Hard Label Black-box Node Injection Attack on GNN

Presenters: Zihao Dong, Guofeng Zhang, Jingchen Tang, Yu Zhou
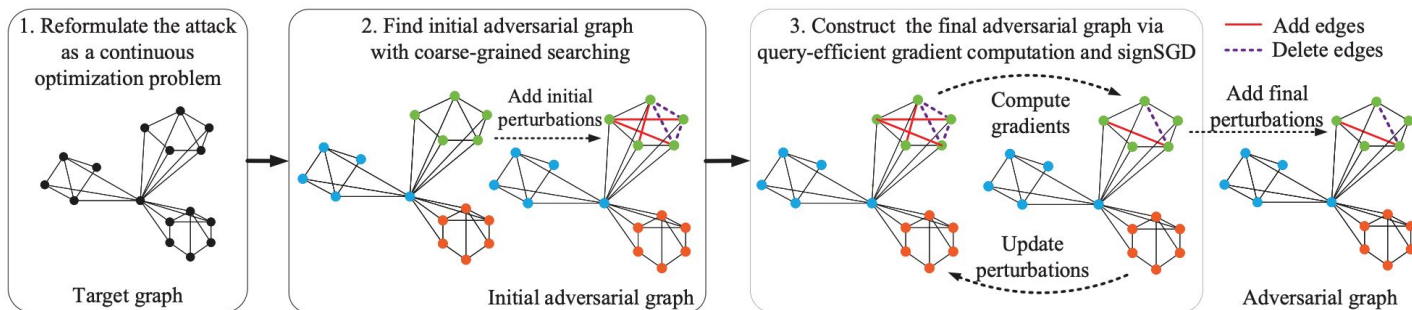
Background: Adversarial Attack

- White Box Attack:
  - The adversarial agent has access to all information of the model, including gradient, model structure, and all outputs
- Soft-Label Black Box Attack:
  - The adversarial agent only has access to the logits output of the model, and has no knowledge about the model's gradient and structure
- **Hard-Label Black Box Attack:**
  - **The adversarial agent only has access to the <u>label output</u> of the model, i.e. argmax(logits)**
- Attack on GNN:
  - Edge Attack & Node Injection Attack
  - Targeted & Non-targeted Attack

Reference Paper: Hard-Label Edge Attack

- Given a target GNN model f and a target graph G with label $y0$,
- Attacker attempts to generate an untargeted adversarial graph G'
- By perturbing the adjacency matrix A of G to be A' (Edge Perturbation)
- The predicted label of G' will be different from $y0$

1. Reformulate the attack as a continuous optimization problem
Target graph

2. Find initial adversarial graph with coarse-grained searching
Add initial perturbations
Initial adversarial graph

3. Construct the final adversarial graph via query-efficient gradient computation and signSGD
Compute gradients
Update perturbations
Add final perturbations

Add edges
Delete edges
Adversarial graph

Optimization Problem

$$\Theta^* = \arg\min_{\Theta} ||A' - A||_0,$$

$$\text{subject to} \quad A' = h(A, \Theta),$$
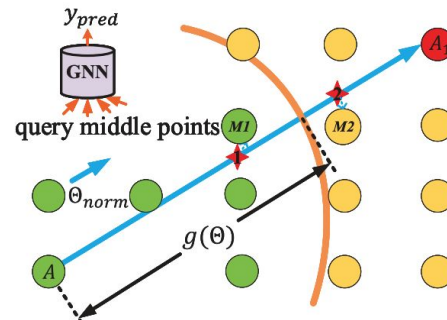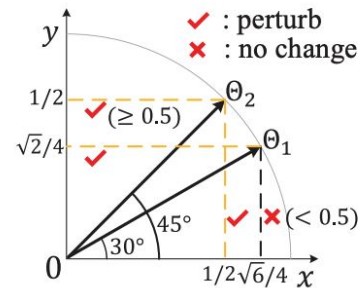$$f(A') \neq y_0,$$
$$r \leq b,$$

- Θ matrix

$$h(A, \Theta)_{ij} = h(A, \Theta)_{ji} = \begin{cases} A_{ij} & \Theta_{ij} < 0.5, j > i, \\ \neg A_{ij} & \Theta_{ij} \geq 0.5, j > i. \end{cases}$$

- $g(\Theta)$ measures the distance from the original graph A to the classification boundary
  - computed via repeatedly querying the target model

$$g(\Theta) = \arg\min_{\lambda > 0} \{ f(h(A, \lambda\Theta_{norm})) \neq y_0 \},$$

Optimization Problem (Cont.)

