Unraveling Graph Counterfactual Explainability: from Theoretical Foundations to Technical Mastery

Tutorial @ ECML-PKDD 2025

B. Prenkaj - M.A. Prado-Romero - A. D'Angelo - E. Zaradoukas - G. Stilo

Artificial Intelligence & Information Mining

Bardh Prenkaj
PostDoc

Efstratios Zaradoukas
PhD. Fellow

Mario A. Prado-Romero
ML Researcher

Andrea D'Angelo
PhD. Fellow

Giovanni Stilo
Associate Prof.

Technische Universität München

GRAN SASSO TECH

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

LUISS

We are with the *Artificial Intelligence & Information Mining* (aiim - pronounced as i'm /aɪm/, and aim /eɪm/) a collective of *Individuals* (/aɪm/) who share a common *Interest* (/eɪm/) in Artificial Intelligence, Data Mining, and Machine Learning

Roadmap (1)

- **Part I: Introduction and Background (20 mins)** [STILO]
  - The rise of Graph Neural Networks (GNNs) in AI
  - Fundamental principles: How do GNNs work?
  - Applications across domains:
    - Traffic Modeling
    - Physical simulations
    - Protein interaction networks
    - Large-scale recommender systems
  - Key challenges in understanding GNN predictions

- **Part II: Explainability in Graphs (30 mins)** [PRENKAJ]
  - The black-box problem in GNNs and its implications
  - Overview of Explainable AI (XAI) techniques for graphs
  - Factual explanations vs. Counterfactual explanations
  - A deep dive into existing factual explainers:
    - GNNExplainer: Strengths and limitations
    - GraphLIME: Local interpretability in GNNs
    - Other noteworthy post-hoc explanation methods

Roadmap (2)

- **Part III: Fundamentals of Graph Counterfactual Explainability (45 mins)** [PRADO]
  - Defining counterfactual explanations in the graph domain
  - Why counterfactuals? Advantages over factual explanations
  - Challenges in generating counterfactuals for graphs:
    - Structural constraints in graphs
    - Ensuring plausibility and feasibility of counterfactuals
    - Computational complexity concerns
  - Taxonomy of GCE methods:
    - *Instance-level explainers*
    - *Model-level explainers*
- **Part IV: Counterfactual Explainability in Evolving Graphs (30 mins)** [PRENKAJ]
  - What makes dynamic graphs different?
    - Temporal dependencies and evolving structures
    - Distribution shifts and their impact on GNN predictions
  - Challenges in applying counterfactual explanations to dynamic graphs:
    - Stability and consistency of counterfactuals over time
    - Identifying actionable interventions in dynamic environments
    - Handling streaming data and online learning settings
  - Existing approaches for counterfactual explainability in dynamic graphs:
    - Adapting instance-based explainers to evolving graphs
    - Extending model-level explanations for temporal settings
    - Learning-based techniques for robust counterfactuals in dynamic graphs

Roadmap (3)

- **Part V: Evaluating Graph Counterfactual Explanations (25 mins)** [ZARADOUKAS]
  - Overview of benchmarking datasets:
    - Synthetic vs. real-world graph datasets
    - Domain-specific datasets: Social networks, molecular graphs, and financial transactions
  - Evaluation metrics and assessment frameworks:
    - Validity and plausibility of counterfactuals
    - Proximity and sparsity constraints
    - Fidelity to the underlying GNN model
  - Discussion on robustness and fairness in GCE methods

- **Part VI: Frameworks In-the-Wild (45 mins)** [D'ANGELO]
  - Introduction to the challenges of developing and evaluating GCE methods
  - GRETEL's design, core components, and their interaction
  - Basic explanation pipeline and evaluation
  - Frameworks for Discrete Time Explanations

- **Part VII: What's next? (20 mins)** [PRENKAJ]
  - Open discussion on the future trends and development in the research/industry area

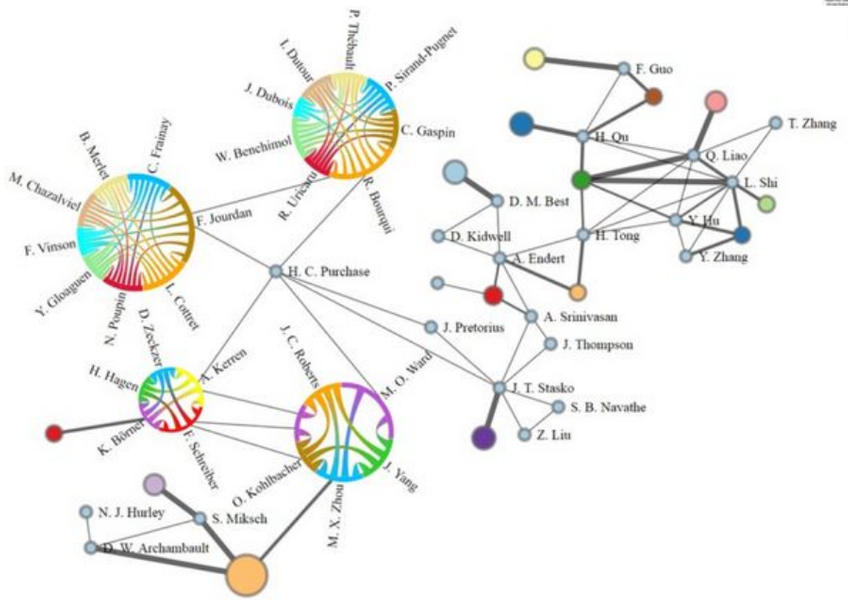
Part I

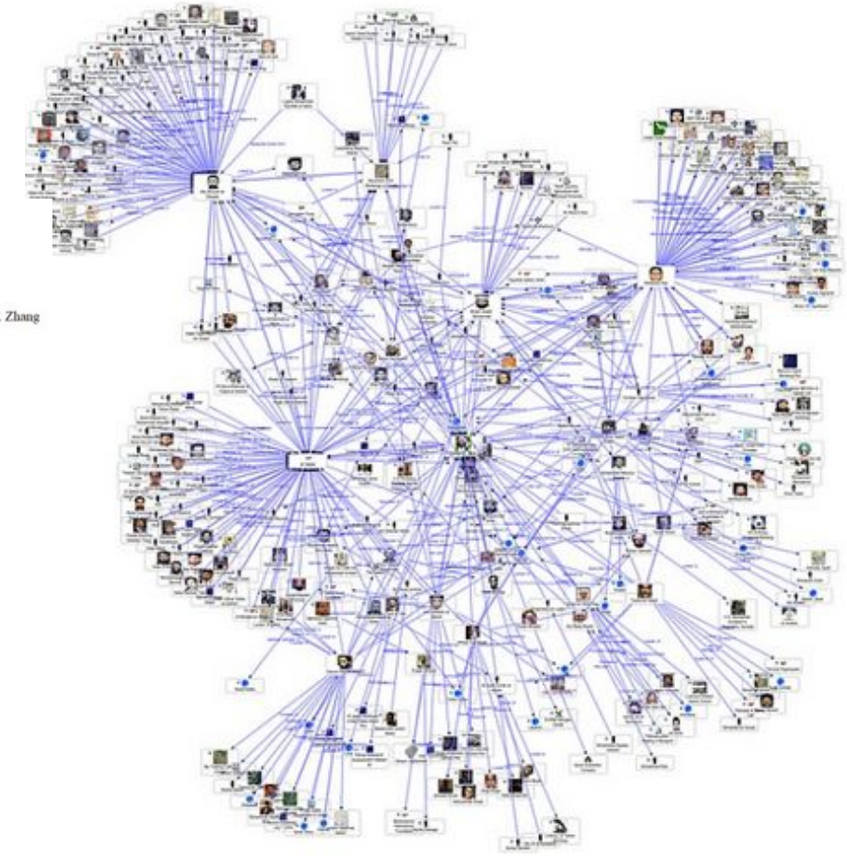
Graphs fundamentals and their Neural Network

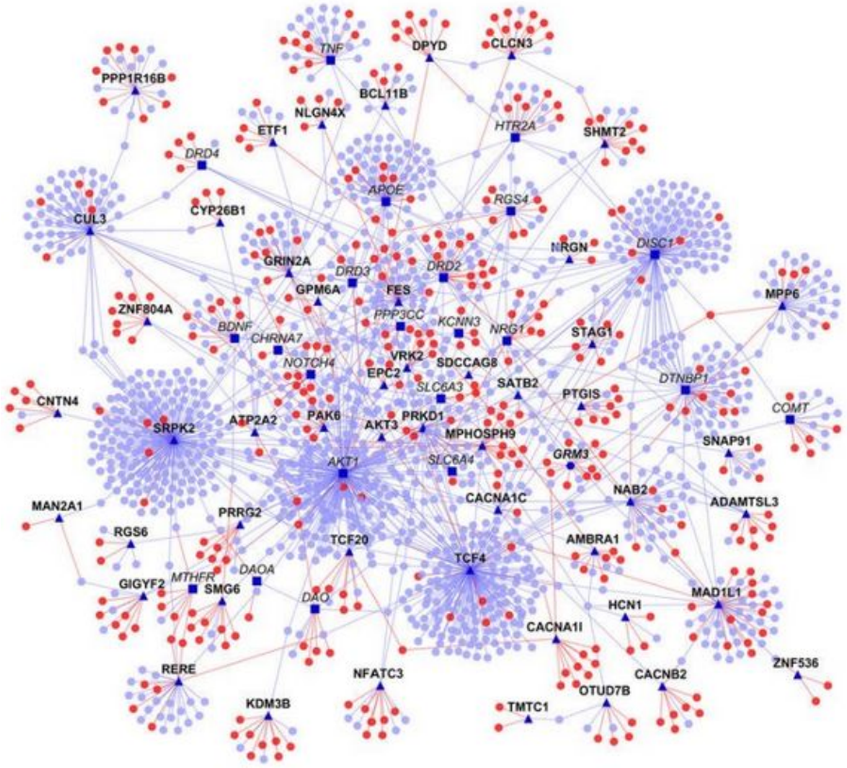
by Giovanni Stilo

images based on Understanding Deep Learning - book by Simon J.D. Prince

What is a graph?







Co-Authorship Networks

Social Networks

Proteins interactions

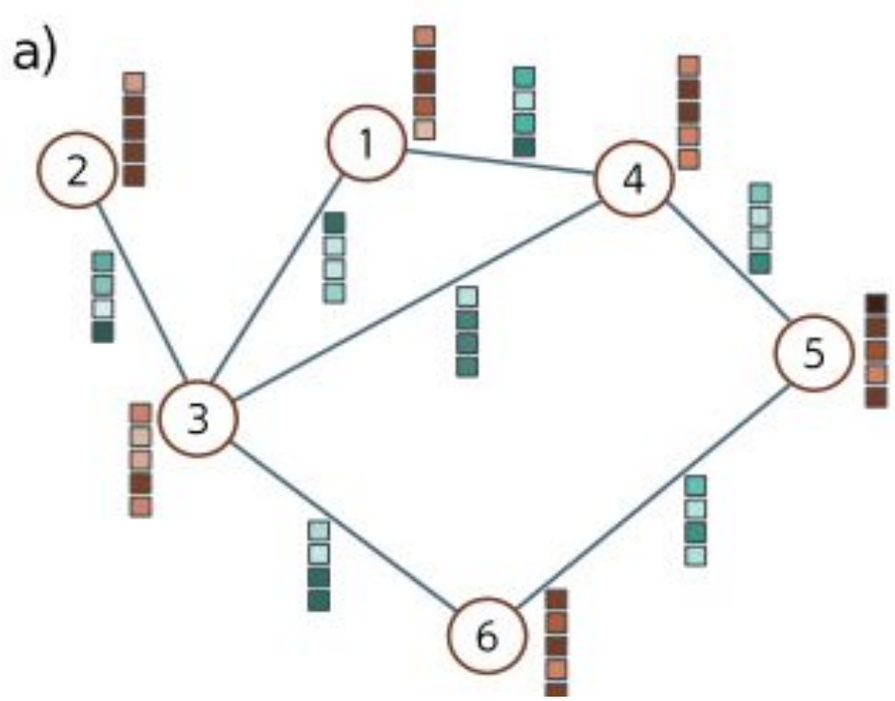
Challenges with graphs

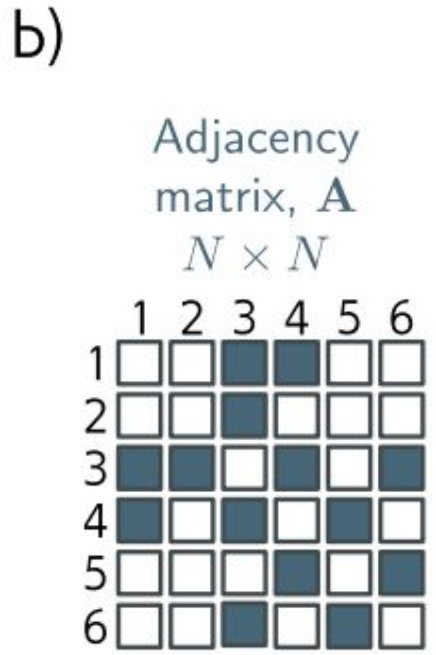
There are **three main challenges** associated with processing graphs:

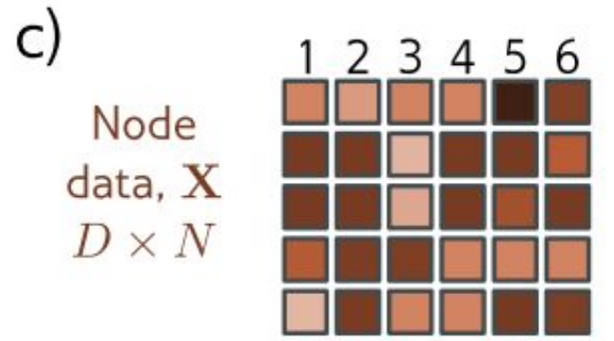
- **Variable topology:** hard to design an NN that is sufficiently expressive and can cope with this variation
- **Huge graphs:** we can have millions of nodes and billions of edges (see Twitter)
- **Single monolithic graph:** the usual protocol of training with many data examples and testing with new data is not always appropriate or possible

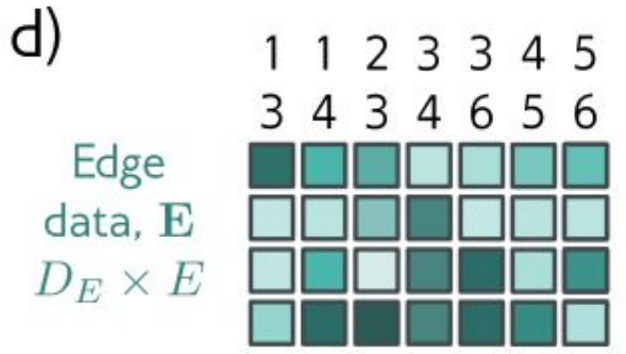
Graph Representation

adjacency matrix, A is $N \times N$;
node embeddings, X is $D \times N$;
edges embeddings, E is $D_E \times N$



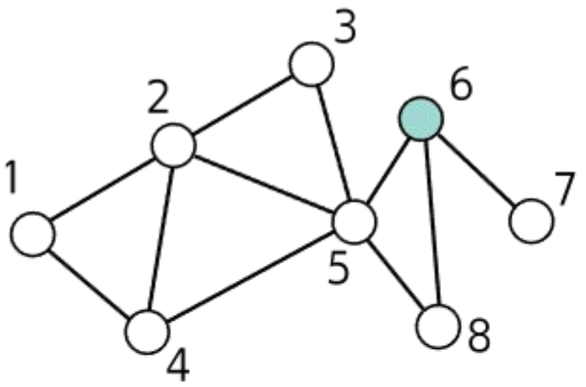






Adjacency Matrix Properties

a)



b)

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

c)

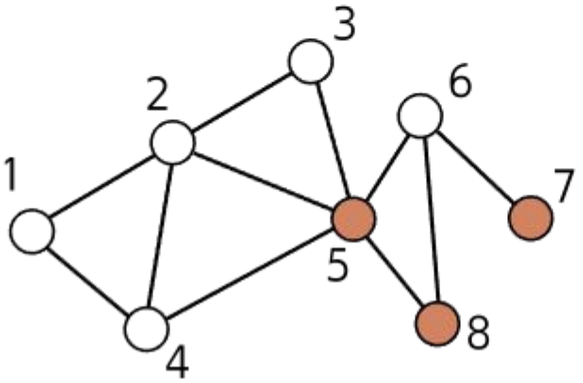
$$\mathbf{A}^2 = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 4 & 1 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 3 & 1 & 1 & 0 & 1 \\ 2 & 2 & 1 & 1 & 5 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}$$

d)

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

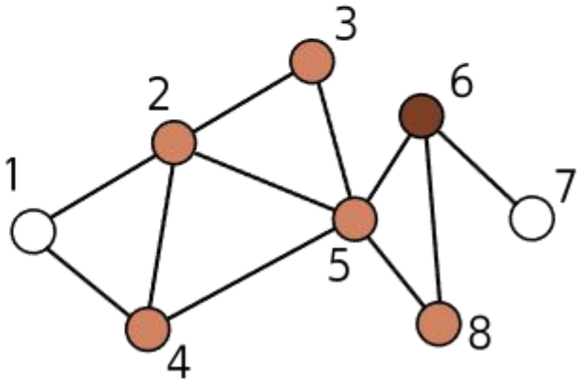
e)

$$\mathbf{Ax} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$



f)

$$\mathbf{A}^2\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$

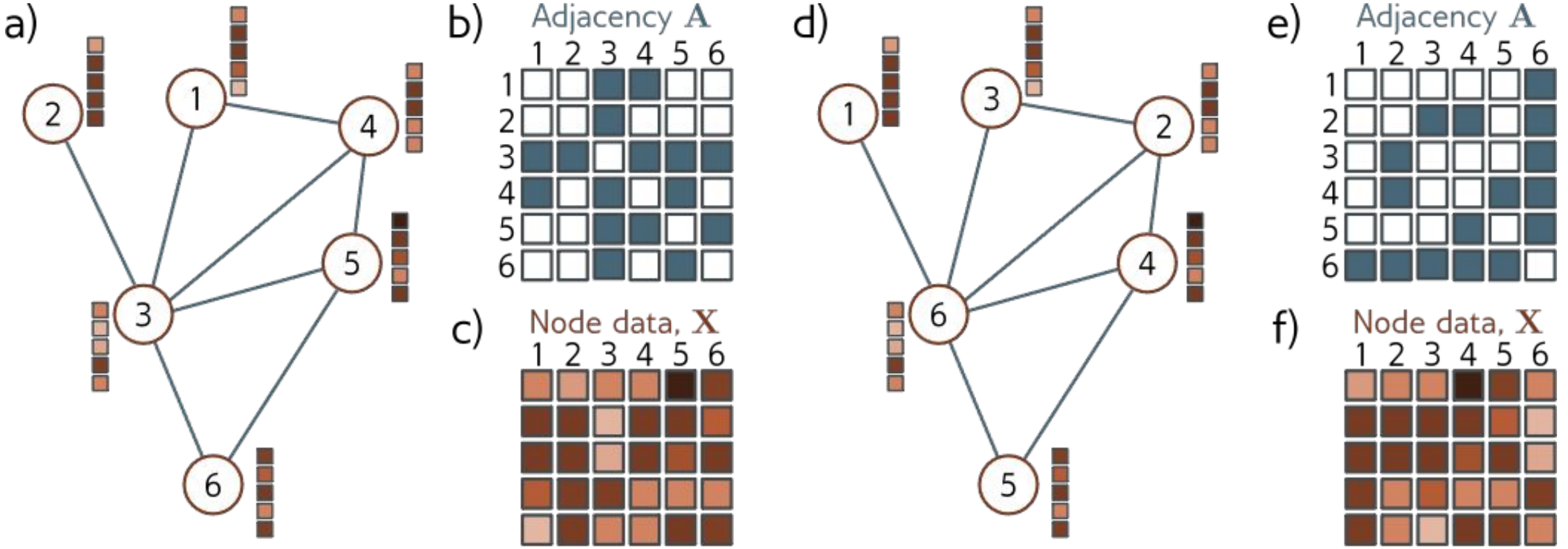


A^l contains the number of unique walks of length l from node m to node n

Nodes Permutation

Node indexing in graphs is arbitrary

Permuting the node **indices** results in a **permutation** of the **columns** of the node data matrix $\boldsymbol{X}$ and a permutation of both the **rows** and **columns** of the adjacency matrix $\boldsymbol{A}$.



$$\mathbf{X'=XP}, \quad \mathbf{A'=P^T AP}$$

Graph Learning

We want to learn a (dense) representation ***H*** of the graph usable for different downstream tasks

A **graph neural network** is a model that takes:

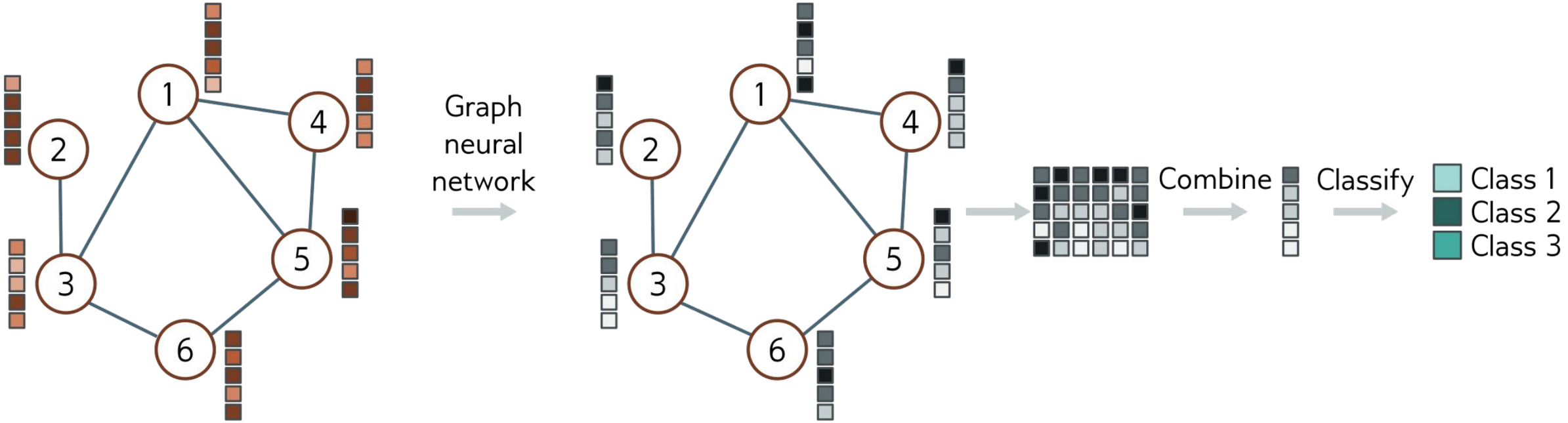
- the node embeddings ***X*** and the adjacency matrix ***A*** as inputs and passes them through a series of *k* layers.
- the node embeddings are updated in each layer to create intermediate "hidden" representations ***h*** before finally computing output embeddings h_K.

Graph Classification Tasks

For example, if we want to **predict:**

- the **temperature** at which a **molecule** becomes **liquid** (a **regression task)**;
- whether a **molecule** is **poisonous** to human beings or not (a **classification task**).

For graph-level tasks, *the output* ***node embeddings are combined*** *(e.g., by averaging), and the resulting vector is* ***mapped*** *via a linear transformation or neural network to a* ***fixed-size vector****.*

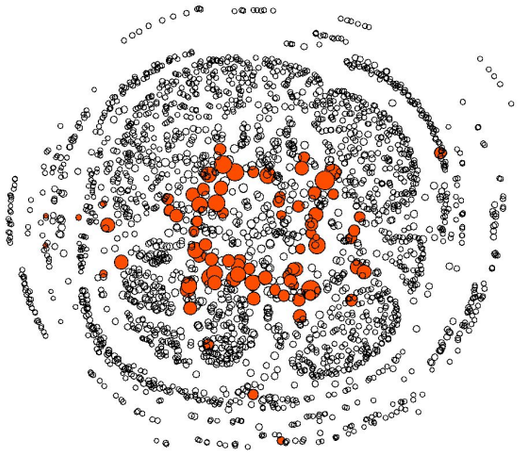


$$Pr(y = 1 \mid \boldsymbol{X}, \boldsymbol{A}) = sig\left(\beta_k + \boldsymbol{\omega}_k \boldsymbol{H}_k \frac{\mathbf{1}}{N}\right)$$

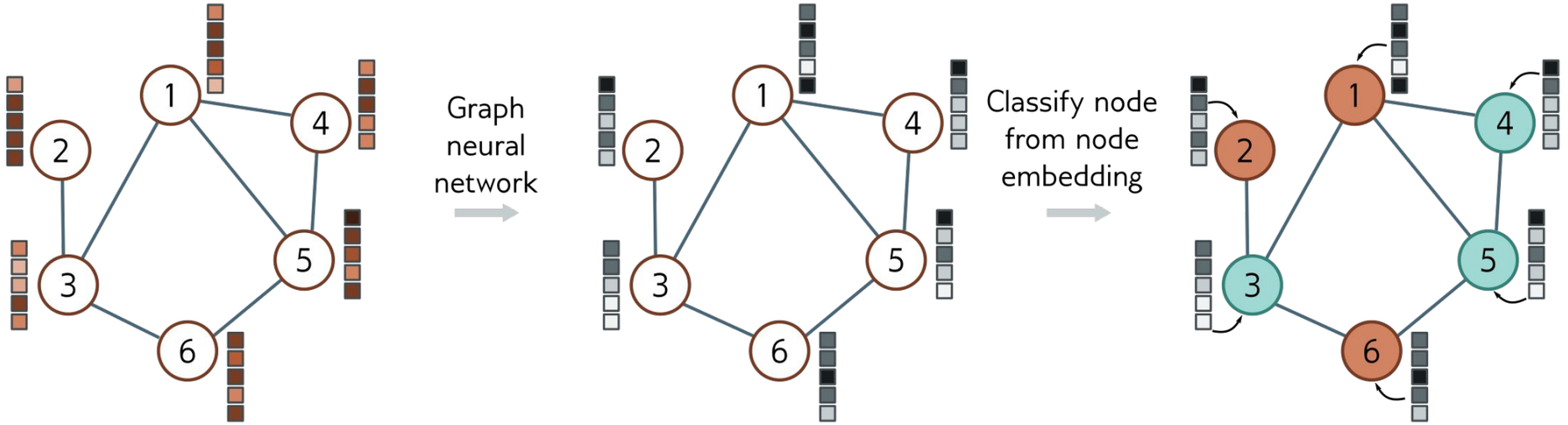
Node Classification Tasks

For example, in an *PPI* network we might want to **predict:**

- the probability that a **given node** might be **attacked/being part** of a certain **disease** (classification) as it is shown for COVID19 (**red**) - PPI (on the right).



*The network assigns **one or more label** (classification) or values (regression) to **each node** of the graph, **using** both the **graph structure** and learned **node embeddings**.*

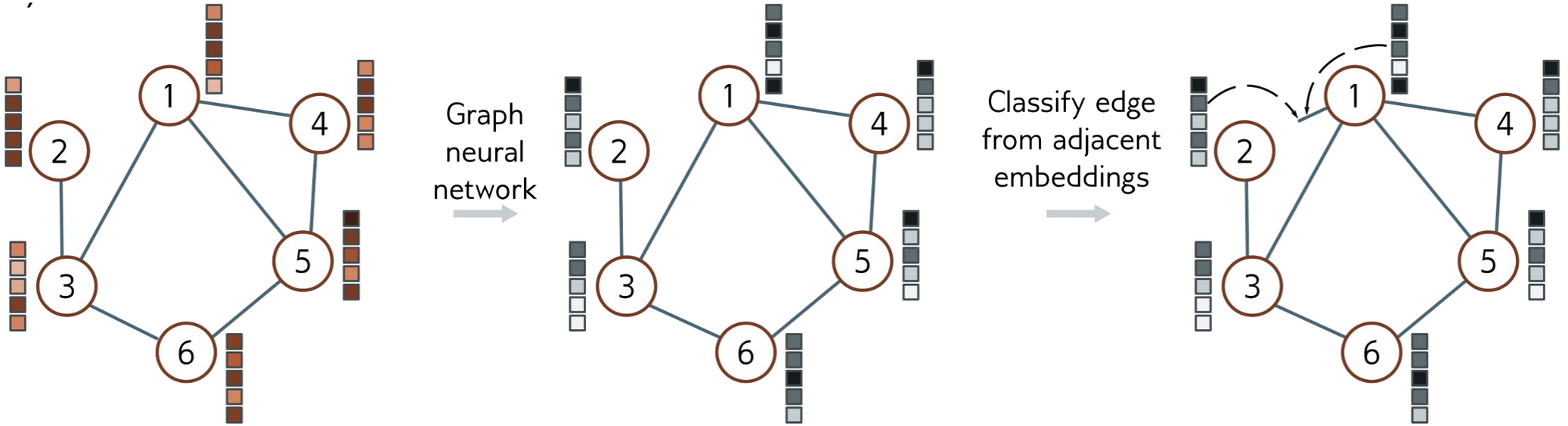


$$Pr\left(y^{(n)} = 1 \mid \boldsymbol{X}, \boldsymbol{A}\right) = sig\left(\beta_k^{(n)} + \boldsymbol{\omega}_k^{(n)} \boldsymbol{H}_k^{(n)}\right)$$

Edge Classification Tasks

For example, in the **social network** setting, we want to **predict** whether two people know each other and suggest that they connect if that is the case (recommendation).

*The network assigns **one or more label** (classification) or values (regression) to **each edges** of the graph, using both the graph **structure** and learned **node embeddings**.*



$$Pr\left(\boldsymbol{y}^{(mn)} = 1 \mid \boldsymbol{X}, \boldsymbol{A}\right) = sig\left(\boldsymbol{H}_k^{(m)T} \cdot \boldsymbol{H}_k^{(n)}\right)$$

GNNs

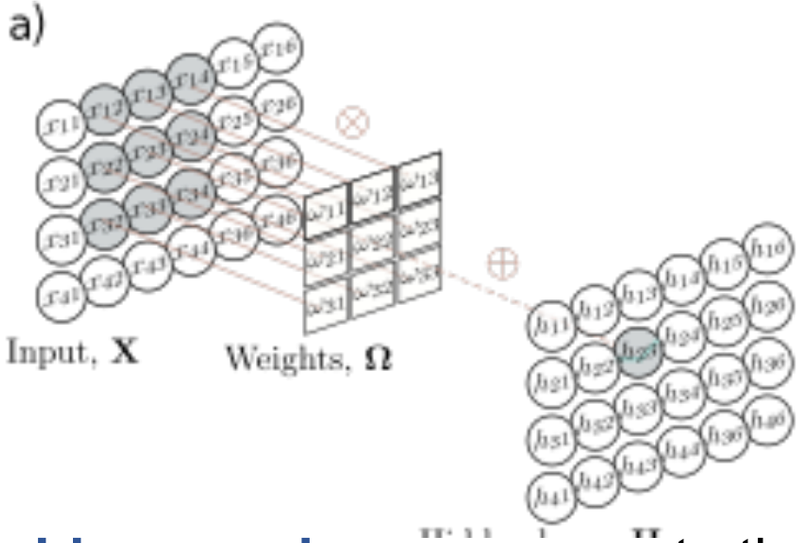
Spatial-based convolutional graph neural networks **(GCN)** are convolutional network that update each node embeddings by **aggregating information from nearby nodes** using the original **graph structure**.

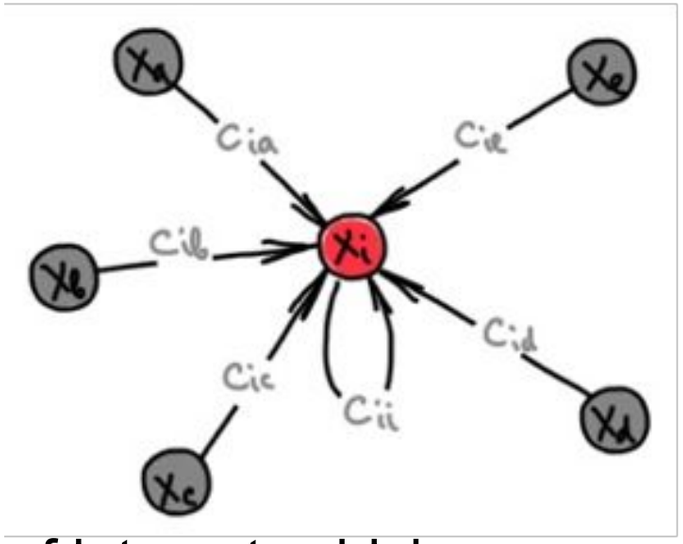
Each **layer** of the GCN is a function *F[·]* with parameters **Φ** that **takes** the node **embeddings** and **adjacency matrix** and outputs new node embeddings:

$$
\begin{aligned}
\boldsymbol{H}_1 &= \mathrm{F}[\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{\phi}_0] \\
\boldsymbol{H}_2 &= \mathrm{F}[\boldsymbol{H}_1, \boldsymbol{A}, \boldsymbol{\phi}_1] \\
\boldsymbol{H}_3 &= \mathrm{F}[\boldsymbol{H}_2, \boldsymbol{A}, \boldsymbol{\phi}_2] \\
\vdots &= \quad \vdots \\
\boldsymbol{H}_{\mathrm{K}} &= \mathrm{F}[\boldsymbol{H}_{\mathrm{K}-1}, \boldsymbol{A}, \boldsymbol{\phi}_{K-1}]
\end{aligned}
$$

Parameter Sharing

- Likewise CNN, we want the **same parameters** at every **node**: **reducing** the number of **parameters** and **sharing** what the network **learns** at each node **across** the **entire** graph.





- **Each neighbor sends a message** to the variable of interest, which **aggregates** these messages to **form** the **update**.

In **images**, the neighbors were **pixels** from a **fixed-size square region** around the current position, so the **spatial relationships** at each position **are the same**.

- In a **graph**, each node may have a **different number of neighbors**, and there are **no consistent relationships**.

GCN (1)

each node at layer *k*, we aggregate information from neighboring nodes by e.g. summing their node embeddings:

$$\boldsymbol{agg}[n,k] = \sum_{m \in \mathbf{ne}[\mathbf{n}]} \boldsymbol{H}_k^{(m)}$$

linear transformation **Ω** to the embedding H of the current node and to his aggregated value, we add a bias term *β*, and pass the result through a nonlinear activation function *a[·]*, which is applied independently to every member of its vector argument:

$$\boldsymbol{H}_{k+1}^{(n)} = a\left[\boldsymbol{\beta}_k + \boldsymbol{\Omega}_k \cdot \boldsymbol{H}_k^{(n)} + \boldsymbol{\Omega}_k \cdot \boldsymbol{agg}[n,k]\right]$$

the n^{th} column of ***A*** contains ones at the positions of neighbors. If **post-multiply** the **embeddings** by **A** the n^{th} column is **agg**[n,k]:

$$\boldsymbol{H}_{k+1} = a[\boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \boldsymbol{H}_k + \boldsymbol{\Omega}_k \boldsymbol{H}_k \boldsymbol{A}] = a[\boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \boldsymbol{H}_k (\boldsymbol{A} + \boldsymbol{I})]$$

GCN (2)

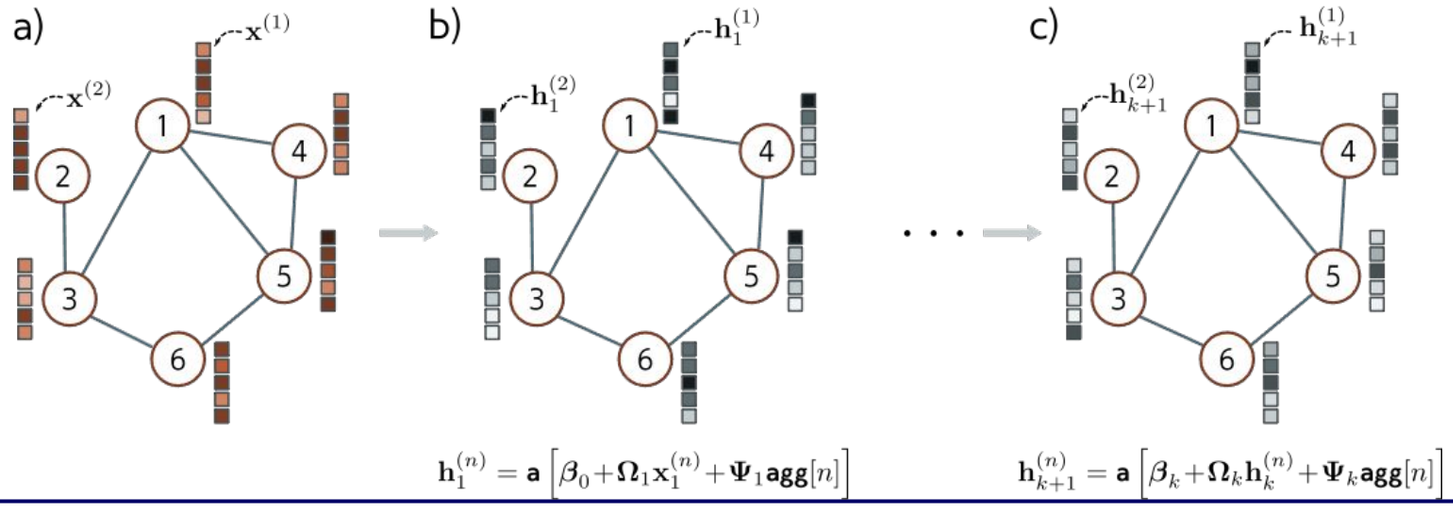
$$\boldsymbol{H}_{k+1} = a[\boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \boldsymbol{H}_k (\boldsymbol{A} + \boldsymbol{I})]$$

This layer **satisfies** the **design** considerations:

- it **is equivariant** to **permutations** of the node **indices**

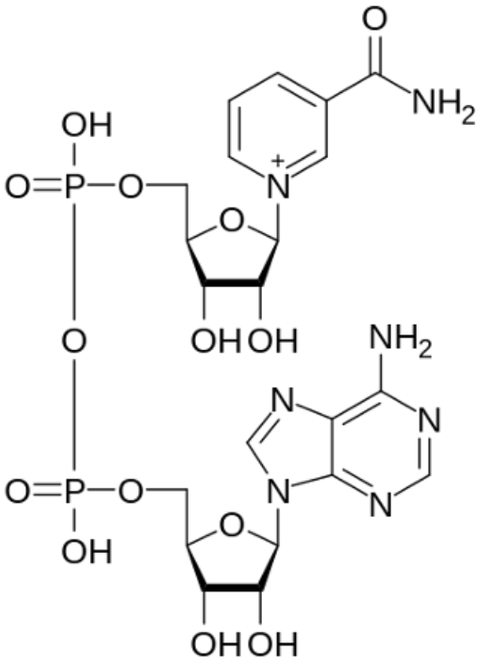
$$\boldsymbol{H}_{k+1}\boldsymbol{P} = \mathrm{F}[\boldsymbol{H}_k\boldsymbol{P}, \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}, \boldsymbol{\phi}_k] = a[\boldsymbol{\beta}_k \mathbf{1}^T\boldsymbol{P} + \boldsymbol{\Omega}_k \boldsymbol{H}_k\boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P} + \boldsymbol{I})]$$

- can **cope** with **any** number of **neighbors** due to the **agg**[n,k] function;
- **exploits** the **graph structure** to provide a relational inductive bias,
- and **shares parameters** throughout the graph i.e. **Ω**.



Graph Classification (Rev.)

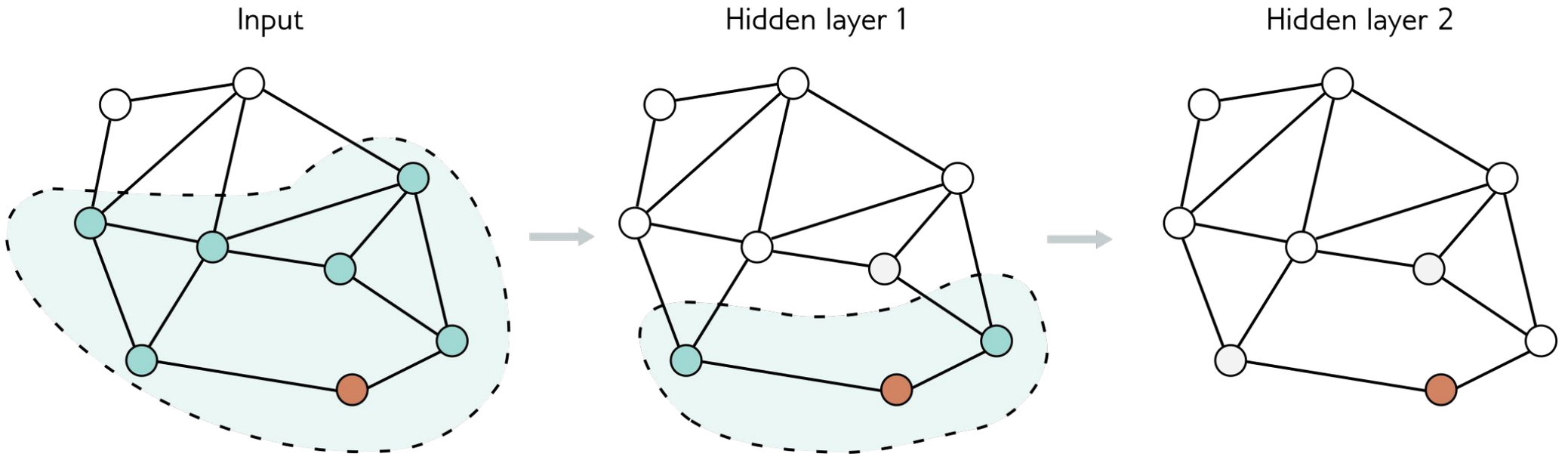
- We want a **neural network** f[**X**,**A**,Φ] that **classifies** (predicts) **molecules** as toxic or harmless.
- The **adjacency matrix** A ∈ R^(N×N) derives from the molecular structure.
- The columns of the **node embedding** matrix are **one-hot vectors** indicating which of the 118 **elements of the periodic** table are present.



$$\boldsymbol{H}_1 = a[\boldsymbol{\beta}_0 \mathbf{1}^T + \boldsymbol{\Omega}_0 \boldsymbol{X}\ (\boldsymbol{A}+\boldsymbol{I})]$$
$$\boldsymbol{H}_2 = a[\boldsymbol{\beta}_1 \mathbf{1}^T + \boldsymbol{\Omega}_1 \boldsymbol{H}_1 (\boldsymbol{A}+\boldsymbol{I})]$$
$$\vdots = \quad \vdots$$
$$\boldsymbol{H}_k = a[\boldsymbol{\beta}_{k-1} \mathbf{1}^T + \boldsymbol{\Omega}_{k-1} \boldsymbol{H}_{k-1} (\boldsymbol{A}+\boldsymbol{I})]$$
$$\mathrm{f}[\boldsymbol{X}, \boldsymbol{A}, \Phi] = sig\left[\beta_k + \boldsymbol{\omega}_k \boldsymbol{H}_k \frac{\mathbf{1}}{N}\right]$$

$$\boldsymbol{X}(\boldsymbol{A}+\boldsymbol{I})^k, \text{e.g } \boldsymbol{H}_3\text{: } \boldsymbol{X}(\boldsymbol{A}+\boldsymbol{I})^3 = \boldsymbol{X}(\boldsymbol{A}^3 + 3\boldsymbol{A}^2 + 3\boldsymbol{A} + \boldsymbol{I}^3)$$

GNNs (briefly)



$$H^{\ell} \text{ depends on } X(A+I)^{\ell}$$

graph expansion problem

If there are **many layers** and the graph is **densely connected:**

every input node may be in the receptive field of every output.

In general we want that k << diam(G)

Part II

eXplainable Artificial Intelligence

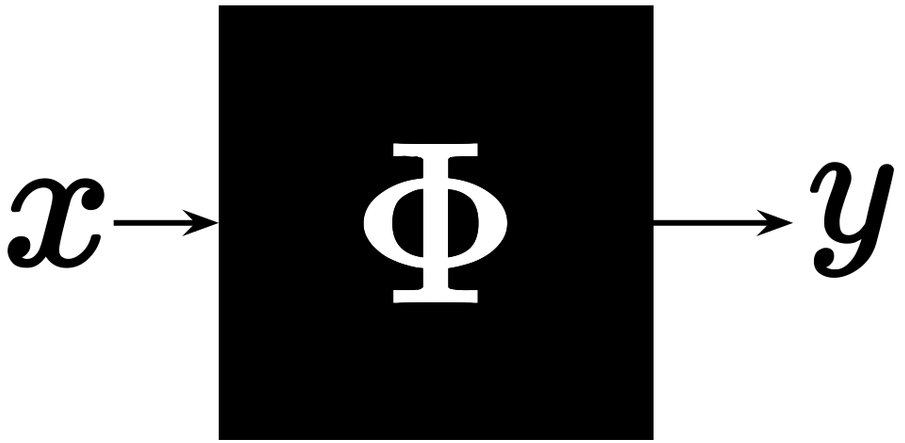
by Bardh Prenkaj

first part of the slides based on: CSEP 590B: Explainable AI from University of Washington

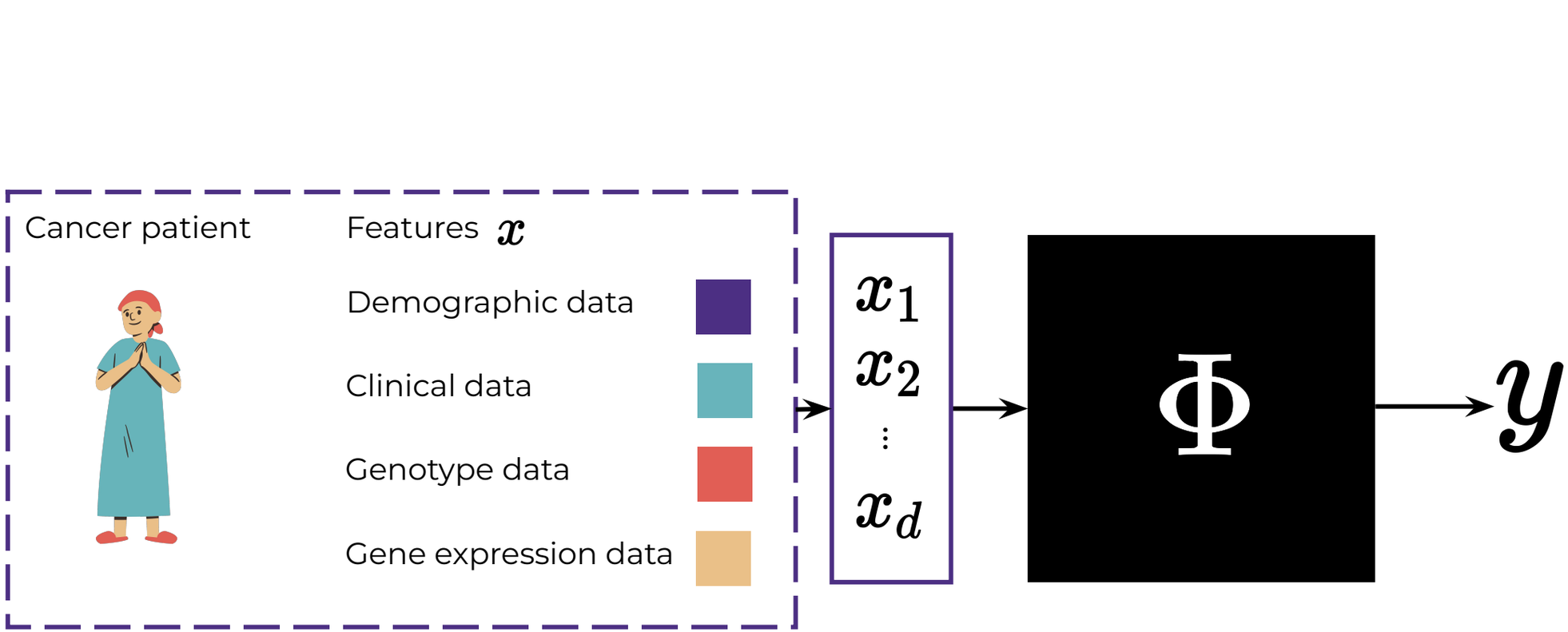
What's going on today in ML?

