Unifying Evolution, Explanation, and Discernment: A Generative Approach for Dynamic Graph Counterfactuals

Bardh Prenkaj*
Technical University of Munich
Munich, Germany

Mario Villaizán-Vallelado
University of Valladolid
Telefónica Research and Development
Valladolid, Spain

Tobias Leemann*
University of Tübingen
Tübingen, Germany

Gjergji Kasneci
Technical University of Munich
Munich, Germany

ABSTRACT

We present GRACIE (Graph Recalibration and Adaptive Counterfactual Inspection and Explanation), a novel approach for generative classification and counterfactual explanations of dynamically changing graph data. We study graph classification problems through the lens of generative classifiers. We propose a dynamic, self-supervised latent variable model that updates by identifying plausible counterfactuals for input graphs and recalibrating decision boundaries through contrastive optimization. Unlike prior work, we do not rely on linear separability between the learned graph representations to find plausible counterfactuals. Moreover, GRACIE eliminates the need for stochastic sampling in latent spaces and graph-matching heuristics. Our work distills the implicit link between generative classification and loss functions in the latent space, a key insight to understanding recent successes with this architecture. We further observe the inherent trade-off between validity and pulling explainee instances towards the central region of the latent space, empirically demonstrating our theoretical findings. In extensive experiments on synthetic and real-world graph data, we attain considerable improvements, reaching ∼99% validity when sampling sets of counterfactuals even in the challenging setting of dynamic data landscapes.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**; **Semi-supervised learning settings**; • **Mathematics of computing** → *Computing most probable explanation*; • **Computer systems organization** → **Neural networks**.

KEYWORDS

Counterfactual Explainability; Dynamic Graphs; Graph Neural Networks; Graph Autoencoders

*These authors contributed equally. Corresponding: bardh.prenkaj@tum.de

ACM Reference Format:
Bardh Prenkaj, Mario Villaizán-Vallelado, Tobias Leemann, and Gjergji Kasneci. 2024. Unifying Evolution, Explanation, and Discernment: A Generative Approach for Dynamic Graph Counterfactuals. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671831

1 INTRODUCTION

Graphs offer an intuitive framework to model the relationships, interactions, and dependencies that govern modern society. Due to their unique expressivity, they drive research in domains such as recommendations in social networks [13] and transaction fraud detection [20]. These are pressing problems. For instance, damages due to online fraud have more than doubled in the recent two years and are expected to reach a record-high of 48bn$ this year [37], such that better detection strategies are in high demand. Graphs are most suitable to model relationships involving tangible users, IP addresses, and other elements [8] that dictate this issue's complexity.

However, progress in domains such as fraud detection is substantially complicated by the dynamic and adversarial nature of the observed data: the application of fraud detection techniques also triggers adaptations in fraud tactics aimed specifically at evading detection [20]. Such dynamic data landscapes require constant updates of models.

Concurrently, the need for explainability has been widely acknowledged for system safety [7], scientific discovery [22], and even compliance with legal requirements [40], which is particularly important in financial applications. Regulations like GDPR [9] and the proposed AI Act [10] emphasize the demand for models that perform well and provide interpretable and actionable insights into their predictions. Counterfactual explanations [40] have emerged as a key element in meeting these regulatory requirements, as they shed light on model decisions by presenting alternative scenarios that would result in different outcomes. In summary, a multitude of pressing problems in both science and the economy **(1)** can be modeled intuitively through graphs, **(2)** evolve dynamically over time, and **(3)** require some form of explainability.

However, recent research [29] has highlighted a significant challenge when considering the dynamic and ever-changing nature of the data that models interact with. Data continuously undergoes changes and distribution shifts, which may warrant updates of the prediction models and can greatly impact the robustness, relevance, and validity of counterfactual explanations [28, 39]. Existing solutions have yet to adequately address the complex interaction between robust counterfactuals and the dynamic nature of data landscapes, a recently overlooked problem. As this holds for graphs, we present a powerful method for combining inference and counterfactual explanation for constantly evolving graph data.

To cope with distributional shift, we leverage the capabilities of generative classifiers, which have been attributed to increased robustness in prior work [34]. These classifiers model the full class-conditional probability, i.e., $p(\boldsymbol{x}|y)$ for an input $\boldsymbol{x}$ and each class y, and predict the class with the highest likelihood using Bayes' Theorem. Learning such a model with variational inference techniques results in GRACIE (Graph Recalibration and Adaptive Counterfactual Inspection and Explanation), a novel approach for generative classification and counterfactual explanations of dynamically changing graph data. GRACIE learns to produce counterfactuals from the underlying prediction model. Then, by exploiting its generative classification properties, decide how to discern between factuals and counterfactuals. This drives the search for counterfactual candidates in a self-supervised and principled manner. In successive steps, factuals and counterfactuals are used to recalibrate the decision boundary and to embody distribution shifts in time.

Unlike previous work, we do not assume that the learned graph representations are linearly separable to produce valid counterfactuals [33]. Additionally, we exploit the learned latent space to search for counterfactuals, thus eliminating the need to sample estimated edge probabilities on the embedded instances [25] and graph-matching heuristics [24], known for their NP-hardness [23].

Specifically, we go beyond the related literature by making the following contributions:

1. **Generative Classification For Graph Data.** Drawing from promising results of the robustness of generative classifiers, we propose a novel model for the generative classification of graph data.
2. **Theoretic Analysis and Insights.** We formalize the problem and derive a proposition that links the generative classification objective to the reconstruction loss of autoencoders and distances in the latent space. This allows for efficient classification with latent variable models and serves as an intuitive explanation for prior work's successes.
3. **Online Counterfactual Method.** We propose GRACIE, a novel, dynamic, and self-adapting approach for generative classification and counterfactual generation of temporally evolving graph data, leveraging the power of class-related Variational Graph Autoencoders (VGAEs).
4. **Empirical Analysis.** We conduct extensive experiments on synthetic and real-world graph data, demonstrating GRACIE's significant improvement in generative classification and counterfactual generation, particularly under dataset shifts. We will publicly release our code online.

2 RELATED WORK

Recently, deep learning relying on GNNs has been beneficial in solving graph-based prediction tasks, such as anomaly detection, link prediction [41], and protein-protein interaction predictions [35]. Despite their remarkable performance, GNNs are black boxes, making them unsuitable for high-impact and high-risk scenarios. The literature has proposed several post-hoc explainability methods to understand *what is happening under the hood* of the prediction models. Specifically, counterfactual explainability is useful to understand how modifications in the input lead to different outcomes. Similarly, a recent field in Graph Counterfactual Explainability (GCE) has emerged [32] that aims at producing counterfactuals for graphs. We provide the reader with an example that helps clarify a counterfactual example in graphs. Suppose we have a network of cryptocurrency transactions where nodes are traders and connections represent digital currency transfers from one trader to another. Assume that trader u sends a large amount of crypto to $\hat{u}$, a high-risk and flagged trader. A fraud detection system might alert a user u about restricting their account to minimal transfers for a designated period due to possible fraud suspicion reasons. A counterfactual explanation of u's account flagging would be *if u had refrained from transferring cryptos to û's account, then u's account would not have been flagged as suspicious*. Generally, GCE methods can be search- [1, 11], heuristic- (among others, [4]), and learning-based approaches (among others, [24, 27, 31]). While all the above methods provide counterfactuals in graphs, none of them is specialized to cope with evolving graphs in time, which adds complexity to finding valid and time-adaptable counterfactuals.

The problem of aligning counterfactual explanations in the presence of distributional drifts has become a relevant subject of study [14]. However, recent work has shown that counterfactuals are still fragile and can become invalidated when data points are deleted [29]. The authors identify influential data points whose deletion at time $t + \delta$ ensures that previously generated counterfactuals at time t become obsolete.

To the best of our knowledge, the only work that tackles the problem of updating invalid counterfactuals in a timely fashion is [33]. The authors represent graphs of the two opposite classes by closely aligning graph representations of the same class while pushing away those of the opposite. When drift distributions occur, they update the learned representations by calibrating their model and fitting a linear regressor, which also considers the similarity of graphs, to separate the newly learned graph representations. However, this work cannot tackle non-linearly separable representations, leading the explainer to fail to find valid counterfactuals. To address this, we propose a novel generative graph counterfactual explainer that separates the factual and counterfactual spaces while maintaining faithful representations of both classes over time.

Here, we propose to learn robust class representations and leverage generative classification to determine the class of never-seen-before graphs when the distributional drifts happen, and the decision boundary of the underlying predictor cannot be trusted. This helps our method adapt its learned representations self-supervisedly.

