 0.10**	**86.31 ± 0.11**	**86.26 ± 0.08**
	APPNP	85.89 ± 0.11	83.31 ± 0.12	81.31 ± 0.12	78.88 ± 0.20	76.55 ± 0.28
	APPNP+F-MLP (ours)	**87.10 ± 0.12**	**86.54 ± 0.13**	**86.48 ± 0.37**	**86.30 ± 0.12**	**85.86 ± 0.18**
	GCNII	85.67 ± 0.18	83.53 ± 0.45	82.41 ± 0.19	80.76 ± 0.96	80.59 ± 0.28
	GCNII+F-MLP (ours)	**86.50 ± 0.49**	**86.44 ± 0.03**	**86.45 ± 0.27**	**86.46 ± 0.94**	**86.34 ± 0.38**

Note: Bold fonts highlight the higher accuracy.

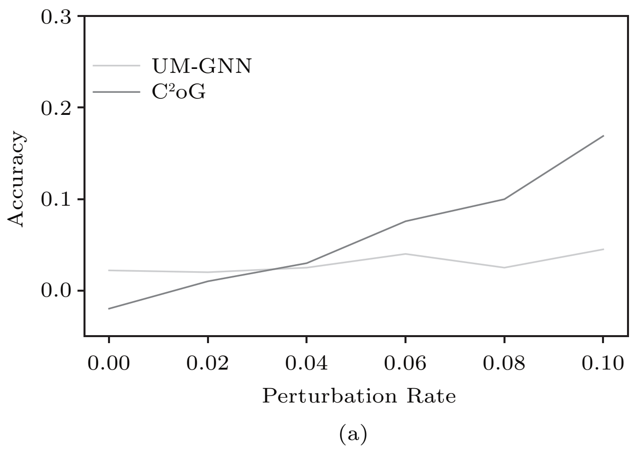


Fig.3. Accuracy improvement over GCN. (a) Results on Cora. (b) Results on Citeseer.

GNN and MLP improves during the co-training process.

In Table 3, results show that state-of-the-art GNNs like APPNP and GCNII are also vulnerable to edge perturbations, while C^{2}oG could consistently improve the accuracy of the structure-dominant models via co-training with an F-MLP model. Specifically, on Cora, C^{2}oG brings up to 8.67% and 9.69% performance gains for APPNP and GCNII, respectively. The performance improvements over APPNP and GCNII are up to 8.11% and 7.05% on Citeseer respectively and the improvements on Pubmed are 9.31% and 5.75% respectively. This demonstrates the applicability of C^{2}oG on GNNs beyond classic GCNs.

5.4 Complexity Analysis and Node Classification Accuracy on Larger Graphs

The computational complexity of C^{2}oG is bounded by the complexity of its sub-models. Specifically, assuming that the number of iterations is k and the worst computational complexity among sub-models is $O(T_c)$, the computational complexity of the co-training process is $O(kT_c)$. There exist trade-offs between the accuracy and the selection of k, which will be elaborated in Subsection 5.5.2. Moreover, even though C^{2}oG requires to train the sub-models for k times, its sub-models are simple and quick convergent. In contrast, GNNGuard requires to compute the attention of each connected node pair at each layer and Soft Median needs to compute the distance for nodes to their medians. Accordingly, the following empirical results show that C^{2}oG has a similar efficiency to Soft Median and beats GNNGuard. In terms of memory complexity, since co-training does not require storing intermediate results or additional parameters, the memory complexity of C^{2}oG is bounded by the maximum memory complexity of the sub-models $\mathcal{O}(T_m)$.

To further verify the effectiveness of C^{2}oG, we evaluate it on five larger graphs [45]. As shown in Table 4, the five new datasets have more nodes and edges than the three datasets we show in Table 1. As baselines, we select the state-of-the-art defense approaches including GNNGuard [10] and Soft Median [43]. Since Metattack fails to attack graphs at such a scale, we adopt the PR-BCD attack [43] with the CW loss as the attack method. We keep the number of co-training iterations as 10 for simplicity. After 10 iterations, 80% of the unlabeled nodes are added into the training set. For each case, we repeat the test for five runs. As illustrated in Table 5, C^{2}oG outperforms both GNNGuard and Soft Median in most cases, especially under higher perturbation rates. In terms of efficiency, C^{2}oG is comparable with Soft Median and is around 10 times faster than GNNGuard, which validates that the proposed co-training