Lack of transparency

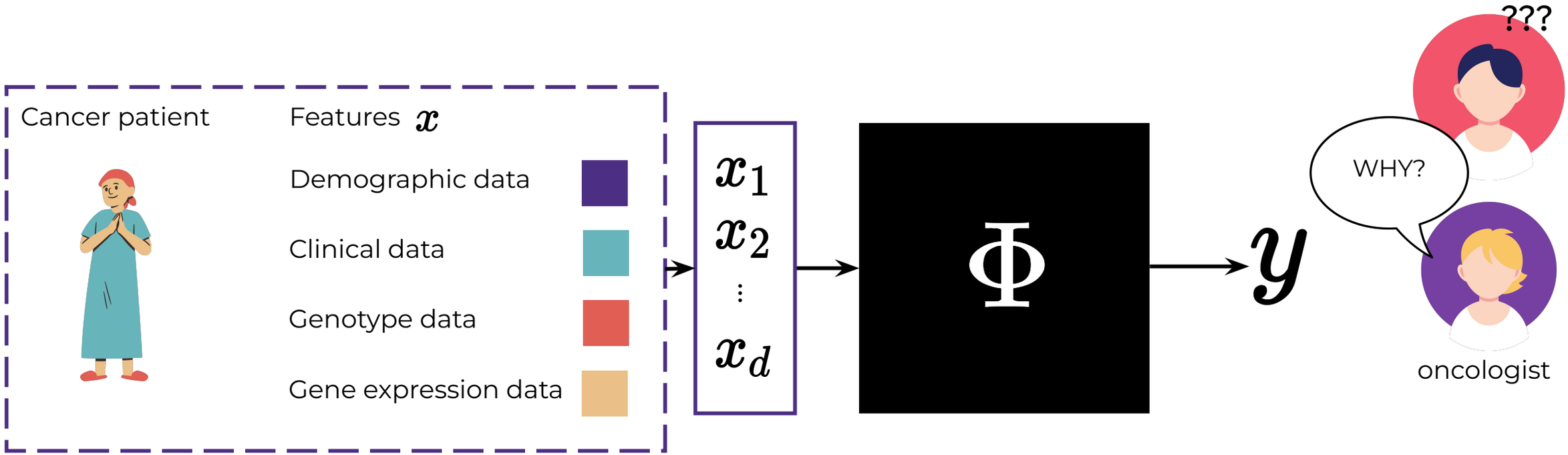
- Identify key factors in underlying processes
- Generate scientific hypotheses



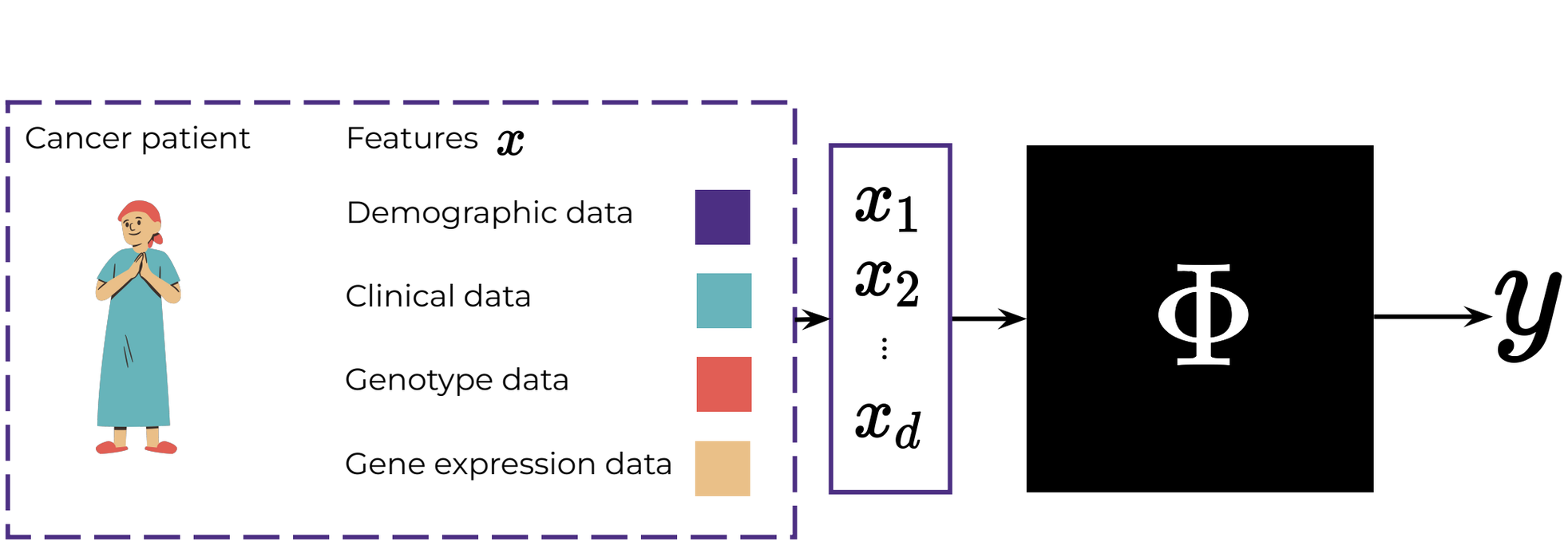
WHY ACCURATE PREDICTIONS ARE IMPORTANT?



WHY ACCURATE PREDICTIONS ARE IMPORTANT?



WHY ACCURATE PREDICTIONS ARE IMPORTANT?



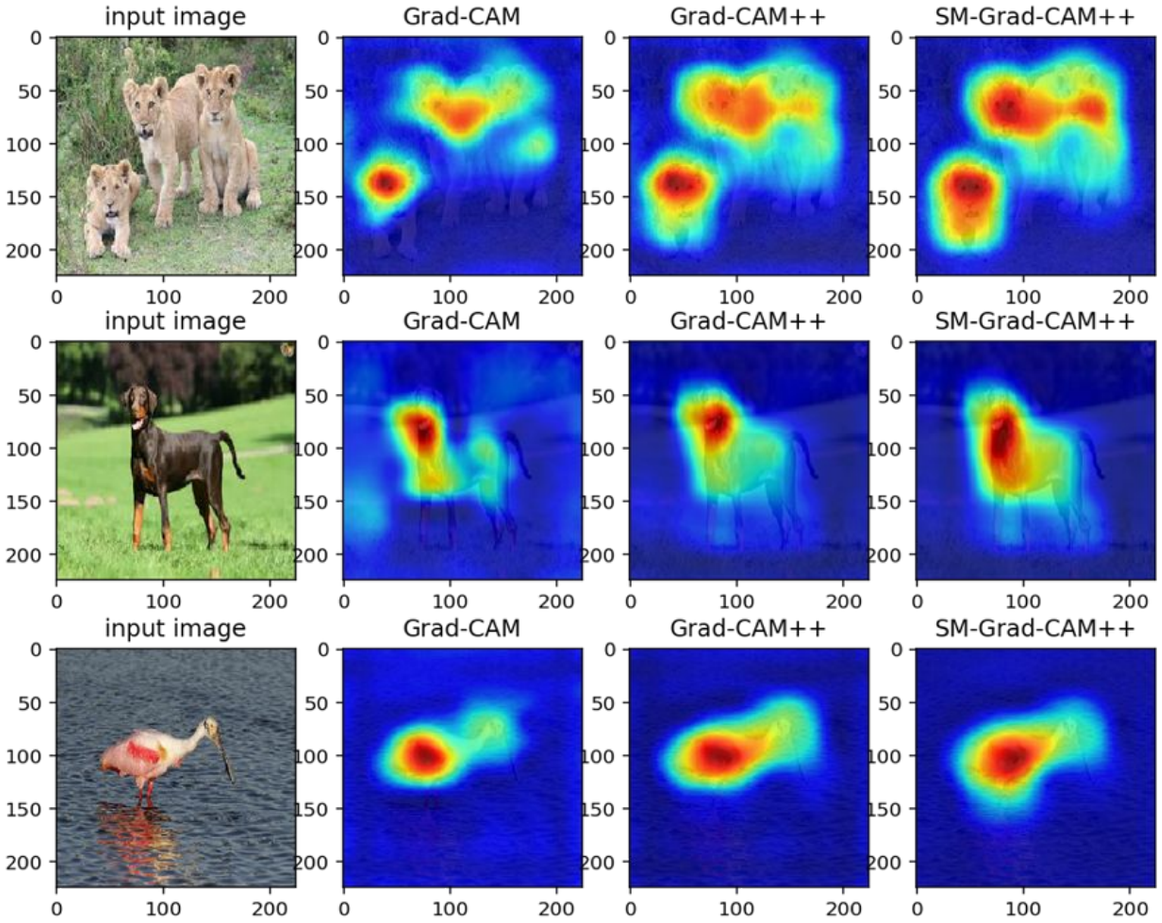
Explainability

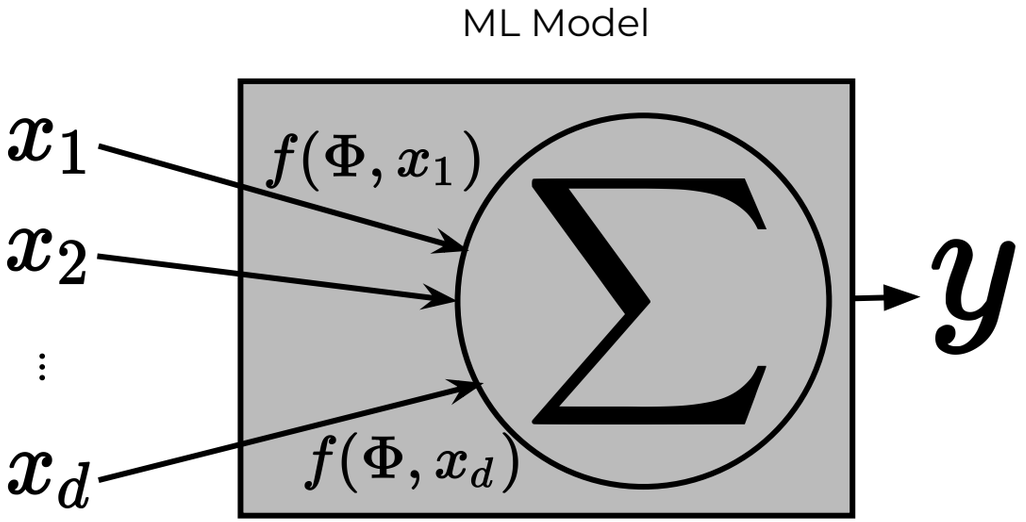
- Which features contributed to a certain prediction and how?
- How to learn or select features that are most interpretable or informative?
- How to make biological or clinical sense of a black-box model?

Importance

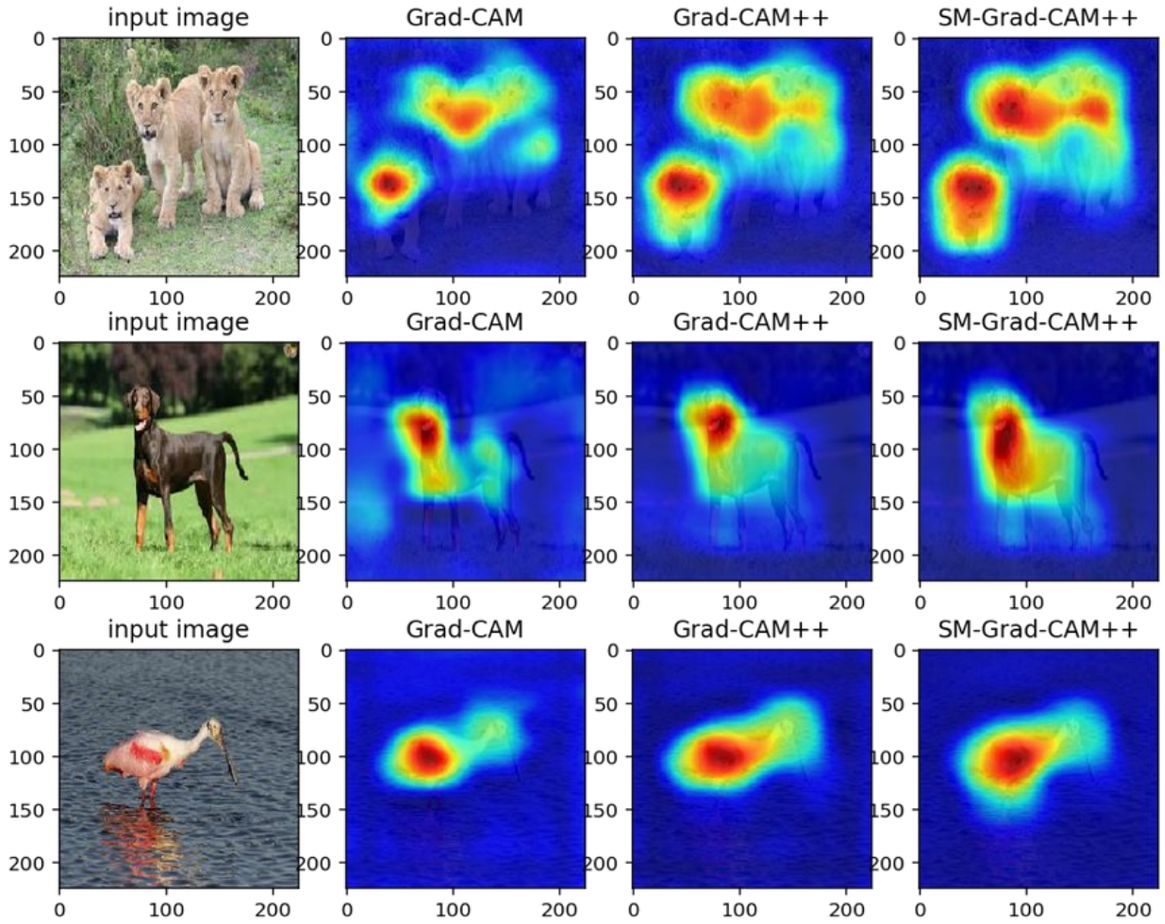
- Identify key factors in underlying processes
- Generate scientific hypotheses
- Diagnose model failures
- Audit for unwanted dependencies
- Enable regulation
- Improve datasets
- Build user/organizational trust
- **Inform users of recourse options (today's tutorial)**

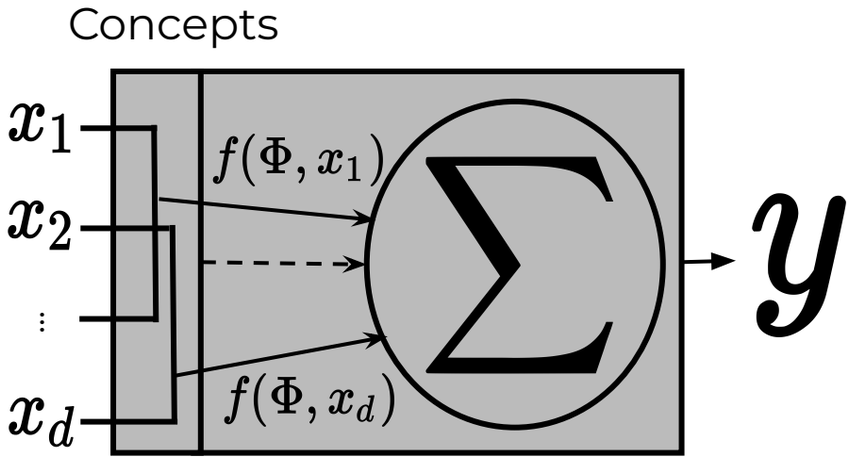
Types of XAI questions we can ask





Types of XAI questions we can ask





Example

- We work at a bank, and our boss is asking us to automate **credit risk evaluation**
- **Step 1**: we get some historical data

$$x = [\mathbf{age}, \mathbf{job}, \mathbf{salary}, \mathbf{sex}, \ldots]$$

$$y \in \{\mathbf{approved}, \mathbf{declined}\}$$

Example

- **Step 2**: train a model Φ, complex one (e.g., GBM) because it gets the best performances out of 100 different models we trained and evaluated

Example

- **Our boss**: “Nice job, can I ask some questions about how the model works?”
- **We**: “Sure, boss!”

Question #1

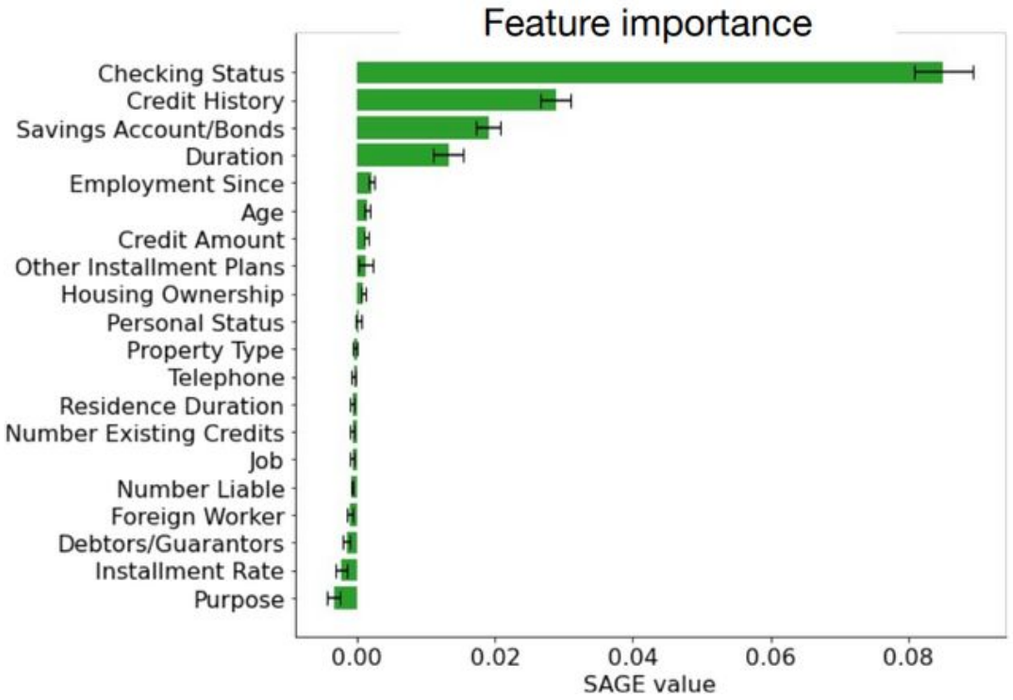
What did the model learn, and how does it make decisions?

- We can’t easily summarize the patterns, rules, concepts learned by a complex model
- Gradient boosted trees have too many parameters to examine
- Coarse summary: count number of splits on each feature

Question #2

Which features are most important overall?

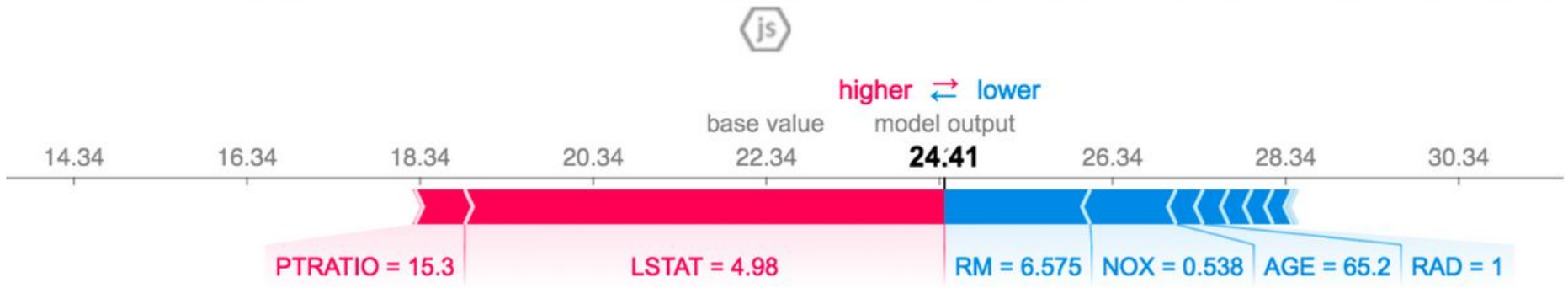
- Can be answered by permutation tests (Breiman, 2001)
- There are in fact many methods for analyzing **global feature importance**



Question #3

For the customers whose loans are denied, can we tell which features led to the decision?

- In this case, must analyze individual predictions (not overall behavior)
- Many methods designed to assess **local feature importance**



Types of Explainability

Feature importance explanations

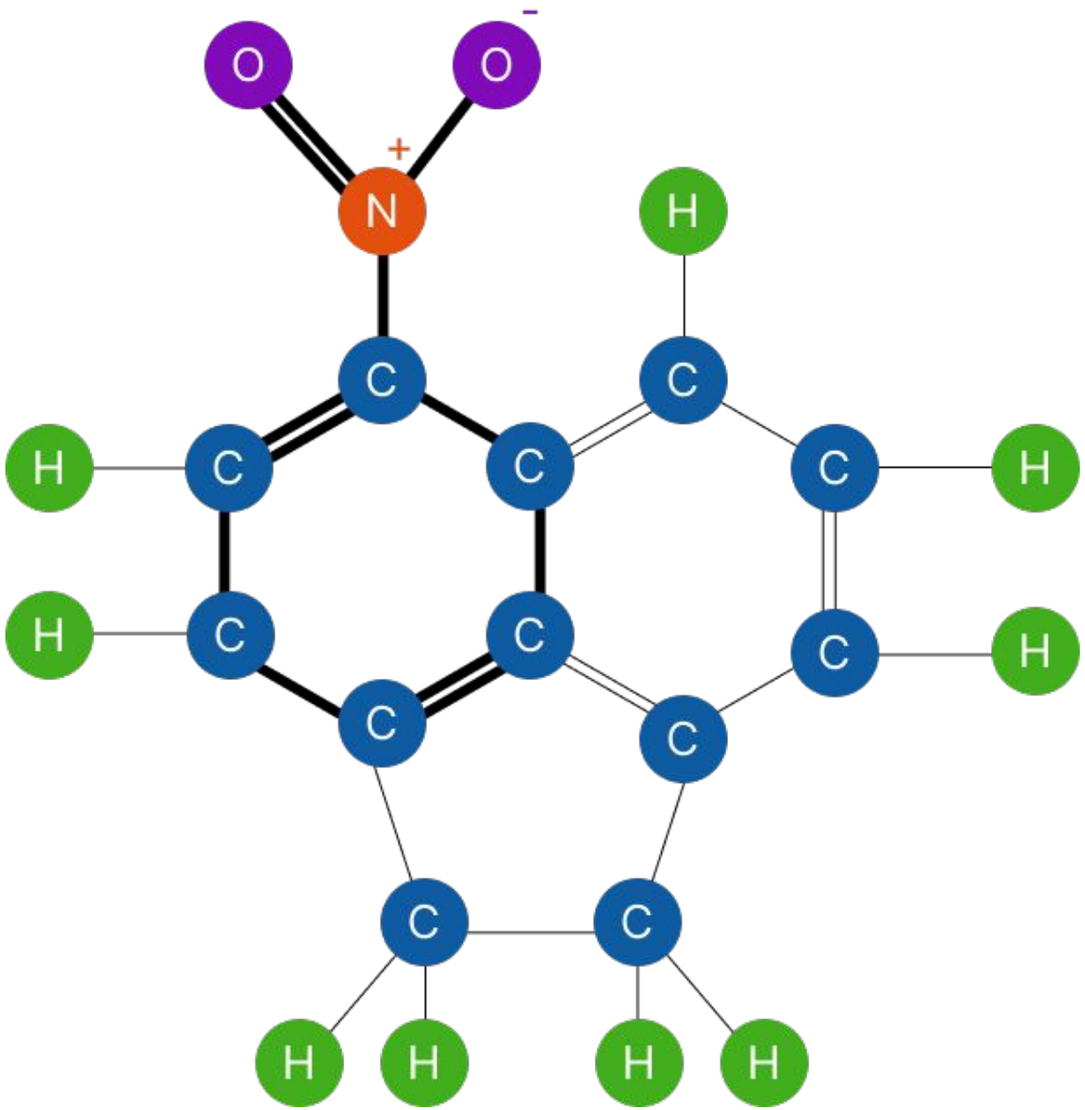
- Removal-based explanations
- Shapley values
- Propagation-based explanations

Some sort of factual explanations

Inherently interpretable models

Counterfactual explanations

Factual explanations on Graphs



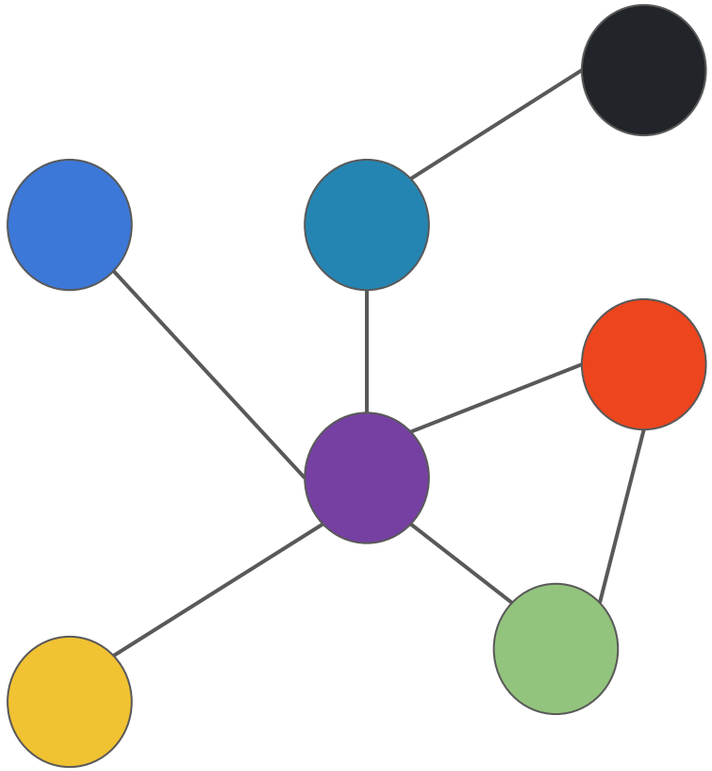
Factual explanations on Graphs

- Find those edges whose subgraph induced on them has the same label as the whole graph (**desideratum #1**)
- This subgraph should be minimal (**desideratum #2**)
- When you remove this subgraph, the remainder should have the opposite class (**desideratum #3**) – *this gives sprout to factual-based counterfactual explainers*

Revisiting GNNExplainer[1] and GraphLIME[2]

based on (1) Ying et al. “GNNExplainer: Generating Explanations for Graph Neural Networks”, NeurIPS 2019
& (2) Huang et al. “GraphLIME:Local Interpretable Model Explanations for Graph Neural Networks”, TKDE 2023

GNNEXPLAINER (INTRO)



GNNExplainer: Generating Explanations for Graph Neural Networks

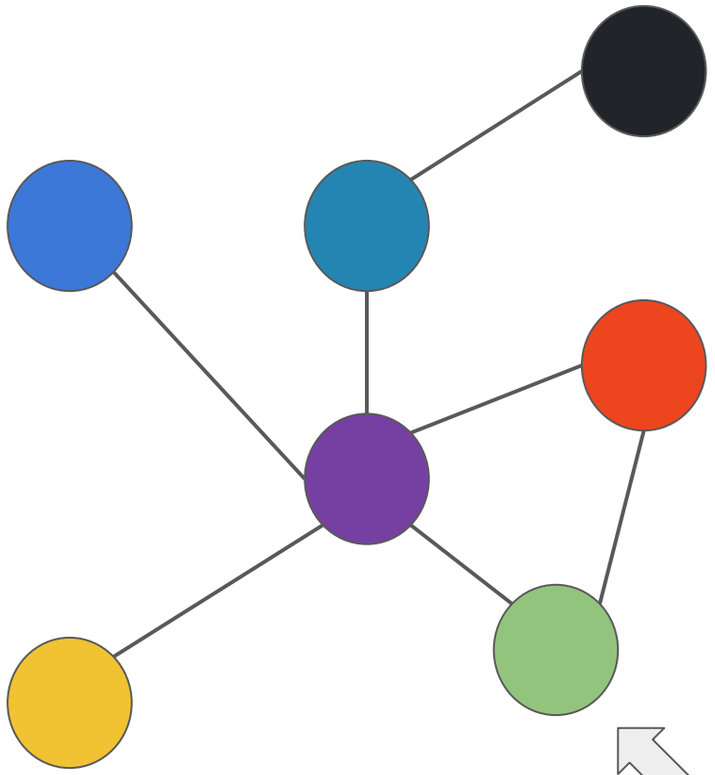
Rex Ying[†] Dylan Bourgeois[†,‡] Jiaxuan You[†] Marinka Zitnik[†] Jure Leskovec[†]

[†]Department of Computer Science, Stanford University
[‡]Robust.AI
{rexying, dtsbourg, jiaxuan, marinka, jure}@cs.stanford.edu

Abstract

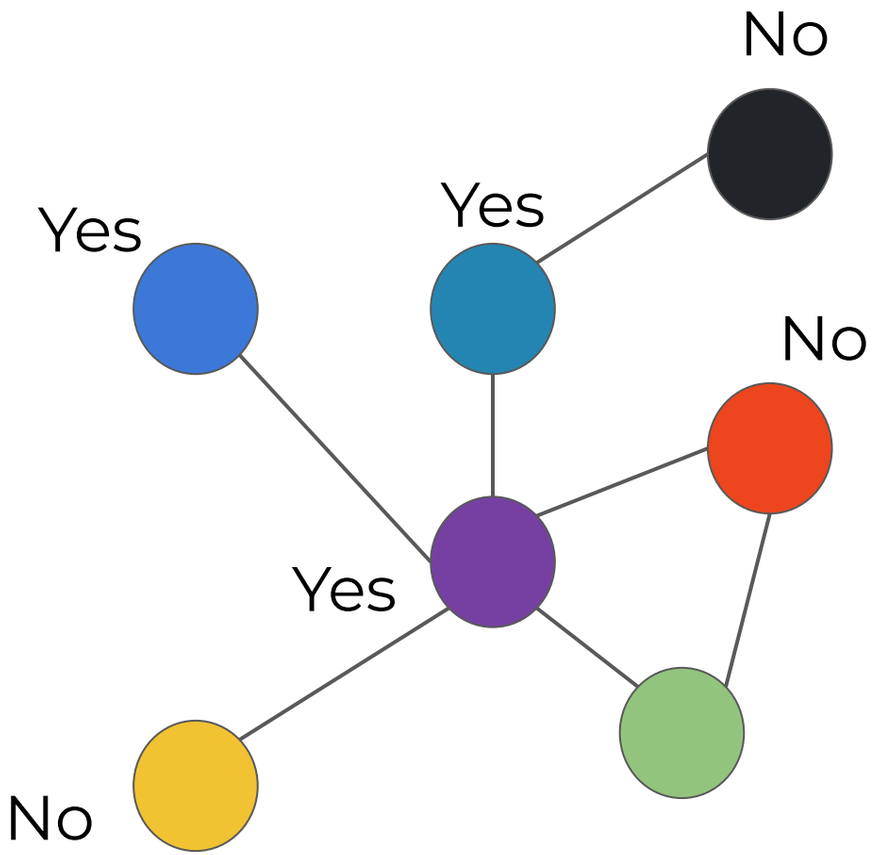
Graph Neural Networks (GNNs) are a powerful tool for machine learning on graphs. GNNs combine node feature information with the graph structure by recursively passing neural messages along edges of the input graph. However, incorporating both graph structure and feature information leads to complex models and explaining predictions made by GNNs remains unsolved. Here we propose GNNEXPLAINER, the first general, model-agnostic approach for providing interpretable explanations for predictions of any GNN-based model on any graph-based machine learning task. Given an instance, GNNEXPLAINER identifies a compact subgraph structure and a small subset of node features that have a crucial role in GNN's prediction. Further, GNNEXPLAINER can generate consistent and concise explanations for an entire class of instances. We formulate GNNEXPLAINER as an optimization task that maximizes the mutual information between a GNN's prediction and distribution of possible subgraph structures. Experiments on synthetic and real-world graphs show that our approach can identify important graph structures as well as node features, and outperforms alternative baseline approaches by up to 43.0% in explanation accuracy. GNNEXPLAINER provides a variety of benefits, from the ability to visualize semantically relevant structures to interpretability, to giving insights into errors of faulty GNNs.

GNNExplainer (Intro)

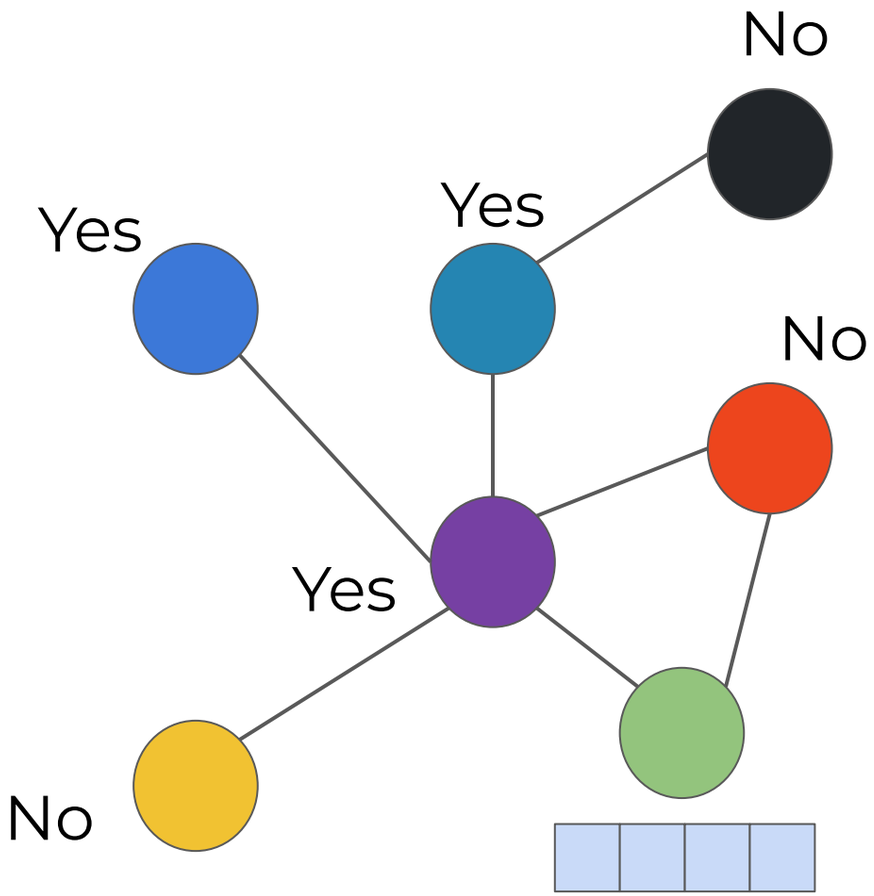


node classification

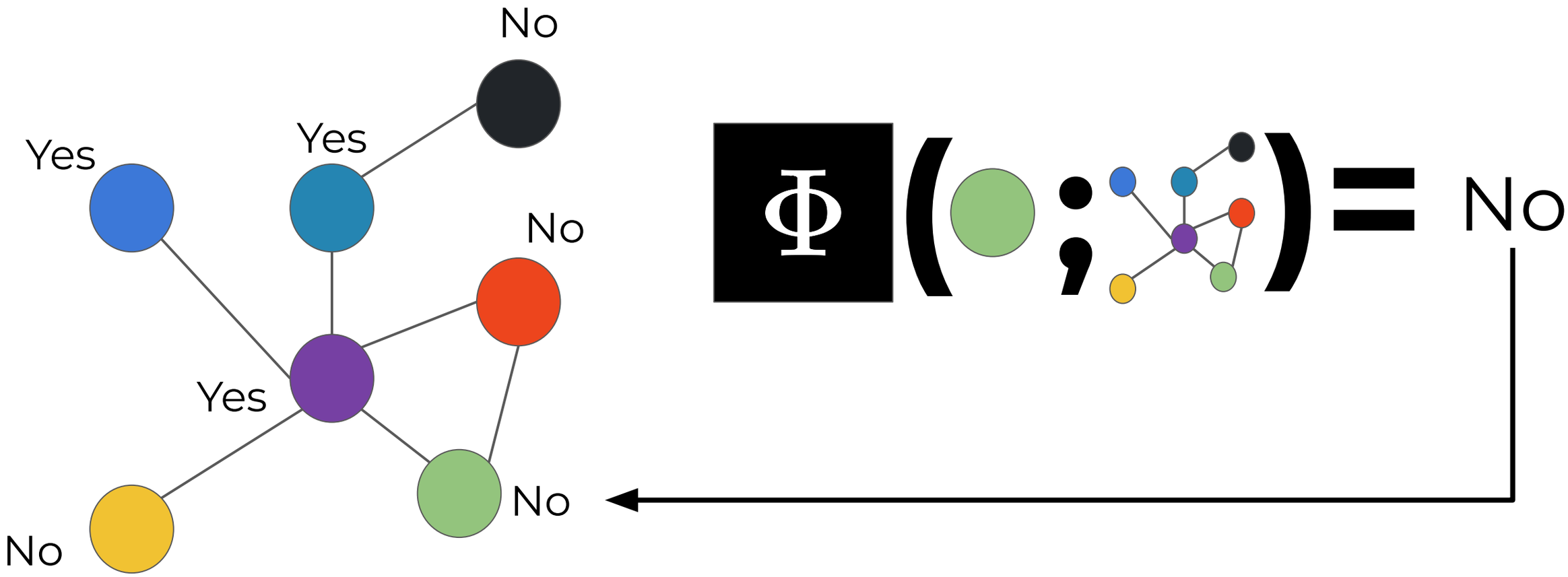
GNNExplainer (Intro)



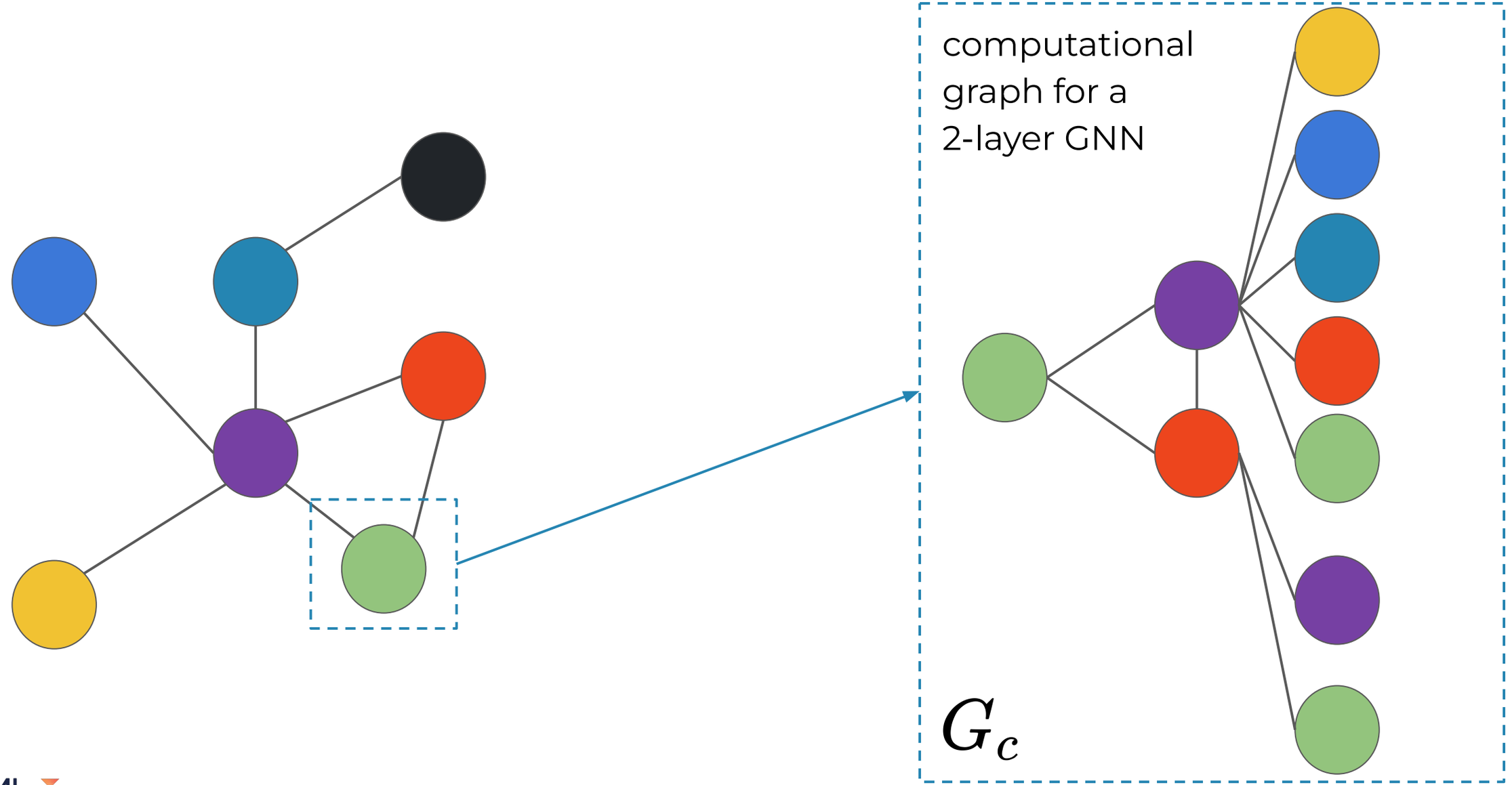
GNNExplainer (Intro)



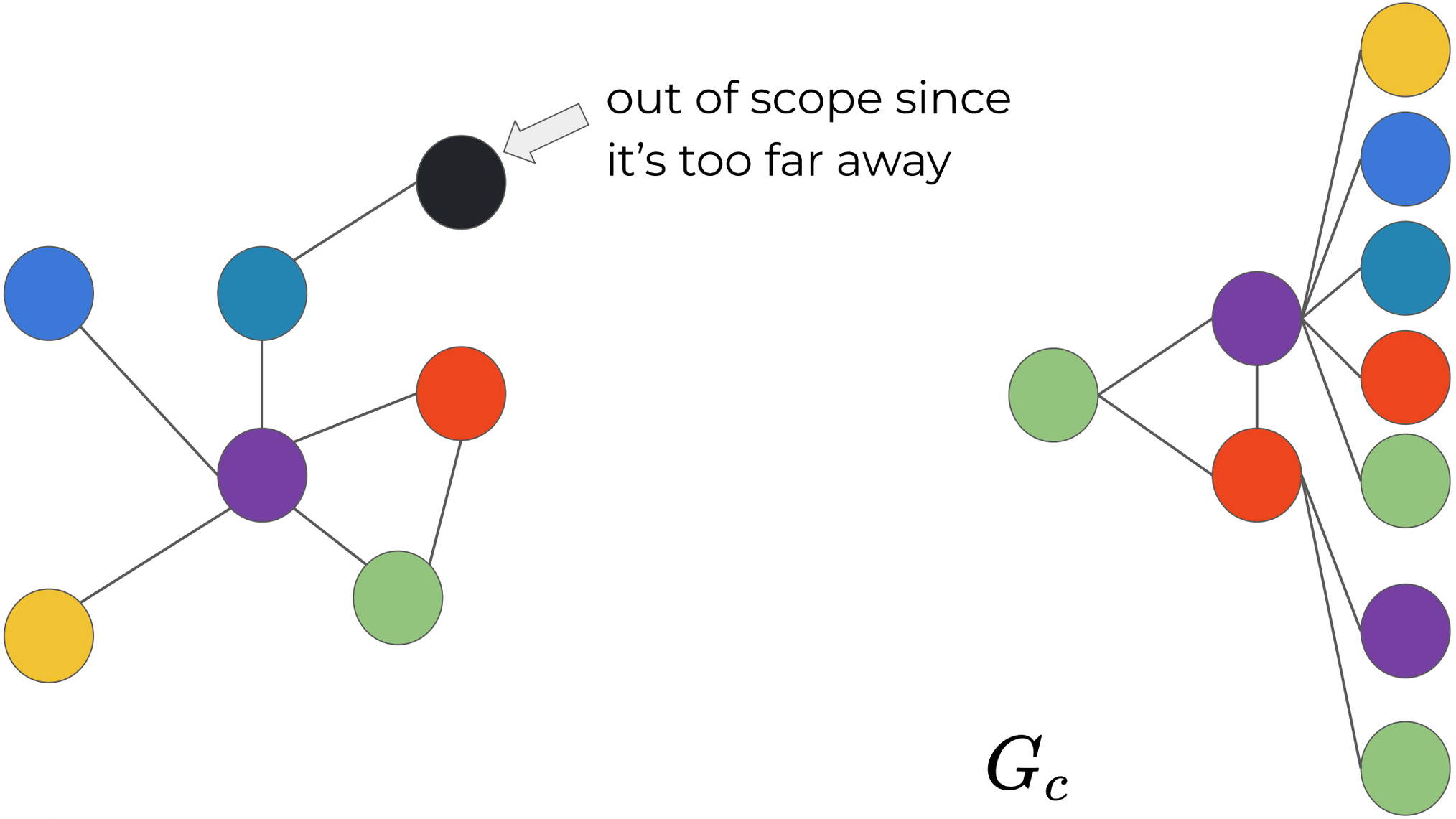
GNNExplainer (Intro)



