Explain the Explainer: Interpreting Model-Agnostic Counterfactual Explanations of a Deep Reinforcement Learning Agent

Ziheng Chen, Fabrizio Silvestri, Gabriele Tolomei, Jia Wang, He Zhu, and Hongshik Ahn

***Abstract*—Counterfactual examples (CFs) are one of the most popular methods for attaching post-hoc explanations to machine learning (ML) models. However, existing CF generation methods either exploit the internals of specific models or depend on each sample's neighborhood; thus, they are hard to generalize for complex models and inefficient for large datasets. This work aims to overcome these limitations and introduces RELAX, a model-agnostic algorithm to generate optimal counterfactual explanations. Specifically, we formulate the problem of crafting CFs as a sequential decision-making task. We then find the optimal CFs via deep reinforcement learning (DRL) with discrete-continuous hybrid action space. In addition, we develop a distillation algorithm to extract decision rules from the DRL agent's policy in the form of a decision tree to make the process of generating CFs itself interpretable. Extensive experiments conducted on six tabular datasets have shown that RELAX outperforms existing CF generation baselines, as it produces sparser counterfactuals, is more scalable to complex target models to explain, and generalizes to both classification and regression tasks. Finally, we show the ability of our method to provide actionable recommendations and distill interpretable policy explanations in two practical, real-world use cases.**

***Impact Statement*—The request for explainable AI/ML models is rapidly increasing. Indeed, attaching human-understandable explanations to model predictions is critical, especially in domains like healthcare, finance, and legal, just to name a few. Amongst the explainability techniques proposed in the literature, counterfactual explanations are one of the most promising. However, two main problems affect those methods: generalizability and efficiency. RELAX – the counterfactual explanation technique based on a deep reinforcement learning agent we introduce in this paper – overcomes the above limitations. Our method is model-agnostic: it can generate valid explanations for *any* model that are about 40% less complex (and thus more human-interpretable) than competitors in 42% less time. Moreover, the explanation process underneath RELAX can be itself human-interpretable, thus making our method valuable in practice, as we demonstrate in the two real-world use cases considered in this work.**

All the authors have contributed equally to this work.

Z. Chen and H. Ahn are with the Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, USA, 11794.
E-mail: ziheng.chen@stonybrook.edu
E-mail: hongshik.ahn@stonybrook.edu

F. Silvestri and G. Tolomei are respectively with the Department of Computer Engineering and the Department of Computer Science, Sapienza University of Rome, Rome, Italy, 00185.
E-mail: fsilvestri@diag.uniroma1.it
E-mail: tolomei@di.uniroma1.it

J. Wang is with the Department of Intelligent Science, The Xi'an Jiaotong-Liverpool University, Suzhou, China, 215000.
E-mail: jia.wang02@xjtlu.edu.cn

H. Zhu is with the Department of Computer Science at Rutgers University-New Brunswick, Piscataway, NJ, USA, 08854.
E-mail: hz375@cs.rutgers.edu

***Index Terms*—Counterfactual explanations, deep reinforcement learning, explainable AI, machine learning explainability.**

I. INTRODUCTION

RECENT years have witnessed surprising advances in machine learning (ML), which in turn have led to the pervasive application of ML models across several domains. Unfortunately, though, many ML systems deployed in the wild are treated as "black boxes", whose complexity often hides the inner logic behind their output predictions. In fact, knowing why an ML model returns a certain output in response to a given input is pivotal for many reasons like model debugging, aiding decision-making, or fulfilling legal requirements [1].

In some cases, end users may be open to trade the lack of transparency of ML models in exchange for their predictive accuracy, e.g., very few people are concerned with comprehending why the smart keyboard of their mobile phones suggests them a certain word to type next, as long as *that* word is actually what they would have typed. Conversely, the demand for more *explainable* ML models has become prominent, especially in social areas like finance [2] and healthcare [3], where human-to-human relationships still play a significant role. For instance, a banker should be able to tell a customer why their mortgage application has been denied by the bank's ML-powered decision support system, or a physician should provide to their patients the motivations behind any automatic ML-based diagnosis.

To properly achieve model transparency, a new initiative named *eXplainable AI* (XAI) has emerged [4]. A large body of work on XAI has flourished in recent years [5], and approaches to XAI can be broadly categorized into two classes [6]: *(i) native* and *(ii) post-hoc*. The former leverages ML models that are inherently interpretable and transparent, such as linear/logistic regression, decision trees, association rules, etc. The latter aims at generating *ex post* explanations for predictions made by opaque or black-box models like random forests and (deep) neural networks.

In this work, we focus on specific post-hoc explanations called *counterfactual explanations*, which are used to interpret predictions of individual instances in the form: "*If* A *had been different,* B *would not have occurred*" [7]. They work by generating modified versions of input samples that result in alternative output responses from the predictive model, i.e., *counterfactual examples* (CFs).

