Advanced Topics in Scalable Learning

CSE 6392 Lecture 3

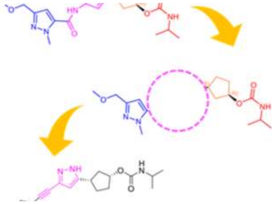
Junzhou Huang, Ph.D.

Department of Computer Science and Engineering

Trustworthy Graph Learning

Deep graph learning are wildly applied to various risk/privacy-sensitive scenarios!

How to deploy these algorithms in a trustworthy manner?

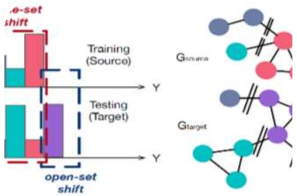


Drug discovery

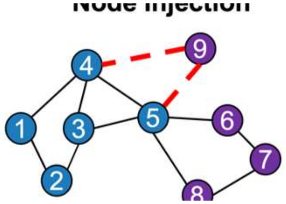
Healthcare

Credit modeling

Fraud detection

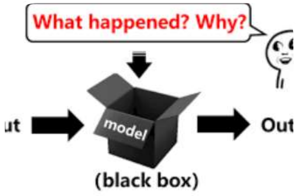


Distribution shift

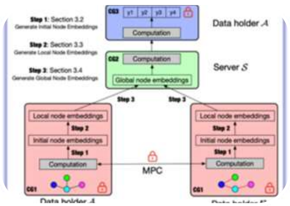


Adversarial attack

Label/attribute noise



Explainability



Privacy Protection

Overview

Reliability

- Inherent noise
- Distribution shift
- Adversarial attack

Explainability

- Post-hoc method
- Self-explainable method

Privacy

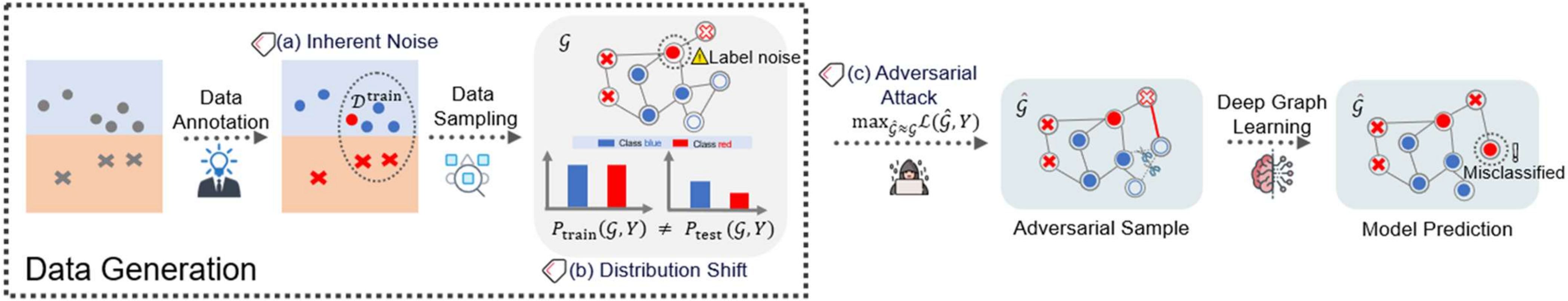
- Federated GNN
- Privacy inference attack
- Privacy enhancing technique

Reliability of GNNs

Reliability of GNNs

- Overview
- GNNs against inherent noise
  - Threat Overview
  - Enhancing Techniques
- GNNs against distribution shift
  - Threat Overview
  - Enhancing Techniques
- GNNs against adversarial attacks
  - Threat Overview
  - Enhancing Techniques
- Toolbox

Overview



- Inherent Noise
- Distribution Shift
- Adversarial Attack

Training data　Structure noise　Attribute noise　Label noise

$$\boxed{\mathcal{D}^{\text{train}}} = (\mathbf{A} + \epsilon_a, \mathbf{X} + \epsilon_x, Y + \epsilon_y)$$

Data distribution

$$\boxed{P}_{\text{train}}(\mathcal{G}, Y) \neq P_{\text{test}}(\mathcal{G}, Y)$$

Adversarial goal

$$\boxed{\max}_{\hat{\mathcal{G}}\approx\mathcal{G}} \mathcal{L}(\hat{\mathcal{G}}, Y)$$

$\boldsymbol{A}$: adjacency matrix
$\boldsymbol{X}$: feature matrix
Y: labels
$\mathcal{G}$: input graph
$\hat{\mathcal{G}}$: perturbed graph

Reliability of GNNs

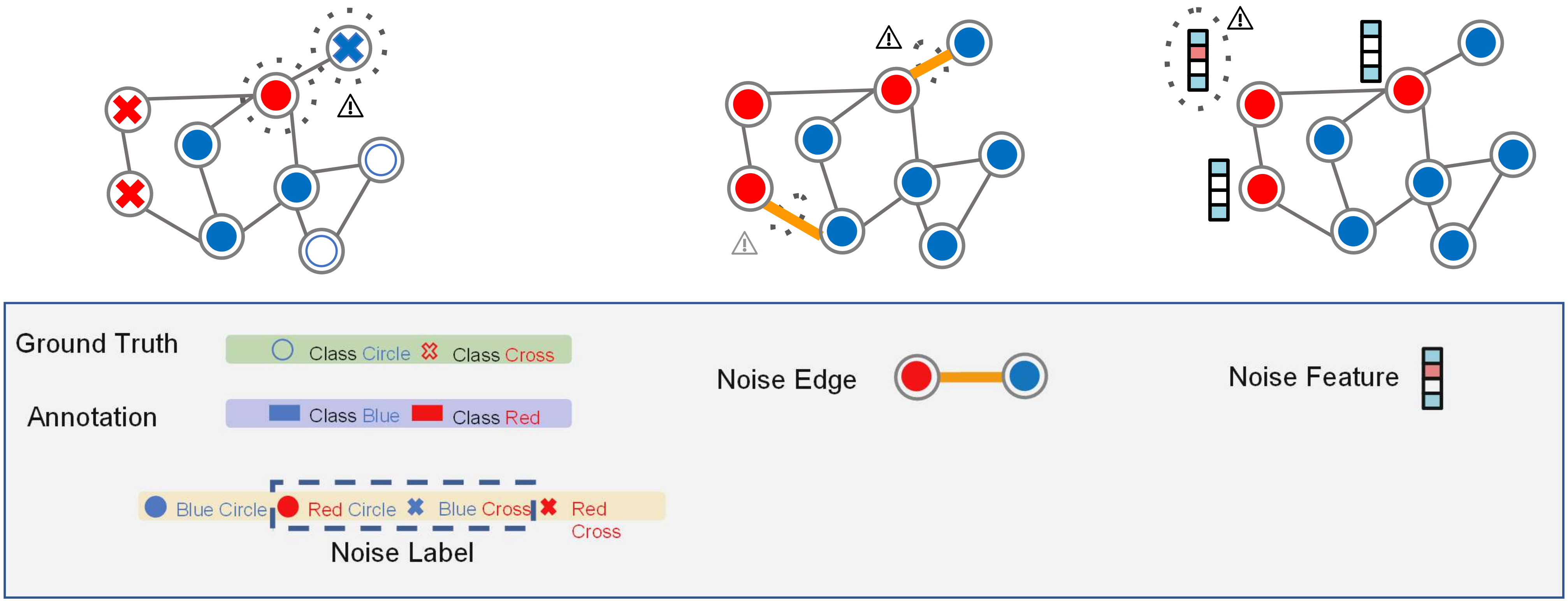
- Overview
- GNNs against inherent noise
  - Threat Overview
  - Enhancing Techniques
- GNNs against distribution shift
  - Threat Overview
  - Enhancing Techniques
- GNNs against adversarial attacks
  - Threat Overview
  - Enhancing Techniques
- Toolbox

Threat Overview: Inherent Noise

Label noise

Structure noise

Attribute noise



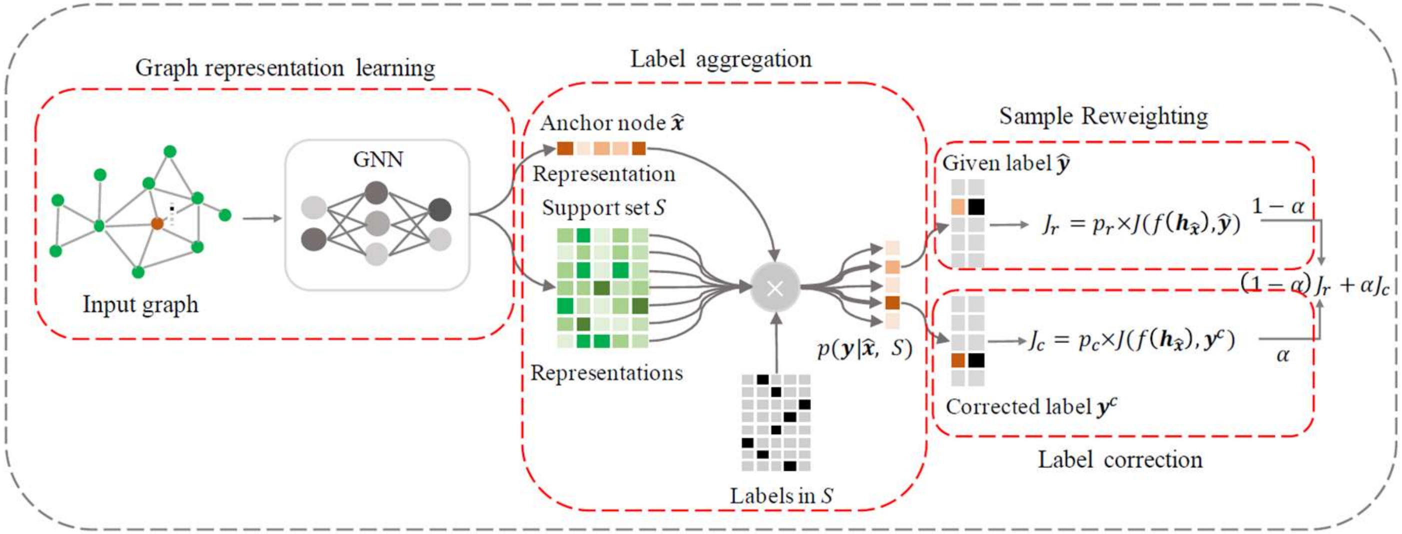
Defending Against Inherent Noise

Enhancing techniques on graph data can be categorized as:

- Graph Denoising
- Regularization Tricks

Graph Denoising: UnionNET

- Label aggregation
- Sample reweighting
- Label correction



Unified Robust Training for Graph Neural Networks against Label Noise. PAKDD 2021

Graph Denoising: UnionNET

Step 1:non-parametric attention mechanism:

$$P(\mathbf{y}|\hat{\mathbf{x}},S)=\sum_{\mathbf{x}_i\in S}\mathcal{A}(\hat{\mathbf{x}},\mathbf{x}_i)\mathbf{y}_i=\sum_{\mathbf{x}_i\in S}\frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T\mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j\in S}\exp(\mathbf{h}_{\mathbf{x}_j}^T\mathbf{h}_{\hat{\mathbf{x}}})}\mathbf{y}_i.$$

Support set: $S=\{(\mathbf{x}_i,\mathbf{y}_i)|\hat{\mathbf{x}}\}^k$
obtained by random walk on training nodes $\{\hat{\mathbf{x}}\}$

Step 2: compute reweighting score

$$p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}},S)=\sum_{\mathbf{x}_i\in S,\mathbf{y}_i=\hat{\mathbf{y}}}\frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T\mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j\in S}\exp(\mathbf{h}_{\mathbf{x}_j}^T\mathbf{h}_{\hat{\mathbf{x}}})}\mathbf{y}_i.$$

$$\mathcal{J}_r=-\sum_{\hat{\mathbf{x}}\in\mathcal{L}}p_r(\hat{\mathbf{y}}|\hat{\mathbf{x}},S)\times\hat{\mathbf{y}}\log(f(\mathbf{h}_{\hat{\mathbf{x}}})),$$

Step 3:label correction loss

$$\mathcal{J}_c=-\sum_{\hat{\mathbf{x}}\in\mathcal{L}}p_c(\mathbf{y}^c|\hat{\mathbf{x}},S)\times\mathbf{y}^c\log(f(\mathbf{h}_{\hat{\mathbf{x}}})),$$

$$p_c(\mathbf{y}^c|\hat{\mathbf{x}},S)=\max_{\mathbf{y}_i}P(\mathbf{y}_i|\hat{\mathbf{x}},S)=\max_{\mathbf{y}_i}\sum_{\mathbf{x}_i\in S}\frac{\exp(\mathbf{h}_{\mathbf{x}_i}^T\mathbf{h}_{\hat{\mathbf{x}}})}{\sum_{\mathbf{x}_j\in S}\exp(\mathbf{h}_{\mathbf{x}_j}^T\mathbf{h}_{\hat{\mathbf{x}}})}\mathbf{y}_i.$$

Step 4: KL-divergence loss

$$\mathcal{J}_p=\sum_{j=1}^{m}p_j\log\frac{p_j}{\overline{f(\mathbf{h_x})}_j},\qquad \overline{f(\mathbf{h_x})}_j=\frac{1}{|L|}\sum_{\mathbf{x}\in\mathcal{L}}f(\mathbf{h_x})_j$$

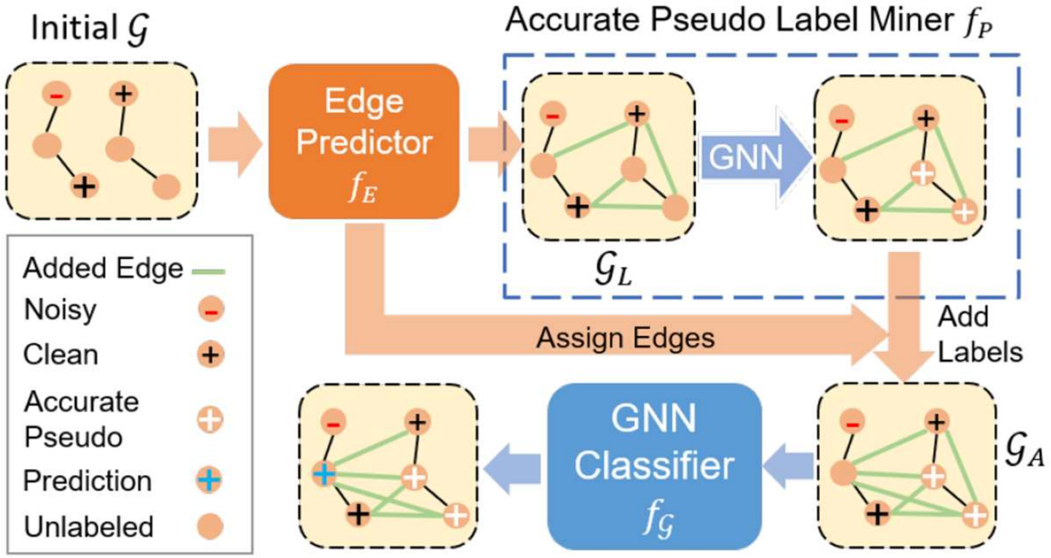
Step 5: Total training loss

$$\mathcal{J}_f=(1-\alpha)\mathcal{J}_r+\alpha\mathcal{J}_c+\beta\mathcal{J}_p$$

Unified Robust Training for Graph Neural Networks against Label Noise. PAKDD 2021