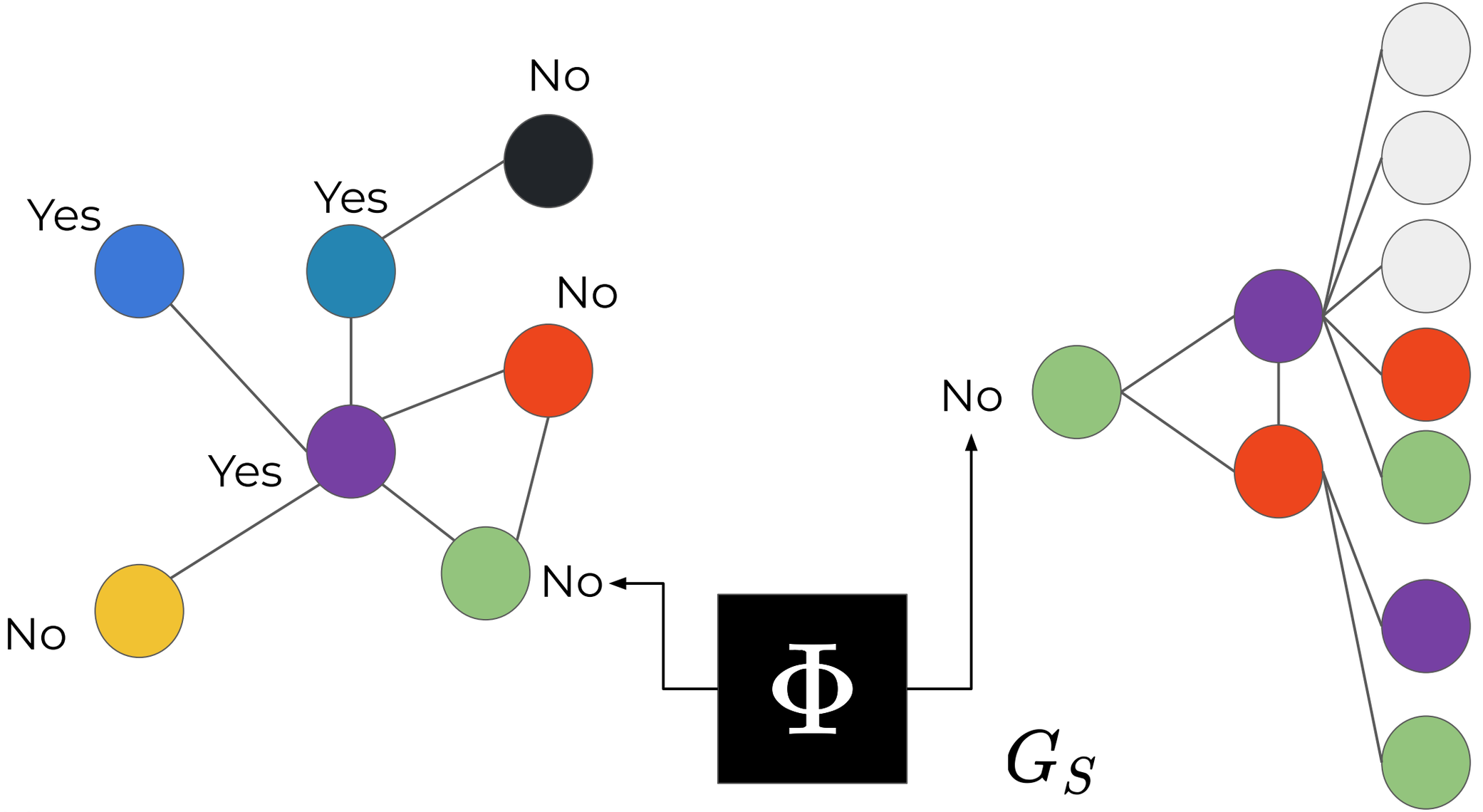
GNNExplainer (Comp. Graph)



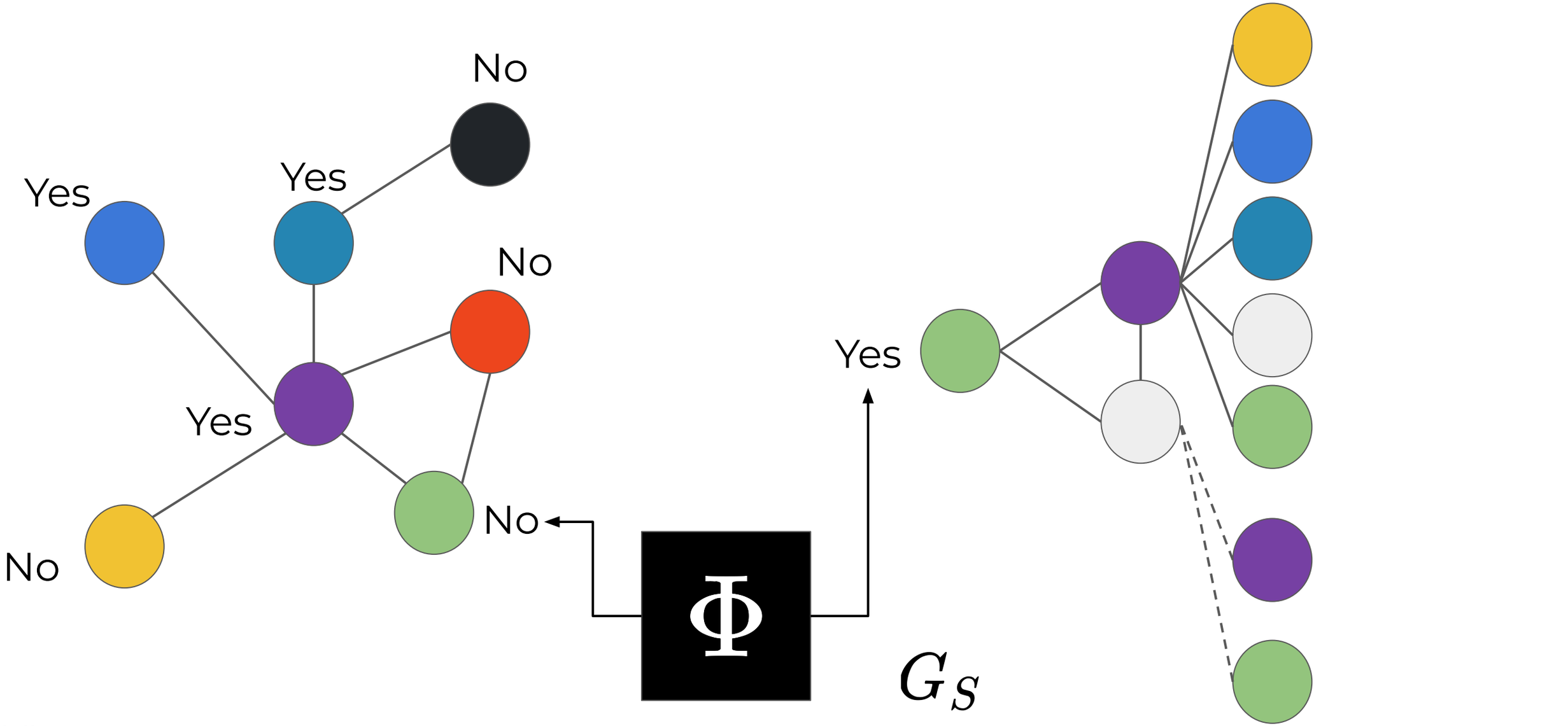
GNNExplainer (Comp. Graph)



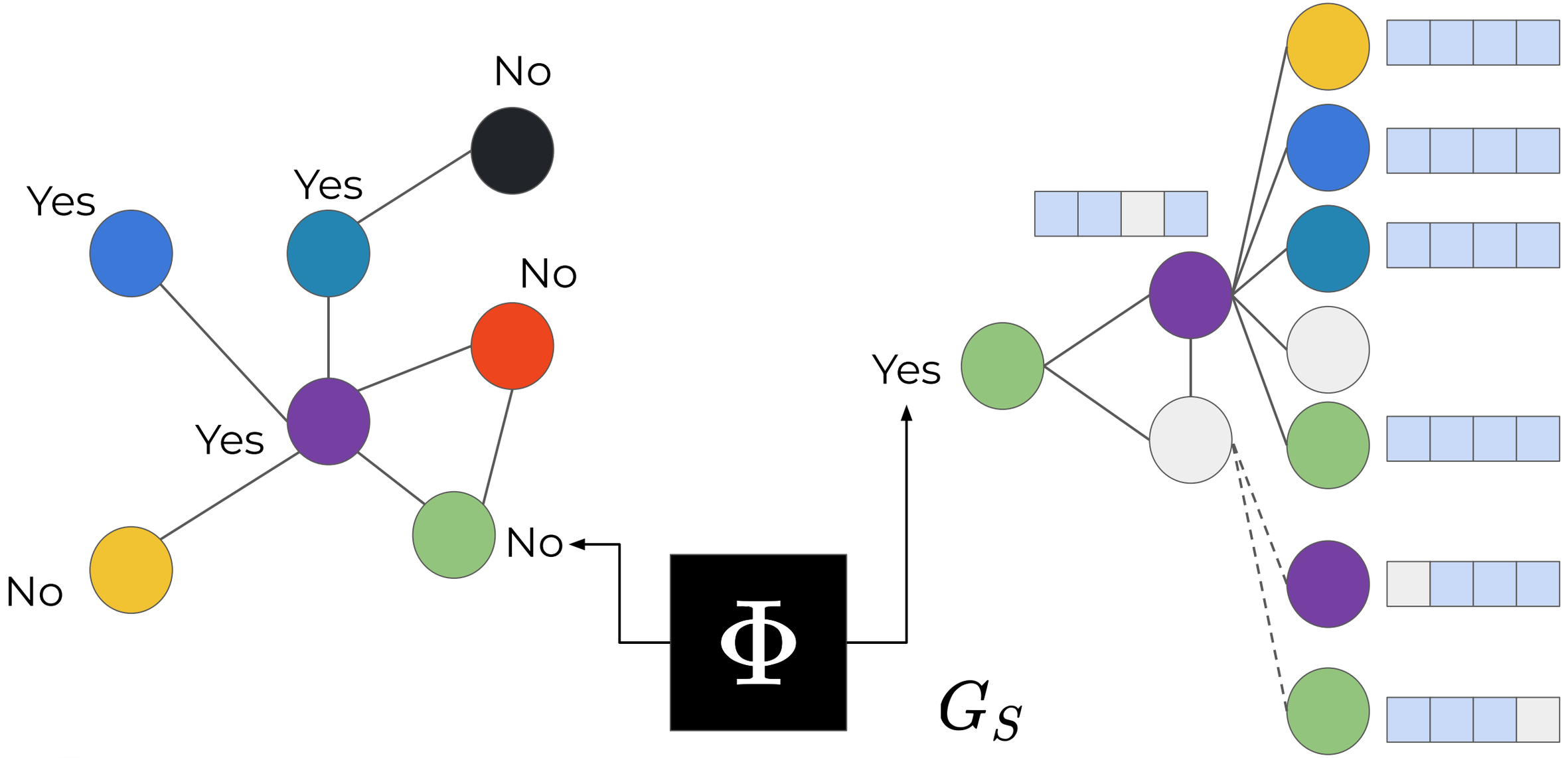
GNNExplainer (Remove nodes)



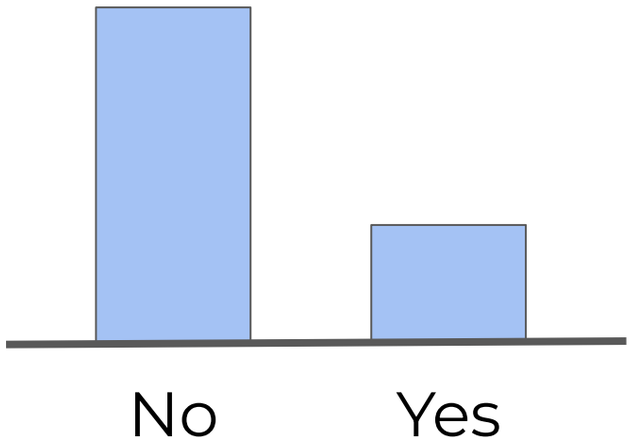
GNNExplainer (Remove nodes)

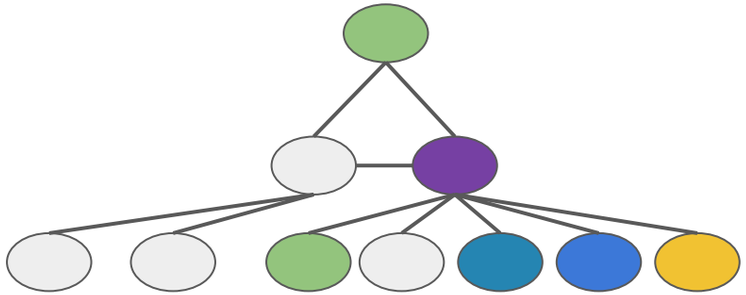


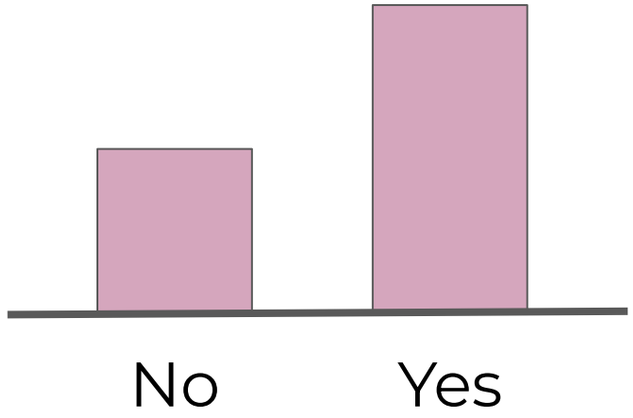
GNNExplainer (Remove nodes)

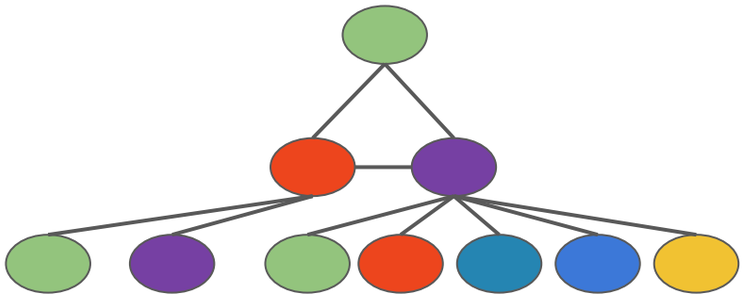


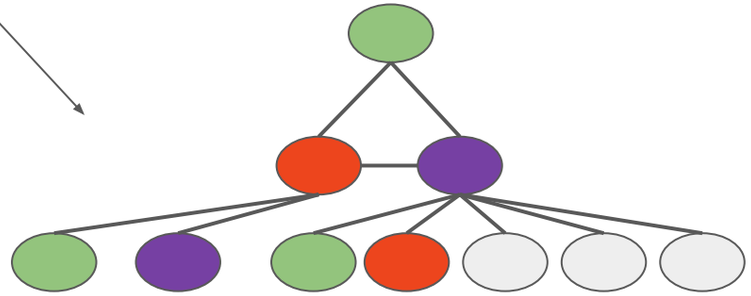
GNNExplainer (Probability distributions)

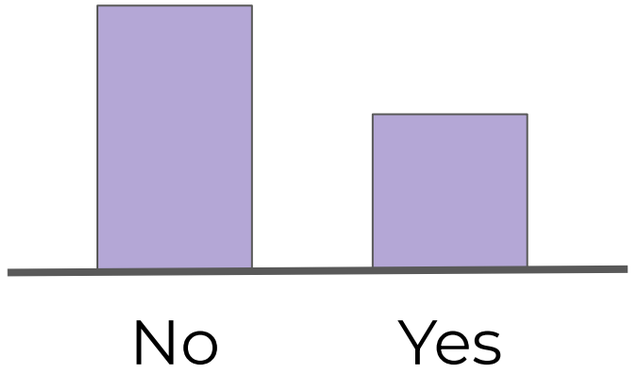












GNNExplainer (Formal)

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y \mid G = G_S, X = X_S)$$

Human-interpretable intuition

Find a subgraph that maintains as much information as possible compared to the full graph

GNNExplainer (Formal)

$$\max_{G_S} MI(Y, (G_S, X_S)) = \underbrace{H(Y)}_{\text{Full comp. graph}} - H(Y \mid \underbrace{G = G_S}_{\text{Subgraph}}, X = X_S)$$

GNNExplainer (Formal)

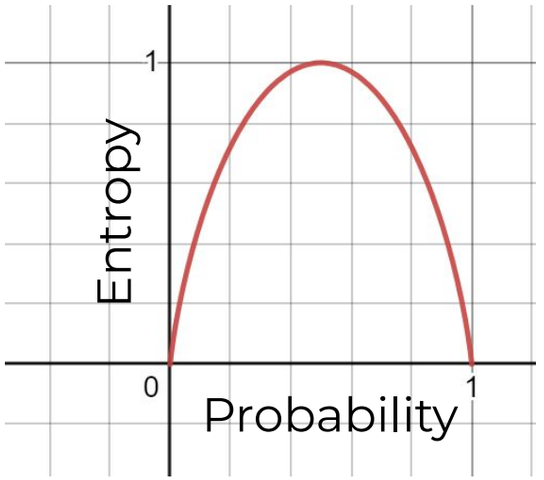
$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y \mid G = G_S, X = X_S)$$

Subgraph of the comp. graph

Subset of node features

GNNExplainer (Formal)

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y \mid G = G_S, X = X_S)$$



$$-\mathbb{E}_{Y|G_S,X_S}[\log P_\Phi(Y|G = G_S, X = X_S)]$$

GNNExplainer (Formal)

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y \mid G = G_S, X = X_S)$$

$$-\mathbb{E}_{Y|G_S,X_S}[\log P_\Phi(Y|G = G_S, X = X_S)]$$

$$\min_M -\sum_{c=1}^{C} 1[y = c] \log P_\Phi(Y = y|G = A_c \odot \sigma(M), X = X_c)$$

Element-wise multiplication of adj. matrix and a learnable mask

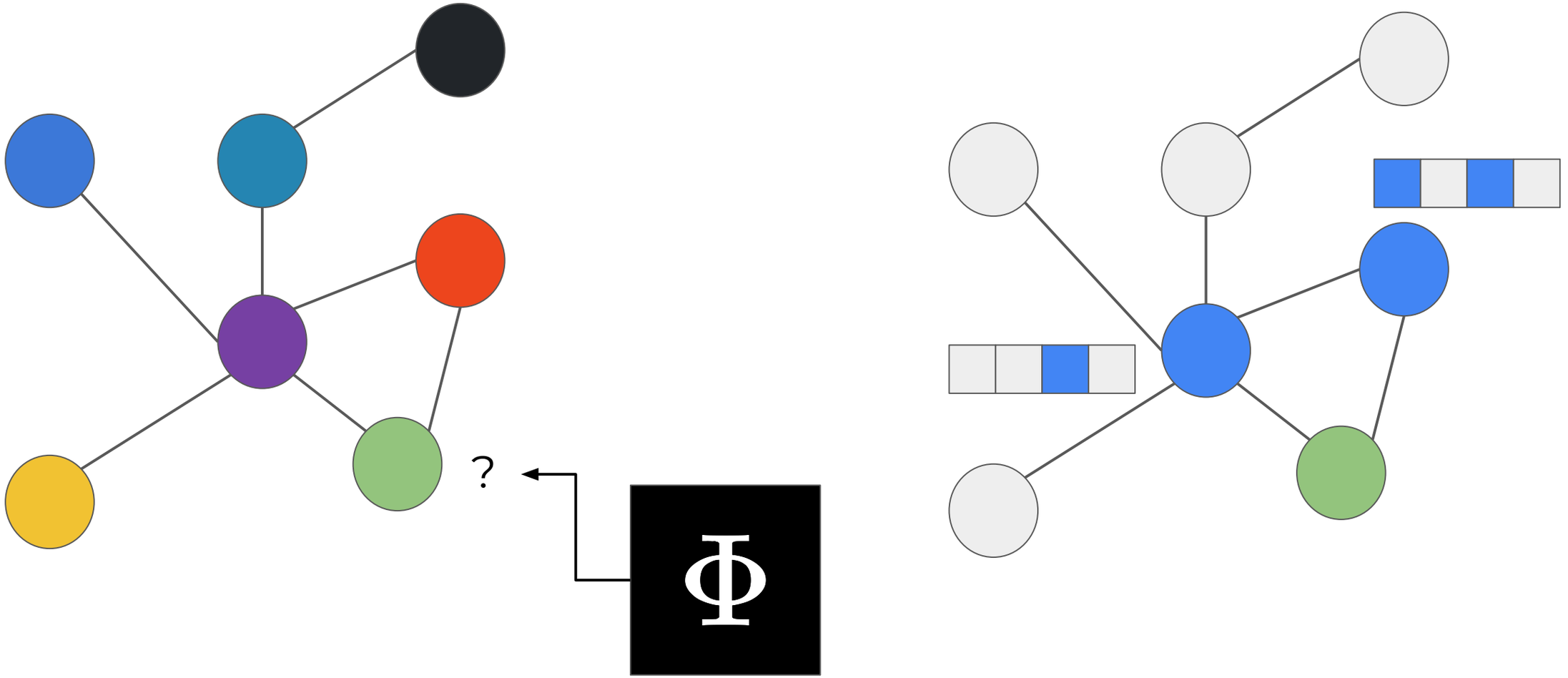
GNNExplainer (Formal)

$$\min_{M} -\sum_{c=1}^{C} 1[y=c] \log P_{\Phi}(Y=y|G=A_c \odot \sigma(M), X=X_c)$$

If you need further explanations around this formula, check out this paper

Li X, Saúde J. *Explain graph neural networks to understand weighted graph features in node classification*. In International Cross-Domain Conference for Machine Learning and Knowledge Extraction 2020 Aug 18 (pp. 57-76). Cham: Springer International Publishing.

GNNExplainer (Explanation Example)



GraphLIME in a Nutshell

- GraphLIME builds **local interpretable models** for specific nodes in the graph
- It uses the **Hilbert-Schmidt Independence Criterion (HSIC) Lasso** for feature selection.
- GraphLIME operates **locally within the subgraph** of the node being explained.

Selecting the Node's Neighborhood

- For a target node, GraphLIME selects its **N-hop neighborhood** (where N is the number of GNN layers)

- This neighborhood captures relevant **structural information** around the node

Building the Local Model

- GraphLIME learns a **nonlinear interpretable model** within the selected subgraph
- This model explains the node's **prediction behavior**
- GraphLIME computes the **K most representative features** using HSIC Lasso

Feature Importance with HSIC Lasso

- HSIC Lasso is a **nonlinear method** for selecting important features
- It balances the trade-off between **model complexity and interpretability**
- HSIC Lasso identifies the most influential features for the node's prediction

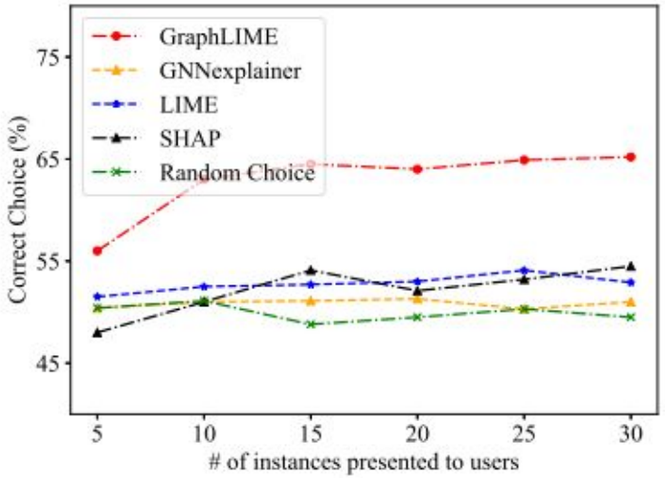
GRAPHLIME'S ALGORITHM

Algorithm 1. Locally Nonlinear Explanation: GraphLIME

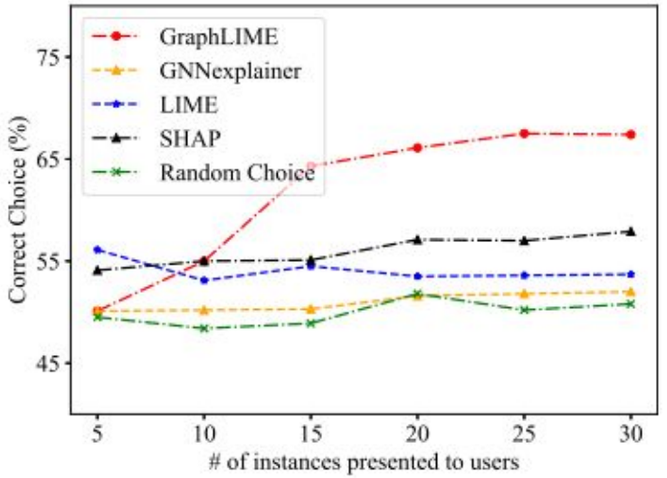
Input: GNN classifier f, Number of explanation features K
Input: the graph $\mathcal{G}$, the node v being explained
Output: K explanation features

$X_n = N\_hop\_neighbor\_sample(v)$
$Z \leftarrow \{\}$
for all $x_i \in X_n$ **do**
    $y_i = f(\boldsymbol{x}_i)$
    $Z \leftarrow Z \cup (\boldsymbol{x}_i, y_i)$
end for
$\boldsymbol{\beta} \leftarrow$ HSIC Lasso(Z) ▷ with x_i as features, y_i as label
$\zeta(v) \leftarrow$ Top-K features as explanations based on $\boldsymbol{\beta}$

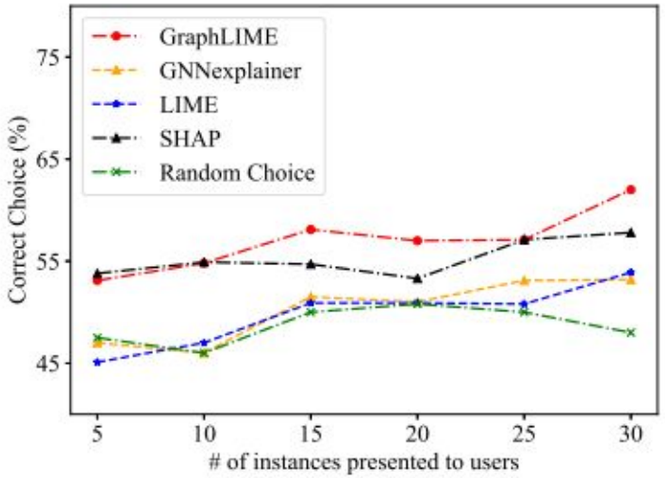
GRAPHLIME VS GNNEXPLAINER



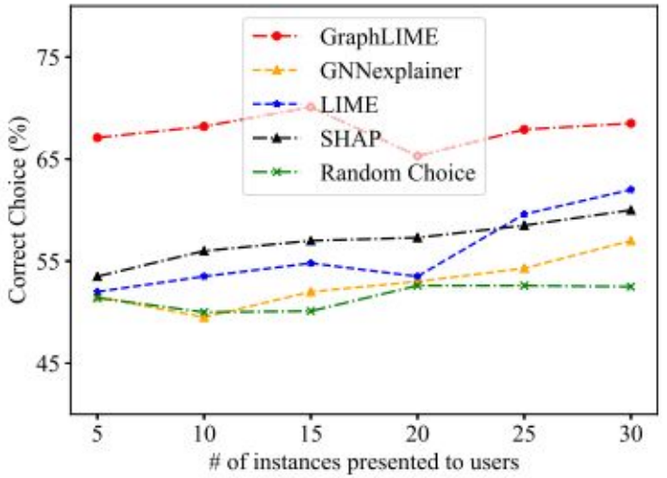
(a) Cora for GraphSAGE



(b) Pubmed for GraphSAGE



(c) Cora for GAT



(d) Pubmed for GAT

Part III

COUNTERFACTUAL EXPLANATIONS IN GRAPHS

by Mario A. Prado-Romero

Based on:

Prado-Romero et al. “A survey on graph counterfactual explanations: definitions, methods, evaluation”, ACM CSUR 2023

Graph Counterfactual Explanation

Multi-class minimal counterfactual examples: Let Φ be a prediction model that classifies x into a class $c \in C$ from a set of classes C. Let X' be the set of possible counterfactual examples x' and $S_{inst}(x, x')$ be a similarity measure that tells how similar x' is to x. Then, we define the set of counterfactual examples w.r.t. Φ as follows:

$$s(c', x) := \max_{x' \in X', x \neq x'} \{\mathcal{S}_{inst}(x, x') \mid \Phi(x') = c'\}$$

$$\mathcal{E}_\Phi(x) = \bigcup_{c' \in C - \{c\}} \{x' \in X' \mid x \neq x', \mathcal{S}_{inst}(x, x') = s(c', x)\}$$

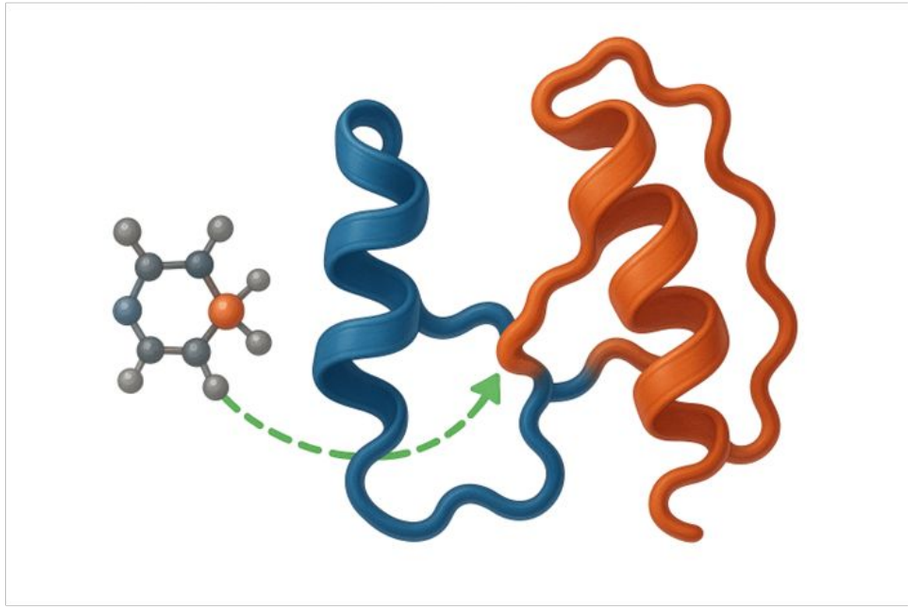
Global minimal counterfactual example: Let Φ be a prediction model that classifies x into a class $c \in C$. Let X' be the set that contains all the possible counterfactual examples x'. We define the global minimal counterfactual example $\mathcal{E}^*_\Phi(x)$ of x, as follows:

$$\mathcal{E}^*_\Phi(x) = \arg\max_{x' \in X'} \mathcal{S}_{inst}(x, x')$$

Graph Counterfactual Applications

Road design in Transportation Networks

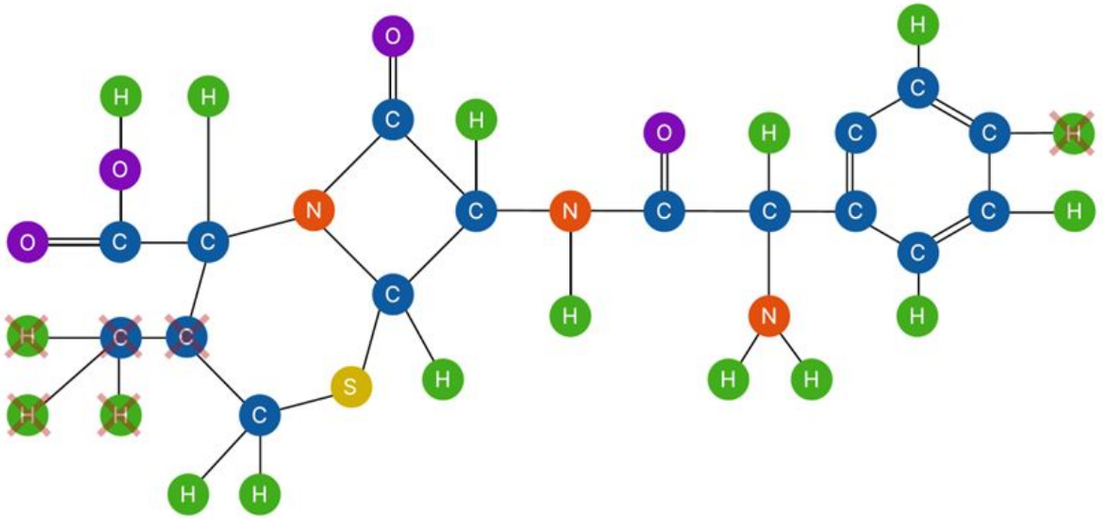
Link prediction in Social Networks



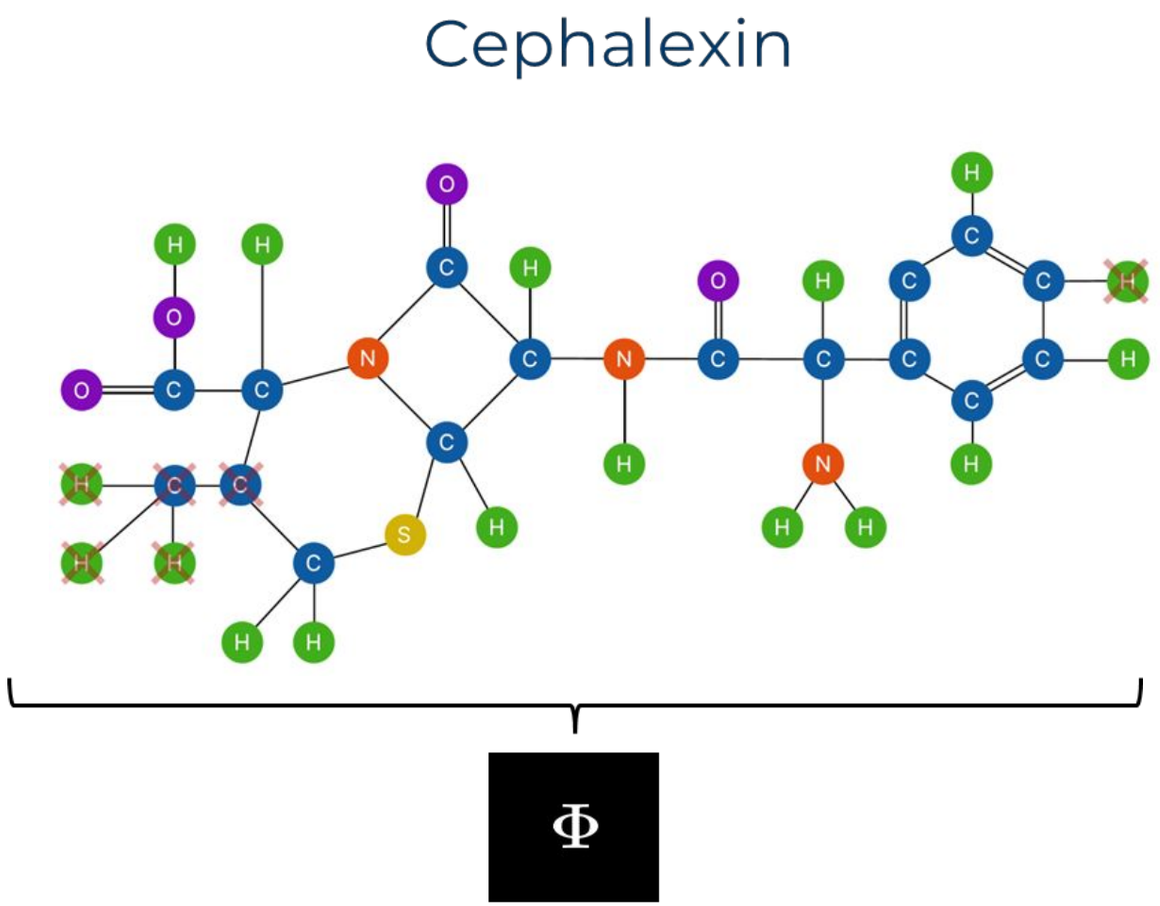
Protein-Ligand binding affinity prediction

GCE for Drug Discovery

Cephalexin

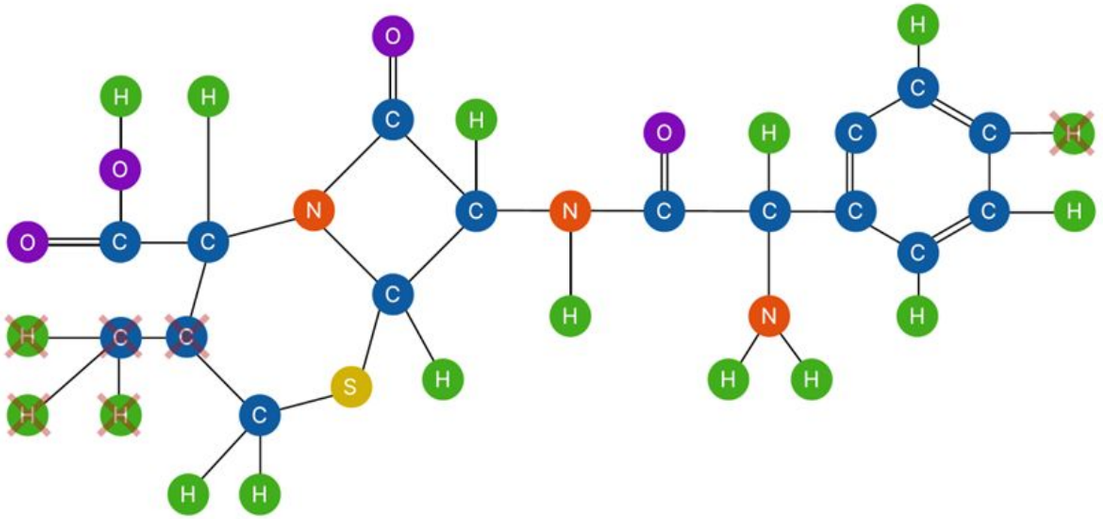


GCE for Drug Discovery

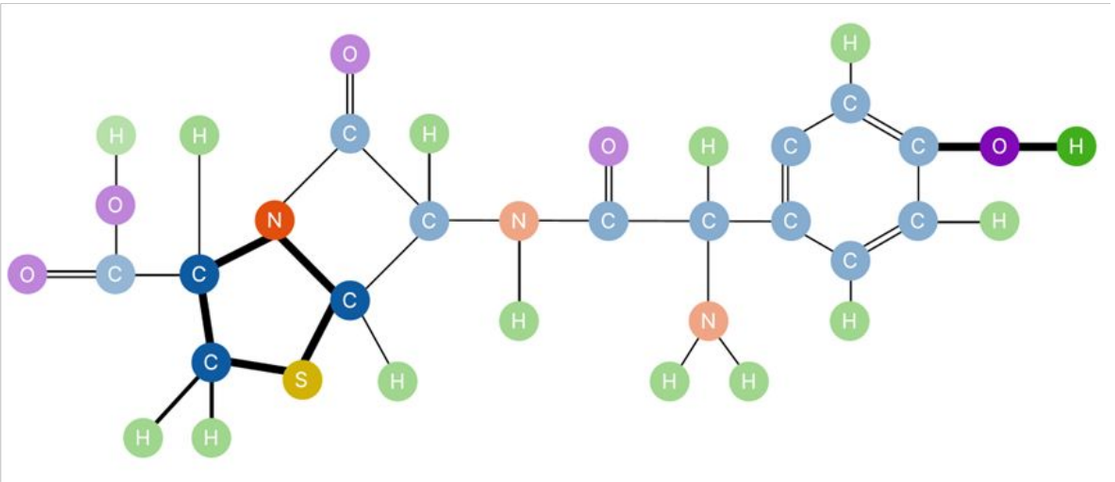


GCE for Drug Discovery

Cephalexin



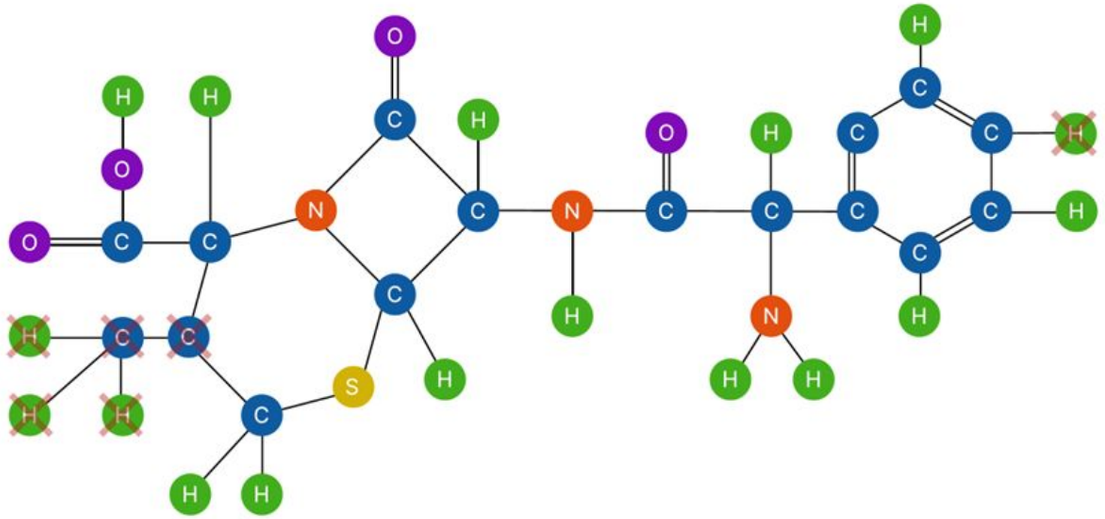
Amoxicillin



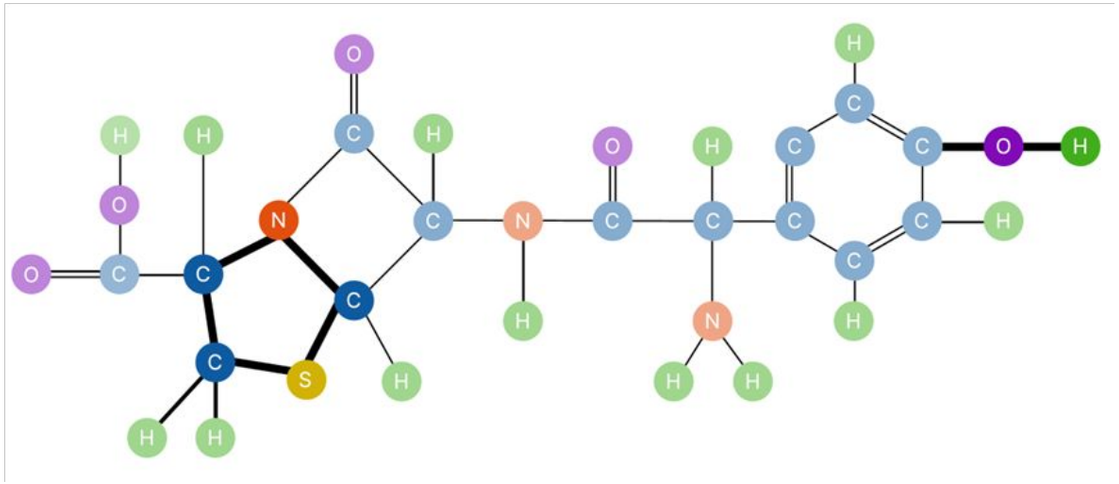
Headache

GCE for Drug Discovery

Cephalexin



Amoxicillin



Headache

No Headache

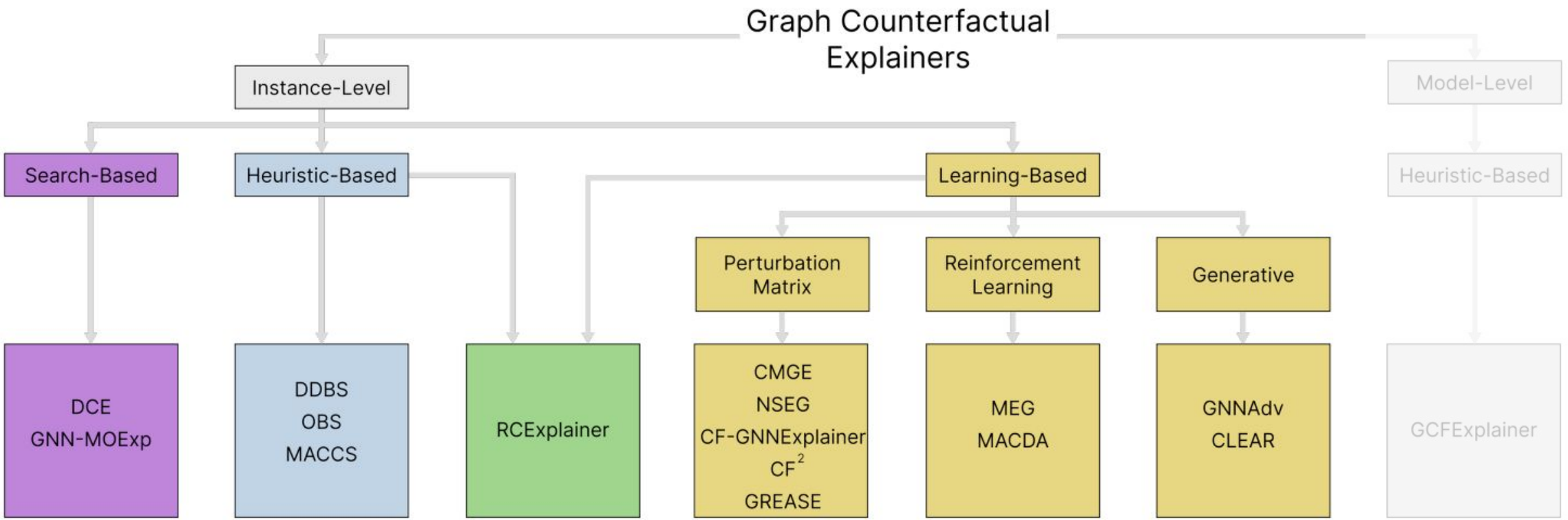
Challenges in generating GCEs

- Structural Constraints
- Ensuring plausibility and feasibility of the counterfactual
- Computational complexity

