Supplementary Material

1 Dataset Statistics

Our experiments use the datasets analyzed in (Shchur et al., 2018) and (Zügner et al., 2018). For targeted node detection, we report results for citation datasets Cora, Citeseer, Pubmed, and political blogs network Polblogs. For subset detection, we also include results for collaboration networks Coauthor CS and Coauthor Physics, co-purchase graphs Amazon Computers and Amazon Photo, and a larger citation network Cora-Full. We followed the pre-processing steps of (Shchur et al., 2018).[1] The detailed dataset statistics are provided in Table 1.

The **edge density** describes the fraction of all possible edges that is present in the graph and can be computed as: $\frac{\#edges}{\frac{1}{2}\#nodes^2}$

Table 1: Statistics of evaluation datasets.

Dataset	#Classes	#Features	#Nodes	Edges	Edge density
Cora	7	1,433	2,485	5,069	0.04%
Citeseer	6	3,703	2110	3,668	0.04%
Polblogs	2	N/A	1222	16714	2.20%
Pubmed	3	500	19,717	44,324	0.01%
Coauthor CS	15	6,805	18,333	81,894	0.01%
Coauthor Physics	5	8,415	34,493	247,962	0.01%
Am-comp	10	767	13,381	245,778	0.07%
Am-photo	8	745	7,487	119,043	0.11%
Cora-Full	67	8,710	18,703	62,421	0.01%

[1] `https://www.kdd.in.tum.de/research/gnn-benchmark/`

2 Graph Adversarial Attack Detection Experiment Details

2.1 Detection of Nodes Perturbed by Target Attacks

In this section we provide more detail regarding the procedure for target node selection for both Nettack and the DICE attack. We follow a target node selection procedure similar to that described in (Zügner et al., 2018; Zhang et al., 2019) to perform the Nettack and Dice attacks. We split the dataset into a training set (10%), validation set (10%) and test set (80%). The validation set is used to evaluate stopping criteria during training. All the hyper-parameters used when performing the Nettack and the surrogate model training (learning rate, hidden dimensions, weight decay, etc.) are set to be the same as the original implementation in (Zügner et al., 2018). We average over five different random initializations, where for each we perform the following steps. We first train our surrogate model on the labeled data. The desired target node number is K. From all nodes in the test set that have been correctly classified, we select (i) the $K/4$ nodes with highest margin of classification, indicating they are clearly correctly classified, (ii) the $K/4$ nodes with the lowest margin, which are nodes that have much more uncertain classifications and (iii) $K/2$ more nodes randomly from the remaining nodes which are correctly classified. These selected nodes serve as the target nodes for the attacks. [Note that for the case $K = 10$, all the target nodes are randomly selected from the nodes which are correctly classified]. We then perform Nettack or Dice on each of the target nodes and obtain the perturbed adjacency matrix. Next, we train the victim (prediction) model and evaluate our designed multi-JSD statistic to measure the local smoothness for all nodes. The objective of our single-node detection scheme is to identify these perturbed nodes without generating too many false alarms for the other clean nodes. As described in the main paper, the calculated multi-JSD metric is used for this task.

2.2 Training details for the prediction model

After obtaining the perturbed neighborhood of all the target nodes, we train a graph-based prediction (victim) model. Then, we perform the adversarial attack detection procedure. As described in the experiments section of the main paper and Section 3.2 of the supplementary material, we experiment with four commonly-used GNNs as the prediction model to demonstrate the generalization ability of our proposed technique. Under the same perturbed graph, we employ GCN (Kipf and Welling, 2017), SGCN (Wu et al., 2019a), GAT (Velickovic et al., 2018) and GMNN (Qu et al., 2019) as the prediction model. We use the same hyper-parameters as employed in the original implementations reported in the four corresponding papers.

