Graph Attacking - Node Injection via Fake Duplicate

Zhuoer Wang
University of Illinois at Urbana-Champaign
Urbana, Illinois, United States
zhuoerw3@illinois.edu

Yulin Zhao
University of Illinois at Urbana-Champaign
Urbana, Illinois, United States
yulin6@illinois.edu

Pengyu Lu
the University of Illinois at Urbana-Champaign
Urbana, Illinois, United States
pengyul4@illinois.edu

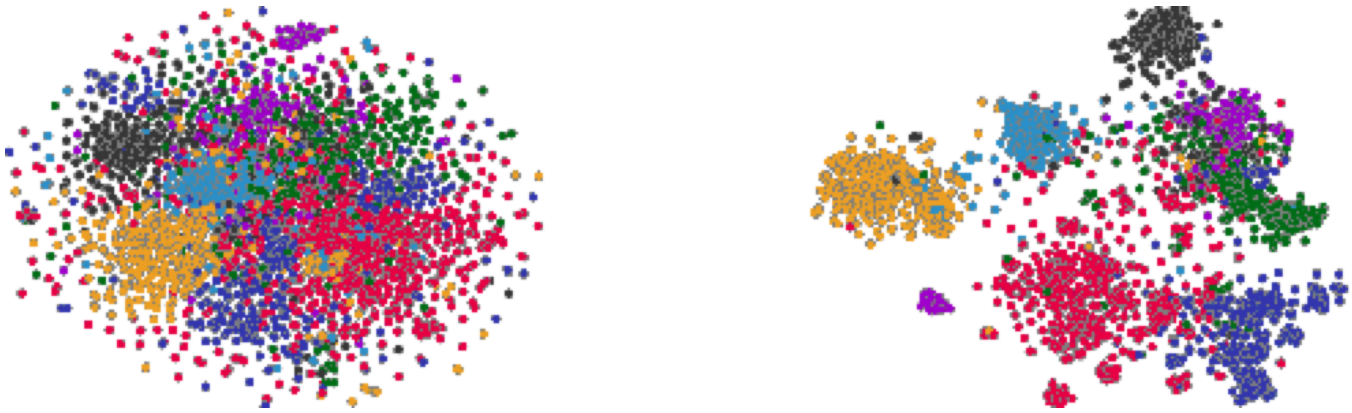

Figure 1: Node classification

ABSTRACT

Node classification is a major task in graph data mining. Attacking is important in enhancing the robustness of classification models. In this project, we proposed a new attacking method by injecting fake duplicated nodes with flipped labels. For the next step, we will implement reinforcement learning for the node duplication process.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

data mining, node classification, graph attack, poisoning attack, graph convolutional network, reinforcement learning

ACM Reference Format:
Zhuoer Wang, Yulin Zhao, and Pengyu Lu. 2022. Graph Attacking - Node Injection via Fake Duplicate. In *Proceedings of* . ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Attacking is an important part of developing defense strategies and improving the robustness of models. However, there are few works on attacking the network graphs. Nowadays, graph mining algorithms, including node classification and link prediction, are widely used in various application cases. Due to the graph network structure, it is hard to measure the change amount or design a very intuitive attacking algorithm. In this project, we propose a new attacking algorithm by adding fake nodes and edges to poison the graph in the training stage. After the poisoning, the accuracy of the node classification task trained on the attacked graph will be significantly lower than that trained on the clean graph. Our major tasks include deciding the attacking methods, setting a constraint by defining an effective metric for small modification, and implementing reinforcement learning to select the attacked nodes or edges. We will have experiments on popular datasets for node classification tasks, including the CORA[5], the GitHub Social Network[6], and the Deezer Europe Social Network[7].

