Graphs' Counterfactual Explainability Landscape: current state and frontiers

giovanni stilo

(mario prado-romero, bardh prenkaj)

giovanni.stilo@univaq.it

aiim
aiim.disim.univaq.it

XAI.it
Italian Workshop on Explainable Artificial Intelligence

SOBIGDATA.it
ITALIAN RESEARCH INFRASTRUCTURE

Part One
Background

Explainability and its importance

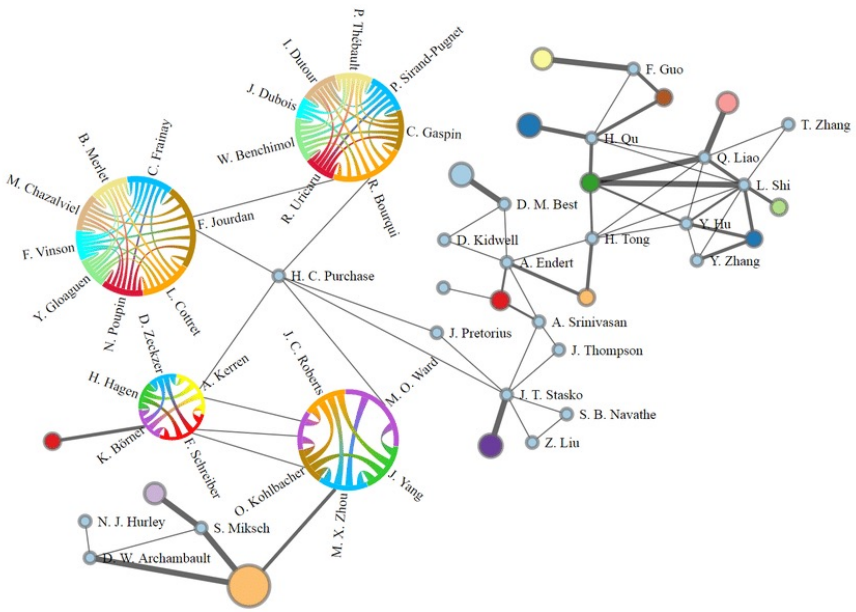
- Explainability is **crucial** in **sensitive** domains
  - It helps users and service providers make informed and reliable decisions [1]
- DNNs suffer from the **black-box** problem [2]
  - Can **not** be **used** in **finance** nor **healthcare** domains (critical domains)
  - White-boxes are preferred for decision-making purposes [3]
- Black-boxes are **more performant** than white-boxes [4-7]

[1] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) 51(5), 1–42 (2018)
[2] Petch, J., Di, S., Nelson, W.: Opening the black box: the promise and limitations of explainable machine learning in cardiology. Canadian Journal of Cardiology (2021)
[3] Verenich, I., Dumas, M., La Rosa, M., Nguyen, H.: Predicting process performance: A white-box approach based on process models. Journal of Software: Evolution and Process 31(6), e2170 (2019)
[4] Aragona, D., Podo, L., Prenkaj, B., Velardi, P.: Coronna: a deep sequential framework to predict epidemic spread. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. pp. 10–17 (2021)
[5] Feng, W., Tang, J., Liu, T.X.: Understanding dropouts in moocs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 517–524 (2019)
[6] Verma, H., Mandal, S., Gupta, A.: Temporal deep learning architecture for prediction of covid-19 cases in india. Expert Systems with Applications 195, 116611 (2022)
[7] Prenkaj, B., Distante, D., Faralli, S., Velardi, P.: Hidden space deep sequential risk prediction on student trajectories. Future Generation Computer Systems 125, 532–543 (2021)

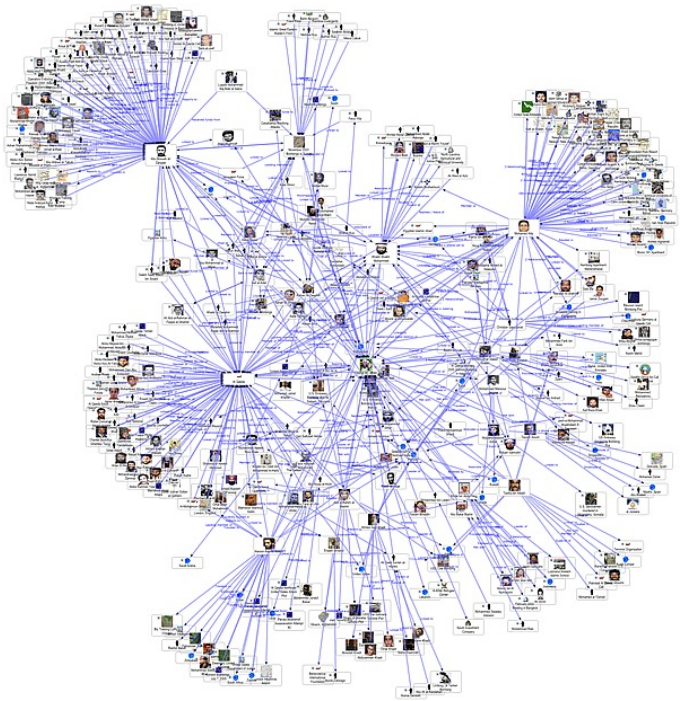
What's a graph?

A **graph** $G = (V, E)$ consists of:

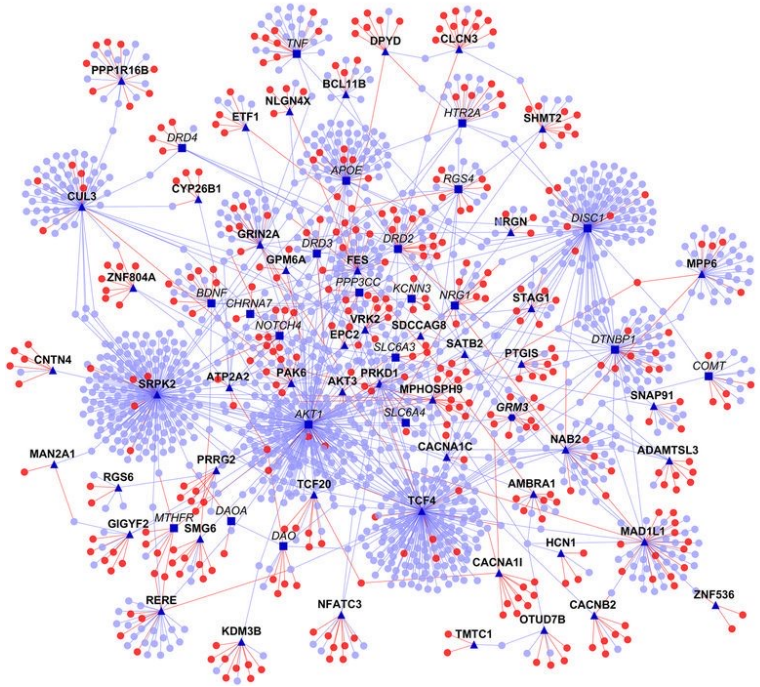
a **nodes set** $V = \{v_1, \dots, v_n\}$ and an **edges set** $E = \{(v_i, v_j) | v_i, v_j \in V\}$



DBLP co-authorship



Social network



Schizophrenia PPI

Challenges

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

1. their **topology is variable**:
   Thus it is hard to design a Neural Network that both **sufficiently expressive** and can cope with this **variation**;

2. graphs may be **enormous**:
   a graph representing connections between users of a social network might have a **millions nodes** and **billion of edges**;