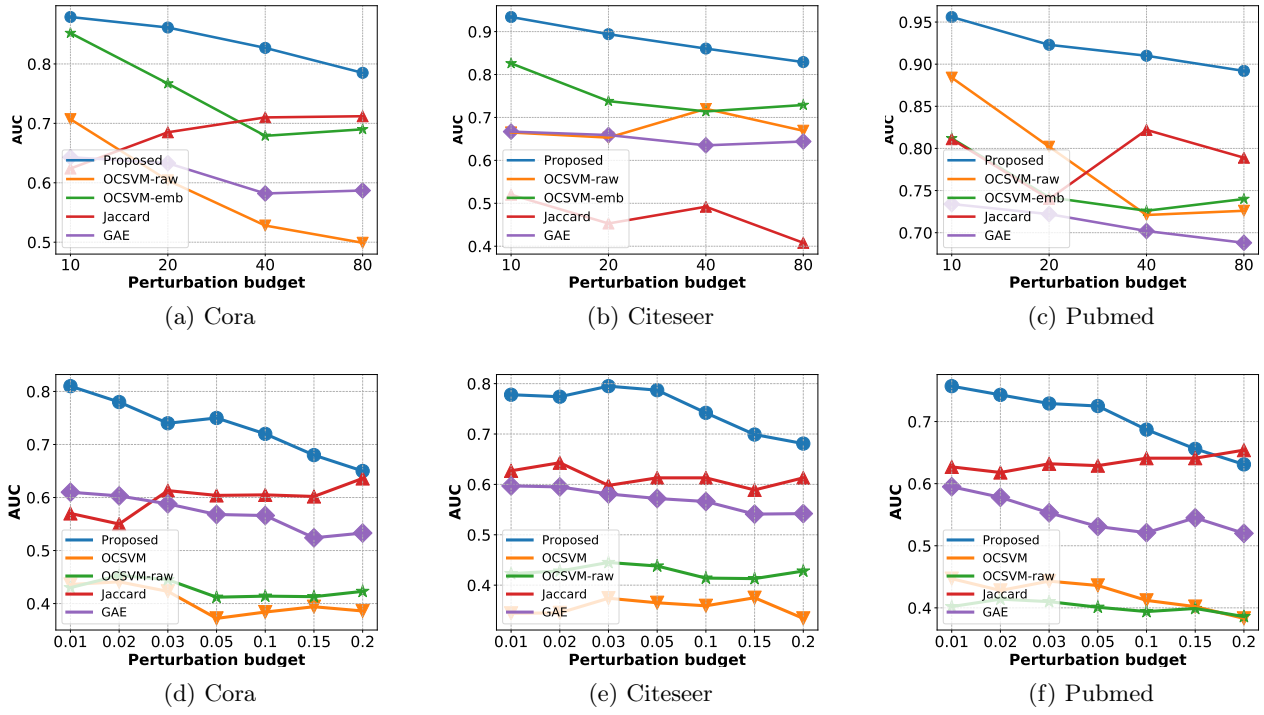
2.3 Detection of Corrupted Nodes Subsets

For the experiments described in the paper, the MMD computation and optimization were performed using the shogun library (Sonnenburg et al., 2010). For each value of M, we divided the test set (and comparison set) in two, and then used one of the halves to choose the optimal bandwidth of the RBF kernel using 3-fold cross validation, selecting from the values $[0.1, 1, 10]$. The γ_1, γ_2 parameters were set to be the determined optimal bandwidth multiplied by 0.001. These values were chosen so that the node feature vector and the adjacency vector have minimal impact on the output of kernel evaluations (they are determined almost entirely by the differences in the graph embedding function outputs).

3 Detection of Targeted Node Attacks Extended Experiments Results

3.1 Detection performance under different severities of attacks

In this additional experiment, we explore the effectiveness of our proposed single node detector under different severities of attack. For the targeted attack Nettack (Zügner et al., 2018), we explore the detection AUC with different numbers of attacked nodes ($K = 10, 20, 40, 80$) for Cora, Citeseer and Pubmed. For the global Dice attack, we explore the detection AUC with different global edge perturbation ratios ($\Delta = 0.01 - 0.2$) for Cora, Citeseer and Pubmed. As demonstrated in Figure 1, our proposed detector outperforms alternatives in the majority of the cases. The performance of all methods degrades slightly as K and Δ increase with the exception of the Jaccard similarity based method (Wu et al., 2019b).



(a) Cora (b) Citeseer (c) Pubmed

(d) Cora (e) Citeseer (f) Pubmed

Figure 1: (a) - (c) Detection AUC with the different anomaly number $K = 10, 20, 40, 80$ for Cora, Citeseer and Pubmed. (d) - (f) Detection AUC with the different global edge perturbation ratio $\Delta = 0.01 - 0.2$ for Cora, Citeseer and Pubmed.

3.2 Generalization of the detection: effectiveness of the detector for other GNN prediction models

In the main paper, we assume that the victim of the adversarial attacks trains the GCN model from (Kipf and Welling, 2017). The GCN is one of the most popular models in the literature and it is reasonable to expect that the GCN or architectures similar to it will potentially be attacked in many realistic scenarios. In this experiment, we evaluate if our proposed target node detector is effective on other commonly used graph neural network models including simplified GCN (SGCN) (Wu et al., 2019a), Graph Attention Networks (Velickovic et al., 2018), and Graph Markov Neural Networks (GMNNs) (Qu et al., 2019). We conduct the same experiments using the SGCN, GAT and GMNN as the prediction model instead of the GCN model. As shown in Table 2, we observe very similar performance, which demonstrates the capability of our proposed detector to generalize to different graph learning models. When using GAT as the predictive model, our detector consistently achieves the best performance; this likely relates to GAT's ability to focus on similar neighbours and ignore noisy edges.

