Targeted Edge Perturbations on GNNs: Exploring Greedy, Heuristic, and Gradient-Driven Approaches

DYNAMO — Dynamic Networks: Analysis and Modeling

Will Corcoran, Wyatt Hamabe, Niyati Mummidivarapu

Faculty Advisor: Prof. Ambuj K. Singh, Graduate Mentor: Danish Ebadulla

UC SANTA BARBARA Early Research Scholars Program

Motivation

Graphs form the backbone of essential technologies from social networks to navigation systems. Graph Neural Networks (GNNs) are crucial for improving and securing these applications. While GNNs are "black box" algorithms, our focus is analyzing the effect of perturbations on GNN's performance and output.

The **Minimum Edge Set Perturbation Problem** aims to discover a set of edges exhibiting a minimal change in output. Congruently, we aim to identify classes of edges that cause vulnerability or structural strength.

The **Lipschitz Constant,** *L*, bounds a model's output (embeddings) relative to a change in input.

$$||X^{(L)} - X_p^{(L)}||_F \leq \sqrt{d}L||\Delta - \Delta_p||_2 \prod_{l=1}^{L} ||\theta^{(l)}||_2$$

L: number of layers, ***X:*** GNN embeddings, ***Δ:*** normalized adjacency matrix, ***d***: input dimensionality, ***θ***: model layer weights [Singh et al. (2024)]

Targeted Edge Perturbation Methods

Classification Density: create an induced subgraph of each class. Choose edges from pairs of nodes in connected components above certain density. *Small accuracy change*.

General Greedy: until budget is reached, iteratively add an edge. If it causes a change in accuracy, remove it, otherwise, continue. *No accuracy change*.

- **Classification:** edges with same label and initial prediction.
- **Degree Threshold:** difference of degree for nodes in edge is within a specified threshold.
- **Metattack-Based:** edges added in previous Metattack. Intended to prepare covert adversarial attack.

Perturbation Rate: edges added relative to the edges in the original graph.

Classification Density

```
ClassDensityPtb():
  valEdges ← []
  d ← value ∈ [0...1]
  ptbRate ← value ≥ 0
  budget ← ptbRate × |G.edges|

  C_1, ..., C_n ←nodes with class 1, ..., n
  for C_i ∈ C_1, ..., C_n do:
    G_i ← IndSubg(C_i)
    for cc ∈ ConComp(G_i) do:
      if density(cc) ≥ d then
        Clique ← MakeClique(cc)
        append(Clique.edges, valEdges)
      end if
    end for
  end for

  while under budget do
    edge ← randSel(valEdges)
    if edge ∉ G then
      addEdge(G, edge)
    end if
  end while
```

General Greedy

```
GeneralGreedyPtb(heuristic):
  valEdges ← []
  ptbRate ← value ≥ 0
  budget ← ptbRate × |G.edges|

  for u ∈ G.nodes do
    for v ∈ G.nodes do
      if u ≠ v and heuristic(u, v) then
        append({u, v}, valEdges)
      end if
    end for
  end for

  while under budget do
    edge ← randSel(valEdges)
    if edge ∉ G then
      addEdge(G, edge)
      if accuracy changes then
        removeEdge(G, edge)
      end if
    end if
  end while
```

Classification Greedy

```
ClassHeuristic(i, j):
  return label[i] == label[j] and pred[i] == pred[j]
      and label[i] == pred[i]

ClassGreedyPtb():
  GeneralGreedyPtb(ClassHeuristic)
```

Degree Threshold Greedy

```
DegreeHeuristic(i, j, threshold):
  return ClassHeuristic(i, j) and
      abs(degree(i) − degree(j)) <= threshold

DegreeGreedyPtb():
  GeneralGreedyPtb(DegreeHeuristic)
```

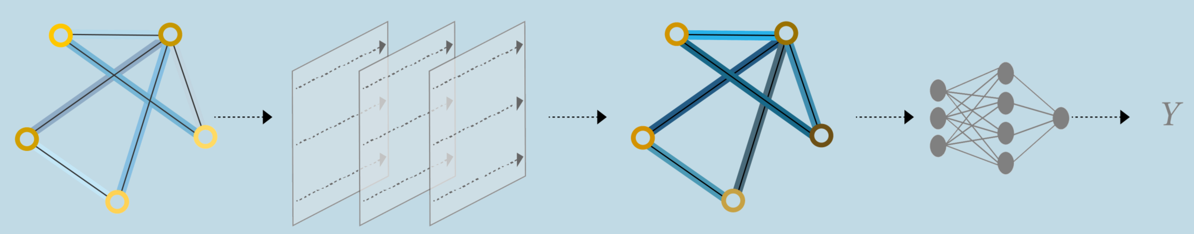
Metattack-Based Greedy

```
MetattackHeuristic(i, j, metattack_edges):
  return (i, j) ∈ metattack_edges

MetattackGreedyPtb():
  metattack_edges = Metattack(G).adj − G.adj
  GeneralGreedyPtb(MetattackHeuristic)
```