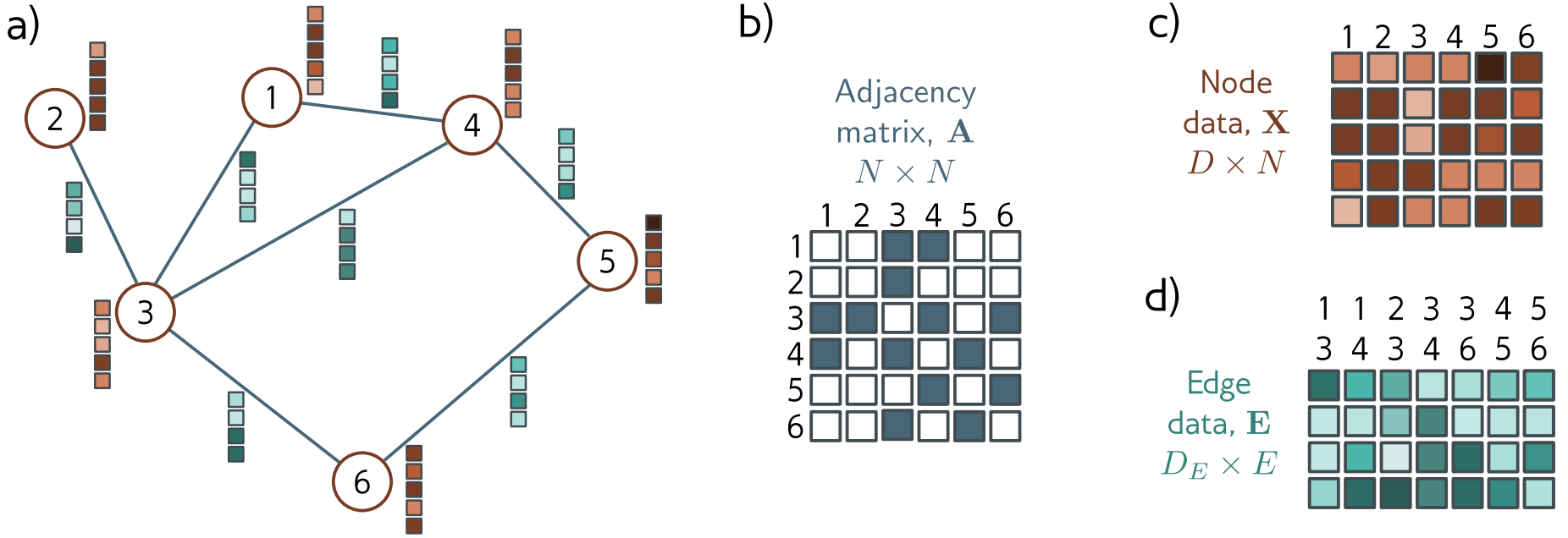
3. there may only be a **single monolithic graph** available:
   so the **usual protocol** of training with many data examples and testing with new data is **not always appropriate or possible**.

Graph Representation (learning)

More formally, a graph consists of **a set of N** nodes connected by **a set of E** edges.
The graph can be **encoded** by **three matrices**:

- **adjacency matrix**, $\boldsymbol{A}$ is an $N \times N$ matrix where entry (m, n) is set to one if there is an edge between nodes m and n and zero otherwise (symmetric for undirected graphs).
- **nodes embeddings** $\boldsymbol{X}$ is an $D \times N$ matrix where n^{th} node has an associated node embedding $\boldsymbol{X}^n$ of length D.
- **edges embeddings** $\boldsymbol{E}$ is an $D_E \times N$ matrix where e^{th} edge has an associated edge embedding $\boldsymbol{E}^e$ of length D_E.

We want to learn a (dense) representation $\boldsymbol{H}$ of the Graph usable for different downstream tasks

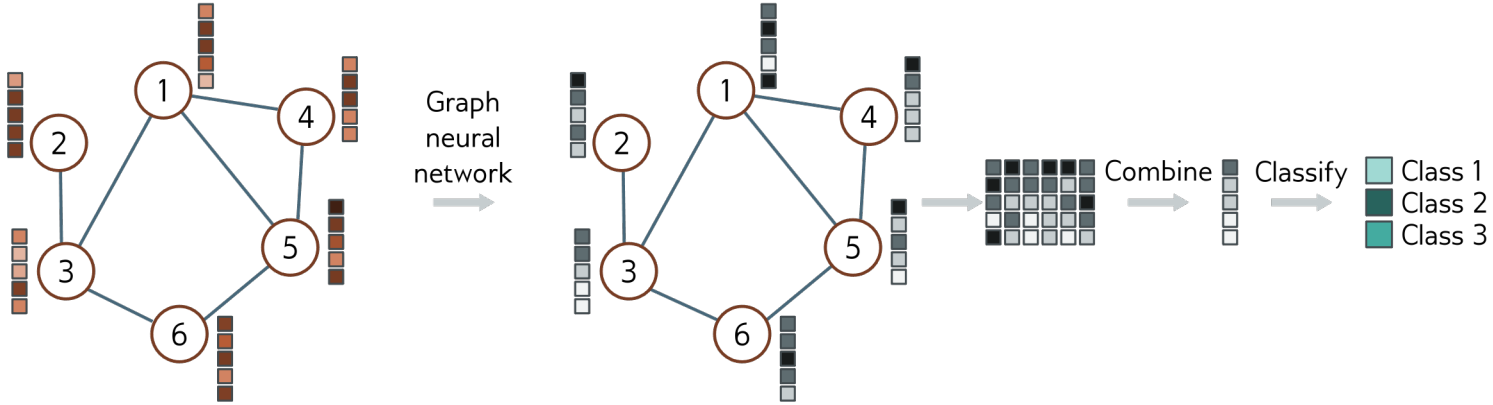


Graph-level Tasks

For example, we might 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**.*



- for regression, the mismatch is computed using the **least squares loss**.
- for **binary classification**, the output is passed through a sigmoid function, and the mismatch is calculated using the **binary cross-entropy loss**.

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

where the scalar β_k and $1 \times D$ vector $\boldsymbol{\omega}_k$ are learned parameters.

$\boldsymbol{H}_k \frac{\mathbf{1}}{N}$ has the effect of summing together all the embeddings and subsequently dividing by N.

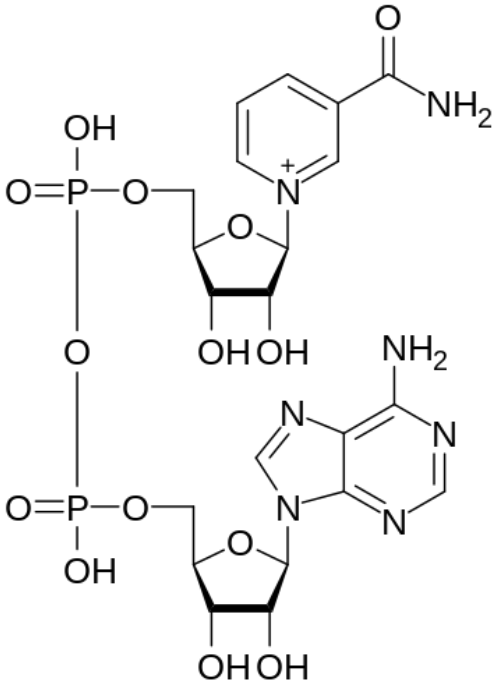
GNN

A **graph neural network** is a model that takes the node embeddings $\boldsymbol{X}$ and the adjacency matrix $\boldsymbol{A}$ as inputs and passes them through a series of $\boldsymbol{K}$ layers. The node embeddings are updated at each layer to create **intermediate "hidden"** representations $\boldsymbol{h}_{k \neq K}$ before finally computing output embeddings $\boldsymbol{h}_K$.

- At the beginning each column of the input node embeddings $\boldsymbol{X}$ just contains information about the **node itself**.
- At the end, each column of the model output $\boldsymbol{h}_K$ includes information about the **node and its context within the graph.**
- Typical GNN mechanisms are:
  - (random) walk based;
  - Message passing;
  - GCN;

Graph Classification e.g.

We want a **neural network** $\mathrm{f}[\boldsymbol{X}, \boldsymbol{A}, \Phi]$ that **classifies** (predicts) **molecules** as toxic.



The **adjacency matrix** $A \in \mathbb{R}^{N \times N}$ derives from the molecular structure. The columns of the **node embedding** matrix $X \in \mathbb{R}^{118 \times N}$ are **one-hot vectors** indicating which of the 118 **elements of the periodic** table are present.

Note that the input **node embeddings** can be **transformed** to an **arbitrary** size D by the **first** weight **matrix** $\boldsymbol{\Omega}_k \in \mathbb{R}^{D \times 118}$.