Table 2: Detection Area-under-Curve (AUC) % under various graph prediction models

Model	**SGCN**	**GCN**	**GAT**	**GMNN**	**SGCN**	**GCN**	**GAT**	**GMNN**
	Cora				Polblogs			
Nettack	86.7	86.2	91.0	86.9	77.6	79.7	85.6	80.3
Dice $\Delta = 0.5$	84.1	83.9	84.7	80.2	71.7	75.3	89.2	74.4
	Citeseer				Pubmed			
Nettack	87.3	87.9	91.5	89.7	82.3	91.0	93.3	92.1
Dice $\Delta = 0.5$	80.1	81.6	88.2	77.1	71.1	70.8	80.1	80.7

3.3 Comparison between the desired False Positive rate and the empirical False Positive rate.

As we discussed in the main paper, the KDE fitting and thresholding procedure requires an i.i.d. assumption on the JSD statistics of different nodes. In a graph setting, nodes influence each other and so the i.i.d. assumption is

Table 3: Comparison between the desired False Positive rate and the actual False Positive rate.

	0.05	0.15	0.25	0.35	0.45	0.55
Cora	0.048	0.147	0.254	0.353	0.455	0.554
Citeseer	0.047	0.145	0.247	0.352	0.448	0.546
Pubmed	0.047	0.153	0.253	0.354	0.449	0.554
Polblogs	0.048	0.153	0.248	0.353	0.456	0.561

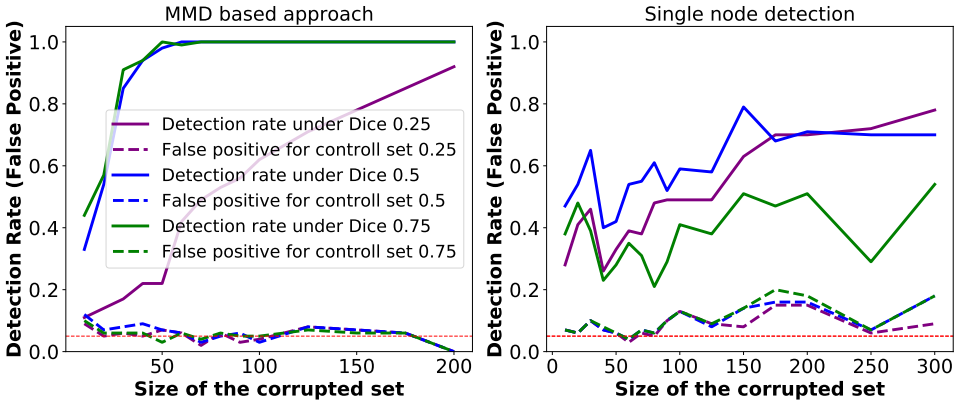


Figure 2: Comparison between two sample test approach and single node detection approach (Citeseer)

violated, and we are not guaranteed to achieve the desired false alarm rate. We can evaluate the impact of the dependencies by comparing the theoretical target false alarm rate with the achieved empirical false alarm rate. As demonstrated in Table 3, the empirical false alarm rate is close to the theoretical target value. In Table 3, the first row is the desired false positive rate that we use to match the tail probability of the kernel density estimator. The following rows are the actual false-alarm rates obtained on the non-perturbed test nodes for different datasets under Nettack. Similar results are obtained under other attacks.

4 Discussion and Additional Results for the Detection of Corrupted Node Sets

4.1 Analysis of the necessity of the Corrupted node subset detection algorithm

We compared the performance of the MMD subset detection approach to a strategy that involves repeated application of single node detection, with declaration of a subset attack if any node in the subset is flagged.

Following the MMD experiment setting, we design a comparison experiment. For every trial of the experiment, we have one perturbed set $\mathcal{S}$, one clean set $\mathcal{S}'$ (that contains the same nodes as $\mathcal{S}$ but has no topology perturbations) and one clean set $\mathcal{R}$.

For the procedure we need to determine a suitable detection threshold. We use all clean sets $\mathcal{R}$ to select a threshold that leads to a false-positive rate below 5 percent. We then use the threshold $\tilde{\tau}$ to evaluate every node in the perturbed node set $\mathcal{S}$. If the multi-JSD statistic of any node in this set is above this threshold, we identify the set $\mathcal{S}$ as being perturbed. We repeat this procedure 100 times under different partitions of $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{T}$, then we report the overall detection rate on all the perturbed sets $\mathcal{S}$ and the false positive rate for clean sets $\mathcal{S}'$.