Graph Denoising: NRGNN

NRGNN is composed of an **edge predictor**, an **accurate pseudo label miner**, and a **GNN classifier.**



- **Edge predictor:**
  - Link unlabeled nodes with similar nodes having noisy/pseudo labels
- **Accurate pseudo label miner:**
  - Obtain accurate pseudo labels
- **GNN classifier:**
  - Link unlabeled nodes with edge predictor
  - Produce robust predictions

NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. KDD 2021

Graph Denoising: NRGNN

Edge Predictor

A: adjacency matrix
X: feature matrix

- **GCN is the backbone of Edge predictor**

$$\mathbf{Z} = GCN(\mathbf{A}, \mathbf{X}).$$

- **Predict the score of node pairs to determine whether adding edges**

$$\mathbf{S}_{ij} = \sigma(\mathbf{z}_i \mathbf{z}_j^T)$$

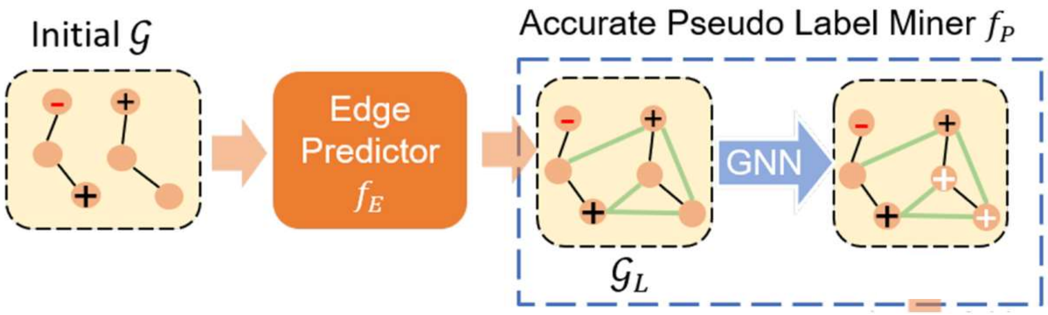
- **Objective function (reconstruction loss with negative sampling)**

$$\min_{\theta_E} \mathcal{L}_E = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in N(v_i)} \left((\mathbf{S}_{ij} - 1)^2 + \sum_{n=1}^{K} \mathbb{E}_{v_n \sim P_n(v_i)} (\mathbf{S}_{in} - 0)^2 \right)$$

NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. KDD 2021

Graph Denoising: NRGNN

Accurate Pseudo Label Miner



- **Link unlabeled nodes with similar labeled nodes:** Reduce effects of label noise
- **Obtain Accurate pseudo labels:** Predictions with large confidence scores

$$\mathcal{Y}_P = \{\hat{y}_i^P \in \hat{\mathcal{Y}}_U^P ; \hat{y}_{ic}^P > T_p\}$$

- **Objective function of f_P:**

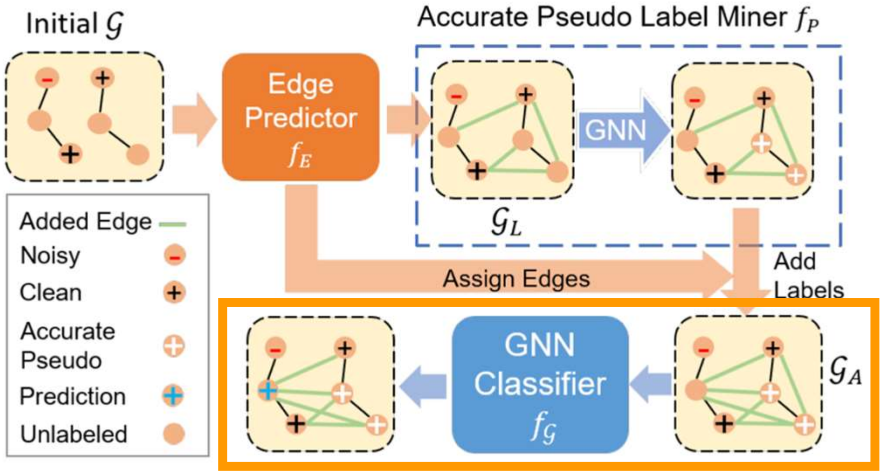
Prediction of pseudo label miner

$$\min_{\theta_P} \mathcal{L}_P = \sum_{v_i \in \mathcal{V}_L} l(\boxed{\hat{y}_i^P}, y_i)$$

NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. KDD 2021

Graph Denoising: NRGNN

GNN Classifier



- **Add label**

Accurate pseudo label are added to the label set

- **Assign edges**

Edge predictor will link the unlabeled nodes with similar extended labeled nodes: $\mathcal{V}_A = \boxed{\mathcal{V}_l} \cup \boxed{\mathcal{V}_P}$

labeled nodes **unlabeled nodes**

- **Accurate pseudo labels and provided noisy labels are covered in the loss function**

$$\mathcal{L}_G = \sum_{\boxed{v_i \in \mathcal{V}_A}} l(\hat{y}_i, y_i) \longrightarrow \text{Extended labeled nodes}$$

Overall objective function

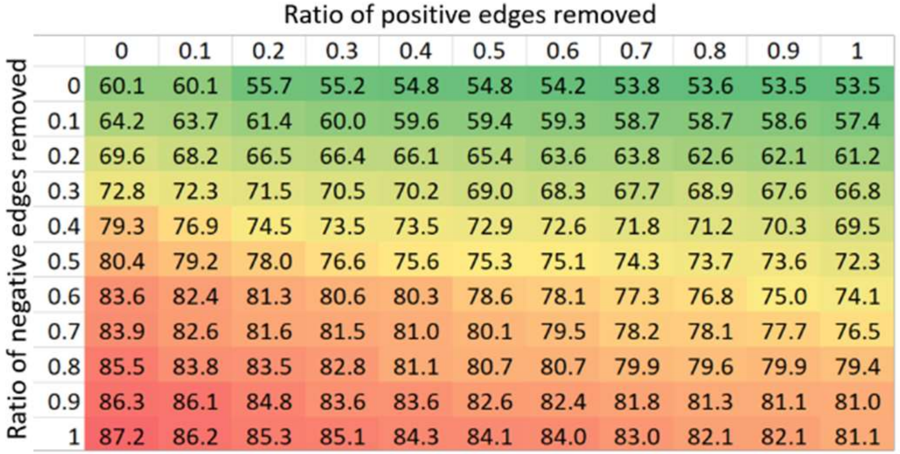
$$\arg\min_{\theta_E, \theta_P, \theta_G} \mathcal{L}_G + \alpha \mathcal{L}_E + \beta \mathcal{L}_P$$

NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. KDD 2021

Graph Denoising: PTDNet

Motivating Example

- positive edges ⇨ high quality node representation
- negative edges ⇨ low predictive accuracy



Ratio of positive edges removed

Ratio of negative edges removed

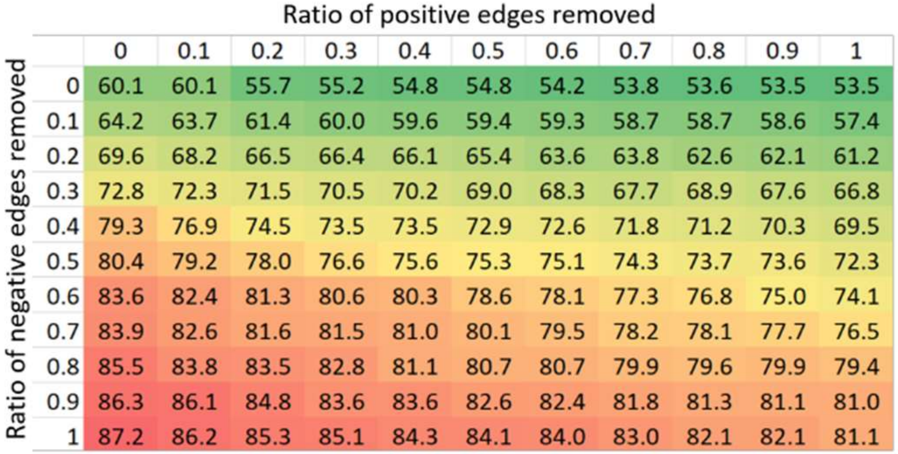
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	60.1	60.1	55.7	55.2	54.8	54.8	54.2	53.8	53.6	53.5	53.5
0.1	64.2	63.7	61.4	60.0	59.6	59.4	59.3	58.7	58.7	58.6	57.4
0.2	69.6	68.2	66.5	66.4	66.1	65.4	63.6	63.8	62.6	62.1	61.2
0.3	72.8	72.3	71.5	70.5	70.2	69.0	68.3	67.7	68.9	67.6	66.8
0.4	79.3	76.9	74.5	73.5	73.5	72.9	72.6	71.8	71.2	70.3	69.5
0.5	80.4	79.2	78.0	76.6	75.6	75.3	75.1	74.3	73.7	73.6	72.3
0.6	83.6	82.4	81.3	80.6	80.3	78.6	78.1	77.3	76.8	75.0	74.1
0.7	83.9	82.6	81.6	81.5	81.0	80.1	79.5	78.2	78.1	77.7	76.5
0.8	85.5	83.8	83.5	82.8	81.1	80.7	80.7	79.9	79.6	79.9	79.4
0.9	86.3	86.1	84.8	83.6	83.6	82.6	82.4	81.8	81.3	81.1	81.0
1	87.2	86.2	85.3	85.1	84.3	84.1	84.0	83.0	82.1	82.1	81.1

Learning to Drop: Robust Graph Neural Network via Topological Denoising, **WSDM 2021**

Graph Denoising: PTDNet

Motivating Example

- positive edges ⇨ high quality node representation
- negative edges ⇨ low predictive accuracy

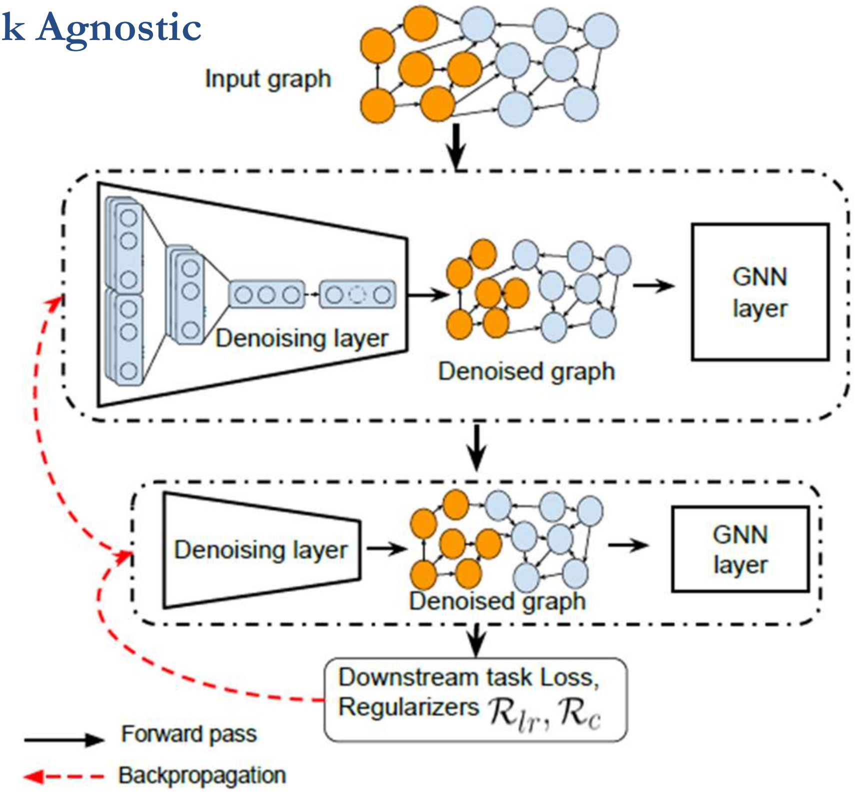


Ratio of positive edges removed

Ratio of negative edges removed

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	60.1	60.1	55.7	55.2	54.8	54.8	54.2	53.8	53.6	53.5	53.5
0.1	64.2	63.7	61.4	60.0	59.6	59.4	59.3	58.7	58.7	58.6	57.4
0.2	69.6	68.2	66.5	66.4	66.1	65.4	63.6	63.8	62.6	62.1	61.2
0.3	72.8	72.3	71.5	70.5	70.2	69.0	68.3	67.7	68.9	67.6	66.8
0.4	79.3	76.9	74.5	73.5	73.5	72.9	72.6	71.8	71.2	70.3	69.5
0.5	80.4	79.2	78.0	76.6	75.6	75.3	75.1	74.3	73.7	73.6	72.3
0.6	83.6	82.4	81.3	80.6	80.3	78.6	78.1	77.3	76.8	75.0	74.1
0.7	83.9	82.6	81.6	81.5	81.0	80.1	79.5	78.2	78.1	77.7	76.5
0.8	85.5	83.8	83.5	82.8	81.1	80.7	80.7	79.9	79.6	79.9	79.4
0.9	86.3	86.1	84.8	83.6	83.6	82.6	82.4	81.8	81.3	81.1	81.0
1	87.2	86.2	85.3	85.1	84.3	84.1	84.0	83.0	82.1	82.1	81.1

