# Trustworthy AI for Graphs

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

# Readings

- Readings are updated on the website (syllabus page)
- **PyG Lecture Readings:**
  - Documentation
  - GitHub
- **Lecture 15 readings:**
  - Trustworthy Graph Neural Networks
  - GraphFramEx Evaluation

# Outline of Today’s Lecture

**1. Intro to Trustworthy GNN**

**2. Adversarial Attacks and Robustness of GNNs**

**3. Explainability for GNNs**

# Outline of Today's Lecture

**1. Intro to Trustworthy GNN**

**2. Adversarial Attacks and Robustness of GNNs**

**3. Explainability for GNNs**

# Trustworthy Graph Learning

- **Trustworthy** AI/GNN includes many components
  - Explainability, fairness, robustness, privacy, …
  - Algorithms to tackle combination of these aspects
- **Challenges**
  - Role of graph topology is previously unexplored in these problems
  - Comprehensive quantiative evaluation

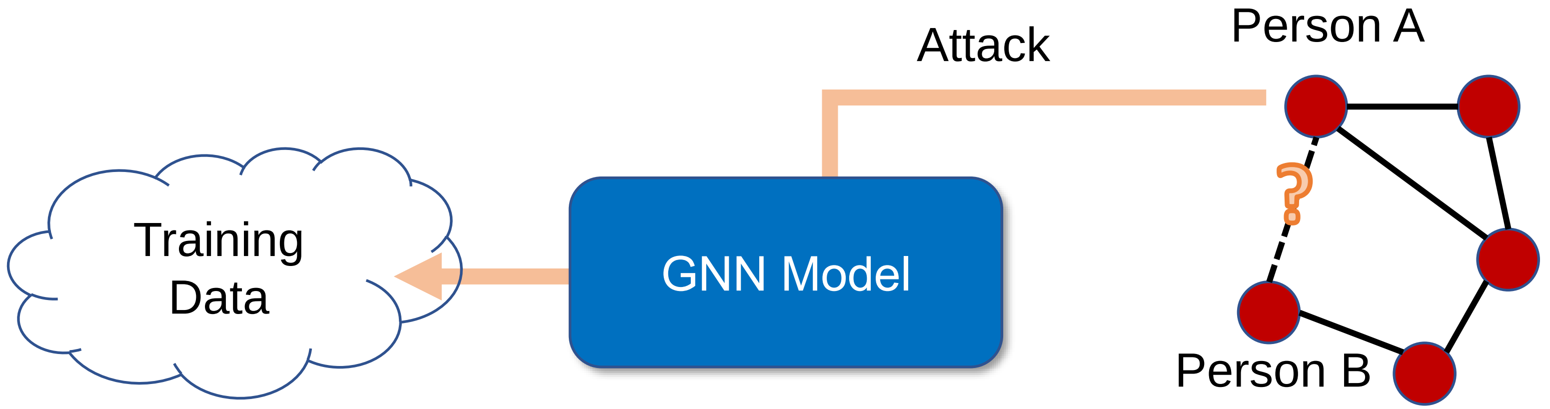
# Aspects of Trustworthy GNNs

- **Robustness** (part 2)
- **Explainability** (part 3)
- Privacy
- Fairness
- Accountability
- Environmental well-being
- Others

**How does each aspect play a role in gaining trust from users of machine learning models?**

# Privacy of GNNs

- Prevent private data within GNNs from being leaked
  - Training data
  - Model parameters
- Ex) Query social relations between two people by attacking the GNN model



# Privacy of GNNs: Attack

- **Model extraction attacks**
  - Steal architecture and parameters of a GNN model.
- **Membership inference attacks**
  - Infer whether a node/link/graph belongs to the training set of a GNN model.
- **Model inversion attacks**
  - Infer a GNN model's inputs from their corresponding outputs.
- **Other privacy attacks**

# Privacy of GNNs: Privacy-Preserving Techniques

- **Federated Learning**
  - Calculate gradients on **individuals** using their own data
  - Aggregate parameters (e.g. gradients/model weights) on the server
- **Differential Privacy**
  - Add noise to data, such that
    - **Meaningless** when viewed individually
    - But approximate the analytics results when **aggregated**
  - Noise can cancel out when performing the mean aggregation operation in GNN layers.
- **Insusceptible Training**

Original task loss

Attack function: try to distinguish the private labels

$$\min_\theta \sum_{v_i \in \mathcal{V}} \boxed{\mathcal{L}_Y\big(f_\theta(v_i)\big)} + \lambda \boxed{\mathcal{L}_A\big(\boxed{\mathcal{F}_A(v_i)}\big)}$$

Privacy-preserving loss: e.g. make the attack function's output probability close to 0.5 for the private labels

- **Security Computation**
  - More related to system/hardware

# Fairness of GNNs

- **Goal**: exclude prejudice or favoritism towards an individual or a group.
  - For example, in a bank's transaction network, the model should not learn to make predictions of loans based on gender, race or other protected characteristics.
- **Prevent Bias & Discrimination**
  - **Bias**: unfair operation in data collection, sampling, measurements, …
  - **Discrimination**: incorporation of intentional or unintentional human prejudices and stereotyping in deep learning models

# Fairness of GNNs: Methods

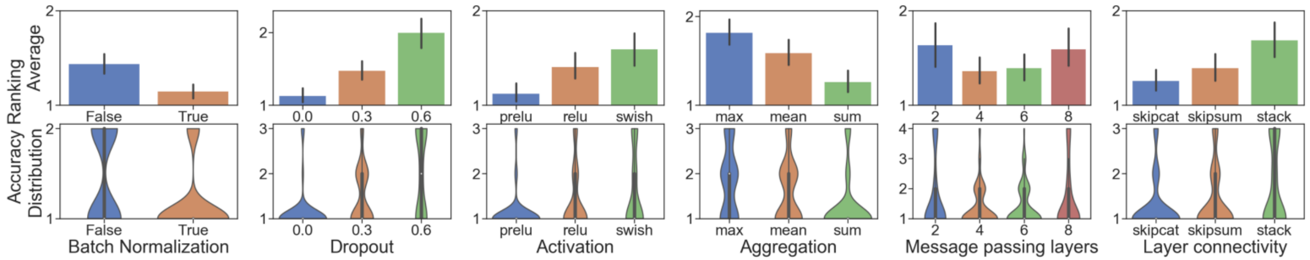
- **Fair representation learning methods**
  - Learn representations, from which one cannot infer sensitive attributes.
  - A common technique is **adversarial training**
- **Fair prediction enhancement methods**
  - **Data augmentation**
    - Perturbation of protected features
  - **Fair graph**
    - Modify graph structures (e.g. drop edges that may induce bias)
  - **Regularisation**
    - Ex) any two individuals who are similar should receive similar algorithmic outcome

$$\| \boxed{\mathbf{Y}[i,:]} - \mathbf{Y}[j,:] \|_F^2 \boxed{\mathbf{S}[i,j]} \leq \delta$$

Predictions of node $i$ Similarity between node $i$ and $j$

# Accountability of GNNs

- **Accountability** refers to the extent to which people can trust GNNs by **assessing** a complex GNN system
- **Benchmarking**: detect violation of utility
  - Architecture design, Model training, Model validation

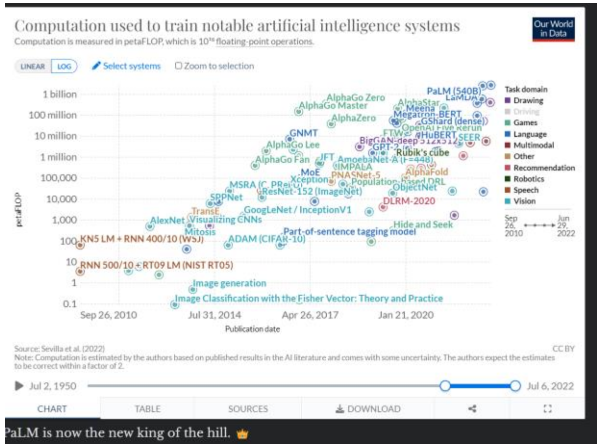



- **Security evaluation**: detect violation of security
  - Data integrity verification
  - Procedure integrity verification

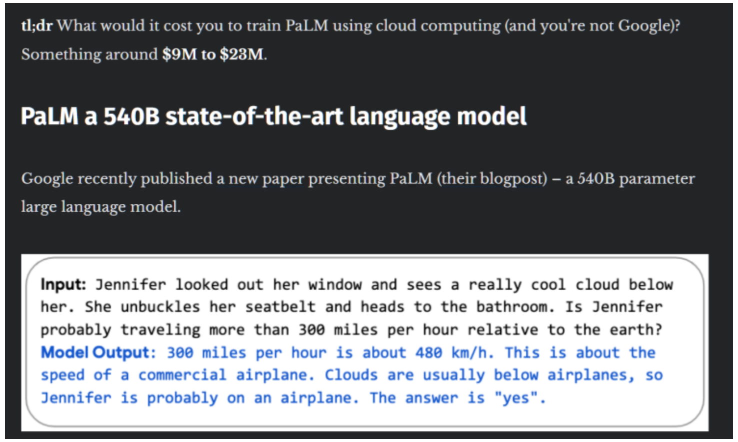
# Environmental Well-Being of GNNs

- GNNs should comform to the fundamential values of the society in which they are deployed
  - Large-scale graph datasets → GNN execution efficiency
  - Deeper or more complex architectures → GNN's deployment on edge devices
  - The unique characteristics of graph data → specially designed software and hardware