For each node in the perturbed set $\mathcal{S}$, we perform the Dice attack with budgets Δ of $(d+1)/4$, $(d+1)/2$ and $3(d+1)/4$, where d is the degree of each node.

We observe in Figure 2 that the detection performance of the MMD is much superior. Due to the multiple hypothesis testing effect, the threshold of the single node detector must be set very high to avoid too many false alarms and as a result the detection rate is very low.

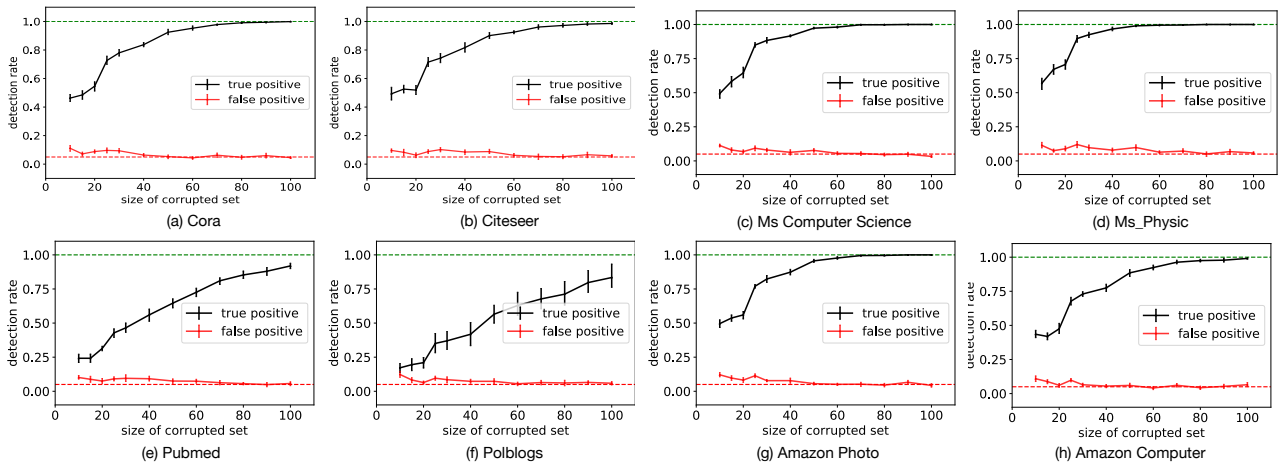


Figure 3: Detection performance (with confidence interval 5/95) for subgraph attack detection for 8 datasets under Dice attack.

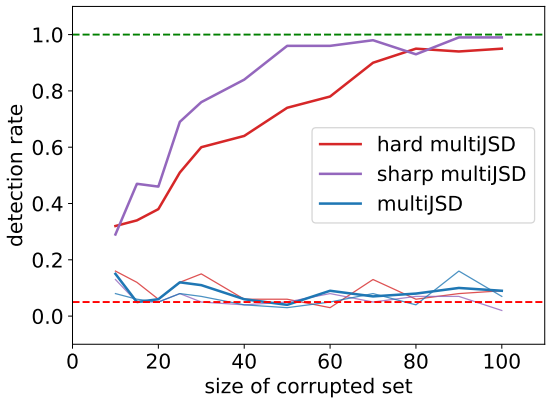


Figure 4: MMD subset detection performance for Cora Full dataset as a function of the size of the corrupted set for the Dice attack.