3 PROBLEM STATEMENT AND MOTIVATION

Preliminary. A temporal graph G_i can be defined as a sequence of graphs $\{G_i^{t_0}, \dots, G_i^{t_m}\}$ where $T = \{t_0, \dots t_m\}$ is a set of discrete snapshots. Here $G_i^{t_0}$ can be considered as the base graph structure which mutates in time into $G_i^{t_j} \in \mathcal{G}\ \forall j \in [1, m]$ where $\mathcal{G}$ is the

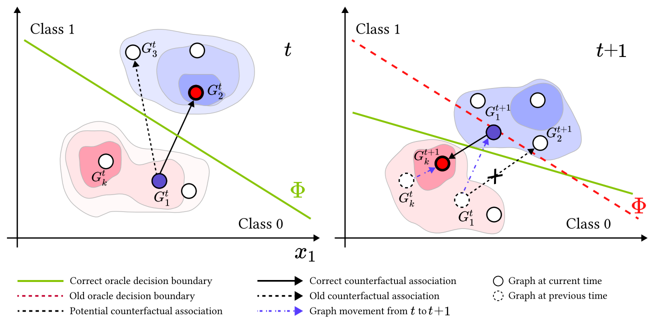
Class 1
G_3^t
G_2^t
t
G_k^t
G_1^t
Φ
Class 0
x_1
Class 1
$t+1$
G_1^{t+1}
G_2^{t+1}
G_k^{t+1}
G_k^t
G_1^t
Φ
Class 0
Correct oracle decision boundary
Old oracle decision boundary
Potential counterfactual association
Correct counterfactual association
Old counterfactual association
Graph movement from t to $t+1$
Graph at current time
Graph at previous time

Figure 1: Counterfactuals require updates in temporal graph problems. (left) The decision boundary of the oracle Φ trained on the data at time t correctly associates graph G_2^t as the counterfactual of G_1^t, satisfying Eq. 1. (right) As drifts occur, graph G_1^t transitions to G_1^{t+1}, crossing the previous (dashed line) decision boundary. Consequently, G_2^{t+1} cannot be a counterfactual for G_1^{t+1}. A new decision boundary must reflect the changes (bold full line) to maintain counterfactual validity.

dataset of all graphs. Each graph is generally a tuple $(\boldsymbol{X}, \boldsymbol{A})$ where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ represents the node feature vectors and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix s.t. n is the node number.

Motivation. In this paper, we address the challenge of counterfactual[1] validity amid data distributional shifts over time. We provide the reader with an example in Fig. 1 to showcase how counterfactuals get invalidated when distribution drifts happen [29]. We illustrate six temporal graphs on two snapshots, t and $t+1$. Let us assume we have 2D representations of these graphs for visualization clarity. Let us consider a specialized explainer, denoted as Ω. At snapshot, t, Ω needs to explain the underlying predictor (oracle) Φ, trained on data from the same snapshot. Φ generally has a specific decision boundary depicted by the bold separation line. At this point, Ω successfully generates a valid counterfactual, G_2^t, for the input G_1^t, instead of G_3^t since the distance between G_2^t and G_1^t is smaller than that of G_3^t and G_1^t. As time progresses to $t+1$, data distribution shifts may invalidate Φ's previous decision boundary, indicated by the bold dashed line. Therefore, Φ will fail to accurately reflect the true data separation – e.g., see how G_1^{t+1} gets misclassified because it has altered its true class w.r.t. the previous snapshot t. At snapshot $t+1$, it is crucial to observe that the counterfactual for G_1 cannot be G_2 since they now belong to the same true class. Instead, now, G_k^{t+1} would be a counterfactual for G_1^{t+1}, since it crosses the "true decision boundary" – bold separation line – unkown to Φ. Recognizing the unreliability of Φ's predictions in successive snapshots, we advocate for a robust mechanism that dynamically updates counterfactuals, ensuring their validity amid data distribution drifts.

[1]G' is a counterfactual of G if their predictions are different according to an underlying predictor (oracle) Φ [40].

Problem formalization. Given a black-box oracle $\Phi : \mathcal{G} \rightarrow \mathcal{Y}$, a counterfactual for G_i^t is produced by maximizing the objective

$$\mathcal{E}_\Phi \left(G_i^t\right) = \underset{G_j^t \in \mathcal{G}}{\arg\max}\, P_{cf}^t \left(G_j^t \,\middle|\, G_i^t, \Phi\left(G_i^t\right), \neg\Phi\left(G_i^t\right)\right) \tag{1}$$

where P_{cf} denotes the probability of G_j^t being a valid in-distribution counterfactual for an adequate description of the original graph G_i^t and the class $\Phi(G_i^t)$, where $\neg\Phi(G_i^t)$ indicates any other[2] class from the one predicted for G_i^t. Notice that the counterfactual produced for each graph G_i^t refers to the same snapshot t.

Within the probabilistic framework, expressed in Eq. 1, prior work [33] focused on discriminative models leading to preliminary promising results. In this work, we shift towards a generative classification perspective which allows us to dynamically update invalid counterfactuals without relying on Φ's outdated classifications.

4 A GENERATIVE CLASSIFICATION PERSPECTIVE

This section studies generative classifiers for robust inference under distribution shifts. Moreover, we formally link the generative classification objective to the reconstruction loss and norms in latent variable models, which allows for efficient inference using VGAEs.

Generative Classification. Generative classifiers (GCs, e.g., Ng and Jordan [26]) are a form of probabilistic models that perform classification through modeling the full joint distribution of features $\boldsymbol{x}$ and class labels y. In contrast to discriminative models, which only model $p\left(y|\boldsymbol{x}\right)$, they explicitly represent $p\left(\boldsymbol{x}|y\right)$ and predict the most likely label $\hat{y}$ via the Chain (or Product) Rule, i.e.,

$$\begin{aligned}\hat{y} =& \underset{y\in\mathcal{Y}}{\text{argmax}}\; p\left(\boldsymbol{x}, y\right) = \underset{y\in\mathcal{Y}}{\text{argmax}}\; p\left(\boldsymbol{x}|y\right) p\left(y\right) = \\ =& \underset{y\in\mathcal{Y}}{\text{argmax}}\; \log p\left(\boldsymbol{x}|y\right) + \log p\left(y\right).\end{aligned} \tag{2}$$

In prior work, GCs have been attested to superior generalization capabilities over discriminative classifiers (DCs) under dataset shifts [34, 38], accurately calibrated posteriors [3], and increased adversarial robustness [21]. A probabilistic model is required to model the class conditional densities. In this work, we rely on Variational Graph Autoencoders [17].

Variational Graph Autoencoders (VGAEs). We consider the following generative model where the graphs G are generated from factored latent representation $\boldsymbol{z} = \left[z_{v_1}, \ldots, z_{v_n}\right]$, and the true class label y:

$$p\left(G|y\right) = \int_{z\in\mathcal{Z}} p\left(G|z, y\right) p\left(z|y\right) dz \tag{3}$$

To represent $p\left(G|y\right)$, we use a single VGAE for each class $y \in \mathcal{Y}$, which is dependent on the class where each node has a latent vector and then define

$$\begin{aligned}p_{\theta_y}\left(G|z, y\right) &= p_{\theta_y}\left(\boldsymbol{A}, \boldsymbol{X}|z, y\right) \\ &= p_{\theta_y}\left(\boldsymbol{X}|\boldsymbol{A}, z, y\right) p_{\theta_y}\left(\boldsymbol{A}|z, y\right)\end{aligned} \tag{4}$$

where first the edges (or their probabilities) $\boldsymbol{A}$ are computed and the node features $\boldsymbol{X}$ are reconstructed subsequently. Each decoder

[2]The provided formulation supports multi-class classification problems. For simplicity, we concentrate on binary classification only.

depends on parameters θ_y for the respective class y. The respective joint distributions are modeled through probabilistic neural networks. We discuss the full parametrization of the generative model in Sec. A. We let

$$p(z) = \prod_{v_i} p(z_{v_i}) = \prod_{v_i} \mathcal{N}(z_{v_i}; \mathbf{0}, \mathbf{I}) \tag{5}$$

be an isotropic Gaussian prior. Such a latent variable model can be learned using standard variational approximation techniques. In variational techniques, the intractable inference density $q(z|G, y)$ required for optimization is approximated through a parametric distribution family element. If this family is expressive enough, e.g., it contains the actual density of the data-generating process, the variational model will converge to the true likelihood [16]. In the graph setting, we let $q(z|\boldsymbol{X}, \boldsymbol{A}, y)$ be a factorized representation,

$$q_{\varphi_y}(z|G, y) = \prod_{v_i} q_{\varphi_y}(z_{v_i}|G, y) \tag{6}$$

and specifically model $q(z_{v_i}|G, y) = \mathcal{N}(z_{v_i}|\boldsymbol{\mu}_{v_i}, \gamma^2 \boldsymbol{I})$, $\boldsymbol{\mu} = [\boldsymbol{\mu}_{v_1}, \ldots, \boldsymbol{\mu}_{v_n}] = \mathrm{GCN}_{\varphi_y}(G)$ is a matrix of mean vectors computed by a Graph Convolutional Network (GCN) with parameters φ_y, where $\gamma^2 > 0$ is a fixed hyperparameter.

As in Kipf and Welling [17], we train VGAEs for each of the classes by optimizing the parameters θ and φ in the objective

$$\begin{aligned}\mathrm{ELBO}_y(\theta_y, \varphi_y) = &\mathop{\mathbb{E}}_{q_{\varphi_y}(z|G,y)}\left[\log p_{\theta_y}(G|z, y)\right] \\ &- \mathrm{KL}\left[q_{\varphi_y}(z|G, y) \middle\| p(z)\right]\end{aligned} \tag{7}$$

where $\mathrm{KL}[q(\cdot)|| p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. Through maximization of the ELBO, we obtain optimal parameters, which we denote by

$$\left(\theta_y^*, \varphi_y^*\right) = \mathop{\mathrm{argmax}}_{\theta_y, \varphi_y} \mathrm{ELBO}_y(\theta_y, \varphi_y) \;\; \forall y \in \mathcal{Y} \tag{8}$$

Having obtained a generative latent variable model of a specific class y according to Eq. 8, we can now exploit its power to perform generative classification. If the variational family is expressive enough, i.e., it actually covers the true data-generating distribution, the ELBO converges to the logarithm of the true class-conditional probability $\log p(G|y)$. Therefore, we can use the generative models to compare different class probabilities and perform generative classification. Instead of maximizing the ELBO, we can equivalently minimize the negative ELBO, which has a surprisingly interpretable form in the variational model supposed in this work, as distilled in the following proposition:

Proposition 1 (Comparing Distance-Augmented Reconstruction Losses performs Implicit GC). *If the density model in Eqs.* (4)-(6) *is sufficiently expressive, i.e., it covers the true data generating process, computing*

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

is equivalent to performing generative classification for an input graph G*, where* f_{φ^*} *and* g_{ϕ^*} *are graph encoding and decoding networks, respectively.*

We provide the reformulation of the ELBO in Sec. A and proof of the proposition in Sec. B. From the above proposition, we see that adding the reconstruction loss[3] and the distance from the origin of the latent representation while compensating for the prior class probability implicitly performs generative classification. The above result comes with several implications. First, once we have a trained variational model, we can perform efficient classification by simply embedding the samples with the respective autoencoders and computing the reconstruction loss and latent norm. Second, our proposition might explain prior work's [33] success in using autoencoders to represent the data distributions and leveraging the reconstruction loss. In the remainder of this work, we will show that the performance can be substantially improved by continuing on the path set out through our theoretic considerations.