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

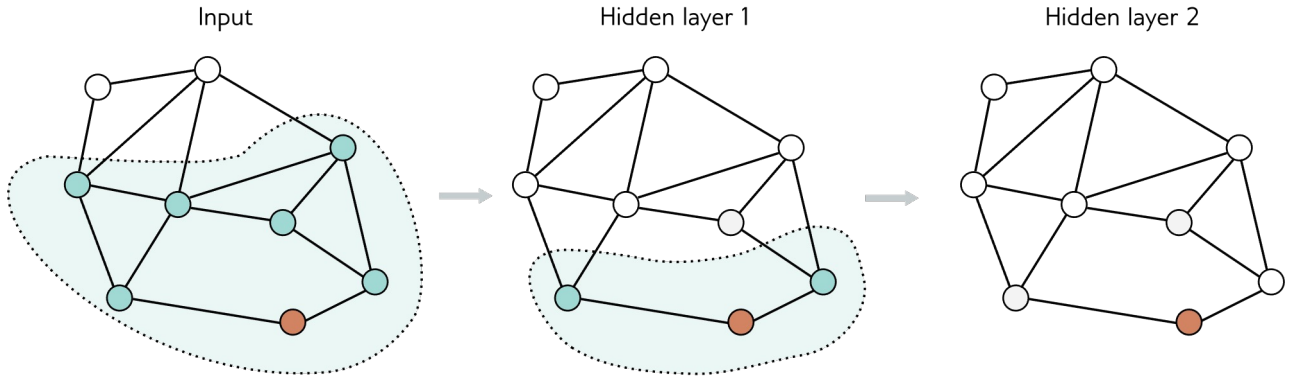
As intuition: $\boldsymbol{H}_k$ depends by $\boldsymbol{X}(\boldsymbol{A} + \boldsymbol{I})^k$, e.g $\boldsymbol{H}_3$: $\boldsymbol{X}(\boldsymbol{A} + \boldsymbol{I})^3 = \boldsymbol{X}(\boldsymbol{A}^3 + 3\boldsymbol{A}^2 + 3\boldsymbol{A} + \boldsymbol{I}^3)$

Batches and Receptive Field

One way to form a batch is to **choose a random subset** of **labeled nodes** at each training step. Each node depends on its neighbors in the previous layer and so on. Do you remember:

$$\boldsymbol{H}_k \text{ depends by } \boldsymbol{X}(\boldsymbol{A}+\boldsymbol{I})^k$$

In a GCN the size of the **receptive field** is equivalent to the *k-hop neighborhood*.

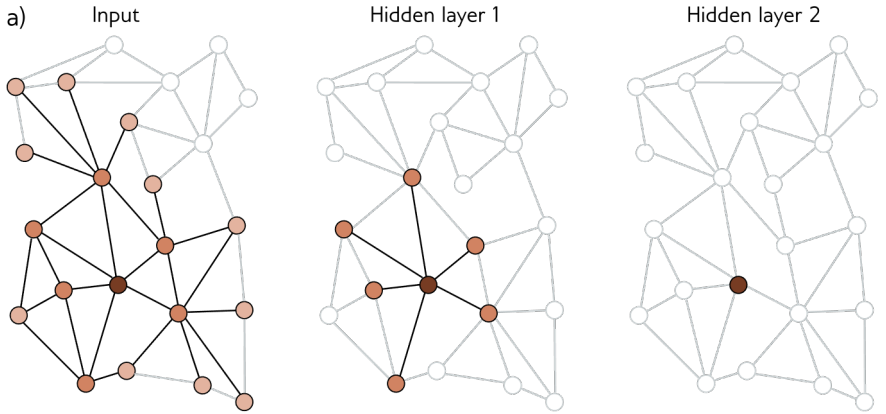


We can hence perform a gradient descent step using the **graph** that forms the **union of the k-hop neighborhoods** of the **batch nodes**; the remaining inputs do not contribute.

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 $k << diam(G)$



Counterfactual example

- We have a **social network** of different users
- User **U posts** an **illicit** advertisement that **violates** the **Terms of Services** (e.g., selling illegal goods online)
- A **counterfactual** explanation **of** U's **account suspension** would be:

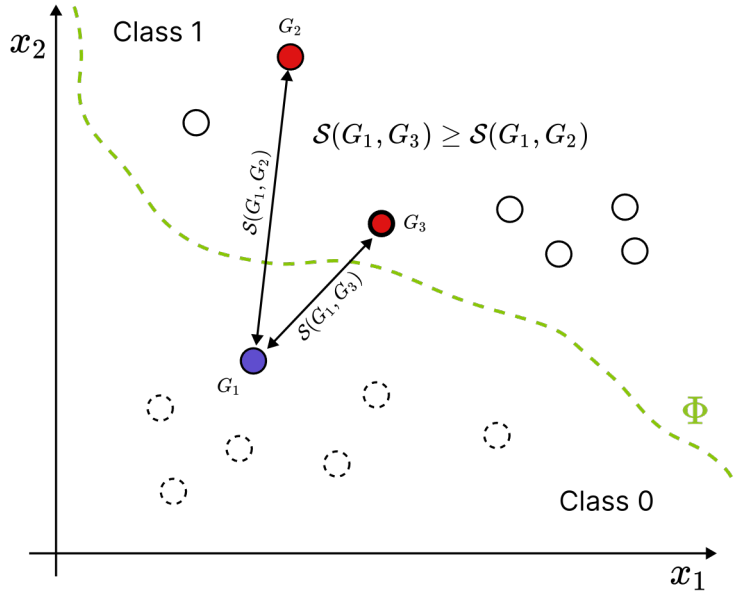
"If the user had not violated the TOS, their account would not have been banned.

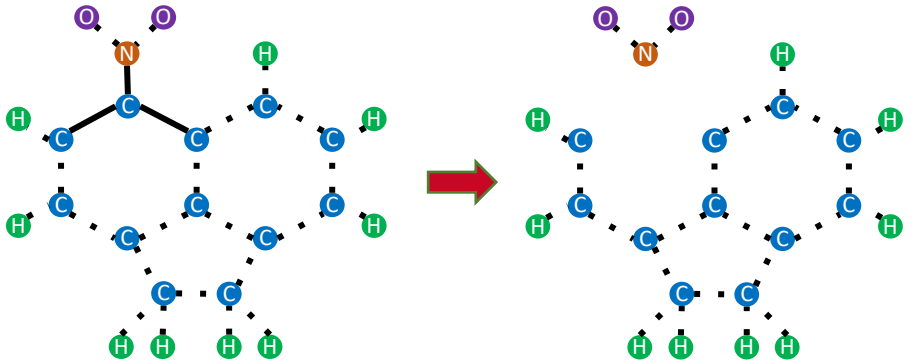
Graph Counterfactual Explainability (GCE)

- Given:
  - A **predictor** (oracle) $\Phi: \mathcal{G} \rightarrow \{0,1\}$ that we want to explain
  - A **graph** G g or a set of graphs

$$\mathcal{G} = \{G_1, \dots, G_n\}$$

  - *A similarity **function** $\mathcal{S}: \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$*

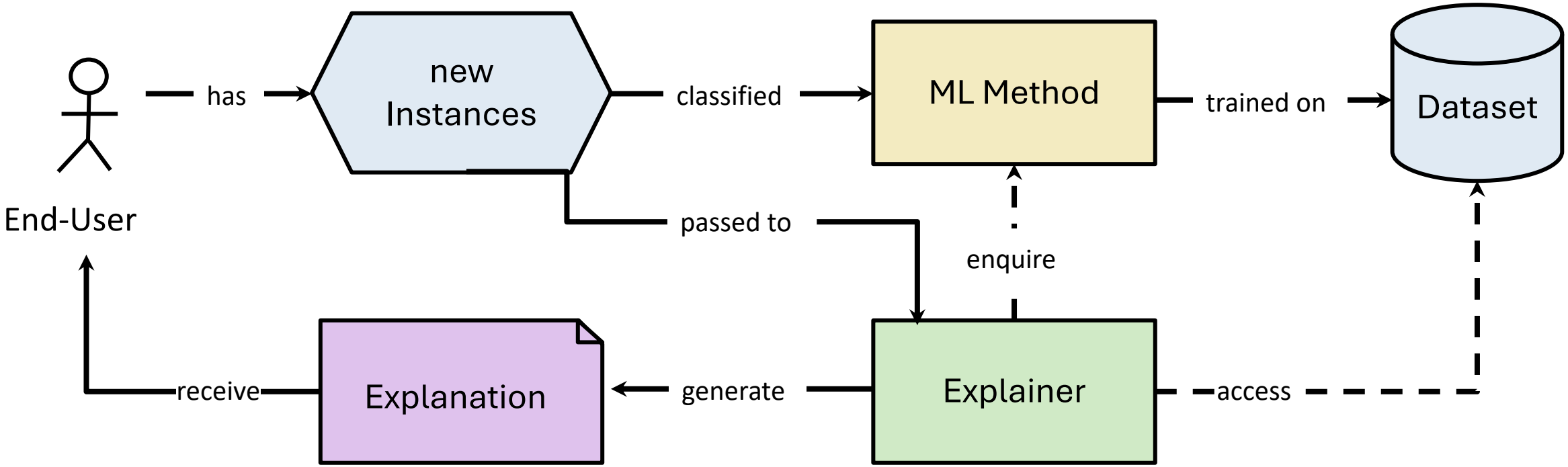




- We want to find a **counterfactual** G'

such that $\mathcal{E}_\Phi(G) = arg \max_{G' \in \mathcal{G}', G \neq G', \Phi(G) \neq \Phi(G')} \mathcal{S}(G, G')$