Graph Neural Networks

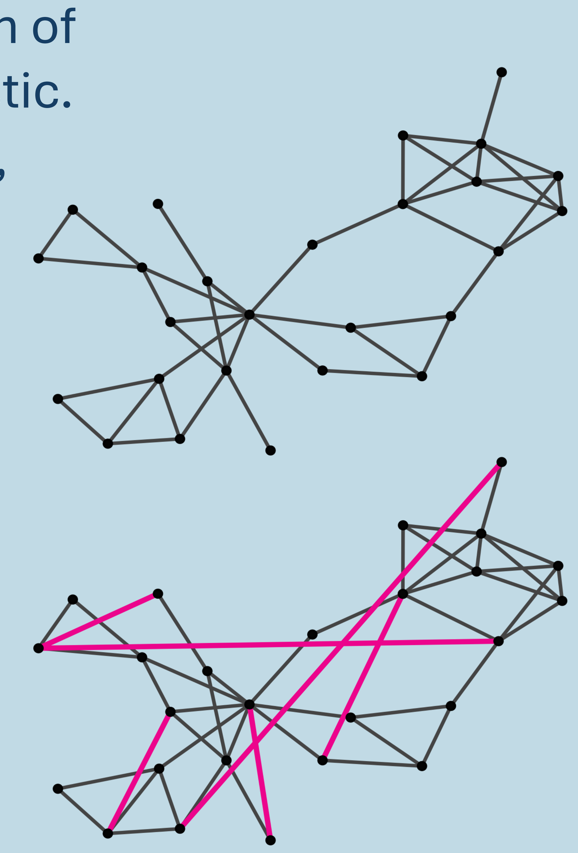
GNNs: branch of neural network architectures dedicated to learning information given graph data.
Utilization: node classification, edge classification, link prediction, nearest neighbors (similarity)
Common Architectures: GCN, GSAINT, GSAGE, GCN-JACCARD, GAT, ChebNet



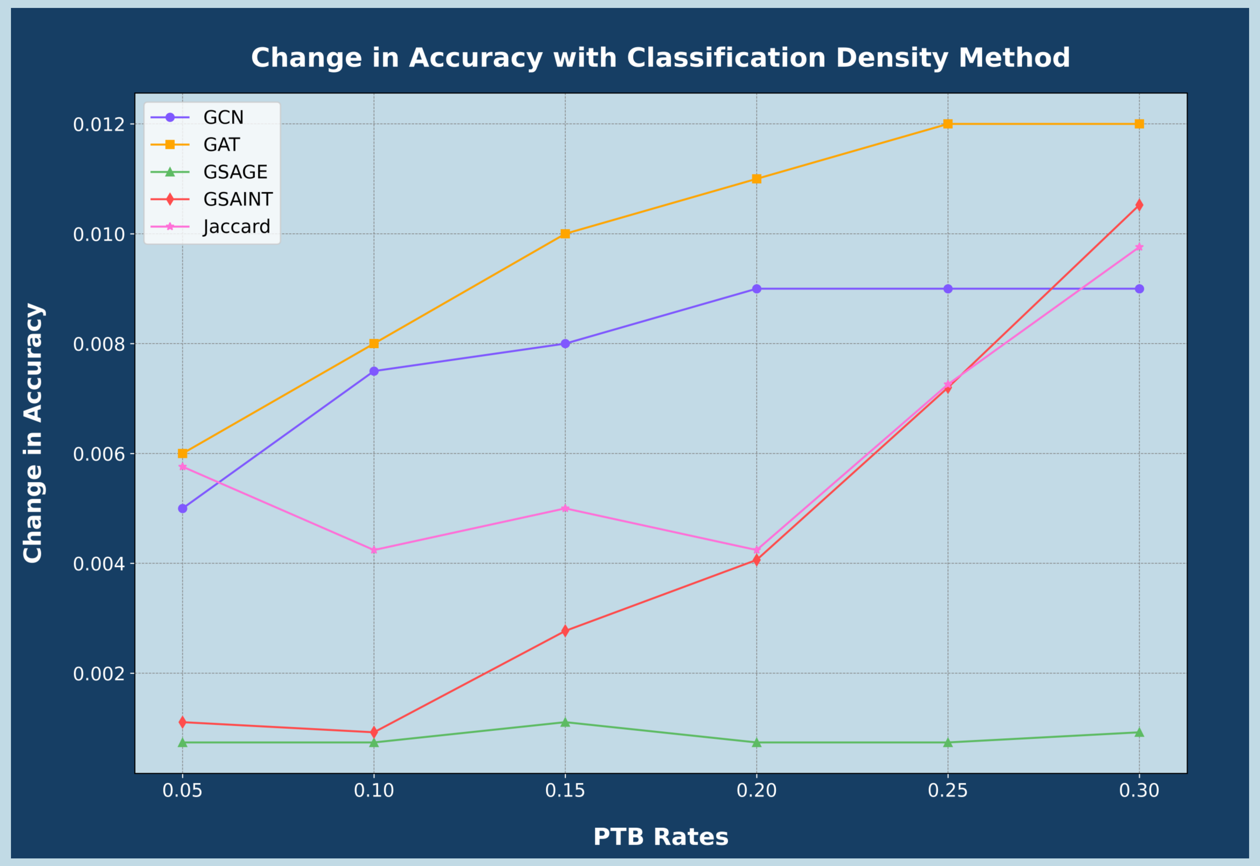
GNN transformations from left to right: input graph → inner architecture → transformed graph → classification processing → output.
Visual Credit: Google Research, 2021