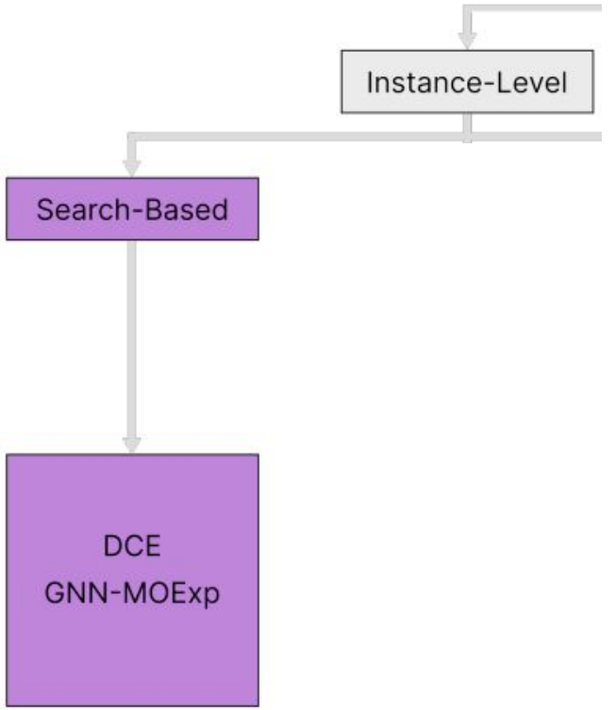
Graph Counterfactual Explanation Methods

Prado-Romero et al. “A survey on graph counterfactual explanations: definitions, methods, evaluation”, ACM CSUR 2023

GCE Methods

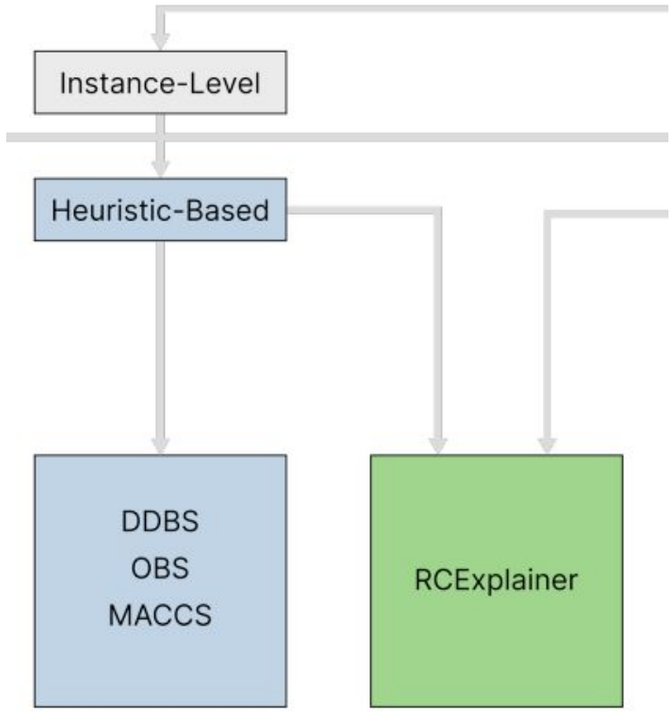

Graph Counterfactual Explainers
Instance-Level
Model-Level
Search-Based
Heuristic-Based
Learning-Based
Heuristic-Based
Perturbation Matrix
Reinforcement Learning
Generative
DCE
GNN-MOExp
DDBS
OBS
MACCS
RCExplainer
CMGE
NSEG
CF-GNNExplainer
CF^2
GREASE
MEG
MACDA
GNNAdv
CLEAR
GCFExplainer

Search-Based GCE Methods



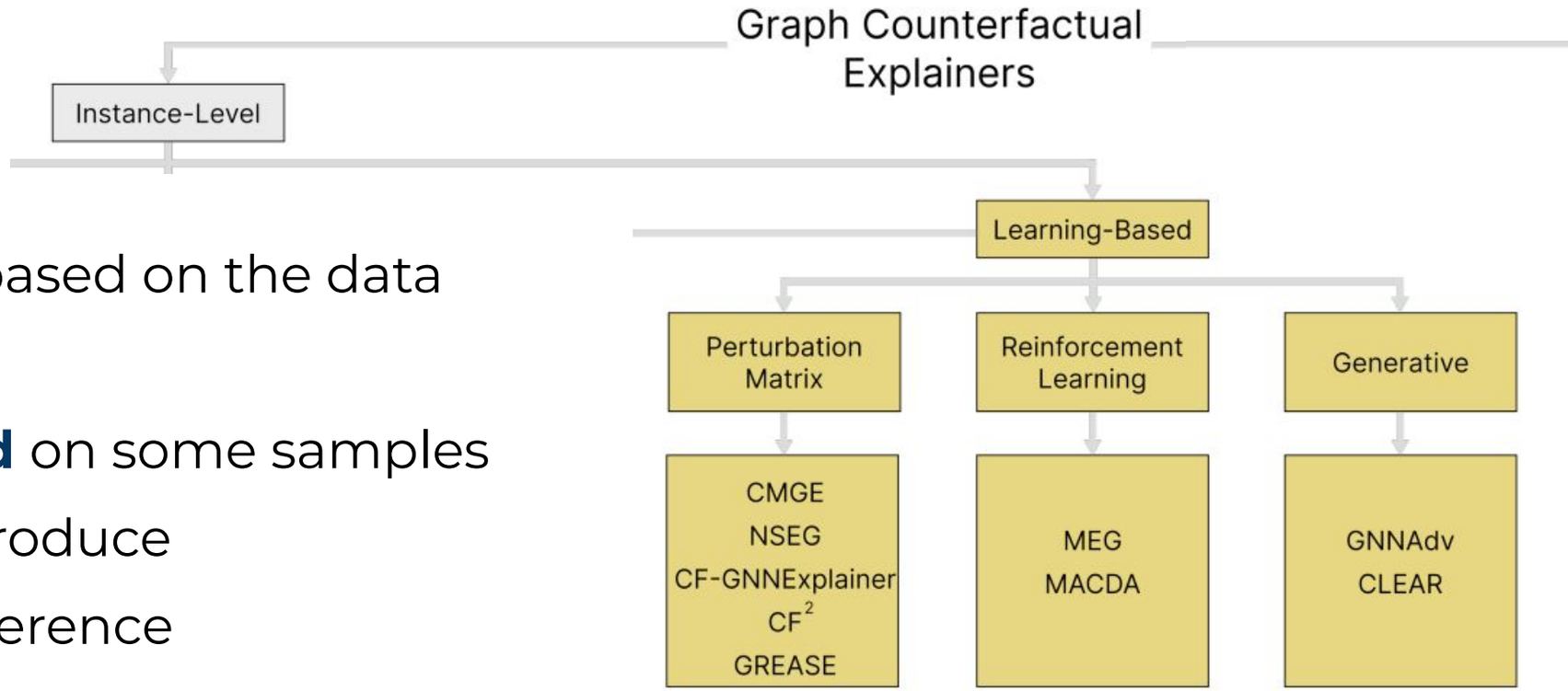
- Find a counterfactual **within the data**
- For a graph $\boldsymbol{G} \in \mathcal{G}$ find a $\boldsymbol{G'} \in \mathcal{G}$ s.t. $\boldsymbol{\Phi}(\boldsymbol{G}) \neq \boldsymbol{\Phi}(\boldsymbol{G'})$
- These methods **fail** to produce a counterfactual if the explainer **cannot access** the original **dataset**

Heuristic-Based GCE Methods



- **Perturb** the original graph such that $\Phi(G) \neq \Phi(G')$ **without** accessing the original dataset
- **Requires** to define the perturbation **rules** after a **careful examination** of the data

Learning-Based GCE Methods

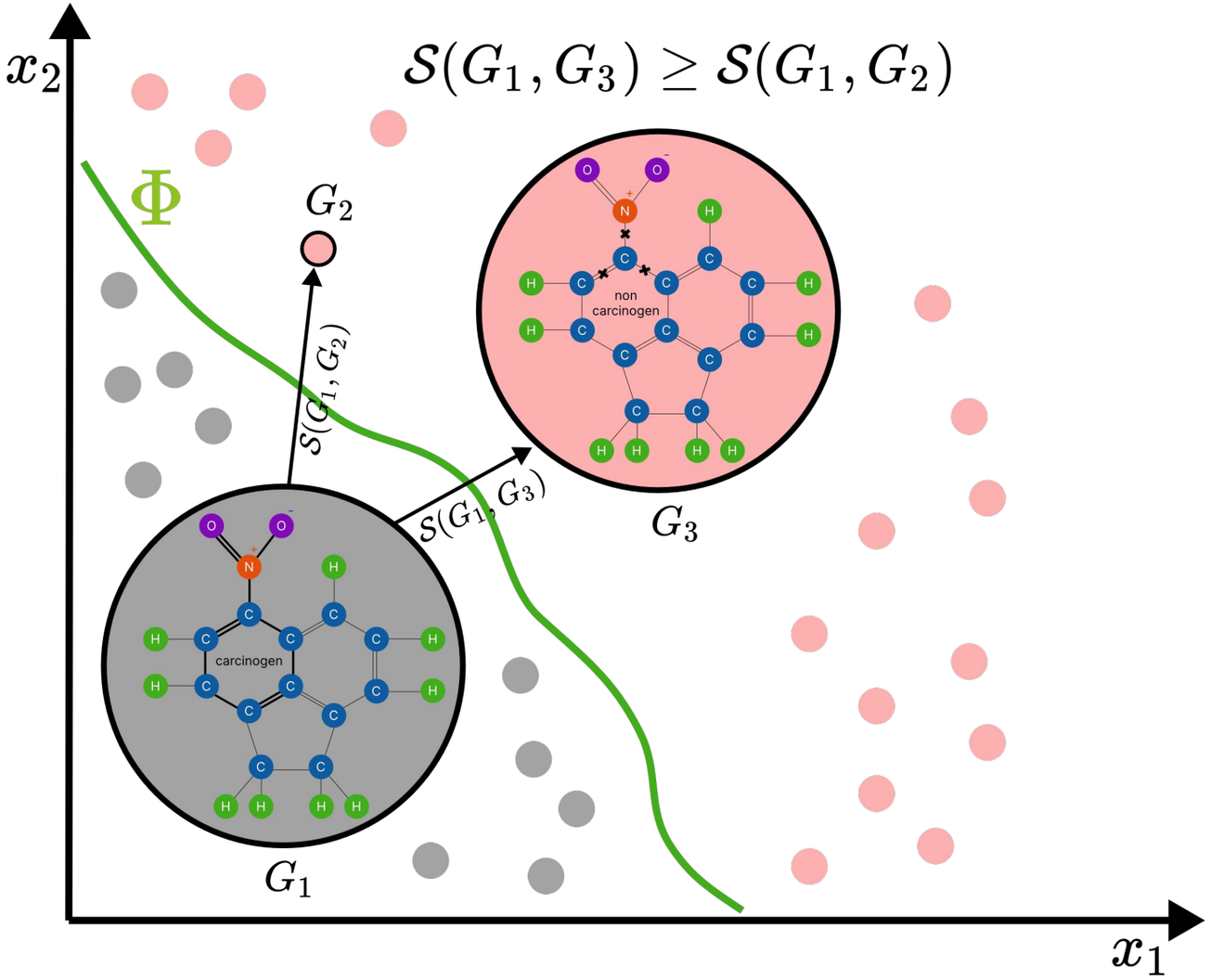


- **Learn the heuristic** based on the data
- Explainers are **trained** on some samples and can be used to produce counterfactuals at inference

BASELINE GRAPH COUNTERFACTUAL EXPLAINER

L. Faber, A. K. Moghaddam, and R. Wattenhofer. 2020. Contrastive Graph Neural Network Explanation. In Proc. of the 37th Graph Repr. Learning and Beyond Workshop at ICML 2020. Int. Conf. on Machine Learning, 28

DCE (Distribution Complaint Explanations)



DCE

$$G^* = \arg \min_{G' \in \mathcal{G}, \Phi(G) \neq \Phi(G')} d(G, G')$$

Questions on DCE

- Does it guarantee to always produce valid counterfactuals?
- Does it guarantee to produce the closest counterfactual to the input we want to explain?
- Are counterfactuals within the data manifold?

HEURISTIC-BASED EXPLAINERS

Abrate C, Bonchi F. Counterfactual graphs for explainable classification of brain networks. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining 2021 Aug 14 (pp. 2495-2504).

Wellawatte GP, Gandhi HA, Seshadri A, White AD. A Perspective on Explanations of Molecular Prediction Models. Journal of Chemical Theory and Computation. 2023 Mar 27;19(8):2149-60.

OBS & DDBS
Oblivious and **D**ata-**D**riven **B**idirectional **S**earch

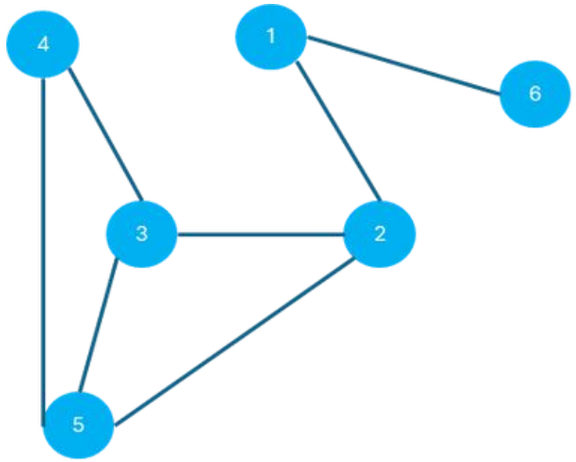
Oblivious Bidirectional Search (OBS)

- **Select a graph instance to explain:** The method is designed for explaining graph classifiers

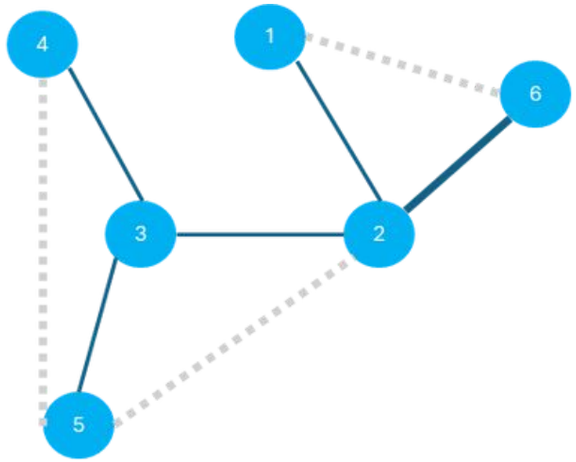
- **Perturb edges until a counterfactual is reached:** Randomly modify the edges of the graph x until the prediction of the model for the perturbed instance x' is different to the prediction for the original instance

- **Reduce the distance between x and x':** Try to undo as much as possible the changes made to obtain the x' while keeping its predicted label different from that of x

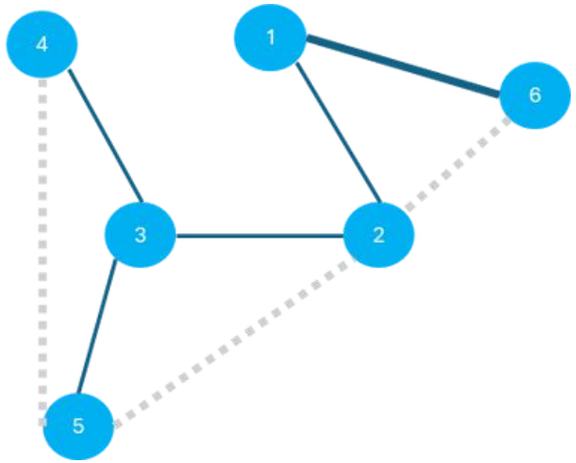
Oblivious Bidirectional Search (OBS)



Cyclic Graph



Step 1:
Find a
Counterfactual



Step 2:
Reduce distance between
original graph and
counterfactual

Data-Driven Bidirectional Search

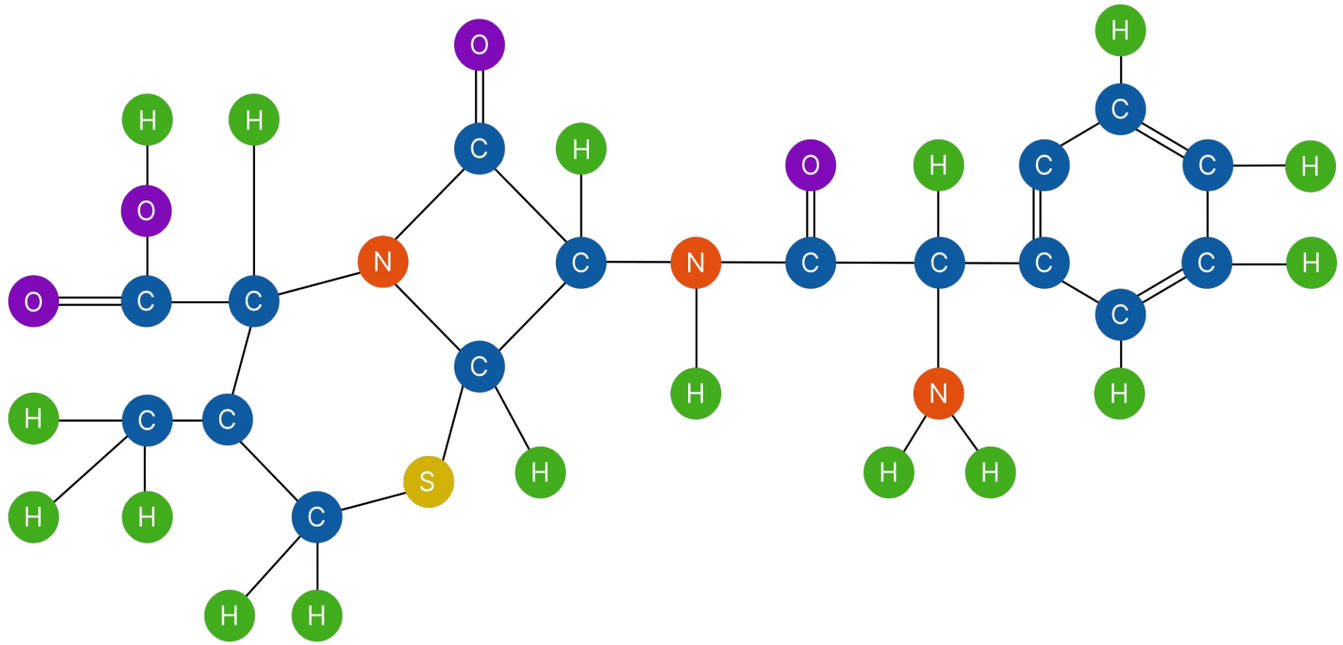
- **Select a graph instance to explain**. The method is designed for explaining graph classifiers.
- **Sort edges** according to their frequency of appearance in each class
- **Perturb edges until a counterfactual is reached**. Considering the order of the edges for the counterfactual class, modify the edges of the graph x until the prediction of the model for the perturbed instance x' is different to the prediction for the original instance
- **Reduce the distance between x and x'**. Try to undo as much as possible the changes made to obtain the x' while keeping its predicted label different from that of x

MACCS
Model Agnostic Counterfactual Compounds with STONED[1]

1. Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) method for exploration of chemical space

MACCS

Uses SMILES representations of molecules



CC1=C(N2C(C(C2=O)NC(=O)C(C3=CC=CC=C3)N)SC1)C(=O)O

MACCS

- Expand the chemical space around the original molecule
- Select some molecules from the expansion space via **clustering**
- Choose multiple ones **closest** to the input

MACCS (EXAMPLE)

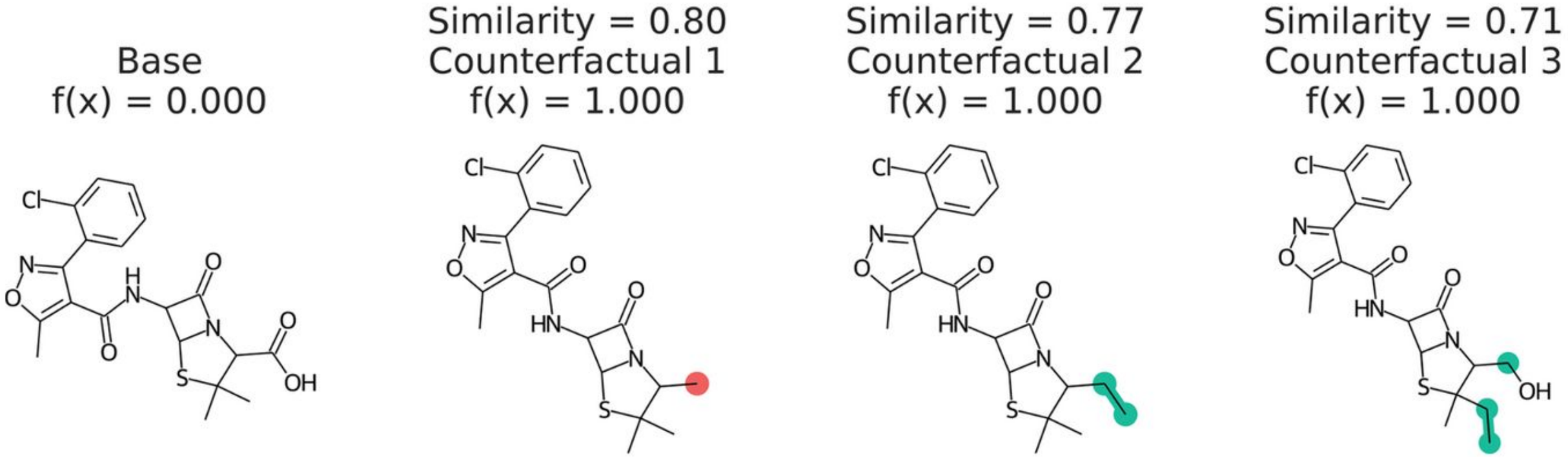


Figure 1. Counterfactuals of a molecule which cannot permeate the blood–brain barrier. Similarity is the Tanimoto similarity of ECFP4 fingerprints.[130] Red indicates deletions and green indicates substitutions and addition of atoms. Republished from ref 9 with permission from the Royal Society of Chemistry. Copyright 2022.

LEARNING-BASED EXPLAINERS

D. Numeroso and D. Bacciu. 2021. Meg: Generating molecular counterfactual explanations for deep graph networks. In 2021 Int. Joint Conf. on Neural Networks. IEEE, 1–8

Wellawatte GP, Gandhi HA, Seshadri A, White AD. A Perspective on Explanations of Molecular Prediction Models. Journal of Chemical Theory and Computation. 2023 Mar 27;19(8):2149-60.

Tan, J., Geng, S., Fu, Z., Ge, Y., Xu, S., Li, Y. and Zhang, Y., 2022, April. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In Proceedings of the ACM Web Conference 2022 (pp. 1018-1027).

Ma, J., Guo, R., Mishra, S., Zhang, A. and Li, J., 2022. Clear: Generative counterfactual explanations on graphs. Advances in Neural Information Processing Systems, 35, pp.25895-25907.

Prado-Romero MA, Prenkaj B, Stilo G. Robust Stochastic Graph Generator for Counterfactual Explanations. AAAI'24

Prenkaj B, Zaradoukas E, Kasneci G. Graph Inverse Style Transfer for Counterfactual Explanations. ICML'25

MEG
Molecular Explanation Generator

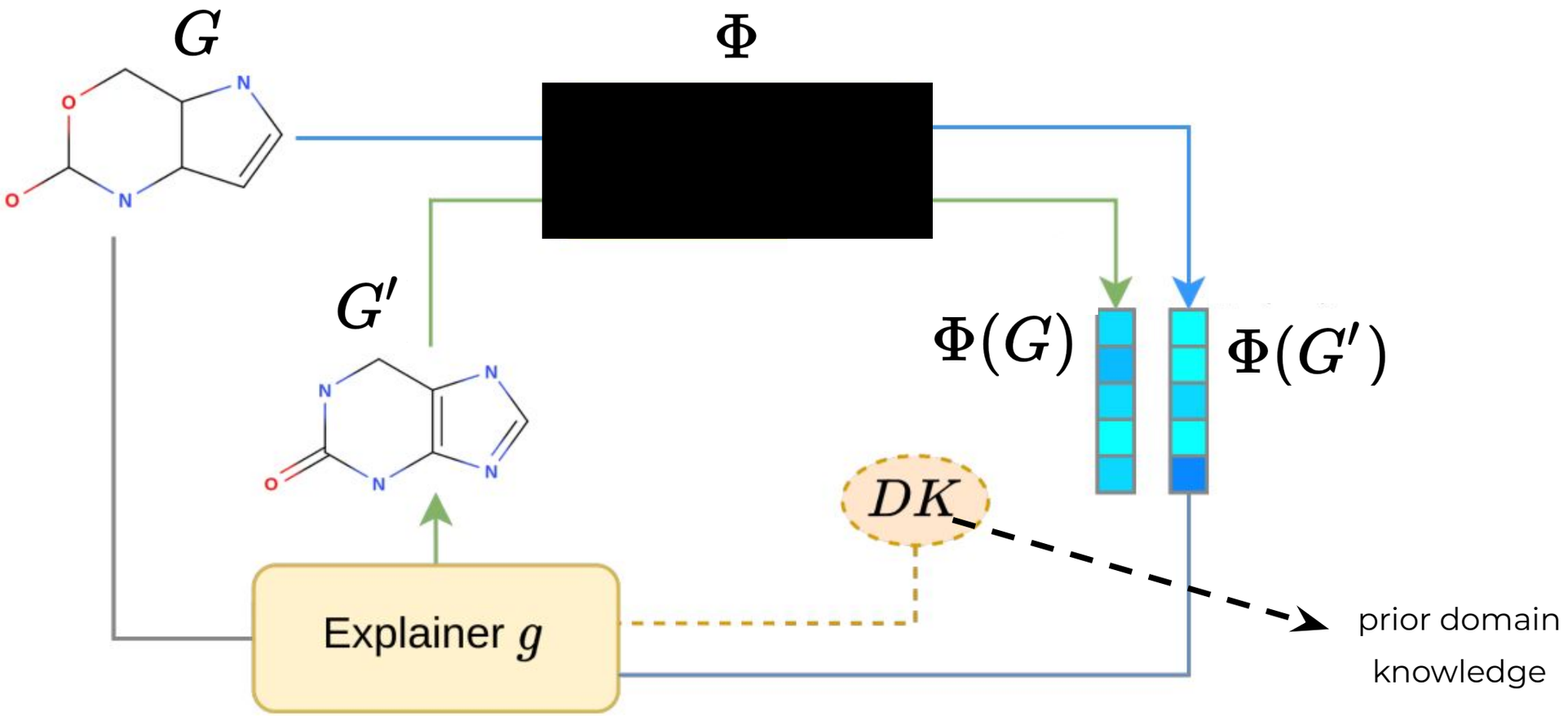
MEG

- **Reinforcement Learning-based Explainer**. It is specifically designed for the molecular domain, however, can be adapted to other domains.
- **The initial state is the original instance to be explained**. From there the RL-Agent will learn to modify it to generate a counterfactual
- **The set of actions considers domain knowledge**. It is based on the MolDQN model to ensure the atom and bonds additions/removals produce valid molecules
- **The multi-objective reward** function exploits the prediction of the model Φ

MEG

- **Goodness of an action-state pair**. It is evaluated by a function Q that is approximated using a deep network
- **Policy**. It is a function outputting the best possible action; uses a decaying ϵ-greedy
- **The training is performed individually for each instance**

MEG (ARCHITECTURE)



MEG

$$\arg\min_{G' \in \mathcal{G}'} (1-\alpha) \cdot \mathcal{S}(\Phi(G), \Phi(G')) + \alpha \cdot d(G, G')$$

Similarity of the predictions between

the original molecule G and the counterfactual G’

MEG

$$\arg\min_{G' \in \mathcal{G}'} (1-\alpha) \cdot \mathcal{S}(\Phi(G), \Phi(G')) + \alpha \cdot d(G, G')$$

The distance between the original molecule G and the counterfactual G' needs to be small

In molecular jargon: maintain chemical-physical characteristics of the original molecule

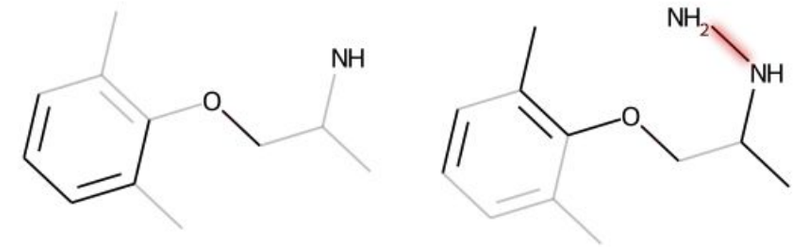
MEG (Graph Distance)

$$d(G, G') = \alpha_1 \frac{G \cdot G'}{||G||^2 + ||G'||^2 - G \cdot G'} + \alpha_2 \frac{\mathbf{h}(G) \cdot \mathbf{h}(G')}{||\mathbf{h}(G)|| \; ||\mathbf{h}(G')||}$$

Tanimoto similarity over the Morgan fingerprints of G and G'

The hidden representation of the graph w.r.t. the oracle

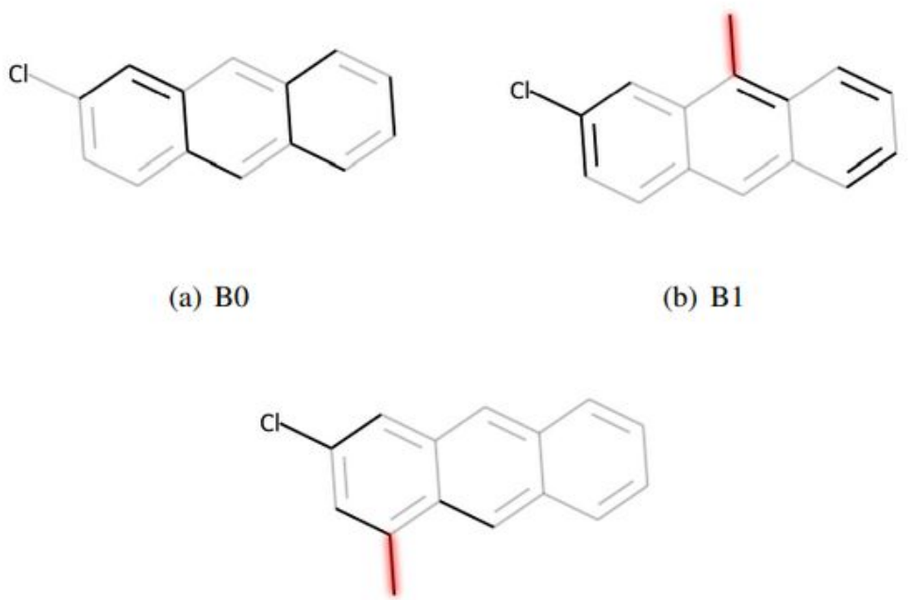
MEG (SOME RESULTS)



(a) A0 (b) A1

(c) A2 (d) A3

Fig. 2: Counterfactual explanations (A1-3) for a non-toxic molecule (A0).



(a) B0 (b) B1

(c) B2

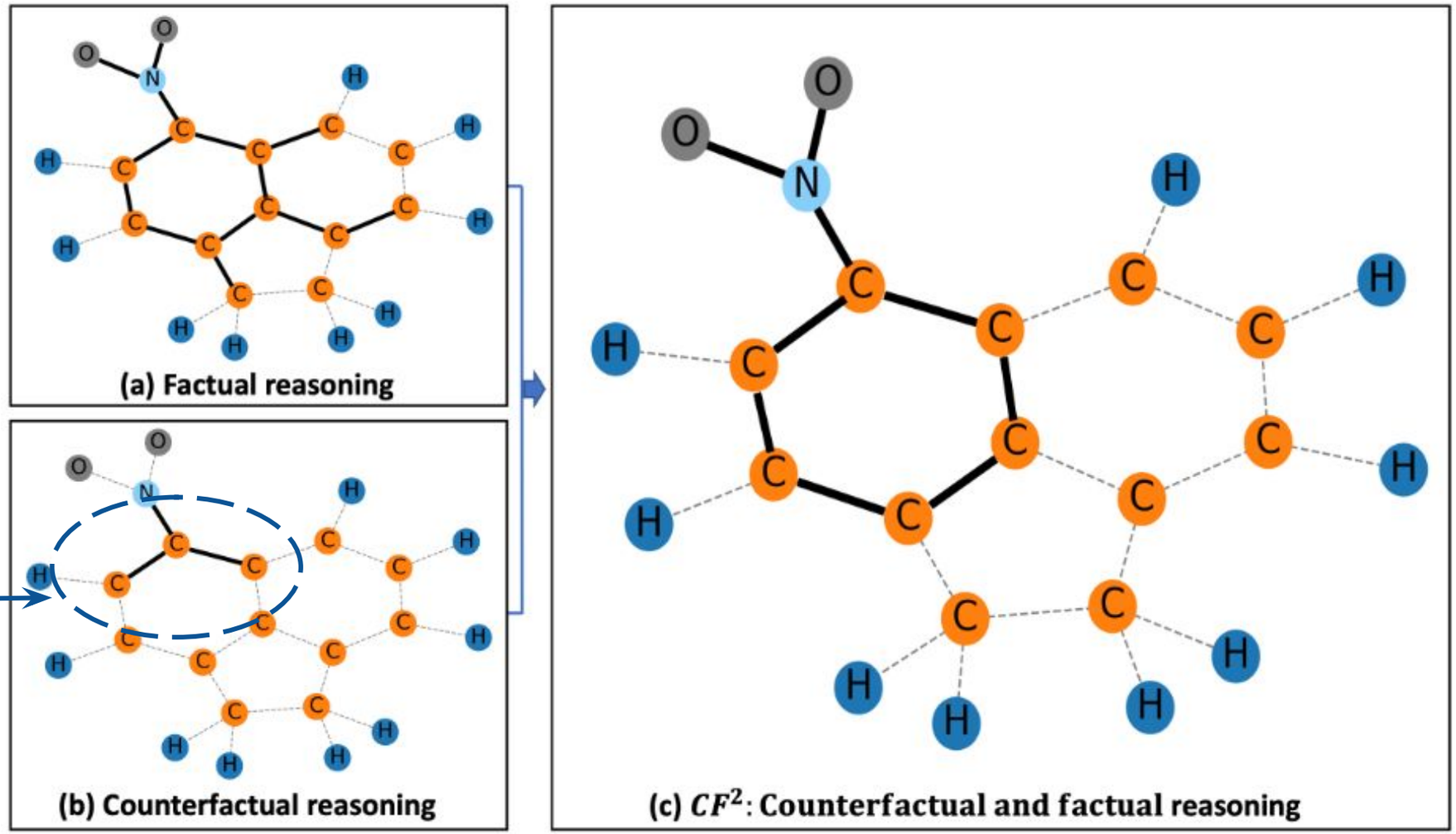
Fig. 3: Toxic molecule (B0) classified as non toxic after the addition of one atom of carbon (B1-2).

CF^2

Counter**F**actual and **F**actual explanations

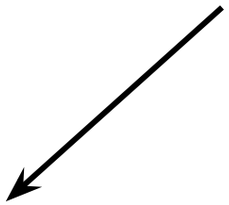
CF²

these edges, if removed, create the counterfactual



CF² (Necessity & Sufficiency)

An explanation needs to be **necessary** and **sufficient**



Counterfactual Reasoning

Factual Reasoning

CF² (Sufficiency)

Factual Reasoning

$$\Phi(A * M, X * F) = y$$

Masked adjacency matrix

Masked node feature vectors

Both masks are learned by the explainer

CF² (Necessity)

Counterfactual Reasoning

$$\Phi(A - A*M, X - X*F) \neq y$$

Why do we subtract the masked adjacency matrix?

CF²

- When there are multiple explanations rely on Occam's Razor

- Measure the complexity of the explanation

$$C(M, F) = ||M||_1 + ||F||_1$$

CF² (Explanation Cost)

- Measure the complexity of the explanation

$$C(M, F) = ||M||_1 + ||F||_1$$

- We need to define the **strength of the explanation** (factual and counterfactual)

$$S_f(M, F) = P(\Phi(A{*}M, X{*}F) = y) \quad \textbf{Factual}$$

$$S_c(M, F) = -P(\Phi(A - A{*}M, X - X{*}F) = y) \quad \textbf{Counterfactual}$$

CF² (Optimization Function)

$$\textbf{minimize } C(M, F)$$
$$\textbf{s.t. } S_f(M, F) > P(\Phi(A * M, X * F) = \neg y)$$
$$S_c(M, F) > -P(\Phi(A - A * M, X - X * F) = \neg y)$$

- When using information in the explanation sub-graph, the predicted label's probability is higher than any other label **(Prediction doesn't change)**

- When the information in the explanation sub-graph is removed, the predicted label's probability will be smaller than at least another label **(Prediction changes)**

CF² (Relaxed Optimization)

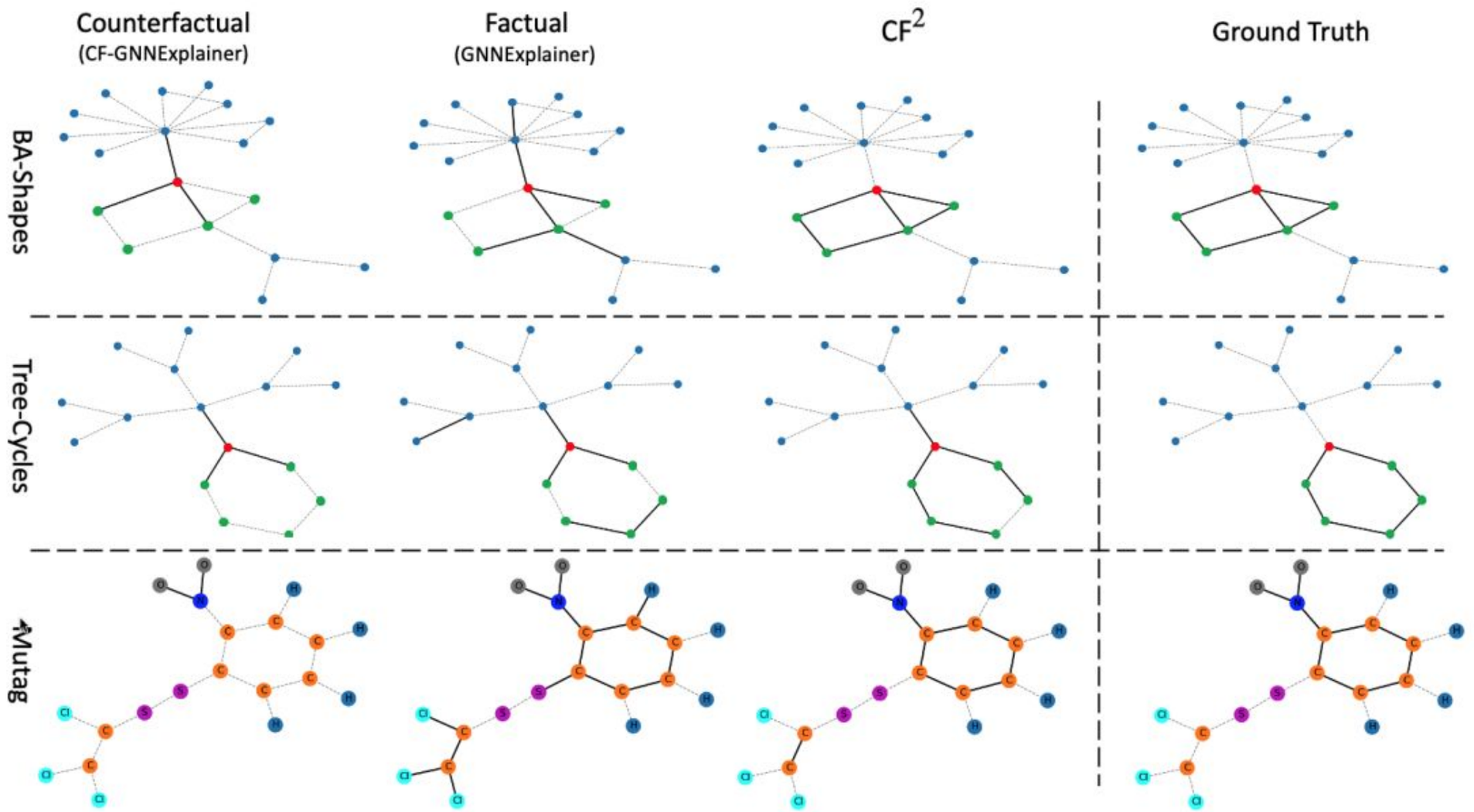
$$L_f = \mathbf{ReLU}(\gamma + P(\Phi(A * M, X * F) = \neg y) - S_f(M, F))$$

$$L_c = \mathbf{ReLU}(\gamma - P(\Phi(A - A * M, X - X * F) = \neg y) - S_c(M, F))$$

} Relax the constraints into a pairwise contrastive loss

$$\mathbf{minimize}\ ||M||_1 + ||F||_1 + \lambda(\alpha L_f + (1 - \alpha)L_c)$$

CF² (Qualitative Results)



CLEAR

Generative **C**ounterfactua**L** **E**xpl**A**nation gene**R**ator for graphs

CLEAR

First work to treat **causality** when producing counterfactuals

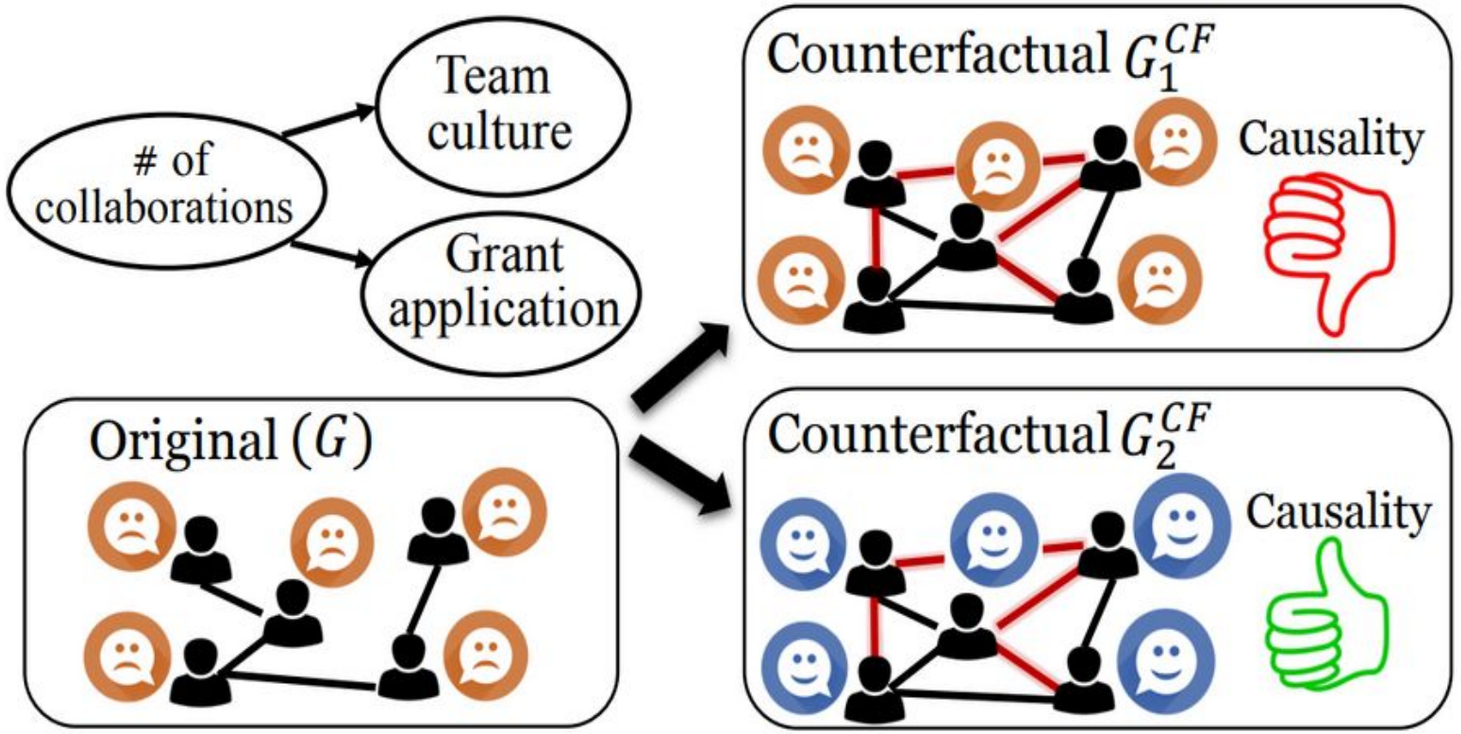


Image taken from: Ma J, Guo R, Mishra S, Zhang A, Li J. Clear: Generative counterfactual explanations on graphs. Advances in Neural Information Processing Systems. 2022 Dec 6;35:25895-907.