Typically, the problem of generating CFs is formulated as an optimization task, whose goal is to find the "closest" data point to a given instance, which crosses the decision boundary induced by a trained predictive model.[1] Depending on the level of access to the underlying predictive model, different CF generation methods have been proposed. More specifically, we can broadly distinguish between three categories of CF generators: *(i) model-specific* [8]–[12], *(ii) gradient-aware* [13], [14], and *(iii) model-agnostic* [15]–[17]. In short, existing CF generators either require full knowledge of the internals of specific models, or can only explain differentiable models, or depend on each sample's neighborhood, making them hardly generalize to complex models and inefficient for large datasets.

To overcome these limitations, we introduce the ***Re**inforcement **L**earning **A**gent e**X**plainer* (RELAX). Specifically, we formulate the problem of crafting CFs as a sequential decision-making task. We then find the optimal CFs via deep reinforcement learning (DRL).

A. Intuitive Description of our Method

The intuition behind RELAX is the following. The transformation of a given input instance into its optimal CF can be seen as the sequence of actions that an agent must take in order to get the maximum expected reward (i.e., the optimal policy). The total expected reward considers both the desired CF prediction goal (i.e., the CF and the original sample must result in different responses when they are input to the predictive model) and the distance of the generated CF from the original instance. At each time step, the agent has either achieved the desired CF transformation or it needs to: *(i)* select a feature to modify and *(ii)* set the magnitude of such change. To generate meaningful CFs, the agent must restrict itself to operate on the set of *actionable* features only, as not every input feature can always be changed (e.g., the "*age*" of an individual cannot be modified). Moreover, even if actionable, some features can only be changed toward one direction (e.g., a person can only increase their "*educational level*"). Plus, the total number of tweaked features, as well as the magnitude of the change, must also be limited, as this would likely result in a more feasible CF. We show that solving the constrained objective to find the optimal CF generator is equivalent to learning the optimal policy of a DRL agent operating in a discrete-continuous hybrid action space.

B. Our Main Contributions

RELAX is a model-agnostic CF generator, as it can be applied to *any* black-box predictive model. Indeed, in our framework the predictive model is just a "placeholder" resembling the environment which the DRL agent interacts with. Furthermore, we develop a distillation algorithm to extract decision rules from the DRL agent's policy in the form of an interpretable decision tree, thus making the process of generating CFs itself explainable.

We validate our method on six tabular datasets (five of them used for classification and one for regression), and compare it against several baselines using four standard quality metrics. Experimental results show that RELAX outperforms all the competitors in every single metric. To further demonstrate the impact of our CF generation method in practice, we consider two separate real-world use cases. In the first use case, we leverage CFs produced by RELAX to recommend patients how to lower their chances of developing diabetes. In the second scenario, we show how the CFs generated by RELAX can be used to suggest actions that a country should take to reduce the risk of mortality due to the COVID-19 pandemic. Finally, the same two use cases are also used to evaluate our policy distillation algorithm.

Overall, we summarize our main contributions as follows:

- RELAX is the first method for generating model-agnostic CFs based on deep reinforcement learning. It is scalable with respect to the number of features and instances. It can explain *any* black-box model trained on tabular input data, regardless of its internal complexity and prediction task (classification or regression).
- RELAX is flexible as it can be asked to modify only a subset of *actionable* features (e.g., "*age*" is not modifiable, whereas "*salary*" is), and change them toward "plausible" directions (e.g., "*education level*" can only be increased).
- We implement two variants of our method: RELAX-GLOBAL and RELAX-LOCAL, with the latter obtained via transfer learning from a pretrained version of the former. Both methods generate +60% valid CFs that are about 40% sparser than those produced by state of the art techniques yet take 42% less CF generation time.
- We tackle the sparse reward problem due to the large state and action space by integrating a hierarchical curiosity-driven exploration mechanism into our DRL agent.
- We propose a distillation algorithm that extracts decision rules from the RELAX agent's policy in the form of an interpretable decision tree, thereby making the CF generation process itself explainable.
- We assess the practical impact of RELAX on two real-world use cases, i.e., for showing the ability of our method to provide actionable recommendations and for distilling interpretable policy explanations.
- The source code implementation of RELAX, the experiments, and the dataset used for the COVID-19 risk of mortality task are publicly available.[2]

The remainder of the paper is organized as follows. In Section II, we review related work. In Section III, we formalize the problem of generating counterfactual explanations, while in Section IV we present RELAX, our proposed method to solve this problem using deep reinforcement learning. Section V describes an explainable DRL algorithm to distill decision rules from the RELAX's agent policy in the form of an interpretable decision tree. We validate our approach and discuss the main findings of our work in Section VI. In Section VII, we further demonstrate the practical impact of RELAX on two real-world use case scenarios. Finally, Section VIII concludes the paper.

[1] We voluntarily left this notion of "closeness" underspecified here; a more precise definition of it is provided in Section III.