Typical XAI Workflow

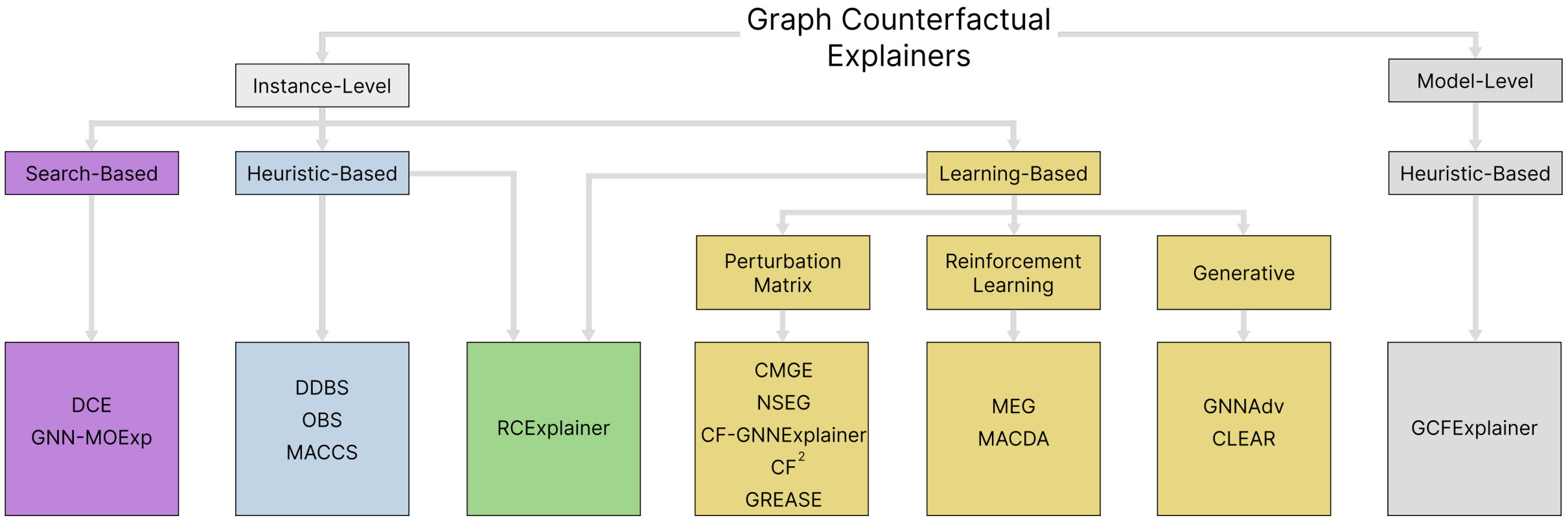


Part Two
SoTA

Based on

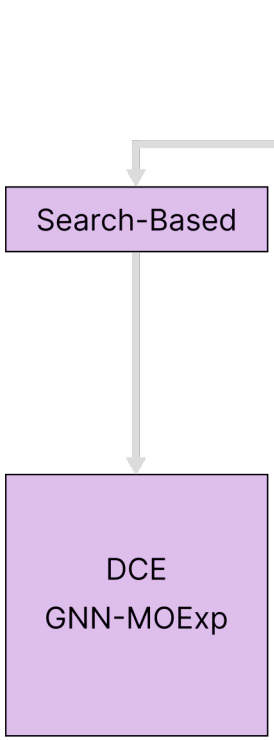
Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. 2023.
A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges.
ACM Compututer Survey (September 2023). https://doi.org/10.1145/3618105

GCE methods Taxonomy



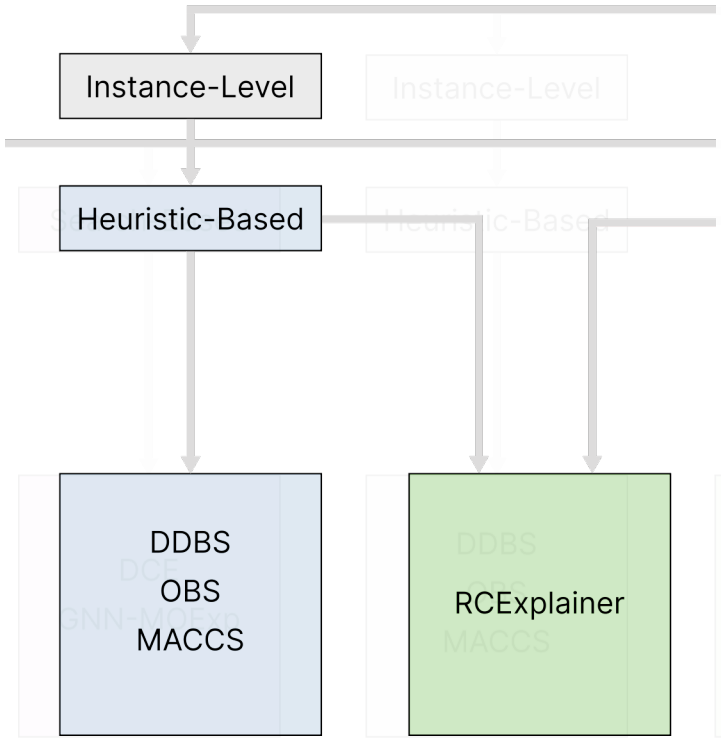
[8] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. 2023. A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges. ACM Comput. Surv. (September 2023). https://doi.org/10.1145/3618105