**Compute Clusters**
Credit: Imaginima/E+/gettyimages




**Large model training involves $10^{24}$ flops**
https://blog.heim.xyz/palm-training-cost/




**540 B parameter pretrained model**
https://blog.heim.xyz/palm-training-cost/

# Environmental Well-Being of GNNs: Methods

- **Scalable GNN**
  - We've already talked about it in Lecture 6
- **Model compression**
  - Knowledge distillation
  - Model pruning
  - Reducing parameters
  - Model quantisation
- **Efficient frameworks and accelerators**
  - Software: PyTorch Geometric, **Efficient AutoML**
  - Hardware: EnGN, HyGCN

# Outline of Today's Lecture

1. Trustworthy AI

**2. Adversarial Attacks and Robustness of GNNs**

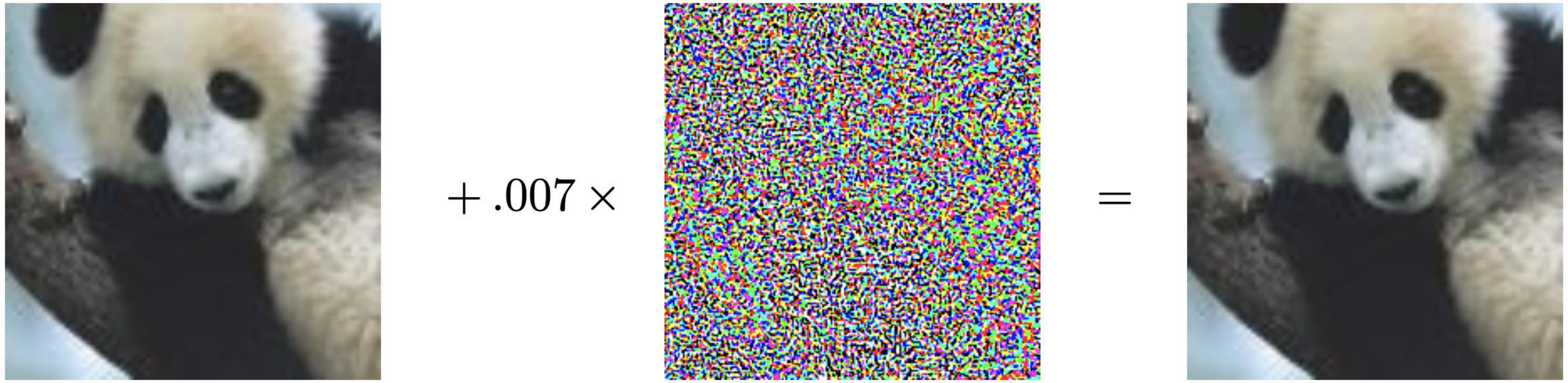
3. Explainability for GNNs

# Deep Learning Performance

- Recent years have seen **impressive performance of deep learning models in a variety of applications.**
  - In computer vision, **deep convolutional networks** have achieved human-level performance on ImageNet (image category classification)
- **Are these models ready to be deployed in real world?**

# Adversarial Examples

- Deep convolutional neural networks are vulnerable to **adversarial attacks:**
  - Imperceptible noise changes the prediction.




"Panda" 57.7% confidence | Carefully calculated noise | "Gibbon" 93.3% confidence

Goodfellow, I., Shlens, J., & Szegedy, C.. (2014). Explaining and Harnessing Adversarial Examples.

- Adversarial examples are also reported in natural language processing [Jia & Liang et al. EMNLP 2017] and audio processing [Carlini et al. 2018] domains.

# Implication of Adversarial Examples

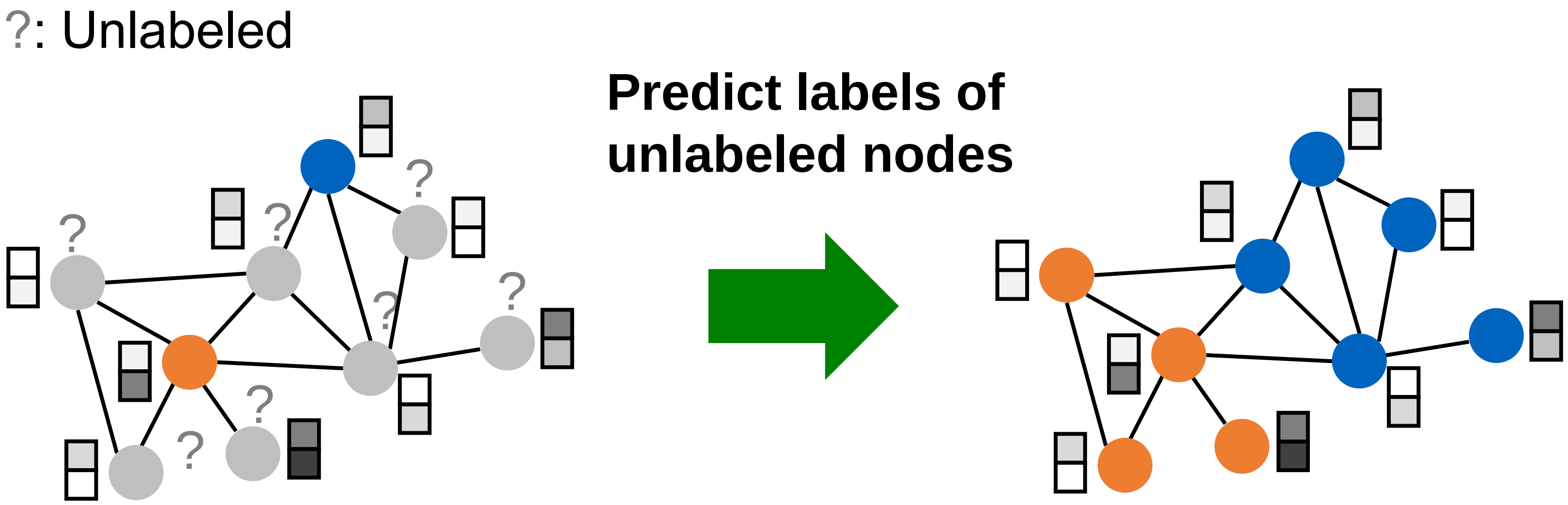
- **The existence of adversarial examples prevents the reliable deployment of deep learning models to the real world.**
  - Adversaries may try to actively hack the deep learning models.
  - The model performance can become much worse than we expect.
- **Deep learning models are often not robust.**
  - It is an active area of research to make these models robust against adversarial examples

# Robustness of GNNs

- **This lecture: How about GNNs? Are they robust to adversarial examples?**
- **Premise**: Common applications of GNNs involve **public platforms** and **monetary interests**.
  - Recommender systems
  - Social networks
  - E-commerce platforms
- Adversaries **have the incentive to** manipulate input graphs and hack GNNs' predictions.

# Setting to Study GNNs' Robustness

- To study the robustness of GNNs, we specifically consider the following setting:
  - **Task**: Semi-supervised node classification
  - **Model**: GCN [Kipf & Welling ICLR 2017]

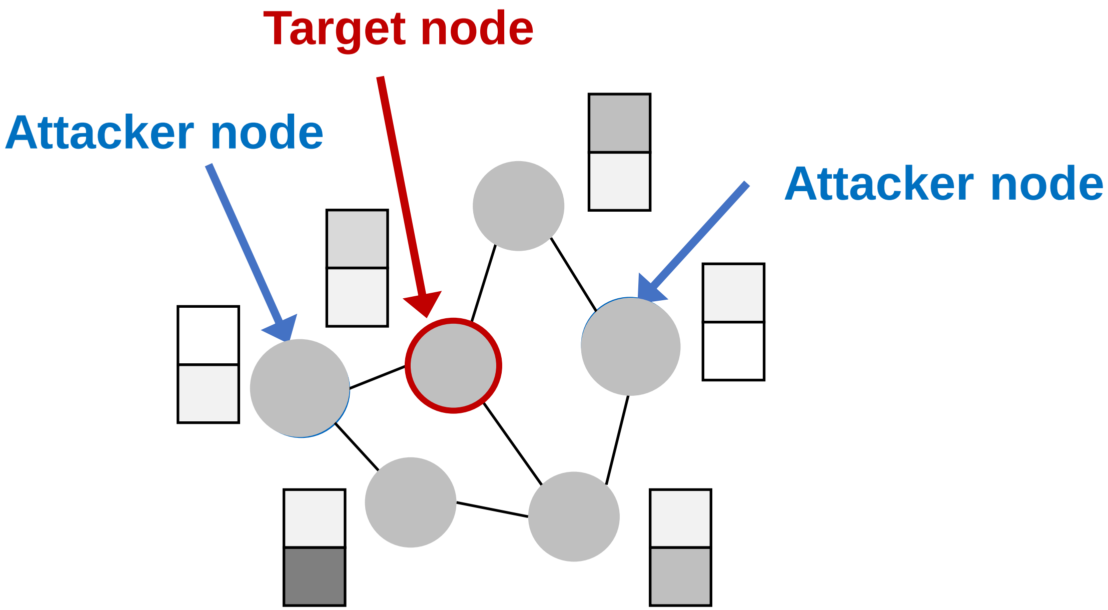


# Roadmap

- We first describe several real-world **adversarial attack possibilities**.
- We then review the GCN model that we are going to attack (**knowing the opponent**).
- We mathematically **formalize the attack problem as an optimization problem.**
- **We empirically see how vulnerable GCN's prediction is to the adversarial attack.**

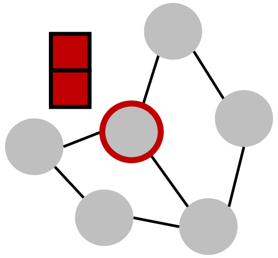
# Attack Possibilities

- What are the attack possibilities in real world?
  - **Target node $t \in V$:** node whose label prediction we want to change
  - **Attacker nodes $S \subset V$:** nodes the attacker can modify