5 METHOD

Here, we present GRACIE, namely **G**raph **R**ecalibration and **A**daptive **C**ounterfactual **I**nspection and **E**xplanation. GRACIE[4] is a counterfactual explanation method that leverages Φ at the initial snapshot t_0 to acquire insights into the underlying data distribution and identify valid counterfactual instances. Importantly, GRACIE achieves this without relying on potentially outdated decision boundaries for snapshots $t_{i+1} > t_i$. We omit the time superscript in the formulas to enhance clarity, except when necessary.

Training. To untap the generative classification capabilities, we need to obtain a latent variable model for a specific class y by optimizing the ELBO as in Eq. 8. In binary classification problems, GRACIE relies on two VGAEs, $\Omega_0, \Omega_1 : \mathcal{G} \rightarrow \mathcal{G}$, corresponding to the negative and positive classes. They are responsible for learning the representation of each class separately. In this way, each VGAE intuitively constructs an embedding space that is limited to instances that belong to the same input distribution since, at t_0, we feed $G_i \in \mathcal{G}$ to Ω_y s.t. $y = \Phi(G_i)$. The encoder of the VGAEs is a 2-layered GCN. Differently from [17], we rely on the Graph Attention Network decoder proposed in [15] to model $p_{\theta_y}(\boldsymbol{A} \mid z, y)$, and thus reconstruct the adjacency matrix. Note that the encoder and the decoder have different parameters θ_y, φ_y, which depend on y.

Maximizing the ELBO in Eq. 8 can be equivalently expressed as the minimization of the following objective function, assuming that we have equal priors:

$$\begin{aligned}-\mathrm{ELBO}_y(\theta_y, \varphi_y) &\propto \mathcal{L}_{rec} + \mathcal{L}_{dist} \\ &\propto \frac{1}{2}\left(\mathop{\mathbb{E}}_{q_{\varphi_y}(z|G)}\left[\frac{||g_{\theta_y}(z) - G||_2^2}{\sigma^2}\right] + ||f_{\varphi_y}(G)||_2^2\right)\end{aligned} \tag{9}$$

We refer interested readers to the derivation of this term in Appendix A. By optimizing Eq. 9 for each VGAE, we learn to correctly reconstruct the input graphs via $\mathcal{L}_{rec}$ and maintain the embeddings close to the center of the latent space via $\mathcal{L}_{dist}$.

Unlike other works in graph autoencoders [12, 15], we reconstruct the node features and the graph topology. Thus, we ensure that the latent space of each VGAE correctly represents the distinguishing edges instead of just the node features of the graphs.

[3]We write $\|G - G'\|_2^2$ for graphs $G = (\boldsymbol{X}, \boldsymbol{A}), G' = (\boldsymbol{X}', \boldsymbol{A}')$ as a shorthand for $\|\boldsymbol{A} - \boldsymbol{A}'\|_2^2 + \|\boldsymbol{X} - \boldsymbol{X}'\|_2^2$.
[4]https://github.com/bardhprenkaj/HANSEL

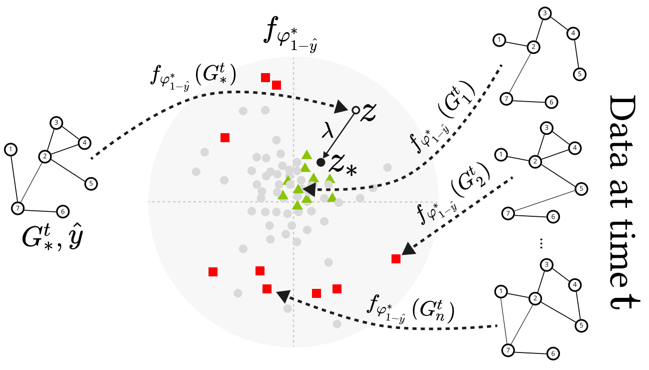
$f_{\varphi^*_{1-\hat{y}}}$
$f_{\varphi^*_{1-\hat{y}}}(G^t_*)$
z
λ
z_*
$G^t_*, \hat{y}$
$f_{\varphi^*_{1-\hat{y}}}(G^t_1)$
$f_{\varphi^*_{1-\hat{y}}}(G^t_2)$
$f_{\varphi^*_{1-\hat{y}}}(G^t_n)$
Data at time t

Figure 2: Search for top-k counterfactual candidates at inference time. G^t_* is mapped to $z = f_{\varphi^*_{1-\hat{y}}}(G^t_*)$ and pulled towards the center ($z_* = \lambda z$). Other graphs $G^t_1, \ldots, G^t_n$ also mapped to the latent space of $f_{\varphi^*_{1-\hat{y}}}$. Based on prior learned embedding (circles), class $1-\hat{y}$ instances move closer to the center, while the same class $\hat{y}$ instances shift away (rectangles). Triangles denote counterfactual candidates closest to z_*.

Inference and Counterfactual Generation. Having obtained the latent variable model, we can perform inference according to Proposition 1. For a never-seen-before instance, G^t_* s.t. $t > 0$, we first compute the most likely class $\hat{y}$ by evaluating the reconstruction loss and the latent norm for each class-specific VGAE. In this way, we can assign G^t_* to class $\hat{y}$, and use $\Omega_{1-\hat{y}}$ as the VGAE to find counterfactual candidates. In more detail, we use the encoder of $\Omega_{1-\hat{y}}$ to compute $z = q_{\phi^*_{1-\hat{y}}}(G^t_*)$, i.e., the latent representation for G^t_*. Now, since our goal is to pull the representations as close to the center of the latent space as possible, we can go inside the learned region by setting $z_* = \lambda z$, where $\lambda \in (0, 1)$. Notice that as the probability for class $1-\hat{y}$ increases as we approach the center, GRACIE refines its search space for suitable counterfactuals. Now, we map the rest of the instances in snapshot t into the same latent space of $\Omega_{1-\hat{y}}$ and find the top-k closest representations to z_*. Since z_* lies near the center of the latent space, the found counterfactuals should also be close by. In this way, the top-k counterfactuals are expected to be reliable since they share the same characteristics of the instances in $t-1$ that $\Omega_{1-\hat{y}}$ has learned to represent. Fig. 2 illustrates the search for counterfactuals.

Dynamic update. In snapshots $t > t_0$, we do not rely on Φ's predictions since they might not represent the reality of the new data (see Fig. 1). Instead, we use the learned representation of the VGAEs. As described above, at inference, for each graph G^t_*, we find k candidate counterfactuals close to the center of the $\Omega_{1-\hat{y}}$ with the generatively predicted class $\hat{y}$. We can use these counterfactuals to update $\Omega_{1-\hat{y}}$, and G^t_* to update $\Omega_{\hat{y}}$. In this way, both VGAEs reflect their representations according to the changes in the data. In the next snapshots, GRACIE is fully unsupervised, relying only on its generative classification power to search for counterfactuals and recalibrate the VGAEs representations. We also update $\log p(y)$ according to the classifications in the current snapshot for each graph and counterfactuals found, such that the classification in the next snapshots can reflect potential shifts in the class priors.

6 EXPERIMENTS

6.1 Datasets, hyperparameters, and evaluation

Datasets. We test GRACIE on a synthetic dataset, namely DynTreeCycles, generated according to [30], and four real datasets, namely DBLP-Coauthors [5], BTCAlpha, BTC-OTC [19], and Bonanza [6]. We extend TreeCycles [42] by introducing the time dimension, allowing graphs to evolve in time and potentially change their class, and name it DynTreeCycles (DTC). DBLP-Coauthors (DBLP) is a dataset of graphs representing coauthor relationships where the edge weights indicate the number of collaborations between two authors in a particular year. We use BTCAlpha (BTC-α) to have an initial investigation for our fraud detection example. The dataset is a network of who-trust-whom traders on the Bitcoin Alpha platform, where each trader expresses a trust score ranging from -10 to +10 on other platform members. BTC-OTC (BTC-β) is a similar dataset to BTC-α based on the Bitcoin OTC platform. Bonanza (BNZ) is a marketplace where users can buy/sell goods. After a purchase, buyers and sellers can rate each other (+1, 0, -1). All datasets have binary classes. Sec. C provides more details.

Evaluation metrics. We use *Validity* and *Graph Edit Distance (GED)* [32] as evaluation metrics. Since we return a list of counterfactuals for each input graph, we evaluate GRACIE by reporting values of these metrics @1, . . . , @k.