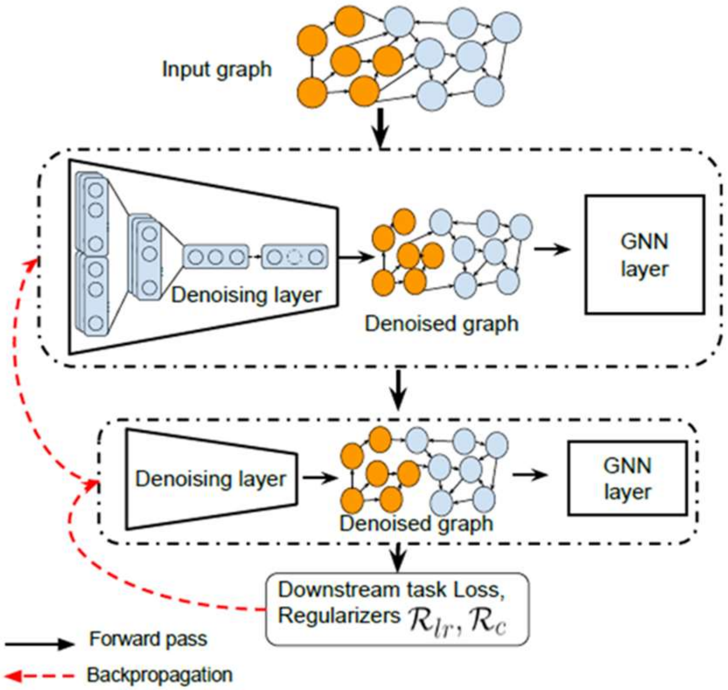
- **Model Agnostic**
- **Task Agnostic**



Graph Denoising: PTDNet

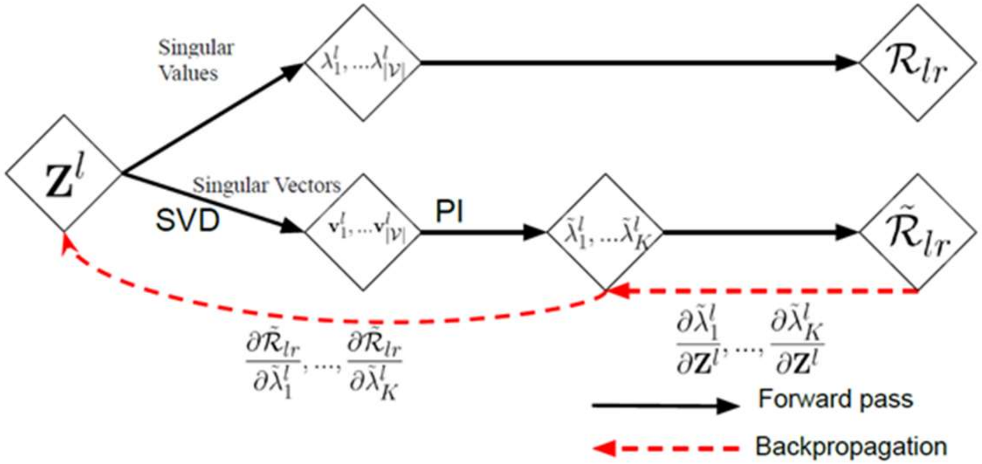
- **The denoising network**

In each denoising network, parameterized neural networks are adopted to learn a weight for each edge. Then we adopt hard concrete distribution to ensure that an edge weight can be exactly 0



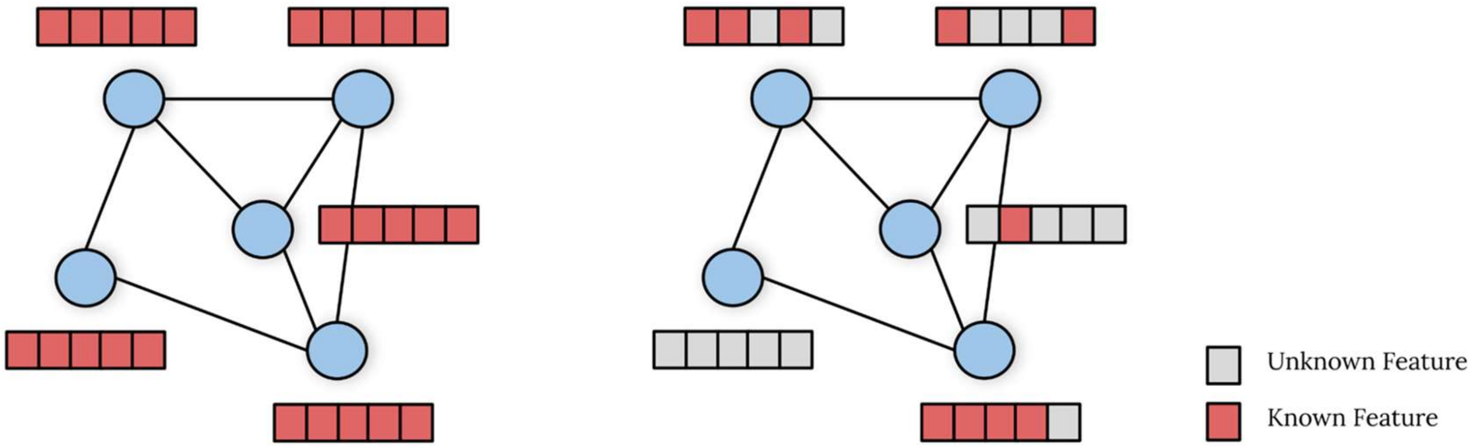
$$\epsilon \sim \text{Uniform}(0, 1), \quad s_{u,v}^l = \sigma((\log \epsilon - \log(1 - \epsilon) + \alpha_{uv}^l)/\tau)$$

- **The low-rank constraint**



Graph Denoising: Feature Propagation

- Most GNNs expect as input a graph with a full feature vector for each node (left). In real-world scenario, **only some of the features are available** (right).



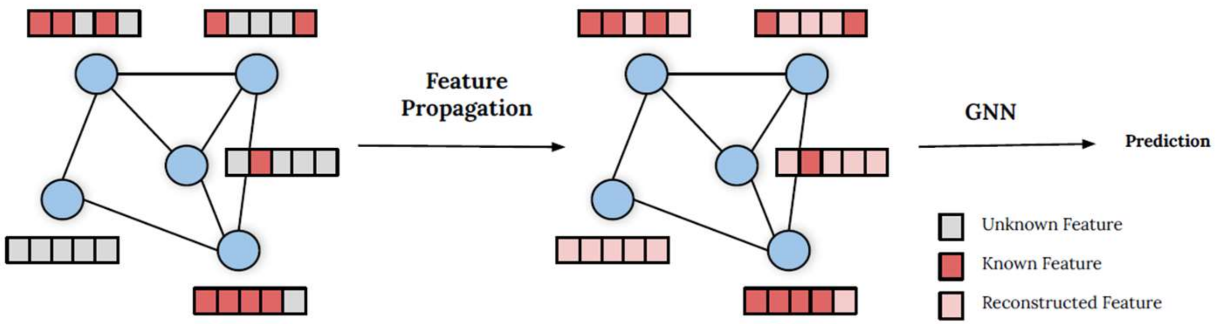
Graph with all features observed

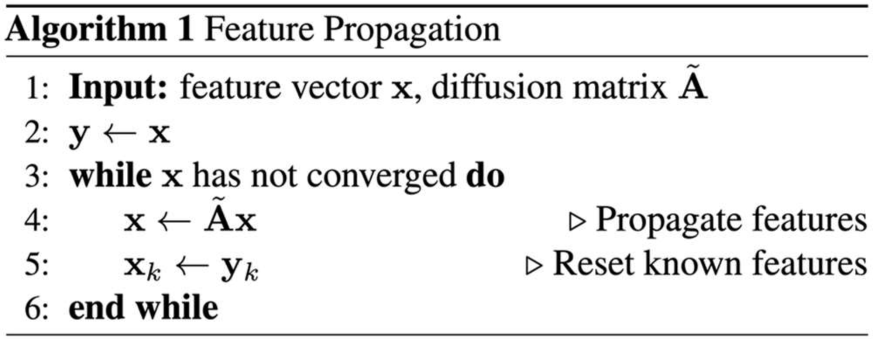
Graph with partially missing features

On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features. ICLR 2021

Graph Denoising: Feature Propagation

- Feature Propagation is a simple and surprisingly powerful approach for learning on graphs **with missing features**.





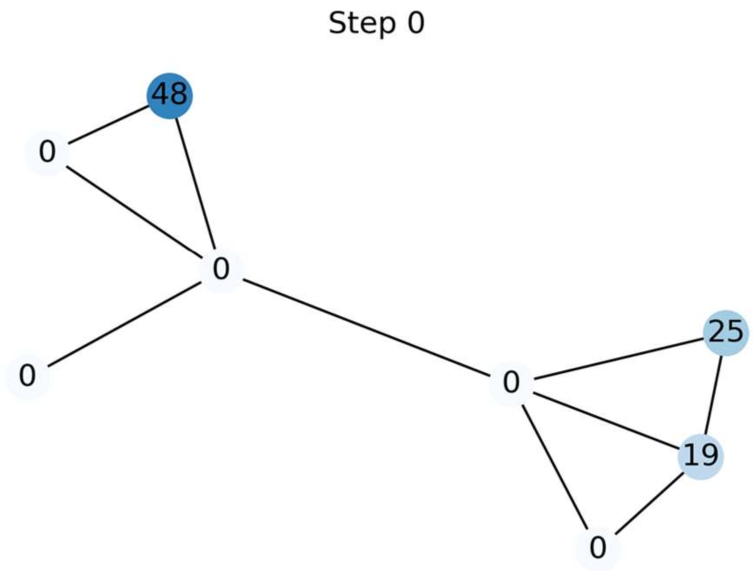
Algorithm 1 Feature Propagation

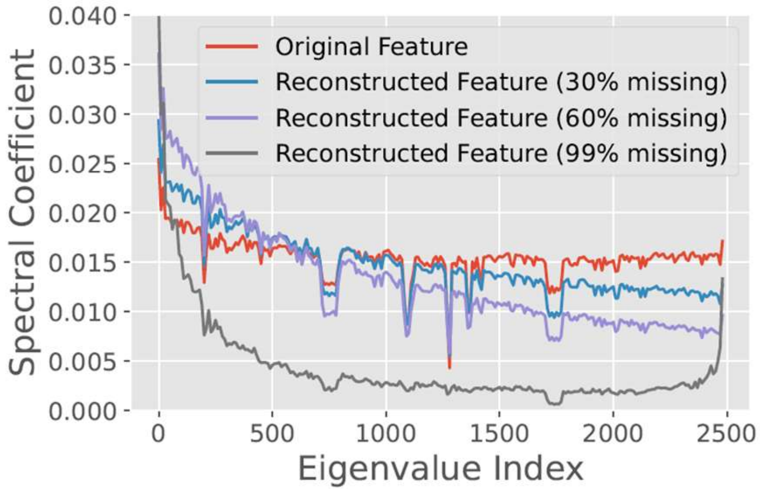
```
Input: feature vector $\mathbf{x}$, diffusion matrix $\tilde{\mathbf{A}}$
$\mathbf{y} \leftarrow \mathbf{x}$
while $\mathbf{x}$ has not converged do
    $\mathbf{x} \leftarrow \tilde{\mathbf{A}}\mathbf{x}$            ▷ Propagate features
    $\mathbf{x}_k \leftarrow \mathbf{y}_k$            ▷ Reset known features
end while
```

On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features. ICLR 2021

Graph Denoising: Feature Propagation

- The reconstruction of missing features by FP at different steps **(left).**
- Graph Fourier transform magnitudes of the original Cora features (red) and those reconstructed by FP for varying rates of missing rates **(right)**.