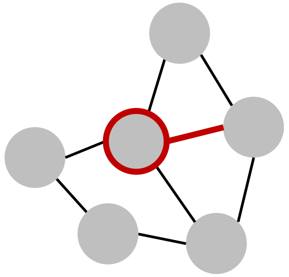


# Attack Possibilities: Direct Attack

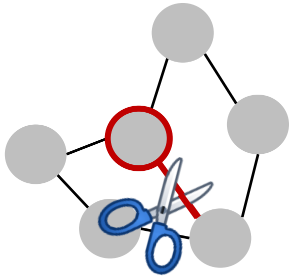
- **Direct Attack: Attacker node is the target node:** $\boldsymbol{S} = \{\boldsymbol{t}\}$



- Modify **target** node feature
  - Change website content



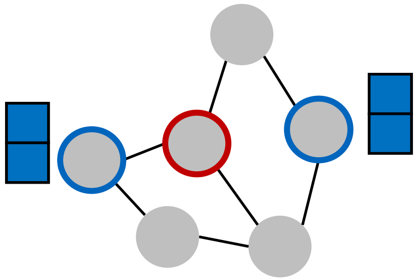
- Add connections to **target**
  - Buy/likes/follows



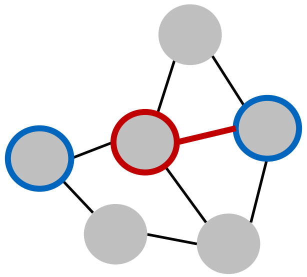
- Remove connections from **target**
  - Unfollow users

# Attack Possibilities: Indirect Attack

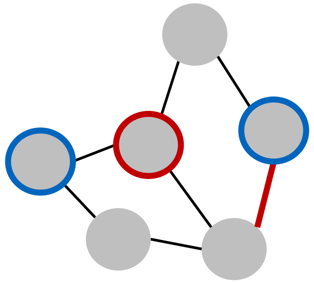
- **Indirect Attack: The target node is not in the attacker nodes**: $\boldsymbol{t} \notin \boldsymbol{S}$



- Modify **attacker** node features
  - Ex) Hijack friends of targets



- Add connections to **attackers**
  - Ex) Create a link, link farm

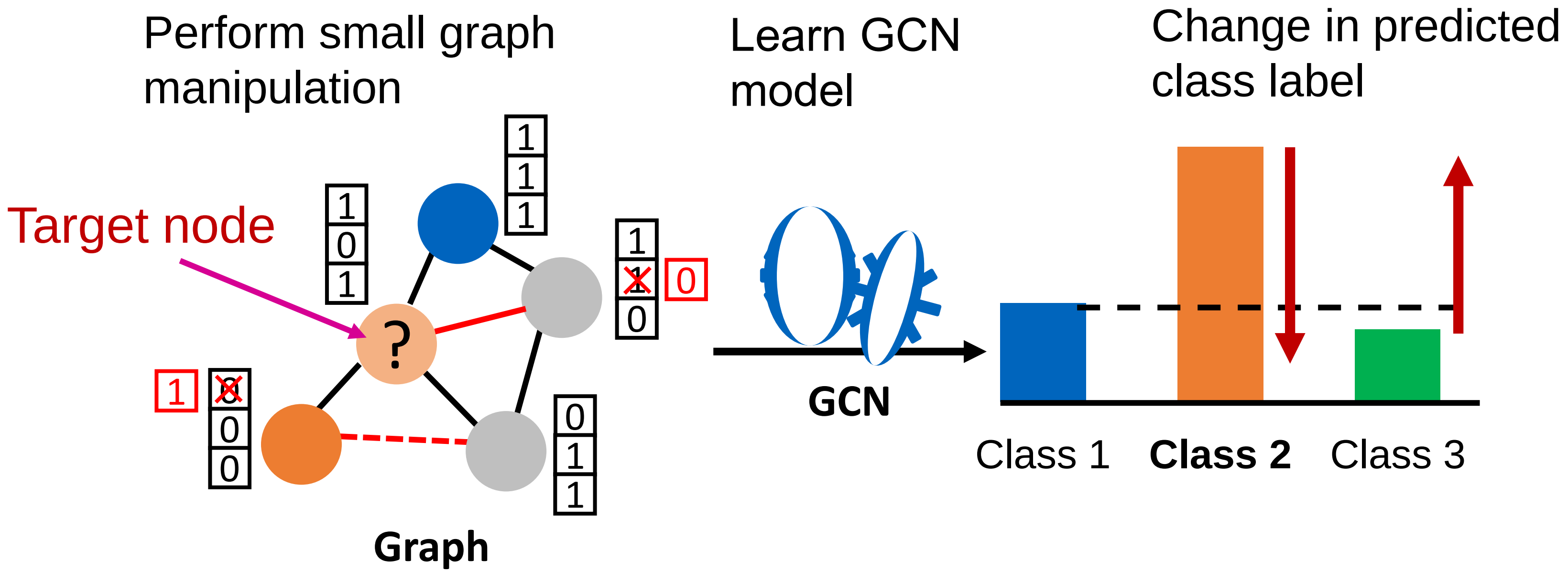


- Remove connections from **attackers**
  - Ex) Delete undesirable link

# Formalizing Adversarial Attackers

- **Objective for the attacker:**

> **Maximize (change of target node label prediction)**
>
> **Subject to (graph manipulation is small)**




**If graph manipulation is too large, it will easily be detected. Successful attacks should change the target prediction with "unnoticeably-small" graph manipulation.**

# Mathematical Formulation (1)

- **Original graph**:
  - $\boldsymbol{A}$: adjacency matrix, $\boldsymbol{X}$: feature matrix
- **Manipulated graph (after adding noise):**
  - $\boldsymbol{A}'$: adjacency matrix, $\boldsymbol{X}'$: feature matrix
- **Assumption**: $(\boldsymbol{A}', \boldsymbol{X}') \approx (\boldsymbol{A}, \boldsymbol{X})$
  - Graph manipulation is **unnoticeably small.**
    - Preserving basic graph statistics (e.g., degree distribution) and feature statistics.
  - Graph manipulation is either **direct** (changing the feature/connection of target nodes) or **indirect**.

# Mathematical Formulation (2)

- **Target node**: $v \in V$
  - Recall that we only consider semi-supervised node classification settings
- GCN learned over the **graph**

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \mathcal{L}_{train}(\boldsymbol{\theta}; \boxed{\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{X}}})$$

**What is $\widetilde{A}$ and $\widetilde{X}$ ?**

- GCN's original prediction on the target node:

$$c_v^* = \text{argmax}_c f_{\boldsymbol{\theta}^*}(\boldsymbol{A}, \boldsymbol{X})_{v,c}$$

**Predict the class $c_v^*$ of vertex $v$ that has the highest predicted probability**

- GCN's manipulated prediction on the target node

$$c_v^{*\prime} = \text{argmax}_c f_{\boldsymbol{\theta}^*}(\boldsymbol{A}', \boldsymbol{X}')_{v,c}$$

- **We want the prediction to change after the graph is manipulated:**

$$c_v^{*\prime} \neq c_v^*$$

# Mathematical Formulation (3)

- What is $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}}$ ?
- Evasion Attack: $\tilde{\boldsymbol{A}} = \boldsymbol{A}, \tilde{\boldsymbol{X}} = \boldsymbol{X}$
  - Attacking happens after the GNN model is trained (at test time)
  - Inductive setting
- Poisoning Attack: $\tilde{\boldsymbol{A}} = \boldsymbol{A}', \tilde{\boldsymbol{X}} = \boldsymbol{X}'$
  - Attacking happens before the GNN model is trained.

# Mathematical Formulation (4)

- **Change of prediction on target node $v$:**

$$\Delta(v; \boldsymbol{A}', \boldsymbol{X}') =$$

$$\log f_{\boldsymbol{\theta}^*}(\boldsymbol{A}', \boldsymbol{X}')_{v, c_v^{*\prime}} - \log f_{\boldsymbol{\theta}^*}(\boldsymbol{A}', \boldsymbol{X}')_{v, c_v^*}$$

Predicted (log) probability of the newly-predicted class $c_v^{*\prime}$

Predicted (log) probability of the originally-predicted class $c_v^*$

**Want to increase this term**

**Want to decrease this term**

# Mathematical Formulation (5)

- **Final optimization objective:**

$$\text{argmax}_{\boldsymbol{A}',\boldsymbol{X}'} \Delta(v; \boldsymbol{A}', \boldsymbol{X}')$$
$$\text{subject to } (\boldsymbol{A}', \boldsymbol{X}') \approx (\boldsymbol{A}, \boldsymbol{X})$$

- **Challenges in optimizing the objective**
  - Adjacency matrix $\boldsymbol{A}'$ is a discrete object: gradient-based optimization cannot be used.
- Several approximations are proposed to make the optimization.