Given a graph G and a set of counterfactual candidates $\mathcal{G}'$, Validity@k measures the correctness of counterfactuals up to the k-th position. Validity (i.e., Validity@1) assesses whether an individual counterfactual graph G' at position 1 in $\mathcal{G}'$ is a correct counterfactual of the input graph G (i.e., $\mathbf{1}[\Phi(G) \neq \Phi(G')]$). Validity@k extends this assessment, defined as in Eq. 10.

$$\text{Validity}(G, \mathcal{G}', k) = \max_{i \in [1,k]} \mathbf{1}\left[\Phi(G) \neq \Phi(\mathcal{G}_i^{'})\right] \tag{10}$$

In simpler terms, Validity@k is 1 if at least one counterfactual up to the k-th position in $\mathcal{G}'$ is a correct counterfactual of the input graph G. We expect the more counterfactuals we sample from the latent space, the higher the Validity@k is (see Sec. 6.3).

GED measures the structural difference between G and its counterfactual G'. It calculates the distance based on a set of actions $p_1, p_2, \ldots, p_n \in \mathcal{P}(G, G')$, representing paths to transform G into G'. These paths involve adding or removing vertices or edges, with each action p_i associated with a cost $\omega(p_i)$. We prefer the one closer to the original instance G when comparing two counterfactual examples. Given G and a set of counterfactuals $\mathcal{G}'$, GED@k is the average GED of all k counterfactual candidates, as shown in Eq. 11.

$$\text{GED}(G, \mathcal{G}', k) = \frac{1}{k}\sum_{j=1}^{k}\left(\min_{\{p_1,\ldots,p_n\} \in \mathcal{P}(G, \mathcal{G}_j^{'})} \sum_{i=1}^{n} \omega(p_i)\right) \tag{11}$$

Fair training/evaluation policy. We split the data using a 90:10 train-test ratio. For each method, we report averages on 10-fold cross-validation. All methods share the same folds and portion of the train-test sets on each fold. We use omniscient oracles for each dataset to model the ground truth at each snapshot. This guarantees that the methods are evaluated on the real classes.

Baselines. We compare against the only time-dependent approach in the literature, namely DyGRACE [33]. For completeness, we

Table 1: Average of Validity@1 (the higher, the better) on 10-fold cross-validations for all snapshots in each dataset. Bold values indicate the best-performing approach; underlined is the second best; † indicates partial results due to non-convergence on some time steps; × indicates non-convergence at all[a].

	DTC	DBLP	BTC-α	BTC-β	BNZ
BDDS	0.465	<u>0.381</u>	<u>0.360†</u>	0.235	0.136
MEG	0.250	0.209	×	<u>0.260</u>	0.120†
CLEAR	0.458	0.024	0.214	0.125	0.000
G-CounteRGAN	0.507	0.256	0.236	×	<u>0.404</u>
DyGRACE	<u>0.525</u>	0.307	0.232	0.000†	0.232
GRACIE	**0.600**	**0.442**	**0.440**	**0.284**	**0.441**

[a]The criterion of non-convergence is to fail to produce at least one counterfactual within 14 days of execution on an HPC SGE Cluster of 6 nodes with 360 cumulative cores, 1.2Tb of RAM, and two GPUs (i.e., one Nvidia A30 and one A100).

adapt CLEAR [24] and G-CounteRGAN [25], two recent generative and time-unaware graph counterfactual explainers. We also compare against a heuristic- and a learning-based method, namely BDDS [1] and MEG [27], respectively. We train the explainers[5] on the first snapshot and use them to produce counterfactuals in the other snapshot without further updates on their learned weights.

Hyperparameters. We obtain the best hyperparameters for GRACIE via Bayesian optimization. We set the learning rate to 10^{-3}; the batch size to 64 for DTC and DBLP, to 24 for BTC-α and BTC-β, and 1 for BNZ; $\lambda = 0.5$; the replace rate for the autoencoder to 0.1; the mask rate to 0.5; the number of attention heads to 16 [15]. We set the number of epochs for DTC to 200 and 100 for the other datasets. We set $k = 50$ for DTC and $k = 10$ for the others. Sec. D shows the hyperparameter search spaces and the best choice for all methods.

6.2 Results

Table 1 shows the average Validity@1 over 10-fold cross-validations for all snapshots in each dataset. GRACIE is the best overall with an average improvement of 14.3%, 16%, 22.2%, 9.23%, and 3.76% on, respectively, DTC, DBLP, BTC-α, BTC-β, and BNZ over the second-best performing strategy. We first conduct a Friedman Test to show that GRACIE has, statistically and significantly, the best performance across the board. Here, we obtain a test statistic equal to 15.920 with a p-value of .0071. Since the p-value is less than .05, we can reject the null hypothesis that the average Validity@1 scores across all datasets are the same for all methods. Then, we perform a Bonferroni-Dunn post-hoc test where the control explainer is GRACIE. The corrected p-value, according to Bonferroni, is .05/5 = .01, where 5 is the number of comparisons (i.e., GRACIE vs. BDDS, GRACIE vs. MEG, etc.). The test suggests that GRACIE is statistically and significantly different (better) across the board (see Table 2). Fig. 3 shows the Validity@1 and GED@1 of all methods in different snapshots. GRACIE reaches an average improvement per snapshot of as much as 23.1%, 73.3%, 120.6%, 36.2%, and 115.7% in DTC, DBLP, BTC-α, BTC-β, and BNZ, respectively, in Validity@1 over the SoA.

[5]BDDS is not trained. It perturbs the input graph until it finds a counterfactual.

Table 2: P-values produced by the Dunn post-hoc test (with and without the Bonferroni correction) where the control explainer is GRACIE.

	GRACIE	
	w/o Bonferroni (*p-value* .05)	w/ Bonferroni (*p-value* .01)
BDDS	2.472×10^{-6}	3.708×10^{-5}
MEG	1.784×10^{-15}	2.676×10^{-14}
G-CounteRGAN	1.090×10^{-5}	1.635×10^{-4}
CLEAR	9.354×10^{-13}	1.403×10^{-11}
DyGRACE	2.014×10^{-6}	3.021×10^{-5}

Assessing counterfactuals in synthetic scenarios with subtle distributional variations. DTC is a synthetic scenario where the distributional shifts are not evident. Here, GRACIE has better Validity@1 than the SoA on 3/4 snapshots. CLEAR and G-CounteRGAN sample from the learned edge probabilities, and do not guarantee breaking cycles when going from a cyclic graph to a tree. They have a Validity@1 of ∼0.5 on all snapshots since they produce complete stochastic graphs and valid counterfactuals when the input instance is a tree. This inherent drawback of producing complete graphs causes these methods to have large GED@1. Only MEG has oscillatory Validity@1, which does not surpass the search-based baseline BDDS. Lastly, as expected, BDDS has the lowest GED@1 since it specifically optimizes to minimize it.

Counterfactual performance on real-world datasets. In DBLP, GRACIE outperforms the second-best, BDDS, on 8/11 snapshots. DyGRACE has compelling results for the first snapshot but cannot discern factuals from counterfactuals in later stages, lacking behind GRACIE of a factor of ∼2 in snapshots 7-10. Additionally, while GRACIE's Validity@1 has a non-decreasing trend, DyGRACE plummets, favoring closer counterfactuals to the original graph despite this major drop. We argue that having valid counterfactuals is more important than being closer to the original graph without crossing the decision boundary. CLEAR fails to produce valid counterfactuals, defaulting to returning the original instance, which accounts for a GED@1 of zero. G-CounteRGAN has a high standard error in Validity@1, making it the least reliable explainer. We argue that image-based convolutional operations adopted as in G-CounteRGAN make it unsuitable for real-world scenarios with complex topologies. MEG shows drastically fluctuating performances in Validity@1, suggesting that DBLP suffers extremely from data distribution shifts. Additionally, with each passing snapshot, MEG overshoots on the other side of Φ's decision boundary, translating into an increasing trend in terms of GED@1.

Interestingly, in BTC-α and BTC-β, the edge directionality is a challenging representation task for GCNs (also noticed in [36]). Nevertheless, GRACIE can represent the edges' directionality, reporting the best Validity@1 on all snapshots. In BTC-α, G-CounteRGAN has a random zig-zag trend on Validity@1, attributed to spurious learning of the edge probabilities on which counterfactuals are sampled. CLEAR has the best GED@1 among the competitors, although it is the worst in producing valid counterfactuals. Notice that GRACIE becomes better at producing valid counterfactuals (upward Validity@1 trend) and can also search for more similar