4.2 Additional experiment results for Corrupted Nodes Subsets Detection

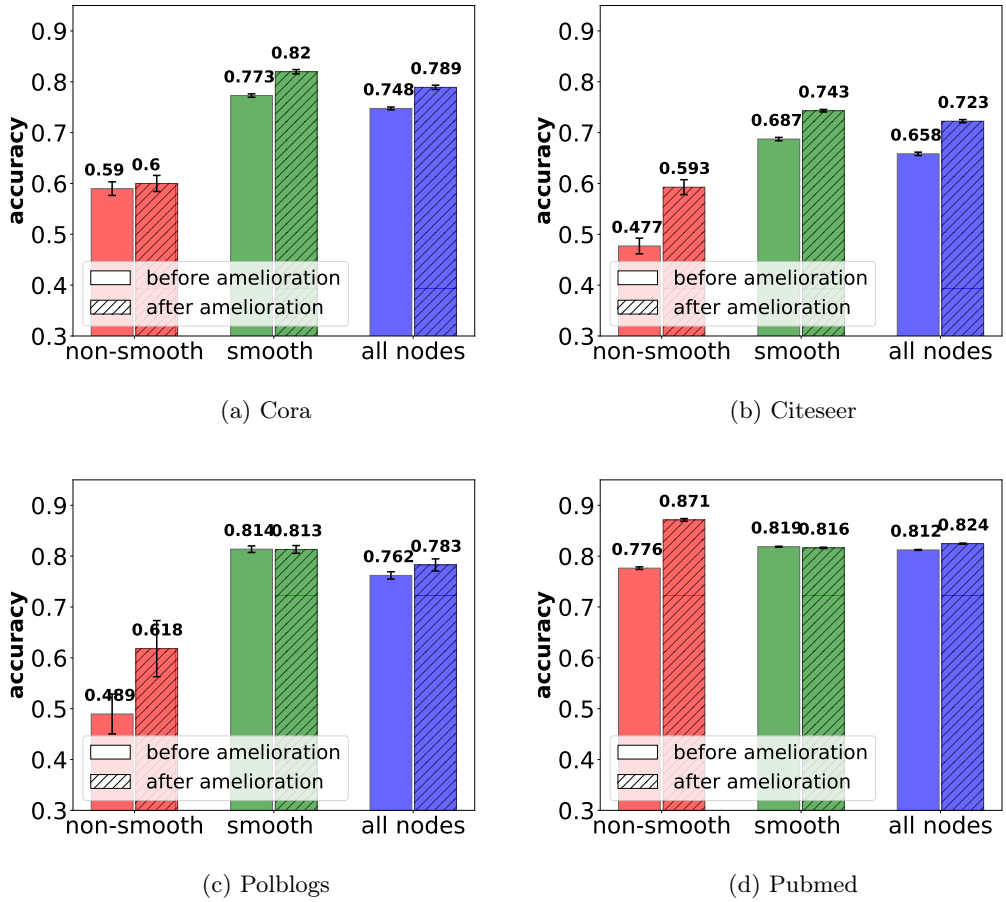
Because of the space limitations in the main paper, we were unable to display all results for the MMD detection test. In this section we provide the MMD detection results for Cora Full. Figure 4 shows the detection performance for this dataset. It is clear that for this dataset, which has many more classes, the multiJSD statistic does not perform well. It is important to apply sharpening of the softmax values to improve the discriminatory capability.

The MMD detection experiment results for the other eight datasets with the sharpened multiJSD metric (with confidence intervals) are shown in Figure 3.

5 Discussion for the Defense Experimental Results

5.1 Case Study

In this case study, we want to trace the main performance improvement from our proposed defense mechanism. As shown in Figure 5, our proposed correction scheme can improve the overall performance when the graph topology is under attack. In particular, it can significantly improve the accuracy of the nodes that have a very non-smooth neighborhood (the multiJSD statistic is above the detection threshold). Additionally, we tracked the relative classification accuracy on different groups of nodes with different inner community ratio before and after



(a) Cora

(b) Citeseer

(c) Polblogs

(d) Pubmed

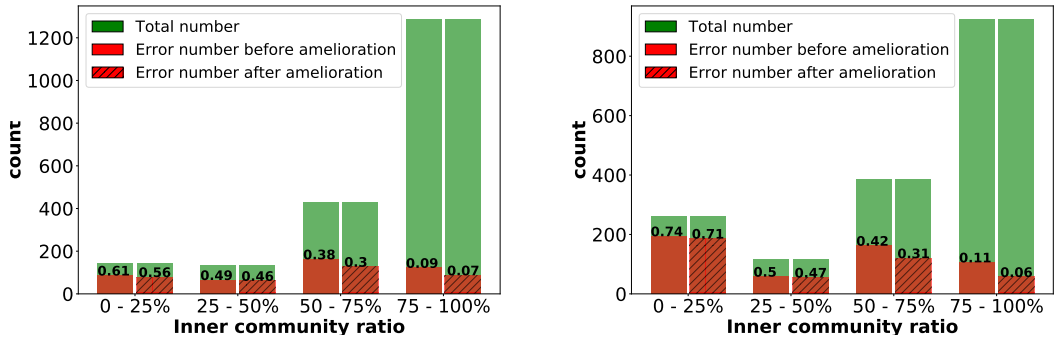
Figure 5: Accuracy comparison before and after the topology amelioration. The uncertainty represents the 95/5 confidence interval for **Cora**, **Citeseer**, **Polblogs**, and **Pubmed** under Meta-attack

applying to proposed defence to modify the topology. The barplots in Figure 6 show that, for a poisoned dataset, the classification accuracy on nodes with lower inner-community ratio (meaning larger discrepancy in terms of neighborhood) improves more significantly once we have applied the defence strategy.