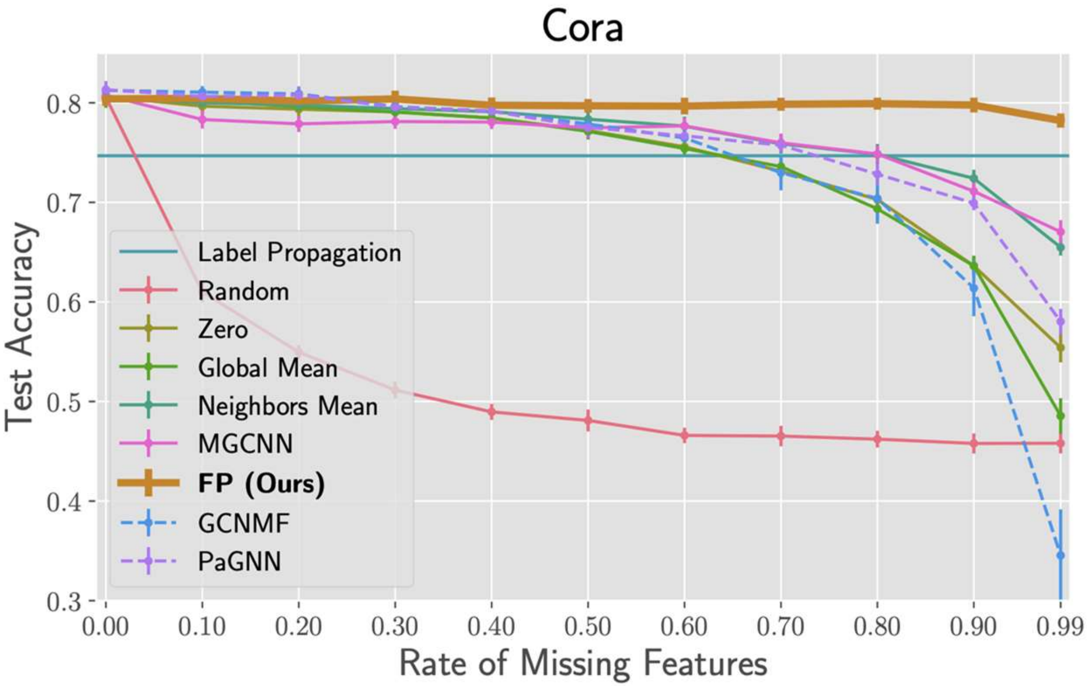




On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features. ICLR 2021

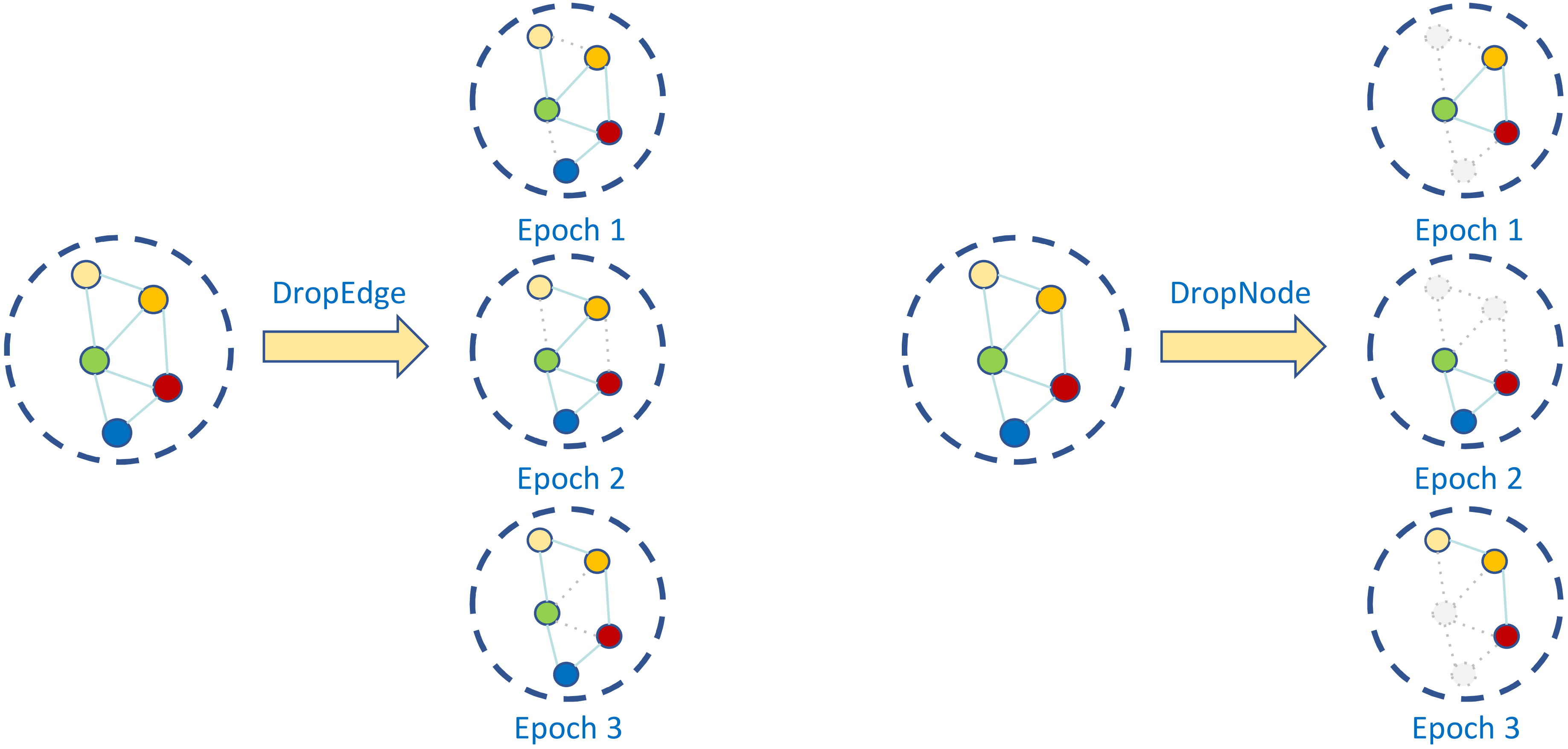
Graph Denoising: Feature Propagation

- Node classification accuracy for varying rates of missing features on the Cora dataset.



On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features. ICLR 2021

Regularization: DropEdge & DropNode



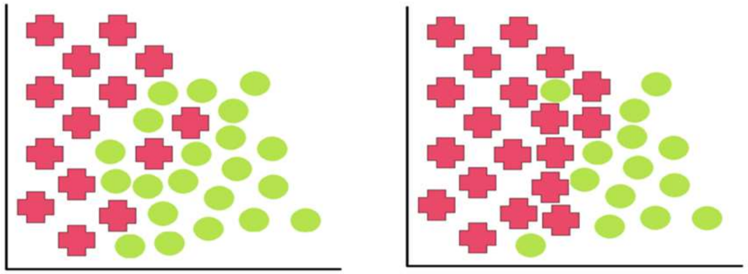
- DropEdge: Towards deep graph convolutional networks on node classification. ICLR 2020
- Graph Contrastive Learning with Augmentations, NeurIPS 2020

Reliability of GNNs

- Overview
- GNNs against inherent noise
  - Threat Overview
  - Enhancing Techniques
- GNNs against distribution shift
  - Threat Overview
  - Enhancing Techniques
- GNNs against adversarial attacks
  - Threat Overview
  - Enhancing Techniques
- Toolbox

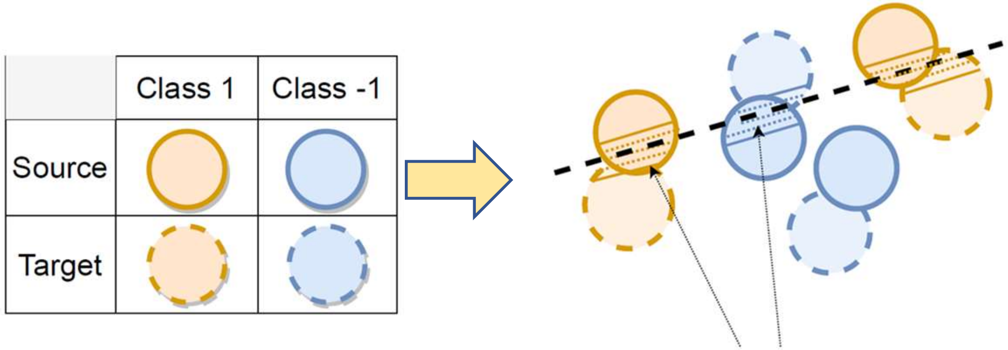
Threat Overview: Distribution Shift

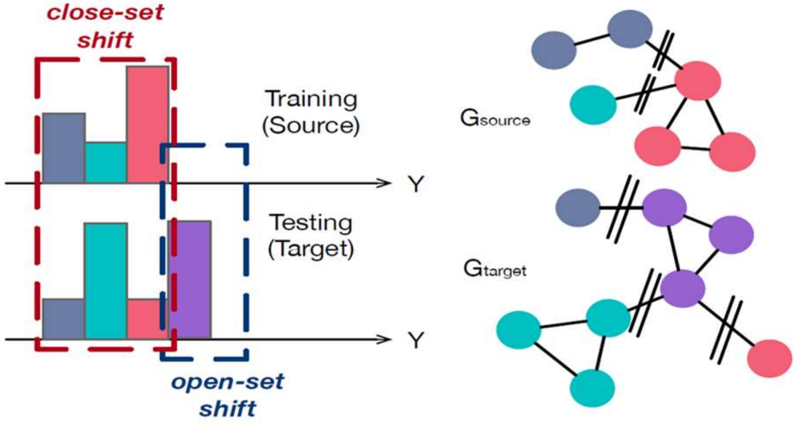
- Distribution shift appears when training and test joint distributions are different. That is, when $P_{train}(\mathcal{G}, Y) \neq P_{test}(\mathcal{G}, Y)$



Training

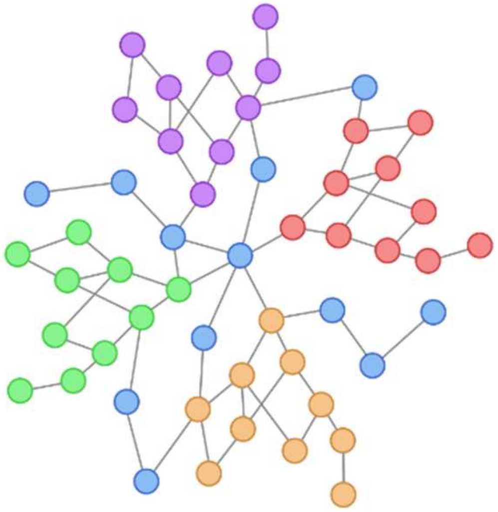
Test





Threat Overview: Distribution Shift

- Why does the distribution shift occur?
  - **Non-IID bias**
  - **Sampling bias**



Robust Graph Neural Networks. Google AI Blog

Defending Against Distribution Shift

Enhancing techniques on graph data can be categorized as:

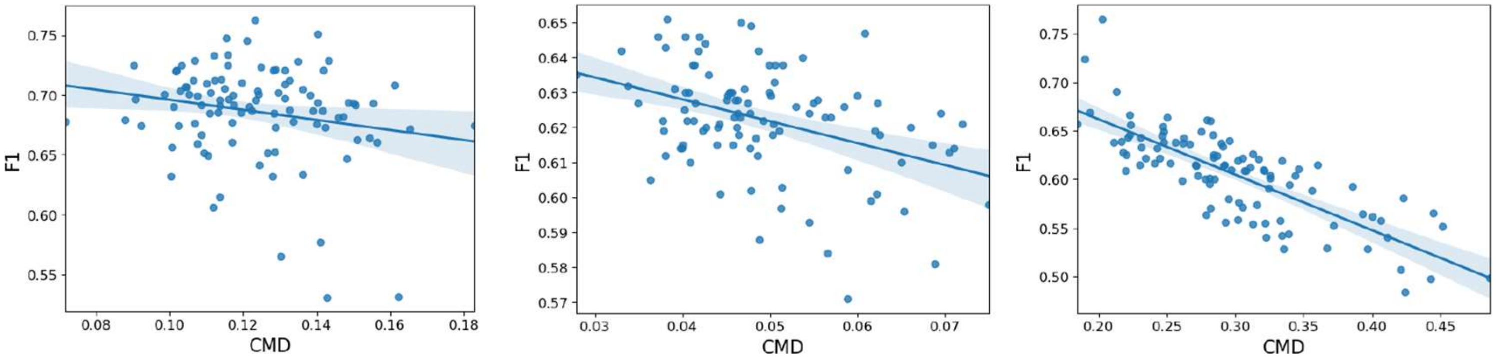
- Invariant Learning
- Graph Augmentation Technique

Invariant Learning: Shift-Robust GNNs

Negative effect of distribution shifts

- Distribution shift (CMD) between training and testing data could be a good indicator of performance (F1)
- As the distribution shift increases, the model's accuracy falls.

$$\mathrm{CMD} = \frac{1}{|b-a|}\|\mathrm{E}(p) - \mathrm{E}(q)\|_2 + \sum_{k=2}^{\infty} \frac{1}{|b-a|^k}\|c_k(p) - c_k(q)\|_2,$$



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

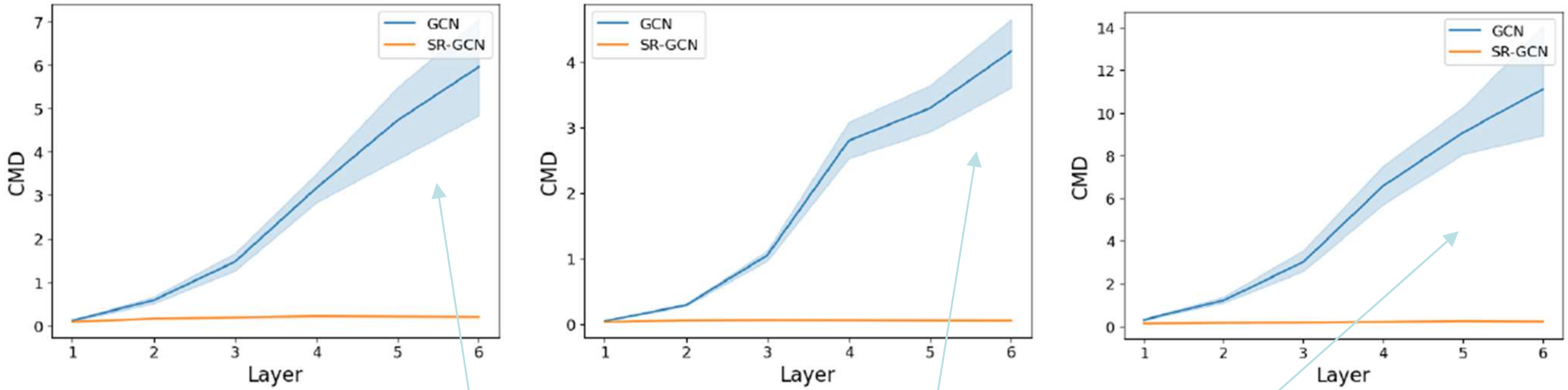
Shift-Robust GNNs: Overcoming the Limitations of Localized Graph Training Data. NeurIPS 2021

Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. ICLR 2017

Invariant Learning: Shift-Robust GNNs

More Motivation

- Distribution shift is also a problem in deeper GNNs



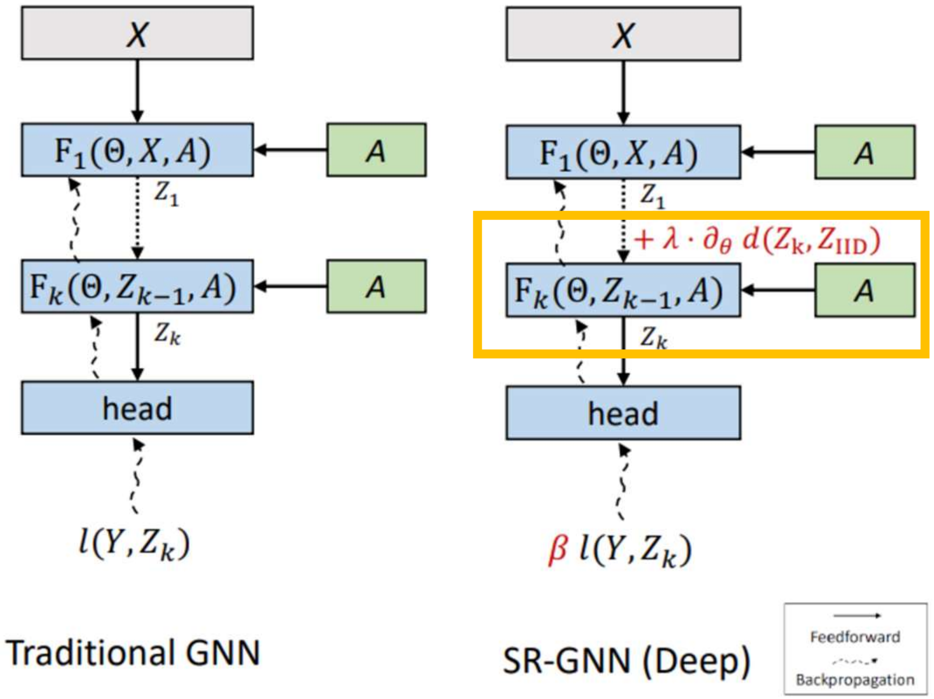
(a) Cora (b) Citeseer (c) Pubmed

Shift gets bigger as models get deeper!