# Nettack: Greedy Scheme

Manipulation budget

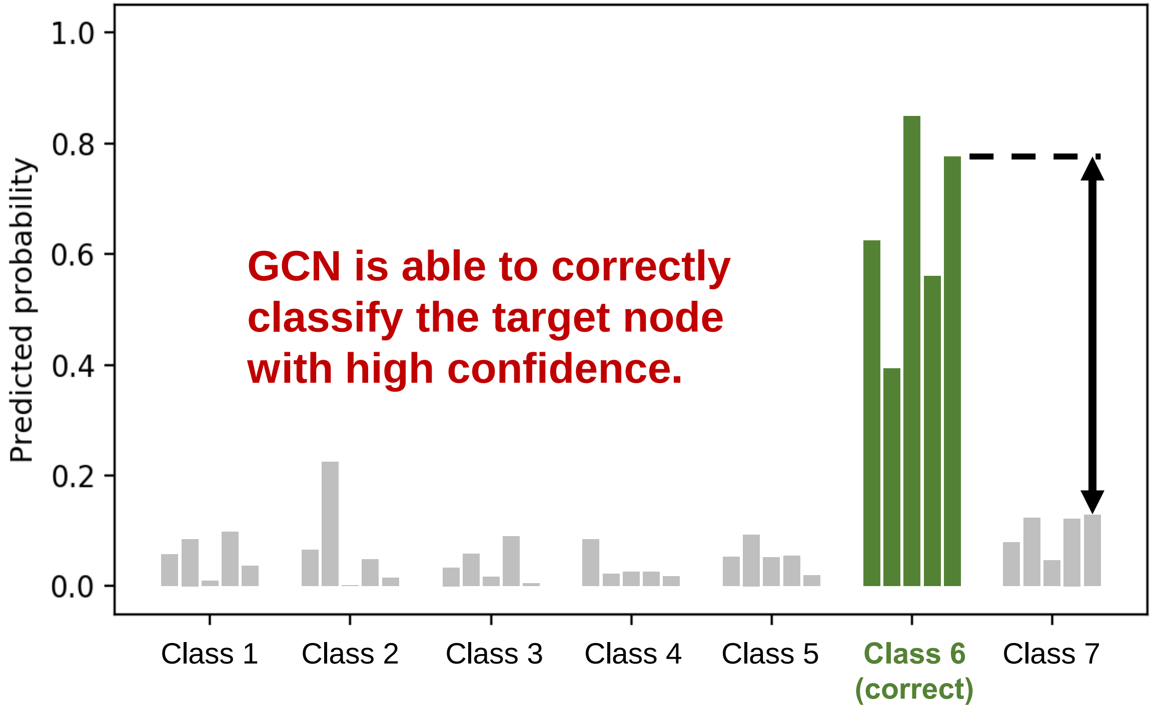
- A **greedy** scheme
- while $|A' - A| + |X' - X| < \boxed{\delta}$
  - Compute a **candidate set** for **struct perturbations** (all should be **unnoticeable**)
  - Pick the one which obtains the **highest** change of predictions
  - Compute a candidate set for **node features perturbations** (all should be **unnoticeable**)
  - Pick the one which obtains the **highest** change of predictions
- Challenges in this scheme:
  - How to make sure the perturbation is **unnoticeable** (budget $\delta$ is not enough)?
  - How to efficiently compute the **candidate sets**?
  - How to efficiently get the one which make the **highest change**?

# Netattack: Candidate Sets

- Efficiently compute **candidate sets** which make **unnoticeable** changes
- Main idea: use **siginificance test**
- **Graph structure** preserving perturbations
  - Preserve a graph's degree distribution (usually follow $p(x) \propto x^{-\alpha}$)
  - Can be incrementally computed (constant time) during changes
- **Feature statistics** preserving perturbations
  - Preserve feature co-occurrence
  - Can be precomputed
- Find more details in [Zügner, KDD 2018]

# Experiments: Adversarial Attack (1)

- **Semi-supervised node classification with GCN on a paper citation network (2,800 nodes, 8,000 edges).**

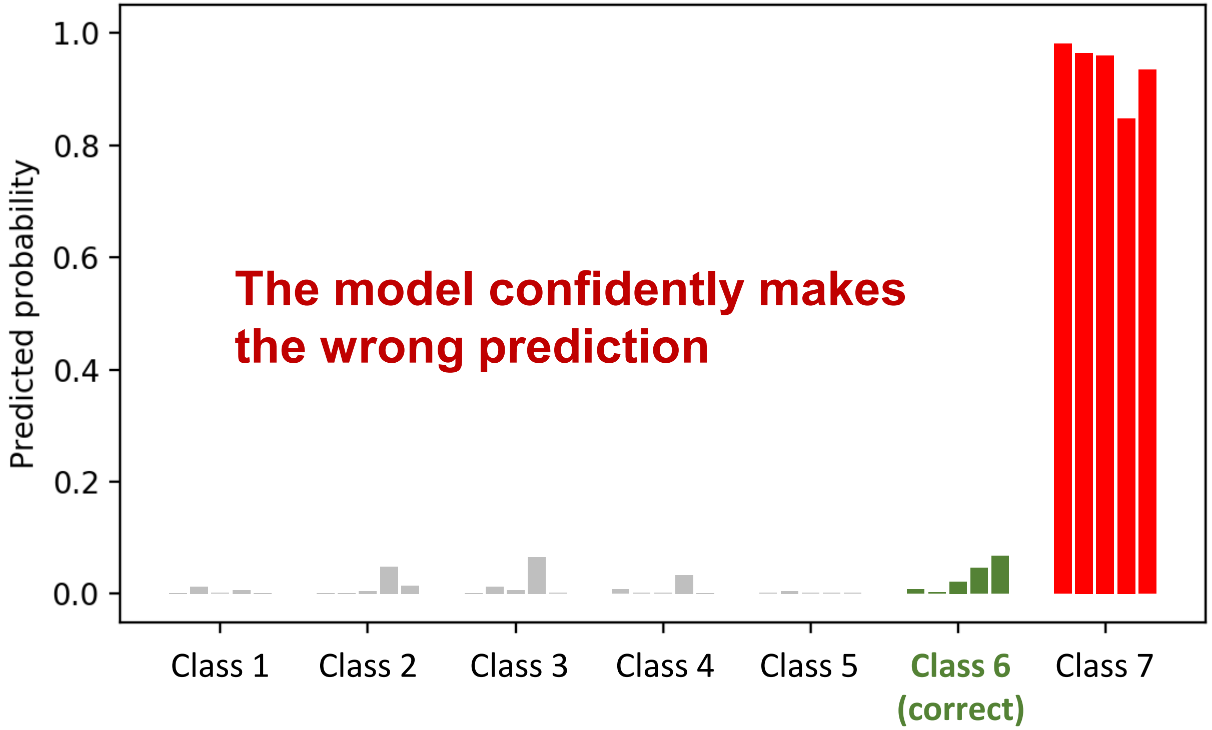



Predicted probabilities over 5 re-trainings
**(without graph manipulation, i.e., clean graph)**

**Classification margin**
> 0: Correct classification
< 0: Incorrect classification

# Experiments: Adversarial Attack (2)

- **GCN's prediction after carefully modifying just 5 edges attached to the target node (direct adversarial attack).**

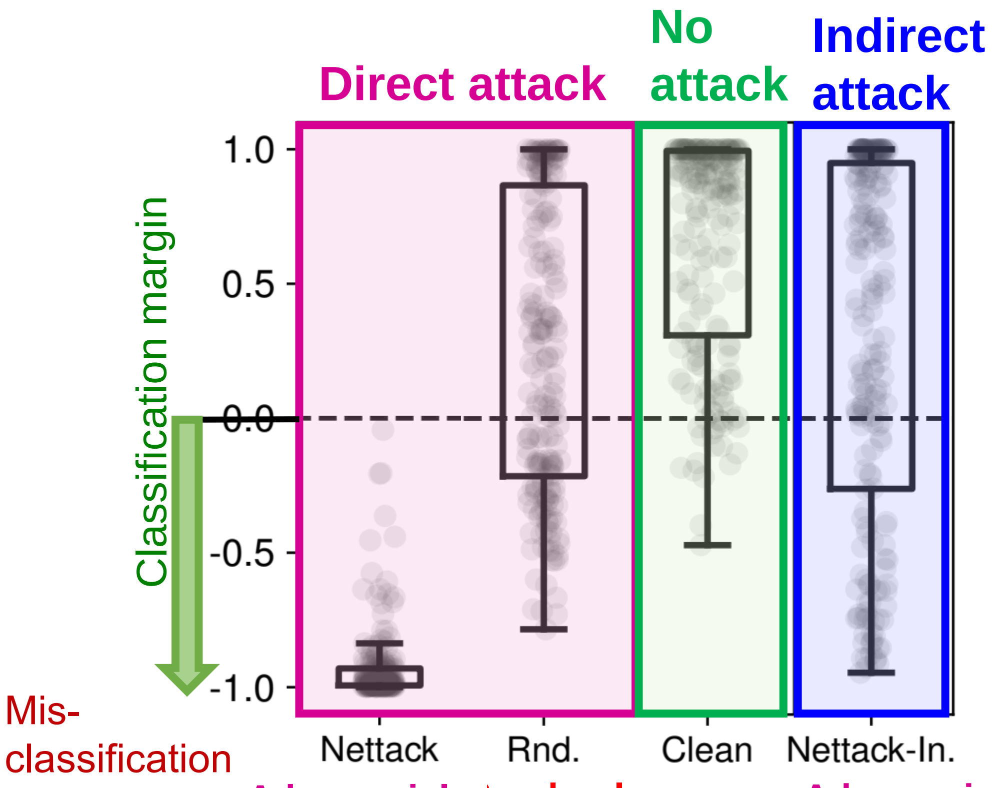



Predicted probabilities over 5 re-trainings (with adversarial attacks)

# Experiments: Attack Comparison

- **Adversarial direct attack** is the strongest attack, significantly worsening GCN's performance (compared to **no attack**).
- **Random** attack is much weaker than **adversarial** attack.
- **Indirect attack** is more challenging than direct attack.

Each dot indicates one attack trial.