, *0.00*

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 TEAM MEMBERS

2.1 Zhuoer Wang

- Prepare the data sets and collaborate with other teammates to reproduce models
- Design our algorithm and implement 1/3 part of the functions
- Work on 1/3 amount of test cases
- Discuss and improve the algorithm

2.2 Yulin Zhao

- Reproduce the previous node classification models
- Design our algorithm and implement 1/3 part of the functions
- Work on 1/3 amount of test cases
- Discuss and improve the algorithm

2.3 Pengyu Lu

- Reproduce the previous adversarial attacking models
- Design our algorithm and implement 1/3 part of the functions
- Work on 1/3 amount of test cases
- Discuss and improve the algorithm

3 RELATED WORK

Our project targeting on the adversarial attack for the node classification model based on Graph Convolutional Network[3]. In recent years, graph adversarial attack has been a heated topic and different algorithms and frameworks have been proposed to perform the attacks on GCN. In 2017, a reinforcement learning based method, NIPA[9], was designed to sequentially modify the adversarial information of the injected node. NIPA targeting on attacking node classification tasks. Later, Ma et al.[4] proposed another attacking framework via rewiring, which tries to modify the graph in a way that minimizes the change in the graph structure. The framework is named Rewatt and is designed for graph classification tasks attacking.

3.1 Graph Convolution Network

Our adversarial attack aims at decreasing the accuracy of the node classification model based on the Graph Convolutional Network[3]. The GCN model learns node representations via propagation in multiple convolutional layers. The layer-wise propagation rule is defined as

$$H^{(l)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l-1)})$$

where $\tilde{A} = A + I_N$is the adjacency matrix of the graph with self-connection, $\tilde{D}_{ij} = \sum_j \tilde{A}_{ij}$, $W^{(i)}$ is the layer-specific weight matrix to be trained and $H^{(l)}$ is the activation matrix of the current layer with $H^{(0)} = X$ where X is the matrix of node feature. $\sigma(\cdot)$ is the activation function of the current layer.

We adopted a two-layer GCN and the model is reorganized by calculating $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ in the pre-processing step, which simplifies the forward model in the following form:

$$f(X, A) = \sigma_2(\hat{A}\sigma_1(\hat{A}XW^{(0)})W^{(1)})$$

where the weights $W^{(0)}$ and $W^{(1)}$ are trained with gradient descent.

3.2 Graph Attacking Algorithm

In 2017, a reinforcement learning method named NIPA[9] is proposed for node injection attacks. In this new method, the link structure of the original graph is not modified since no nodes or edges from the original graph are deleted. NIPA determines the label of the injected nodes and the fake edges in the graph using a Finite Horizon Markov Decision Process(MDP) and hierarchical Q-learning network. Each state change in MDP is regarded as one poisoning attack action, in which only one label of an injected node is changed and only one new fake edge between this injected node and one other node is established.

Another graph adversarial attack method that inspires us is Graph Adversarial Attack via Rewiring[4]. This paper aims at modifying the structure of the original graph to perform the attack. To ensure that the modification changes the structure of the graph in an "unnoticeable" way, the authors proposed a rewiring operation. The rewiring operation involves three nodes, noticed as N1, N2, and N3. The three nodes should satisfy the following condition: N2 is a first-hop neighbor of N1, and N3 is a second-hop neighbor of N1 but not a first-hop neighbor of N1. The rewiring operation deletes the existing edge between N1 and N2 and adds a fake edge that connects N1 and N3. During the attack, K rewiring operations are allowed where K is related to the size of the graph.

Table 1: Dataset Summary

	Nodes	Edges	Features	Classes
CORA	2,708	5,429	1,433	7
Deezer	28,281	92,752	30,978	2
Github	37,700	289,003	4,005	2

4 DATASETS

We conduct our experiments on three widely used datasets for node classification, CORA[5], the GitHub Social Network[6], and the Deezer Europe Social Network[7]. The CORA dataset is multiclass, while the GitHub and Deezer are binary-class datasets.

- CORA is a graph dataset built with scientific publications classified into seven categories. The features are a set of selected words. The edges represent the citation relationship.
- Deezer dataset is collected from the music streaming service of Deezer. Each node is a user and the edges represent the friendship between users. The features are generated from the music genre.
- GitHub dataset contains users who have starred more than ten repositories. The edges represent the follower relationship among the selected users, and features are generated based on the user's location, employer, and repository starred.

Table 1 is a summary of the datasets. For each dataset, we randomly select 80% of the nodes as the training set and the remaining 20% as the test set.