Shift-Robust GNNs: Overcoming the Limitations of Localized Graph Training Data. NeurIPS 2021

Invariant Learning: Shift-Robust GNNs

Shift-Robust GNNs



- **Solution**: Regularizations to make GNNs robust against domain shift.
- **Normal GNN** - Fully differentiable deep models allow applying domain shift regularization at any layer

- We can regularize a layer in this network to force the features to be representative for both a biased and unbiased samples:

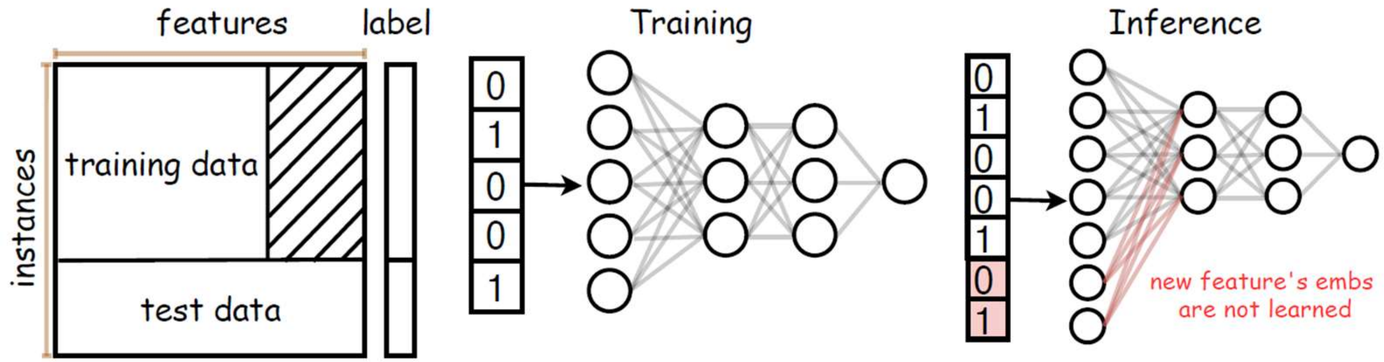
$$\mathcal{L} = \frac{1}{M}\sum_i l(y_i, z_i) + \lambda \cdot d(Z_{\text{train}}, Z_{\text{IID}}).$$

Shift-Robust GNNs: Overcoming the Limitations of Localized Graph Training Data. NeurIPS 2021

Invariant Learning: FATE (FeATure Extrapolation Networks)

Challenges and Limitations of Neural Networks

- New features dynamically appear (unseen features in test set)
  - $P_{train}(X,Y) \neq P_{test}(X,Y)$
  - Scenarios: heterogeneous data sources, multi-modal data
- How can neural networks deal with new features?
  - Retraining from scratch: **time-consuming**
  - Incremental learning on new features: **over-fitting & catastrophic forgetting**

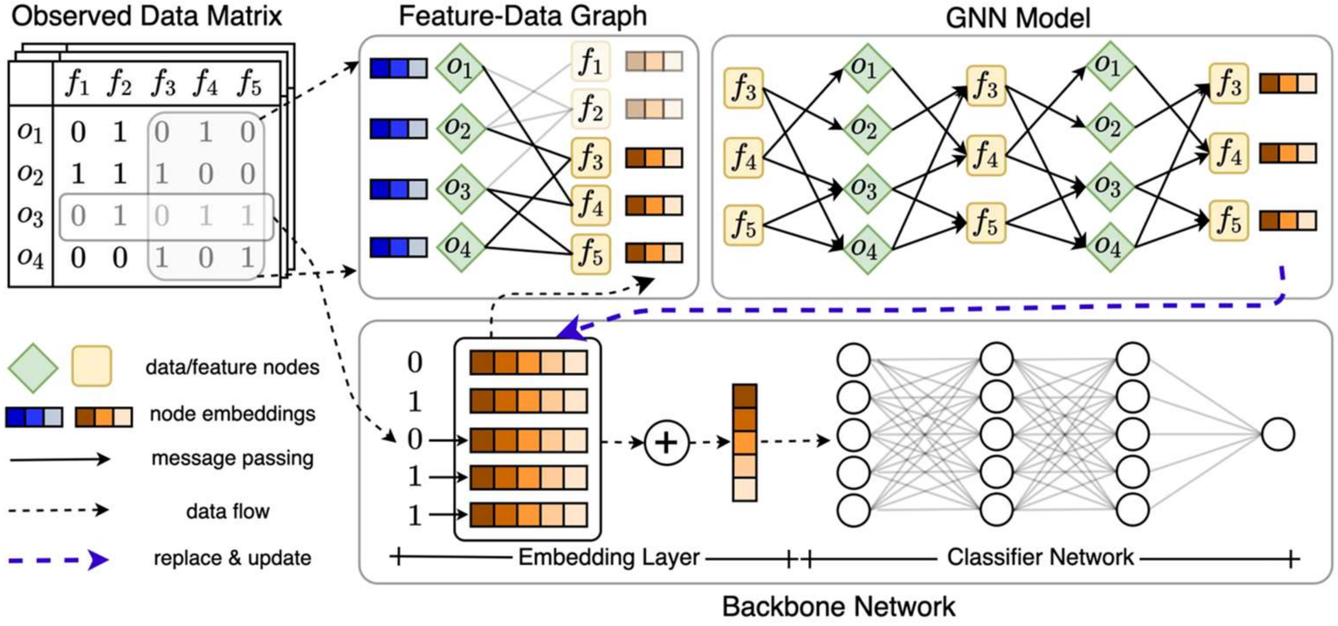


Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach. NeurIPS 2021

Invariant Learning: FATE (FeATure Extrapolation Networks)

Overall Framework: FATE

- **Low-level backbone**: take each instance as input and output prediction
- **High-level GNN**: take feature-data matrix as input and update feat. embeddings

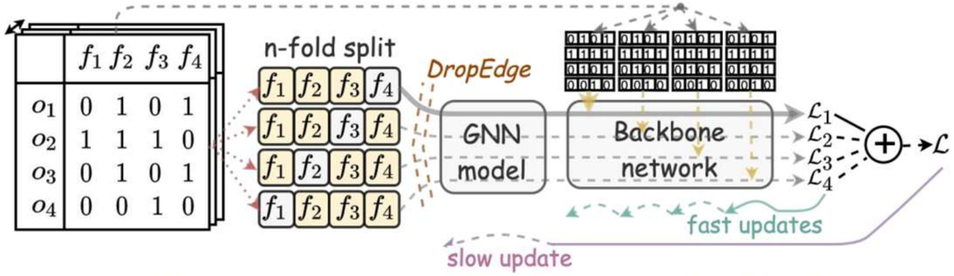


Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach. NeurIPS 2021

Invariant Learning: FATE (FeATure Extrapolation Networks)

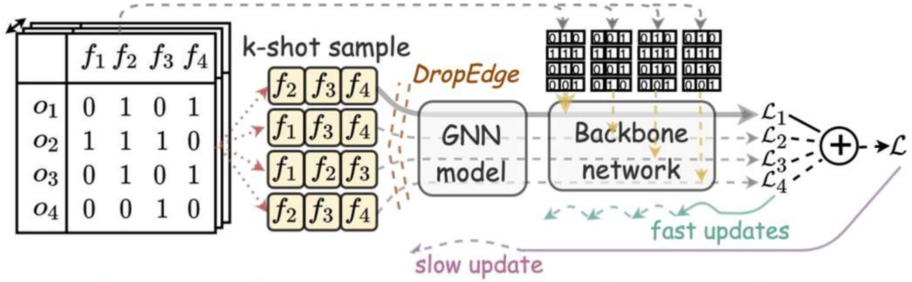
Training Approach

- Two useful techniques for learning to extrapolate
  - Proxy training data: Self-supervised learning and inductive learning
  - Asynchronous Updates: Fast/slow for backbone/GNN
- DropEdge regularization
- Scaling to large systems: Mini-batches along the instance dimension (complexity $O(Bd)$)



	f_1	f_2	f_3	f_4
o_1	0	1	0	1
o_2	1	1	1	0
o_3	0	1	0	1
o_4	0	0	1	0

(a) *Self-supervised learning with n-fold splitting*



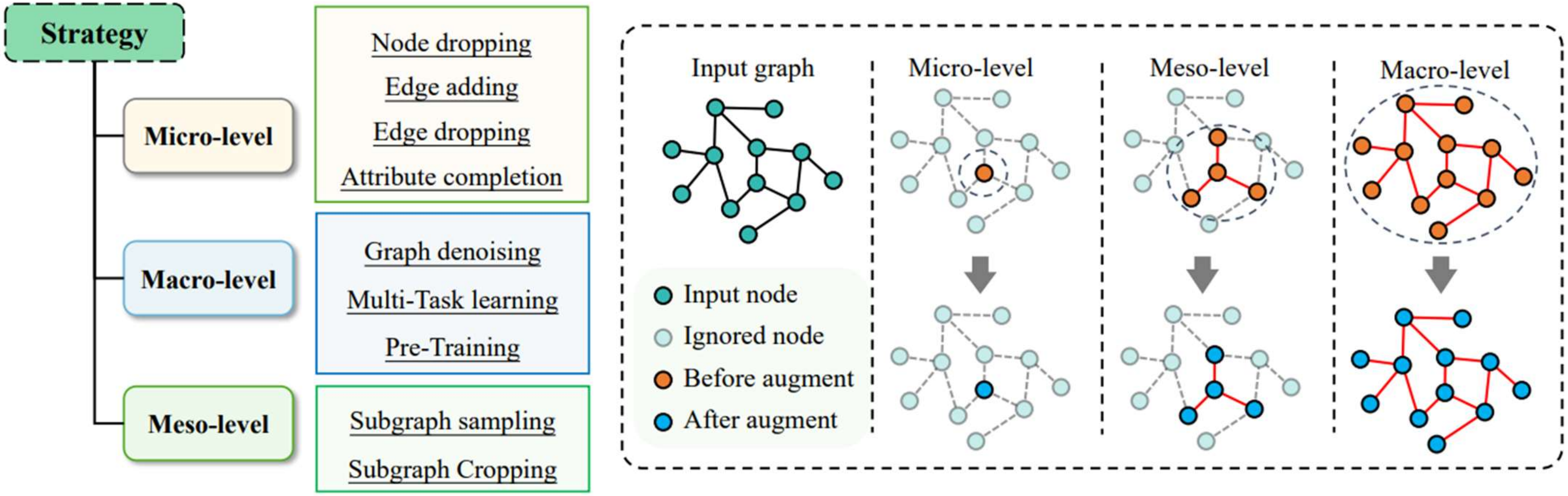
	f_1	f_2	f_3	f_4
o_1	0	1	0	1
o_2	1	1	1	0
o_3	0	1	0	1
o_4	0	0	1	0

(b) *inductive learning with k-shot sampling*

Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach. NeurIPS 2021

Graph Augmentation Learning

- Graph Augmentation Learning helps models generalize to out-of-distribution samples and boosts model performance at test time.