[2] https://github.com/Mewtwo1996/ReLAX.git

II. Related Work

The set of work that are relevant to this paper cover three main areas: *(i)* counterfactual explanations for machine learning predictions, *(ii)* deep reinforcement learning in parametrized action space, and *(iii)* explainable reinforcement learning. In the following, we review the most significant contributions on each area above, separately.

A. Counterfactual Explanations for Machine Learning

An extensive review of the literature can be found in [6]. A possible approach to counterfactual explanation is NEAREST-CT [18]; instead of generating a synthetic CF for a given input sample, this approach selects the nearest CF data point from the training set. More sophisticated methods can be classified into *model-specific*, *gradient-aware*, and *model-agnostic*.

Model-specific. This category of counterfactual explanation methods is tailored for a particular family of ML models. Tolomei *et al*. [19] proposed FEATTWEAK, one of the first model-specific explanation method specifically designed for random forests, which exploits the internal structure of the learned trees to generate synthetic counterfactual instances. Another approach that is conceived for explaining tree ensembles is called FOCUS [20]. It frames the problem of finding counterfactual explanations as an optimization task and uses probabilistic model approximations in the optimization framework. Meanwhile, the rise of deep learning has given way to more complex and opaque neural networks (NNs). In this regard, DEEPFOOL [21] – which was originally designed for crafting adversarial examples to undermine the robustness of NNs – has proven effective also as a CF generation method. However, CFs obtained from adversarial learning techniques often require changing almost all the features of the original instances, making them unrealistic to implement. Thus, Le *et al*. [18] propose GRACE: a technique that explains NN model's predictions using sparser CFs, which are suitable also for high-dimensional datasets. More recently, Lucic *et al*. [22] have proposed CF-GNNExplainer, a counterfactual explanation method explicitly designed for graph neural networks.

Gradient-aware. These explanation methods need access to the model's gradients, and thus apply only to differentiable models. Amongst them, Mothilal *et al*. [23] propose DiCE, a framework for generating and evaluating a diverse and feasible set of counterfactual explanations from the model's gradients.

Model-agnostic. This class of approaches can generate explanations for *any* model. In this category, Guidotti *et al*. [5] introduce LORE. First, LORE trains an interpretable surrogate model (i.e., a decision tree) on the local neighborhood (generated by a genetic algorithm) of the sample to explain. Then, from the logic of the surrogate decision tree, it derives an explanation consisting of *(i)* a decision rule, which explains the reasons for the prediction; and *(ii)* a set of counterfactual rules, suggesting the changes to make to the instance's features. More recently, Karimi *et al*. [10] propose MACE, which frames the generation of model-agnostic CFs into solving a sequence of satisfiability problems, where both the distance function (objective) and predictive model (constraints) are represented as logic formulae.

Our RELAX method is totally model-agnostic and as general and flexible as possible. Moreover, unlike existing model-agnostic methods, RELAX is much more efficient in generating optimal CFs. Indeed, RELAX requires to train a DRL agent that makes use only of the input/output nature of the target predictive model to explain, regardless of its internal complexity or its gradients (as opposed to DiCE). Our method better scales to high-dimensional, large datasets than LORE: the genetic algorithm used to build each synthetic sample's neighborhood may be unfeasible for large feature spaces. Plus, LORE also requires to train a locally-interpretable decision tree that is tight to each generated neighborhood, and therefore may be prone to overfitting. RELAX can also seamlessly handle more complex models than MACE (e.g., deeper NNs), which needs to construct a first-order logic characteristic formula from the predictive model and test for its satisfiability. This may be intractable when the formula (i.e., the model to explain) is too large. Finally, as opposed to RELAX, neither LORE nor MACE control over the sparsity of the generated CFs; LORE does not even take into account their actionability.

B. Reinforcement Learning with Hybrid Action Space

Many real-world reinforcement learning (RL) problems requires complex controls with discrete-continuous hybrid action space. For example, in *Robot Soccer* [24], the agent not only needs to choose whether to shoot or pass the ball (i.e., discrete actions) but also the associated angle and force (i.e., continuous parameters). Unfortunately, most conventional RL algorithms cannot deal with such a heterogeneous action space directly. The straightforward methods either discretize the continuous action space into a large discrete set [25], or convert a discrete action into a continuous action method [26], but they significantly increase the problem complexity. To overcome this issue, a few recent works propose to learn RL policies over the original hybrid action space directly. Specifically, they consider a parameterized action space containing a set of discrete actions $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and corresponding continuous action-parameter $v_k \in V_k \subseteq \mathbb{R}$. In this way, the action space can be represented as: $\mathcal{A} = \bigcup_k \{(a_k, v_k) \mid a_k \in A, v_k \in V_k\}$. Masson *et al*. [24] propose a learning framework Q-PAMDP that alternatively learns the discrete action selection via Q-learning and employs policy search to get continuous action-parameters. Following the idea of Q-PAMDP, Khamassi *et al*. [27] treat two actions separately. The only difference is that they use policy gradient to optimize the continuous parameters. Both methods are on-policy and assume that continuous parameters are normally distributed. Also, Wei *et al*. [28] propose a hierarchical approach to deal with the parameterized action space, where the parameter policy is conditioned on the discrete policy. Although efficient, this method is found to be unstable due to its joint-learning nature. Recently, in order to avoid approximation as well as reduce complexity, Xiong *et al*. [29] introduce P-DQN, which seamlessly combines and integrate both DQN [30] and DDPG [31]. Empirical study indicates that P-DQN is efficient and robust.