GCE Search-Based



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

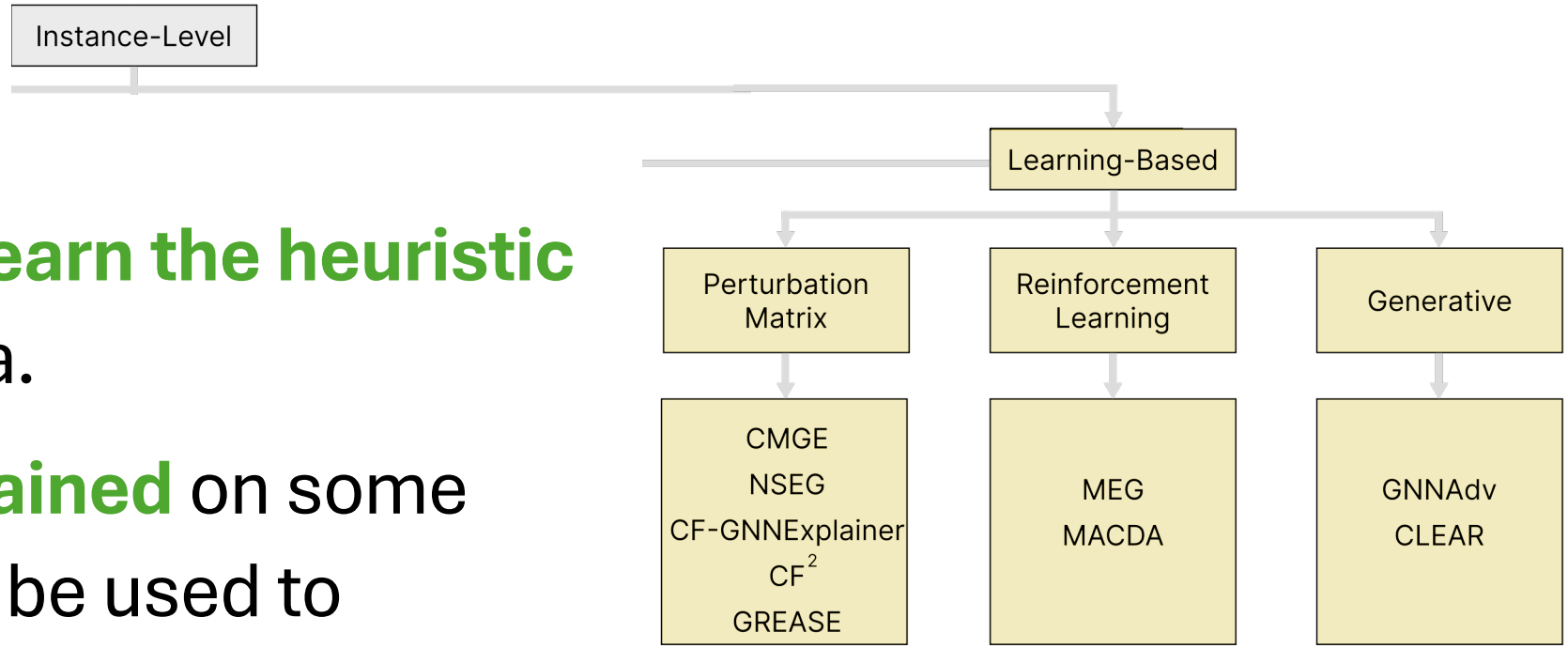
GCE Heuristic-Based



- **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**

GCE Learning-Based

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



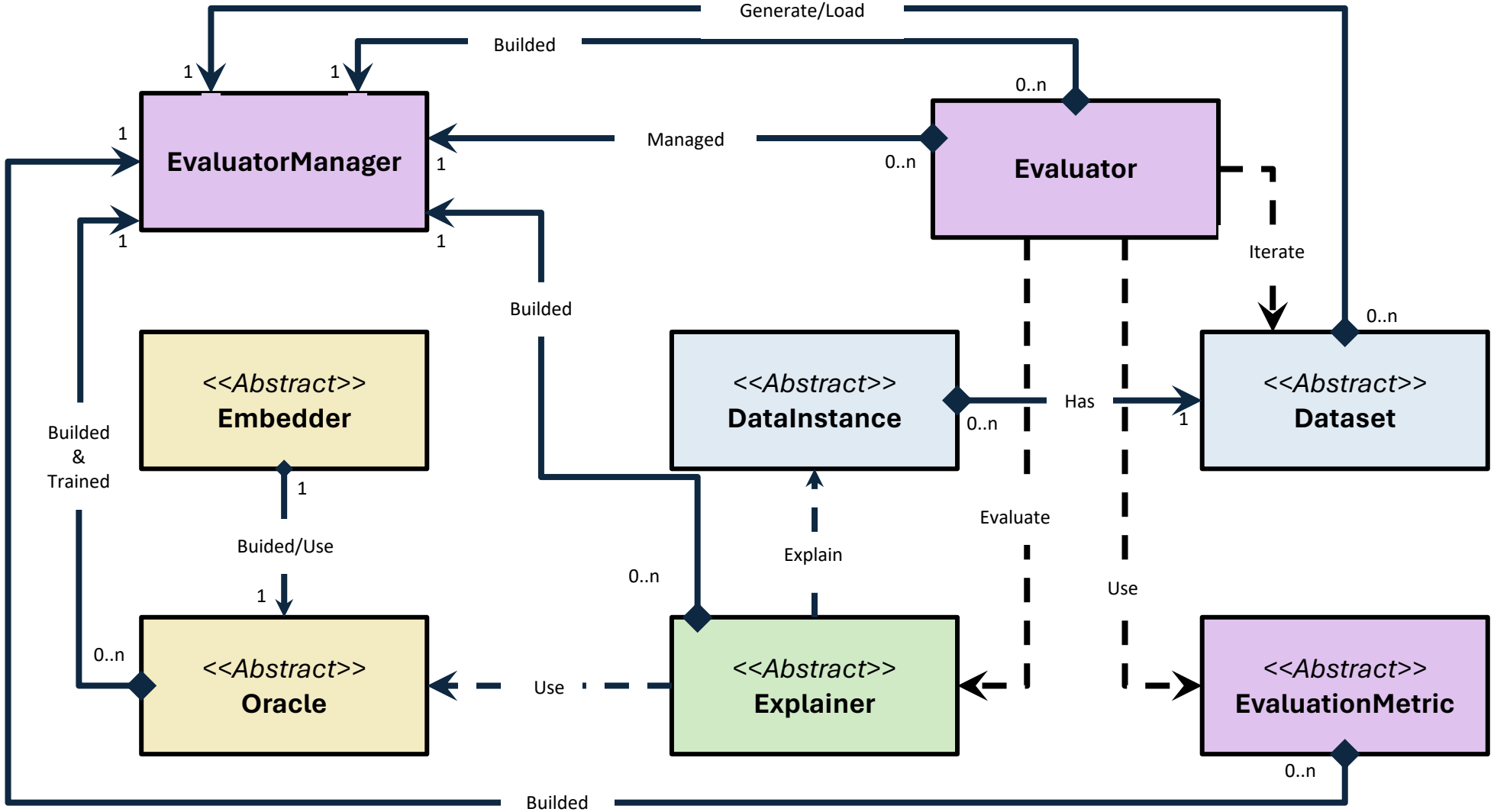
Comparison between GCE Methods

Method	Model Agnosticism	Model Access	Factual-Based Explanations	Minimal CE	Domain Agnosticism	Training Data Accessibility	Explanation Level	Classification Task	Generation Type	Approach
DDBS [1]	✓	·	·	✓	✓	✓	Instance	G	$E(+,-)$	Heuristic
OBS [1]	✓	·	·	✓	✓	·	Instance	G	$E(+,-)$	Heuristic
RCExplainer [4]	✓	✓	✓	·	✓	✓	Instance	G, V	$E(-)$	Heuristic & Learning
GNN-MOExp [35]	✓	·	✓	·	✓	·	Instance	V	sub-graph	Search
MEG [52]	✓	✓	·	✓	·	~	Instance	G	$E(+,-), V(+,-)$	Learning
GNNAdv [67]	✓	✓	·	✓	✓	·	Instance	G	$E(+,-)$	Learning
CMGE [80]	·	✓	✓	·	·	✓	Instance	G	$E(+,-), V(-)$	Learning
NSEG [12]	✓	✓	✓	·	✓	·	Instance	G, V	$E(-), F(*)$	Learning
CF-GNNExplainer [38]	✓	✓	·	✓	✓	·	Instance	V	$E(-)$	Learning
CLEAR [40]	✓	·	·	✓	✓	✓	Instance	G	$E(+,-), F(*)$	Learning
MACDA [49]	✓	·	·	✓	·	~	Instance	(G_1, G_2)	$E(+,-), V(+,-)$	Learning
CF^2 [68]	✓	·	✓	·	✓	·	Instance	G, V	$E(-), V(-), F(-)$	Learning
MACCS [77]	✓	·	·	✓	·	·	Instance	G	$E(+,-), V(+,-)$	Heuristic
GCFExplainer [27]	✓	·	·	·	✓	·	Model	G	$E(+,-), V(+,-)$	Heuristic