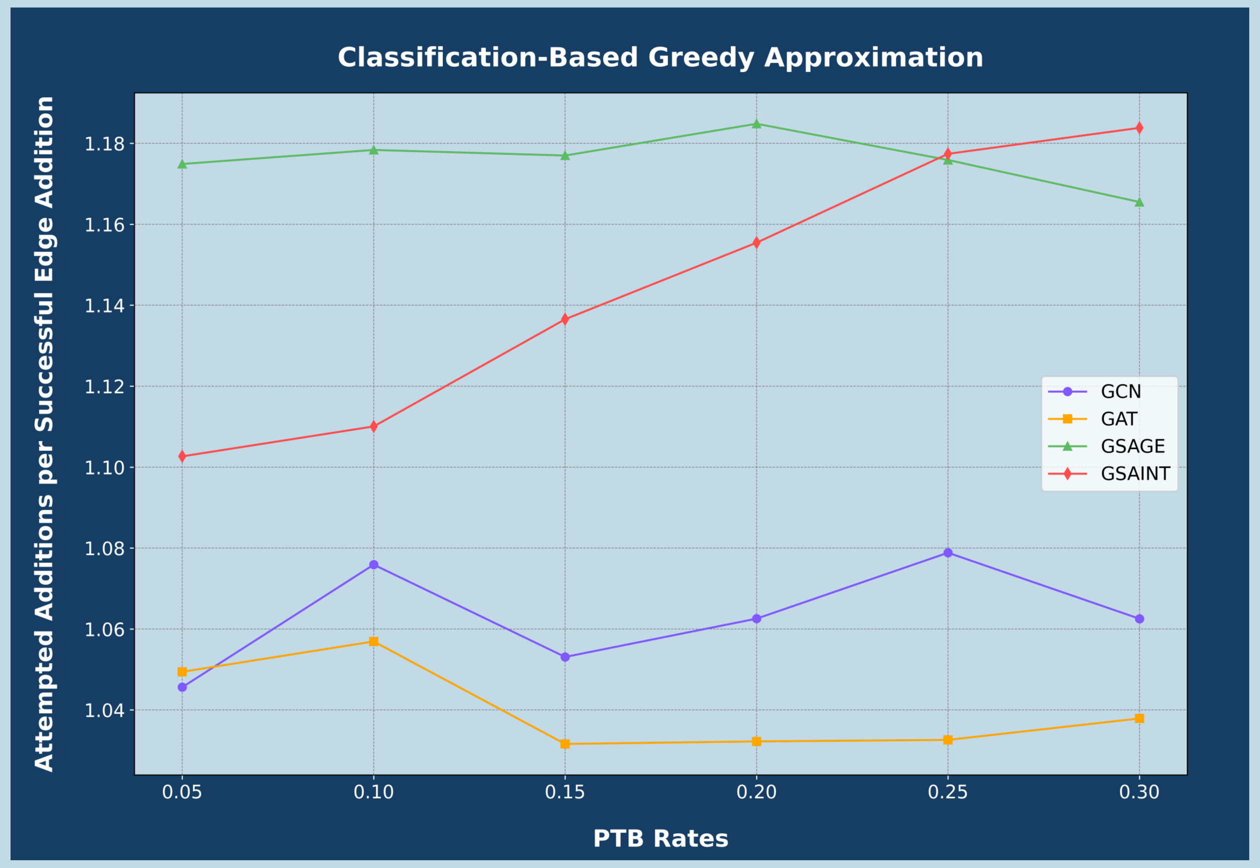
Perturbations

Edge Perturbation: addition of edges based on some heuristic.
Example heuristics: degree, homophily, k-nearest neighbors, classification.
Miss Rate: how often added edges change the accuracy (specific to greedy approx.)
Visual (right):
Top: second largest connected component (26 nodes, 43 edges) in CORA.
Bottom: 15% perturbation.