C. Explainable Reinforcement Learning

As with traditional ML, the RL community has also been asked for post-hoc explanation methods that extract explainable policies from trained agents. Prior work has applied *program synthesis* to extract a high-level program from a DRL policy to retain the RL policy's performance while rendering the policy's decision logic interpretable in the program [32]–[34]. As opposed to these methods, our algorithm does not need to know a suitable context-free grammar to specify a search space a priori. Decision trees were previously used by various post-hoc explanation algorithms [35]–[37] to visualize a DRL policy's decision rules succinctly. These algorithms in general distill the decision knowledge of an RL agent, guided by the agent's Q-value function, into a discrete decision tree or a soft decision tree [35] following the DAGGER imitation learning technique [38]. Unlike our approach, these methods do not consider RL agents in parameterized action spaces with both discrete actions and continuous parameters.

III. Problem Formulation

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an input feature space and $\mathcal{Y}$ an output label space. Without loss of generality, we consider both the K-ary *classification* setting, i.e., $\mathcal{Y} = \{0, \dots, K-1\}$, and the *regression* setting, i.e., $\mathcal{Y} \subseteq \mathbb{R}$. Suppose there exists a predictive model $h_{\boldsymbol{\omega}} : \mathcal{X} \mapsto \mathcal{Y}$, parameterized by $\boldsymbol{\omega}$, which accurately maps any input feature vector $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathcal{X}$ to its label $h_{\boldsymbol{\omega}}(\boldsymbol{x}) = y \in \mathcal{Y}$.

The idea of counterfactual explanations is to reveal the rationale behind predictions made by $h_{\boldsymbol{\omega}}$ on individual inputs $\boldsymbol{x}$ by means of counterfactual examples (CFs). More specifically, for an instance $\boldsymbol{x}$, a CF $\widetilde{\boldsymbol{x}} \neq \boldsymbol{x}$ according to $h_{\boldsymbol{\omega}}$ is found by perturbing a subset of the features of $\boldsymbol{x}$, chosen from the set $\mathcal{F} \subseteq \{1, \dots, n\}$. The general goal of such modification is to transform $\boldsymbol{x}$ into $\widetilde{\boldsymbol{x}}$ so to change the original prediction, i.e., $h_{\boldsymbol{\omega}}(\widetilde{\boldsymbol{x}}) \neq h_{\boldsymbol{\omega}}(\boldsymbol{x})$ [13]. In particular, this depends on whether $h_{\boldsymbol{\omega}}$ is a classifier or a regressor. In the former case, the objective would be to transform $\boldsymbol{x}$ into $\widetilde{\boldsymbol{x}}$ so to change the original predicted class $c = h_{\boldsymbol{\omega}}(\boldsymbol{x})$ into another $\widetilde{c} = h_{\boldsymbol{\omega}}(\widetilde{\boldsymbol{x}})$, such that $\widetilde{c} \neq c$. Notice that $\widetilde{c}$ can be either specified upfront (i.e., *targeted* CF) or it can be *any* $\widetilde{c} \neq c$ (i.e., *untargeted* CF). In the case $h_{\boldsymbol{\omega}}$ is a regressor, instead, the goal is trickier: one possible approach to specifying the validity of a counterfactual example $\widetilde{\boldsymbol{x}}$ is to set a threshold $\delta \in \mathbb{R} \setminus \{0\}$ and let $|h_{\boldsymbol{\omega}}(\widetilde{\boldsymbol{x}}) - h_{\boldsymbol{\omega}}(\boldsymbol{x})| \geq \delta$. However, CFs found via such a thresholding are sensitive to the choice of δ [39].

Either way, as long as the CF classification or regression goal is met, several CFs can be generally found for a given input $\boldsymbol{x}$. This may lead to a situation where many CFs are in fact unrealistic or useless, as they are too far from the original instance. Therefore, amongst all the possible CFs, we search for the *optimal* $\widetilde{\boldsymbol{x}}^*$ as the most "reasonable", "closest" to $\boldsymbol{x}$. Intuitively, this is to favor CFs that require the *minimal* perturbation of the original input.