RETEL

Framework

GRETEL modules interaction



Empirical Explainers comparison

Dataset	Method	Runtime ↓	GED ↓	Oracle Calls ↓	Correctness ↑	Sparsity ↓	Fidelity ↑	Oracle Accuracy ↑
Tree-Cycles	RAND@5	**0.01 ± 0.003**	92.18 ± 5.44	0.00 ± 0.00	0.55 ± 0.50	1.45 ± 0.09	0.55 ± 0.50	1.00 ± 0.00
	RAND@10	0.02 ± 0.006	123.74 ± 7.43	0.00 ± 0.00	0.51 ± 0.50	1.94 ± 0.12	0.51 ± 0.50	1.00 ± 0.00
	RAND@15	0.01 ± 0.004	147.93 ± 8.26	0.00 ± 0.00	0.58 ± 0.50	2.33 ± 0.13	0.58 ± 0.50	1.00 ± 0.00
	DCE	0.13 ± 0.00	50.36 ± 0.00	501.00 ± 0.00	**1.00 ± 0.00**	0.79 ± 0.00	**1.00 ± 0.00**	1.00 ± 0.00
	OBS	0.07 ± 0.01	57.31 ± 0.03	149.45 ± 21.13	0.96 ± 0.01	0.90 ± 0.00	0.96 ± 0.01	1.00 ± 0.00
	DDBS	8.87 ± 0.10	71.79 ± 0.24	1342.62 ± 11.95	0.59 ± 0.01	1.13 ± 0.00	0.59 ± 0.01	1.00 ± 0.00
	MACCS	–	–	–	–	–	–	–
	CLEAR	2.47 ± 0.08	79.76 ± 3.60	0.00 ± 0.00	0.53 ± 0.10	1.26 ± 0.06	0.53 ± 0.10	1.00 ± 0.00
	CF^2	0.41 ± 0.01	**31.54 ± 0.12**	0.00 ± 0.00	0.47 ± 0.10	**0.50 ± 0.00**	0.47 ± 0.10	1.00 ± 0.00
	MEG	272.11 ± 5.66	159.70 ± 1.34	0.00 ± 0.00	0.53 ± 0.00	2.51 ± 0.02	0.53 ± 0.00	1.00 ± 0.00
ASD	RAND@5	1.45 ± 0.46	618.06 ± 8.27	0.00 ± 0.00	0.00 ± 0.00	0.80 ± 0.01	0.00 ± 0.00	0.79 ± 0.08
	RAND@10	2.76 ± 1.25	1152.93 ± 20.19	0.00 ± 0.00	0.00 ± 0.00	1.49 ± 0.02	0.00 ± 0.00	0.79 ± 0.08
	RAND@15	1.33 ± 0.39	1600.78 ± 18.22	0.00 ± 0.00	0.00 ± 0.00	2.08 ± 0.03	0.00 ± 0.00	0.79 ± 0.08
	DCE	**0.09 ± 0.02**	1011.69 ± 0.00	102.00 ± 0.00	**1.00 ± 0.00**	1.31 ± 0.00	**0.54 ± 0.00**	0.79 ± 0.08
	OBS	3.24 ± 1.13	**9.89 ± 0.11**	347.73 ± 15.11	**1.00 ± 0.00**	**0.01 ± 0.00**	**0.54 ± 0.00**	0.79 ± 0.08
	DDBS	83.46 ± 34.04	11.79 ± 0.29	362.05 ± 14.56	**1.00 ± 0.00**	0.02 ± 0.00	**0.54 ± 0.00**	0.79 ± 0.08
	MACCS	–	–	–	–	–	–	–
	CLEAR	0.45 ± 0.04	1739.60 ± 131.16	0.00 ± 0.00	0.47 ± 0.13	2.25 ± 0.17	0.25 ± 0.18	0.79 ± 0.08
	CF^2	0.69 ± 0.01	655.49 ± 2.87	0.00 ± 0.00	0.46 ± 0.09	0.85 ± 0.00	0.37 ± 0.15	0.79 ± 0.08
	MEG	×	×	×	×	×	×	×
BBBP	RAND@5	**0.01 ± 0.03**	30.98 ± 33.27	0.00 ± 0.00	0.85 ± 0.35	0.52 ± 0.23	0.62 ± 0.69	0.86 ± 0.02
	RAND@10	**0.01 ± 0.03**	52.98 ± 58.96	0.00 ± 0.00	0.86 ± 0.35	0.93 ± 0.41	0.65 ± 0.66	0.86 ± 0.02
	RAND@15	0.02 ± 0.12	82.97 ± 137.37	0.00 ± 0.00	0.85 ± 0.36	1.32 ± 0.70	0.61 ± 0.69	0.86 ± 0.02
	DCE	37.51 ± 5.21	27.92 ± 0.12	2040.00 ± 0.00	**1.00 ± 0.00**	0.59 ± 0.00	**0.72 ± 0.00**	0.86 ± 0.02
	OBS	2.92 ± 0.07	0.00 ± 0.00	314.61 ± 0.00	0.00 ± 0.00	**0.00 ± 0.00**	0.61 ± 0.00	0.86 ± 0.02
	DDBS	×	×	×	×	×	×	×
	MACCS	31.35 ± 0.97	**11.23 ± 0.08**	1221.33 ± 0.22	0.40 ± 0.00	0.19 ± 0.00	0.23 ± 0.00	0.86 ± 0.02
	CLEAR@1	213.21 ± 5.64	27056.29 ± 9.69	0.00 ± 0.00	0.87 ± 0.02	91.89 ± 0.18	0.64 ± 0.03	0.86 ± 0.02
	CLEAR@5	214.80 ± 6.97	26711.57 ± 112.67	0.00 ± 0.00	0.85 ± 0.02	90.71 ± 0.37	0.62 ± 0.03	0.86 ± 0.02
	CLEAR@15	251.93 ± 36.01	25986.66 ± 170.41	0.00 ± 0.00	0.85 ± 0.01	88.20 ± 0.66	0.62 ± 0.06	0.86 ± 0.02
	CF^2	84.41 ± 49.06	25.72 ± 0.63	0.00 ± 0.00	0.85 ± 0.02	0.09 ± 0.00	0.63 ± 0.03	0.86 ± 0.02
	MEG	90.66 ± 29.51	269.35 ± 0.39	0.00 ± 0.00	0.51 ± 0.04	0.91 ± 0.00	0.32 ± 0.04	0.86 ± 0.02