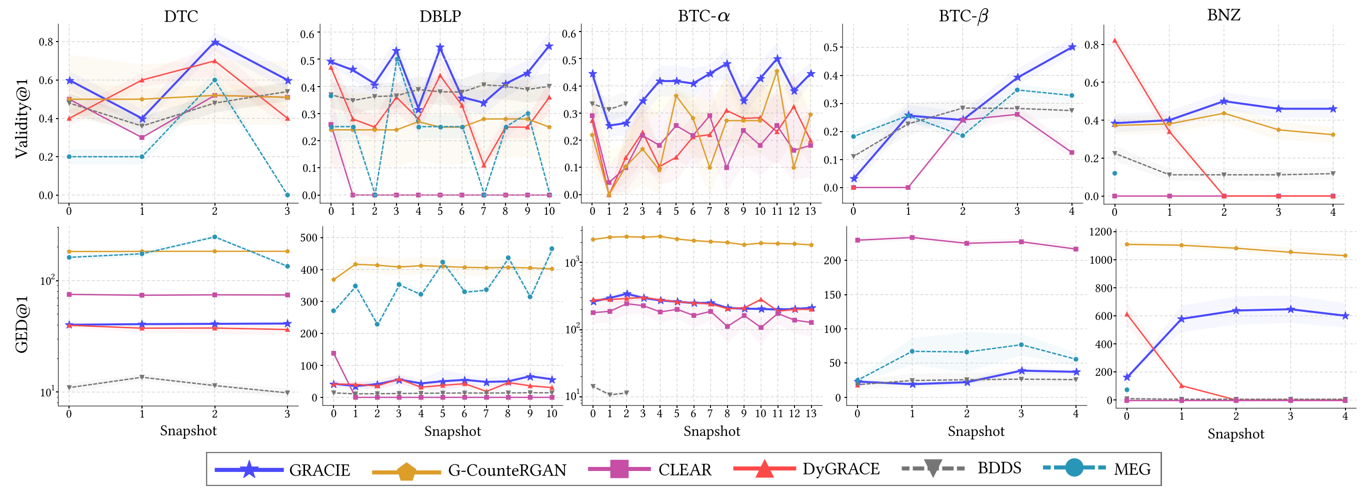

Figure 3: Average Validity@1 (*the higher, the better*) and GED@1 (*the lower, the better*) on 10-fold cross-validation.

ones w.r.t. the original graph (non-increasing GED@1 trend). Notice that BDDS does not produce results for all snapshots since it fails to converge its search space. Lastly, G-CounteRGAN considers the adjacency matrix a black-and-white image, which does not produce valid results. Moreover, the plain image convolution operations do not consider the real topology of the graph, which is node-invariant. Rather, they consider each node as a fixed pixel in a grid, making its GED@1 extremely high compared to the SoA.

BTC-β is the only dataset where DyGRACE cannot produce valid counterfactuals in the first snapshot. On the other snapshots, it does not converge within 14 days of execution. After careful examination, we believe that the training set of the linear regression used to separate the factual from counterfactuals is too resource-demanding (i.e., a Cartesian product on the learned graph representations), exceeding 13 TB of memory. G-CounteRGAN exceeds 64 TB of memory due to its flattened downstream linear layers after the convolution operations. We invite the reader to appreciate this dataset's complexity since SoA approaches cannot reach a Validity@1 of more than ~0.3 in many snapshots. Nevertheless, GRACIE's Validity@1 trend leads us to believe that the two VGAEs tend to become experts in correctly representing the two classes after a while (e.g., cold-start). We will investigate this in future works.

In BNZ, GRACIE has the steadiest performance similar to the second-best, G-CounteRGAN, outperforming it on all snapshots and presenting half the GED@1. Interestingly, DyGRACE has the best Validity@1 in the first snapshot – better than GRACIE. Nevertheless, it fails to adapt to the data distribution drift in the next snapshots by not producing valid counterfactual candidates. Notice that DyGRACE's GED@1 is zero since its default behavior is to return the original graph if it cannot return a counterfactual. Moreover, by analyzing the Validity@1 and GED@1 trends for GRACIE, DyGRACE, and G-CounteRGAN, as expected, we can state that they are directly proportionate. For instance, when GED@1 increases, the Validity@1 does so for GRACIE (see snapshots 0-2). Contrarily, when GED@1 decreases, the Validity@1 decreases for G-CounteRGAN. MEG does not converge on many snapshots, and CLEAR fails to explain Φ, defaulting to produce the original graph.

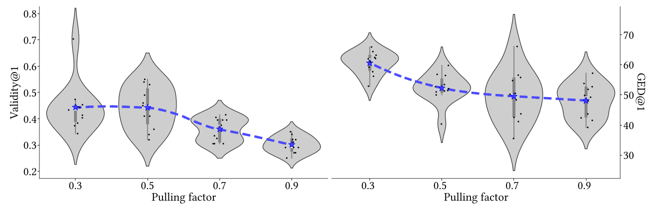

Figure 4: Validity@1 and GED@1 vs. the pulling factor λ.

6.3 Ablation studies

λ's impact on validity and recourse cost: unveiling counterfactual trade-offs. Fig. 4 illustrates Validity@1 and GED@1 when the pulling parameter λ changes on DBLP. As expected from the theoretical observations in Sec. 5, the lower the pulling factor, the better the chances of finding valid counterfactuals, but the farther they are w.r.t. the original graph. Contrarily, the higher the pulling factor, the lower the validity, and the nearer the counterfactuals (see dashed lines). In other words, for a graph G_* with generated classification $\hat{y}$ and its latent representation $\boldsymbol{z}_* = \lambda f_{\varphi^*_{\neg y}}(G)$, when $\lambda \longrightarrow 0$ the pulling factor makes $\boldsymbol{z}_*$ go towards the center of the space. This pulling phenomenon aids $\boldsymbol{z}_*$ in finding neighboring instances that have a high chance of being valid counterfactuals since $f_{\varphi^*_{\neg y}}$ is specialized for their representation. Contrarily, the instances most distant to $\boldsymbol{z}_*$, which is now mapped near the center, are likely to be invalid counterfactuals (cf. Proposition 1). Thus, they get discarded in the "update" policy. While we prefer to achieve a higher validity in finding counterfactuals, λ is a parameter that permits users to define the trade-off between counterfactual-factual discernment and generation cost (notice the upward Validity@1 and GED@1 trends when $\lambda \longrightarrow 0$).

Valid counterfactual sampling in the latent representation space. Fig. 5 illustrates the trend of the Validity@k on 10-fold cross-validations. Here, we illustrate box plots that indicate the distribution of the performances for a specific k aggregated for all snapshots.

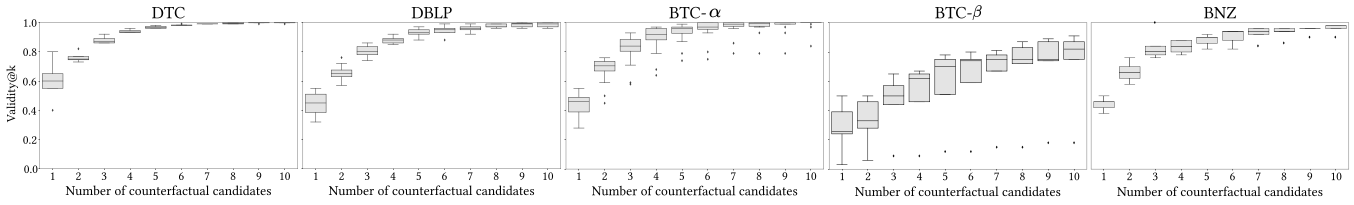

Figure 5: Validity (averages on 10-fold cross-validations) trend over all snapshots grouped on the top-k counterfactual candidates.

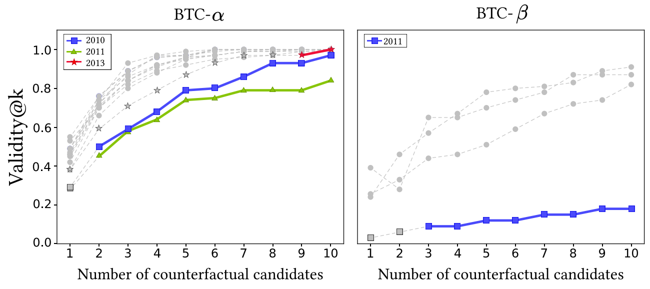

Figure 6: Validity trend on the top-k sampled counterfactuals expanded for BTC-α and BTC-β. The dashed lines illustrate those snapshots representing the inliers in the box plots of Fig. 5. The filled lines are outliers for a specific k.

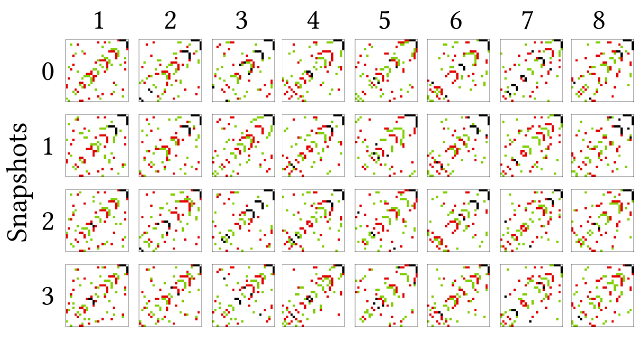

Figure 7: (best viewed in color) Valid counterfactuals produced by GRACIE on 8 randomly chosen graphs in the test set (columns) in DTC for each snapshot.

The plots show that GRACIE benefits from sampling multiple counterfactual candidates, thus leading to consistently better results. The sampling trend indicates a clear improvement from Validity@1 to Validity@k. Interestingly, in BTC-α and BTC-β, Validity@k is influenced by outlier scores for particular snapshots, as illustrated via diamonds. We noticed that the training set of several folds in the first portions of the monitoring time contains ego networks with similar structures but different edge weights labeled as fraud and genuine users. In this way, these similar graphs were used to train the two VGAEs, which learn a similar representation of both classes, making it difficult to separate them and correctly update the counterfactuals in successive snapshots. Fig. 6 expands the box-plot view for BTC-α and BTC-β of Fig. 5. Here, we show the average Validity@k for each snapshot expressed in years. The dashed lines illustrate those years where the average Validity is an inlier w.r.t. to the box plots of Fig. 5. Meanwhile, full lines show those values for each k that are outliers. Notice that there is a one-to-one correspondence with the diamonds (outliers) in Fig. 5 and the highlighted lines in Fig. 6 for BTC-α and BTC-β. For instance, in BTC-α, notice how the Validity@k in 2013 is normal until $k \in \{9, 10\}$. Interestingly, in BTC-β, 2011 is an outlier for $k \in [3, 10]$. Verifying the validity trend for 2011, we notice that it is the lowest across the board. This makes us believe that the graphs of the two classes do not have emphasizing and distinguishable topological characteristics w.r.t. the other years (snapshots).