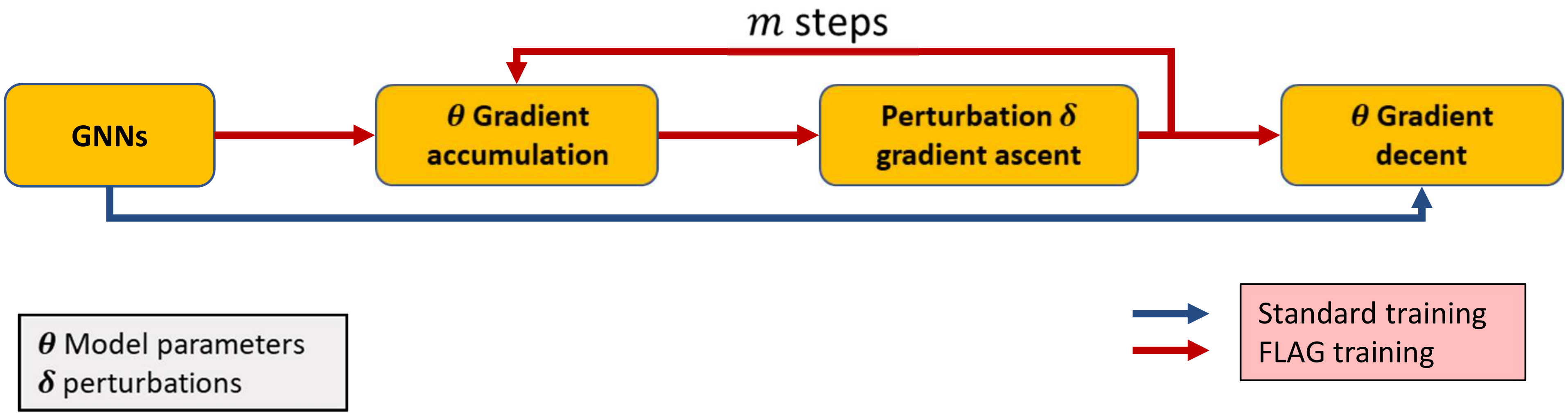


Graph Augmentation Learning. WWW 2022

Graph Augmentation Learning: FLAG

FLAG: Free Large-scale Adversarial Augmentation on Graphs

- Improve model generalization via adversarial training



m steps

GNNs

θ Gradient accumulation

Perturbation δ gradient ascent

θ Gradient decent

θ Model parameters
δ perturbations

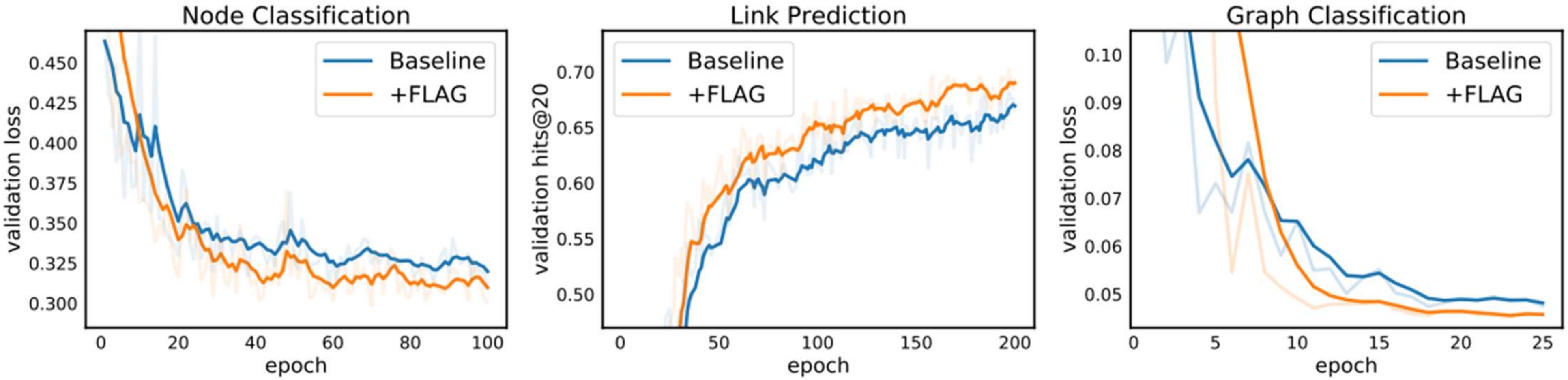
Standard training
FLAG training

Robust Optimization as Data Augmentation for Large-scale Graphs. CVPR 2022

Graph Augmentation Learning: FLAG

FLAG: Free Large-scale Adversarial Augmentation on Graphs

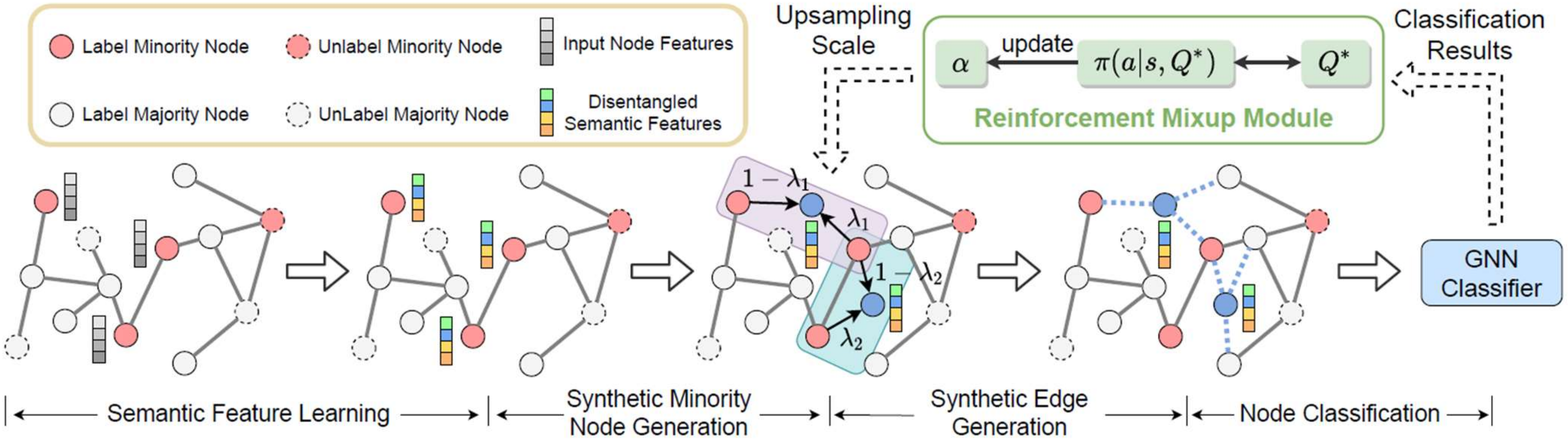
- FLAG can address overfitting problem and improves model robustness against out-of-distribution samples



Robust Optimization as Data Augmentation for Large-scale Graphs. CVPR 2022

Graph Augmentation Learning: GraphMixup

- Class-imbalanced is another distribution shift where $P_A(\mathcal{G}, \boldsymbol{Y}) \neq P_B(\mathcal{G}, Y)$, A and B are two groups
- GraphMixup improves class-imbalanced node classification on graphs by self-supervised context prediction

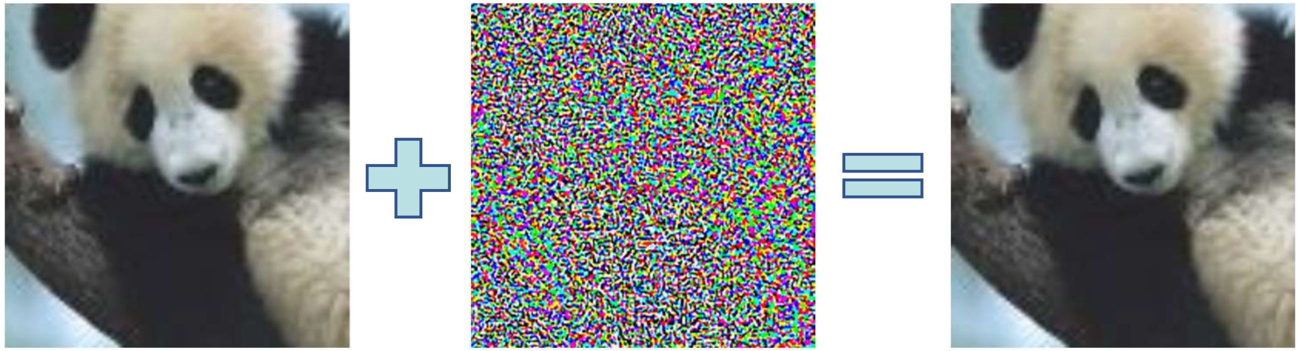


GraphMixup: Improving Class-Imbalanced Node Classification on Graphs by Self-supervised Context Prediction. ICML 2021

Reliability of GNNs

- Overview
- GNNs against inherent noise
  - Threat Overview
  - Enhancing Techniques
- GNNs against distribution shift
  - Threat Overview
  - Enhancing Techniques
- GNNs against adversarial attacks
  - Threat Overview
  - Enhancing Techniques
- Toolbox

Adversarial Attacks on Deep Learning



Classified as panda

x

Small adversarial noise

ϵ

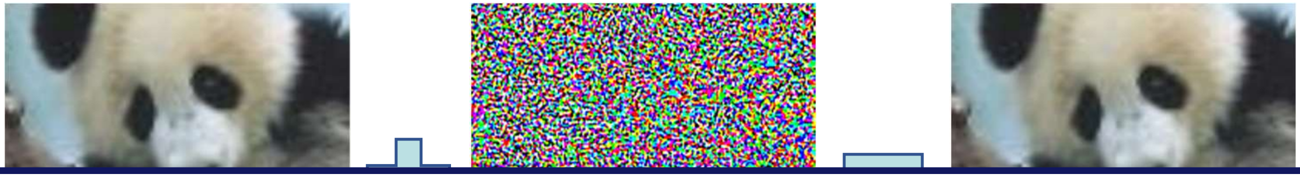
Classified as gibbon

x'

Find x' satisfying $||x' - x|| \leq \Delta$ s.t. $C(x') \neq y$

Explaining and harnessing adversarial examples. ICLR 2015

Adversarial Attacks on Deep Learning



Do Graph Neural Networks Suffer the Same Problem?

Classified as panda

Small adversarial noise

Classified as gibbon

x

ϵ

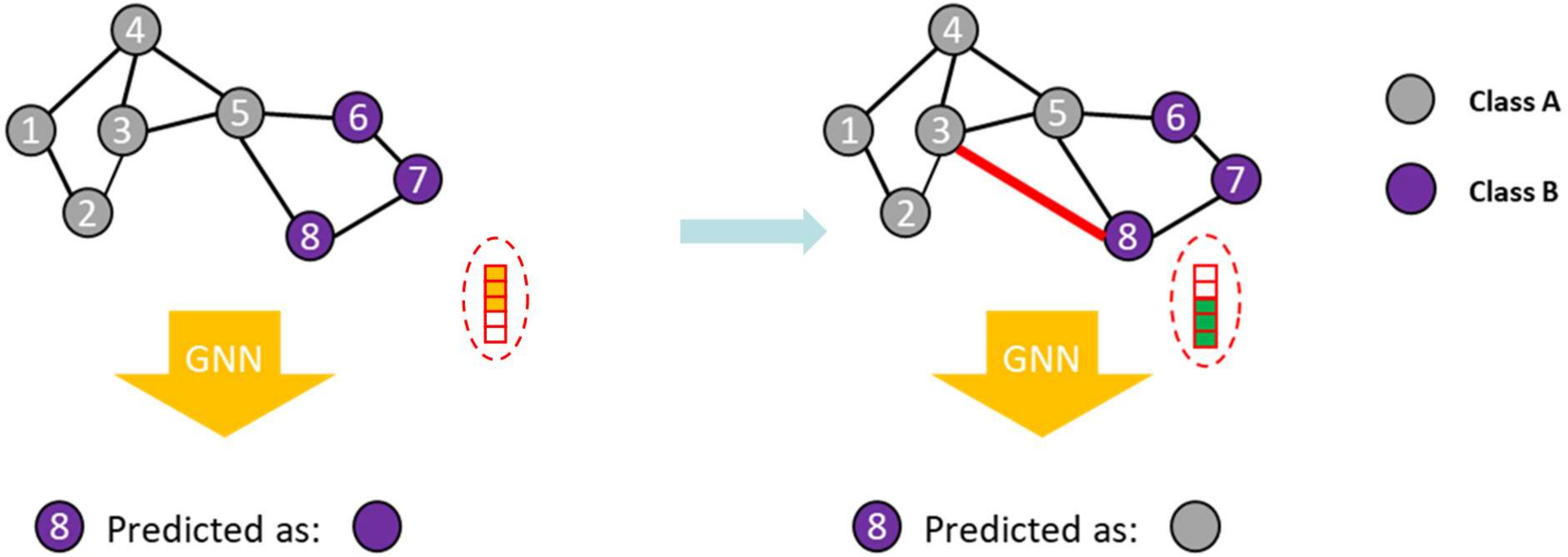
x'

Find x' satisfying $||x' - x|| \leq \Delta$ s.t. $C(x') \neq y$

Explaining and harnessing adversarial examples. ICLR 2015

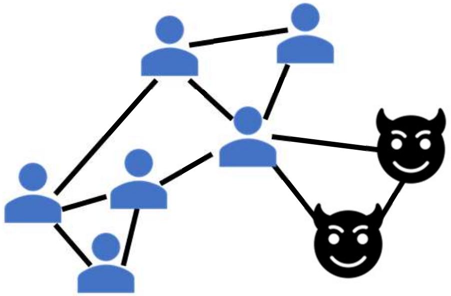
Adversarial Attacks on Deep Graph Learning

- Adversarial attacks on GNNs aims to change their prediction by modifying the edges or features



Adversarial Attacks on Deep Graph Learning

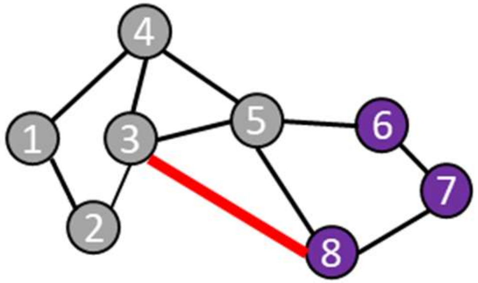
Consequences



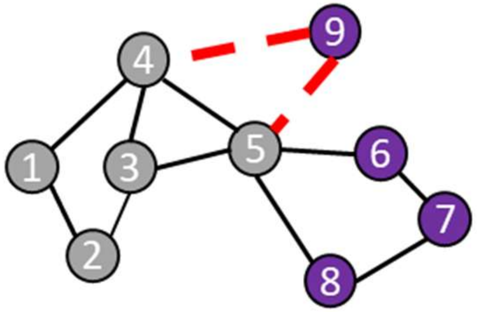
- Financial Systems
  - Credit Card Fraud Detection
- Recommender Systems
  - Social Recommendation
  - Product Recommendation
- ….