6 Bitcoin Dataset Experiments Details and Additional Results

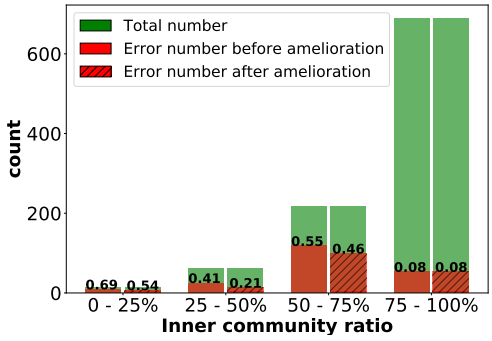
6.1 Dataset preparation: Elliptic Bitcoin Dataset

In the original Bitcoin dataset (Weber et al., 2018), there is a time stamp associated with each transaction. The data is collected over a period of approximately two weeks, and this is divided into 49 distinct "time steps". Each time step contains a single connected component of transactions that appeared on the blockchain within less than three hours of each other. In our experiment, we select for analysis the 5 time steps that contain the most illicit transactions. The statistics of the dataset for these 5 time steps are shown in Table 4.

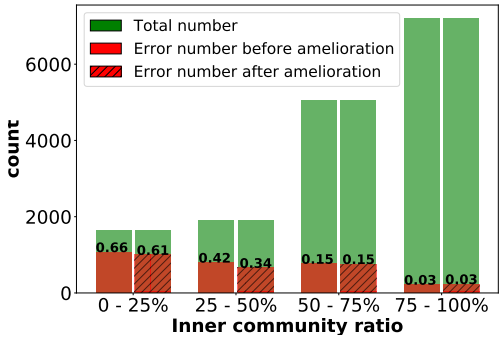


(a) Cora

(b) Citeseer



(c) Polblogs



(d) Pubmed

Figure 6: Accuracy comparison before and after the topology amelioration for different cross-community ratio nodes.

Table 4: Dataset statistics for Elliptic Dataset. The numbers in the brackets represent the time slice index in the original dataset.

	Transaction Number	Illicit Transaction Number	Edge Number
TS 1 (12)	472	127	982
TS 2 (19)	395	83	830
TS 3 (23)	666	97	1468
TS 4 (28)	617	222	1254
TS 5 (31)	532	129	1140

6.2 Selection of target nodes for the Elliptic Dataset

In a real world application, and specifically in this Bitcoin transaction network, the target node selection strategy employed in Section 2.1 might not be representative of anticipated attacks. We simulate attacks by modifying the topology of the transaction network. In practice, an attack is more likely to take the form of an entity trying to disguise an illicit transaction by linking it with licit transactions. That is, some of the received Bitcoin from an illicit transaction will be distributed to reputable parties. Only those responsible for the illicit transactions have a motive to modify the future transaction flows of the Bitcoin to hide their activities. We therefore focus on perturbing nodes in the graph that are associated with illicit transactions. We use target attack number $K = 40$ in all experiments, where 40 illicit transactions (nodes) maliciously alter their bitcoin flow to disguise their nature.

We conduct two types of targeted node attack to obtain the perturbed graph: Nettack (Zügner et al., 2018) and Dice (Waniek et al., 2018). The Dice attack reflects a scenario where an attacker has tried to disguise an illicit transaction by making in parallel multiple licit transactions with reputable recipients. The Nettack models a scenario where an entity is trying to disguise multiple illicit transactions and has a global budget on the number of fake connections that can be inserted. We use the same settings for Nettack's constraint on unnoticeability and Dice's attack budget $\Delta = 0.5$ as used for the other node classification experiments.

6.3 Training Details

After obtaining the perturbed neighborhood of all the target nodes, we train a GCN (Kipf and Welling, 2017) model as the prediction (victim) model to perform the adversarial attack detection procedure. We use all the default hyper-parameters from the original paper (Kipf and Welling, 2017) to perform the training. We split the dataset into a training set, validation set and test set under the ratio of 50% train, 10% and 40% respectively.

6.4 Additional Results and Analysis

6.4.1 Impact of the graph-based attack on the bitcoin dataset

In Table 6 of the main paper, we demonstrate the dramatic negative impact that the graph adversarial attack algorithms can have on performance for the Bitcoin datasets. We focused on how the attacks can change the classification outcome, thus disguising the illicit transactions. We now examine the impact of the attacks from a different perspective, examining how the classification margin is affected by the attack.

For node v with true class c_{true}, the classification margin is:

$$\text{margin}_v = p(Z_v = c_{true}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) - \max_{c \neq c_{true}} p(Z_v = c|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) \quad (1)$$

When a positive classification margin is pushed closer towards zero, the classifier is much more uncertain of its decision. For values smaller than 0, the nodes are misclassified, meaning that the disguised illicit transactions fool our model and are identified by the classifier as licit transactions.

Figure 7 shows how the classification margin changes after the application of Nettack for time step 3 of the Bitcoin dataset. For this dataset, Nettack results in a considerable reduction in the classification margin, with many values becoming negative.