Table 4. Statistics of the Larger Datasets

Dataset	Number of Nodes	Number of Edges	Number of Classes	Number of Features
Cora-Full	19 793	63 421	70	8 710
Coauthor-CS	34 493	247 962	5	8 415
Coauthor-Physics	18 333	81 894	15	6 805
Amazon-computers	13 752	245 861	10	767
Amazon-photo	7 650	119 081	8	745

Table 5. Node Classification Accuracy (%) on Clean Graphs and Perturbed Graphs at a Larger Scale

Dataset	Model	Perturbation Rate (%)				
		0	5	10	15	20
Cora-Full	GCN	64.47 ± 0.19	57.91 ± 0.34	55.87 ± 0.38	56.69 ± 0.23	55.07 ± 0.25
	GNNGuard①	59.14 ± 0.17	58.34 ± 0.57	57.99 ± 0.25	57.74 ± 0.60	57.89 ± 0.18
	Soft Median②	61.71 ± 0.24	55.90 ± 1.42	54.92 ± 0.08	54.25 ± 0.32	52.66 ± 0.06
	C^{2}oG	**66.37 ± 0.20**	**64.60 ± 0.25**	**64.33 ± 0.13**	**64.03 ± 0.20**	**63.62 ± 0.21**
Coauhour-CS	GCN	92.46 ± 0.06	85.82 ± 0.24	83.39 ± 0.21	82.78 ± 0.55	80.26 ± 0.15
	GNNGuard	92.36 ± 0.09	91.93 ± 0.19	91.67 ± 0.05	91.58 ± 0.09	91.95 ± 0.10
	Soft Median	92.99 ± 0.03	91.31 ± 0.08	90.36 ± 0.10	89.66 ± 0.09	87.87 ± 0.17
	C^{2}oG	**94.40 ± 0.14**	**93.91 ± 0.09**	**93.81 ± 0.13**	**93.60 ± 0.10**	**93.50 ± 0.14**
Coauhour-Physics	GCN	95.63 ± 0.05	89.69 ± 0.03	87.16 ± 0.27	85.14 ± 0.23	86.77 ± 0.11
	GNNGuard	95.78 ± 0.08	95.43 ± 0.10	95.06 ± 0.07	**94.98 ± 0.06**	**95.10 ± 0.10**
	Soft Median	95.83 ± 0.14	93.67 ± 0.17	88.84 ± 0.45	85.76 ± 0.26	87.06 ± 0.12
	C^{2}oG	**96.18 ± 0.04**	**95.71 ± 0.03**	**95.19 ± 0.06**	94.87 ± 0.09	**95.10 ± 0.04**
Amazon-computers	GCN	**88.77 ± 0.51**	81.20 ± 0.68	72.54 ± 1.47	71.46 ± 1.01	73.31 ± 1.52
	GNNGuard	87.85 ± 0.59	80.60 ± 1.26	69.69 ± 2.61	64.78 ± 2.85	72.13 ± 1.98
	Soft Median	88.66 ± 0.57	82.54 ± 0.25	76.53 ± 0.56	73.02 ± 0.37	**76.86 ± 0.49**
	C^{2}oG	87.72 ± 0.24	**85.80 ± 0.83**	**83.70 ± 1.76**	**82.82 ± 2.05**	**77.77 ± 5.76**
Amazon-photo	GCN	93.27 ± 0.15	86.22 ± 0.34	79.22 ± 1.34	78.31 ± 2.15	72.98 ± 1.72
	GNNGuard	93.19 ± 0.25	86.40 ± 1.01	79.92 ± 1.80	75.27 ± 2.83	70.17 ± 1.50
	Soft Median	92.13 ± 0.75	87.74 ± 0.56	86.40 ± 0.30	83.00 ± 0.47	77.67 ± 0.46
	C^{2}oG	**94.57 ± 0.50**	**92.00 ± 1.09**	**90.72 ± 2.38**	**89.92 ± 0.31**	**89.37 ± 1.69**

①https://github.com/mims-harvard/GNNGuard, Sept. 2022.

②https://github.com/sigeisler/robustness_of_gnns_at_scale, Sept. 2022.

approach is quickly convergent.