**Adversarial** direct attack

**Randomly** add edges between target nodes and nodes with different labels.

**Adversarial** indirect attack

Adopted from Zügner et al. KDD 2018

# Outline of Today's Lecture

1. Trustworthy AI

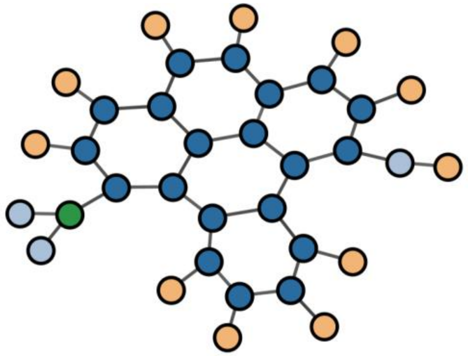
2. Adversarial Attacks and Robustness of GNNs

**3. Explainability for GNNs**

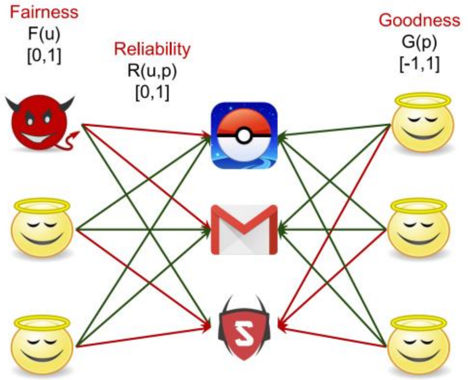
# Explainability: Motivation (1)

- Example questions after training GNNs:
  - Why is an item recommended to a user?
  - Why is the molecule mutagenic?
  - Why is the user classified as fraudulent?
- How to let the domain experts understand and trust the GNN model?


Recommender System



Mutagenic Molecule



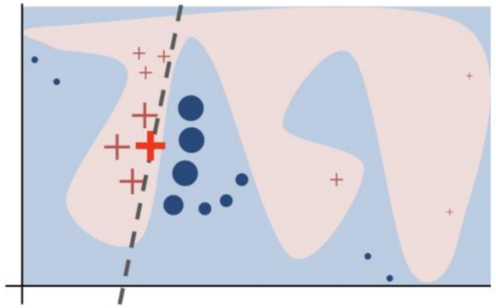

Fraudulent Detection

# Explainability: Motivation (2)

- Example questions after training GNNs:
  - Why is an item recommended to a user?
    Explain link prediction
  - Why is the molecule mutagenic?
    Explain graph classification
  - Why is the user classified as fraudulent?
    Explain node classification
- **Need to provide explanations to GNN models!**

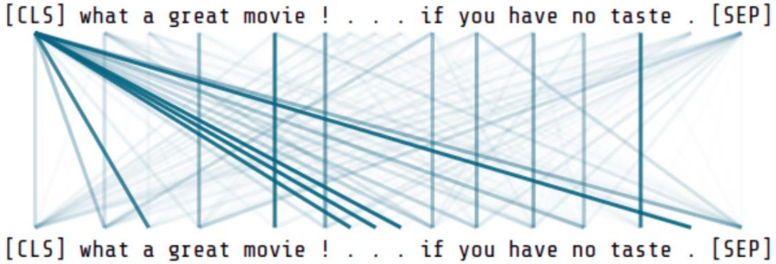
# Deep Learning Explainability Methods: Examples

- **Proxy Model**
  - Learn a inherently interpretable model locally approximating the original model (e.g. a linear model, interpret by weights).



- **Saliency Maps**
  - Compute gradients of outputs w.r.t. input pixels.

- **Attention Mechanisms**
  - Visualize attention weights in a attention models

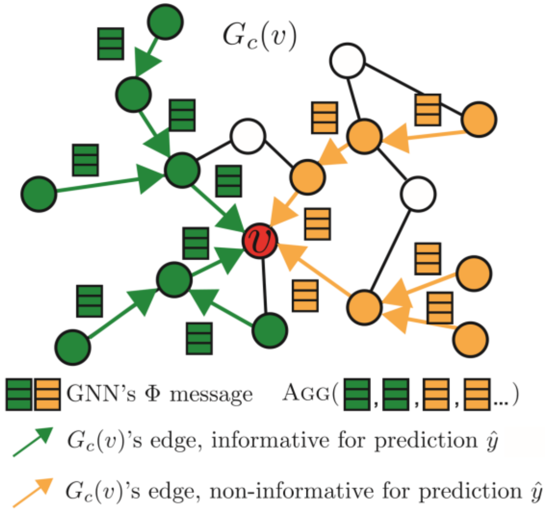


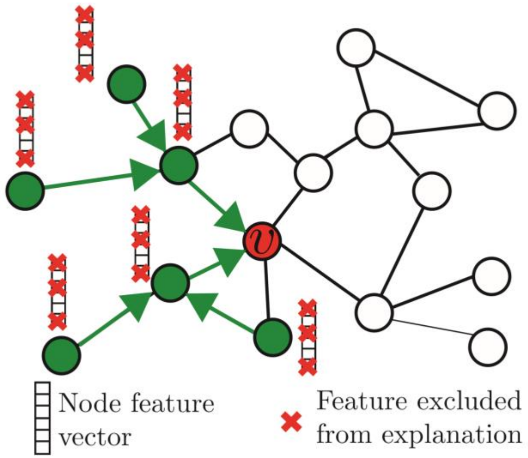
# Challenges of Applying these Methods to Graphs

- Why is it non-trivial to apply these explainability methods to graph structure?
  - predictions on graphs are induced by a complex combination of nodes and paths of edges between them (not only nodes).

# How to explain a GNN

- Message passing structure
- The importance of node features




**Structural explanation** **Feature explanation**

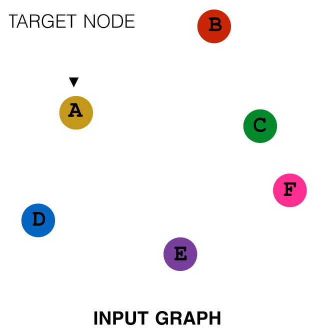
- GNNExplainer explain both aspects simultaneously

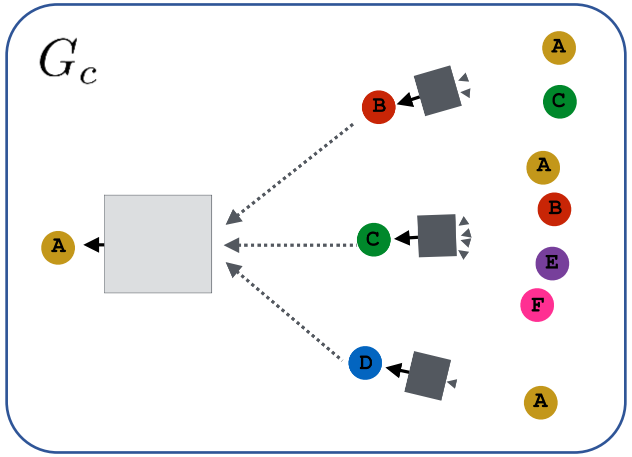


GNN-Explainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

# GNNExplainer Input

- Without loss of generality, consider node classification task:



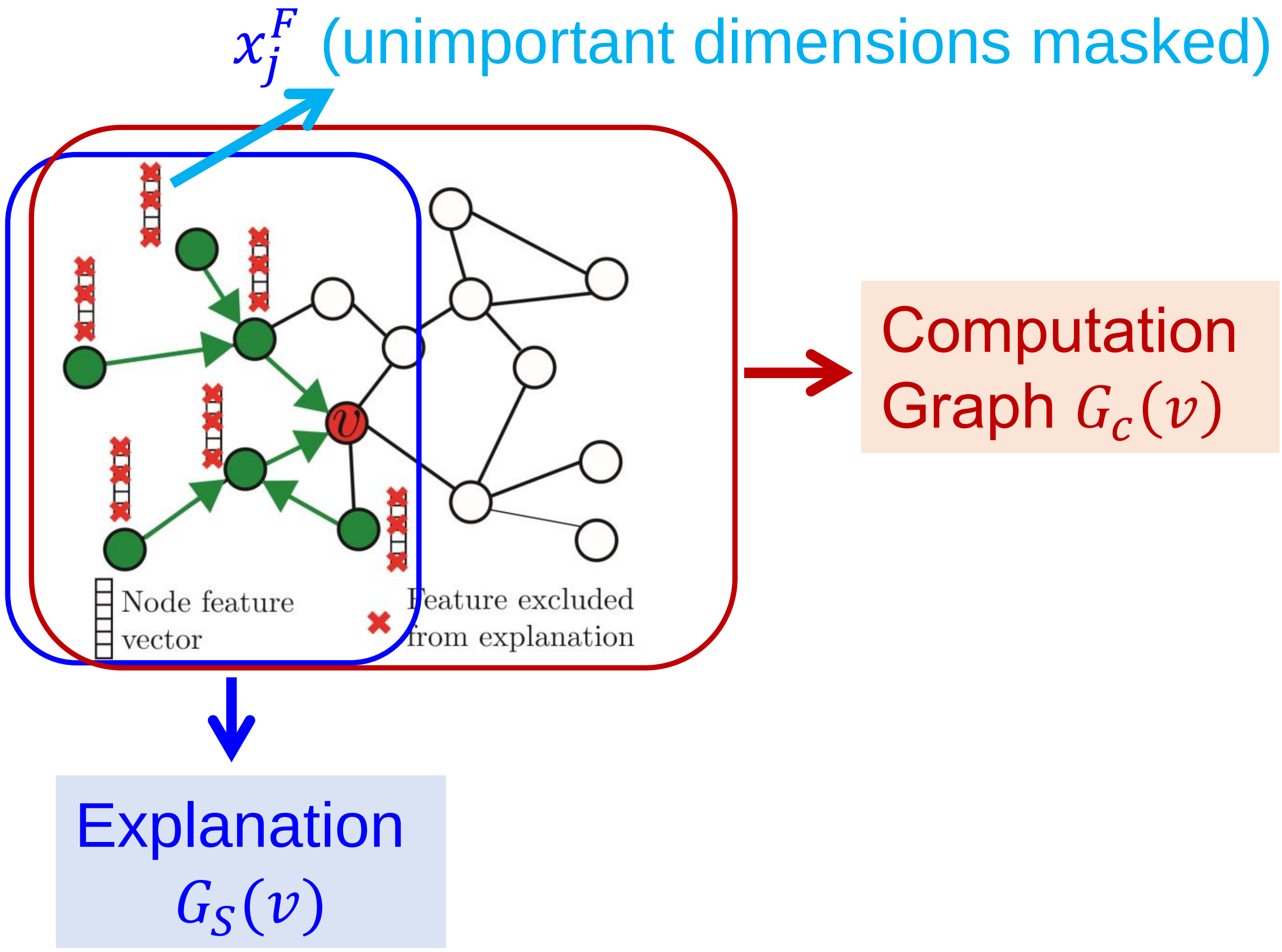




**Suppose GNN predicts label $\hat{y}$ for node $v$**

- Input computation graph: $G_c(v)$
- Adjacency matrix of $G_c$ : $A_c(v) \in \{0,1\}^{n \times n}$
- Node Feature: $X_c(v) = \{x_j | v_j \in G_c(v)\}$