More formally, let $g_{\boldsymbol{\theta}} : \mathcal{X} \mapsto \mathcal{X}$ be a counterfactual generator, parametrized by $\boldsymbol{\theta}$, that takes as input $\boldsymbol{x}$ and produces as output a counterfactual example $\widetilde{\boldsymbol{x}} = g_{\boldsymbol{\theta}}(\boldsymbol{x})$. For a given sample $\mathcal{D}$ of i.i.d. observations drawn from a probability distribution, i.e., $\mathcal{D} \sim p_{\text{data}}(\boldsymbol{x})$, we can measure the cost of generating CFs with $g_{\boldsymbol{\theta}}$ for all the instances in $\mathcal{D}$, using the following counterfactual loss function:

$$\mathcal{L}_{\text{CF}}(g_{\boldsymbol{\theta}}; \mathcal{D}, h_{\boldsymbol{\omega}}) = \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \ell_{\text{pred}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x}); h_{\boldsymbol{\omega}}) + \lambda \ell_{\text{dist}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x})). \tag{1}$$

The first component (ℓ_{pred}) penalizes when the CF prediction goal is *not* satisfied. Let $\mathcal{C} \subseteq \mathcal{X}$ be the set of inputs which do *not* meet the CF prediction goal, e.g., in the case of classification, $\mathcal{C} = \{\boldsymbol{x} \in \mathcal{X} \mid h_{\boldsymbol{\omega}}(\boldsymbol{x}) = h_{\boldsymbol{\omega}}(g_{\boldsymbol{\theta}}(\boldsymbol{x}))\}$. Once the set $\mathcal{C}$ is defined in terms of the desired CF prediction goal, we can compute ℓ_{pred} as follows:

$$\ell_{\text{pred}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x}); h_{\boldsymbol{\omega}}) = \mathbb{1}_{\mathcal{C}}(\boldsymbol{x}), \tag{2}$$

where $\mathbb{1}_{\mathcal{C}}(\boldsymbol{x})$ is the well-known 0-1 indicator function, which evaluates to 1 if $\boldsymbol{x} \in \mathcal{C}$, or 0 otherwise.

The second component $\ell_{\text{dist}} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{>0}$ is any arbitrary distance function that discourages $\widetilde{\boldsymbol{x}}$ to be too far away from $\boldsymbol{x}$. For example, $\ell_{\text{dist}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x})) = ||\boldsymbol{x} - g_{\boldsymbol{\theta}}(\boldsymbol{x})||_p$, where $||\cdot||_p$ is the L^p-norm. In this work, inspired by previous approaches, we set ℓ_{dist} equal to L^1-norm [5].

In addition, λ serves as a scaling factor to trade off between ℓ_{dist} and ℓ_{pred}. Notice, though, that not every input feature can be lightheartedly modified to generate a valid CF, either because it is strictly impossible to do it (e.g., the "*age*" of an individual cannot be changed), and/or due to ethical concerns (e.g., the "*race*" or the "*political tendency*" of a person). Therefore, we must restrict $\mathcal{F}$ to the set of *actionable* features only. Notice that $\mathcal{F}$ is domain-specific (i.e., what subset of input features are considered actionable depends on the application domain). Anyway, the size of $\mathcal{F}$ is known and fixed apriori, i.e., $1 \leq |\mathcal{F}| \leq n$. Plus, we may want to limit the maximum number of actionable features that can be perturbed at most, i.e., $p_{\max} \leq m$, where $1 \leq m \leq |\mathcal{F}|$.

Eventually, we can find the optimal CF generator $g^* = g_{\boldsymbol{\theta}^*}$ as the one whose parameters $\boldsymbol{\theta}^*$ minimize (1), i.e., by solving the following constrained objective:

$$\begin{aligned} \boldsymbol{\theta}^* \;=\; & \arg\min_{\boldsymbol{\theta}} \Big\{ \mathcal{L}_{\text{CF}}(g_{\boldsymbol{\theta}}; \mathcal{D}, h_{\boldsymbol{\omega}}) \Big\} \\ & \text{subject to:} \quad p_{\max} \leq m. \end{aligned} \tag{3}$$

This in turn allows us to generate the optimal CF $\widetilde{\boldsymbol{x}}^*$ for any $\boldsymbol{x}$, as $\widetilde{\boldsymbol{x}}^* = g^*(\boldsymbol{x})$. Finally, the resulting optimal counterfactual explanation can be simply computed as $\boldsymbol{e}_{\boldsymbol{x}} = \widetilde{\boldsymbol{x}}^* - \boldsymbol{x}$ [40]. The overview of a generic counterfactual explanation framework is shown in Fig. 1.

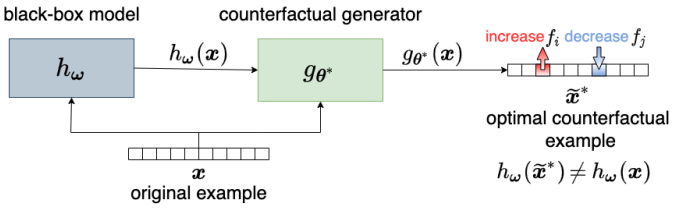

Fig. 1: Overview of a generic counterfactual explainer.

IV. Proposed Framework: ReLAX

In this work, we propose to find the optimal CF generator for any arbitrary predictive model – i.e., to solve the constrained optimization problem defined in (3) – via deep reinforcement learning (DRL). We call our method ***Re**inforcement **L**earning **A**gent e**X**plainer* (ReLAX).