5.5 Ablation Study

In this subsection, we conduct an ablation study on the model calibration and label balancing techniques. We also evaluate how the number of iterations and the number of nodes to be added in each iteration affect the performance.

5.5.1 Model Calibration and Class Balancing

Fig. 4 shows the reliability diagram of the GCN with/without model calibration. The reliability diagram demonstrates that the vanilla GCN is underconfident. After temperature scaling, the model's outputs can reflect the correctness likelihood more accurately. To validate the benefit of the class balancing technique, we train an ensemble of GCN+MLP with/without class balancing on Cora and present the results in Fig. 5. We report the confusion matrix of the ensemble model in each iteration. Results show that without class balancing, the co-training process will overfit to the dominant class. As the co-training process continues, more and more test data are labeled as the dominant class, thus impairing the performance of the ensemble model.

5.5.2 Number of Iterations and Added Nodes

There are two hyper-parameters to set in C^{2}oG, which are the number of iterations and the number of nodes to add in each iteration. In our experiments, we add 100, 250 and 500 nodes in each iteration and evaluate C^{2}oG's performance from zero iteration (the ensemble without co-training) to maximum iterations (until no test data left). Results are shown in Fig.6. As the co-training process continues, the per-

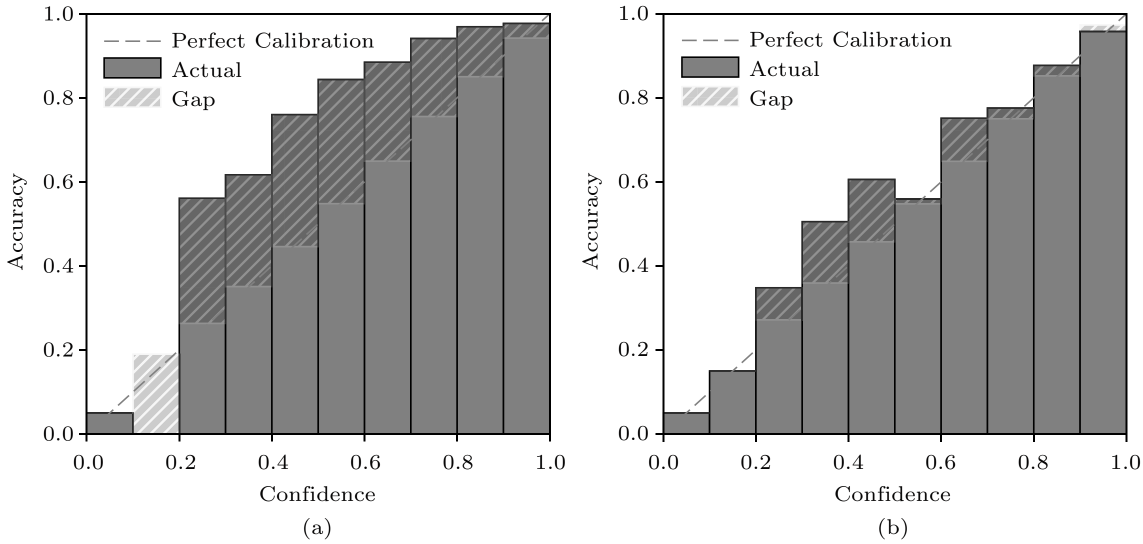


Fig.4. Reliability diagram of the GCN. (a) Without model calibration. (b) With model calibration.