# GNNExplainer Output

- GNN model $\phi$ learns $P_{\phi}(Y \mid A_c(v), X_c(v))$
- $Y$ denotes predicted label of $v$
- **GNNExplainer** outputs $(A_S, X_S^F)$
- Graph $G_S$ with adjacency matrix $A_S$ is a subgraph of graph with adjacency matrix $A_c(v)$ (omit $v$)
- $X_S^F = \{x_j^F \mid v_j \in G_S\}$ are features for $G_S$
- Mask $F$ masks out unimportant dimensions



# Explain by Mutual Information

- **Mutual information (MI)**
  - A measure of the mutual correlation between the two random variables.
  - Good explanation should have **high correlation** with model prediction
  - Relation to entropy:

$$MI(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

- GNNExplainer **Objective**:
  - Maximize MI between **label** and **explanation**

$$\max_{G_S} MI\big(Y;(A_S, X_S)\big) = H(Y) - H(Y|A = A_S, X = X_S^F)$$

# Explain by Optimization

- By relation to entropy:
  equivalent to minimize conditional entropy

$$\max_{A_S} MI\big(Y|(A_S, X_S)\big) = \min_{A_S} H(Y|\boxed{A = A_S,} \boxed{X = X_S^F})$$

Subgraph Feature subset

- Finding $A_S$ that minimizes the conditional entropy is computationally expensive!
  - Exponentially many possible $A_S$
- Solution: Treat explanation as a distribution of "**plausible explanations**", instead of a single graph
  - Optimize the expected explanation
  - Benefit 1: captures multiple possible explanations for the same node
  - Benefit 2: turns discrete optimization to continuous

# GNNExplainer Model

- continuous relaxation
  - Optimize the expected adjacency matrix $A_S$

expectation of explanations

$$\min_{\mathcal{A}} \boxed{\mathbb{E}_{A_S \sim \mathcal{A}}} H(Y \mid A = A_S, X = X_S)$$

  - View $\mathbb{E}_{A_S \sim \mathcal{A}}$ as an adjacency matrix where entries are continuous
- Approximation

continuous

$$\min_{\mathcal{A}} H(Y \mid A = \boxed{\mathbb{E}_{\mathcal{A}}[A_S]}, X = X_S)$$

- Optimize the expectation by masking

Element-wise multiply

- Use $A_C \odot \text{Mask}$ to represent $\mathbb{E}_{\mathcal{A}}[A_S]$
- If $\text{Mask}_{ij}$ close to 1, keep edge $(i, j)$; if close to 0, drop edge $(i, j)$.



GNN-Explainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

# GNNExplainer Model

- Let $\text{Mask} = \sigma(\boldsymbol{M})$ be the adjacency mask
  - Continuous relaxation: $\sigma(M) \in \mathbb{R}$ instead of binary
  - **Sigmoid** function $\sigma$ squashes $\boldsymbol{M}$ into $[0, 1]$
  - Masking: Element-wise multiply $\sigma(\boldsymbol{M})$ by $A_c$
- Assume edges are independent

$$P_{\mathcal{A}}(A_S) = \prod_{e=(j,k)\in G_c(v_i)} A_S[j,k]$$

- Objective:

$$\min_{M} -H(\log P_\phi (Y = y | G = A_C \odot \sigma(M), X = X_S)$$

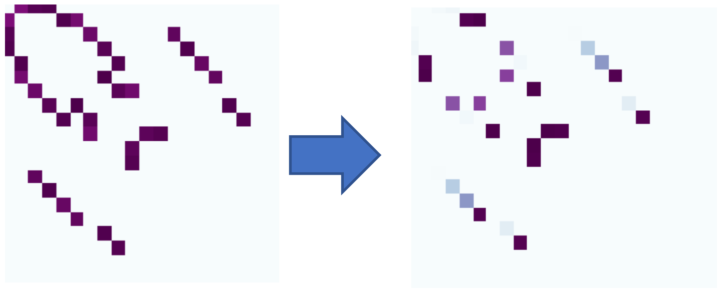
# GNNExplainer Model

- Optimize $M$ :

$$\min_{M} -H(\log P_{\phi}(Y = y | A = A_c \odot \sigma(M), X = X_S))$$

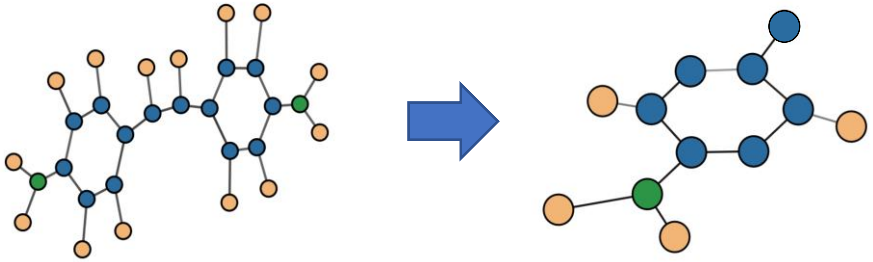
- $A_c \odot \sigma(M)$ is the relaxed adjacency matrix
- Threshold $A_c \odot \sigma(M)$ to get $G_S$
- Example:

$A_c \odot \sigma(M)$



Before training After training

Graph after thresholding



Before training After training

# Feature Explanation

- Similarly select features by masking
  $X_S^F = \{x_j^F | v_j \in G_S\}, \quad x_j^F = [x_{j,t_1}, \ldots, x_{j,t_k}]$

  For the selected dimensions, $\sigma(F_{t_i}) \to 1$

- Problem: Zero value could be important!
- Solution: Measure feature importance by how much drop in model confidence when features are replaced with random values.
  (See paper for details)

# Regularization Constraints

- **Optimize feature and adjacency masks jointly with regularization**
- **Concise explanation**
  - Mask size: $\mathrm{Sum}\big(\sigma(M)\big)$
  - Feature size: $\mathrm{Sum}(\sigma(F))$
- **Final Objective**

$$\min_{M} -H\big(\log P_{\phi}\,(Y = y|G = A_c \odot \sigma(M), X = X_S^F\big) + \underbrace{\lambda_1\,\mathrm{Sum}\big(\sigma(M)\big) + \lambda_2 \mathrm{Sum}(\sigma(F))}_{\text{Sum of entries in feature and adjacency masks}}$$

- Threshold $A_c \odot \sigma(M)$ to get the explanation $G_S$

# GNNExplainer Model

- **Task extensions**
  - Link prediction: optimize mask on union of 2 node neighborhoods
  - Graph classification: optimize mask on graph

- **Can adapt to different architectures**
  - Graph Attention Networks
  - Gated Graph Sequence
  - Graph Networks
  - GraphSAGE
  - Jumping Knowledge Networks

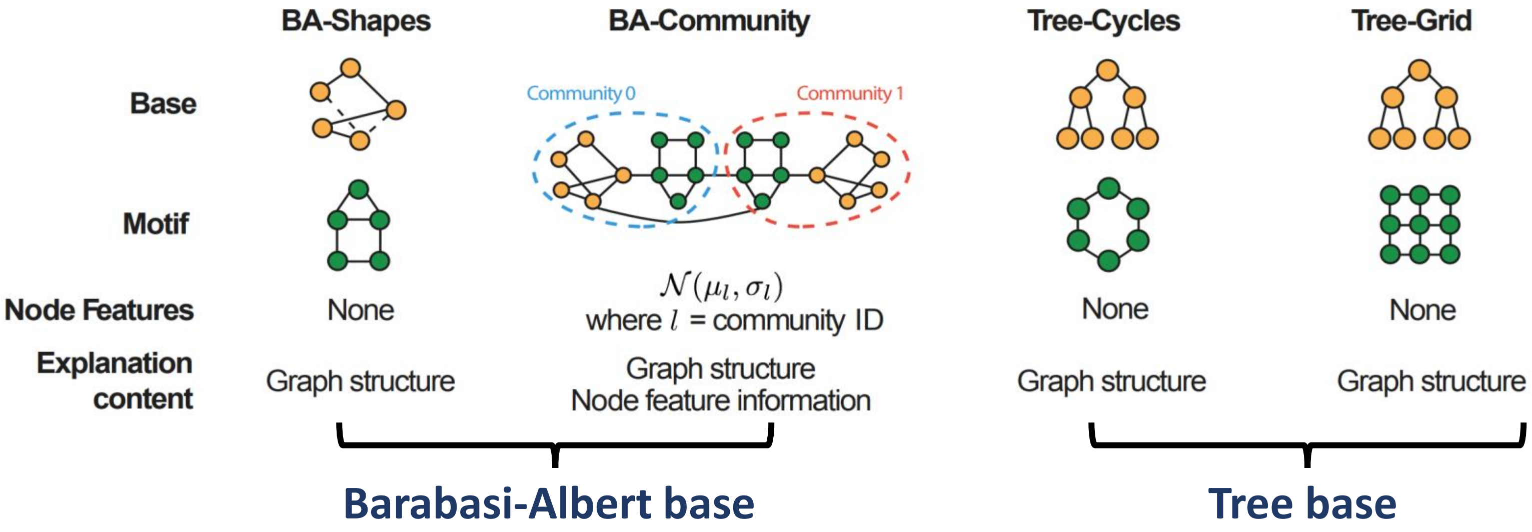
We use GNN
general formulation

# Experiments: Baselines

- GNN saliency map based on gradients of output score w.r.t. inputs
- Attention values based on Graph Attention Networks (GAT)
  - Edge importance indicated by average attention weights across layers for each edge