A. Markov Decision Process Formulation

We consider the problem of computing the optimal counterfactual example $\widetilde{\boldsymbol{x}}^*$ from $\boldsymbol{x} \in \mathcal{D}$ – i.e., the optimal CF generator g^* defined in (3) – as a sequential decision-making task. More precisely, we refer to the standard reinforcement learning setting, where at each time step an agent: *(i)* takes an action (i.e., selects a feature of the original sample $\boldsymbol{x}$ to tweak *and* the magnitude of such change) and *(ii)* receives an observation (i.e., the prediction output by $h_{\boldsymbol{\omega}}$ on the input just modified according to the action taken before) along with a scalar reward from the environment. The process continues until the agent eventually meets the specified CF prediction goal and the optimal CF $\widetilde{\boldsymbol{x}}^*$ is found.

We formulate this process as a standard Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, p_0, r, \gamma\}$. Below, we describe each component of this framework, separately.

***States** ($\mathcal{S}$).* At each time step t, the agent's state is $S_t = s_t$, where $s_t = (\boldsymbol{x}_t, \boldsymbol{f}_t) \in \mathcal{S}$ represents the current modified sample ($\boldsymbol{x}_t$) along with the set of features changed so far ($\boldsymbol{f}_t$). More specifically, $\boldsymbol{f}_t \in \{0,1\}^{|\mathcal{F}|}$ is an $|\mathcal{F}|$-dimensional binary indicator vector, where $\boldsymbol{f}_t[k] = 1$ iff the actionable feature k has been modified in one of the actions taken by the agent *before* time t. Initially, when $t = 0$, $\boldsymbol{x}_0 = \boldsymbol{x}$ and $\boldsymbol{f}_0 = 0^{|\mathcal{F}|}$.

If the prediction goal is met, e.g., $h_{\boldsymbol{\omega}}(\boldsymbol{x}_t) \neq h_{\boldsymbol{\omega}}(\boldsymbol{x})$, the agent reaches the end of the episode and the process terminates returning $\widetilde{\boldsymbol{x}}^* = \boldsymbol{x}_t$ as the CF for $\boldsymbol{x}$. Otherwise, the agent must select an action $A_t = a_t$ so to: *(i)* pick a feature to change amongst those which have not been modified yet and *(ii)* decide the magnitude of that change.

***Discrete-Continuous Hybrid Actions** ($\mathcal{A}$).* To mimic the two-step behavior discussed above, differently from completely discrete or continuous actions, we consider a discrete-continuous hybrid action space, according to the two-tier hierarchical structure detailed below.

For an arbitrary step t, we maintain the set of feature identifiers that the agent is allowed to modify $\mathcal{F}_t$; initially, when $t = 0$, $\mathcal{F}_0 = \mathcal{F}$ as the agent can pick *any* of the actionable features to change. Then, at each time step $t > 0$, the agent first chooses a high level action k_t from the discrete set $\mathcal{F}_t \subset \mathcal{F} = \mathcal{F} \setminus \bigcup_{j=0}^{t-1} k_j$. This is to allow each feature to be selected at most in one action. Upon choosing $k_t \in \mathcal{F}_t$, the agent must further select a low level parameter $v_{k_t} \in \mathbb{R}$, which specifies the magnitude of the change applied to feature k_t. It is worth noticing that we can confine the action's range and direction by directly putting a constraint on the low level parameter v_{k_t}. Overall, $a_t = (k_t, v_{k_t})$, and our discrete-continuous hybrid action space is:

$$\mathcal{A}_t = \{(k_t, v_{k_t}) \mid k_t \in \mathcal{F}_t, v_{k_t} \in \mathbb{R}\}.$$

***Transition Function** ($\mathcal{T}$).* Let $a_t = (k_t, v_{k_t}) \in \mathcal{A}_t$ be the generic action that the agent can take at time t. The action a_t deterministically moves the agent from state s_t to state s_{t+1}, by operating on $\boldsymbol{x}_t$ and $\boldsymbol{f}_t$ as follows:

$$\mathcal{T}((\boldsymbol{x}_t, \boldsymbol{f}_t), a_t, (\boldsymbol{x}_{t+1}, \boldsymbol{f}_{t+1})) = \begin{cases} 1, & \text{if } \boldsymbol{x}_t \overset{a_t}{\rightsquigarrow} \boldsymbol{x}_{t+1} \wedge \boldsymbol{f}_t \overset{a_t}{\rightsquigarrow} \boldsymbol{f}_{t+1} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The statements $\boldsymbol{x}_t \overset{a_t}{\rightsquigarrow} \boldsymbol{x}_{t+1}$ and $\boldsymbol{f}_t \overset{a_t}{\rightsquigarrow} \boldsymbol{f}_{t+1}$ are shorthand for $\boldsymbol{x}_{t+1}[k_t] = \boldsymbol{x}_t[k_t] + v_{k_t}$ and $\boldsymbol{f}_{t+1}[k_t] = 1$, respectively. This corresponds to increasing the value of feature k_t by the magnitude v_{k_t}, and updating the binary indicator vector $\boldsymbol{f}_t$ accordingly.