6.4 Qualitative Inspection

Fig. 7 illustrates the difference in adjacency matrices between 8 randomly chosen instances from the test set in DTC and their corresponding valid counterfactuals in each snapshot. In each adjacency matrix, we color with *white* if the edge is neither in the input instance nor in the counterfactual candidate; with *red* the removal of the edge from the original input; with *green* an addition of the edge in the counterfactual; and with *black* an edge both in the input instance and the counterfactual. This pictorial briefly overviews the most preeminent edge operations (i.e., adding or removing edges) and the GED@1 between the original graph and its counterfactual. For example, a method with a lower GED@1 exposes an overall blacker image board since there are fewer perturbations on the original graph. A green-dominated sub-block indicates generating non-existing edges, while one dominated by red means many removed edges. GRACIE exposes a mixture of the three colors, which suggests that it supports all three edge perturbation operations. We are aware that the counterfactuals shown here exhibit many edge perturbation operations (also supported by the GED@1 in Fig. 3) due to GRACIE's learning procedure. Recall from Sec. 5 that each VGAE learns to represent graphs of the same class. At inference, for a graph G^* with predicted class $\hat{y}$ (see Proposition. 1), GRACIE searches for the counterfactuals in the latent space of the VGAE responsible for representing the opposite class $1 - \hat{y}$. As shown in Fig. 2, G^* gets mapped into $\boldsymbol{z}$ and then pulled inwards to $\boldsymbol{z}_*$, shifting the real representation towards the center of the latent space. Here, the nearest counterfactuals do not necessarily have the lowest GED w.r.t. G^* since we prioritize producing valid counterfactuals ratherthan closer ones. Notice that one can use the pulling factor λ to engender counterfactuals with lower GED@1 (see Fig. 4). We will investigate how to optimize for GED (a non-convex and non-differentiable function) to jointly find valid and similar counterfactuals for a given input graph in future works.

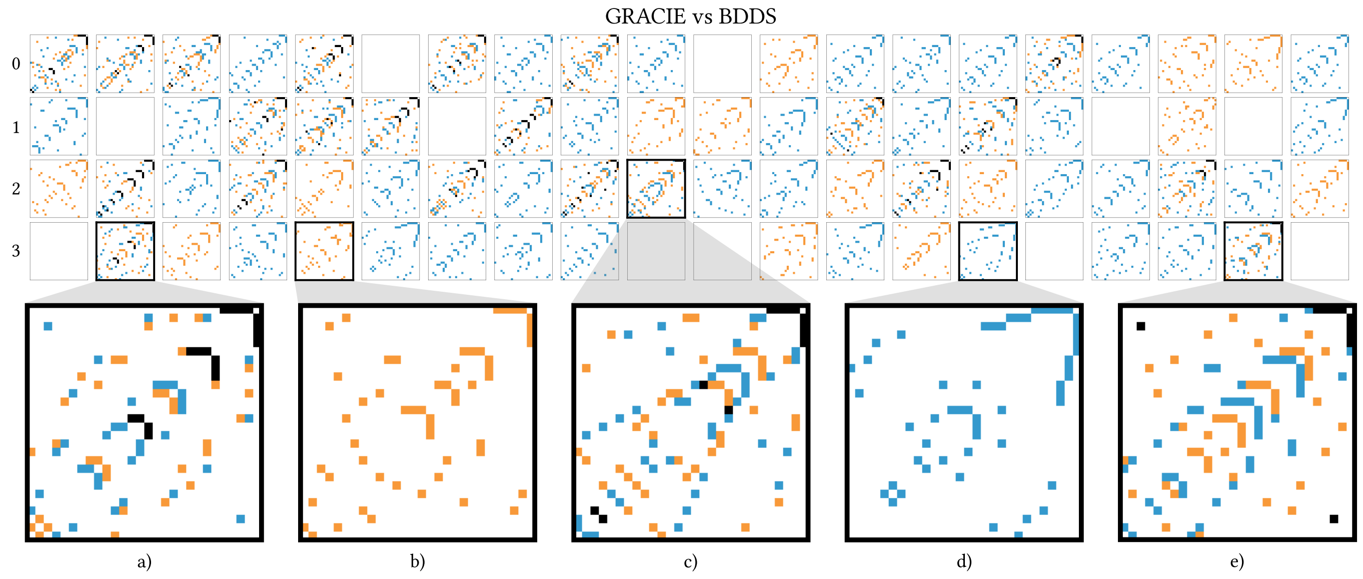

Figure 8: (best viewed in color) Qualitative illustration of the difference between the counterfactual candidates produced by GRACIE and BDDS on all snapshots (rows) on 10% randomly chosen graphs of the test set (columns) in DTC. Each element in the illustration represents the adjacency matrix of the graphs. We zoom in on three scenarios: a), c), and e) both explainers produce valid counterfactuals, b) GRACIE fails, and d) BDDS fails.

Since GRACIE and BDDS expose the most similar behavior on average on all datasets (see Table 1), we decided to directly compare them to understand which is their generation behaviors. Fig. 8 showcases a comparative view of counterfactual candidates generated by GRACIE and BDDS on DTC. For visualization clarity, we chose here to represent only 10% of randomly selected graphs from the test set in the entire dataset. Each image sub-block represents an adjacency matrix of the produced counterfactual candidate. The visual encoding employs colors to highlight differences between the counterfactual outputs of the two explainers. Common edges shared by both counterfactuals are shaded in black, symbolizing consensus between the methods. Additionally, edges exclusively generated by BDDS and not by GRACIE are colored in orange. On the other hand, edges solely engendered by GRACIE, but not BDDS, are depicted in blue. The illustration also accounts for graphs where neither explainer produces a valid counterfactual; these instances remain blank image sub-blocks within the visualization. Furthermore, instances wherein only one explainer successfully generates a counterfactual are represented by single-colour adjacency matrices. For instance, a matrix entirely orange implies that GRACIE fails to produce valid counterfactuals for that instance.

In contrast, a fully blue matrix shows that BDDS fails to generate viable counterfactuals. Note that GRACIE performs more edge operations than BDDS (i.e., notice the higher concentration of blue in the adjacency matrices, as supported by the GED trend in Fig. 3). Oppositely, BDDS exposes fewer edges in total (i.e., the sum of black and orange edges vs the sum of black and blue for GRACIE). This phenomenon is due to BDDS's underlying oblivious bidirectional search mechanism, which tries to add/remove edges at each iteration until it reaches a valid counterfactual until a maximum number of iterations is reached. This visualization effectively highlights the differences between GRACIE and BDDS in producing counterfactual candidates across the specified datasets and instances, offering insights into their strengths and weaknesses.

7 CONCLUSION

We presented GRACIE, one of the first generative approaches to address dynamic counterfactual explainability in the context of temporal graphs. GRACIE leverages VGAEs, self-supervisedly, to learn class representations and adapt to data distribution shifts that might invalidate counterfactuals in time. Unlike other approaches, GRACIE does not assume linear separatability between the latent representations. We performed extensive experiments on one synthetic and four real-world dynamic graph datasets, with an improvement of ∼13.1% in producing more valid counterfactuals than SoA approaches. We demonstrated that exploiting the pulling factor aligns with our intuition that the center of the latent space of the VGAEs should be used to find valid counterfactuals, maintaining a good trade-off with the similarity with the input. We also illustrated that sampling more candidates near the latent representation produces valid counterfactuals.

In the future, we will explore multi-class problems and the ability of GRACIE to generalize, simultaneously producing valid never-seen-before counterfactuals for multiple classes. Another avenue for investigation is to improve the class representations via more potent generative models. Lastly, we will incorporate uncertainty in determining valid counterfactuals during their search in the latent space. A possible direction could be to rely on hyperbolic spaces and exploit their intrinsic uncertainty modeling.

REFERENCES

[1] Carlo Abrate and Francesco Bonchi. 2021. Counterfactual graphs for explainable classification of brain networks. In *Proc. of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 2495–2504.

[2] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net.

[3] 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.

[4] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. 2021. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 5644–5655.

[5] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proc. of the National Academy of Sciences* 115, 48 (2018), E11221–E11230.

[6] Tyler Derr, Cassidy Johnson, Yi Chang, and Jiliang Tang. 2019. Balance in signed bipartite networks. In *Proc. of the 28th ACM International Conference on Information and Knowledge Management.* 1221–1230.

[7] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[8] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proc. of the 29th ACM international conference on information & knowledge management.* 315–324.

[9] European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council. *Official Journal of the European Union* (2016).

[10] European Union. 2023. Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts. *Official Journal of the European Union* (2023).

[11] 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.

[12] Shaohua Fan, Xiao Wang, Chuan Shi, Emiao Lu, Ken Lin, and Bai Wang. 2020. One2multi graph autoencoder for multi-view graph clustering. In *proceedings of the web conference 2020.* 3070–3076.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference.* 417–426.

[14] Johannes Haug and Gjergji Kasneci. 2021. Learning parameter distributions to detect concept drift in data streams. In *2020 25th International Conference on Pattern Recognition (ICPR).* IEEE, 9452–9459.

[15] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proc. of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 594–604.

[16] Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.

[17] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning.*

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proc.* OpenReview.net.