GRETEL v1

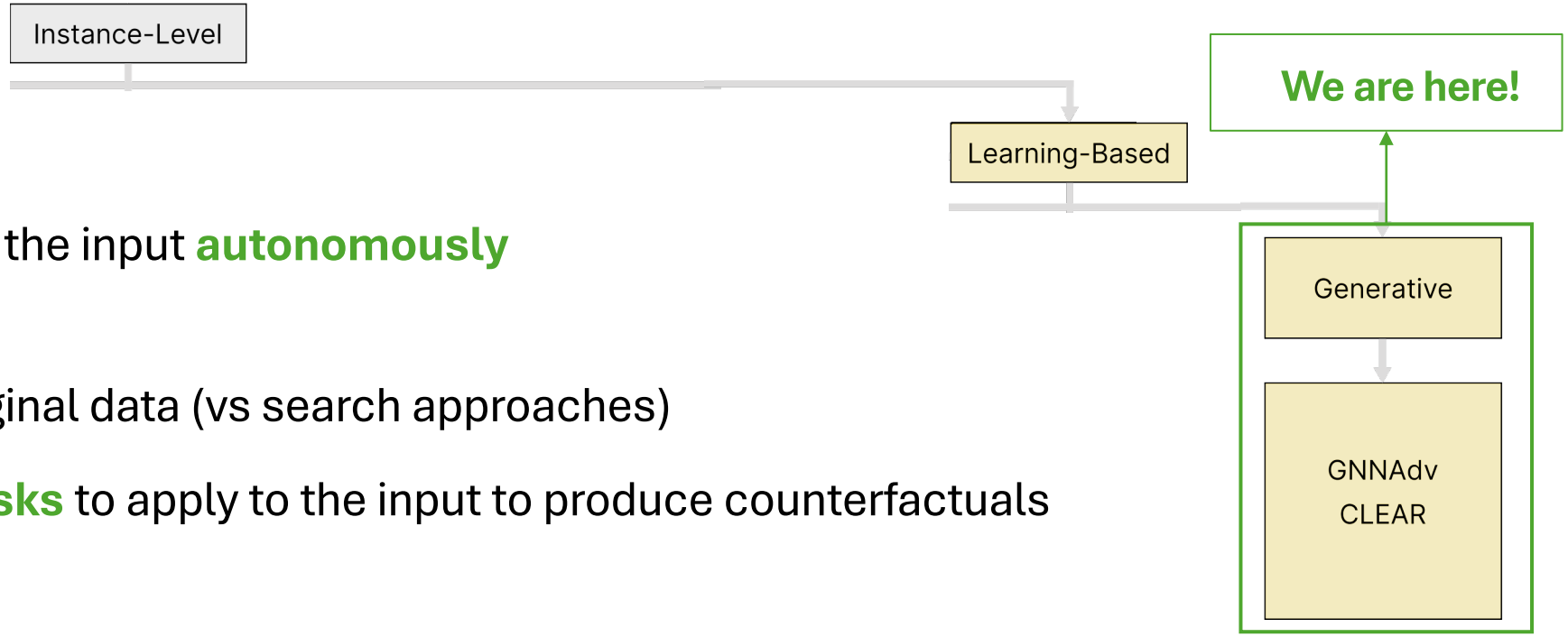
Demo

https://github.com/MarioTheOne/GRETEL

Part Three

G-CounterGAN

GCE Generative

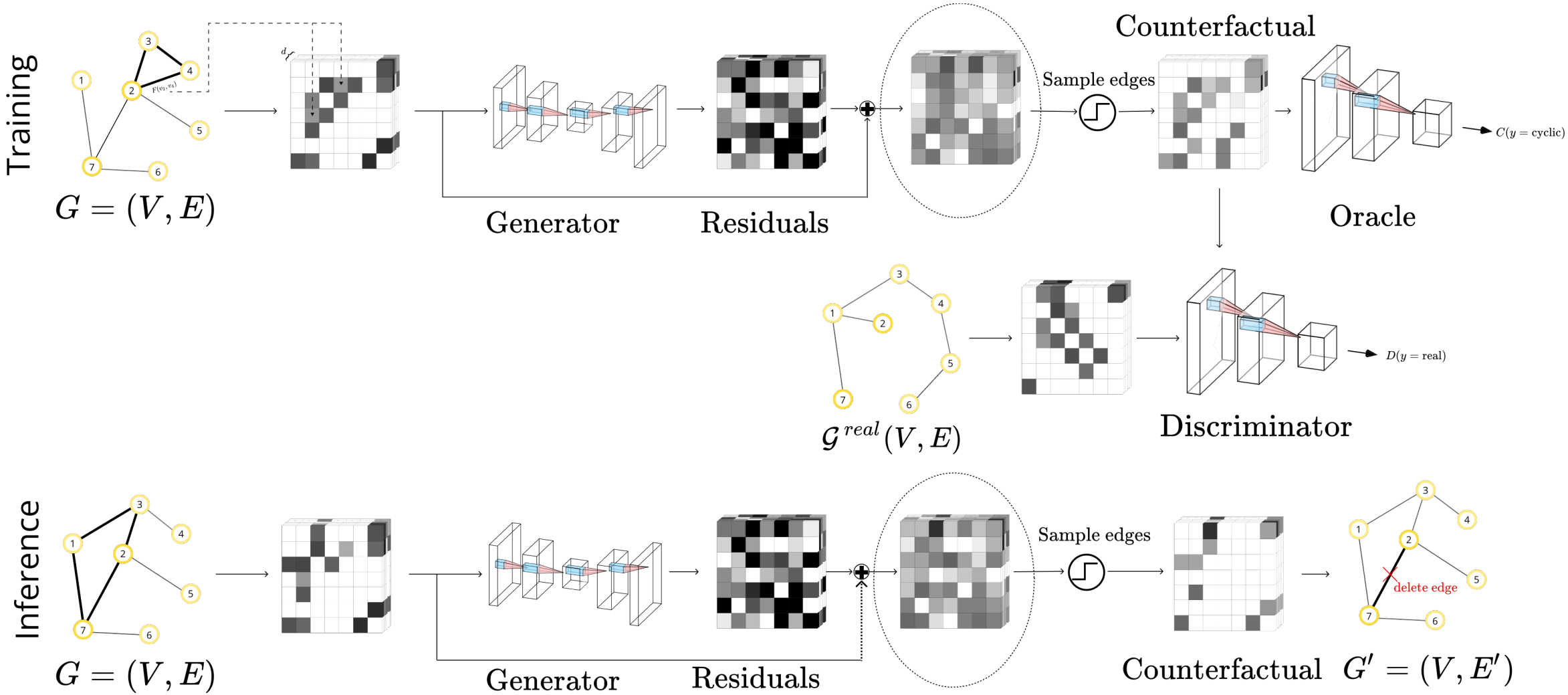


Generative approaches:

- **Learn the perturbation** of the input **autonomously** (vs heuristic approaches)
- **Aren't confined** to the original data (vs search approaches)
- **Don't rely on learned masks** to apply to the input to produce counterfactuals (vs learning approaches)

- Generative strategies **allow** to produce **multiple counterfactuals** from a **learned latent space**
- **don't need** to **access** the **oracle** at explanation time

Architecture overview



High Level Loss

$$\mathcal{L}_{\text{OUR}} = \mathcal{L}_1(G, D) + \mathcal{L}_\Phi(G, c) + Reg(G, A)$$

- c is the **class** to **explain**, and $Reg(G, A)$ is a **regularization** term that **controls** the sparsity of the residuals (i.e., feature perturbations)

Generator/Discriminator Loss

$$\mathcal{L}_{OUR} = \mathcal{L}_1(G, D) + \mathcal{L}_\Phi(G, c) + Reg(G, A)$$

$$\mathcal{L}_1(G, A) = E_{A \sim p_{data}} \log D\,(A)$$
$$+ E_{A \sim p_{data}} \log\Big(1 - D\big(A + G(A)\big)\Big)$$

- We modify the **generator** to **take original** input data **instead** of **noise** sampled from a normal distribution to **directly** generates **Counterfactuals** of the input instance

Embedding the Oracle (1)

$$\mathcal{L}_{OUR} = \mathcal{L}_1(G, D) + \mathcal{L}_\Phi(G, c) + Reg(G, A)$$

$$\mathcal{L}_\Phi(G, c) = E_{A \sim p_{data}} \log\left(\mathbb{I}\left[\Phi\left(A + G(A)\right) \neq c\right]\right)$$

- **Sampling instances** from the data distribution **could** make G **generate** **null** residuals
- Hence, **this loss** component **leverage** the **Oracle** to **steer** the **generator** away from this behavior, making it produce plausible counterfactuals

Embedding the Oracle (2)

$$\mathcal{L}_{OUR} = \mathcal{L}_1(G, D) + \mathcal{L}_{\Phi}(G, c) + Reg(G, A)$$

$$\mathcal{L}_{\Phi}(G, c) = E_{A \sim p_{\cdot}} \log\left(\mathbb{I}\left[\Phi\left(A + G(A)\right) \neq c\right]\right)$$

- **Problem:**

  The oracle is a black-box and we cannot access its gradients to optimize this loss!

Final Loss

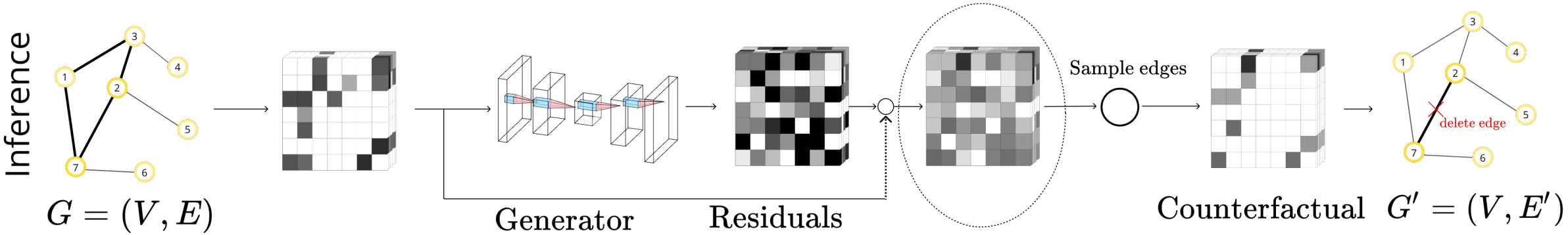
$$\mathcal{L}_{OUR} = \frac{\sum_{A \in \mathcal{A}} \left(1[\Phi(A) = c] \cdot \log D\,(A) \right)}{\sum_{A \in \mathcal{A}} \mathbb{I}[\Phi(A) = c]}$$

$$+ \frac{1}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \log\left(1 - D\left(A + G(A)\right)\right) + Reg(G, A)$$

- **Solution:**
- We **weight** the **first** term of the loss by the **prediction scores** of the oracle;
- $\mathbb{I}[\Phi(A) = c]$ is an **indicator function** that returns 1 if Φ classify the instance in the class c; while $\mathcal{A}$ is the set of all adjacency matrices corresponding to $G \in \mathcal{G}$

G-CounteRGAN Inference

- Counterfactuals **generation** are made by **sampling** edges by the edge probabilities **learned** in the **latent space**;
- We **keep** the **node order** to avoid CLEAR's graph matching problem (NP-hard)



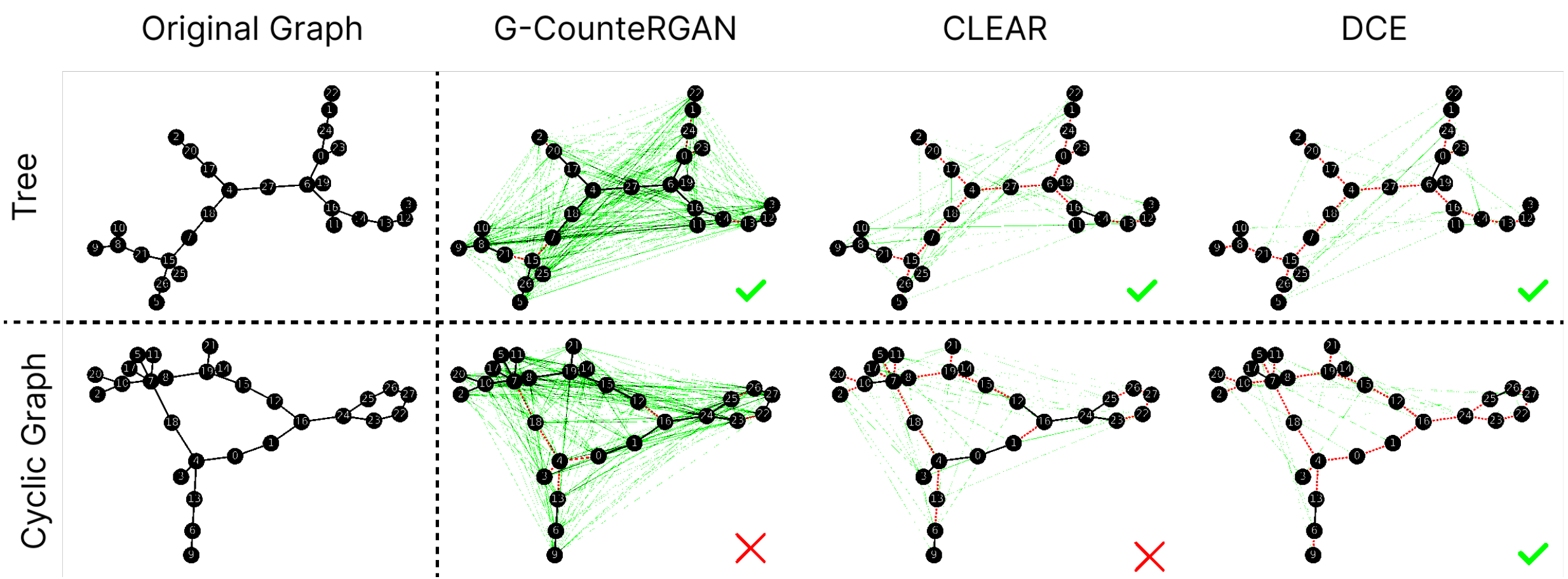
[9] Ma, J., Guo, R., Mishra, S., Zhang, A., Li, J.: CLEAR: generative counterfactual explanations on graphs. In: NeurIPS (2022)