Types of Perturbations

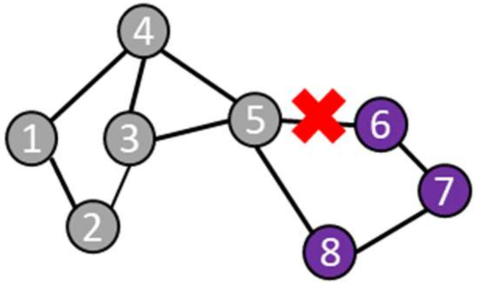
Edge Insertion



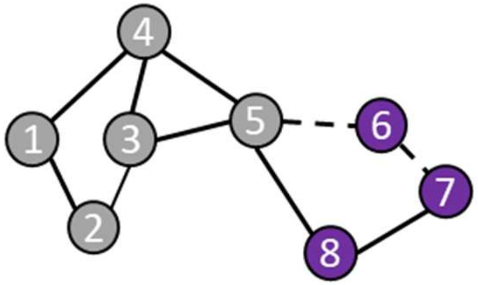
Node Injection



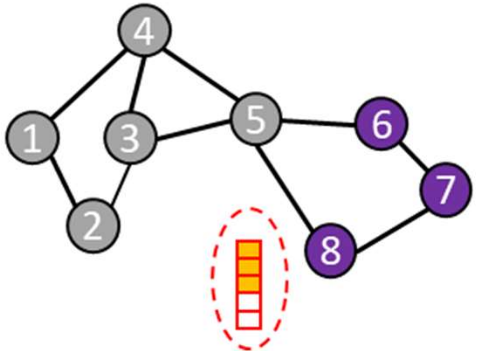
Edge Deletion



Node Deletion

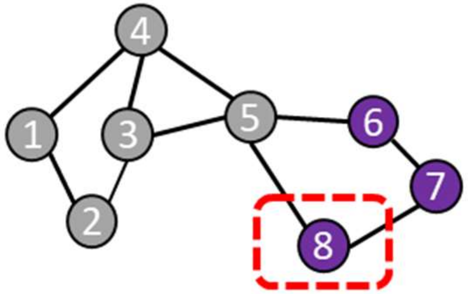


Feature Modification



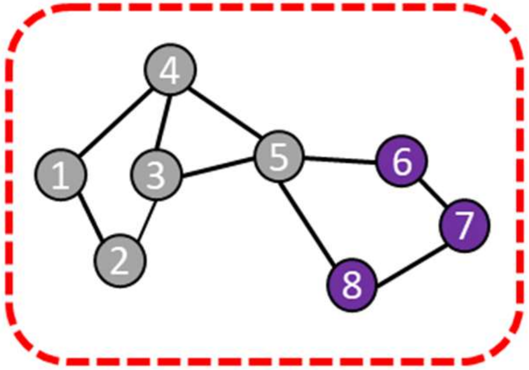
Types of Attacking Type

Targeted Attack



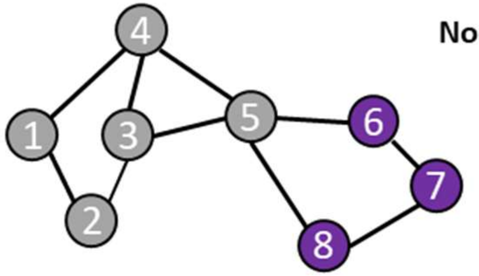
8 Target Node

Non-Targeted Attack



All nodes as candidates

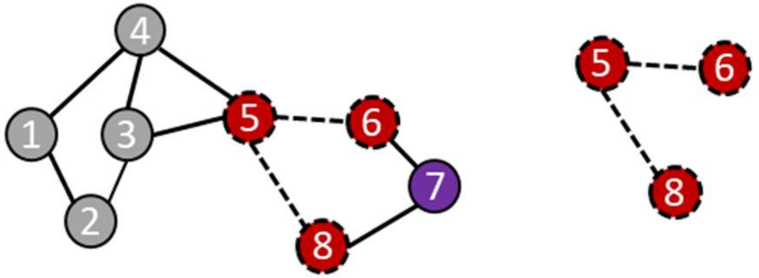
Universal Attack



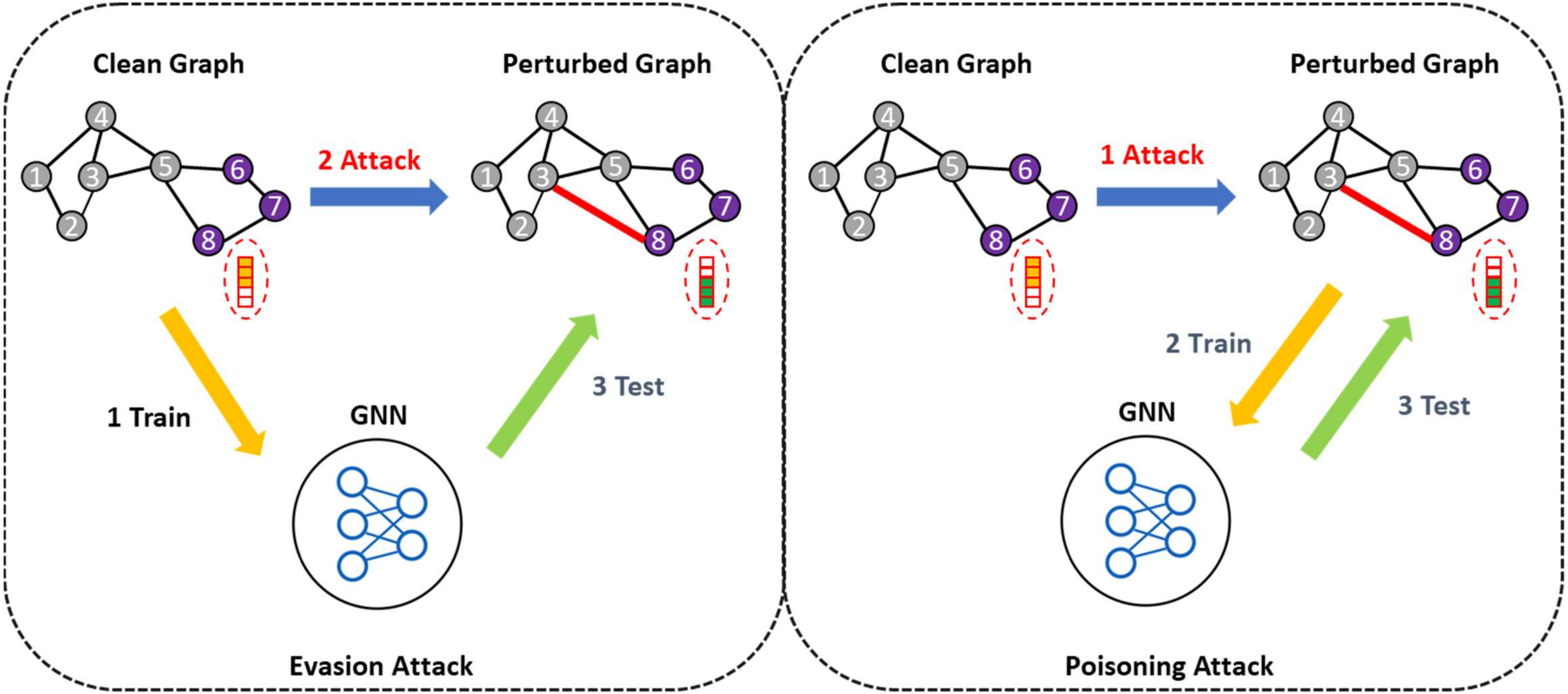
Node agnostic

? Any Target Node

Backdoor Attack



Evasion & Poisoning Attack



Attack Methods

Attack Methods	Node Injection	Edge Insertion/ Deletion	Feature Modification	Targeted	Non-Targeted	Evasion	Poisoning	Universal	Backdoor
Nettack (KDD 18)		✓	✓	✓		✓	✓		
Metattack (ICLR 19)		✓			✓		✓		
GF-Attack (AAAI 20)		✓		✓		✓			
CD-Attack (WWW 20)		✓		✓			✓		
GUA (IJCAI 21)		✓		✓			✓	✓	
GTA (USENIX 21)		✓			✓	✓	✓		✓

Attack Methods

Attack Methods	**Node Injection**	**Edge Insertion/ Deletion**	**Feature Modification**	**Targeted**	**Non-Targeted**	**Evasion**	**Poisoning**	**Universal**	**Backdoor**
Nettack (KDD 18)		✓	✓	✓		✓	✓		
Metattack (ICLR 19)		✓			✓		✓		
GF-Attack (AAAI 20)		✓		✓		✓			
CD-Attack (WWW 20)		✓		✓			✓		
GUA (IJCAI 21)		✓		✓			✓	✓	
GTA (USENIX 21)		✓			✓	✓	✓		✓

Nettack (KDD 18)

- General form of graph adversarial attack as a bi-level optimization problem (poisoning setting):

Prediction of attacked model

$$\arg\max_{\hat{A},\hat{X}} \mathcal{L}_{atk}\left(f_{\theta^*}(A',X')\right) = \sum_{u\in V_t} \ell\left(\boxed{f_{\theta^*}(A',X')_u},\boxed{c_{old,u}}\right)$$

Prediction of clean model

$$s.t. \quad \theta^* = arg\min_{\theta} \ \mathcal{L}_{train}(f_\theta(A',X'))\ ,|A'-A| + |X'-X| \leq \Delta$$

- **Types of attack**:
  - Targeted attack, Poison/Evasion Attack, White-Box Attack
  - Structure and feature modifications

A: adjacent matrix
X: node features matrix
A': modified structure
X': modified feature
V_t: set of target nodes
$c_{old,u}$: the predicted class label of the clean model.
Δ: perturbation budget

- **Core idea**: Establishing a linear surrogate model:

$$f_{S,\theta}(A,X) = \text{softmax}\left(\hat{A}\ \cancel{ReLU}(\hat{A}X\Theta^{(1)})\Theta^{(2)}\right) = \text{softmax}(\hat{A}^2X\boxed{\Theta})$$

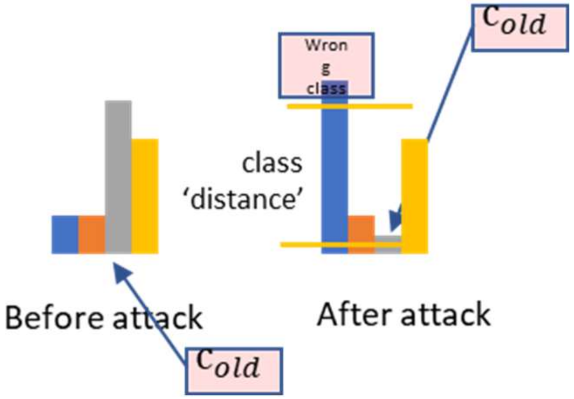
$\Theta^{(1)}\Theta^{(2)}$, trained on the clean data.

[KDD18] Adversarial Attacks on Neural Networks for Graph Data

Nettack (KDD 18)

Optimization approach: one perturbation that maximizes the class 'distance' of the surrogate model $f_{S,\theta}(A,X)$ before and after attack.

$$\ell\left(f_{S,\theta^*}(A,X)\,,c_{old,u}\right) = \max_{c\neq c_{old}} Z(u)_c - Z(u)_{\,c_{old}} \quad Z = f_{S,\theta^*}(A,X)$$



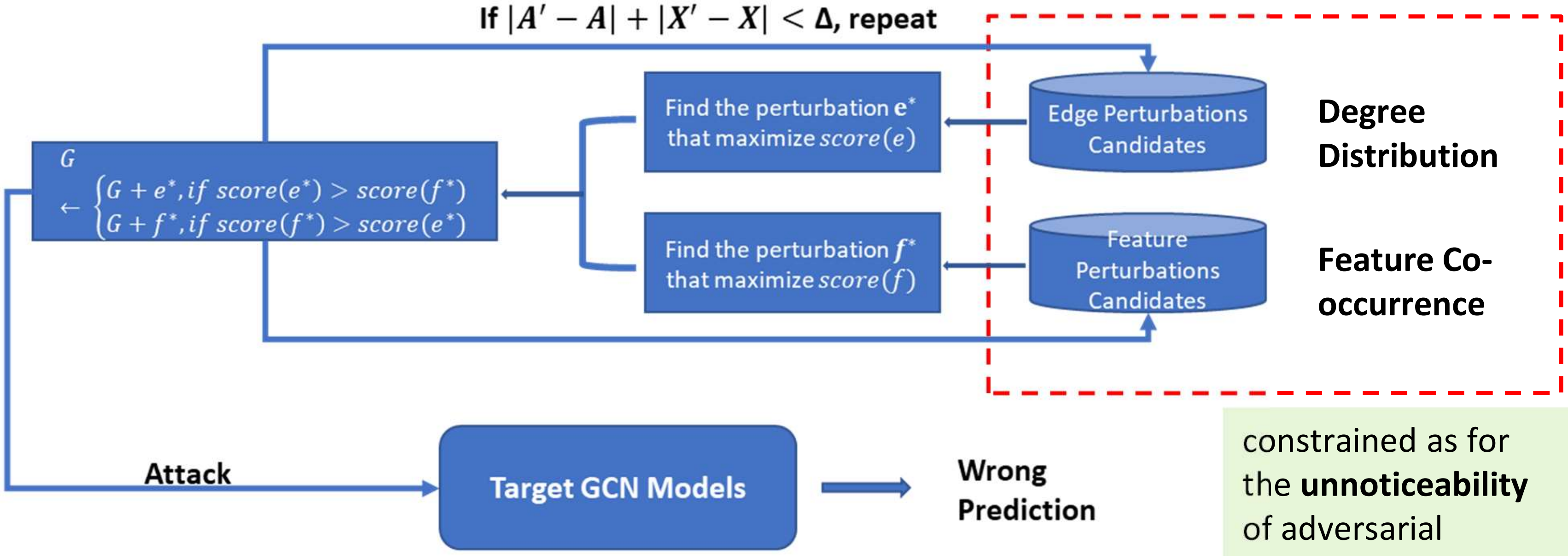
Attack on edge: $score(e) = \ell\left(f_{S,\theta^*}(A',X)\,,c_{old,u}\right)$, $A' := A \pm e$

Attack on feature: $score(f) = \ell\left(f_{S,\theta^*}(A,X')\,,c_{old,u}\right)$, $X' := X \pm f$

[KDD18] Adversarial Attacks on Neural Networks for Graph Data

Nettack (KDD 18)

Optimization approach: one perturbation that maximizes the class 'distance' of the surrogate model $f_{S,\theta}(A, X)$ before and after attack.



If $|A' - A| + |X' - X| < \Delta$, repeat

$G \leftarrow \begin{cases} G + e^*, if\ score(e^*) > score(f^*) \\ G + f^*, if\ score(f^*) > score(e^*) \end{cases}$

Find the perturbation $\mathbf{e}^*$ that maximize $score(e)$

Find the perturbation f^* that maximize $score(f)$

Edge Perturbations Candidates

Feature Perturbations Candidates

Degree Distribution

Feature Co-occurrence

Attack

Target GCN Models

Wrong Prediction

constrained as for the **unnoticeability** of adversarial samples.

Overview

[KDD18] Adversarial Attacks on Neural Networks for Graph Data

GF-Attack (AAAI 20)

Graph Embedding Models	Graph-shift filter S	Polynomial Function $h(x)$
GCN	$L^{sym} - I_n$	$h(x) = x$
SGC	$L^{sym} - I_n$	$h(x) = x$
ChebyNet	$L^{sym} - I_n$	$h(x) = \sum_{k=0}^{K} T_k\,(x)$
LINE	$I_n - L^{rw}$	$h(x) = x$
DeepWalk	$I_n - L^{rw}$	$h(x) = \sum_{k=0}^{K} x^k$

S: graph-shift filter;
h(x): polynomial function used for constructing graph filter $\mathcal{H}$.

Reliability of GNNs

- Overview
- GNNs against inherent noise
  - Threat Overview
  - Enhancing Techniques
- GNNs against distribution shift
  - Threat Overview
  - Enhancing Techniques
- GNNs against adversarial attacks
  - Threat Overview
  - Enhancing Techniques
- Toolbox

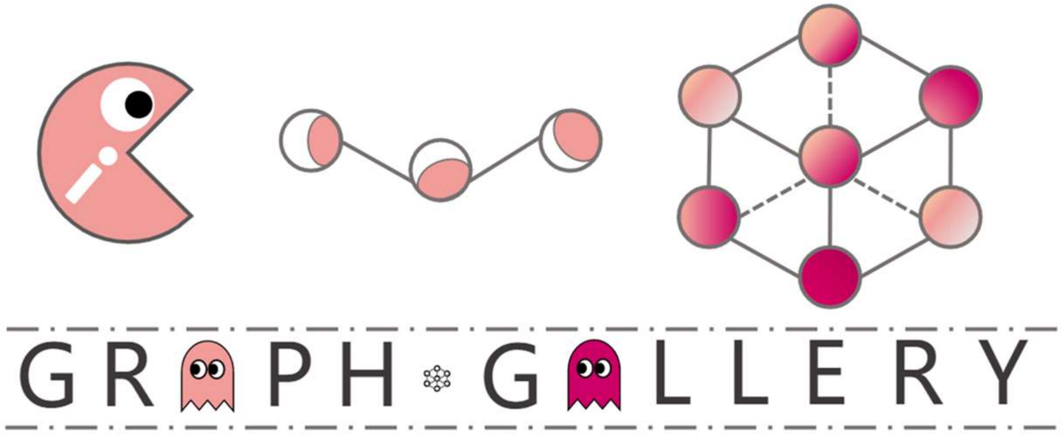
Deep Graph Learning Toolbox

- **DrugOOD**: OOD benchmarks for drug design
- **GreatX**: PyTorch based graph reliability toolbox

GraphGallery

GraphGallery: PyTorch based GNN model gallery

Link: https://github.com/EdisonLeeeee/GraphGallery



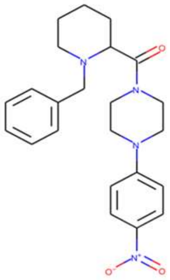
PyTorch is all you need!