[19] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proc. of the Eleventh ACM International Conference on Web Search and Data Mining.* 333–341.

[20] Eren Kurshan and Hongda Shen. 2020. Graph computing for financial crime and fraud detection: Trends, challenges and outlook. *International Journal of Semantic Computing* 14, 04 (2020), 565–589.

[21] 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.

[22] Zelong Li, Jianchao Ji, and Yongfeng Zhang. 2022. From Kepler to Newton: Explainable AI for Science Discovery. In *ICML 2022 2nd AI for Science Workshop.*

[23] Lorenzo Livi and Antonello Rizzi. 2013. The graph matching problem. *Pattern Analysis and Applications* 16 (2013), 253–283.

[24] Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. 2022. CLEAR: Generative Counterfactual Explanations on Graphs. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.).

[25] Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. 2022. CounteRGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs. In *Proc. of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, Vol. 180. PMLR, 1488–1497.

[26] Andrew Ng and Michael Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems* 14 (2001).

[27] Danilo Numeroso and Davide Bacciu. 2021. Meg: Generating molecular counterfactual explanations for deep graph networks. In *2021 International Joint Conference on Neural Networks (IJCNN).* IEEE, 1–8.

[28] Martin Pawelczyk, Klaus Broelemann, and Gjergji. Kasneci. 2020. On Counterfactual Explanations under Predictive Multiplicity. In *Proc. of the 36th Conference on Uncertainty in Artificial Intelligence (UAI).* PMLR, 809–818.

[29] Martin Pawelczyk, Tobias Leemann, Asia Biega, and Gjergji Kasneci. 2023. On the Trade-Off between Actionable Explanations and the Right to be Forgotten. In *Proc. of the Eleventh International Conference on Learning Representations.*

[30] Mario Alfonso Prado-Romero, Bardh Prenkaj, and Giovanni Stilo. 2023. Developing and Evaluating Graph Counterfactual Explanation with GRETEL. In *Proc. of the Sixteenth ACM International Conference on Web Search and Data Mining.* 1180–1183.

[31] Mario Alfonso Prado-Romero, Bardh Prenkaj, and Giovanni Stilo. 2024. Robust Stochastic Graph Generator for Counterfactual Explanations. In *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21518–21526.

[32] Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. 2024. A Survey on Graph Counterfactual Explanations: Definitions, Methods, Evaluation, and Research Challenges. *ACM Comput. Surv.* 56, 7 (apr 2024).

[33] Bardh Prenkaj, Mario Villaizan-Vallelado, Tobias Leemann, and Gjergji Kasneci. 2023. Adapting to Change: Robust Counterfactual Explanations in Dynamic Data Landscapes. arXiv:2308.02353 [cs.LG]

[34] Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Interspeech 2007-8th Annual Conference of the International Speech Communication Association.*

[35] Manon Réau, Nicolas Renaud, Li C Xue, and Alexandre MJJ Bonvin. 2023. DeepRank-GNN: a graph neural network framework to learn patterns in protein–protein interfaces. *Bioinformatics* 39, 1 (2023), btac759.

[36] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. 2024. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference.* PMLR, 25–1.

[37] Statista. 2023. Value of e-commerce losses to online payment fraud worldwide from 2020 to 2023. https://www.statista.com/statistics/1273177/ecommerce-payment-fraud-losses-globally/.

[38] 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.

[39] Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. 2021. Towards Robust and Reliable Algorithmic Recourse. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34.

[40] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[41] Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. 2022. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* (2022).

[42] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019).

A DECOMPOSITION OF THE ELBO

Let $G = (\boldsymbol{A}, \boldsymbol{X}) \in \mathcal{A} \times \mathcal{X}$, where $\mathcal{A} \subset \mathbb{R}^{n \times n}$ represents possible adjacency matrices and $\mathcal{X} \subset \mathbb{R}^n$ represents possible node features. Let $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^k$ be a latent representation, where usually $n \gg k$. Note that we can construct a mapping from graphs $\mathcal{G}$ to a Euclidean space $\mathcal{X}$, so the above results also apply where G is a graph. Let p_θ be the distribution induces by the generative decoder $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ with parameters θ and q_φ be the distribution induces by the encoder $f_\varphi : \mathcal{X} \rightarrow \mathcal{Z}$ with parameters φ. We can derive the following lower bound on the posterior probability, which is known as the evidence lower bound (ELBO)

$$\begin{aligned}
\log p(G) &= \log \int_{\mathcal{Z}} p_\theta(G,z)dz \\
&= \log \int_{\mathcal{Z}} \frac{q_\varphi(z|G)}{q_\varphi(z|G)} p_\theta(G,z)dz \\
&= \log \mathbb{E}_{q_\varphi(z|G)}\left[\frac{p_\theta(G,z)}{q_\varphi(z|G)}\right] \geq \mathbb{E}_{q_\varphi(z|G)}\left[\log \frac{p_\theta(G,z)}{q_\varphi(z|G)}\right] \\
&= \mathbb{E}_{q_\varphi(z|G)}\left[\log \frac{p_\theta(G|z)p_\theta(z)}{q_\varphi(z|G)}\right] \\
&= \mathbb{E}_{q_\varphi(z|G)}\left[\log p_\theta(G|z)\right] + \mathbb{E}_{q_\varphi(z|G)}\left[\log p_\theta(z)\right] \\
&\quad - \mathbb{E}_{q_\varphi(z|G)}\left[\log q_\varphi(z|G)\right] \\
&:= \mathcal{L}_{rec} + \mathcal{L}_{dist} - \mathcal{L}_{uncert} := \text{ELBO}_{\phi,\theta}(G)
\end{aligned} \tag{12}$$

We now plug in the proposed parametric densities:

$$\begin{aligned}
p_\theta(z) &\sim \mathcal{N}(z; \mathbf{0}, \boldsymbol{I}) \\
p_\theta(\boldsymbol{A}|z) &\sim \mathcal{N}(\boldsymbol{A}; g_\theta(z), \sigma^2 \boldsymbol{I}) \\
p_\theta(\boldsymbol{X}|\boldsymbol{A},z) &\sim \mathcal{N}(\boldsymbol{X}; h_\theta(\boldsymbol{A},z), \sigma^2 \boldsymbol{I}) \\
q_\varphi(z|G) &\sim \mathcal{N}(z; f_\varphi(G), \gamma^2 \boldsymbol{I})
\end{aligned} \tag{13}$$

In our problem setup, we specifically have

$$f_\varphi(G) = \text{GCN}_{\varphi_y}(G) \tag{14}$$

and g_θ is defined for each element (v_i, v_j) via

$$g_\theta(z)_{v_i,v_j} = A_\theta(z_i)^\top A_\theta(z_j) \tag{15}$$

where A_θ is a mapping and h_θ is again defined through a GCN (or any NN), i.e.,

$$h_\theta(\boldsymbol{A}, z) = \text{GCN}_\theta(\boldsymbol{A}, z) \tag{16}$$

We can condense

$$\begin{aligned}
\log p_\theta(G|z) &= \log p_\theta(\boldsymbol{A}|z) + \log p_\theta(\boldsymbol{X}|\boldsymbol{A},z) \\
&= -\frac{1}{2}\mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|g_\theta(z) - \boldsymbol{A}\|_2^2}{\sigma^2}\right] \\
&\quad -\frac{1}{2}\mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|h_\theta(\boldsymbol{A},z) - \boldsymbol{X}\|_2^2}{\sigma^2}\right] - \frac{l+m}{2}(\log 2\pi + \log\sigma^2) \\
&= -\frac{1}{2}\mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|g'_\theta(z) - G\|_2^2}{\sigma^2}\right] - \frac{l+m}{2}(\log 2\pi + \log\sigma^2)
\end{aligned} \tag{17}$$

Instead of writing the two-step decoding process, we will now introduce $g'_\theta(z)$ which denotes the entire stochastic generating process such that $G = (\boldsymbol{A}, \boldsymbol{X}) = g'_\theta(z)$, and denote the sum of the two error norms $\|g'_\theta(z) - G\|_2^2 = \|g_\theta(z) - \boldsymbol{A}\|_2^2 + \|h_\theta(\boldsymbol{A},z) - \boldsymbol{X}\|_2^2$ This results in

$$\mathcal{L}_{rec} = -\frac{1}{2}\mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|g'_\theta(z) - G\|_2^2}{\sigma^2}\right] - \frac{l+m}{2}(\log 2\pi + \log\sigma^2) \tag{18}$$

$$\begin{aligned}
\mathcal{L}_{dist} &= -\text{CE}[q_\varphi(z|G)), p_\theta(z)] \\
&= -\text{D}_{\text{KL}}[q_\varphi(z|G)), p_\theta(z)] - H[q_\varphi(z|G))] \\
&= \mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|z\|_2^2}{2}\right] - \frac{k}{2}\log 2\pi = -\frac{k}{2}\left(\log 2\pi + \gamma^2\right) - \frac{1}{2}\|f_\varphi(G)\|_2^2
\end{aligned} \tag{19}$$

$$\begin{aligned}
-\mathcal{L}_{uncert} &= \mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|z - f_\varphi(G)\|_2^2}{2\sigma^2}\right] + \frac{k}{2}\log 2\pi + \frac{1}{2}\log|\gamma^2 \mathbb{I}| \\
&= \frac{k}{2}\left(\log 2\pi + 1 + \log\gamma^2\right) = \text{const.}
\end{aligned} \tag{20}$$

In summary, the log probability can be approximated by

$$\begin{aligned}
\text{ELBO}_{\phi,\theta}(G) &\propto \mathcal{L}_{dist} + \mathcal{L}_{rec} \\
&\propto -\frac{1}{2}\left(\mathbb{E}_{q_\varphi(z|G)}\left[\frac{\|g_\theta(z) - G\|_2^2}{\sigma^2}\right] + \|f_\varphi(G)\|_2^2\right)
\end{aligned} \tag{21}$$

In summary, we assign a high likelihood to samples encoded close to the center and have good reconstruction.