CLEAR

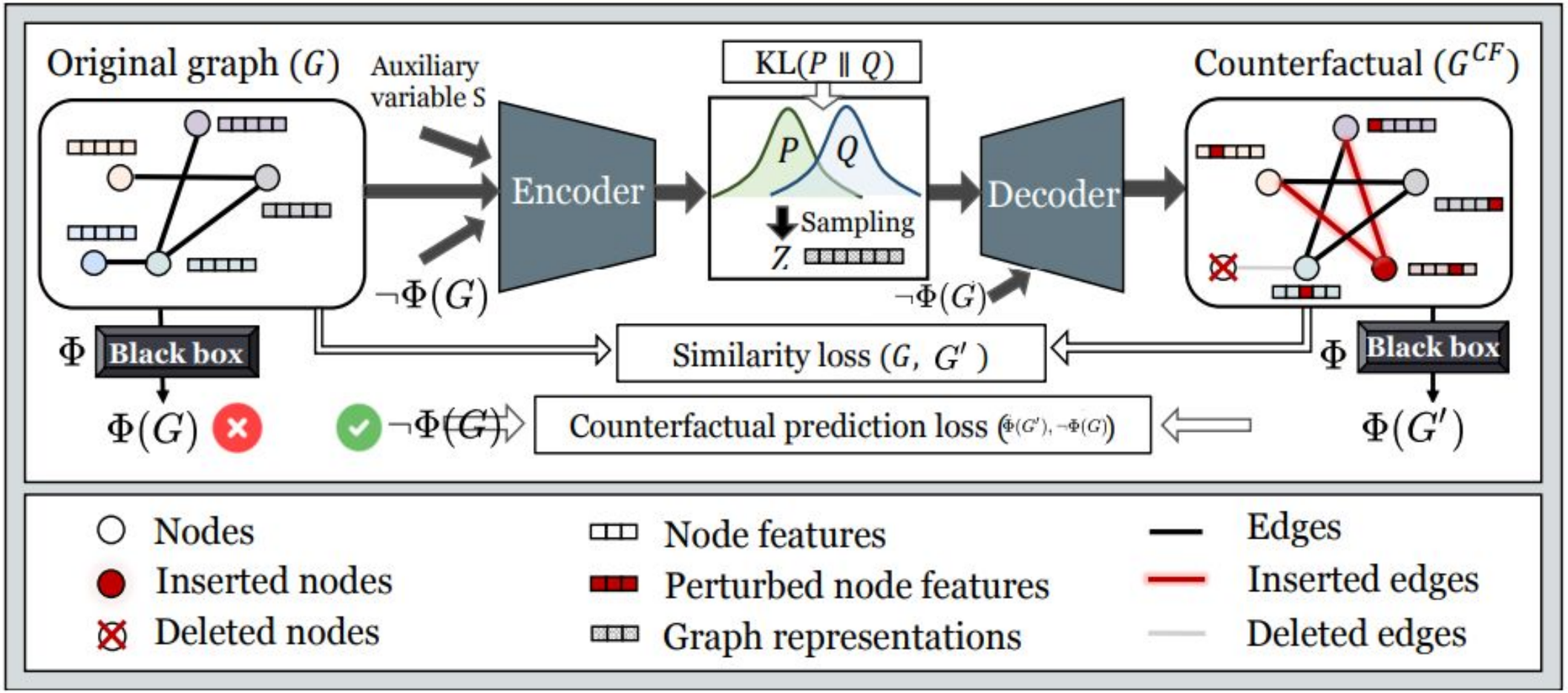


Image taken from: Ma J, Guo R, Mishra S, Zhang A, Li J. Clear: Generative counterfactual explanations on graphs. Advances in Neural Information Processing Systems. 2022 Dec 6;35:25895-907.

CLEAR

$$\ln P(G' \mid \neg\Phi(G), G) \geq \mathbb{E}_Q \left[\ln P(G' \mid Z, \neg\Phi(G), G) \right] - \mathrm{KL}\left(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \right)$$

CLEAR

$$\ln P(G' \mid \neg\Phi(G), G) \geq \mathbb{E}_Q \left[\ln P(G' \mid Z, \neg\Phi(G), G) \right] - \mathrm{KL}\left(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \right)$$

log likelihood of the probability of the generated counterfactual conditioned on the hidden representation Z of G and the opposite class of G

CLEAR

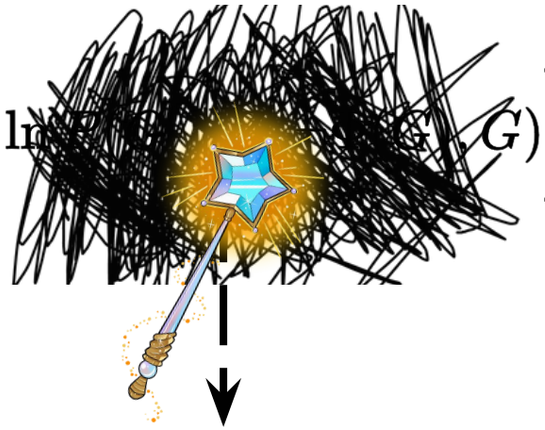
$$\ln P(G' \mid \neg\Phi(G), G) \geq \mathbb{E}_Q \left[\ln P(G' \mid Z, \neg\Phi(G), G) \right] - \mathrm{KL}\left(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \right)$$

This term is difficult to optimize since we don't have a ground-truth for counterfactuals

SO...

CLEAR

$$\ln P(G' \mid \neg\Phi(G), G) \geq \mathbb{E}_Q \left[\ln P(G' \mid Z, G) \right] - \mathrm{KL}\Big(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \Big)$$



$$-\mathbb{E}_Q \Big[d(G, G') + \alpha \ell\big(\Phi(G'), \neg\Phi(G)\big) \Big]$$

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. arXiv preprint, 2019.

CLEAR

$$\ln P(G' \mid \neg\Phi(G), G) \geq \mathbb{E}_Q \left[\ln P(G' \mid Z, G) \right] - \mathrm{KL}\Big(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \Big)$$

$$-\mathbb{E}_Q \Big[d(G, G') + \alpha \ell\big(\Phi(G'), \neg\Phi(G)\big) \Big]$$

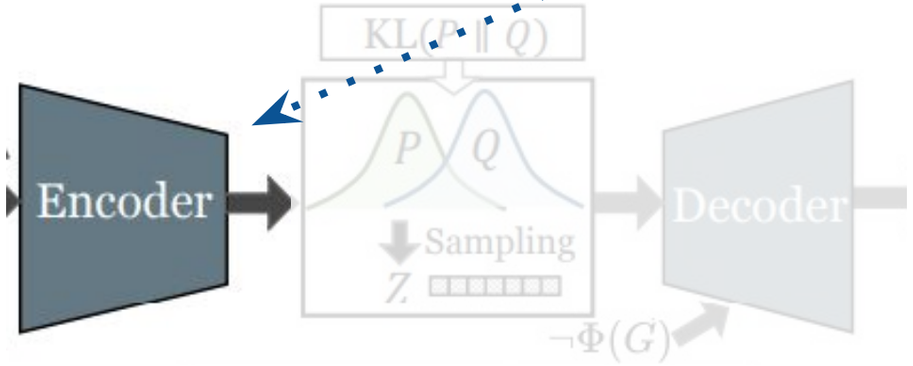
distance/similarity function

counterfactual loss (measure the similarity between counterfactual label and desired "opposite" outcome)

CLEAR

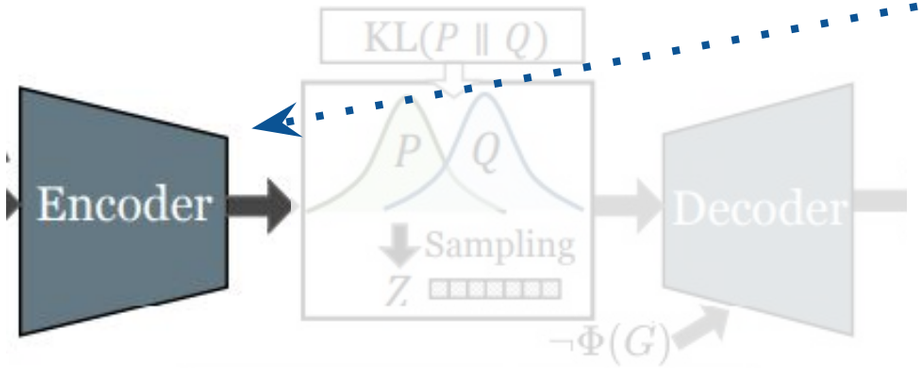
$$\mathcal{L} = \mathbb{E}_Q \left[d(G, G') + \alpha \ell\big(\Phi(G'), \neg\Phi(G)\big) \right] + \mathbf{KL}\Big(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \Big)$$

Q is learned by the encoder



CLEAR

$$\mathcal{L} = \mathbb{E}_Q \left[d(G, G') + \alpha \ell \left(\Phi(G'), \neg\Phi(G) \right) \right] + \mathbf{KL} \left(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \right)$$

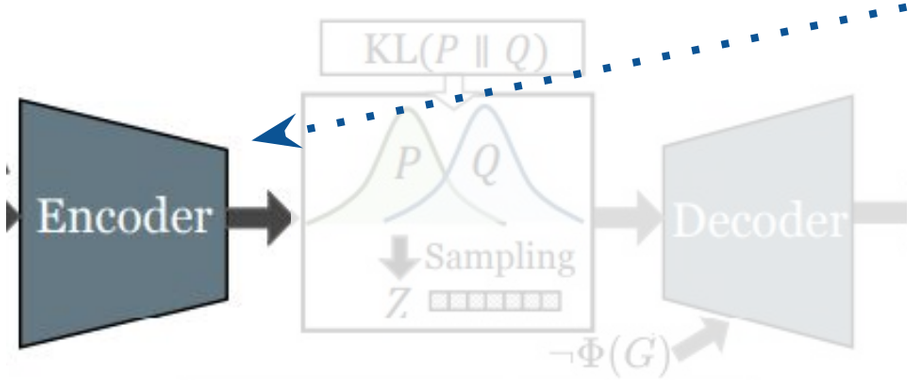


$$P(Z \mid G, \neg\Phi(G)) = \mathcal{N} \left(\mu_z \left(\neg\Phi(G) \right), \mathrm{diag}(\sigma_z^2 \left(\neg\Phi(G) \right)) \right)$$

The Gaussian distribution as prior to enforce the learned distribution Q to be close to the prior by minimizing their KL divergence

CLEAR

$$\mathcal{L} = \mathbb{E}_Q \left[d(G, G') + \alpha \ell\big(\Phi(G'), \neg\Phi(G)\big) \right] + \mathbf{KL}\Big(Q(Z \mid G, \neg\Phi(G)) \,\|\, P(Z \mid G, \neg\Phi(G)) \Big)$$

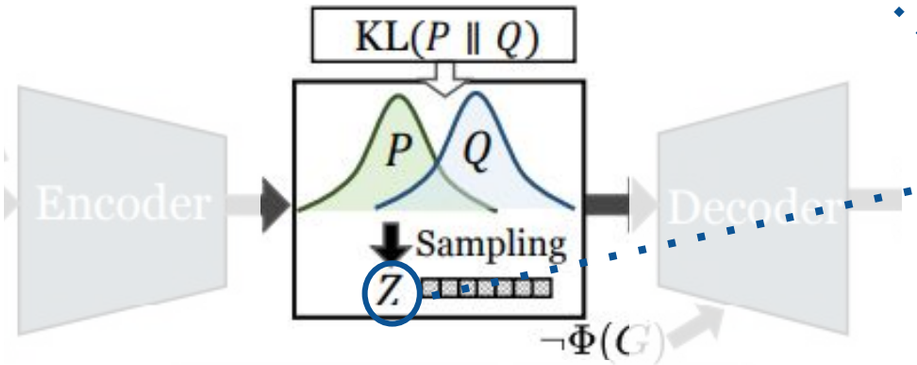


$$P(Z \mid G, \neg\Phi(G)) = \mathcal{N}\Big(\mu_z\big(\neg\Phi(G)\big), \mathrm{diag}\big(\sigma_z^2\big(\neg\Phi(G)\big)\big) \Big)$$

Mean and diagonal covariance of the prior distribution **learned by a NN**

CLEAR

$$\mathcal{L} = \mathbb{E}_Q \left[d(G, G') + \alpha \ell\big(\Phi(G'), \neg\Phi(G)\big) \right] + \mathbf{KL}\Big(Q(Z \mid G, \neg\Phi(G)) \,||\, P(Z \mid G, \neg\Phi(G)) \Big)$$

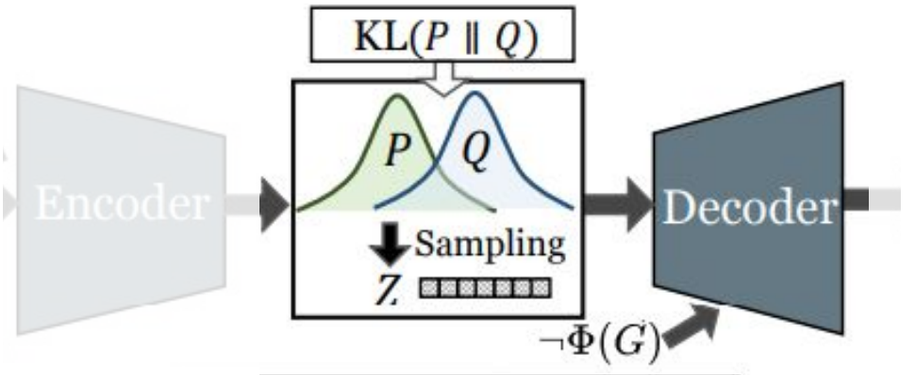


Z is sampled via the reparameterization trick

Kingma DP, Welling M. *Auto-encoding variational bayes*. arXiv preprint arXiv:1312.6114. 2013 Dec 20.

CLEAR

- One can generate **multiple counterfactuals** from sampling multiple Z
- The decoder produces a probabilistic graph where edges have weights $\mathbb{R}_0^1$
- Binarize the graph according to the Bernoulli distribution



CLEAR

$$d(G, G') = d_A(A, \text{Bernoulli}(A')) + d_X(X, X')$$

distance between the original adjacency matrix and the generated one

distance between the original node features and the generated ones

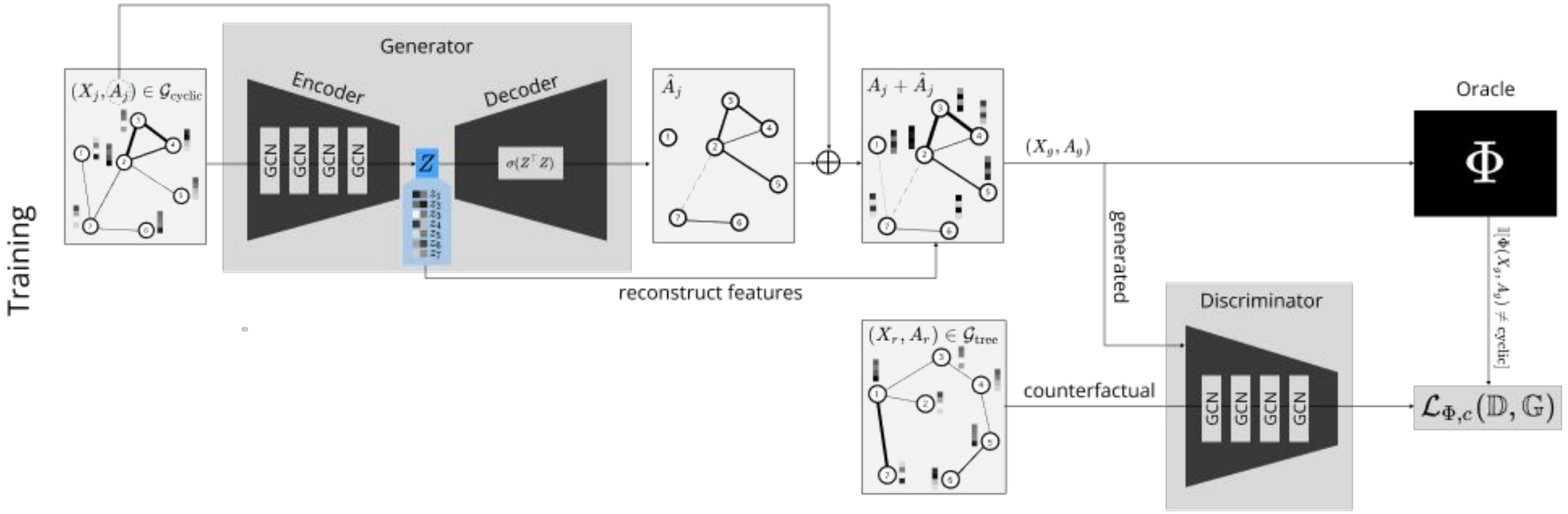
RSGG-CE

Robust **S**tochastic **G**raph **G**enerator for **C**ounterfactual **E**xplanations

RSGG-CE

- Cornerstone paper in the debate "*Are generative counterfactual explanation approaches worth it?*"
- **Besides CLEAR, all other explainers are discriminative**
- Uses Residual GANs to learn how to generate counterfactuals

RSGG-CE (TRAINING)



RSGG-CE (TRAINING)

Note: We're considering only the adj. matrix. All components of the loss in the original formulation contain node features X

Use a residual GAN to generate residuals instead of complete synthetic data

$$\mathcal{L}(\mathbb{G}, \mathbb{D}) = \mathop{\mathbb{E}}_{A \sim p_{data}} \Big[\log \mathbb{D}(A) \Big] + \mathop{\mathbb{E}}_{A_z \sim p_z} \Big[\log \big(1 - \mathbb{D}(A_z + \mathbb{G}(A_z)) \big) \Big]$$

The generator $\mathbb{G}$ seeks to maximize the loss and the discriminator $\mathbb{D}$ seeks to minimize it

RSGG-CE (TRAINING)

$$\mathcal{L}(\mathbb{G},\mathbb{D}) = \mathop{\mathbb{E}}_{A\sim p_{data}}\Big[\log \mathbb{D}(A)\Big] + \mathop{\mathbb{E}}_{A_z\sim p_z}\Big[\log\big(1-\mathbb{D}\big(A_z+\mathbb{G}(A_z)\big)\big)\Big]$$

the input of the generator is a latent adjacency matrix sampled from a probability distribution

residual connections on the generator

RSGG-CE (TRAINING)

$$\mathcal{L}(\mathbb{G},\mathbb{D}) = \mathop{\mathbb{E}}_{A\sim p_{data}}\Big[\log \mathbb{D}(A)\Big] + \mathop{\mathbb{E}}_{A_z\sim p_z}\Big[\log\big(1-\mathbb{D}(A_z+\mathbb{G}(A_z))\big)\Big]$$

This restricts the latent space of the generator to be the same as the input space

RSGG-CE (TRAINING)

- Obviously, the previous loss function doesn't work for generating counterfactuals. It works to generate factual graphs

- We need to integrate the decision of the oracle to guide the generation of plausible counterfactuals

RSGG-CE (TRAINING)

We need to integrate the decision of the oracle to guide the generation of plausible counterfactuals

$$\mathbb{I}[\Phi(A) \neq c] = \begin{cases} 1 & \text{if } \Phi(A) \neq c \\ 0 & \text{otherwise} \end{cases}$$

We'll use this indicator function in the discriminator to induce the generator to correctly generate counterfactuals

RSGG-CE (TRAINING)

what are the instances belonging to class c?

$$\mathcal{A}_c = \{A \mid A \in \mathcal{A} \ \wedge \neg \mathbb{I}[\Phi(A) \neq c]\}$$

what are the instances **NOT** belonging to class c?

$$\mathcal{A}_{\neg c} = \{A \mid A \in \mathcal{A} \ \wedge \mathbb{I}[\Phi(A) \neq c]\}$$

RSGG-CE (TRAINING)

$$\mathcal{L}(\mathbb{G},\mathbb{D}) = \mathop{\mathbb{E}}_{A\in\mathcal{A}}\Big[\log \mathbb{D}(A)\Big] + \mathop{\mathbb{E}}_{A_z\in\mathcal{A}}\Big[\log(1-\mathbb{D}\big(A_z+\mathbb{G}(A_z)\big))\Big]$$

$$\mathcal{L}_{\Phi,c}(\mathbb{G},\mathbb{D}) = \overbrace{\sum_{A\in\mathcal{A}_{\neg c}} \log \mathbb{D}(A)}^{\mathbb{D}\text{'s optimization on real data}} + \overbrace{\sum_{A\in\mathcal{A}_c, A_g = A+\mathbb{G}(A)} \mathbb{I}\big[\Phi(A_g)\neq c\big]\log\mathbb{D}(A_g)}^{\mathbb{D}\text{'s optimization on generated data}} + \underbrace{\sum_{A\in\mathcal{A}_c}\log\big(1-\mathbb{D}(A+\mathbb{G}(A))\big)}_{\mathbb{G}\text{'s optimization}}$$

RSGG-CE (TRAINING)

$$\mathcal{L}(\mathbb{G},\mathbb{D}) = \mathop{\mathbb{E}}_{A\in\mathcal{A}}\Big[\log \mathbb{D}(A)\Big] + \mathop{\mathbb{E}}_{A_z\in\mathcal{A}}\Big[\log(1-\mathbb{D}(A_z+\mathbb{G}(A_z)))\Big]$$

$$\mathcal{L}_{\Phi,c}(\mathbb{G},\mathbb{D}) = \overbrace{\sum_{A\in\mathcal{A}_{\neg c}} \log \mathbb{D}(A)}^{\mathbb{D}\text{'s optimization on real data}} + \overbrace{\sum_{A\in\mathcal{A}_c, A_g = A+\mathbb{G}(A)} \mathbb{I}\big[\Phi(A_g)\neq c\big]\log \mathbb{D}(A_g)}^{\mathbb{D}\text{'s optimization on generated data}}$$

$$\underbrace{+\sum_{A\in\mathcal{A}_c} \log\big(1-\mathbb{D}(A+\mathbb{G}(A))\big)}_{\mathbb{G}\text{'s optimization}}$$

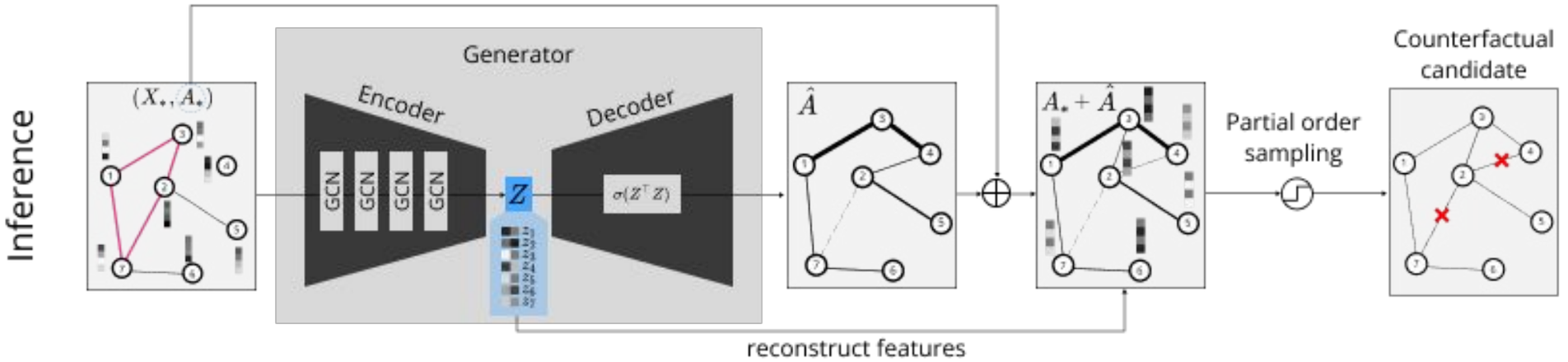
RSGG-CE (TRAINING)

$$\mathcal{L}(\mathbb{G},\mathbb{D}) = \mathop{\mathbb{E}}_{A\in\mathcal{A}}\Big[\log \mathbb{D}(A)\Big] + \mathop{\mathbb{E}}_{A_z\in\mathcal{A}}\Big[\log(1-\mathbb{D}\big(A_z+\mathbb{G}(A_z)\big))\Big]$$

$$\mathcal{L}_{\Phi,c}(\mathbb{G},\mathbb{D}) = \overbrace{\sum_{A\in\mathcal{A}_{\neg c}} \log \mathbb{D}(A)}^{\mathbb{D}\text{'s optimization on real data}} + \overbrace{\sum_{A\in\mathcal{A}_c, A_g = A+\mathbb{G}(A)} \mathbb{I}\big[\Phi(A_g)\neq c\big]\log \mathbb{D}(A_g)}^{\mathbb{D}\text{'s optimization on generated data}}$$

$$+ \underbrace{\sum_{A\in\mathcal{A}_c} \log\big(1-\mathbb{D}(A+\mathbb{G}(A))\big)}_{\mathbb{G}\text{'s optimization}}$$

RSGG-CE (INFERENCE)

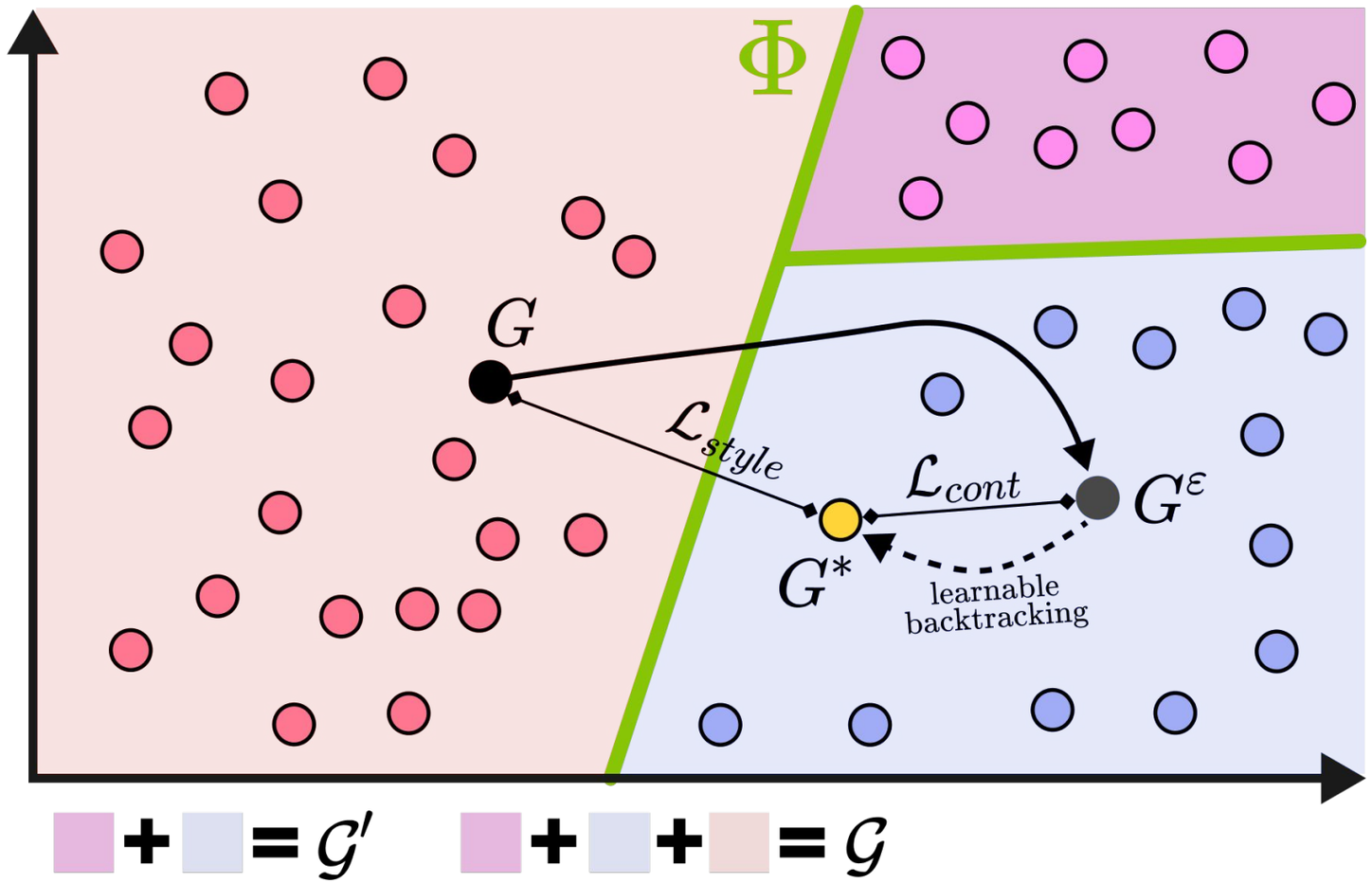


RSGG-CE (RECAP)

- We train the generator on graphs of class c (**the class we want to explain**)
- We train the discriminator on graphs different from class c and the synthetic data generated
- Because the generator needs to fool the discriminator, it'll learn to produce graphs of class **not c**

GIST
Graph **I**nverse **S**tyle **T**ransfer for Counterfactual Explainability

GIST


Φ
G
$\mathcal{L}_{style}$
$\mathcal{L}_{cont}$
G^{ε}
G^{*}
learnable
backtracking
+ = $\mathcal{G}'$
+ + = $\mathcal{G}$

GIST (GRAPH STYLE AND CONTENT)

$$G = (X, A) \begin{cases} X \in \mathbb{R}^{n \times d} \\ A \in \mathbb{R}^{n \times n} \end{cases}$$

$$L^{(G)} = D - A$$

$$\lambda_1(L^{(G)}) \leq \lambda_2(L^{(G)}) \leq \cdots \leq \lambda_n(L^{(G)})$$

GIST (GRAPH STYLE AND CONTENT)

$$L^{(G)} = D - A$$

$$\lambda_1(L^{(G)}) \leq \lambda_2(L^{(G)}) \leq \cdots \leq \lambda_n(L^{(G)})$$

These guys

capture **global structural patterns**, e.g., connectivity and symmetry, that are largely **invariant** to specific node identities

GIST (STYLE TRANSFER IN GRAPHS)

Definition 4.1. Given a graph $G = (X, A)$, s.t. $n = |X|$, the goal is to generate $G^* = (X^*, A^*)$ by passing through an intermediary known graph $G^\varepsilon = (X^\varepsilon, A^\varepsilon) \in \mathcal{G}^*$ with $\Phi(G) \neq \Phi(G^\varepsilon)$, such that the following conditions are met.

(1) *Style transfer* – the global structural properties of G^* align with G, i.e.,

$$\sum_{i=1}^{n} \left|\lambda_i(\tilde{L}^{(G)}) - \lambda_i(\tilde{L}^{(G^*)})\right| \leq \sum_{i=1}^{n} \left|\lambda_i(\tilde{L}^{(G)}) - \lambda_i(\tilde{L}^{(G^\varepsilon)})\right|, \tag{5}$$

(2) *Content preservation* – the local structure of G^* resembles G^ε, i.e.,

$$\min_{G^* \in \mathcal{G}'} \underbrace{\left|\left|X^* - X^\varepsilon\right|\right|_1}_{\text{reconstruct node feature}} + \underbrace{\text{BCE}(A^*, A^\varepsilon)}_{\substack{\text{reconstruct existing} \\ \text{and non-existing edges}}} . \tag{6}$$

GIST (THEORETICAL IMPLICATIONS)

Definition 4.2. Given $\tilde{L}^{(G)}$ and $\tilde{L}^{(G^\varepsilon)}$, two real, symmetric and commuting matrices, the normalized Laplacian of G^* is defined as the convex combination in Equation (7).

$$\tilde{L}^{(G^*)} = \alpha \cdot \tilde{L}^{(G^\varepsilon)} + (1-\alpha) \cdot \tilde{L}^{(G)}, \tag{7}$$

where $\alpha \in [0,1]$ is the interpolation factor that controls the trade-off between content and style.

Lemma 4.3. *Given $\tilde{L}^{(G)}$ and $\tilde{L}^{(G^\varepsilon)}$, two real, symmetric and commuting matrices, the eigenvalues of G^* are defined as the convex combination*

$$\lambda_i(\tilde{L}^{(G^*)}) = \alpha \cdot \lambda_i(\tilde{L}^{(G^\varepsilon)}) + (1-\alpha) \cdot \lambda_i(\tilde{L}^{(G)}). \tag{8}$$

Theorem 4.4. *If G and G^ε are connected graphs, then G^* whose normalized Laplacian is defined as in Equation (7) is connected for any $\alpha \in [0,1]$.*

Theorem 4.5. *Let $\Delta(G) = \lambda_2(L^{(G)}) - \lambda_1(L^{(G)})$ denote the spectral gap of G, where $\lambda_2(L^{(G)})$ and $\lambda_1(L^{(G)})$ are the second- and first-lowest eigenvalues of G. Then:*

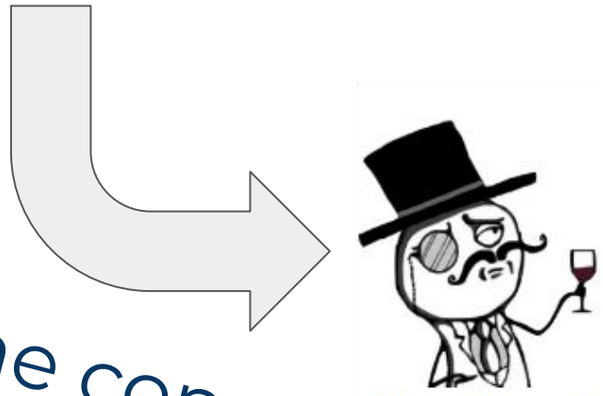
$$\min(\Delta(G), \Delta(G^\varepsilon)) \leq \Delta(G^*) \leq \max(\Delta(G), \Delta(G^\varepsilon)). \tag{9}$$

Corollary 4.6. *The Frobenius norm difference between G^ε and G^* is*

$$\left\| L^{(G^\varepsilon)} - L^{(G^*)} \right\|_F = (1-\alpha) \left\| L^{(G^\varepsilon)} - L^{(G)} \right\|_F. \tag{10}$$

GIST (FORWARD PASS...)

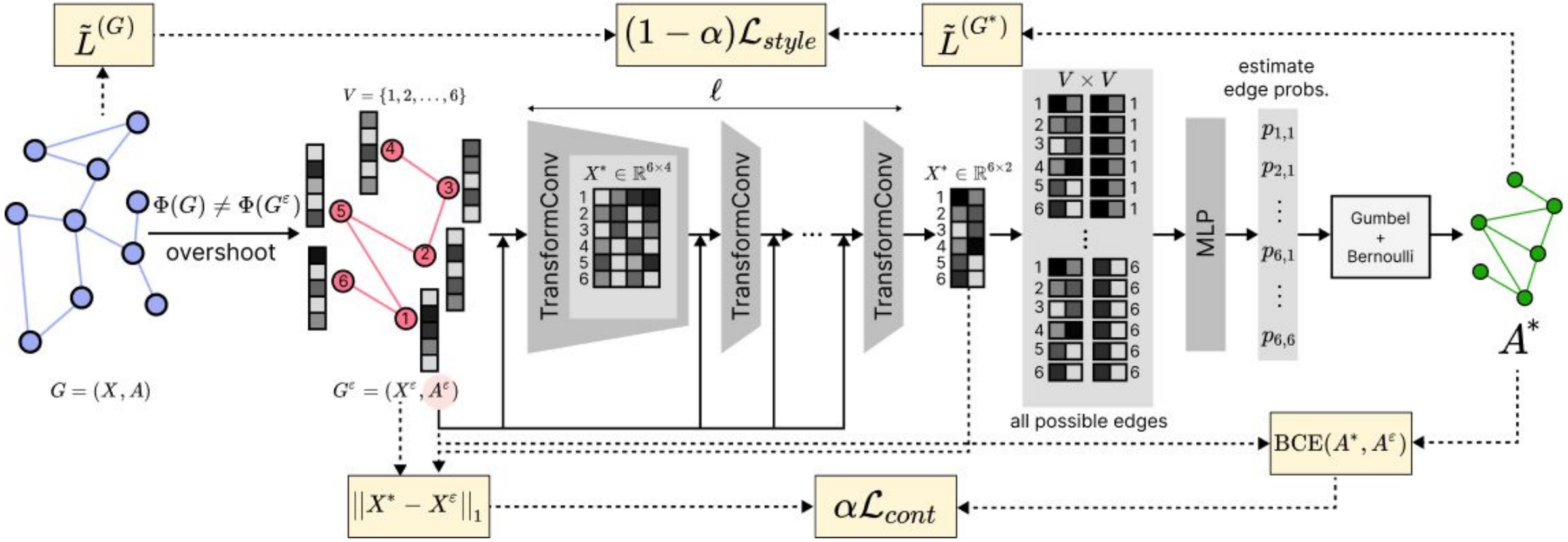
... just take a random graph of another class



for the connoisseurs

$$k^* = \min\{k \in [1, n] \mid \Phi(G) \neq \Phi(G_k)\}$$
$$\forall G_k \in U(\{G \mid G \in \mathcal{G}\})$$
$$G^\varepsilon := G_{k^*}$$

GIST (BACKWARD PASS...)



GIST (RECAP)

- TL;DR: Overshoot the decision boundary, then go back

- The first formalization and application of Graph Style Transfer

- Graph Counterfactual Explainability is still avoiding causality
  (and when not, it implements random variables!!!)

- Three core innovations:
  - style-transfer that **preserves structural coherence** while refining local node features and edge connections
  - theoretical guarantees on **spectral alignment** and **connectivity preservations**
  - modular architecture compatible with **diverse GNN** backbones

Global Counterfactual Explainers

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2023. Global Counterfactual Explainer for Graph Neural Networks. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23). Association for Computing Machinery, New York, NY, USA, 141–149. https://doi.org/10.1145/3539597.3570376

GCFExplainer

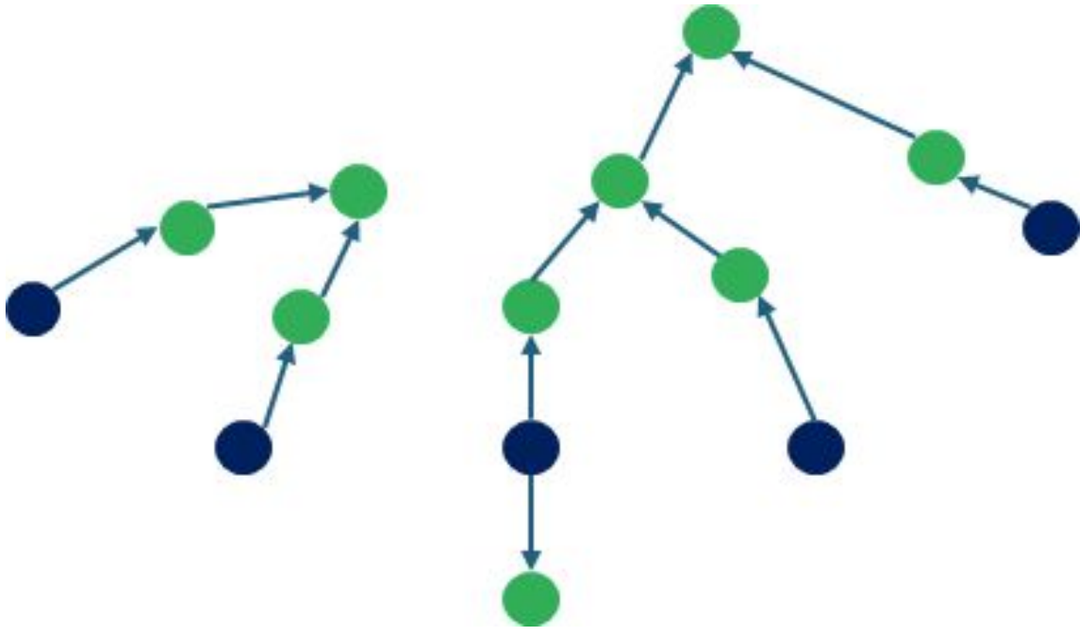
- The **input** is a set of instances to explain $\mathbb{G}$
- The **output** is a set of counterfactual instances $\mathbb{C}$ that are considered an explanation of the input set
- **Coverage:** the proportion of graphs in $\mathbb{G}$ that have a close counterfactual in $\mathbb{C}$ under a given distance threshold θ:

$$coverage(\mathbb{C}) = |\{G \in \mathbb{G} \mid \min_{\{C \in \mathbb{C}\}} \{d(G, C)\} \leq \theta\}|/|G|$$

GCFExplainer (Search Space)

- The search space of counterfactual graphs consists of all graphs that are in the same domain as the input graphs and within a distance of θ.
- The number of potential graphs within θ increases exponentially with θ since the space of graph edits is combinatorial.
- GCFExplainer uses an edit map to organize these graphs as a meta-graph $\mathfrak{G}$, where individual nodes are graphs that are created via a different number of edits from the input graphs and each edge represents a single edit.

Search Space

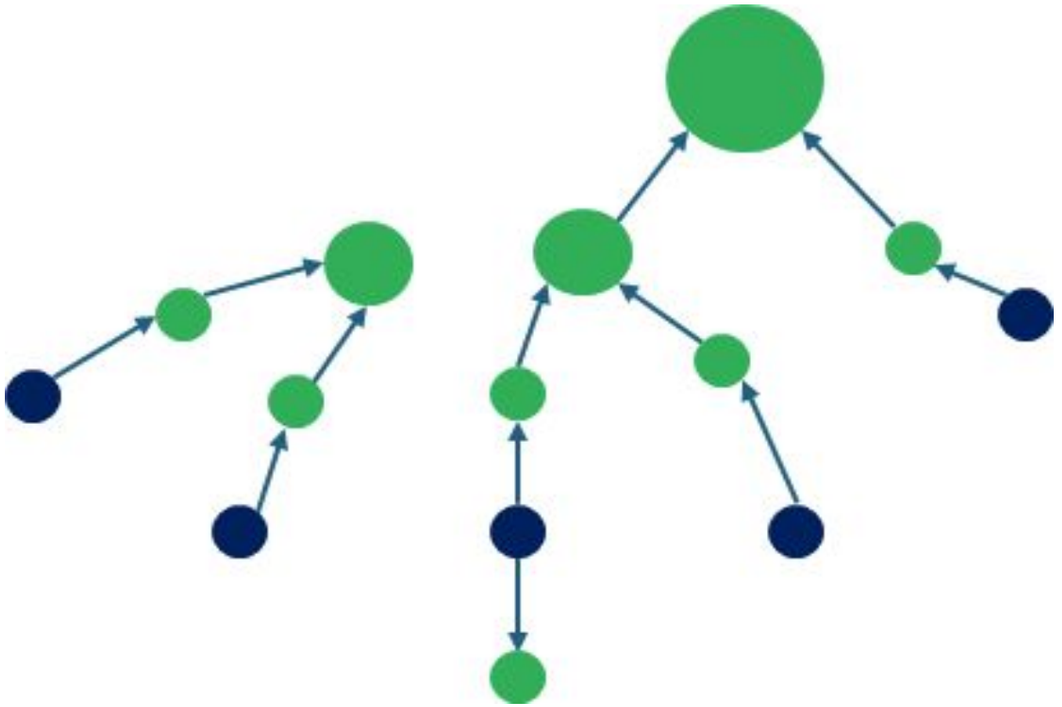


Vertex-Reinforced Random Walk

- To search for good counterfactual candidates, GCFExplainer leverages vertex-reinforced random walks (VRRW) on the edit map $\mathfrak{G}$

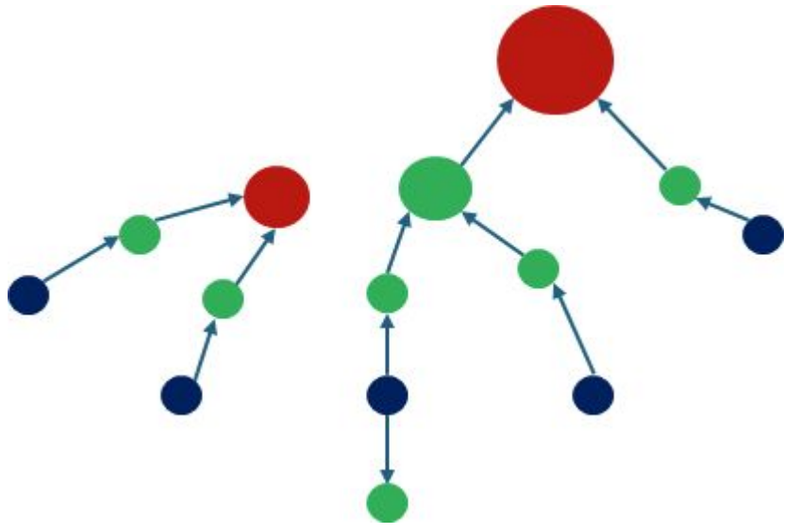
- VRRW has the nice property of converging to a set of nodes that are both important (i.e., cover many input graphs) and diverse (i.e., non-overlapping coverage), which will form a small set of counterfactual candidates for further processing

Vertex-Reinforced Random Walk



Iterative Summary Computation

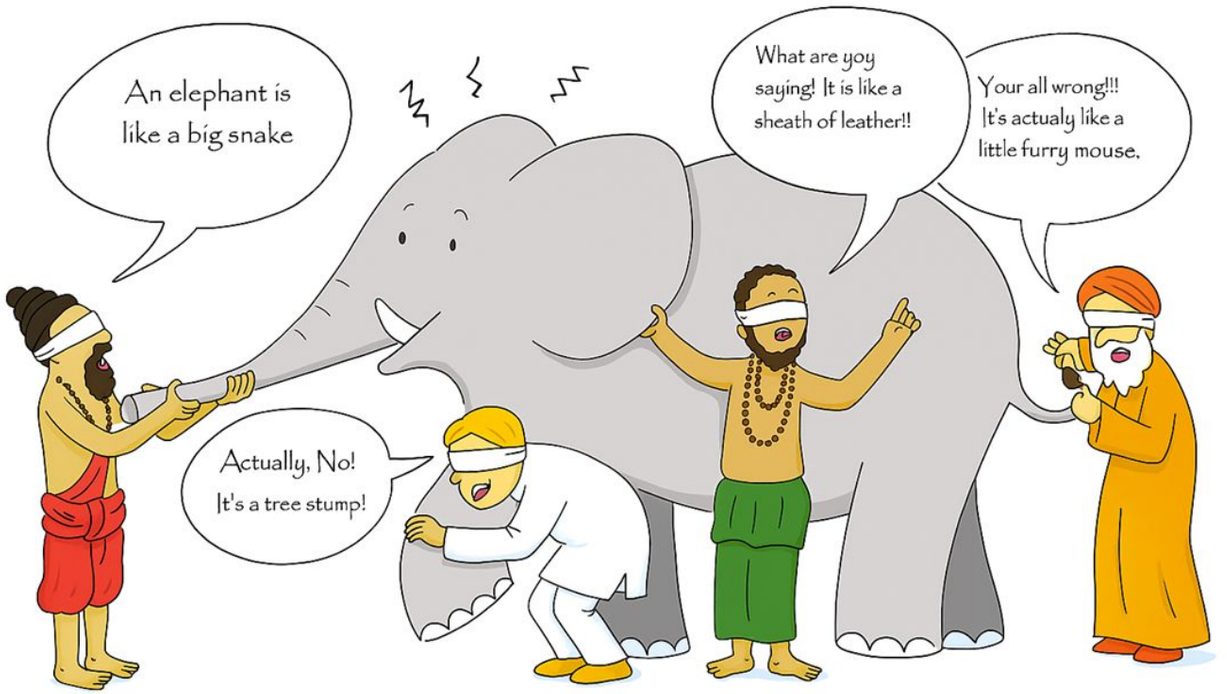
After obtaining good counterfactual candidates from VRRW, GCFExplainer creates the final set of the counterfactual graphs (i.e., the summary) as the recourse representation by iteratively adding the best candidate based on the maximal gain of the coverage given the already added candidates



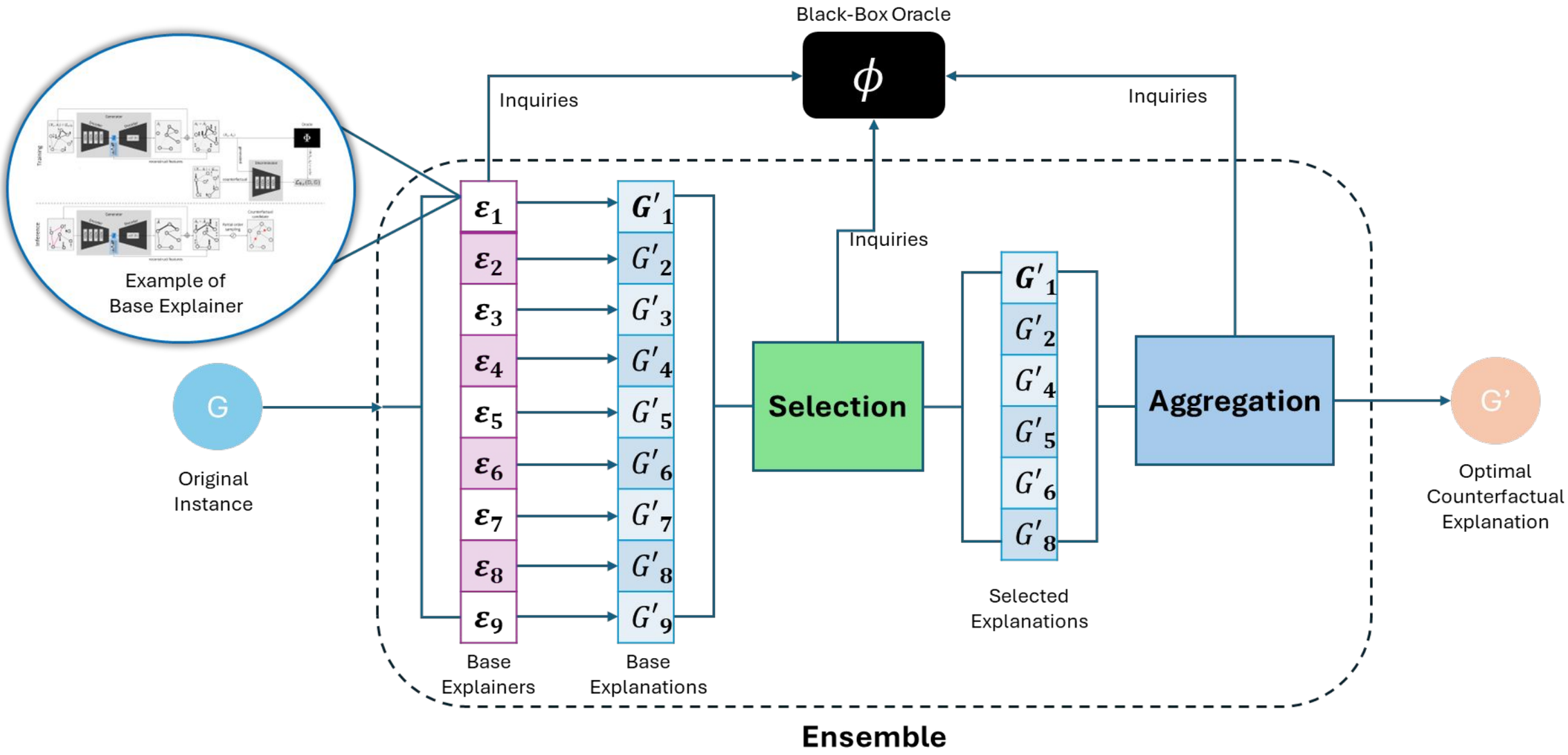
Meta-Explainers

Combining Explainers

Can we combine the strengths of different explainers into an ensemble that performs well across different domains?