- $g\_hat(\Theta)$ denotes a distance vector which starts from A and ends at classification boundary at the direction of Θ with a length of $g(\Theta)$
- $p(\Theta)$ denotes the number of elements of $g\_hat(\Theta)$ that exceed 0.5
- signSGD to solve the optimization problem

$$\Theta^* = \arg\min_{\Theta} p(\Theta), \quad \text{subject to} \quad r \leq b.$$

$$\nabla p(\Theta) = \frac{1}{Q} \sum_{q=1}^{Q} sign\left(\frac{p(\Theta + \mu u_q) - p(\Theta)}{\mu} u_q\right)$$

Algorithm 1 Generating an adversarial graph for a target graph with a hard label black-box access

Input: A trained target GNN model f, a target graph A, perturbation budget b

Output: Adversarial graph A'

1: Search initial vector Θ_0 via coarse-grained searching;
2: **for** $t = 1, 2, \ldots, T$ **do**
3:      Randomly sample $u_1, \ldots, u_Q$ from a Gaussian distribution;
4:      Compute $g(\Theta_t)$, $g(\Theta_t + \mu u_q)$ for $q = 1, \ldots, Q$ via binary search;
5:      Compute $p(\Theta_t)$, $p(\Theta_t + \mu u_q)$ for $q = 1, \ldots, Q$ using Eq. (6);
6:      Estimate the gradient $\nabla p(\Theta_t)$ using Eqs. (8) and (9);
7:      Update $\Theta_{t+1} \leftarrow \Theta_t - \eta_t \nabla p(\Theta_t)$;
8: **end for**
9: Compute $A' = h(A, \Theta_T)$, $r = \|A' - A\|_0 / N(N-1)$;
10: **if** $r \leq b$ **then return** A' # succeed
11: **else return** A # failed
12: **end if**

Our Approach - General Idea

- Goal:
  - extend the above paper's method to a <u>Hard Label Black Box Node Injection Attack</u>.
  - To the best of our knowledge, this will be the very first <u>Hard Label Black Box Node Injection Attack on GNN</u>
- Method:
  - "Leave room for new nodes", try inject 1, 2,…, n nodes iteratively,
  - until success or exhaust our budget

- Note:
  - we add the iterative approach to some of the datasets because of the randomness in some of our approaches
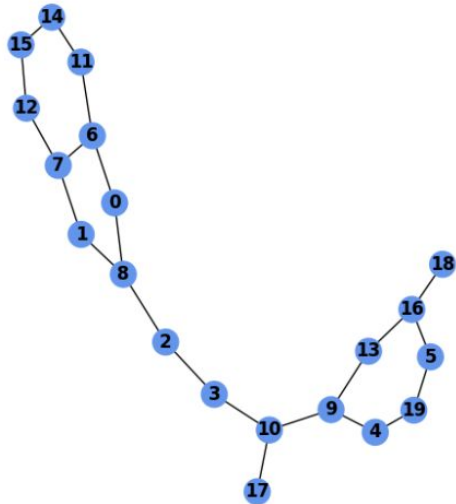
Single Node Injection Attack

Inspired by Single Pixel Injection Attacks on computer vision models (Su et al, 2017), we first attempt a single node injection attack on Graph Classification GNN models.

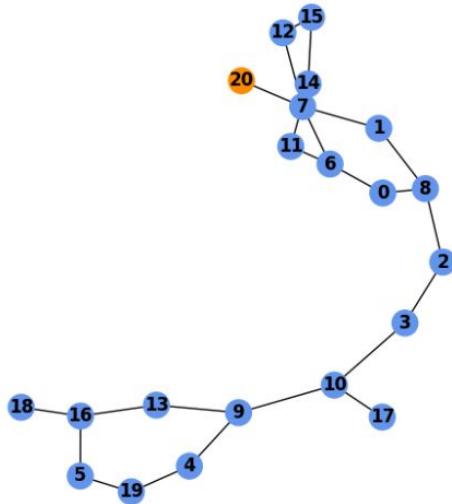We also allow edge perturbation on the whole graph to maximize attack success rate.