# Experiments: Datasets (1)

- Synthetic task: is a node part of a given motif?
  - 100 Motifs are randomly attached to nodes in base graphs (500 nodes)
  - Node classification (structural roles)



# Experiments: Datasets (2)

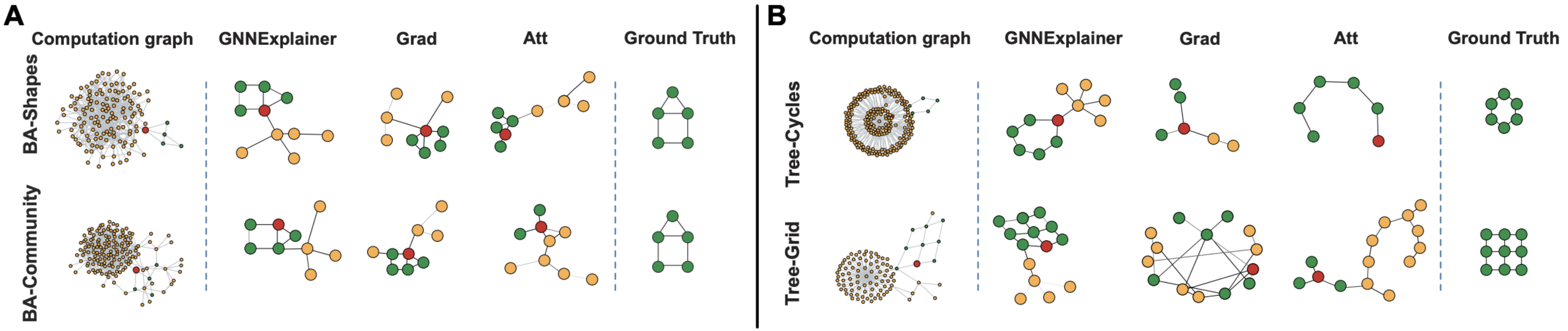
- Real-world tasks
  - Social networks (Reddit-binary dataset)
    - Reddit community prediction
  - Chemistry (Mutagenic molecule dataset)
    - Chemical property prediction
  - Graph classification

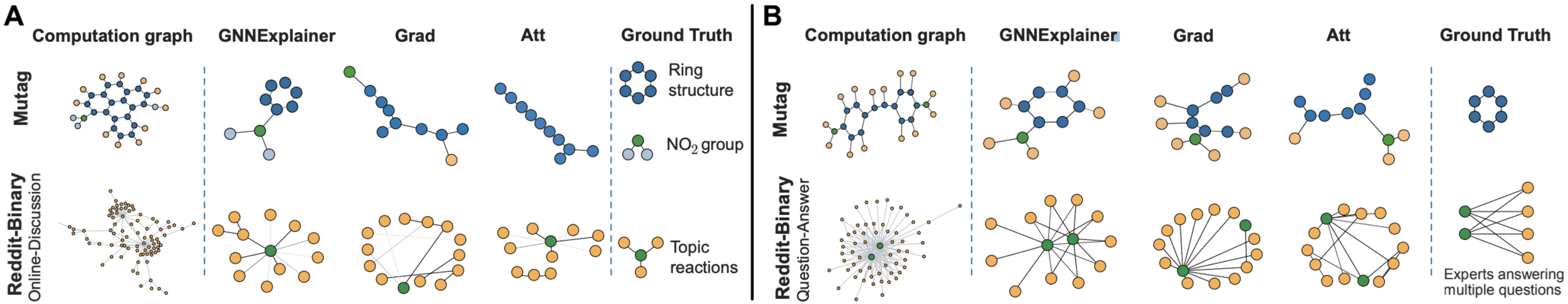
# Results: Quantiative Analysis

- Node classification with ground-truth
- Measures accuracy of explanation with respect to ground-truth

| | BA-House | BA-Comm | Tree-Cycle | Tree-Grid |
|---|---|---|---|---|
| Grad | 88.2 | 73.9 | 82.4 | 61.2 |
| Att | 81.5 | 75.0 | 90.5 | 66.7 |
| GNN-Explainer | **92.5** | **83.6** | **94.8** | **87.5** |

# Results: Qualitative Analysis



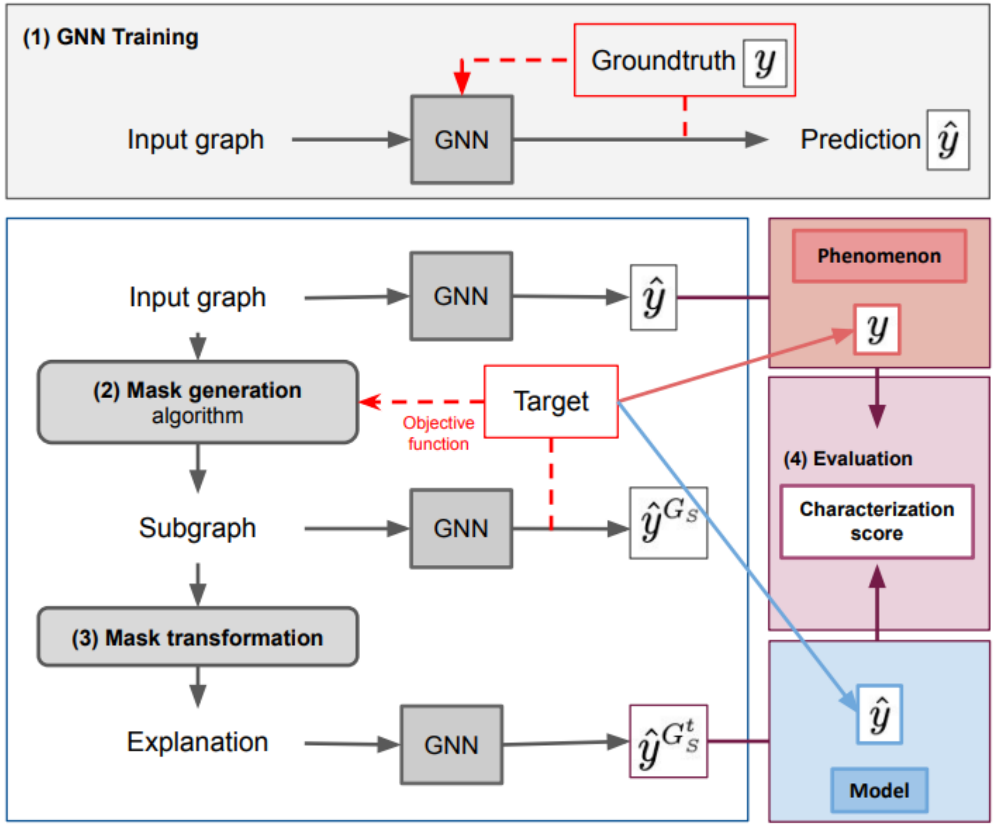



# Explainability Method Evaluation

- **Challenge: groundtruth might not always be available**
- Evaluation is **multi-dimensional**

- **Goal** (phenomenon *vs.* model)
- **Masking** strategy
- **Type** (sufficiency vs. necessity)

- **GraphFramEx**
- Benchmarks and evaluation criteria for graph explainability



# Explanation Goal

- **Phenomenon** Explanation
  - Explain the underlying reasons for the ground truth phenomenon
- **Model** Explanation
  - Explain why model makes a particular prediction
- Adapt the **fidelity** measure for both cases

Phenomenon

$$fid_{+} = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}\left(\hat{y}_i^{G_{C\setminus S}} = y_i\right)\right|$$

$$fid_{-} = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}\left(\hat{y}_i^{G_S} = y_i\right)\right|$$

Model

$$fid_{+} = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\left(\hat{y}_i^{G_{C\setminus S}} = \hat{y}_i\right)$$

$$fid_{-} = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}\left(\hat{y}_i^{G_S} = \hat{y}_i\right)$$

# Masking Strategy

- **Hard mask**
  - Produce a subgraph as an explanation for the prediction
- **Soft mask**
  - A number between 0 to 1 on edges and features to indicate their importance