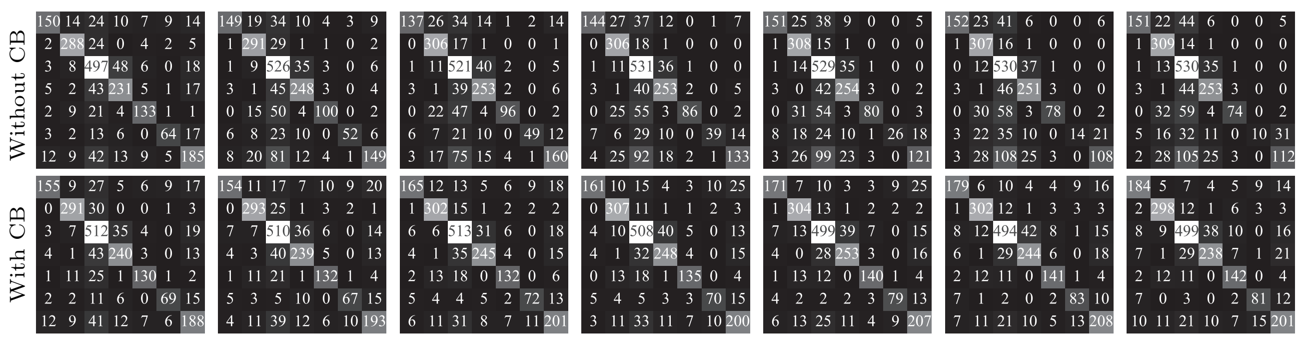


Fig.5. Confusion matrix ($\boldsymbol{M}$) during the co-training process (GCN+MLP on Cora). M_{ij} denotes how many instances with an actual class i is predicted as class j. Without class balancing (CB), the co-training process will overfit to the dominant class (class 2 in this case), thus leading to the decrease of the accuracy as the co-training continues.

formance of C^{2}oG is improved quickly in the beginning and stabilizes in the later iterations. Adding less nodes per iteration can slightly improve the performance of C^{2}oG. However, it also takes longer to train C^{2}oG, which indicates a trade-off between effectiveness and efficiency. Furthermore, compared with the ensemble of sub-models without co-training (the starting points), C^{2}oG can improve the performance by a large margin, especially on the perturbed data.

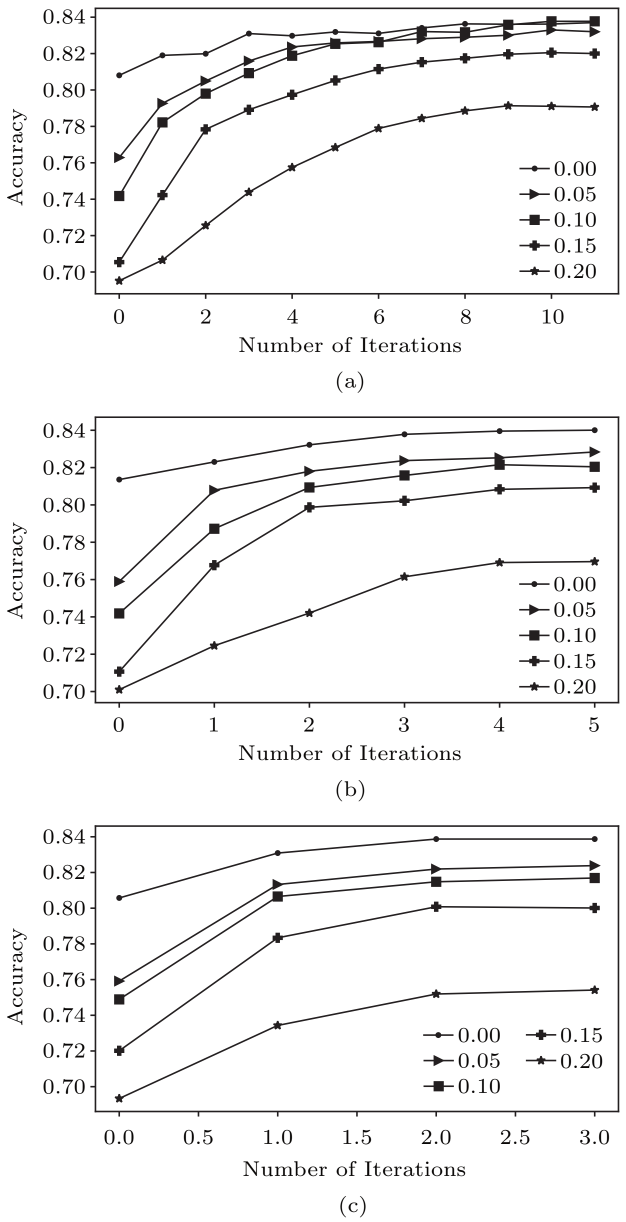


Fig. 6. Node classification accuracy using different hyper-parameters. Different shapes denote the results under different perturbation rates. (a) 100 nodes are added per iteration. (b) 250 nodes are added per iteration. (c) 500 nodes are added per iteration.