DrugOOD

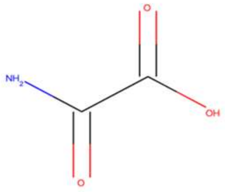
- Paper: https://arxiv.org/pdf/2201.09637.pdf
- Code: https://github.com/tencent-ailab/DrugOOD
- Project: https://drugood.github.io/

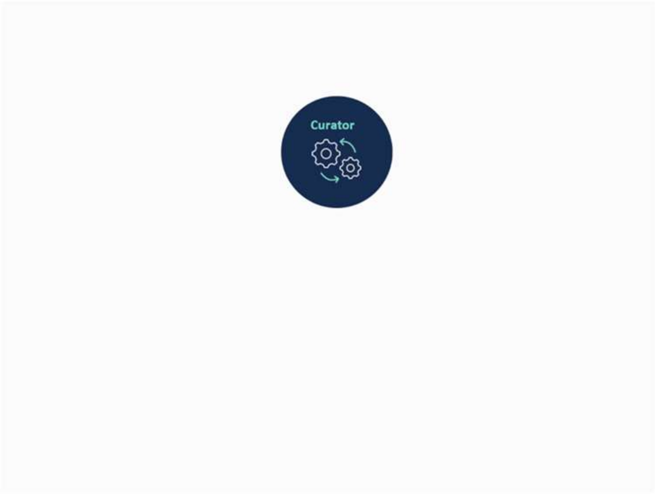
Distribution shift in Drug AI

Training



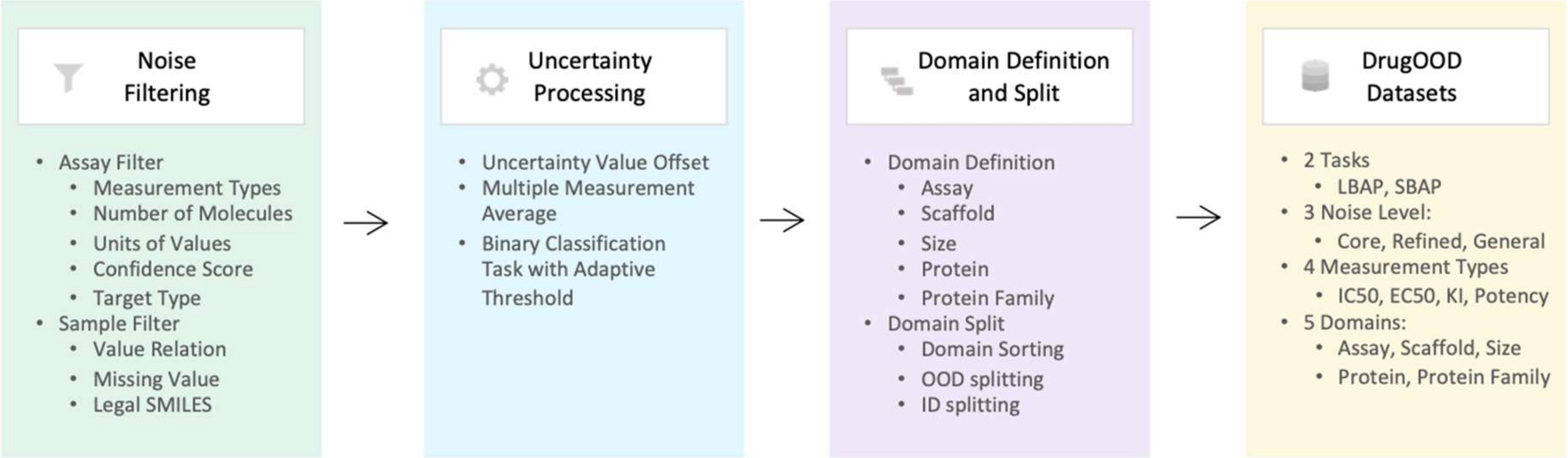
Testing





Overview of DrugOOD

- Automated OOD Dataset Curator with Real-world Domain and Noise Annotations
  - Five domain definitions (scaffold, assay, molecule size, protein, protein family) reflect the real distribution offset scenarios. Three noise levels (core, refined, general) can anchor different noise levels



Noise Filtering

- Assay Filter
  - Measurement Types
  - Number of Molecules
  - Units of Values
  - Confidence Score
  - Target Type
- Sample Filter
  - Value Relation
  - Missing Value
  - Legal SMILES

→

Uncertainty Processing

- Uncertainty Value Offset
- Multiple Measurement Average
- Binary Classification Task with Adaptive Threshold

→

Domain Definition and Split

- Domain Definition
  - Assay
  - Scaffold
  - Size
  - Protein
  - Protein Family
- Domain Split
  - Domain Sorting
  - OOD splitting
  - ID splitting

→

DrugOOD Datasets

- 2 Tasks
  - LBAP, SBAP
- 3 Noise Level:
  - Core, Refined, General
- 4 Measurement Types
  - IC50, EC50, KI, Potency
- 5 Domains:
  - Assay, Scaffold, Size
  - Protein, Protein Family

Config example

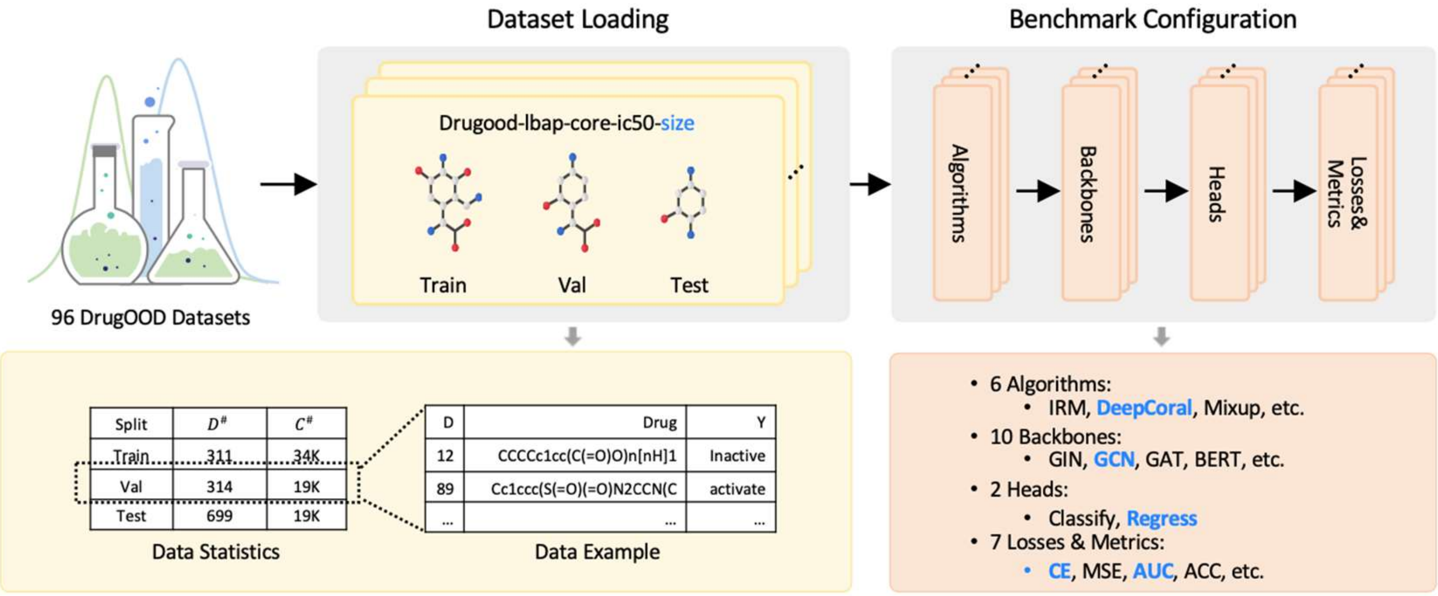
- Automated OOD Dataset Curator
    - Fully customizable for users.
    - 96 realized datasets are provided

Curation configuration example

```
# configure curation pipeline
curator = dict(
    path = dict(
        task=dict(type=’lbap’,
        subset=’lbap_core_ec50_assay’),
        source_root=’data/chembl_29.db’,
        target_root=’data/’)
    uncertainty = dict(
        delta=dict({’<’: -1, \
            ’<=’: -1, ’>’: 1, ’>=’: 1}))
    classification_threshold = dict(
        lower_bound=4,
        upper_bound=6,
        fix_value=5)
    fractions = dict(
        train_fraction_ood=0.6,
        val_fraction_ood=0.2,
        iid_train_sample_fractions=0.6,
        iid_val_sample_fractions=0.2)
    noise_filter = dict(
        assay=dict(
            measurement_type=[’EC50’],
            assay_value_units=[’nM’, ’uM’],
            molecules_number=[50, 3000],
            confidence_score=9),
        sample=dict(
            filter_none=[],
            smile_exist=[],
            smile_legal=[],
            value_relation=[’=’, ’~’]))
    domain = dict(
        domain_generate_field=’assay_id’,
        domain_name=’assay’,
        sort_func=’domain_capacity’,
        sort_order=’descend’,
        protein_family_level=1)
)
```

Robust Optimization Baseline

- Rigorous OOD benchmarking
  - Six SOTA OOD algorithms with various backbones



Split	$D^\#$	$C^\#$
Train	311	34K
Val	314	19K
Test	699	19K

Data Statistics

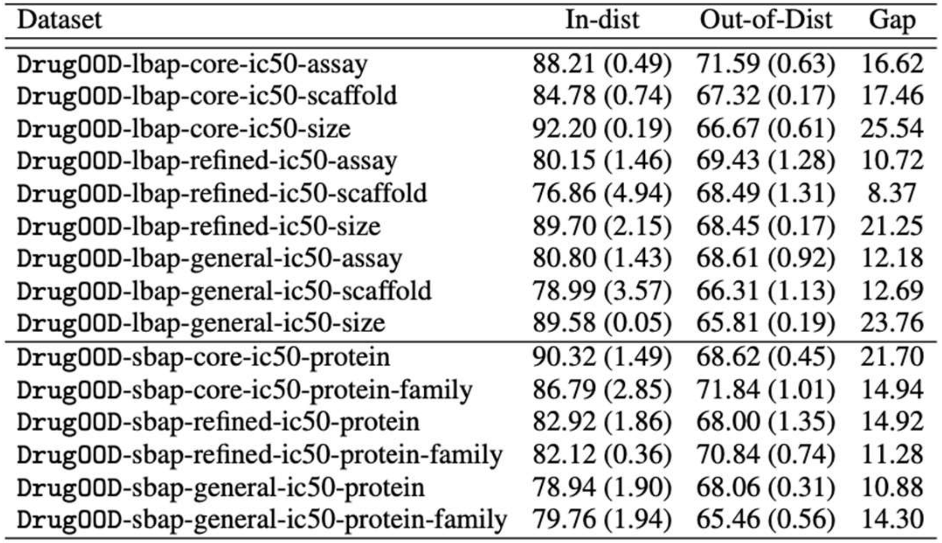
D	Drug	Y
12	CCCCc1cc(C(=O)O)n[nH]1	Inactive
89	Cc1ccc(S(=O)(=O)N2CCN(C	activate
...	...	...

Data Example

- 6 Algorithms:
  - IRM, DeepCoral, Mixup, etc.
- 10 Backbones:
  - GIN, GCN, GAT, BERT, etc.
- 2 Heads:
  - Classify, Regress
- 7 Losses & Metrics:
  - CE, MSE, AUC, ACC, etc.

Predefined Benchmark

The benchmark tests revealed that the in-distribution out-of-distribution (ID-OOD) classification performance (AUC score) on DrugOOD datasets by more than 20%, verifying the authenticity and challenge of the domain definition and noise calibration methods in this dataset.



Dataset	In-dist	Out-of-Dist	Gap
DrugOOD-lbap-core-ic50-assay	88.21 (0.49)	71.59 (0.63)	16.62
DrugOOD-lbap-core-ic50-scaffold	84.78 (0.74)	67.32 (0.17)	17.46
DrugOOD-lbap-core-ic50-size	92.20 (0.19)	66.67 (0.61)	25.54
DrugOOD-lbap-refined-ic50-assay	80.15 (1.46)	69.43 (1.28)	10.72
DrugOOD-lbap-refined-ic50-scaffold	76.86 (4.94)	68.49 (1.31)	8.37
DrugOOD-lbap-refined-ic50-size	89.70 (2.15)	68.45 (0.17)	21.25
DrugOOD-lbap-general-ic50-assay	80.80 (1.43)	68.61 (0.92)	12.18
DrugOOD-lbap-general-ic50-scaffold	78.99 (3.57)	66.31 (1.13)	12.69
DrugOOD-lbap-general-ic50-size	89.58 (0.05)	65.81 (0.19)	23.76
DrugOOD-sbap-core-ic50-protein	90.32 (1.49)	68.62 (0.45)	21.70
DrugOOD-sbap-core-ic50-protein-family	86.79 (2.85)	71.84 (1.01)	14.94
DrugOOD-sbap-refined-ic50-protein	82.92 (1.86)	68.00 (1.35)	14.92
DrugOOD-sbap-refined-ic50-protein-family	82.12 (0.36)	70.84 (0.74)	11.28
DrugOOD-sbap-general-ic50-protein	78.94 (1.90)	68.06 (0.31)	10.88
DrugOOD-sbap-general-ic50-protein-family	79.76 (1.94)	65.46 (0.56)	14.30

Table 6: The in-distribution (ID) vs out-of-distribution (OOD) of datasets with measurement type of IC50 trained with ERM. We adopt the AUROC to estimate model performance; the higher score is better. All datasets show performance drops due to distribution shift, with substantially better ID performance than OOD performance.

GreatX

GreatX: PyTorch based graph reliability toolbox

Link: https://github.com/EdisonLeeeee/GreatX

Graph REliAbility
ToolboX

GreatX is great!