***Reward** (r).* The reward is computed on the basis of the objective function defined in (3) and has the following form:

$$r(s_t, a_t) = \begin{cases} 1 - \lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{if } h_{\boldsymbol{\omega}}(\boldsymbol{x}_t) \neq h_{\boldsymbol{\omega}}(\boldsymbol{x}) \\ -\lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{otherwise,} \end{cases} \tag{5}$$

where $\ell_{\text{dist}}^t = \ell_{\text{dist}}(\boldsymbol{x}, \boldsymbol{x}_t)$ and $\lambda \in \mathbb{R}_{>0}$ is a parameter that controls how much weight to put over the distance between the current modified instance ($\boldsymbol{x}_t$) and the original sample ($\boldsymbol{x}$). In other words, the agent aims to reach a trade-off between achieving the CF prediction goal and keeping the distance of the counterfactual from the original input sample $\boldsymbol{x}$ as lower as possible.

***Policy** ($\pi_{\boldsymbol{\theta}}$).* We define a *parametrized* policy $\pi_{\boldsymbol{\theta}}$ to maximize the expected reward in the MDP problem. Our ultimate goal, however, is to find an optimal policy $\pi^* = \pi_{\boldsymbol{\theta}^*}$ that solves (3). It is worth noticing that finding the optimal policy π^* that maximizes the expected return in this environment is equivalent to minimizing (1), and thereby finding the optimal CF generator g^*. The equivalence between these two formulations can be shown as follows:

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \ell_{\text{pred}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x}); h_{\boldsymbol{\omega}}) + \lambda \ell_{\text{dist}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x})) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} -\ell_{\text{pred}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x}); h_{\boldsymbol{\omega}}) - \lambda \ell_{\text{dist}}(\boldsymbol{x}, g_{\boldsymbol{\theta}}(\boldsymbol{x})) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \begin{cases} 1 - \lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{if } h_{\boldsymbol{\omega}}(\boldsymbol{x}_t) \neq h_{\boldsymbol{\omega}}(\boldsymbol{x}) \\ -\lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{otherwise,} \end{cases} \\
&= \arg\max_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \sum_{t=1}^{T} r(s_t, \pi_{\boldsymbol{\theta}}(s_t)).
\end{aligned}$$

Here, T defines the maximum steps taken by the agent for each sample $\boldsymbol{x}$ and is set to 50,000. Finally, remember that we set ℓ_{dist} equal to L^1-norm, which is notoriously hard to optimize via gradient-based methods. However, in our DRL framework, this is incorporated into the reward function, which avoids us relying on its gradient.

B. Policy Optimization

We use the P-DQN framework [29] to find the optimal policy π^*. As mentioned before, at each time step the agent takes a hybrid action $a_t \in \mathcal{A}_t$ to perturb the currently modified $\boldsymbol{x}_t$, obtained from the original input $\boldsymbol{x}$.

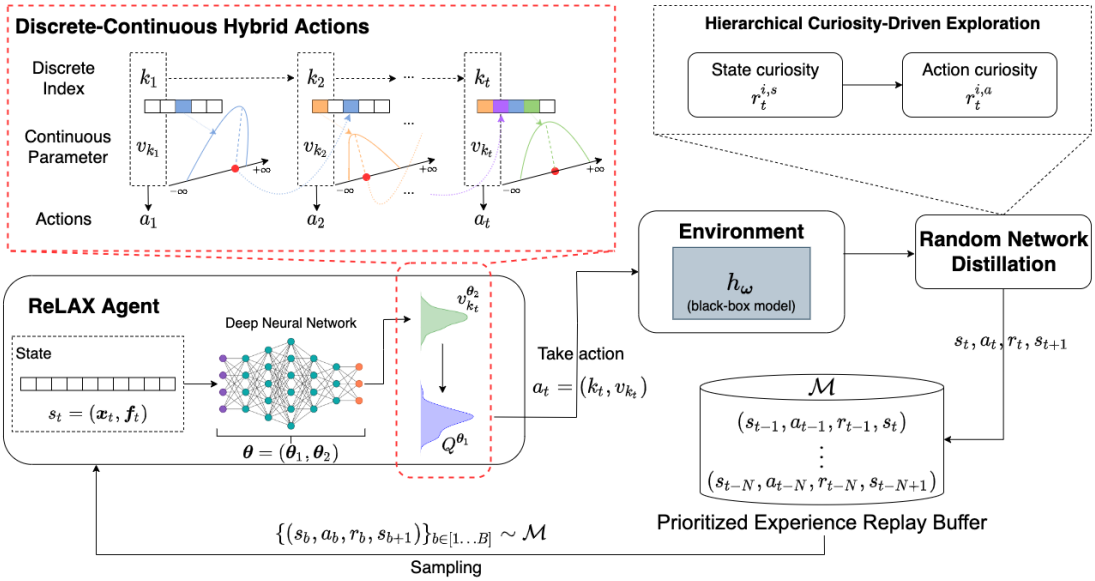

Fig. 2: Overview of our proposed RELAX framework.