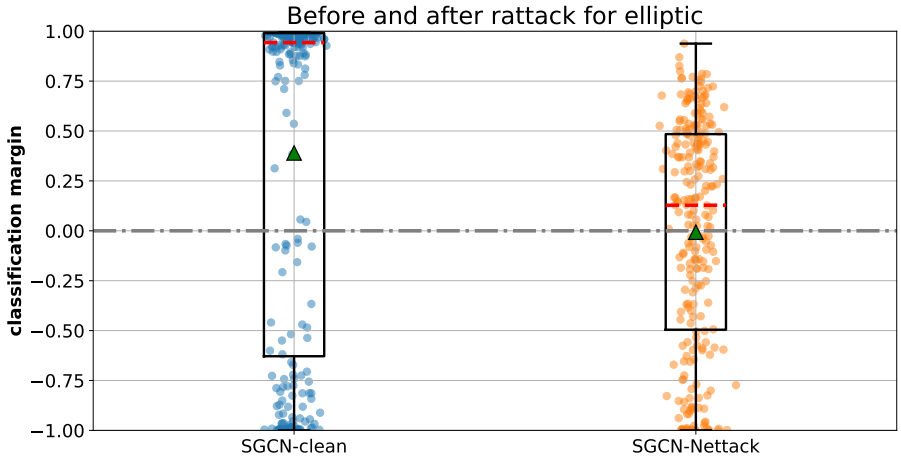


Figure 7: Performance comparison before and after the random attack for all illicit transactions in the Elliptic Bitcoin Dataset (TS 3).

6.4.2 Target node detector on the Dice attack

In Table 6 of the main paper, we show how effective Nettack is in deceiving a GCN classifier and disguising illicit transactions. In this section, we extend the experiment to the Dice (Waniek et al., 2018) attack with $\Delta = 0.5$. In the right column of Table 5, we report the detection AUC for our single node detector. The proposed detector achieves high AUC scores for Dice attack as well, successfully detecting most attacked transactions without generating many false alarms. The performance is fairly consistent over all time steps, although the proposed method performs worse for time step 2 for the Dice attack. This time step has the fewest illicit nodes and the fewest edges of the graphs we analyze.

Table 5: Left: Prediction accuracy (%) for the illicit transactions before and after the adversarial graph perturbations (Nettack, Dice). Right: Detection Area-under-Curve (AUC) under various attacks using GCN as the prediction model.

	Clean	**Under Nettack**	**Under Dice**	**Detection Nettack**	**Detection Dice**
TS 1	92.5	85	72.5	83.5	70.7
TS 2	72.5	50.0	32.2	78.0	68.4
TS 3	90.0	62.5	50.6	86.3	85.4
TS 4	97.5	67.5	51.4	87.1	87.6
TS 5	95.0	55.5	31.9	75.9	84.0

6.5 Case study

In Table 6 of the main paper, we show how effective the attacks are in deceiving a GCN classifier and disguising illicit transactions. In Figure 8, we visualize a special case to demonstrate the severity of the attacks on this real application and to highlight the effectiveness of our defense strategy against existing adversarial attacks.

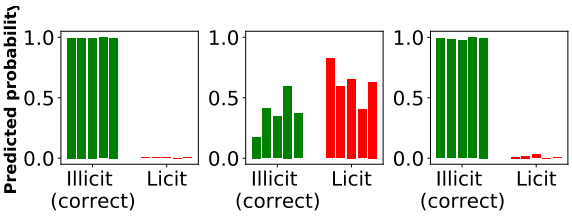


Figure 8: Left: For clean data, a GNN can identify all illicit transactions. Middle: After the adversarial graph attack (Nettack) (Zügner et al., 2018), the GNN makes many errors. Right: After the defence, our model can identify almost all malicious transactions correctly despite the disguise.

7 Adaptive Attack Experiment Details and Additional Results

7.1 Motivation

We further investigate a possible adaptive adversarial attack setting where we assume that the attackers know how our proposed detector works. Many adversarial detection methods have been proposed for other tasks such as classification of image data and most of them have proven ineffective if an adaptive attack is applied (Carlini and Wagner, 2017; Tramer et al., 2020; Roth et al., 2019). We aim to investigate if the same phenomenon will happen under a graph adversarial attack setting and further explore if an advanced attack can be designed to avoid being detected.

Knowledge of our proposed multi-JSD metric provides the graph model attackers with a new avenue to disguise their attack by using the metric as an extra constraint. In general, adversarial perturbations aim to fool the victim model but at the same time remain unnoticeable. However, in contrast to images, where 'unnoticeable' can be defined based on human perception, it is less clear how unnoticeable should be defined in a graph setting.