Ensemble Strategy: Selection and Aggregation



Ideal-Point Multi-Criteria Selection (IPMCS)

Given an explainee instance G, an oracle Φ, and a set of base explainers $\mathbb{E} = \{\mathcal{E}_\Phi^1, \mathcal{E}_\Phi^2, \dots, \mathcal{E}_\Phi^K\}$ then IPMCS performs the following steps:

i. Get the set of explanations $\mathbb{C} = \bigcup_{i=1}^{K} \mathcal{E}_\Phi^i(\mathrm{G})$

ii. Given a set of criteria $H = \{h_1, h_2, \dots, h_k\}$ calculate an ideal point $Z = [z_1, z_2, \dots, z_k]$ where $z_i = \max\{h_i(G') \mid G' \in \mathbb{C}\}$

iii. Select the explanations $\mathrm{G}'_\mathrm{i} = \mathrm{argmax}_{\mathrm{G}'_\mathrm{i} \in \mathbb{C}} \mathrm{d}(\mathrm{Z}, \mathrm{H}(\mathrm{G}'_\mathrm{i}))$

Learning to Select Explainers

1) Dataset-Level Explainer Selection (DLES):

i. On training phase, uses the metrics set H and the ideal-point method on each training instance to determine the base explainer $\mathcal{E}_{\Phi}^{*}$ that performs the best on average.

ii. On inference time uses $\mathcal{E}_{\Phi}^{*}$ to explain every instance

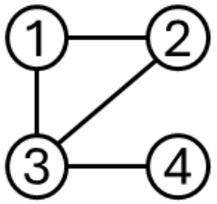
2) Instance-Level Explainer Selector (ILES):

i. On training phase, uses the explainer selected by IPMCS for each instance as the instance label creating a new training dataset for explainer selection, then trains a GNN on it.

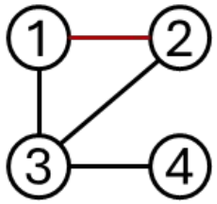
ii. On inference time, uses the GNN to predict the best explainer for each instance based on the characteristics of that instance.

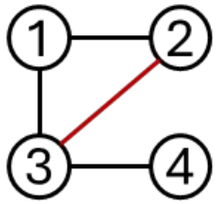
Explanation Aggregation: Change Frequency

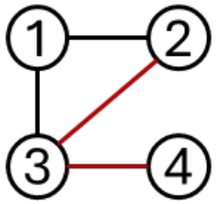
Original Instance



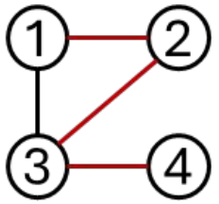
Counterfactual Instances







Aggregated changes



0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	1	0	0
1	0	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	0	1	0
0	1	0	0
0	0	0	0

0	0	0	0
0	0	1	0
0	1	0	1
0	0	1	0

0	**.3**	0	0
.3	0	**.6**	0
0	**.6**	0	**.3**
0	0	**.3**	0

Changes Matrices

Change Frequency Matrix

Explanation Aggregation: Strategies

1) **Frequency Aggregation:** Applies to the original instance all the changes that appear with a frequency higher or equal than a value τ. Union ($\tau \geq 0$) and Intersection ($\tau \geq 1$) are special cases.

2) **Iterative Random Aggregation:** Iteratively selects random changes from the changes frequency matrix and applied them to the original instance until finds a counterfactual or reaches the maximum iteration threshold p.

3) **Stochastic Aggregation:** Consider the frequency of changes as probabilities and apply then, iteratively and in order, to the original instance until a counterfactual is found.

4) **Bidirectional Aggregation:** Behaves similarly to the Iterative Random in a first stage. In a second stage it tries to reduce the size of the solution by randomly undoing some of the changes while preserving the counterfactuality of the solution.

Part IV

Counterfactual Explainability in Evolving Graphs

by Bardh Prenkaj

Based on:

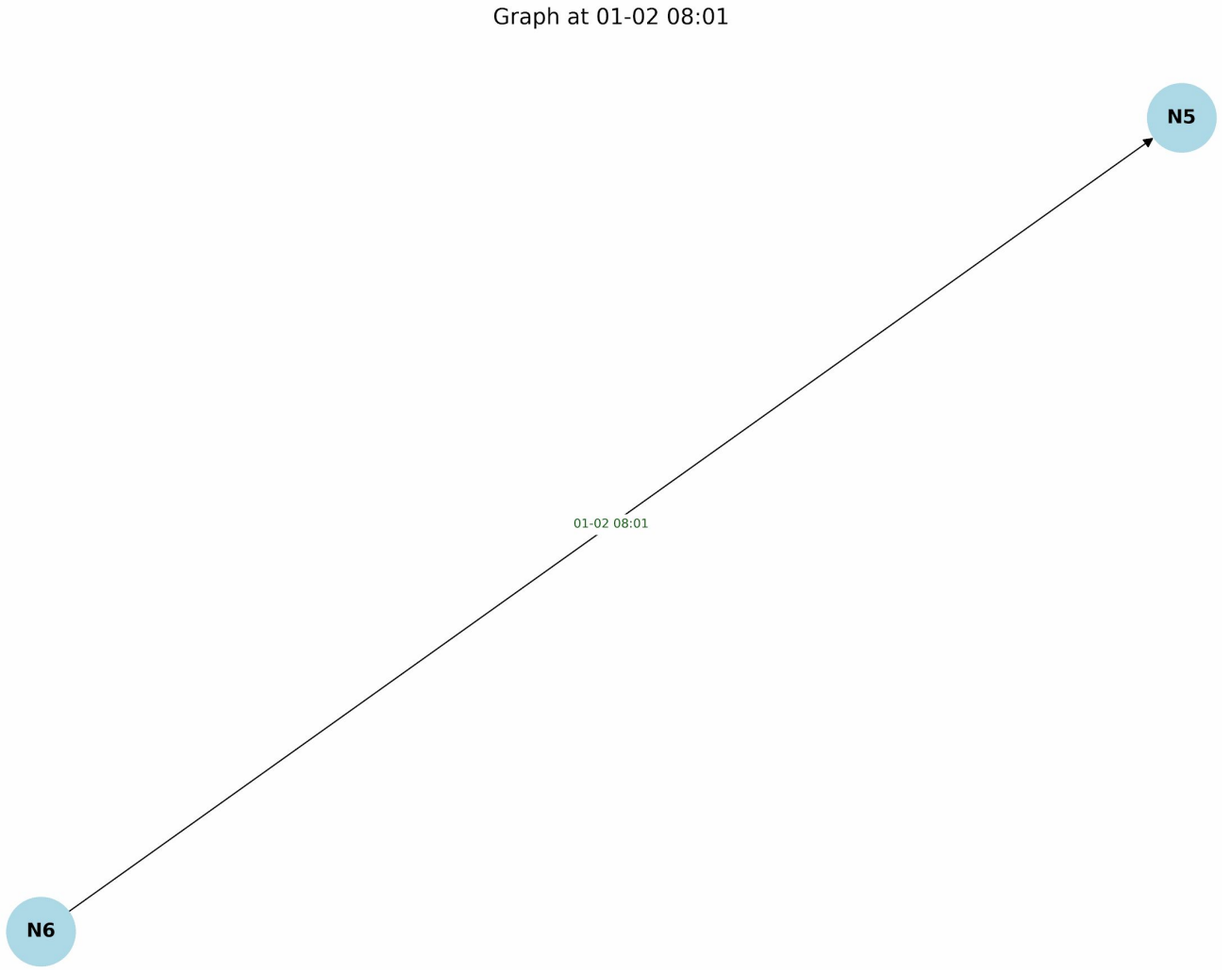
Prenkaj et al. “Unifying Evolution, Explanation, and Discernment: A Generative Approach for Dynamic Graph Counterfactuals”, KDD 2024

Zu et al. “CoDy: Counterfactual Explainers for Dynamic Graphs”, ICML 2025

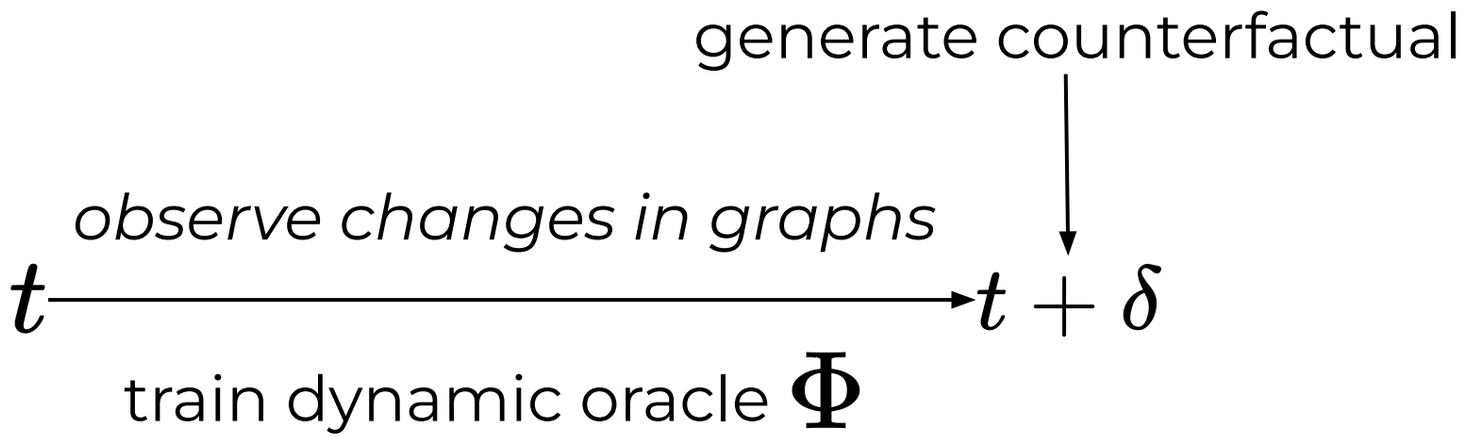
What Makes Dynamic Graphs Different?

- Time determines the generation of the counterfactuals
- Oracles are usually Time-Graph Convolutional Networks (TGCNs) but alternatives exist
- Two different approaches:
  - Continuous-Time Dynamic Graphs (**CTDG**) - *fine-grained res*
  - Discrete-Time Dynamic Graphs (**DTDG**) - *window- or snapshot-based*

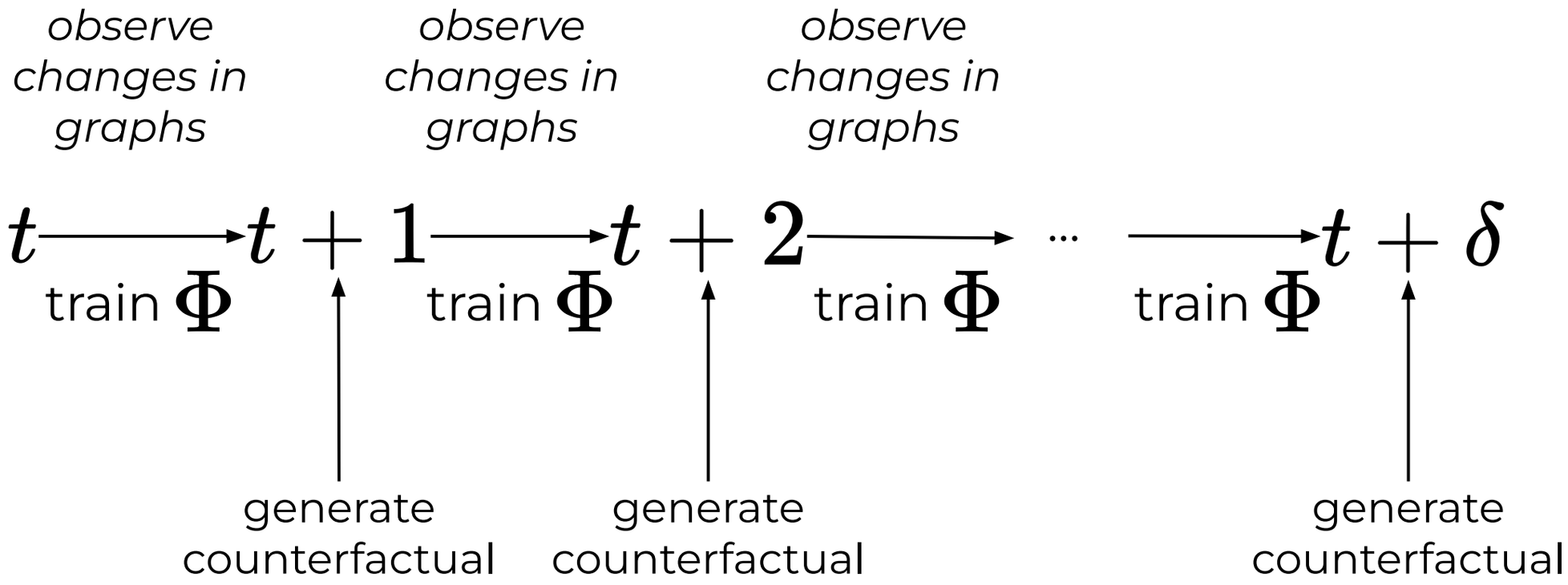
Temporal dependencies and evolving structures



Continuous-Time Dynamic Graphs (CTDG)



Discrete-Time Dynamic Graphs (DTDG)



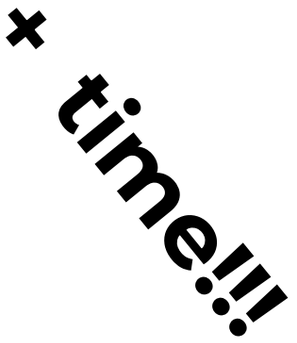
CTDG vs DTDG (recap)

	Discrete-Time Dynamic Graphs (**DTDGs**)	Continuous-Time Dynamic Graphs (**CTDGs**)
Advantages	- **Simplicity**: Easier to conceptualize and work with due to their representation as a sequence of static snapshots - **Established Methods**: A wide range of existing explainability methods are specifically designed for DTDGs.	- **Realism**: More accurately represent real-world systems, such as social networks, where interactions occur continuously and at irregular intervals. - **Granularity**: Capture fine-grained, event-based dynamics, which might be lost when aggregating events into discrete snapshots.
Disadvantages	- **Loss of Granularity**: The process of creating static snapshots can lead to a loss of information about the exact timing and order of events within a time interval. - **Inapplicability to Continuous Data**: Not readily applicable to continuously evolving data without significant adaptation, which can be computationally expensive or lead to inaccurate representations.	- **Limited Explainability Methods**: A scarcity of dedicated explainability methods, with only a few notable approaches like T-GNNExplainer [1] and TempME [2] having been proposed - **Complexity**: More complex to model and analyze due to the continuous and asynchronous nature of events, requiring specialized algorithms and data structures.

[1] Xia et al. Explaining temporal graph models through an explorer-navigator framework. In ICLR'23
[2] Chen et al. TempME: Towards the explainability of temporal graph neural networks via motif discovery. In NeurIPS'23.

CoDY: Counterfactual Explainers for Dynamic Graphs

- **Challenge #1**: Understanding predictions from Temporal Graph Neural Networks (TGNNs) is difficult. Existing methods focus on factual explanations, but not on counterfactuals.



- **Challenge #2**: Unlike factual explanations, counterfactuals require exploring a vast, combinatorially explosive space of possible graph modifications to find a minimal set of changes that would flip a model's prediction.

CoDY = 1st counterfactual for CTDGs (edge prediction)

CoDy: Problem Definition

$$\mathcal{E}(\Phi, G_{t_i}, e_i) = \bigcup_{k=0}^{|G_{t_i} \setminus e_i|} \text{binom}(G_{t_i} \setminus e_i, k)$$

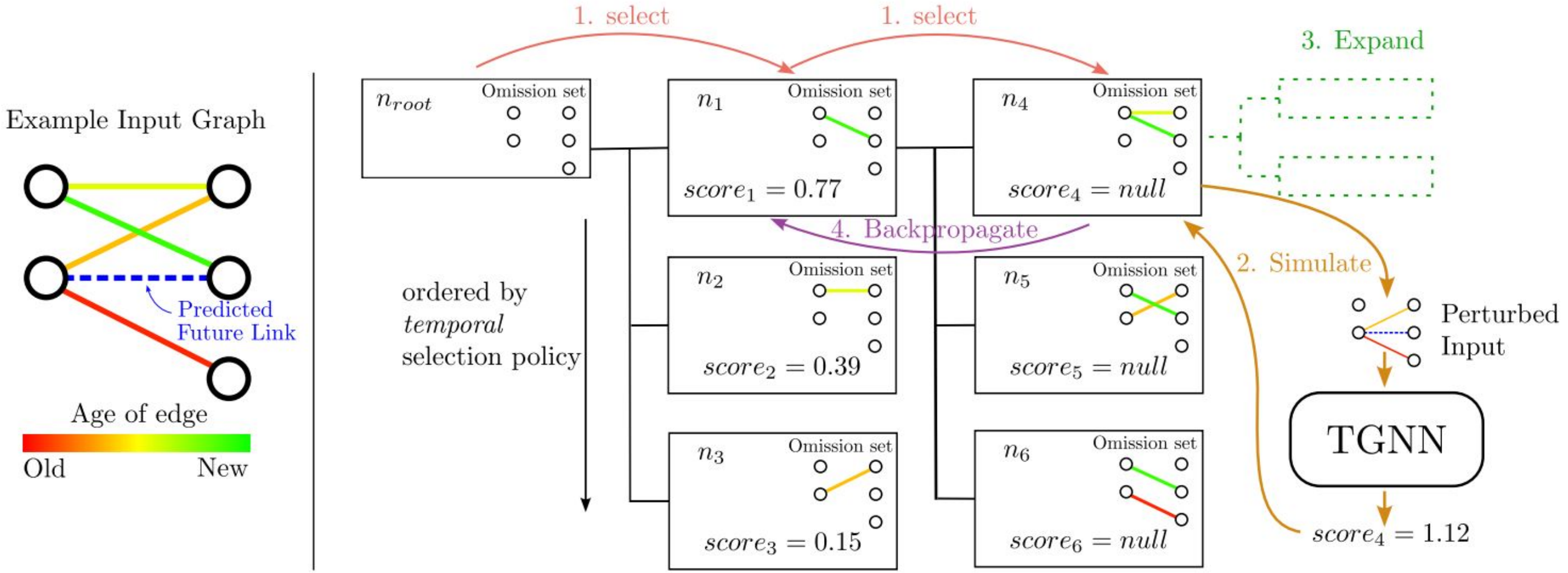
A counterfactual explainer provides a rationale for the model's prediction. For future link predictions, $\mathcal{E}$ is a function that takes as input the TGNN oracle, the temporal graph, and the future link, and outputs a subset of the original graph's events as counterfactual explanation, *i.e.* *any combination of past events*

CoDy

- Uses a Monte Carlo Tree Search (MCTS)-based algorithm to explore the vast space of possible graph modifications.

- The search is guided by heuristic selection policies that incorporate three key types of information:
  - **Temporal**: Importance of events based on their timing.
  - **Spatial**: Importance of events based on their local graph structure.
  - **Local Gradient**: Information about the model's sensitivity to small changes.

CoDy



The left side shows an example of a temporal input graph, where the explained future link is highlighted in blue, and the other links are color-coded based on their ages. On the right, one iteration of the CoDy algorithm is illustrated. Each rectangle represents a node in the search tree, corresponding to a specific perturbation of the input graph.

CoDy: Search Algorithm

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}

Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$

$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$

$it \leftarrow 0$

while $it < it_{max}$ *and* n_{root} *is selectable* **do**

  $n_{selected} \leftarrow$ **select**(n_{root}, δ)

  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$

  expand$(n_{selected}, p_{orig})$

  backpropagate$(parent_{selected})$

  $it \leftarrow it + 1$

end

$n_{best} \leftarrow$ **select_best**(n_{root})

return s_{best}

CoDy: Search Algorithm (select)

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}

Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$

$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$

$it \leftarrow 0$

while $it < it_{max}$ *and* n_{root} *is selectable* **do**

  $n_{selected} \leftarrow$ **select**(n_{root}, δ)

  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$

  expand$(n_{selected}, p_{orig})$

  backpropagate$(parent_{selected})$

  $it \leftarrow it + 1$

end

$n_{best} \leftarrow$ **select_best**(n_{root})

return s_{best}

$$\text{sel\_score}(n_k) = \alpha \text{score}_k + (1 - \alpha)\text{score}_{\text{explore}}(n_k)$$

'Upper Confidence Bound 1'

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, 47(2):235–256, 2002. ISSN 1573-0565

CoDy: Search Algorithm (simulate)

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}
Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$
$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$
$it \leftarrow 0$
while $it < it_{max}$ *and* n_{root} *is selectable* **do**
  $n_{selected} \leftarrow$ **select**(n_{root}, δ)
  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$
  expand$(n_{selected}, p_{orig})$
  backpropagate$(parent_{selected})$
  $it \leftarrow it + 1$
end
$n_{best} \leftarrow$ **select_best**(n_{root})
return s_{best}

Eq 6: $\Delta(p_{orig}, p_j) = |p_{orig} - p_j|$

$$\text{score}_j = \max\left(0, \frac{\Delta(p_{orig}, p_j)}{|p_{orig}|}\right)$$

if $\text{score}_j > 1$, then counterfactual

CoDy: Search Algorithm (Expand)

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}
Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$
$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$
$it \leftarrow 0$
while $it < it_{max}$ *and* n_{root} *is selectable* **do**
  $n_{selected} \leftarrow$ **select**(n_{root}, δ)
  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$
  expand$(n_{selected}, p_{orig})$
  backpropagate$(parent_{selected})$
  $it \leftarrow it + 1$
end
$n_{best} \leftarrow$ **select_best**(n_{root})
return s_{best}

- The expansion adds child nodes to the selected node.
- Following the definition of the search tree, each new child node is initialized with a set of events to omit from the input graph

CoDy: Search Algorithm (Backprop.)

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}
Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$
$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$
$it \leftarrow 0$
while $it < it_{max}$ *and* n_{root} *is selectable* **do**
  $n_{selected} \leftarrow$ **select**(n_{root}, δ)
  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$
  expand$(n_{selected}, p_{orig})$
  backpropagate$(parent_{selected})$
  $it \leftarrow it + 1$
end
$n_{best} \leftarrow$ **select_best**(n_{root})
return s_{best}

Recursively updates the search tree, starting from the parent node of the expanded node and proceeding until the root node

$$\text{score}_j = \frac{\max(0, \frac{\Delta(p_{orig}, p_j)}{|p_{orig}|}) + \sum_{n_k \in C_j} (\text{score}_k \cdot sel_k)}{sel_j}$$

child nodes of n_j

sel_j: total number of selections of node n_j

CoDy: Search Algorithm (Final)

Algorithm 1 Search algorithm of CoDy.

Input: TGNN model Φ, input graph $\mathcal{G}$, explained event ε_i, selection policy δ, max iterations it_{max}

Output: best explanation found

$p_{orig} \leftarrow \Phi(\mathcal{G}(t_i), \varepsilon_i)$

$n_{root} \leftarrow (\varnothing, null, null, \varnothing, 0, null, 1)$

$it \leftarrow 0$

while $it < it_{max}$ *and* n_{root} *is selectable* **do**

  $n_{selected} \leftarrow$ **select**(n_{root}, δ)

  simulate$(n_{selected}, \Phi, \mathcal{G}, \varepsilon_i)$

  expand$(n_{selected}, p_{orig})$

  backpropagate$(parent_{selected})$

  $it \leftarrow it + 1$

end

$n_{best} \leftarrow$ **select_best**(n_{root})

return s_{best}

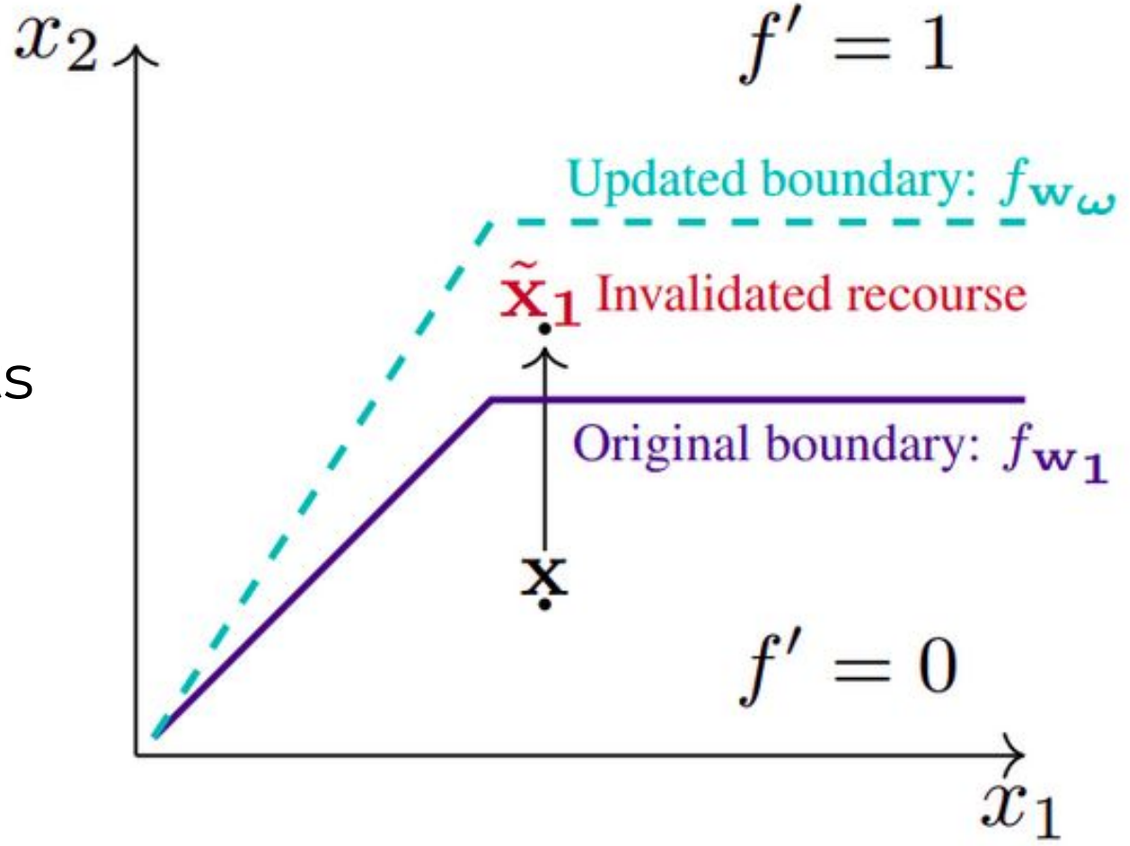
- Select the minimum number of events:
  - If collisons, choose the one with the **largest change** as in **Eq. 6**
  - If no counterfactuals are found, select the perturbation set producing the greatest prediction shift according to **Eq. 6**

CoDy (Recap)

- CoDy is the first method to provide counterfactual explanations for dynamic graphs.
- By leveraging MCTS and heuristic search policies, it efficiently finds minimal changes to alter a TGNN's prediction.
- CoDy highlights the distinct advantage of counterfactual explanations for understanding complex, time-evolving systems.
- **CoDy can’t handle node features**

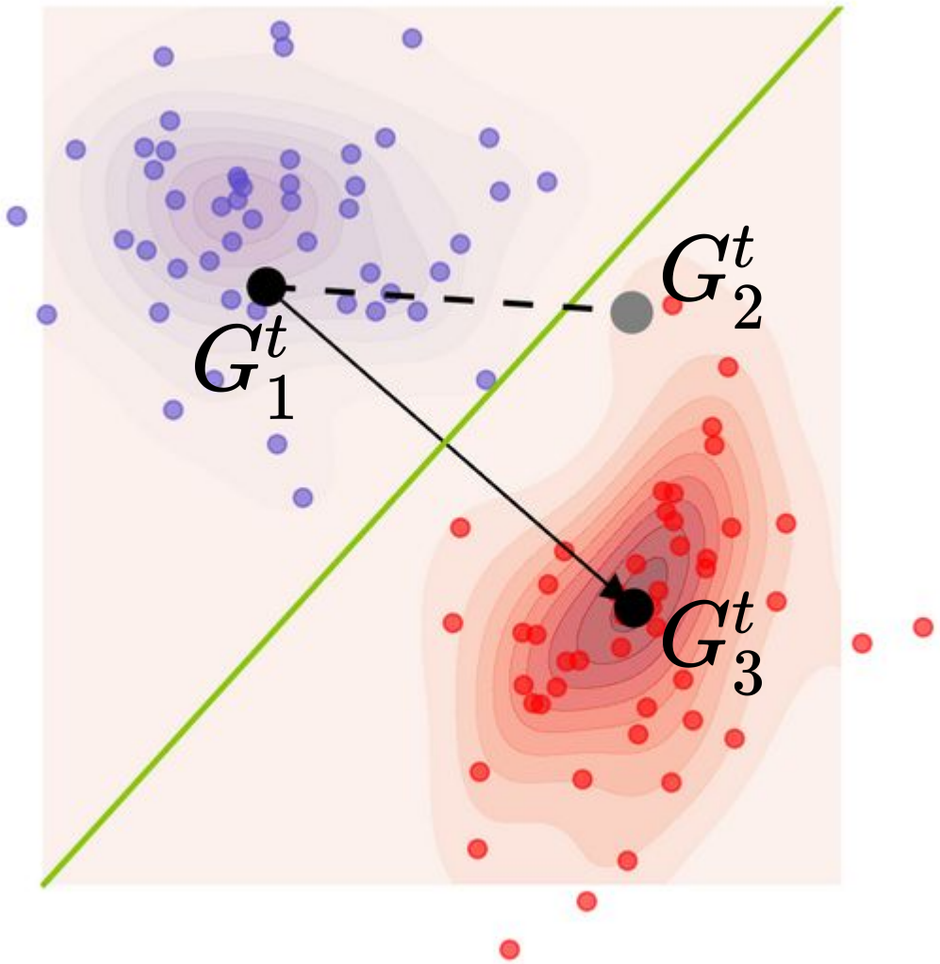
"The Right to be Forgotten"

- Counterfactuals can become invalidated when data is deleted
- Pawleczyk et al. identify data points that, when deleted at **t + δ**, invalidate the counterfactuals at time **t**

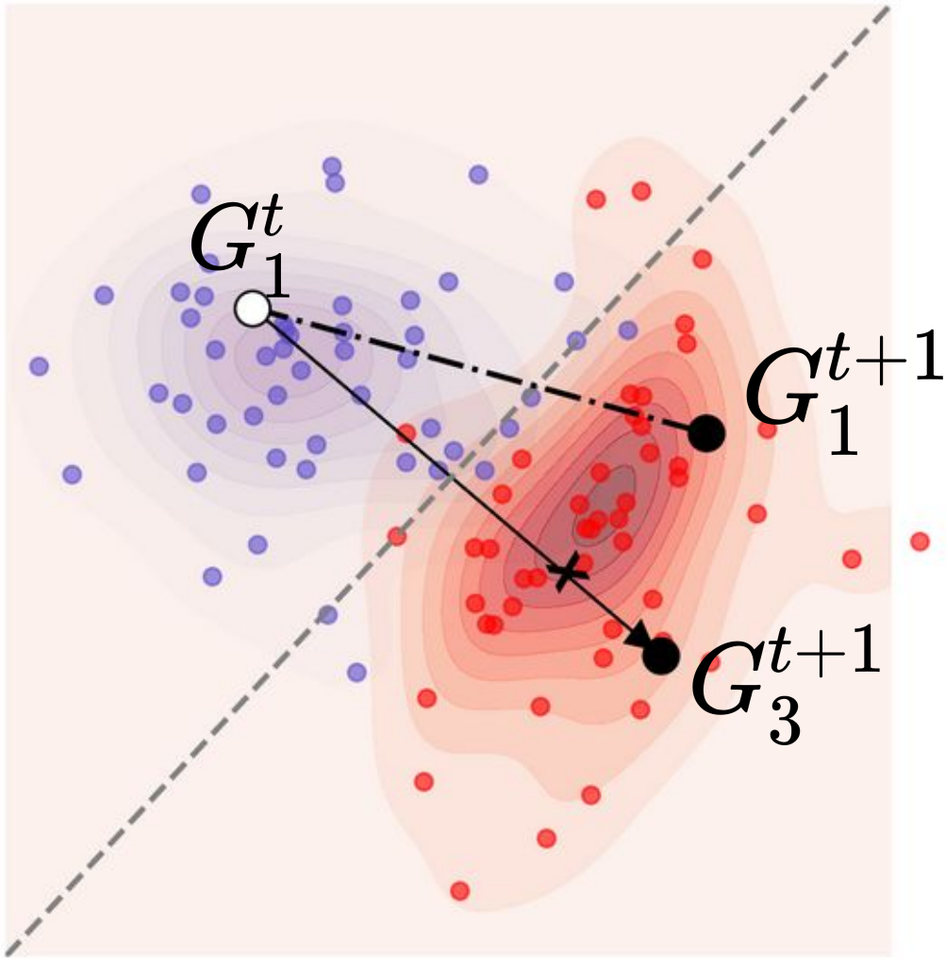


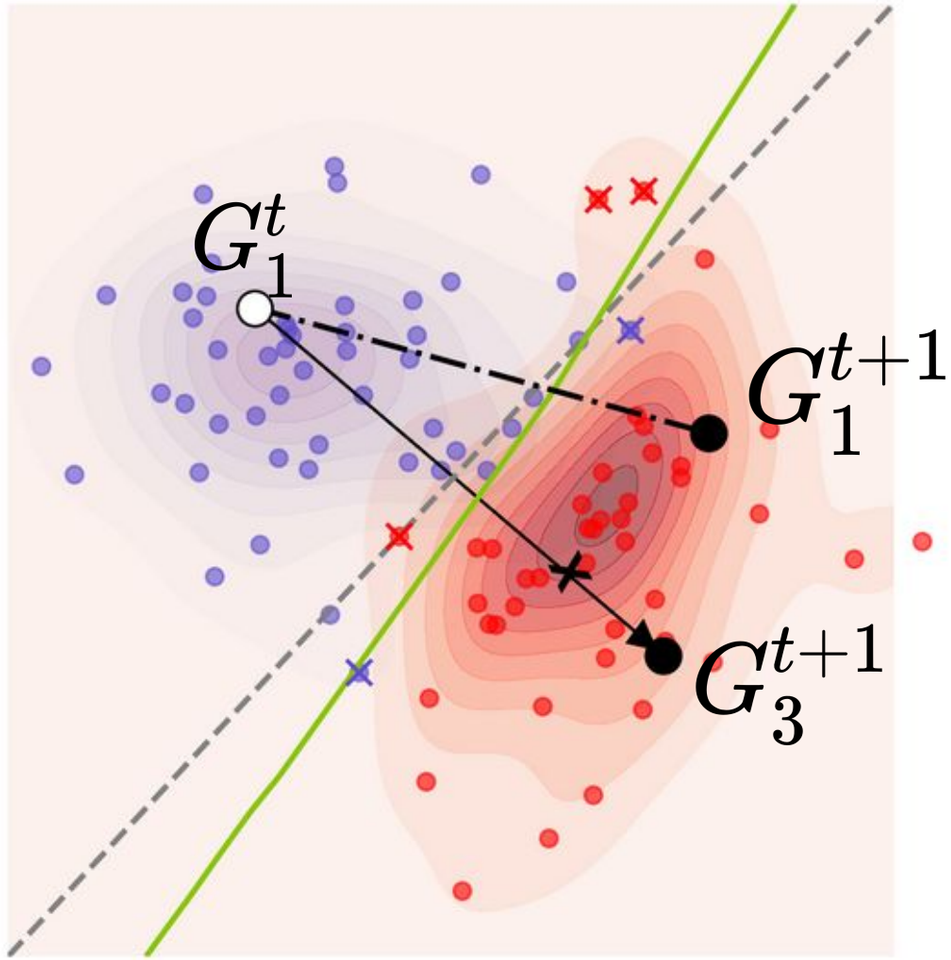
Pawelczyk et al. On the Trade-Off between Actionable Explanations and the Right to be Forgotten. ICLR'23

Distribution Shifts

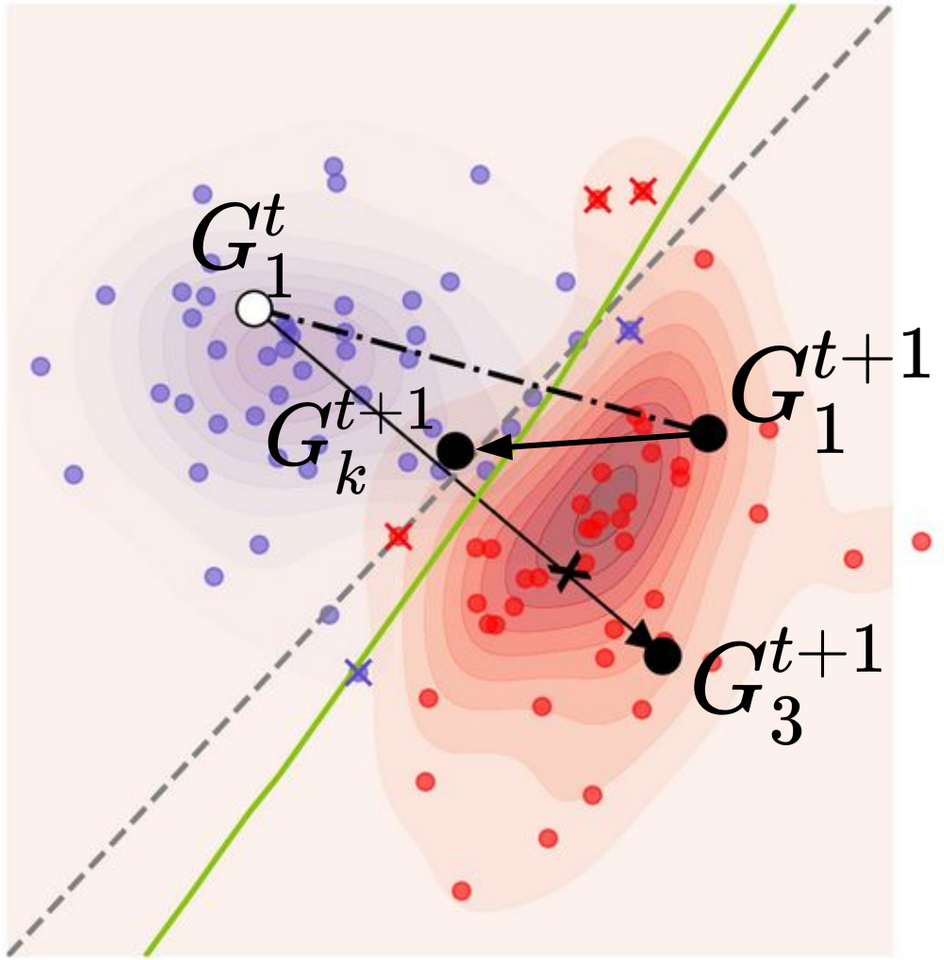


Distribution Shifts





Distribution Shifts



Problem Formulation

$$\mathcal{E}_\Phi(G_i^t) = \arg\max_{G_j^t \in \mathcal{G}} P_{cf}^t(G_j^t \mid G_i^t, \Phi(G_i^t), \neg\Phi(G_i^t))$$

probability of G_j^t being in-distribution counterfactual of G_i^t

Any other class that isn't $\Phi(G_i^t)$

Shifting towards a generative classification (GC) perspective

Prenkaj et al. Adapting to Change: Robust Counterfactual Explanations in Dynamic Data Landscapes. arXiv:2308.02353 [cs.LG]

Generative Classification (GC) Perspective

Generative Classifiers (GCs) perform classification by modeling the full joint distribution of features x and class labels y

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(x, y) = \arg\max_{y \in \mathcal{Y}} p(x|y)\, p(y) =$$

$$= \arg\max_{y \in \mathcal{Y}} \log p(x|y) + \log p(y)\,.$$

Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. NeurIPS'01

Generative Classification (GC) Perspective

- Superior generalization capabilities over discriminative classifiers
- Accurately calibrated posteriors
- Increased adversarial robustness

Ilkay Ulusoy and Christopher M Bishop. 2006. Comparison of generative and discriminative techniques for object detection and classification. In Toward Category-Level Object Recognition. Springer, 173–195

Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. 2020. Training normalizing flows with the information bottleneck for competitive generative classification. Advances in Neural Information Processing Systems 33 (2020), 7828–7840

Yingzhen Li, John Bradshaw, and Yash Sharma. 2019. Are generative classifiers more robust to adversarial attacks?. In International Conference on Machine Learning. PMLR, 3804–3814.

Variational Graph Autoencoders (VGAEs)

$$p\left(G|y\right) = \int_{\mathbf{z}\in\mathcal{Z}} p\left(G|\mathbf{z},y\right) p\left(\mathbf{z}|y\right) dz$$

VGAEs (Decoder)

- To represent $p\ (G \mid y)$, we use a single VGAE for each class, which is dependent on the class where each node has a latent vector and then define

$$p_{\theta_y}\left(G|\mathbf{z},y\right) = p_{\theta_y}\left(\mathbf{A},\mathbf{X}|\mathbf{z},y\right)$$
$$= p_{\theta_y}\left(\mathbf{X}|\mathbf{A},\mathbf{z},y\right) p_{\theta_y}\left(\mathbf{A}|\mathbf{z},y\right)$$

VGAEs (Encoder)

$$q_{\varphi_y}\left(\mathbf{z}|G,y\right)=\prod_{v_i} q_{\varphi_y}\left(\mathbf{z}_{v_i}|G,y\right)$$

$$q\left(z_{v_i}|G,y\right)=\mathcal{N}\left(z_{v_i}|\mu_{v_i},\gamma^2\mathbf{I}\right),$$

$$\mu=\left[\mu_{v_1},\ldots,\mu_{v_n}\right]=\mathrm{GCN}_{\varphi_y}\left(G\right)$$

Bridging Reconstruction and GC

$$\mathbf{ELBO}_y\left(\theta_y, \varphi_y\right) = \underset{q_{\varphi_y}(z|G,y)}{\mathbb{E}}\left[\log p_{\theta_y}\left(G|z,y\right)\right] - \mathbf{KL}\left[q_{\varphi_y}\left(z|G,y\right) \middle\| p\left(z\right)\right]$$

$$\left(\theta_y^*, \varphi_y^*\right) = \arg\max_{\theta_y, \varphi_y} \mathbf{ELBO}_y\left(\theta_y, \varphi_y\right) \ \ \forall y \in \mathcal{Y}$$

Bridging Reconstruction and GC

- Having obtained a generative latent variable model of a specific class, we can now exploit its power to perform generative classification
- If the variational family is expressive enough, the ELBO converges to the logarithm of the true class-conditional probability
- Use the generative models to compare different class probabilities and perform generative classification

Bridging Reconstruction and GC

Proposition 1: Comparing Distance-Augmented Reconstruction Losses performs Implicit GC

If the density model is sufficiently expressive, i.e., it covers the true data generating process, computing

$$\hat{y} = \arg\min_{y \in \mathcal{Y}} \frac{1}{2} \left(\mathop{\mathbb{E}}_{q_{\varphi_y^*}(z|G,y)} \left[\frac{\|g_{\theta_y^*}(z) - G\|_2^2}{\sigma^2} \right] + \|f_{\varphi_y^*}(G)\|_2^2 \right] - \log p(y),$$

is equivalent to performing generative classification for an input graph.