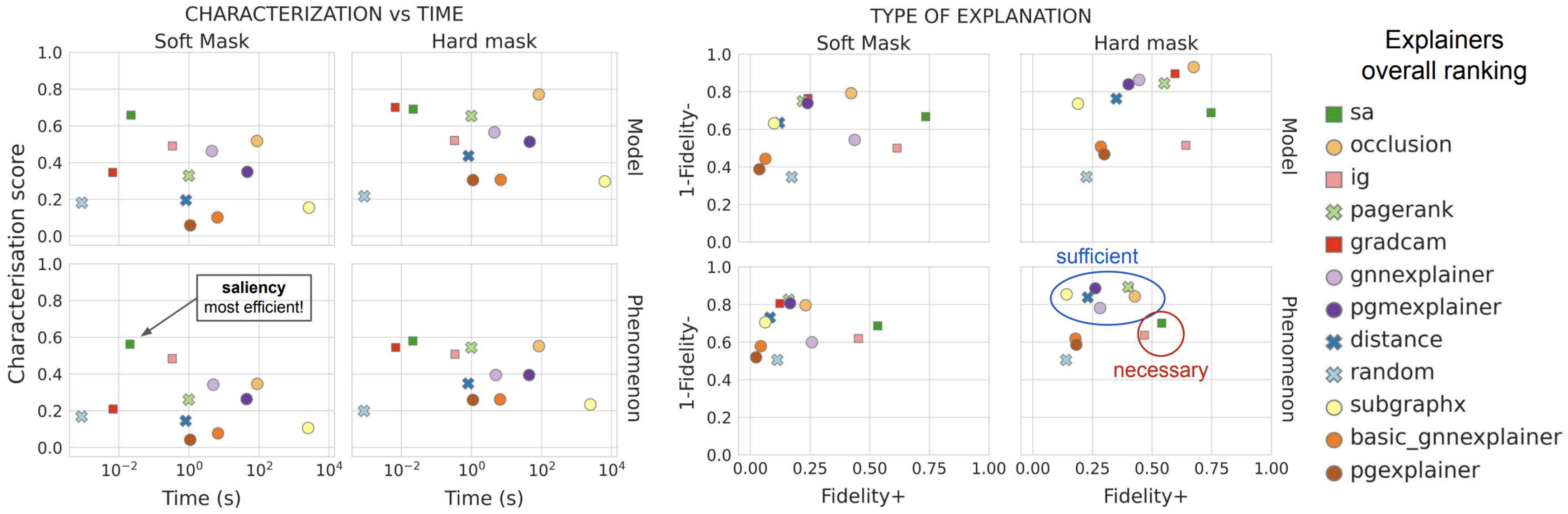
# Types of Explanations

- **Sufficiency**
  - An explanation is sufficient if it leads by its own to the initial prediction of the model explanation. ($fid_{-} \to 0$)
- **Necessity**
  - An explanation is necessary if the model prediction changes when removing it from the initial graph. ($fid_{+} \to 1$)
- **Characterization** score to summarize the explanation quality

$$charact = \frac{w_{+}+w_{-}}{\frac{w_{+}}{fid_{+}}+\frac{w_{-}}{1-fid_{-}}} = \frac{(w_{+}+w_{-})\times fid_{+}\times(1-fid_{-})}{w_{+}\cdot(1-fid_{-})+w_{-}\cdot fid_{+}}$$

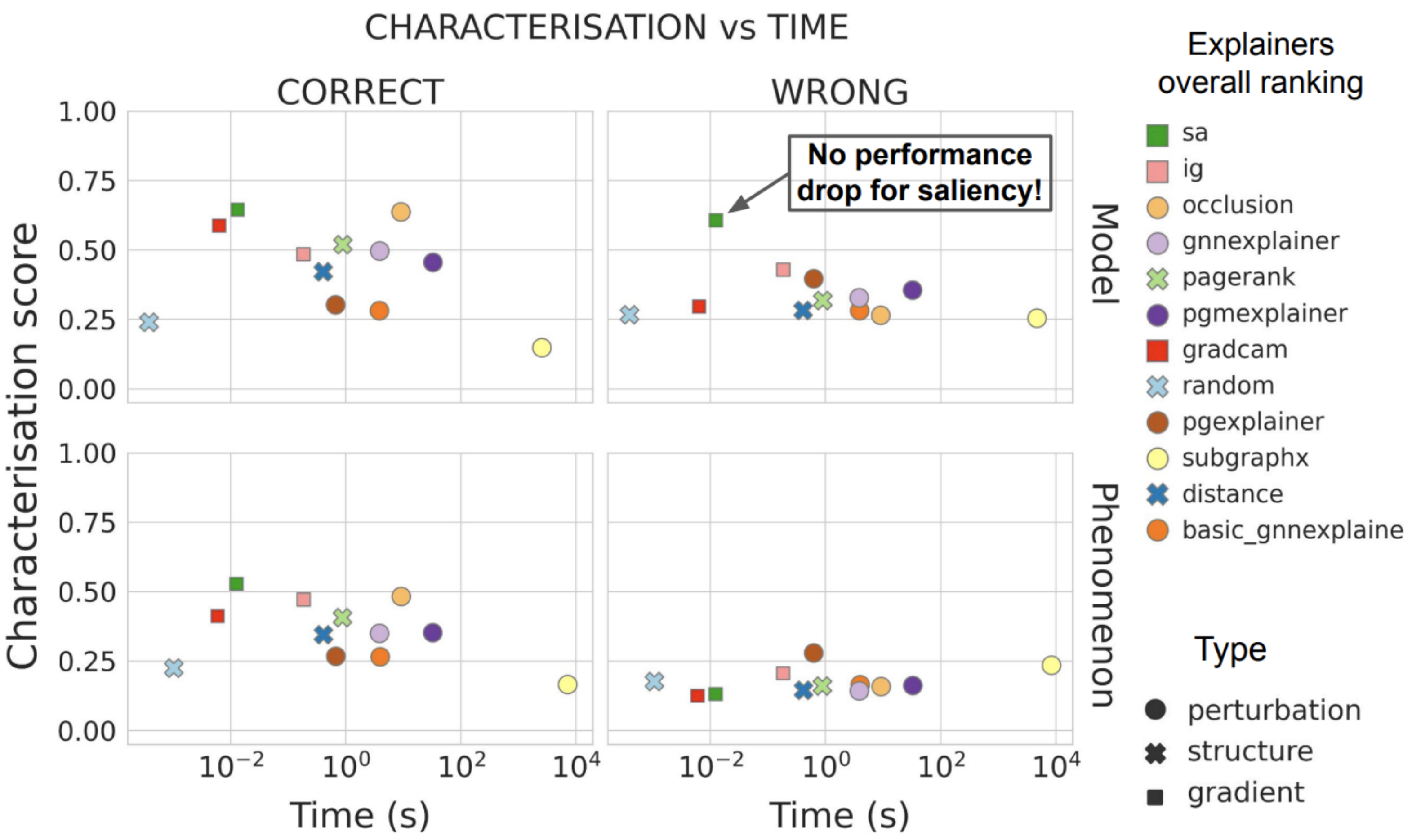
# Results: Explain Efficiency vs. Characterization Score

- Saliency has the highest overall characterization score and efficiency.
- Occlusion has the best overall score in the setting of hard mask.



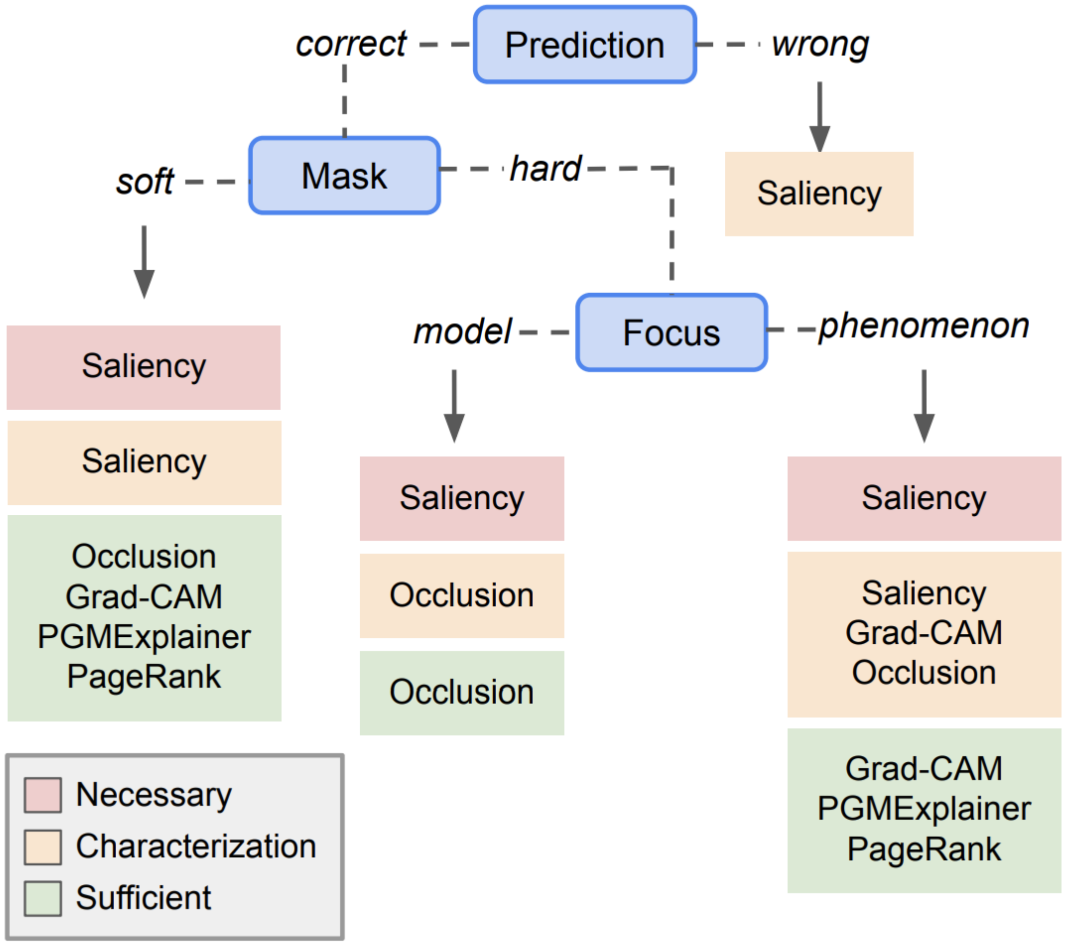
# Results: Correct & Wrong Predictions

- Correct predictions → phenomenon & model
- Wrong predictions → model



# Results: How to Select an Explainability Method

- Decision Tree which guides users to select the optimal method



# Summary of the Lecture

- Trustworthy GNN
  - Robustness, explainability, privacy, fairness, accountability, environmental well-being,…
- Adversarial Attacks and Robustness of GNNs
  - Adversarial examples
  - Attack possibilities: direct attack, indirect attack
  - Mathematical formulation
- Explainability of GNN
  - GNNExplainer
  - GraphFramEx