5.6 Adaptive Attacks

Evaluating C^{2}oG against adaptive attacks is necessary since attackers may attack C^{2}oG by conducting both structure and feature perturbations in practice. However, adaptive attacks on C^{2}oG are non-trivial since the co-training process works in a non-differential manner. Therefore, the attacker cannot find the optimal ratio to split the perturbation budget among different views via automatic optimizations. To evaluate C^{2}oG against adaptive attackers, we assume that the attacker will explore the effectiveness of different budget splits to find the optimal one. Specifically, we adapt the Metattack[8] to attack the MLP model, keep the total perturbation budget as 20% of the total number of edges, and evaluate C^{2}oG's performance under different ratios between feature and structure perturbations. Results are shown in Fig.7. We observe that the classification accuracy of C^{2}oG is higher than that of both the GCN model and the MLP model regardless of different budget splits, which validates the effectiveness of C^{2}oG under adaptive attacks.

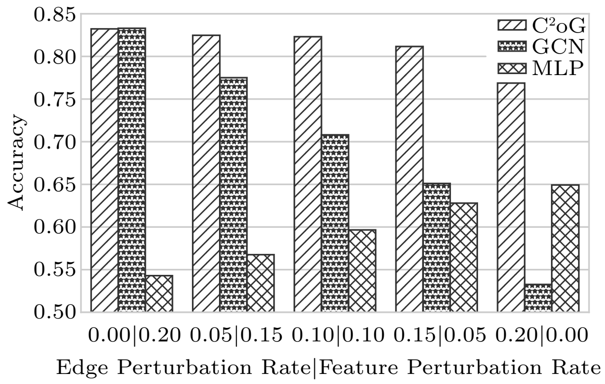


Fig. 7. Node classification accuracy of GCN, MLP and C^{2}oG under different ratios of feature and structure perturbations on Cora.

6 Conclusions

In this paper, we presented a calibrated co-training framework, named C^{2}oG, to learn a robust model that integrates the feature information and the structure information of the graph data. C^{2}oG is simple-to-implement but effective in improving the robustness for various models against adversarial attacks. The complementarity of the feature view and the structure view of the graph data diversifies the outputs of sub-models and weakens the transferability of adversarial attacks between sub-models. Evaluation results validated the effectiveness of C^{2}oG on both clean and perturbed graphs. C^2oG is generic and can be applied to various types of models beyond classic GCNs. In addition, C^2oG can still achieve good robustness under the adaptive attack setting where the defense internals are known to the attackers.

References

[1] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In *Proc. the 5th International Conference on Learning Representations*, April 2017.

[2] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In *Proc. the 6th International Conference on Learning Representations*, April 30-May 3, 2018.

[3] Hamilton W L, Ying Z, Leskovec J. Inductive representation learning on large graphs. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2017, pp.1024-1034.

[4] Fey M, Lenssen J E. Fast graph representation learning with PyTorch geometric. arXiv:1903.02428, 2019. https://arxiv.org/abs/1903.02428v3, Oct. 2021.

[5] Dai H, Li H, Tian T, Huang X, Wang L, Zhu J, Song L. Adversarial attack on graph structured data. In *Proc. the 35th International Conference on Machine Learning*, July 2018, pp.1115-1124.

[6] Zügner D, Akbarnejad A, Günnemann S. Adversarial attacks on neural networks for graph data. In *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, August 2018, pp.2847-2856. DOI: 10.1145/3219819.3220078.

[7] Bojchevski A, Günnemann S. Adversarial attacks on node embeddings via graph poisoning. In *Proc. the 36th International Conference on Machine Learning*, June 2019, pp.695-704.

[8] Zügner D, Günnemann S. Adversarial attacks on graph neural networks via meta learning. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[9] Wu H, Wang C, Tyshetskiy Y, Docherty A, Lu K, Zhu L. Adversarial examples for graph data: Deep insights into attack and defense. In *Proc. the 28th International Joint Conference on Artificial Intelligence*, August 2019, pp.4816-4823. DOI: 10.24963/ijcai.2019/669.

[10] Zhang X, Zitnik M. GNNGuard: Defending graph neural networks against adversarial attacks. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2020, pp.9263-9275.

[11] Chen L, Li X, Wu D. Enhancing robustness of graph convolutional networks via dropping graph connections. In *Proc. the European Conference Machine Learning and Knowledge Discovery in Databases*, Sept. 2020, pp.412-428. DOI: 10.1007/978-3-030-67664-3_25.