Reference: https://arxiv.org/pdf/1710.08864.pdf
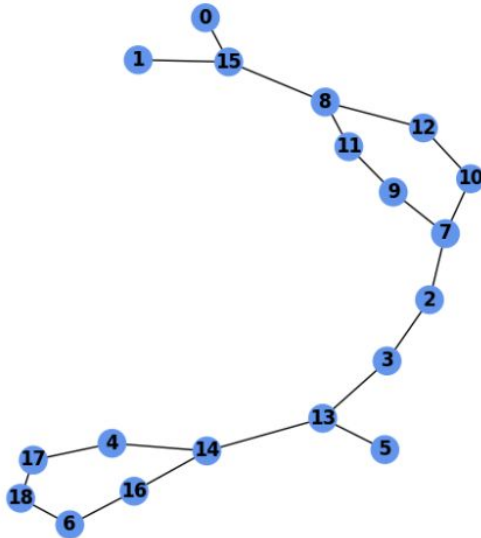
Single Node Injection Attack

Nodes 4 and 5 are connected and the connection between nodes 7 and 11 was removed

Node 17 and the injected node 19 are now connected

Single Node Injection Attack

Before Node Injection

After Node Injection
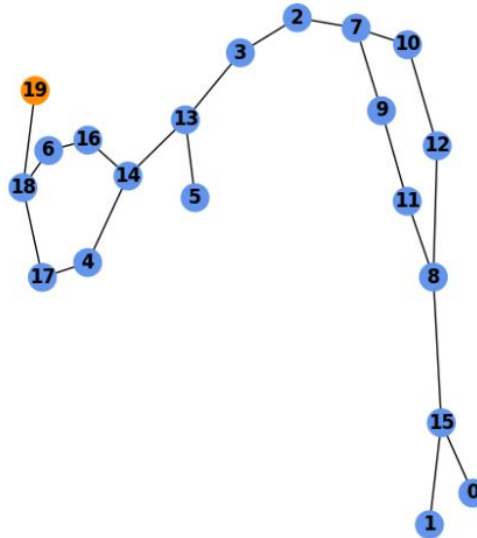
After Perturbation
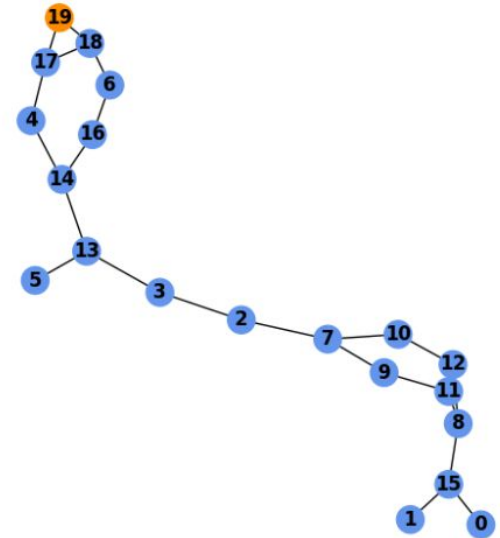
Details of Single Node Injection Attack

We use two different feature initialization methods for the injected node:

1. Mean (average of all nodes)

2. Random (random value in the range of all node values)

We also consider two methods for connection the injected node to the graph:

1. Random Node (connect to a random node)

2. Mode Node (connect to the node with highest rank)

NCl1 Dataset

Chemical Substance Graphs

feature initialization	Connection method	Success Rate (SR)	Success Count	Average Perturbation (AP)	Average Attack Time (AT)
mean	Mode node	80.00%	232	4.3707	56.6134
mean	Random node	81.3559%	240	5.1250	37.4665
random	Mode node	81.7568%	242	5.8595	37.7517
random	**Random node**	**83.5570%**	**249**	**5.6113**	**39.5264**