B GENERATIVE CLASSIFICATION WITH VGAES

In classification problems, we attempt to predict the class $\hat{y} \in \mathcal{Y}$ with the highest posterior probability. If we have a class conditional distribution model, we can reformulate the terms to arrive at the following formulation:

$$\begin{aligned}
\hat{y} &= \underset{y\in\mathcal{Y}}{\text{argmax}}\; p(y|G) = \underset{y\in\mathcal{Y}}{\text{argmax}}\; \frac{p(G|y)p(y)}{p(G)} \\
&= \underset{y\in\mathcal{Y}}{\text{argmax}}\; \log p(G|y) + \log p(y)
\end{aligned} \tag{22}$$

Suppose we now have a converged conditional density model with class-dependent decoders g_{θ_y} and encoders f_{φ_y} such that we can compute $\log p(G|y)$ as in Eqn. 22 (we basically add the condition on y to all the terms). If the density model is expressive enough, i.e., it covers the ground truth distribution, maximizing the ELBO results in a likelihood that represents the true likelihood [16]. We then have

$$\text{ELBO}_{\phi^*,\theta^*}(G|y) = \log p(G|y) \tag{23}$$

where ϕ^*, θ^* are the maximizing parameters of the ELBO. Plugging in the results, we obtain

$$\begin{aligned}
\hat{y} &= \underset{y\in\mathcal{Y}}{\text{argmax}}\; \log p(G|y) + c = \underset{y\in\mathcal{Y}}{\text{argmax}}\; \text{ELBO}_{\phi^*,\theta^*}(G|y) + c \\
&= \underset{y\in\mathcal{Y}}{\text{argmax}}\; -\frac{1}{2}\left(\underset{q_{\varphi^*}(z|G,y)}{\mathbb{E}}\left[\frac{\|g_{\theta^*_y}(z) - G\|_2^2}{\sigma^2}\right] + \|f_{\varphi^*_y}(G)\|_2^2\right) + c \\
&= \underset{y\in\mathcal{Y}}{\text{argmin}}\; \frac{1}{2}\left(\underset{q_{\varphi^*}(z|G,y)}{\mathbb{E}}\left[\frac{\|g_{\theta^*_y}(z) - G\|_2^2}{\sigma^2}\right] + \|f_{\varphi^*_y}(G)\|_2^2\right) - c,
\end{aligned} \tag{24}$$

where $c = \log p(y)$.

C DATASETS

Here, we provide details on the datasets used to assess the performance of GRACIE. Table 3 illustrates the their characteristics.

DynTree-Cycles (DTC) contains cyclic (1) and acyclic (0) graphs, and it is an established benchmark dataset for graph counterfactuals [4]. We extend this dataset by introducing the time dimension, allowing graphs to evolve in time and potentially change their class, and name it DynTree-Cycles (DTC). We repeat the dataset generation in [32] at each time step. In this way, a particular graph G_i^t can change its structure in $t+1$ and remain in the same class or move to the opposite one. This emulates a synthetic process of tracing the evolution of the graphs in the dataset according to time. Here,

Table 3: Statistics of the datasets.

	DTC	DBLP	BTC-α	BTC-β	BNZ
# of time steps	4	10	13	5	5
# of instances per time step	2,000	739	756	310	500
Avg # of nodes	28	13.54	82.29	12.38	39.86
Avg # of edges	27.62	41.91	152.12	27.09	251.42
Max # of nodes	28	148	278	349	382
Avg (out) node degree	1.97	4.12	3.62	2.46	5.19
# of classes	2	2	2	2	2
Class distribution at t_0	46.5 : 53.5	75.5 : 24.5	78.2 : 21.8	81.7: 18.3	55.8 : 44.2
Graph type	undirected	undirected	directed	directed	directed

we guarantee that the number of instances per snapshot is the same. All the instances in the dataset contain the same number of nodes and are connected graphs. Notice that the dataset has a balanced distribution between its two classes, making it conducive for learning-based explainers. However, its average node degree of 1.97 suggests the graphs are sparsely connected, challenging GCNs to capture intricate relationships between the nodes.

Additionally, notice that this dataset poses a difficult scenario since explainers need to learn both edge additions/removal operations given the duality aspect of the instances. In other words, explainers need to learn how to remove edges to pass from a cyclic graph to an acyclic graph and how to add edges to pass from a tree to a cyclic graph. Although the dataset poses a difficult scenario for most learning-based approaches – see Table 1 – GRACIE does not need to learn how to add or cut edges for an input graph. GRACIE's two-class representations and its neighboring mechanism permits us to search for counterfactual candidates entirely in the latent space.

DBLP-Coauthors (DBLP) consists of graphs representing authors, where edges denote co-authorship relationships and edge weights signify the number of collaborations in a given year. We focus on the time frame $[2000, 2010]$ and consider ego-networks of authors with at least ten collaborations in 2000. To trace the ego-network evolution from 2000 to 2010, we propagate ego-networks from the previous year whenever an author has no collaborations in a specific year t. Ego networks are labeled 1 if the central node has an impactful network in a particular year t w.r.t. the other instances; otherwise, we assign a 0. Here the graphs exhibit a notably higher average node degree of 4.12, implying denser interconnectivity between nodes. This characteristic may present challenges in handling the complexity of interrelated features and distinguishing relevant patterns. Additionally, the graphs are, on average, bigger than in DTC which makes the GCNs harder to learn effective node representations due to the "representational clash" of their receptive field [2].

BTCAlpha (BTC-α) and BTC-OTC (BTC-β) link to our initial fraud detection example and consist of who-trust-whom networks of traders on the Bitcoin Alpha and Bitcoin OTC platforms, respectively. Since Bitcoin users are anonymous, there is a need to maintain a record of users' reputations to prevent transactions with fraudulent and risky users. Members of the platforms rate others on a scale of -10 (total distrust) to +10 (total trust). For each year t in the dataset, we trace the connected components of the graph at time step t and label each one with 1 if it contains more "fraudulent" ratings than trustworthy ones; otherwise, with 0. It is interesting to notice that the graphs in this dataset are directed (i.e., asymmetric adjacency matrices), which might hamper [18] the performances of the graph convolution layers in the VGAEs (see BTC-β in Fig. 3 and 6).

Bonanza (BNZ). The Bonanza website is similar to eBay and Amazon Marketplace in that users create an account to buy or sell various goods. After a buyer purchases a product from a seller, both can provide a rating about the other along with a short comment. At the time of collection, Bonanza was using a rating scale of *Positive* (+1), *Neutral* (0), and *Negative* (-1) to rate another user after a transaction. For each year t in the dataset, we trace the connected components of the ego-network of each seller and label with 1 if it contains at least *ten good reviews* (i.e., the ratings' sum of its induced edges is ≥ 10); otherwise, with 0.

D HYPERPARAMETER SETTINGS

We rely on Bayesian optimization to obtain the best parameters for GRACIE and DyGRACE. We do the hyperparameter optimization only on the first time step t_0 with objective function Validity@1. For GRACIE, we use the search spaces in Table 4. For DyGRACE, we use the hyperparameters in Table 5.

Table 4: The hyperparameter search spaces and best choice for each dataset for GRACIE.

Hyperparameter	Search space	Best Choice				
		DTC	DBLP	BTC-α	BTC-β	BNZ
Learning rate	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Batch size	$\{1, 2, 4, 8, 16, 24, 32, 64\}$	64	64	24	24	1
Epochs	$[50, 350]$ with a step of 50	200	100	100	100	100
k	$[10, 100]$ with a step of 10	50	10	10	10	10

Table 5: The hyperparameter search spaces and best choice for each dataset for DyGRACE.

Hyperparameter	Search space	Best Choice				
		DTC	DBLP	BTC-α	BTC-β	BNZ
Learning rate	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-3}	10^{-4}	10^{-1}	10^{-1}	10^{-3}
Batch size	$\{1, 2, 4, 8, 16, 24, 32, 64\}$	24	24	4	4	1
Encoder out. dim.	$[1, 8]$	2	4	4	4	2
Epochs	$[10, 300]$ with a step of 100	20	150	100	100	100
k	$[5, 50]$ with a step of 5	50	10	10	10	10

We follow the suggestion of [32] to set the hyperparameters for the time-unrelated strategies (i.e., CLEAR, G-CounteRGAN, and MEG). For CLEAR, we set the epochs to 50, $\alpha = 0$, the dropout to 0.1, $\lambda_{cfe} = 0.1$, $\lambda_{kl} = 0.1$, $\lambda_{sim} = 0.1$, learning rate to 10^{-3}, and weight decay to 10^{-5}. For G-CounteRGAN, we set the number of training iterations to 250, the number of discriminator steps to 3, the number of generator steps to 2, and the binarization threshold to 0.5. For MEG, we set a maximum of 10 steps to perturb the adjacency matrix of the input graphs. We set the number of episodes to 10, the learning rate to 10^{-4}, the batch size to 1 since it can only perturb one graph at a time, $\gamma = 0.95$, and polyak to 0.995. Finally, we derive the input dimension MEG requires as n^2 where n is the maximum number of nodes in a dataset.