Performances on TreeCycles@n

Dataset	Method	Runtime ↓	GED ↓	Oracle calls ↓	Correctness ↑	Sparsity ↓	Fidelity ↑	Oracle accuracy ↑
Tree-Cycles@28	DCE ∗	0.125	42.570	501.000	**1.000**	0.766	**1.000**	1.000
	OBS ∗	**0.067**	49.444	139.545	0.965	0.889	0.965	1.000
	iRand ∗	0.218	**0.588**	484.599	0.569	**0.011**	0.569	1.000
	CF^2 †	0.581	27.566	**0.000**	0.496	0.496	0.496	1.000
	CLEAR ‡	22.121	64.006	**0.000**	0.504	1.152	0.504	1.000
	G-CounteRGAN ‡	4.120	271.822	**0.000**	0.524	4.893	0.524	1.000
Tree-Cycles@32	DCE ∗	**0.143**	50.112	501.000	**1.000**	0.788	**1.000**	1.000
	OBS ∗	**0.143**	57.542	159.260	0.964	0.905	0.964	1.000
	iRand ∗	0.339	**0.590**	627.342	0.575	**0.009**	0.575	1.000
	CF^2 †	0.412	31.542	**0.000**	0.474	0.496	0.474	1.000
	CLEAR ‡	30.227	80.351	**0.000**	0.526	1.265	0.526	1.000
	G-CounteRGAN ‡	0.298	359.698	**0.000**	0.504	5.659	0.504	1.000
Tree-Cycles@48	DCE ∗	0.214	82.000	501.000	**1.000**	0.858	**1.000**	1.000
	OBS ∗	**0.147**	89.268	237.678	0.935	0.934	0.935	1.000
	iRand ∗	1.627	**0.533**	1594.374	0.527	**0.006**	0.527	1.000
	CF^2 †	3.771	47.568	**0.000**	0.494	0.498	0.494	1.000
	CLEAR ‡	31.081	171.983	**0.000**	0.506	1.800	0.506	1.000
	G-CounteRGAN ‡	6.612	1121.550	**0.000**	0.506	117.737	0.506	1.000

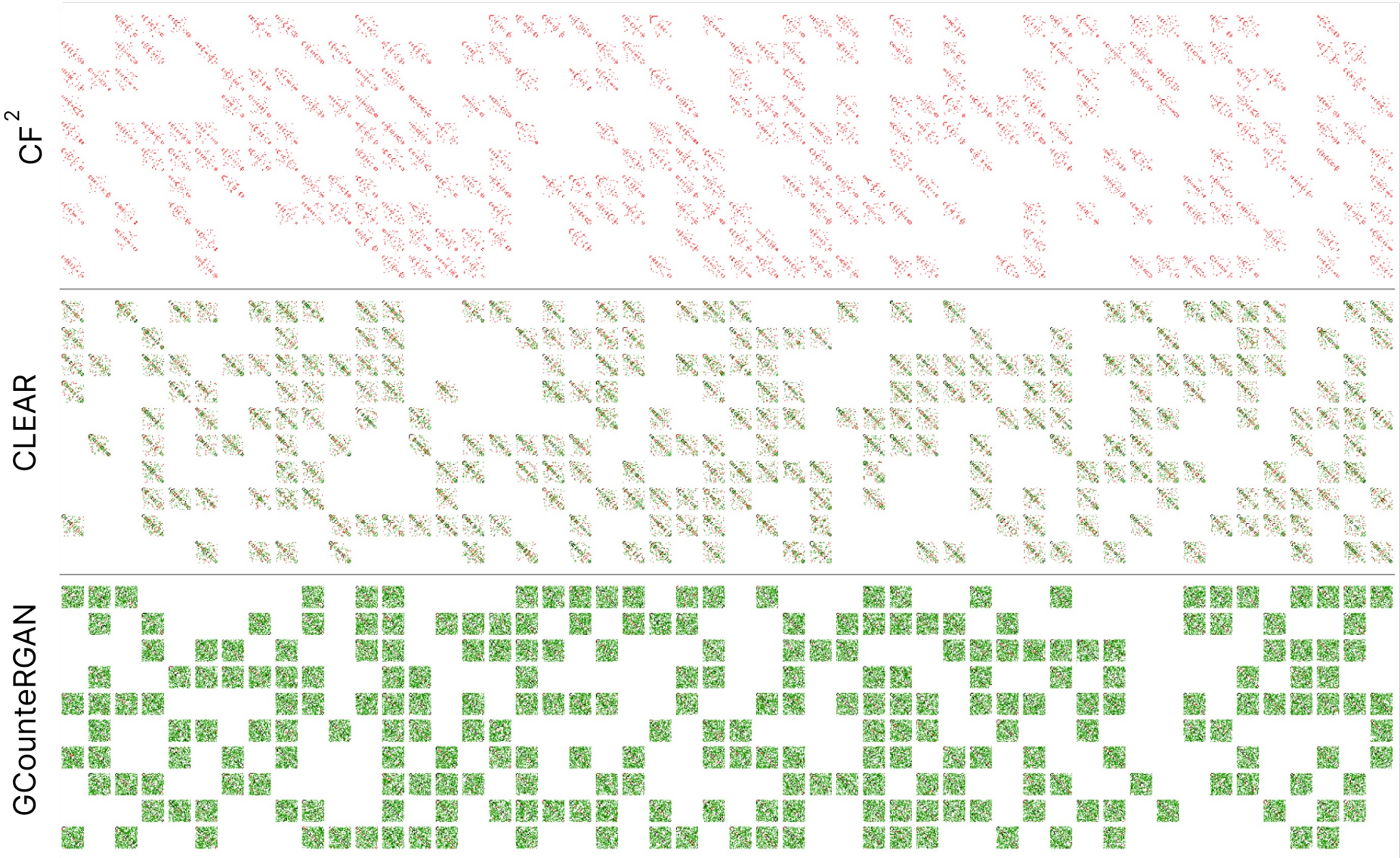
† symbolizes learning-based approaches; ‡ indicates generative approaches; ∗ depicts search (heuristic) methods.
Bold values are the best overall; underlined are second-best on average per dataset

Anecdotal Counterfactual Visualization



Counterfactual produced by G-CounteRGAN, CLEAR, and the optimistic baseline DCE on Tree-Cycles@28. As in the original graph, green edges are additions, red ones are removals, and **black** ones are maintained. An × denotes an invalid counterfactual, and a ✓ is valid.

Anecdotal Counterfactual Visualization



Counterfactual produced by CF², CLEAR, and G-CounteRGAN on Tree-Cycles@28.

Part Five
Conclusive Remarks

Conclusion

Q: "Can generative approaches compete with search/heuristic baselines in producing valid counterfactuals in synthetic datasets?" A: Not yet. But …

- They are crucial to **produce multiple counterfactuals** by sampling their learned latent spaces
- **They can learn** the input perturbation strategy differently from heuristic-based approaches
- They have the **potential** to become akin to a **Swiss-knife in GCE**

Thanks for your attention!

Slides & More

Bibliography

[1] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) 51(5), 1–42 (2018)

[2] Petch, J., Di, S., Nelson, W.: Opening the black box: the promise and limitations of explainable machine learning in cardiology. Canadian Journal of Cardiology (2021)

[3] Verenich, I., Dumas, M., La Rosa, M., Nguyen, H.: Predicting process performance: A white-box approach based on process models. Journal of Software: Evolution and Process 31(6), e2170 (2019)

[4] Aragona, D., Podo, L., Prenkaj, B., Velardi, P.: Coronna: a deep sequential framework to predict epidemic spread. In: Proceedings of the 36th Annual ACM Symposium on Applied Computing. pp. 10–17 (2021)

[5] Feng, W., Tang, J., Liu, T.X.: Understanding dropouts in moocs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 517–524 (2019)

[6] Verma, H., Mandal, S., Gupta, A.: Temporal deep learning architecture for prediction of covid-19 cases in india. Expert Systems with Applications 195, 116611 (2022)

[7] Prenkaj, B., Distante, D., Faralli, S., Velardi, P.: Hidden space deep sequential risk prediction on student trajectories. Future Generation Computer Systems 125, 532–543 (2021)

[8] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. 2023. A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges. ACM Comput. Surv. Just Accepted (September 2023). https://doi.org/10.1145/3618105

[9] Ma, J., Guo, R., Mishra, S., Zhang, A., Li, J.: CLEAR: generative counterfactual explanations on graphs. In: NeurIPS (2022)