5 PROPOSED ATTACKING ALGORITHM

Inspired by the ReWatt[4] and NIPA[9], we proposed our own attacking algorithm Node Injection via Fake Duplicate (NIFD) as described in Algorithm 1 on node classification tasks. The intuition is to create a fake node vf to introduce noise by flipping class labels and dilute the node's influence by grabbing neighbors. The perturbation to the graph is small but we believe it could effectively attack the node classification model.

The graph dataset for node classification contains several parts: the graph information G (including the nodes V and the edges E), the node features X, and the class label y. Our algorithm NIFD will poison the (V, E, X, Y) to (V', E', X', Y'), where $(V, E, X, Y) \approx (V, E', X', Y')$. NIFD will adapt to reinforcement learning to optimize each state to achieve the best attack results. For each step, NIFD takes five steps: take a node v, create a fake node vf with the same feature and different label, create a bidirectional edge between v and vf, substitute several $e[v, n]$ with $e[vf, n]$, and substitute several $e[n, v]$ with $e[n, vf]$.

NIFD can be easily proved as having a small perturbation on the graph information. If not flipping the class label, the NIFD algorithm just creates an extra hop in each path, which dilute the influence but has very little perturbation in the graph structure and information. Then it flips the class label without changing features, which creates noise in the graph. After attacking, we preserve most of the information contained in this dataset.

Since graph structure and information are similar, NIFD will force the model to learn more about the graph information, instead of just features. If the model is very good, it should keep a similar performance. Otherwise, under our NIFD attacking algorithm, we would observe some performance decrease.

The difficulty of NIFD is how to pick node v and its neighboring edges. We plan to introduce reinforcement learning to automatize this process. We will develop an RL agent to learn those two actions per state and define the rewards to be Eq. 1.

$$r_t(s_t, a_1, a_2) = \begin{cases} 1 & \text{if } acc(s_t) > acc(s_{t+1}) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

, where r_t denotes the reward of the state t, a_1 denotes the action of picking the node v, a_2 denotes the action of picking the edges $e[v, n]$ and $e[n, v]$, and $acc[s_t]$ denotes the accuracy of the state t. The RL logic is not complete and we are in the progress of developing it. The code of the RL agent is purposed to be finished by the end of October.

Algorithm 1 NIFD (Attacking Logic)

1: **procedure** NIFD(V, E, X, Y)
2: Pick a node v
3: Inject a fake node vf with $X[v] = X[vf]$ and $y[v] \neq y[vf]$
4: Create $e[v, vf]$, $e[vf, v]$
5: Break several $e[v, n]$ and create $e[vf, n]$
6: Break several $e[n, v]$ and create $e[n, vf]$
7: **end procedure**

6 EXPERIMENT RESULTS

We have implemented our GCN node classification model and a simplified NIFD (without the RL agent). We defined the logic of picking the node v as choosing the node having the most nodes and of picking neighbors n as having a randomly 50% chance. The 80% of the CORA dataset is used for training and attacking (train with the poisoned graph info), and the rest is used for testing (test with the clean graph info). For each attack, we injected 100 fake nodes and runs the same setting 10 times to reduce the randomization factors. The results are shown in Table 2. We can see on the CORA dataset our simplified NIFD algorithm results in a 3% decrease in the training set but does not have a solid influence on the test set. It might be due to the wrong assumption of the simplified NIFA (i.e. picking the node with the most edges). We believe after applied reinforcement learning, it can be further improved and realize the desired decrease in the test set.

7 NEXT STEPS

7.1 Reinforcement Learning for attacking

Currently, we select the nodes and edges to attack with very simple rules. For the next step, we plan to use reinforcement learning for this step, probably Q-learning. The state is always the intermediate step of the attacked graph with the fake node added in the previous step. The reinforcement agent will first pick a node as the sample of the fake node, and then the agent selects the set of edges for the fake node. Finally, the agent assigns a new label for the fake node. We will use Eq.1 as our reward function. We should define a constraint for small modifications on the graph and use it as the condition to terminate the learning process.

7.2 Compare with existing attacking models

There are several existing projects on graph attacking for node classification problems. We want to try reproducing their models on our datasets and compare attacking results with our models.