Reference paper best SR: 78%
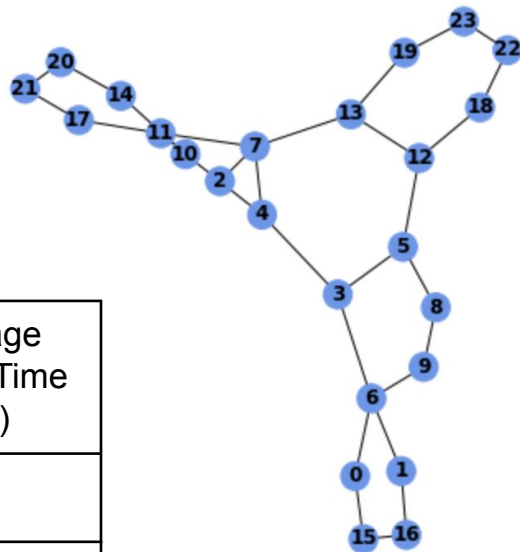
COIL-DEL Dataset

Image Features as Graph

feature initialization	Connection method	Success Rate (SR)	Success Count	Average Perturbation (AP)	Average Attack Time (AT)
mean	Mode node	97.4522%	306	3.509	26.32
mean	Random node	97.1246%	304	3.514	42.91
random	Mode node	97.3510%	294	3.235	29.22
random	**Random node**	**97.7199%**	300	**3.632**	**26.99**

Reference paper best SR: 92%

IMDB-BINARY Dataset

Social Network Graphs

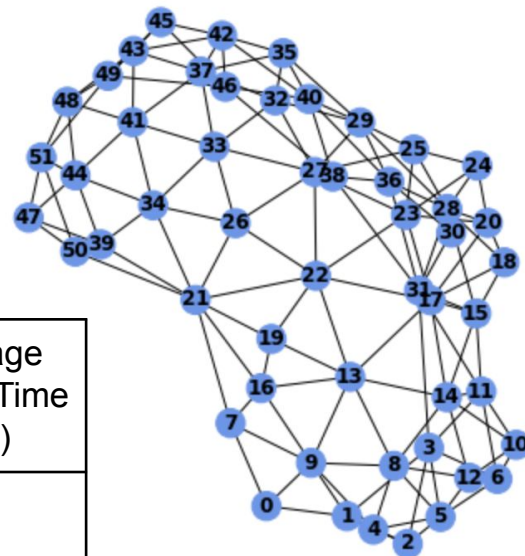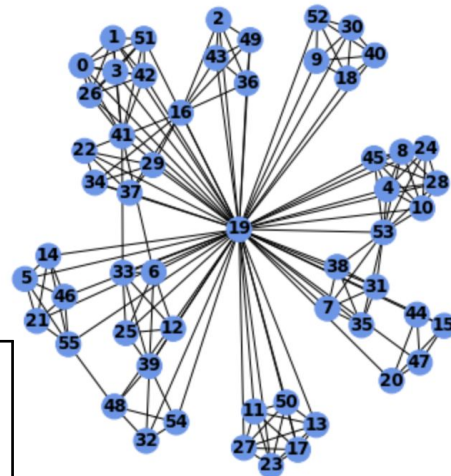| feature initialization	Connection method	Success Rate (SR)	Success Count	Average Perturbation (AP)	Average Attack Time (AT)
mean	Mode node	98.6111%	71	30.0563	30.3956
mean	Random node	98.6111%	71	30.9155	32.2228
random	Mode node	97.2222%	70	31.0571	30.2109
random	**Random node**	**98.6111%**	**71**	**30.6338**	**32.4291**

Reference paper best SR: 89%

Limitations of this method

1. In practice, it may not be feasible to perturbe edges in the original graph. For example in social networks, the attacker is only able to modify edges (follow relationships) on the injected node (new user created by attacker), but not the relationships of existing users.