Results



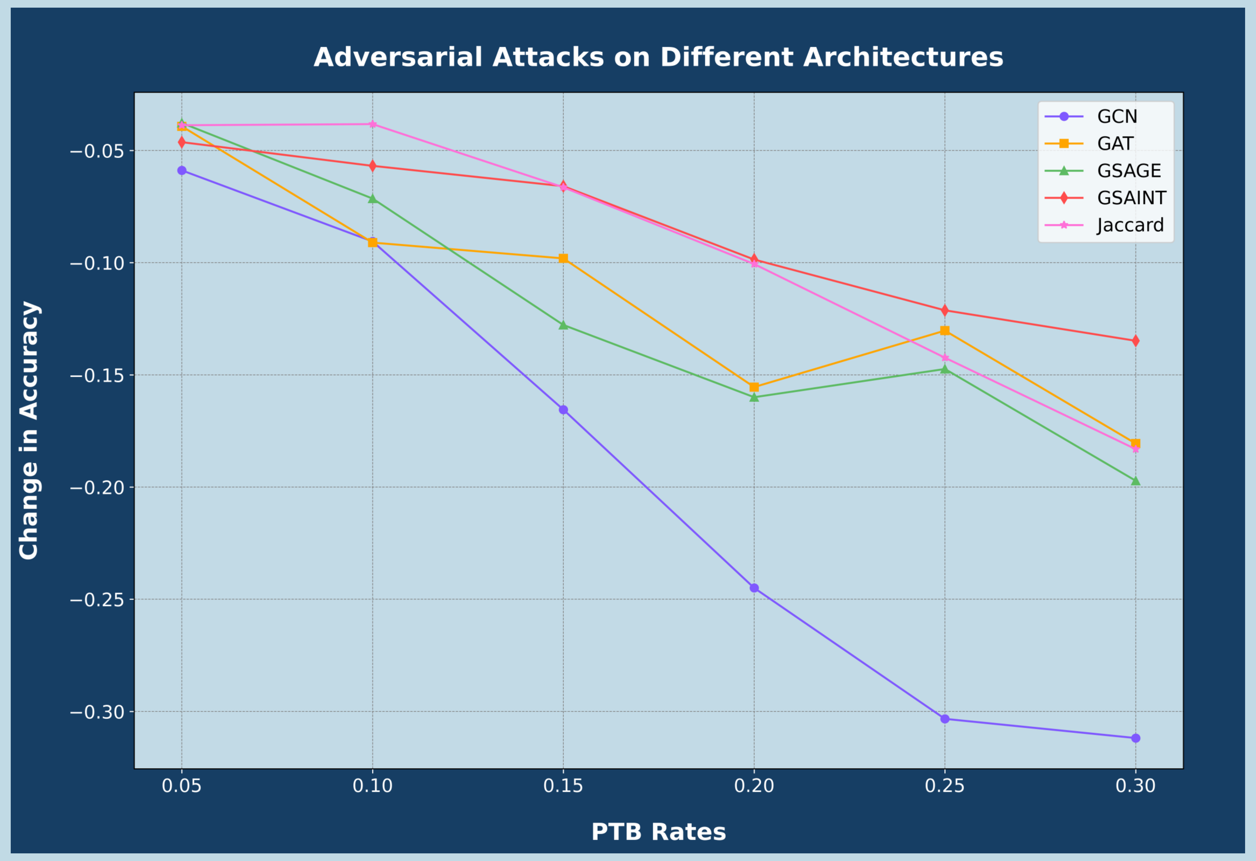


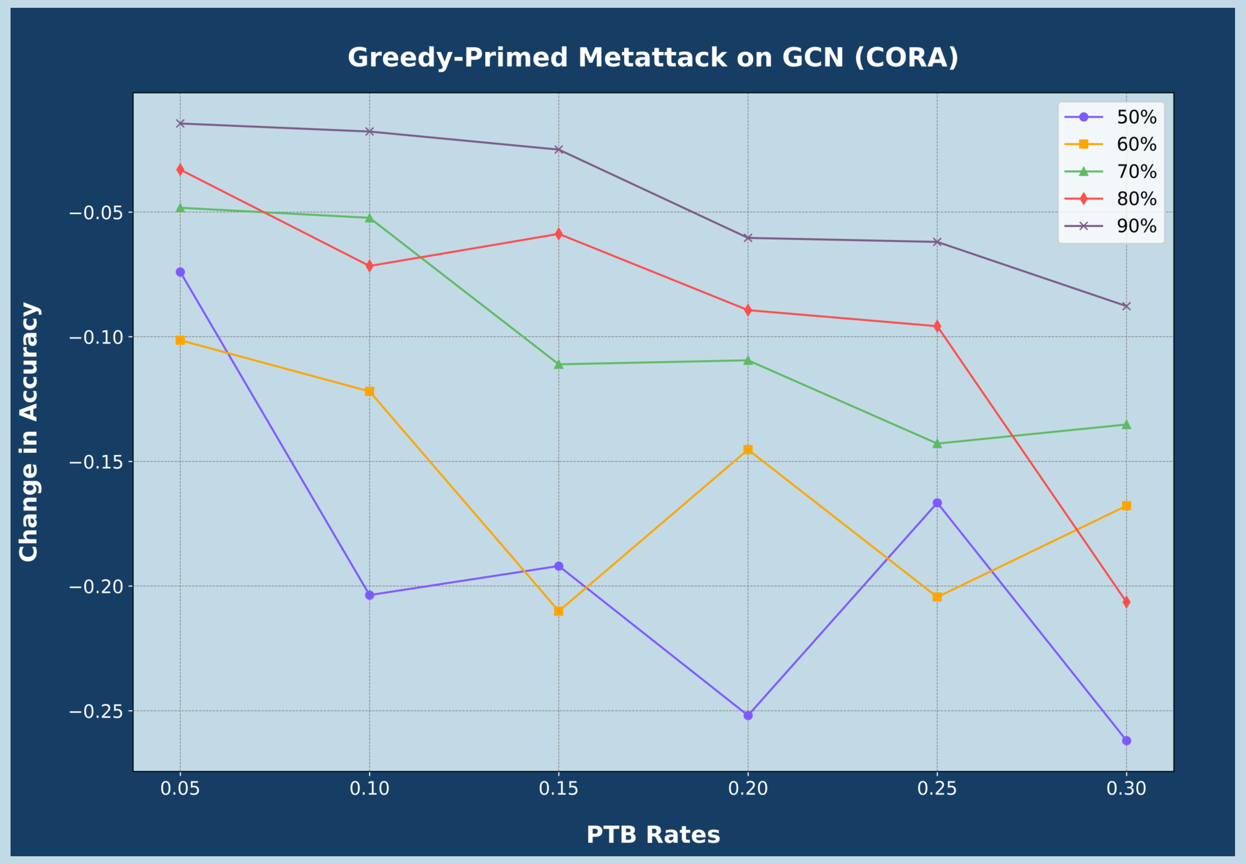
Top: Even with high (30%) PTB Rates, ClassDensityPtb(), where d = 0.05, leads to a maximum 1.2% change in acc.

Top: ClassGreedyPtb() shows a near linear relation (with few misses) between attempts per edge and PTB Rates.

Bottom: Metattack implemented with DeepRobust showing large change in acc. (up to 30%) for 30% PTB rate.

Bottom: Greedy-Primed Metattack can be undetected for >50% of PTB Period and achieve similar accuracy changes.





Future Work

MESP is an NP-hard problem, hence, approximations are necessary. Our work opens these questions:

- Does there exist a perturbation rate where *greedy approximation* misses fail to be linear in change?
- Is there a heuristic that minimizes the number of misses per edge addition?
- Is miss rate a valid measure of GNN robustness?
- Can these edge perturbations be used to create a domino-style adversarial attack?
- Does there exist an approximation method running in a lower time complexity than *O(pmnt)*?
- Does there exist an architecture or data set where the *greedy approximation* method performs without reason (i.e. in a non-linear fashion)?

Adversarial Attacks

Adversarial Attack: a manipulation of input graph such that the ML model makes incorrect decisions.
Meta Gradient Attack (Metattack): Gradients indicate the direction and rate of change in the model's output with respect to its input.

- *Initial Perturbation*: introduces random change to input data.
- *Gradient Calculation*: calculates the gradient of the model's loss with respect to the new input.
- *Iterative Optimization*: the attack makes a small change to input and recalculates the gradient, repeated over many iterations.

Greedy-Primed Metattack: run *Metattack-Based Greedy* for budget *b* followed by *Metattack* for *1-b*.