[12] Luo D, Cheng W, Yu W, Zong B, Ni J, Chen H, Zhang X. Learning to drop: Robust graph neural network via topological denoising. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, March 2021, pp.779-787. DOI: 10.1145/3437963.3441734.

[13] Jin W, Ma Y, Liu X, Tang X, Wang S, Tang J. Graph structure learning for robust graph neural networks. In *Proc. the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, August 2020, pp.66-74. DOI: 10.1145/3394486.3403049.

[14] Entezari N, Al-Sayouri S A, Darvishzadeh A, Papalexakis E E. All you need is low (rank): Defending against adversarial attacks on graphs. In *Proc. the 13th ACM International Conference on Web Search and Data Mining*, Feb. 2020, pp.169-177. DOI: 10.1145/3336191.3371789.

[15] Chang H, Rong Y, Xu T, Bian Y, Zhou S, Wang X, Huang J, Zhu W. Not all low-pass filters are robust in graph convolutional networks. In *Proc. the Annual Conference on Neural Information Processing*, Dec. 2021, pp.25058-25071.

[16] Pang T, Xu K, Du C, Chen N, Zhu J. Improving adversarial robustness via promoting ensemble diversity. In *Proc. the 36th International Conference on Machine Learning*, June 2019, pp.4970-4979.

[17] Kariyappa S, Qureshi M K. Improving adversarial robustness of ensembles with diversity training. arXiv:1901.09981, 2019. http://arxiv.org/abs/1901.09981, Oct. 2021.

[18] Yang H, Zhang J, Dong H, Inkawhich N, Gardner A, Touchet A, Wilkes W, Berry H, Li H. DVERGE: Diversifying vulnerabilities for enhanced robust generation of ensembles. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2020, pp.5505-5515.

[19] McCallum A K, Nigam K, Rennie J, Seymore K. Automating the construction of Internet portals with machine learning. *Information Retrieval*, 2000, 3(2): 127-163. DOI: 10.1023/A:1009953814988.

[20] Blum A, Mitchell T. Combining labeled and unlabeled data with co-training. In *Proc. the 11th Annual Conference on Computational Learning Theory*, July 1998, pp.92-100. DOI: 10.1145/279943.279962.

[21] Wang W, Zhou Z H. Analyzing co-training style algorithms. In *Proc. the 18th European Conference on Machine Learning*, Sept. 2007, pp.454-465. DOI: 10.1007/978-3-540-74958-5_42.

[22] Guo C, Pleiss G, Sun Y, Weinberger K Q. On calibration of modern neural networks. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.1321-1330.

[23] Jin W, Derr T, Wang Y, Ma Y, Liu Z, Tang J. Node similarity preserving graph convolutional networks. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, March 2021, pp.148-156. DOI: 10.1145/3437963.3441735.

[24] Shanthamallu U S, Thiagarajan J J, Spanias A. Uncertainty-matching graph neural networks to defend against poisoning attacks. In *Proc. the 35th AAAI Conference on Artificial Intelligence*, Feb. 2021, pp.9524-9532.

[25] Hansen L K, Salamon P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, 12(10): 993-1001. DOI: 10.1109/34.58871.

[26] Breiman L. Bagging predictors. *Machine Learning*, 1996, 24(2): 123-140. DOI: 10.1023/A:1018054314350.

[27] Dietterich T G. Ensemble methods in machine learning. In *Proc. the 1st International Workshop on Multiple Classifier Systems*, June 2000, pp.1-15. DOI: 10.1007/3-540-45014-9_1.

[28] Kuncheva L I, Whitaker C J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 2003, 51(2): 181-207. DOI: 10.1023/A:1022859003006.

[29] Abney S P. Bootstrapping. In *Proc. the 40th Annual Meeting of the Association for Computational Linguistics*, July 2002, pp.360-367. DOI: 10.3115/1073083.1073143.

[30] Balcan M, Blum A, Yang K. Co-training and expansion: Towards bridging theory and practice. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2004, pp.89-96.

[31] Wang W, Zhou Z. A new analysis of co-training. In *Proc. the 27th International Conference on Machine Learning*, June 2010, pp.1135-1142.

[32] Wang W, Zhou Z H. Theoretical foundation of co-training and disagreement-based algorithms. arXiv:1708.04403, 2017. http://arxiv.org/abs/1708.04403, Oct. 2021.