2. For large complex graphs and datasets with small number of labels, injecting only a single node has little effect on the whole graph, so we end up having to perturbe a very large number of edges
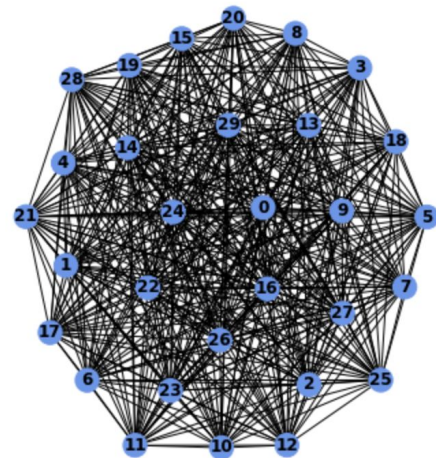
Our Approach - Detail

When trying to inject k nodes:
- Initialize k node features, and simply put the new nodes into the node list of the graph
    - Initialization method may vary across datasets (will introduce later in experiments)
- Initialize the initial connection, i.e. connect the new nodes to some original node, otherwise they have very small influence on the graph
    - Connection Initialization method can be: connect to node with highest degree, or randomly choose a node
- "Cut" the Adversarial Perturbation Matrix (Θ) to only the entries involving the new nodes:

$$\Theta \in R^{k*n}$$

- Rest of the algorithm will be the same as the original paper (i.e. we perform direction search and binary search along the chosen direction like the original paper to find the decision boundary)

Performance of Multi-Node Injection

Some Key Challenges:

- Initialization of node feature
- How to enforce the injected node can keep connected to the graph after perturbing the edges?

In this sets of experiments we will explore the effect of different feature initialization and connection initialization methods

Because of limit of computation resource, we only had chance to finish experiments using GIN

COIL-DEL Dataset

Node Feature: (x, y) coordinates, where x and y are integers but represented as floats (1.000, 5.000, etc)

No two nodes have same coordinates in this datasets, therefore our initialization is a Gaussian Dist whose mean and std are from original node features (because of randomness, non iterative)

Connection Init: random vs mode (node with highest degree)

method	budget	SR%	success	Injected %	No need	Pred change	Perturb edge
mode	0.1	50.35	94	99.16	102	51	13.51
mode	0.15	63.89	112	100	102	72	11.57
random	0.1	48.68	100.2	100	102	40	15.51
random	0.15	61.46	122	100	102	55	17.38

IMDB-BINARY

The node features are just [1.]'s, so node feature initialization is trivial; because initialization is deterministic, we will try to reduce perturbation by employing the iterative approach

method	budget	SR%	success	Injected %	No need	Pred change	Perturb edge
mode	0.1	27.63	7	100	24	14	167.1429
mode	0.15	53.95	20	100	24	21	104.4
random	0.1	27.63	6	100	24	15	191.67
random	0.15	51.32	20	100	24	19	111.58

NCl1

Node Feature: 37-dim 1-hot encoding of atom type

Node Feature Init: count each atom type, and generate new atom following a probability based on the counts (i.e. the prob of generating atom x will be p = count(x)/#atom), non iterative

method	budget	SR	success	Injected %	No need	Pred change	Perturb edge
mode	0.1	47.43	101	99.17	104	44.6	128.17
mode	0.15	63.19	140	98.82	104	54	165.06
random	0.1	41.04	101	100	104	25	143.63
random	0.15	54.40	140	100	104	27	185.53

For IMDB and NCl1

- Notice the perturbation count in these 2 datasets
- Our method needs further improvement on these 2 datasets

Feature Manipulation of injected nodes

Algorithm:

1. Find the node that has most connections in the original graph and its feature.
2. Find the feature in current graph that has the largest L1 distance to the feature of the node in step 1
3. Slightly perturb the feature in step 2 such that it is not the same to any feature in graph (or to other injected nodes)
4. Connect injected node to the node find in step 1
5. Iterate to find best perturbations (Limited to injected nodes)

Performance of feature manipulation

Experiments are only currently conducted on COIL dataset

	Success rate	Edge perturbation	Attack time	Query count
10%	48.61	10.84	9.06	544.2
10% (I)	48.95	8.25	11.1	580.8
15%	63.89	10.65	9.56	644.0
15% (I)	64.24	8.65	13.2	786.6
20%	71.18	11.88	9.74	632.0
20% (I)	71.53	9.39	15.7	822.8

Demo

Future works and discussions

1. Investigate more concrete optimization process of finding injected feature of nodes
2. Find better connection initialization that is less noticeable. (more than one edge/node?)
3. Create initial search as in original paper to improve the success rate and speed up the attach
4. Improve our methods on datasets like IMDB and NCI1, which have small number of prediction classes