Bridging Reconstruction and GC

Proposition 1: Comparing Distance-Augmented Reconstruction Losses performs Implicit GC

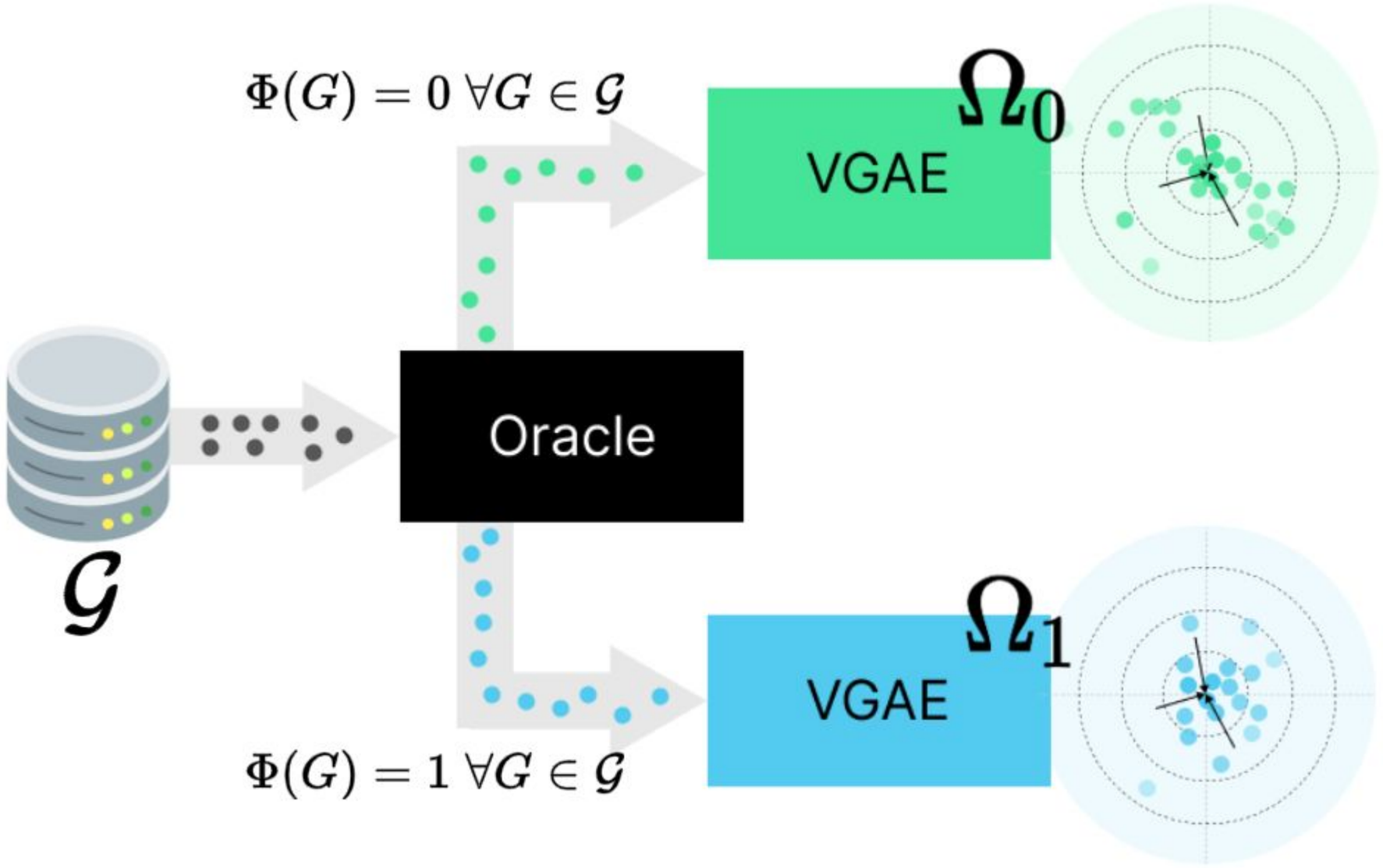
If the density model is sufficiently expressive, i.e., it covers the true data generating process, computing

$$\hat{y} = \arg\min_{y \in \mathcal{Y}} \frac{1}{2} \left(\mathop{\mathbb{E}}_{q_{\varphi_y^*}(z|G,y)} \left[\frac{\| g_{\theta_y^*}(z) - G \|_2^2}{\sigma^2} \right] + \| f_{\varphi_y^*}(G) \|_2^2 \right] - \log p(y) ,$$

is equivalent to performing generative classification for an input graph.

decoder **encoder**

GRACIE: Class Representation Experts

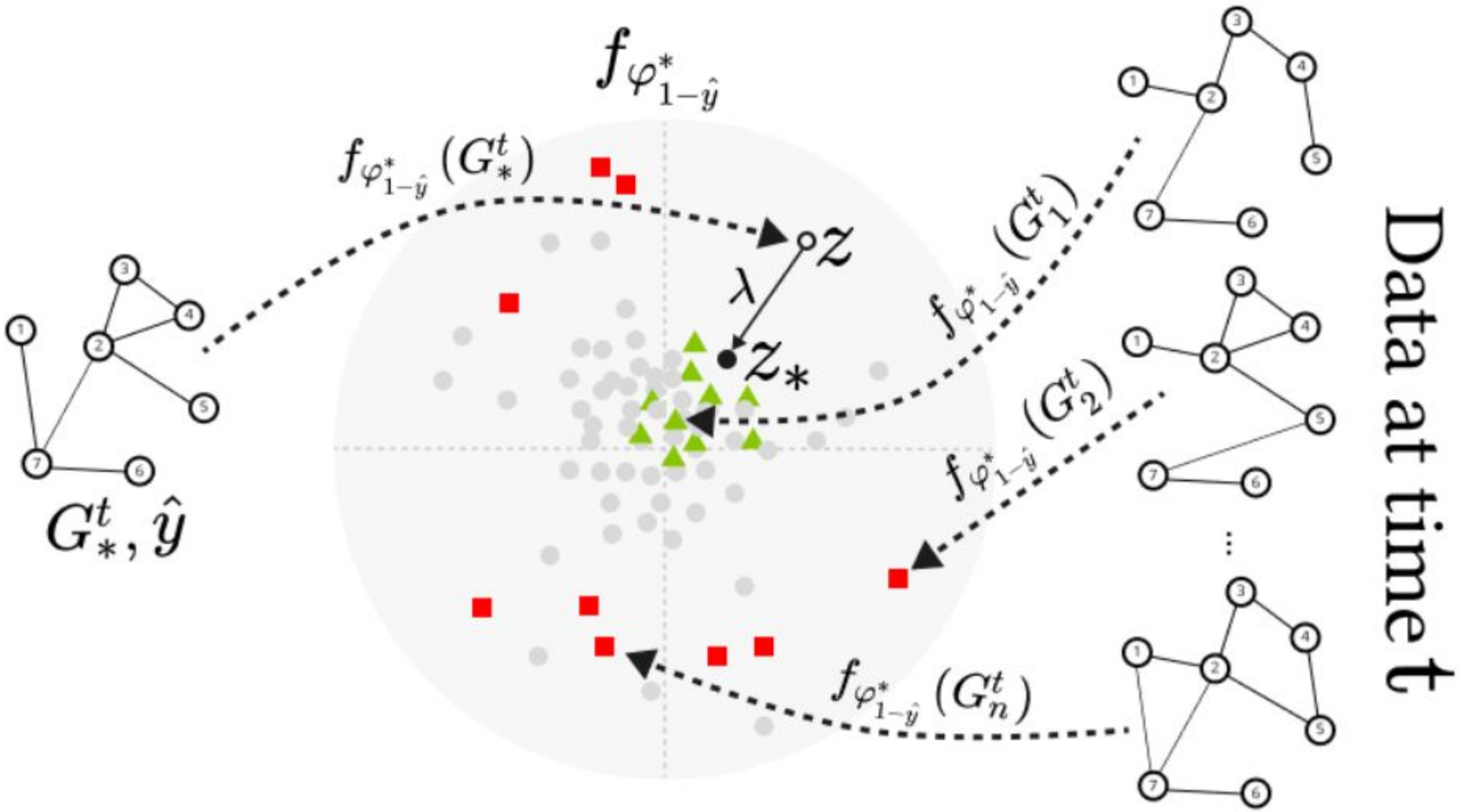


GRACIE: TRAINING

$$-\mathrm{ELBO}_y\left(\theta_y, \varphi_y\right) = \mathcal{L}_{rec} + \mathcal{L}_{dist}$$

$$= \frac{1}{2}\left(\mathop{\mathbb{E}}_{q_{\varphi_y}(\mathbf{z}|G)}\left[\frac{||g_{\theta_y}(\mathbf{z}) - G||_2^2}{\sigma^2}\right] + ||f_{\varphi_y}(G)||_2^2\right)$$

Inference and Finding Latent Counterfactuals



Dynamic Update

- Use the learned representation of the VGAEs
- For each graph, find k candidate counterfactuals close to the center of the VGAE responsible to learn the counterfactual class
- We can use these counterfactuals to update the counterfactual VGAE and the graph itself to update the factual VGAE
- **GRACIE is semi-supervised in the first snapshot, and completely unsupervised in the next snapshots**

GRACIE: RECAP

- GRACIE is one of the first generative approaches to address dynamic counterfactual explainability in the context of temporal graphs
- It leverages VGAEs, self-supervisedly, to learn class representations and adapt to data distribution shifts
- **The center of the latent space of the VGAEs should be used to find valid counterfactual**

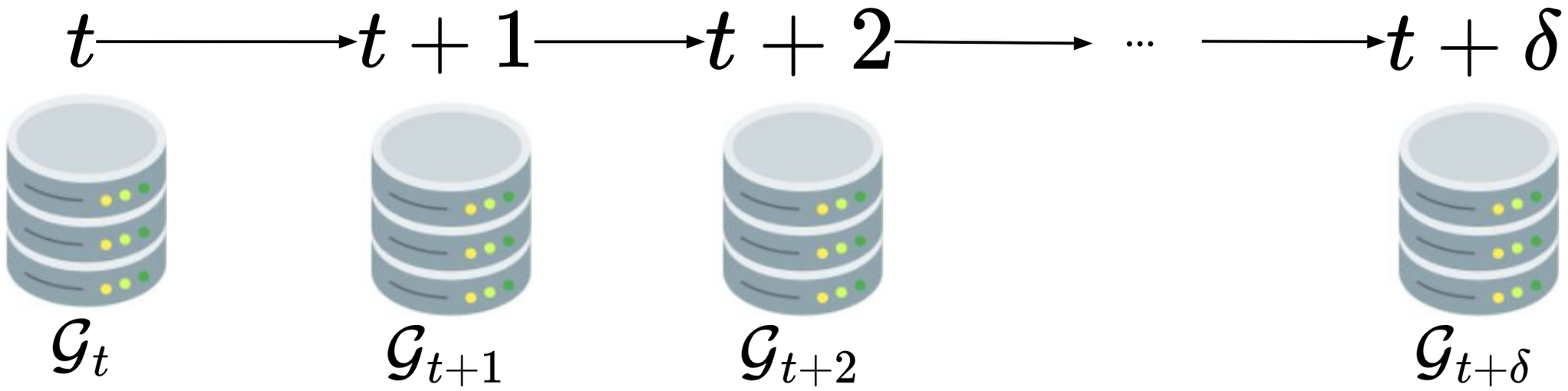
Adapting Existing Instance-Based GCEs to Dynamic Graphs

How would You Do It?

- How you choose CTDG or DTDG?
- How can a time-independent oracle become a time-relevant one (**note: not time-dependent**)?
- How would the evaluation look like now?

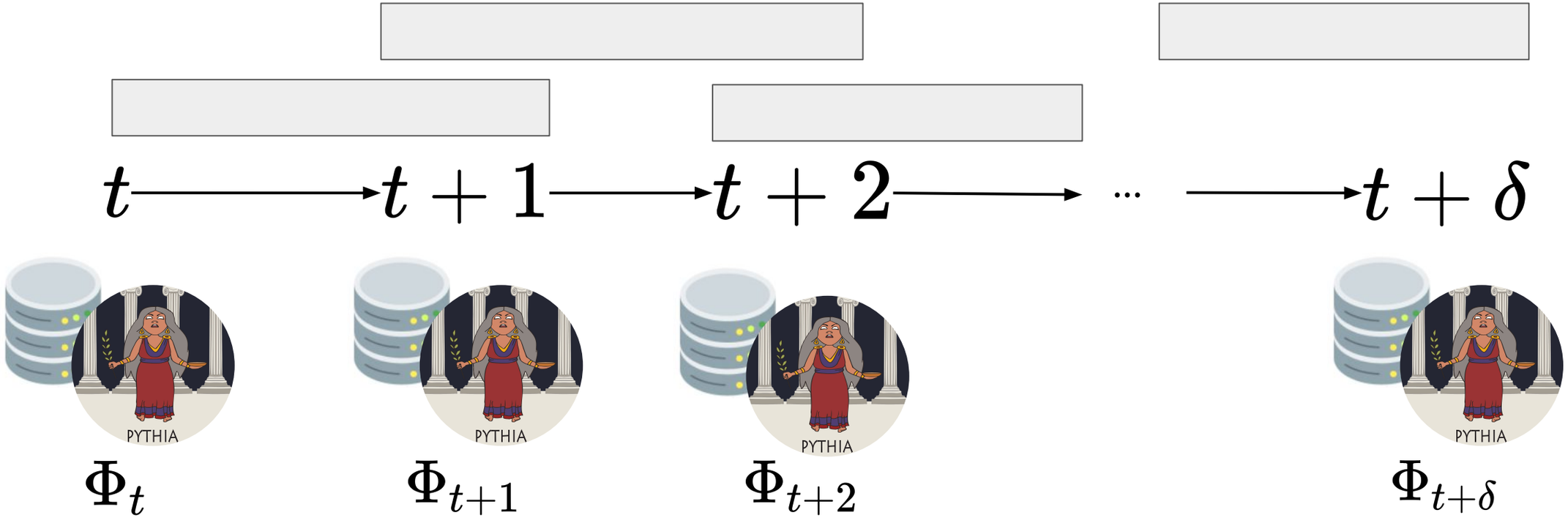
Easy Hack

- **Choose DTDG; treat each snapshot as an isolated dataset**



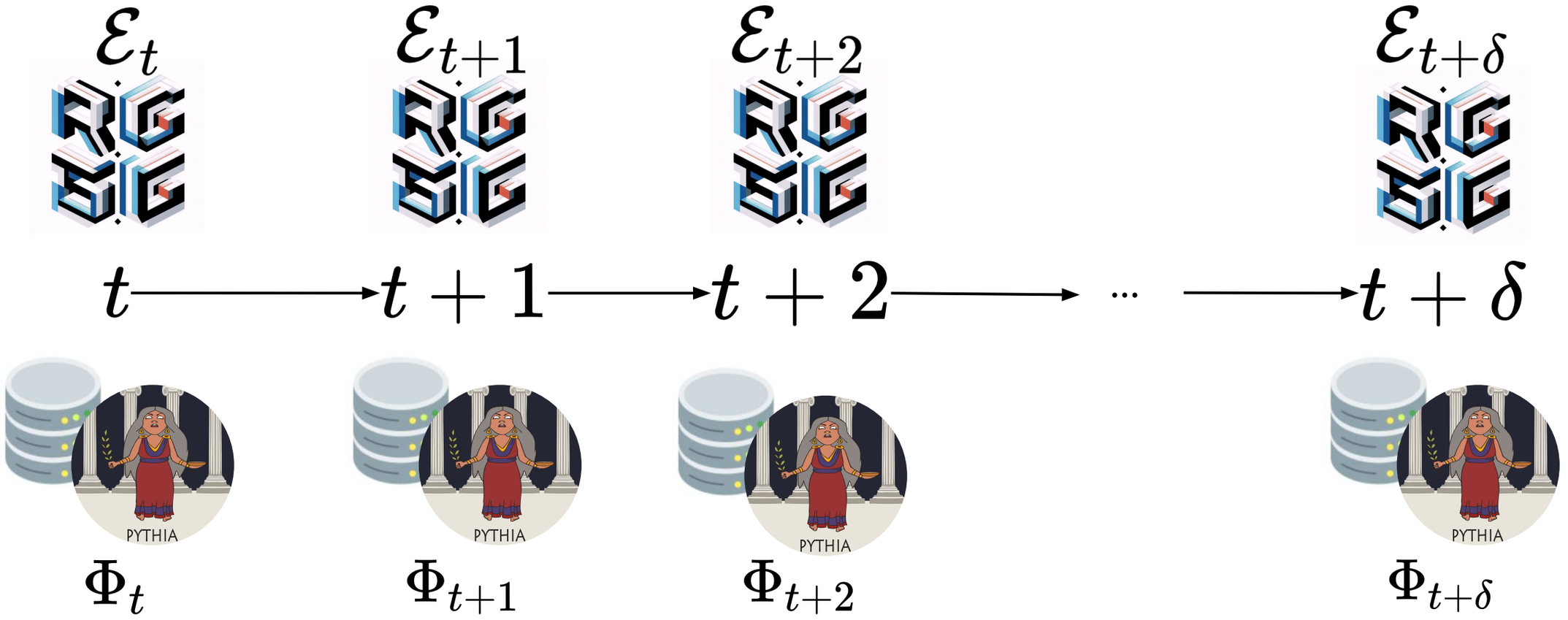
Easy Hack

- **Choose DTDG; train the oracles on each snapshot or in a sliding window fashion;**



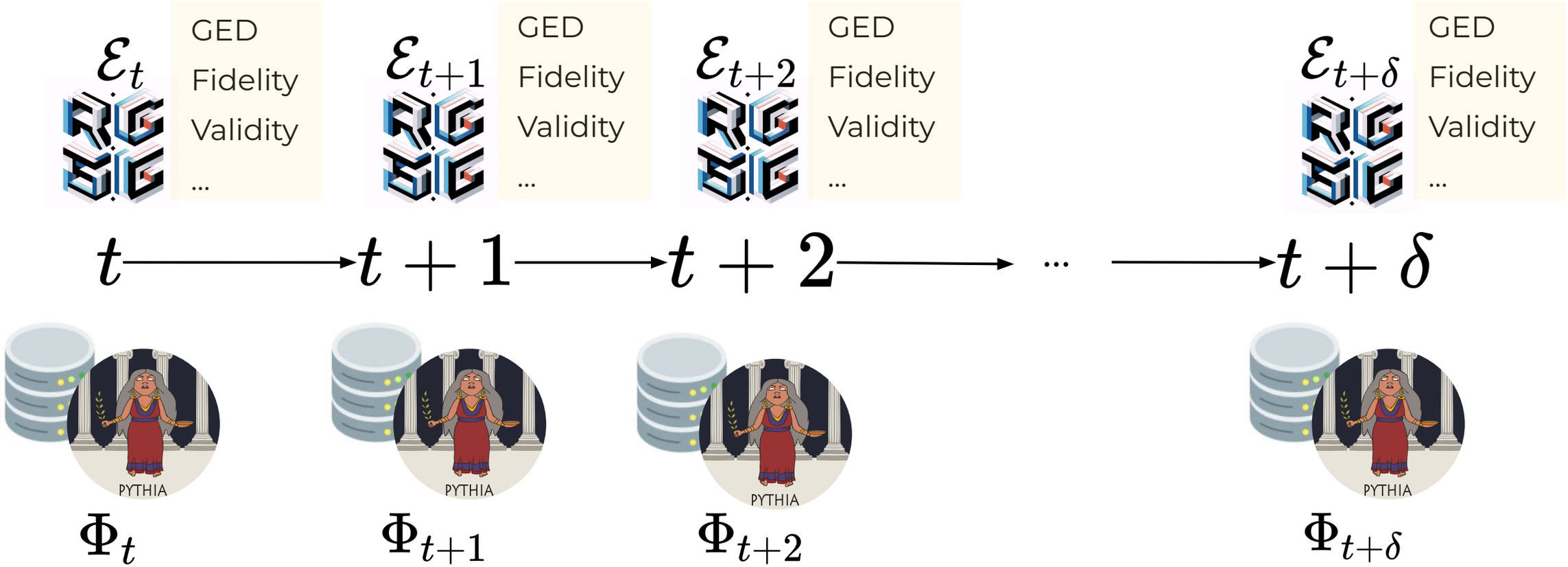
Easy Hack

- **Choose DTDG; explain each oracle on every snapshot in isolation;**



Easy Hack

- **Choose DTDG; evaluate on each snapshot and take averages;**



Easy Hack

An easy hack but not fair since the explainers aren’t time-dependent nor time-relevant.

Challenges and hiccups in Dynamic GCE

Challenges and Hiccups

- Lack of real-world benefits for dynamic counterfactuals
  - *What's a real-case scenario where this would make sense?*
- DTDGs assume same set of main nodes (seeds) in different snapshots (GRACIE)
  - *Might be realistic when monitoring bank transactions*
- CTDGs seem to be only suitable for link prediction tasks
- Misaligned generative classifications when distribution shifts happen (notice GRACIE's first snapshot training)

Part V

Evaluating Graph Counterfactual Explanations

by Efstratios Zaradoukas

Based on:

Prenkaj, B., Zaradoukas, E., & Kasneci, G. (2025). *Graph inverse style transfer for counterfactual explainability*. In **Proceedings of the 42nd International Conference on Machine Learning (ICML)**

Evaluation datasets

Overview of GCE benchmarking datasets

Simple Synthetic data with ground truth

Dataset	Domain	Publicly Available Repository (Data or Code)	Used by
Tree-Cycles [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
Tree-Grid [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[73, 77, 113]
Tree-Infinity	synthetic	https://github.com/MarioTheOne/GRETEL	[37]
BA-Shapes [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
BA-Community [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[77]
BA-2motifs [98]	synthetic	https://github.com/flyingdoog/PGExplainer	[70, 77]
ADHD [134]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/adhd	[80]
ASD [135, 136]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/autism/asd	[80]
BBBP [148]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=bbbp.zip	[120]
HIV [137–139]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=hiv.zip	[120, 123]
Ogbg-molhiv [140]	molecular	https://huggingface.co/datasets/OGB/ogbg-molhiv	[79]
Mutagenicity [141]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/Mutagenicity.zip	[70, 77, 123]
NCI1 [143]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/NCI1.zip	[70, 77, 123]
TOX21 [142]	molecular	https://tripod.nih.gov/tox21/challenge/data.jsp	[75]
ESOL [144]	molecular	https://github.com/deepchem/deepchem	[70, 75]
Proteins [145]	molecular	https://chrsmrrs.github.io/datasets/docs/datasets/	[123]
Davis [149]	molecular	http://staff.cs.utu.fi/~aatapa/data/DrugTarget/	[76]
PDBBind [150]	molecular	http://www.pdbbind.org.cn/	[76]
CiteSeer [146]	social	https://linqs.org/datasets/	[70, 112]
IMDB-M [147]	social	https://virginia.app.box.com/s/941v9pwh83lfw5vnwfbgcertlsoivg5j	[79]
CORA [151]	social	https://relational.fit.cvut.cz/dataset/CORA	[112]
Musae-Facebook [152]	social	https://www.kaggle.com/datasets/rozemberczki/musae-facebook-pagepage-network	[112]
LastFM [153]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/lastfm	[114]
Yelp [154]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018/	[114]

Overview of GCE benchmarking datasets

Dataset	Domain	Publicly Available Repository (Data or Code)	Used by
Tree-Cycles [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
Tree-Grid [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[73, 77, 113]
Tree-Infinity	synthetic	https://github.com/MarioTheOne/GRETEL	[37]
BA-Shapes [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
BA-Community [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[77]
BA-2motifs [98]	synthetic	https://github.com/flyingdoog/PGExplainer	[70, 77]
ADHD [134]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/adhd	[80]
ASD [135, 136]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/autism/asd	[80]
BBBP [148]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=bbbp.zip	[120]
HIV [137–139]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=hiv.zip	[120, 123]
Ogbg-molhiv [140]	molecular	https://huggingface.co/datasets/OGB/ogbg-molhiv	[79]
Mutagenicity [141]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/Mutagenicity.zip	[70, 77, 123]
NCI1 [143]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/NCI1.zip	[70, 77, 123]
TOX21 [142]	molecular	https://tripod.nih.gov/tox21/challenge/data.jsp	[75]
ESOL [144]	molecular	https://github.com/deepchem/deepchem	[70, 75]
Proteins [145]	molecular	https://chrsmrrs.github.io/datasets/docs/datasets/	[123]
Davis [149]	molecular	http://staff.cs.utu.fi/~aatapa/data/DrugTarget/	[76]
PDBBind [150]	molecular	http://www.pdbbind.org.cn/	[76]
CiteSeer [146]	social	https://linqs.org/datasets/	[70, 112]
IMDB-M [147]	social	https://virginia.app.box.com/s/941v9pwh83lfw5vnwfbgcertlsoivg5j	[79]
CORA [151]	social	https://relational.fit.cvut.cz/dataset/CORA	[112]
Musae-Facebook [152]	social	https://www.kaggle.com/datasets/rozemberczki/musae-facebook-pagepage-network	[112]
LastFM [153]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/lastfm	[114]
Yelp [154]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018/	[114]

Brain networks without attributes

Overview of GCE benchmarking datasets

Dataset	Domain	Publicly Available Repository (Data or Code)	Used by
Tree-Cycles [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
Tree-Grid [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[73, 77, 113]
Tree-Infinity	synthetic	https://github.com/MarioTheOne/GRETEL	[37]
BA-Shapes [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
BA-Community [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[77]
BA-2motifs [98]	synthetic	https://github.com/flyingdoog/PGExplainer	[70, 77]
ADHD [134]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/adhd	[80]
ASD [135, 136]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/autism/asd	[80]
BBBP [148]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=bbbp.zip	[120]
HIV [137–139]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=hiv.zip	[120, 123]
Ogbg-molhiv [140]	molecular	https://huggingface.co/datasets/OGB/ogbg-molhiv	[79]
Mutagenicity [141]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/Mutagenicity.zip	[70, 77, 123]
NCI1 [143]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/NCI1.zip	[70, 77, 123]
TOX21 [142]	molecular	https://tripod.nih.gov/tox21/challenge/data.jsp	[75]
ESOL [144]	molecular	https://github.com/deepchem/deepchem	[70, 75]
Proteins [145]	molecular	https://chrsmrrs.github.io/datasets/docs/datasets/	[123]
Davis [149]	molecular	http://staff.cs.utu.fi/~aatapa/data/DrugTarget/	[76]
PDBBind [150]	molecular	http://www.pdbbind.org.cn/	[76]
CiteSeer [146]	social	https://linqs.org/datasets/	[70, 112]
IMDB-M [147]	social	https://virginia.app.box.com/s/941v9pwh83lfw5vnwfbgcertlsoivg5j	[79]
CORA [151]	social	https://relational.fit.cvut.cz/dataset/CORA	[112]
Musae-Facebook [152]	social	https://www.kaggle.com/datasets/rozemberczki/musae-facebook-pagepage-network	[112]
LastFM [153]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/lastfm	[114]
Yelp [154]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018/	[114]

Attributed molecular datasets without ground-truth explanations

Overview of GCE benchmarking datasets

Dataset	Domain	Publicly Available Repository (Data or Code)	Used by
Tree-Cycles [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
Tree-Grid [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[73, 77, 113]
Tree-Infinity	synthetic	https://github.com/MarioTheOne/GRETEL	[37]
BA-Shapes [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[70, 73, 77, 113]
BA-Community [31]	synthetic	https://github.com/RexYing/gnn-model-explainer	[77]
BA-2motifs [98]	synthetic	https://github.com/flyingdoog/PGExplainer	[70, 77]
ADHD [134]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/adhd	[80]
ASD [135, 136]	-omics	https://github.com/MarioTheOne/GRETEL/tree/main/data/datasets/autism/asd	[80]
BBBP [148]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=bbbp.zip	[120]
HIV [137–139]	molecular	https://www.kaggle.com/datasets/mmelahi/cheminformatics?select=hiv.zip	[120, 123]
Ogbg-molhiv [140]	molecular	https://huggingface.co/datasets/OGB/ogbg-molhiv	[79]
Mutagenicity [141]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/Mutagenicity.zip	[70, 77, 123]
NCI1 [143]	molecular	https://ls11-www.cs.tu-dortmund.de/people/morris/graphkerneldatasets/NCI1.zip	[70, 77, 123]
TOX21 [142]	molecular	https://tripod.nih.gov/tox21/challenge/data.jsp	[75]
ESOL [144]	molecular	https://github.com/deepchem/deepchem	[70, 75]
Proteins [145]	molecular	https://chrsmrrs.github.io/datasets/docs/datasets/	[123]
Davis [149]	molecular	http://staff.cs.utu.fi/~aatapa/data/DrugTarget/	[76]
PDBBind [150]	molecular	http://www.pdbbind.org.cn/	[76]
CiteSeer [146]	social	https://linqs.org/datasets/	[70, 112]
IMDB-M [147]	social	https://virginia.app.box.com/s/941v9pwh83lfw5vnwfbgcertlsoivg5j	[79]
CORA [151]	social	https://relational.fit.cvut.cz/dataset/CORA	[112]
Musae-Facebook [152]	social	https://www.kaggle.com/datasets/rozemberczki/musae-facebook-pagepage-network	[112]
LastFM [153]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/lastfm	[114]
Yelp [154]	social	https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/yelp2018/	[114]

Social Networks without ground truth. Usually have many binary attributes or none

Overview of GCE benchmarking datasets

Small molecules

Name	Source	Statistics				Labels/Attributes	
		Graphs	*Classes*	*Avg. Nodes*	*Avg. Edges*	*Node Labels*	*Edge Label.*
AIDS	[16,17]	2000	2	15.69	16.20	+	+
alchemy_full	[29]	202579	R (12)	10.10	10.44	+	+
aspirin	[36]	111763	R (1)	21.00	151.52	+	–
benzene	[36]	527984	R (1)	12.00	64.94	+	–
BZR	[7]	405	2	35.75	38.36	+	–

Social networks

Name	Source	Statistics			Labels/Attributes				
		Graphs	*Classes*	*Avg. Nodes*	*Avg. Edges*	*Node Labels*	*Edge Labels*	*Node Attr.*	*Geometry*
COLLAB	[14]	5000	3	74.49	2457.78	–	–	–	–
dblp_ct1	[32]	755	2	52.87	99.78	temporal	–	–	–
dblp_ct2	[32]	755	2	52.87	99.78	temporal	–	–	–
DBLP_v1	[26]	19456	2	10.48	19.65	+	+	–	–
deezer_ego_nets	[30]	9629	2	23.49	65.25	–	–	–	–

Synthetic

Name	Source	Statistics				Labels/Attributes					Download (ZIP)
		Graphs	*Classes*	*Avg. Nodes*	*Avg. Edges*	*Node Labels*	*Edge Labels*	*Node Attr.*	*Geometry*	*Edge Attr.*	
COLORS-3	[27]	10500	11	61.31	91.03	–	–	+ (4)	–	–	COLORS-3
SYNTHETIC	[3]	300	2	100.00	196.00	+	–	+ (1)	–	–	SYNTHETIC
SYNTHETICnew	[3,10]	300	2	100.00	196.25	–	–	+ (1)	–	–	SYNTHETICnew
Synthie	[21]	400	4	95.00	172.93	–	–	+ (15)	–	–	Synthie
TRIANGLES	[27]	45000	10	20.85	32.74	–	–	–	–	–	TRIANGLES

Bioinformatics

Name	Source	Statistics			Labels/Attributes						Download (ZIP)
		Graphs	*Classes*	*Avg. Nodes*	*Avg. Edges*	*Node Labels*	*Edge Labels*	*Node Attr.*	*Geometry*	*Edge Attr.*	
DD	[6,22]	1178	2	284.32	715.66	+	–	–	–	–	DD
ENZYMES	[4,5]	600	6	32.63	62.14	+	–	+ (18)	–	–	ENZYMES
KKI	[26]	83	2	26.96	48.42	+	–	–	–	–	KKI
OHSU	[26]	79	2	82.01	199.66	+	–	–	–	–	OHSU
Peking_1	[26]	85	2	39.31	77.35	+	–	–	–	–	Peking_1
PROTEINS	[4,6]	1113	2	39.06	72.82	+	–	+ (1)	–	–	PROTEINS
PROTEINS_full	[4,6]	1113	2	39.06	72.82	+	–	+ (29)	–	–	PROTEINS_full

Computer vision

Name	Source	Statistics			Labels/Attributes						Download (ZIP)
		Graphs	*Classes*	*Avg. Nodes*	*Avg. Edges*	*Node Labels*	*Edge Labels*	*Node Attr.*	*Geometry*	*Edge Attr.*	
COIL-DEL	[16,18]	3900	100	21.54	54.24	–	+	+ (2)	–	–	COIL-DEL
COIL-RAG	[16,18]	3900	100	3.01	3.02	–	–	+ (64)	–	+ (1)	COIL-RAG
Cuneiform	[25]	267	30	21.27	44.80	+	+	+ (3)	3D	+ (2)	Cuneiform
Fingerprint	[16,19]	2149	15	7.06	5.76	–	–	+ (2)	2D	+ (2)	Fingerprint
FIRSTMM_DB	[11,12,13]	41	11	1377.27	3074.10	+	–	+ (1)	–	+ (2)	FIRSTMM_DB

Morris, Christopher, et al. "TUDataset: A collection of benchmark datasets for learning with graphs." ICML 2020 Workshop on Graph Representation Learning and Beyond 2020.

Synthetic Datasets

- Designed with simple, interpretable structures
- **Examples:** Tree-Cycles, Tree-Grid, Tree-Infinity, BA-Shapes, BA-Community, BA-2motifs
- **Characteristics:**
  - Small-medium graphs with ground-truth motifs
  - Ideal for testing GNN explainability accuracy
  - High interpretability, low noise
- **Use Case:** Benchmarking explainers on well-understood graph patterns

~Omics Datasets – Biomedical Graphs

- Represent data from neuroscience and genomics
- **Examples:** ADHD, ASD
- **Characteristics:**
  - Complex biological/medical relationships
  - Real-world clinical relevance
  - Harder to interpret than synthetic datasets
- **Use Case:** Explaining GNN predictions in disease classification and biomedical research

Molecular Datasets

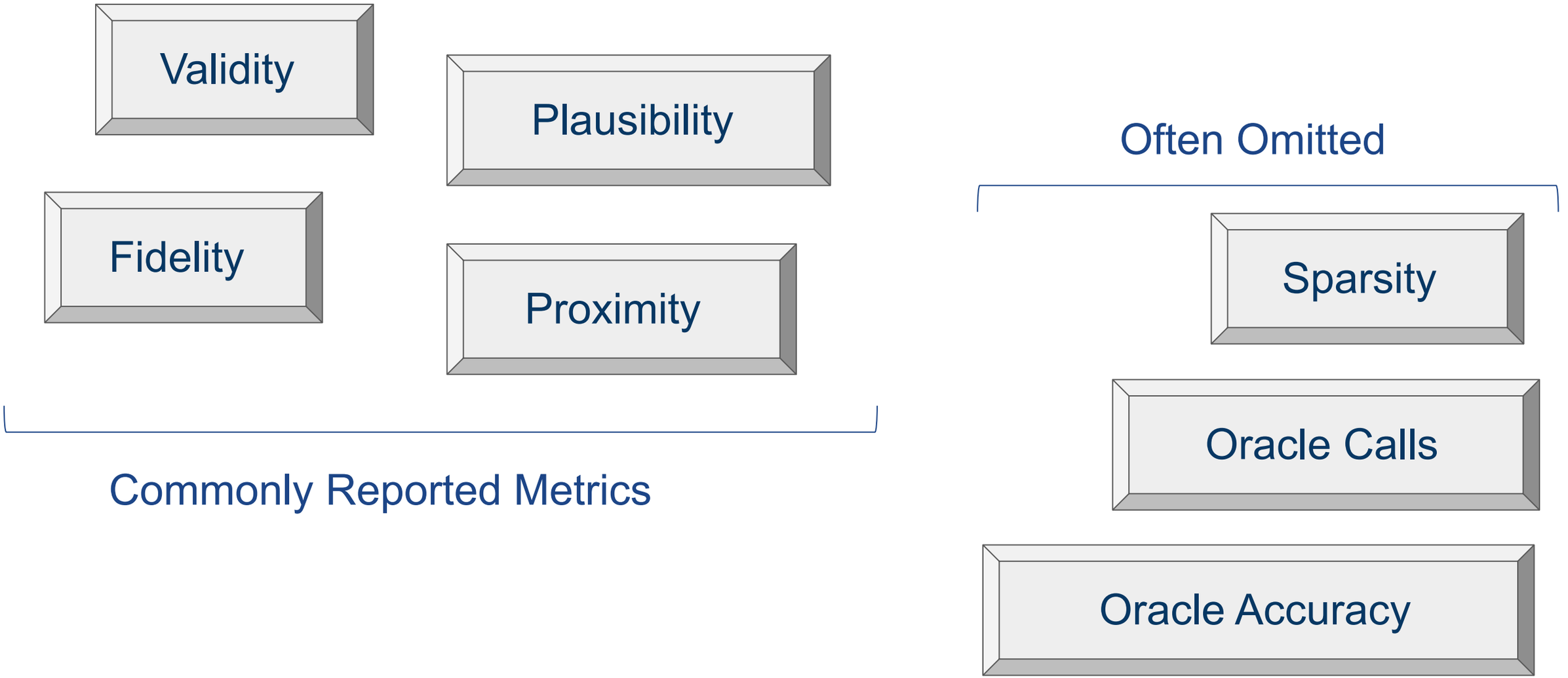
- **Examples:** BBBP, HIV, Ogbg-molhiv, Mutagenicity, NCI1, TOX21, ESOL, Proteins, Davis, PDBBind
- **Characteristics:**
  - Graphs represent molecules (atoms = nodes, bonds = edges)
  - Range from small curated datasets (ESOL) to large public benchmarks (Ogbg-molhiv)
  - Tasks: toxicity prediction, solubility, protein binding affinity
- **Use Case:** Testing explainability of models in chemical property prediction for drug discovery and cheminformatics

Social Datasets

- Represent interactions between people, documents, or platforms
- **Examples:** CiteSeer, IMDB-M, CORA, Musae-Facebook, LastFM, Yelp
- **Characteristics:**
  - Large, complex, and noisy real-world graphs
  - Common tasks include node classification, link prediction and recommendations
  - Lower interpretability compared to synthetic/molecular datasets
- **Use Case:** Studying explainability in dynamic social & information networks

EVALUATION METRICS

Overview of evaluation Metrics



Validity

- **Definition**: Validity checks whether a counterfactual actually flips the model's decision relative to the original instance.
- **Formally**, for the original instance G, the counterfactual G', and oracle Φ, validity is an indicator function:

$$\Omega(G, G') = \mathbb{I}[\Phi(G) \neq \Phi(G')]$$

- **Binary signal only**: validity ignores *how* confident Φ is and the *distance* between G and G′.
- **Model access**: requires querying the oracle Φ
- **Doesn't guarantee realism**: a valid G′ may be implausible or infeasible in the data domain.

Plausibility

- We need to ensure G’ obeys **domain constraints** (e.g., valence in molecules, degree/type rules, connectivity).
- Two ways to ensure plausibility:
  - Enforce constraints during search
  - Post-hoc checks
- **Common pitfall**: unconstrained search yields infeasible yet “valid” (label-flipping) counterfactuals.

Proximity

- **Graph Edit Distance (GED):** quantifies the structural distance between the original graph G and its counterfactual G'. The distance is evaluated based on a set of actions {p1, p2, . . . , pn}, that represent a path to transform G into G'. Each action p_i is associated with a $\omega(p_i)$ cost

$$\mathrm{GED}(G, G') = \min_{\{p_1,\ldots,p_n\} \in \mathcal{P}(G,G')} \sum_{i=1}^{n} \omega(p_i)$$

- We prefer counterfactuals that are **closer** to the original instance G

Sparsity

- Using the sparsity metric we compare the **similarity** between the input instance and its counterfactual concerning **input attributes**

- With the similarity S(G, G') ∈ [0,1], we compute:

$$\text{Sparsity} = \frac{1 - \mathcal{S}(G, G')}{|G'|}$$

Fidelity

- **Goal:** Measure how well a counterfactual G' reflects the trained oracle Φ's decision boundary.
- Assuming χ(G) measures if the original instance is classified correctly, fidelity is defined as :

$$\chi(G) = \mathbb{I}\big[\Phi(G) = y\big]$$

$$\Psi(G, G') = \chi(G) \; - \; \mathbb{I}\big[\Phi(G') = y\big]$$

Oracle Related metrics

- **Oracle accuracy:**
  - Test accuracy of the oracle Φ
  - Low-accuracy oracles make any GCE questionable

- **Oracle calls:**
  - No. of Φ queries used to obtain G’
  - Latency-agnostic proxy for computational complexity and scalability.

From Numbers to Trust

Quantitative Metrics

Validity: does G' flip the label?

Fidelity: does G' reflect the oracle's boundary?

Proximity/Sparsity: how small are the edits?

Cost: runtime & oracle calls.

Fairness

Robustness

Robustness in GCEs

TL;DR: Do explanations persist under small changes?

- **Why it matters? →** If tiny changes flip your explanation, stakeholders can't trust the recommended actions.
- **Definition:** Stability of the counterfactual G ' under small input perturbations, model re-seeding, and explainer randomness.
- **Factors:**
  - input noise (edge/node flips, feature jitter),
  - model randomness (different seeds/retrains)
  - search randomness (multiple restarts)

Fairness in GCEs

TL;DR: Are explanations equally good and equally feasible for everyone?

- GCEs should be **equitable**: quality and feasibility of the suggested changes shouldn't depend on sensitive attributes.
- **Three** lenses of fairness:
  - Model-level fairness of the oracle Φ (baseline).
  - Explanation-level fairness: Do validity, sparsity, GED, and fidelity degrade for some groups?
  - Action-level fairness: Are the required edits equally feasible/costly across groups?

Fairness in GCEs (Where to intervene)

- **Before explanations:** train a fair(er) GNN (reweighting, adversarial debiasing).
- **During search:** add fairness/feasibility penalties, enforce symmetric constraints.
- **After generation:** audit gaps, replace or flag inequitable counterfactuals.

Part VI
FRAMEWORKS IN THE WILD

by Andrea D’Angelo

Based on:

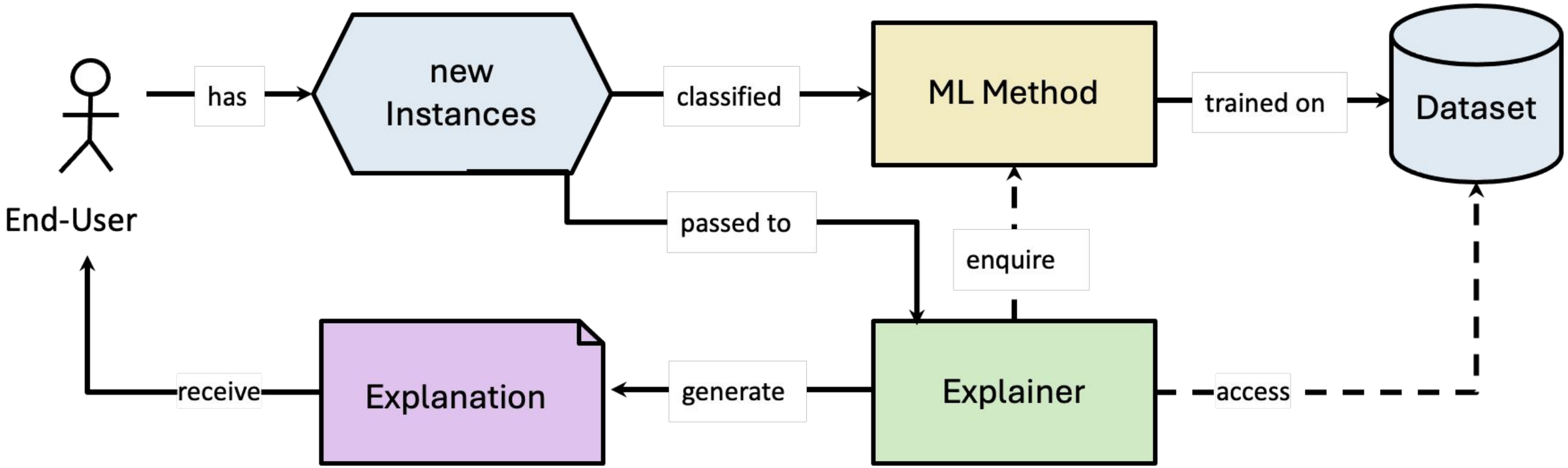
Morris, Christopher, et al. "TUDataset: A collection of benchmark datasets for learning with graphs." ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020). 2020.

SUMMING UP

- **Over the course of this tutorial,** we described the complex environment of Graph Counterfactual Explanations.
- **Many** datasets, models, explainers, and evaluation techniques are present in the literature.
- **Their number continues to grow!**
- **A unified framework** is necessary to enable fair comparison and benchmarking.

XAI workflow

- **A framework should implement all parts of the XAI workflow.**



XAI workflow (2)

- **This comes with unique challenges:**
- **Datasets and oracles** are strictly intertwined;
- **Explainers** need to be tested on the exact same settings;
- **Adding** new elements should be seamless and easy;

THE GRETEL FRAMEWORK

GRETEL WHO?