7.3 More Experiments

Currently, we only experiment with the CORA dataset and only use GCN as the classification model. We plan to do more experiments on the other two datasets mentioned above to see if our attacking algorithms can be widely applied to various graphs. Moreover, we plan to test our attack on other popular node classification models, for example, Node2Vec, or RNN. As long as we have successfully implemented our attacking model with reinforcement learning, we will conduct these experiments.

7.4 Wrap up

Finally, we plan to set aside one or two weeks to wrap up the project. This includes cleaning up the codes, writing the final report, and also reflecting on the project to see what are the potential improvements.

7.5 Time Schedule

Table 3 is the timeline of our plan.

8 CONCLUSION

For the time of the midterm report, we have read several state-of-the-art papers about graph convolutional networks, node classification, and attacking algorithms. Inspired by ReWatt[4] and NIPA[9], we proposed our own algorithm Node Injection via Fake Duplicate (NIFD). We implemented a simplified version of NIFD and observed that its effect is not very significant. We will further introduce reinforcement learning to develop the completed version and validate its performance. We may further modify our algorithm based on observations. If the performance is good on CORA, we will also compare it with other attacking algorithms and test it on other datasets.

PAPER LIST

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Adversarial Attacks on Node Embeddings. *CoRR* abs/1809.01093 (2018). arXiv:1809.01093 http://arxiv.org/abs/1809.01093
[2] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial Attack on Graph Structured Data. *CoRR* abs/1806.02371 (2018). arXiv:1806.02371 http://arxiv.org/abs/1806.02371

Table 2: Attacking GCN performance on CORA datasets

Train Acc After NIFD (Before)	Train Acc Change	Test Acc After NIFD (Before)	Test Acc Change
92.78% (97.13%)	**−4.35%**	85.12% (86.57%)	**−1.45%**
94.42% (95.87%)	**−1.46%**	84.75% (82.21%)	2.54%
92.6% (96.99%)	**−4.39%**	84.03% (85.3%)	**−1.27%**
93.35% (97.13%)	**−3.77%**	84.39% (84.21%)	0.18%
93.4% (96.43%)	**−3.03%**	84.94% (82.4%)	2.54%
92.87% (97.31%)	**−4.44%**	82.4% (83.48%)	**−1.09%**
93.89% (97.26%)	**−3.38%**	83.48% (85.3%)	**−1.81%**
92.6% (96.01%)	**−3.41%**	82.94% (82.4%)	0.54%
94.2% (96.29%)	**−2.1%**	85.12% (84.57%)	0.54%
93.58% (96.38%)	**−2.81%**	84.75% (84.03%)	0.73%

Table 3: Proposed milestone for next steps

Now - Nov. 6	Implement Reinforcement Learning
Nov. 6 - Nov.13	Test our attacking model on selected datasets
Nov. 13 - Nov.20	Reproduce existing models and compare with ours
Nov. 20 - Nov.28	Analyse and refine our algorithm
Nov. 28 - Dec. 4	Tentative Week
Dec. 5 - Dec. 12	Clean up code and finalize report

[3] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907 http://arxiv.org/abs/1609.02907

[4] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2021. Graph Adversarial Attack via Rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 1161–1169. https://doi.org/10.1145/3447548.3467416

[5] Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. 2004. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval* 3 (2004), 127–163.

[6] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. arXiv:1909.13021 [cs.LG]

[7] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 1325–1334.

[8] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. 2018. Data Poisoning Attack against Unsupervised Node Embedding Methods. *CoRR* abs/1810.12881 (2018). arXiv:1810.12881 http://arxiv.org/abs/1810.12881

[9] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2019. Node Injection Attacks on Graphs via Reinforcement Learning. https://doi.org/10.48550/ARXIV.1909.06543

[10] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. The Vulnerabilities of Graph Convolutional Networks: Stronger Attacks and Defensive Techniques. *CoRR* abs/1903.01610 (2019). arXiv:1903.01610 http://arxiv.org/abs/1903.01610

[11] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2019. ADMIRING: Adversarial Multi-network Mining. In *2019 IEEE International Conference on Data Mining (ICDM)*. 1522–1527. https://doi.org/10.1109/ICDM.2019.00201

[12] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM. https://doi.org/10.1145/3219819.3220078