[33] Ma F, Meng D, Xie Q, Li Z, Dong X. Self-paced co-training. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.2275-2284.

[34] Ma F, Meng D, Dong X, Yang Y. Self-paced multi-view co-training. *Journal of Machine Learning Research*, 2020, 21: Article No. 57.

[35] Qiao S, Shen W, Zhang Z, Wang B, Yuille A L. Deep co-training for semi-supervised image recognition. In *Proc. the 15th European Conference Computer Vision*, Sept. 2018, pp.142-159. DOI: 10.1007/978-3-030-01267-0_9.

[36] Tramèr F, Papernot N, Goodfellow I, Boneh D, McDaniel P. The space of transferable adversarial examples. arXiv:1704.03453, 2017. http://arxiv.org/abs/1704.03453, Oct. 2021.

[37] Klicpera J, Bojchevski A, Günnemann S. Predict then propagate: Graph neural networks meet personalized PageRank. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[38] Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and deep graph convolutional networks. In *Proc. the 37th International Conference on Machine Learning*, July 2020, pp.1725-1735.

[39] Chung F R. Spectral Graph Theory. American Mathematical Society, 1997.

[40] Von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing*, 2007, 17(4): 395-416. DOI: 10.1007/s11222-007-9033-z.

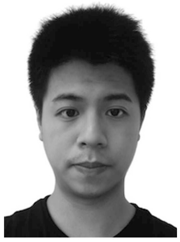
[41] Teixeira L, Jalaian B, Ribeiro B. Are graph neural networks miscalibrated? arXiv:1905.02296, 2019. http://arxiv.org/abs/1905.02296, Oct. 2021.

[42] Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. *AI Magazine*, 2008, 29(3): 93. DOI: 10.1609/aimag.v29i3.2157.

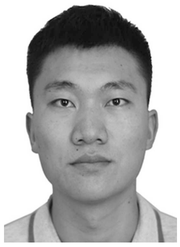
[43] Geisler S, Schmidt T, Şirin H, Zügner D, Bojchevski A, Günnemann S. Robustness of graph neural networks at scale. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2021, pp.7637-7649.

[44] Li Y, Jin W, Xu H, Tang J. DeepRobust: A platform for adversarial attacks and defenses. In *Proc. the 35th AAAI Conference on Artificial Intelligence*, Feb. 2021, pp.16078-16080.

[45] Shchur O, Mumme M, Bojchevski A, Günnemann S. Pitfalls of graph neural network evaluation. In *Proc. the Relational Representation Learning Workshop of 2018 NeurIPS*, Dec. 2018.

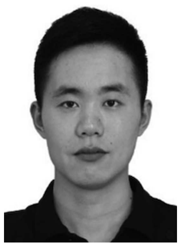


Xu-Gang Wu received his B.S. degree in computer science and technology from National University of Defense Technology, Changsha, in 2017. He is now pursuing his Ph.D. degree in the College of Computer, National University of Defense Technology, Changsha. His research interests include AI security and explainable AI.

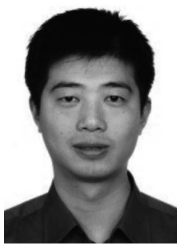


Hui-Jun Wu received his Ph.D. degree in computer science and technology from the University of New South Wales, Sydney, in 2019. He is now an assistant professor in the College of Computer, National University of Defense Technology, Changsha. His research interests include the robustness and interpretability of machine learning models, and system software for high-performance computing.

Xu Zhou received his B.S., M.S., and Ph.D. degrees in computer science and technology, from National University of Defense Technology, Changsha, in 2007, 2009, and 2014, respectively. He is now an assistant professor in the College of Computer, National University of Defense Technology, Changsha. His research interests include operating systems and parallel computing.



Xiang Zhao received his Ph.D. degree in computer science and technology, from the University of New South Wales, Sydney, in 2014. He is currently a professor in the National University of Defense Technology, Changsha, and he is the head of the Knowledge Systems Engineering Group. His research interests include graph data management and mining, with a special focus on knowledge graphs.



Kai Lu received his B.S. degree and Ph.D. degree in computer science and technology, from National University of Defense Technology, Changsha, in 1995 and 1999, respectively. He is now a professor in the College of Computer, National University of Defense Technology, Changsha. His research interests include operating systems, parallel computing, and software security.