Gretel is a **generic platform** that allows the researchers to **speed up** the process of **developing** *and* **testing** new
Graph Counterfactual Explanation Methods

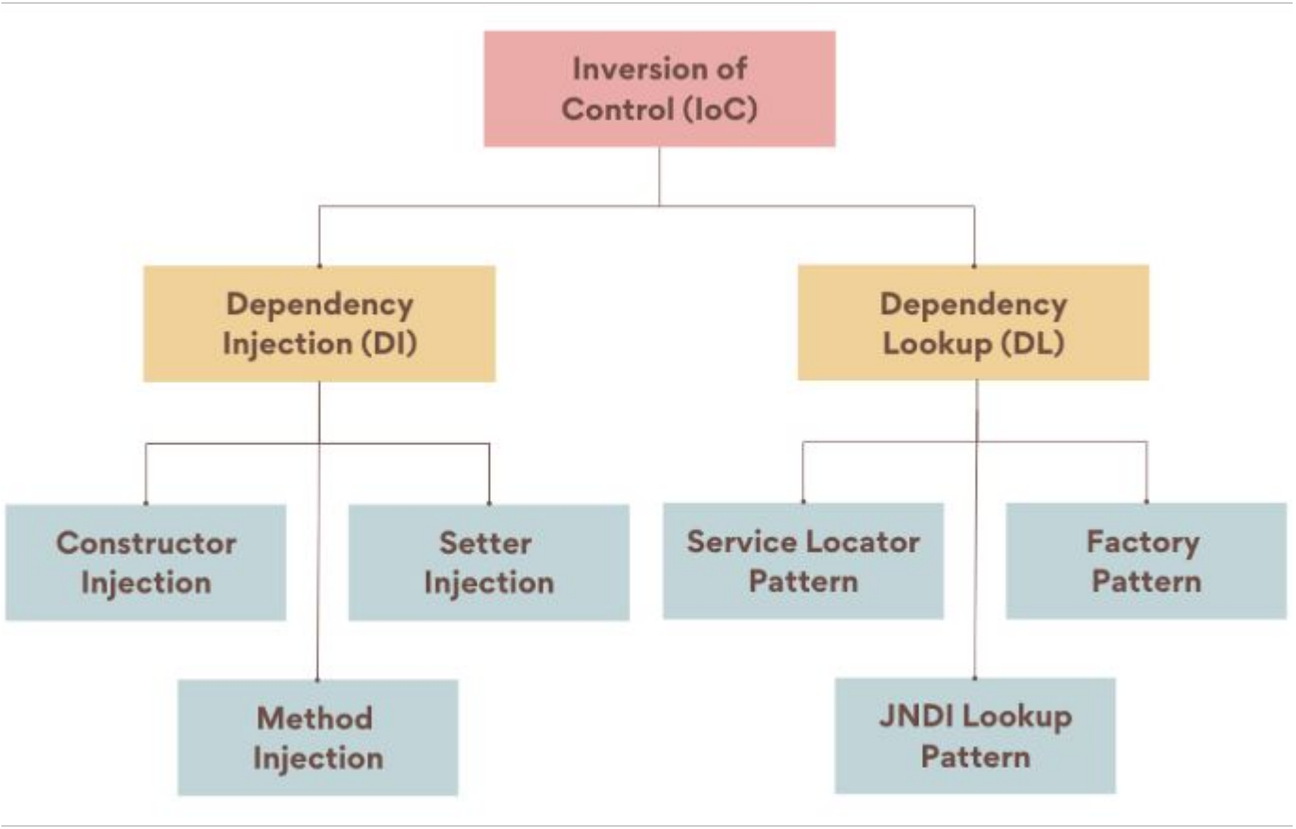
- Object Oriented paradigm;
- Inversion of Control;
- Modular + Extensible;
- Reproducibility Ready

CIKM'22 (v1)

https://github.com/aiim-research/GRETEL (v2)

WSDM '23 (v1)

Design Patterns (1)



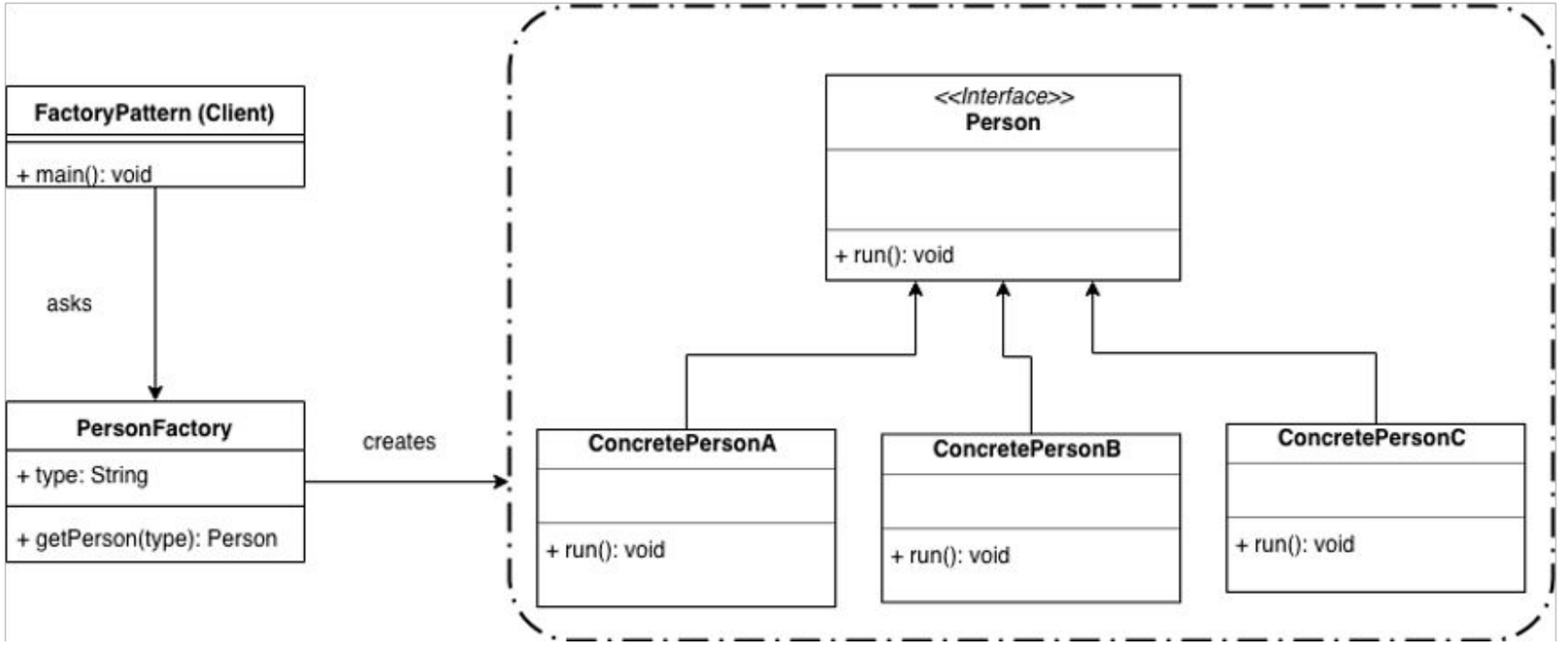
- **Inversion of Control (IoC):** shifts control of object creation and execution to an external framework.

Design Patterns (2)

- **Inversion of Control (IoC):** shifts control of object creation and execution to an external framework.

- **All experiments in GRETEL** are defined through a single, external configuration file.

Design Patterns (3)



- **Factory Pattern:** abstracts object instantiation. It encapsulates object creation in a dedicated factory class.

Design Patterns (4)

- **Factory Pattern:** abstracts object instantiation. It encapsulates object creation in a dedicated factory class.

- **GRETEL** can instantiate any object from any external library (think Pytorch, sci-kit learn) autonomously.

Design Patterns (5)

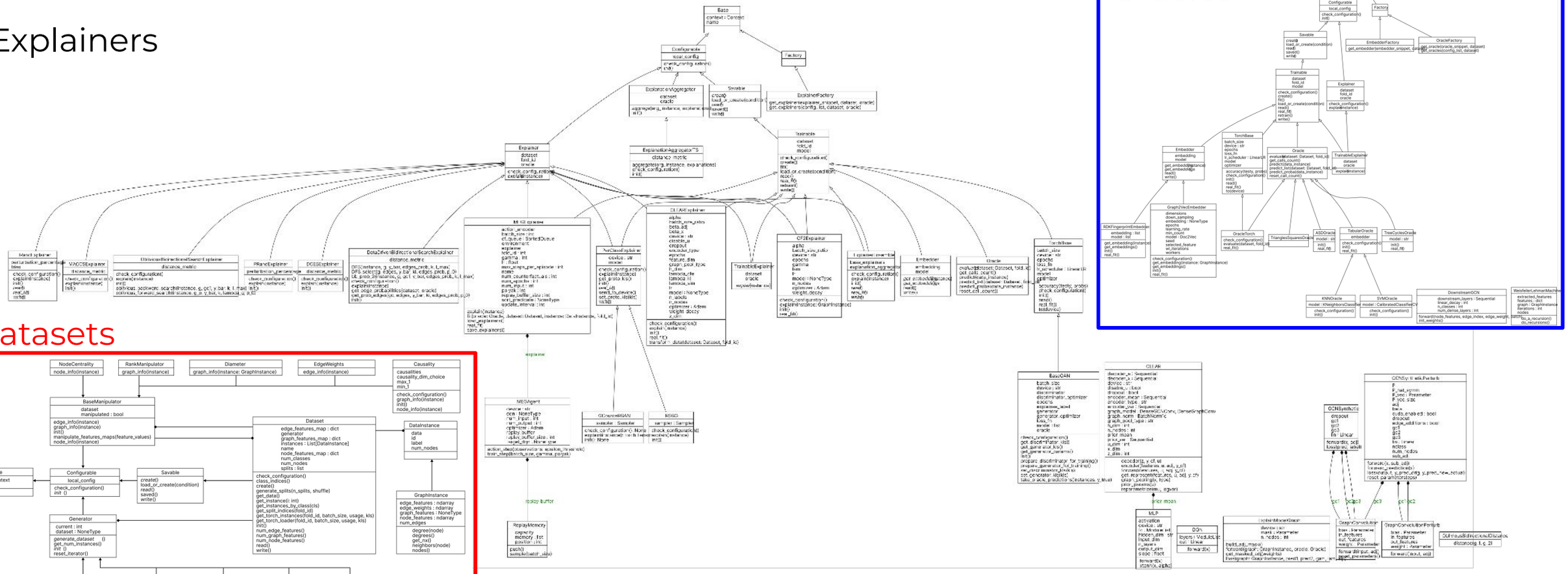
- **GRETEL** can instantiate any object from any external library (think Pytorch, sci-kit learn) autonomously.

This is particularly important given the huge body of work that already exists for datasets and GNNs.

As a result, experiments in GRETEL are defined through a **single JSON configuration file**.

CAGING THE COMPLEXITY

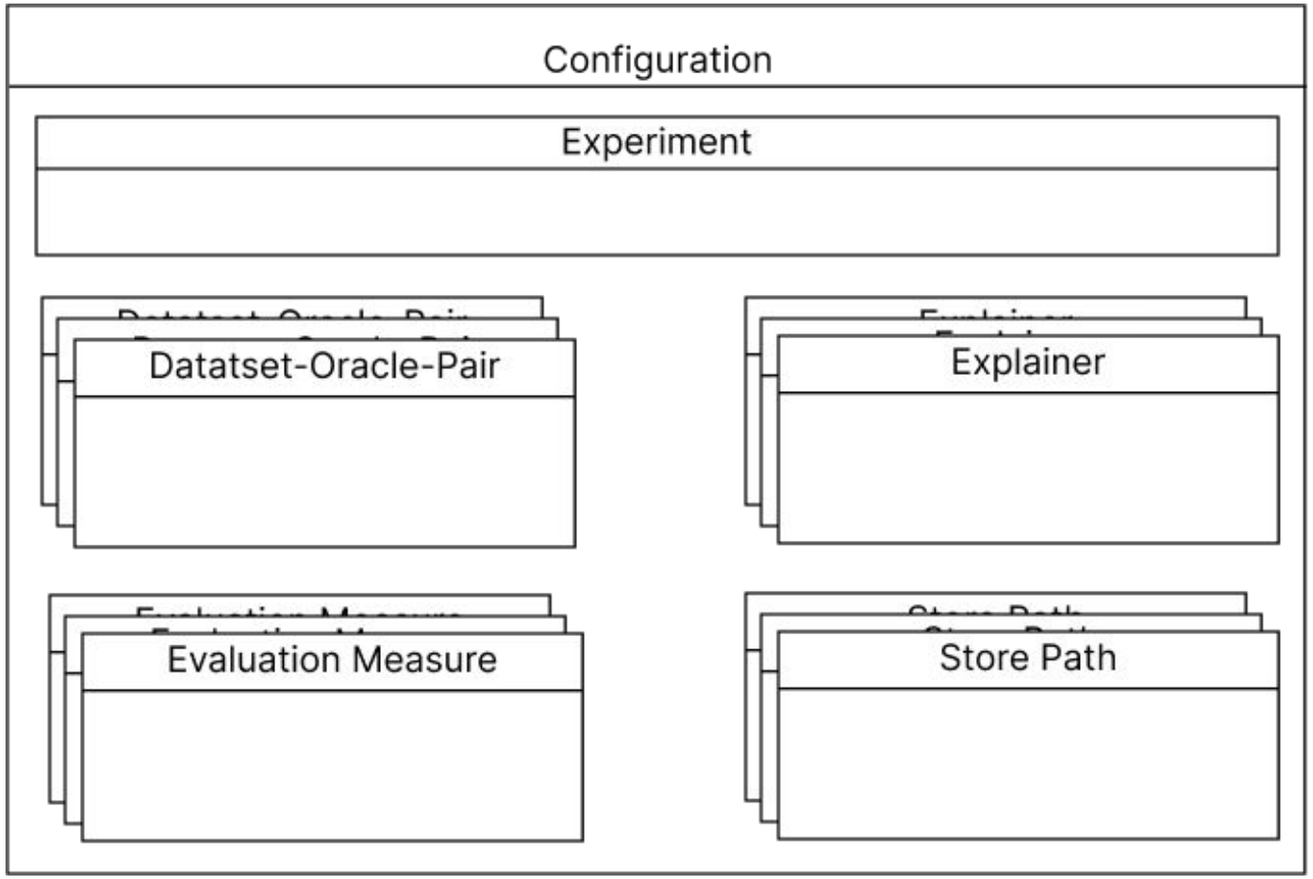
Explainers



Oracles

Datasets

Configuration Overview



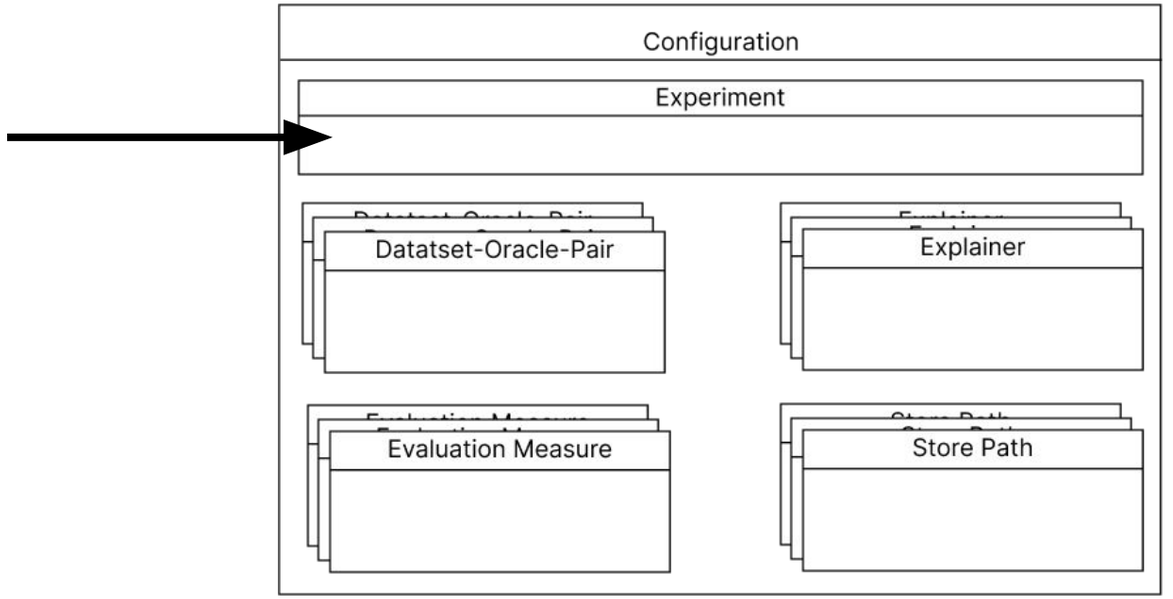

```
{
    "experiment" : {
        "scope": "examples_configs",
        "parameters" : {}
    },
    "do-pairs": [
        {"dataset" : { ... }, "oracle": { ... }},
        .
        {"dataset" : { ... }, "oracle": { ... }},
    ],
    "explainers": [
        { ... },
        .
        { ... }
    ],
    "evaluation_metrics": [
        { ... },

        .
        { ... }
    ],
    "store_paths": [
        { ... },

        .
        { ... }
    ]
}
```

We will walk through it

Experiment Section & propagate (1)




```
"experiment": {
        "scope": "examples_configs",
        "parameters": {
            "lock_release_tout":120,
            "propagate":[
                {"in_sections" : ["explainers"],"params": {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params": {"fold_id": -1}},
            ]
        }
    },
```

Experiment Section & propagate (2)

```
"experiment": {
        "scope": "examples_configs",
        "parameters": {
            "lock_release_tout":120,
            "propagate":[
                {"in_sections" : ["explainers"],"params": {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params": {"fold_id": -1}},
            ]
        }
    },
```

- The **Scope** is used to group results from different runs in the same folder.

Experiment Section & propagate (2)

```
"experiment": {
        "scope": "examples_configs",
        "parameters": {
            "lock_release_tout":120,
            "propagate":[
                {"in_sections" : ["explainers"],"params": {"fold_id": 0}},
                {"in_sections" : ["do-pairs/oracle"],"params": {"fold_id": -1}},
            ]
        }
    },
```

- The **Propagate** keyword is used to repeat the same parameters in multiple objects in a certain section.
- Here, “fold_id”:0 will be repeated across all explainers.

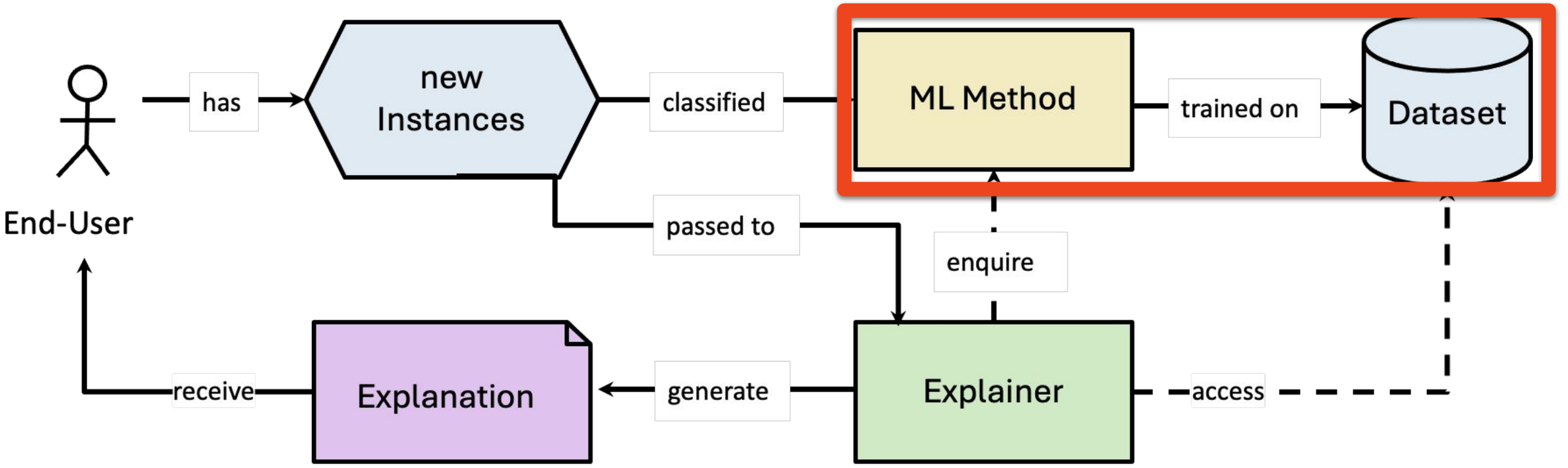
Simple Object Configuration

```
"class": "src.dataset.dataset_base.Dataset",
"parameters": {
      "generator": {
                  "class": "src.dataset.generators.treecycles_rand.TreeCyclesRand",
                  "parameters": {
                        "num_instances": 128,
                        "num_nodes_per_instance": 32,
                        "ratio_nodes_in_cycles": 0.2
                   }}}
```

- All the parameters needed to instantiate a class are defined in the JSON configuration file.
- **All** objects in the configuration follow the structure class/parameters.

Configuring an Experiment

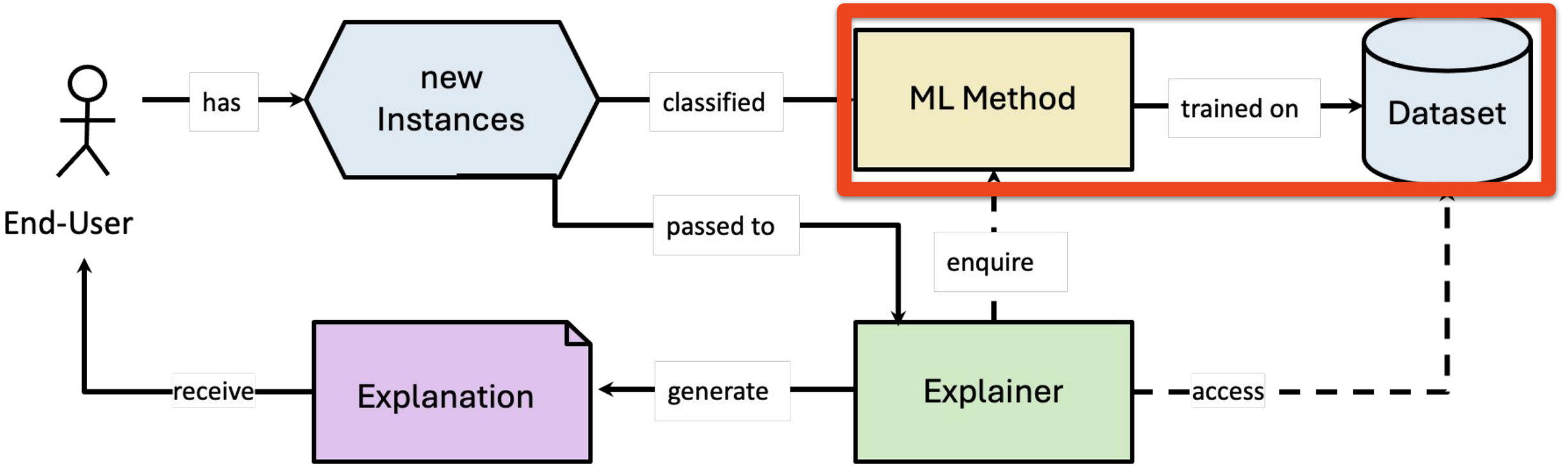
Dataset-oracle pair



- The explainer in this framework uses the ML method as an **Oracle.**

  An Oracle is always linked to the dataset it was trained on.

Dataset-oracle pair (2)



- As a result, each Dataset-Oracle pair (D-O pair) will always contain two objects: a Dataset and the model it was trained on (Oracle).

DATASET (1)

- The **Dataset** class defines all mechanisms necessary to Generate, Manipulate, Write and load a dataset.

- In GRETEL, a Dataset is **always Generated**. It can be by reading a file, or by a defined process.
- The **Manipulators** are classes defining functions that are automatically applied to the Datasets once generated.
- Once generated and manipulated, a dataset can be stored and loaded later.

Dataset (2)

```
"dataset": {
  "class": "src.dataset.dataset_base.Dataset",
  "parameters": {
      "generator": {
        "class": "src.dataset.generators.trecycles_rand.TreeCyclesRand",
        "parameters": {
           "num_instances": 128,
           "num_nodes_per_instance": 32,
           "ratio_nodes_in_cycles": 0.2
         }
        "manipulators": [
           { "class": "src.n_dataset.manipulators.centralities.NodeCentrality", \
               "parameters": {} },
           { "class": "src.n_dataset.manipulators.weights.EdgeWeights", \
               "parameters": {} }
         ]
    }
}
```

Oracle (Simple)

```
"oracle": {
    "class": "src.oracle.tabulars.svm.SVMOracle",
    "parameters": {
        "fold_id": -1,
        "embedder": {
            "class": "src.embedder.molecule.model.RDKFingerprintEmbedder",
            "parameters": {}
        },
        "model": {  "parameters": {} }
    }
 }
```

- All the parameters needed to instantiate a class are defined in the JSON configuration file.

Oracle (Real)

```
"oracle": {
  "class": "src.oracle.nn.torch.OracleTorch",
  "parameters": {
    "epochs": 200,
    "batch_size": 32,
    "optimizer": {
      "class": "torch.optim.RMSprop",
      "parameters": { "lr":0.01}
    },
    "loss_fn": {
      "class": "torch.nn.CrossEntropyLoss",
      "parameters": { "reduction": "mean" }
     },
     "model": {
      "class": "src.oracle.nn.gcn.DownstreamGCN",
      "parameters":{
          "num_conv_layers":3,
          "num_dense_layers":1,
          "conv_booster":2,
          "linear_decay":1.8}
     }
  }
```

The OracleTorch is a bigger object that contains the model, along with its params

Oracle (Real)

```
"oracle": {
  "class": "src.oracle.nn.torch.OracleTorch",
  "parameters": {
    "epochs": 200,
    "batch_size": 32,
    "optimizer": {
      "class": "torch.optim.RMSprop",
      "parameters": { "lr":0.01}
    },
    "loss_fn": {
      "class": "torch.nn.CrossEntropyLoss",
      "parameters": { "reduction": "mean" }
     },
     "model": {
      "class": "src.oracle.nn.gcn.DownstreamGCN",
      "parameters":{
          "num_conv_layers":3,
          "num_dense_layers":1,
          "conv_booster":2,
          "linear_decay":1.8}
     }
}
```

The optimizer and the loss are instantiated directly from torch

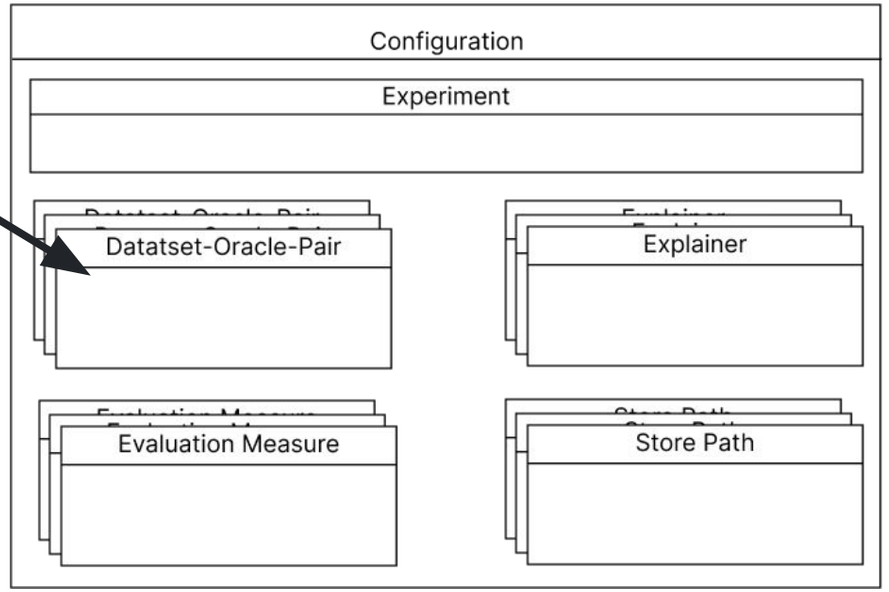
Oracle (Real)

```
"oracle": {
  "class": "src.oracle.nn.torch.OracleTorch",
  "parameters": {
    "epochs": 200,
    "batch_size": 32,
    "optimizer": {
      "class": "torch.optim.RMSprop",
      "parameters": { "lr":0.01}
    },
    "loss_fn": {
      "class": "torch.nn.CrossEntropyLoss",
      "parameters": { "reduction": "mean" }
     },
     "model": {
      "class": "src.oracle.nn.gcn.DownstreamGCN",
      "parameters":{
          "num_conv_layers":3,
          "num_dense_layers":1,
          "conv_booster":2,
          "linear_decay":1.8}
     }
}
```

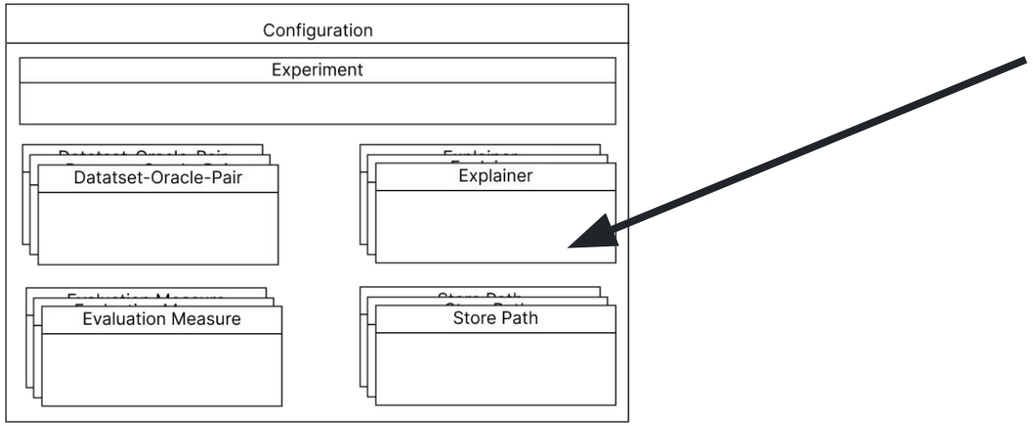
GRETEL comes with built-in standard GCNs that are customizable directly from the configuration file.

DO-PAIRS

```
"do-pairs": [
    {"dataset" : { ... }, "oracle": { ... }},
    .
    {"dataset" : { ... }, "oracle": { ... }},
],
```



EXPLAINERS




```
"explainers": [
    {"class": "src.explainer.heuristic.obs.ObliviousBidirectionalSearchExplainer",
    "parameters":{}},
    {"class": "src.explainer.search.i_rand.IRandExplainer", "parameters": {"p": 0.01, "t": 3}},
    {"class": "src.explainer.generative.cf2.CF2Explainer",
    "parameters":{"epochs": 50, "batch_size_ratio": 0.2, "lr" : 0.02, "alpha" : 0.7, "lam" : 20, \
          "gamma" : 0.9}
          },
    {"class": "src.explainer.generative.clear.CLEARExplainer",
    "parameters":{ "epochs": 100, "lr": 0.01, "lambda_cfe": 0.1, "alpha": 0.4, \
          "batch_size_ratio": 0.15 }
          },
]
```

How to add a new explainer

- All explainers in GRETEL extend either **Explainer** or **PerClassExplainer**.

- They must override the init function (with all the parameters from the JSON file) and the **explain** function, encapsulating the logic of the predictor.
- The explain function outputs a **GraphInstance** object, that is the result of the processing logic.

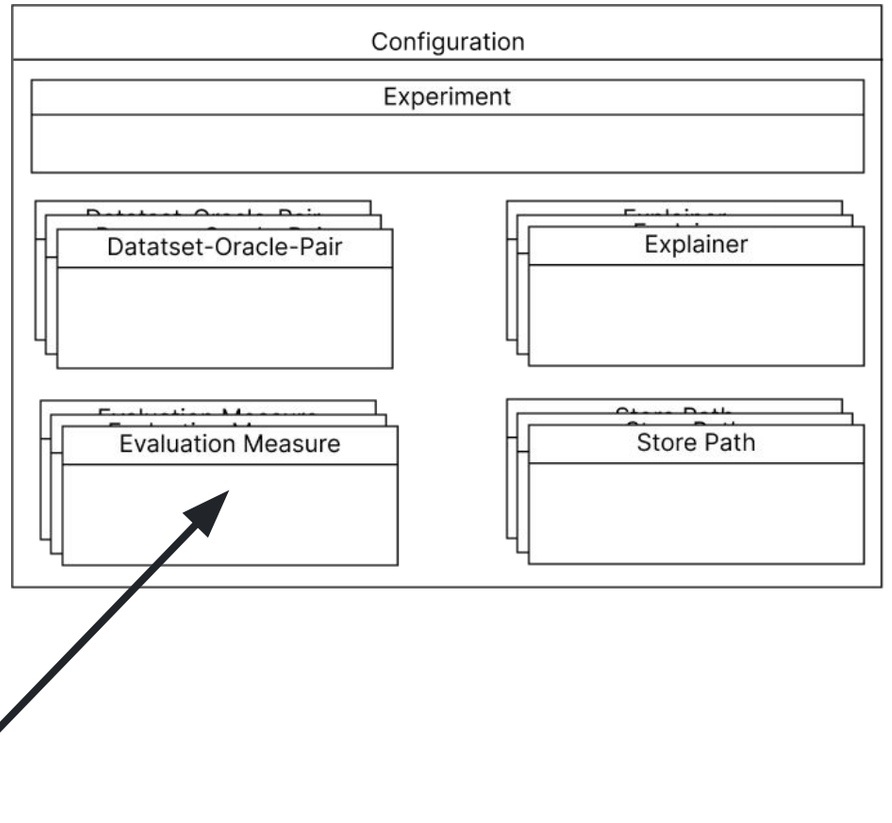
How to add a new explainer (2)

- Once the new Explainer class is ready, you just have to add it to the JSON configuration file as you would with any other explainer.

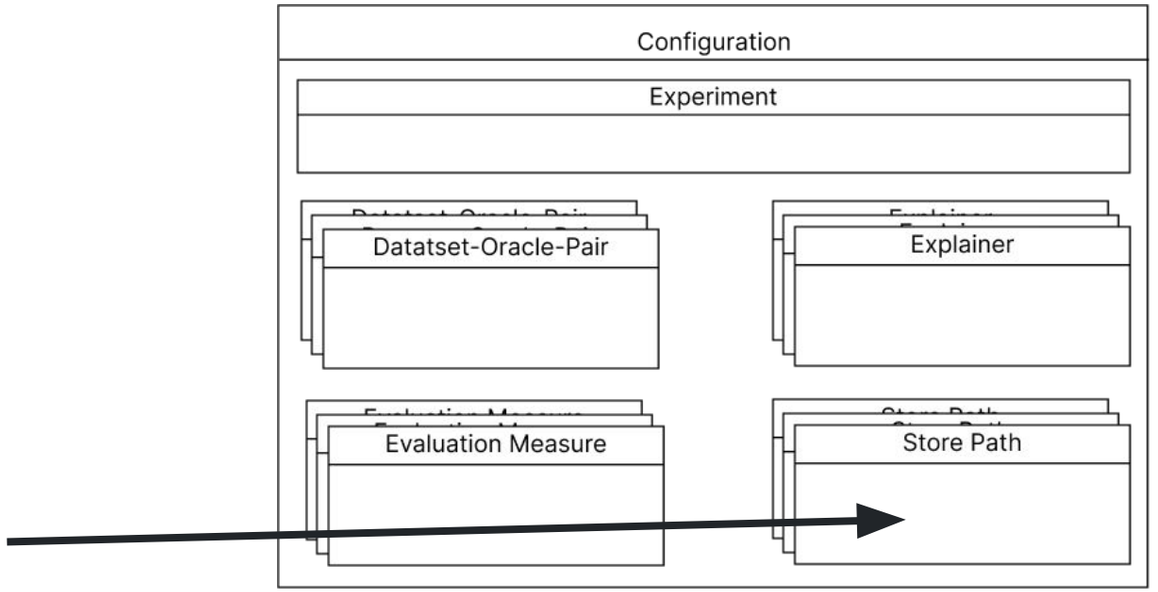
- Please find more info, and a step-by-step guide, on the Github's Wiki, or on the other sources at the end of this presentation.

Evaluations Measure

```
{"evaluation_metrics": [
    {"name": "runtime", "parameters": {}},
    {"name": "graph_edit_distance", "parameters": {}},
    {"name": "oracle_calls", "parameters": {}},
    {"name": "correctness", "parameters": {}},
    {"name": "sparsity", "parameters": {}},
    {"name": "fidelity", "parameters": {}},
    {"name": "oracle_accuracy", "parameters": {}}
  ]}
```



Paths & Cache




```
"store_paths": [
        {"name": "dataset_store_path", "address": "./data/cache/datasets/"},
        {"name": "oracle_store_path", "address": "./data/cache/oracles/"},
        {"name": "embedder_store_path", "address": "./data/cache/oracles/"},
        {"name": "explainer_store_path", "address": "./data/cache/explainers/"},
        {"name": "log_store_path", "address": "./output/logs/"},
        {"name": "output_store_path", "address": "./output/results/"}
    ]
```

COMPOSE (MECHANISM)

```
"compose_man": "config/snippets/datasets/centr_and_weights.json"
```

Will be replaced by the corresponding file:

```
{
    "manipulators": [
    { "class": "src.dataset.manipulators.centralities.NodeCentrality",
      "parameters": {} },
    { "class": "src.dataset.manipulators.weights.EdgeWeights",
      "parameters": {} }
    ]
 }
```

Configuration w Compose & Propagate

```
{
  "experiment": {
    "scope": "examples_configs",
    "parameters": {
        lock_release_tout":120,
        "propagate":[
            {"in_sections" : ["explainers"], "params": {"fold_id": 0}},
            {"in_sections" : ["do-pairs/oracle"], "params": {"fold_id": -1}},
            {"in_sections": ["do-pairs/dataset"], "params": { "compose_man": \
                                                    "config/snippets/datasets/centr_and_weights.json" }
        ]
    }
  },

"do-pairs":[ {"compose_bbbp_svm" : "config/snippets/do-pairs/BBBP_SVM-MOL.json"} ],
"explainers": [{"class": "src.explainer.search.dces.DCESExplainer"}],
"compose_mes" : "config/snippets/default_metrics.json",
"compose_strs" : "config/snippets/default_store_paths.json"
 }
```

Nested Compose

BBBP_SVM-MOL.json:

```
{
    "dataset" : {"compose_gcn" : "config/snippets/datasets/BBBP.json"},
    "oracle": {
        "class": "src.oracle.tabulars.svm.SVMOracle",
        "parameters": {
            "embedder": {
                "class": "src.embedder.molecule.model.RDKFingerprintEmbedder",
                "parameters": {}
            },
            "model": {  "parameters": {} }
        }
    }
}
```

GitHub - Work Locally (AAAI branch)

https://github.com/aiim-research/GRETEL/tree/aaai

```
git clone -b aaai https://github.com/aiim-research/GRETEL.git
```

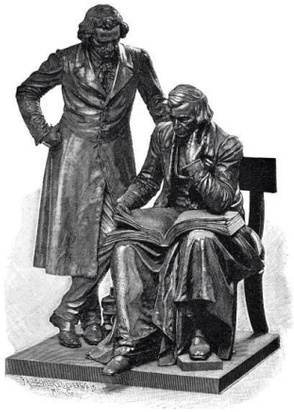
https://github.com/aiim-research/GRETEL/wiki#first-steps-with-gretel

Frameworks for Discrete Time Dynamic Graph Explanations

HANSEL WHO?

- Extension of GRETEL to Dynamic Graphs;
- Modular + Extensible;
- Config files remain unaltered;
- Adapts Static Explainers to Dynamic Scenario;

And the loop is almost closed; we just need the ***Witch and the cake house*** to appear in a framework now...

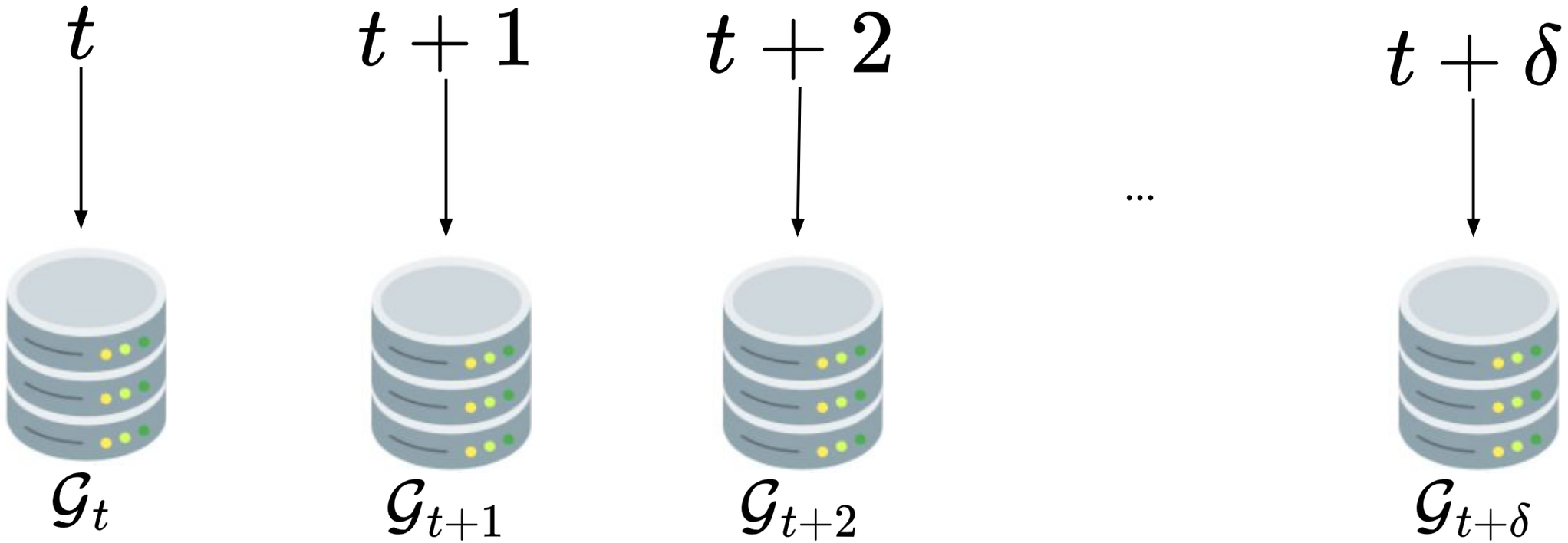


https://github.com/bardhprenkaj/HANSEL (v1)

KDD'24

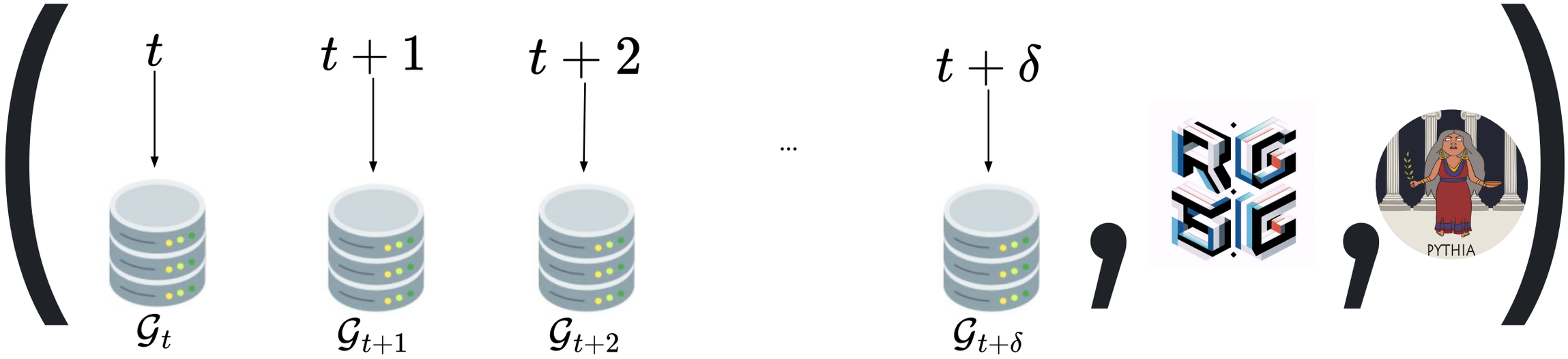
What are the Changes?

- The Dataset is now a Python Collection (aka **dict[str, Dataset]**)



What are the Changes?

- The Evaluator is now a Python Collection (aka **List[Tuple[dict[str, Dataset], Oracle, Explainer]]**)



What's Next?

Questions We Had in our last Tutorial

Research Question (AAAI'24)	Answer	Support
Are generative counterfactual explainers worth it?	Yes	You can generate plausible yet new counterfactuals, maintaining privacy (see GIST, RSGG-CE)
What's the difference between counterfactual explanations and adversarial attacks?	None (mathematically)	They optimize the same underlying mathematical function; however, there are works that study non-adversarial counterfactual explanations (see Leemann et al 2024)
How sure are we about the generated counterfactuals? Can we incorporate uncertainty in them?	Not quite	GRACIE tried to incorporate built-in latent space uncertainty with ball-like space where the center is certain and circumference is uncertain to "hopefully produce" certain counterfactuals
Are the produced counterfactuals actionable?	Somehow	One can use GED to see the actionability in graphs. However, we still can't measure the realness of this actionability.
How can we incorporate domain knowledge into the explanation methods?	We don't know	Nobody is addressing this to the best of our knowledge.
Instead of moving forward to produce counterfactuals, can we overshoot and then go backwards?	Yes (but not minimal counterfactual)	GIST does exactly this. It uses graph style transfer to go backwards to the original instance and produces a semi-minimal counterfactual.
How to ensure our explanations methods are stable and robust, producing similar explanations for similar instances?	We have a proposal	Given two instances G_1 and G_2, one can use overshoot to the same graph G^{ε}, and then use style transfer to go backwards. If G_1 and G_2 are similar, their counterfactuals should be similar. **Maybe next paper?**

Open Research Questions

- **RQ1:** Talk-like-a-graph is here [1]. **Can we use text to represent a graph and then generate text counterfactuals and then go back to a graph repr?**
  - *This could maybe address the semanticity and actionability aspect of counterfactuals.*
- **RQ2:** How can we incorporate **domain knowledge** into the explanation methods?
- **RQ3**: **How uncertain** must the oracle be such that the produced **counterfactual is adversarial**?

[1] Fatemi B, Halcrow J, Perozzi B. Talk like a graph: Encoding graphs for large language models. arXiv preprint arXiv:2310.04560. 2023 Oct 6.

Thanks for your attention!

QUESTIONS?

Bardh Prenkaj
bardh.prenkaj@tum.de

Efstratios Zaradoukas
efstratios.zaradoukas@tum.de

Mario A. Prado-Romero
marioalfonso.prado@gransassotech.org

Andrea D'Angelo
andrea.dangelo6@graduate.univaq.it

Giovanni Stilo
gstilo@luiss.it