Existing graph adversarial attack models use two main criteria to define 'unnoticeable'. They either constrain a budget on how many edges can be changed for each target node (Waniek et al., 2018; Chang et al., 2020; Zügner and Günnemann, 2019) or they require preservation of characteristics of the graph such as the degree distribution (Zügner et al., 2018). However, a slight change of both criteria can easily lead to a huge impact on the predictions of the downstream task, which explains why our designed detector is effective against the existing attacks. An alternative, improved version of the existing graph attacks can be designed based on considering our proposed metric as an additional component of the definition of 'unnoticeable'.

7.2 Experiment details

We conduct the adaptive attack experiment on two targeted attack algorithms including Nettack (Zügner et al., 2018) and Dice (Waniek et al., 2018) with the assumption that the adversary has complete knowledge of the inner workings of our defense (the attacker is aware that a local discrepancy statistic test will be applied). Our designed adaptive attacks add a constraint that the multi-JSD statistic must not be increased to the extent that the node would be flagged as attacked by the proposed detection test. For a target node i, we only accept local perturbations $\tilde{\mathcal{G}}_i$ where the local discrepancy test (multi-JSD statistics) fulfills $\text{JSD}\left(\tilde{\mathcal{P}}_i\right) < \tau$, where τ is the threshold that the defense strategy uses to filter out high local discrepancy nodes and $\tilde{\mathcal{P}}_i = \{\tilde{p}_j\}_{j \in \tilde{\mathcal{N}}(i)}$ is the set of softmax probabilities (computed using the surrogate model) in the perturbed neighbourhood of node i.

We select 40 target nodes for each attack. For Dice attack, we use the perturbation budget $\Delta = \frac{1}{2}(d+1)$, where d is the degree of the targeted node For each target node, the proposed perturbations are only permitted if the resultant multi-JSD is below a threshold τ. In our experiment τ is selected to correspond to the 5% false positive rate on training data. In adapt-DICE, each proposed random cross-community edges is evaluated and the modification is rejected if the multi-JSD threshold is exceeded. In adapt-Nettack, we conduct a local greedy search for the node that has the greatest impact on classification, but only accept those that respect the multi-JSD limit. For both attacks, if no feasible edge can be found given the maximize search iteration (we use 50 iterations in our experiments), we will reduce the perturbation budget until we are able to find a perturbation edge sets that satisfy the multi-JSD constraint.

7.3 Extended results

In Section 6.3 of the main paper, we report the detection rate of the adaptive attacks before and after adding the constraint. It clearly demonstrates the effectiveness of the adaptive attack to avoid being flagged as an anomaly. In Table 6, we report the AUC comparison for the adaptive attack experiments to further validate our observation. The detection AUC drops from 5.6% to 26.2% across four different datasets.

Table 6: Comparison of AUC (%) at 5% false-alarm threshold between the original attacks and the adaptive attacks multi-JSD constraint.

	Nettack	**Nettack-mJSD**	**DICE**	**DICE-mJSD**
Cora	86.4	76.1	84.4	74.9
Citeseer	83.2	66.4	82.0	71.8
Polblogs	88.9	62.7	67.3	61.7
Pubmed	91.0	65.8	65.8	59.8

References

Carlini, N. and Wagner, D. (2017). Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. ACM Workshop on Artificial Intelligence and Security*.

Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Zhu, W., and Huang, J. (2020). A restricted black-box adversarial framework towards attacking graph embedding models. In *Proc. AAAI Int. Conf. Artificial Intelligence*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learning Representations*.

Qu, M., Bengio, Y., and Tang, J. (2019). GMNN: Graph Markov neural networks. In *Proc. Int. Conf. Machine Learning*.

Roth, K., Kilcher, Y., and Hofmann, T. (2019). The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *CoRR*.

Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F. d., Binder, A., Gehl, C., and Franc, V. (2010). The shogun machine learning toolbox. *J. Mach. Learn. Res.*, 11:1799–1802. version 6.1.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. (2020). On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *Proc. Int. Conf. Learning Representations*.

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

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., and Leiserson, C. E. (2018). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In *2019 KDD Deep Learning on Graphs: Methods and Applications workshop (DLG'19)*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *Proc. Int. Conf. Machine Learning*, pages 6861–6871, Long Beach, California, USA.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. (2019b). Adversarial examples for graph data: Deep insights into attack and defense. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 4816–4823.

Zhang, Y., Pal, S., Coates, M., and Üstebay, D. (2019). Bayesian graph convolutional neural networks for semi-supervised classification. In *Proc. AAAI Int. Conf. Artificial Intelligence*.

Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, pages 2847–2856.

Zügner, D. and Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. *CoRR*, abs/1902.08412.