In Q-learning, one aims at finding the optimal Q-value function representing the expected discounted reward for taking action a_t at a given state s_t. Inside the Q-value function, the continuous parameter v_{k_t} is associated with the discrete action k_t, which means v_{k_t} is the optimal action given state s_t and k_t: $v_{k_t} = \arg\sup_v Q(s_{t+1}, k_t, v)$. We therefore cast this as a function $v^Q_{k_t}(s_t)$. Thus, the Bellman equation can be written as:

$$Q(s_t, k_t, v_{k_t}) = \mathbb{E}_{r_t, s_{t+1}}(r_t + \gamma \max_{k_t} Q(s_{t+1}, k_t, v^Q_{k_t}(s_{t+1}))|s_t, a_t = (k_t, v_{k_t})).$$

As with DQN, a deep neural network (DNN) $Q^{\boldsymbol{\theta}_1}(s_t, k_t, v_{k_t})$ is used to approximate the Q-value function, and we fit $v^Q_{k_t}$ with another deterministic policy network $v^{\boldsymbol{\theta}_2}_{k_t}(s_t)$, where $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the parameters of the two DNNs. To find the optimal $\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2$, we minimize the following loss functions via stochastic gradient descent:

$$\mathcal{L}_Q(\boldsymbol{\theta}_1) = [Q^{\boldsymbol{\theta}_1}(s_t, k_t, v_{k_t}) - y_t]^2,$$

$$\mathcal{L}_\pi(\boldsymbol{\theta}_2) = -\sum_{k_t \in \mathcal{F}} Q(s_t, k_t, v^{\boldsymbol{\theta}_2}_{k_t}(s_t)),$$

where the y_t is the n-step target [41]. Theoretically, as the error diminishes, the Q network converges to the optimal Q-value function and the policy network outputs the optimal continuous action. However, this method is unstable in practice. This is because DQN only sample transition pairs uniformly from the replay buffer. Therefore, we adopt prioritized experience replay [42] to effectively learn from pairs with high expected learning value. As a measurement for learning potential, prioritized experience replay samples transition pairs with probability p_j based on their TD error: $p_t \propto |R_j + \gamma Q_{\text{target}}(s_j, Q(s_j, a_j)) - Q(s_{j-1}, a_{j-1})|^\beta$, where β is a parameter to determine how much prioritization is used. During the training procedure, the transition pairs are stored into the replay buffer with their priority, and a new data structure – i.e., a sum tree – is employed to efficiently update the priority and sample pairs from the replay buffer.

C. Hierarchical Curiosity-Driven Exploration

In the procedure of generating counterfactual examples, the sparse reward problem is inevitably met due to the large state and action space. Reward shaping is a typical solution that converts the sparse reward to a dense one [43] [44]; however, it is hard to design the intermediate rewards for all black box models. Hence, we develop a hierarchical curiosity-driven exploration mechanism to P-DQN.

Specifically, we use a *Random Network Distillation* (RND) module [45], i.e., a curiosity-driven approach to generate state curiosity $r^{i,s}_t$ by quantifying the next-state novelty, which will be further combined with the external rewards r_t to generate the modified reward $r'_t = r^{i,s}_t + r_t$. RND transforms the exploring procedure into a supervised learning task by minimizing the following mean squared error:

$$r^{i,s}_t = ||\hat{f}(s_t, \eta_1) - f(s_t)||^2,$$

where $f(s_t)$ is the fixed target network and $\hat{f}(s_t, \eta_1)$ is a trainable predictor network learning to distill the target network. The loss of trained input state (s_t) will decrease as the frequency of visited states increases; therefore, $r^{i,s}_t$ of novel states are expected higher.

Considering that the bonus is obtained after reaching the next state s_{t+1} and will be combined with the environment reward, the agent tends to follow the existing experiences rather than keeps exploring unknown states by taking various actions. To encourage the exploration of different actions at each state, we introduce the RND module to reflect the frequency of each action. At state s_t, the curiosity of a hybrid action $a_t = (k_t, v^{\boldsymbol{\theta}_2}_{k_t}(s_t))$ is given by:

$$r^{i,a}_t = ||\hat{g}(s_t, a_t, \eta_2) - g(s_t, a_t)||^2,$$

where g and $\hat{g}$ have the same role as f and $\hat{f}$, respectively.

Here, we leverage state curiosity $r_t^{i,s}$ to enhance high-level index planning in discrete spaces, and $r_t^{i,a}$ is adopted as low-level action curiosity for enthusiastically searching continuous parameters. Finally, we organically incorporate the two-level curiosity loss into the loss functions defined above in Section IV-B:

$$\mathcal{L}'_Q(\boldsymbol{\theta}_1, \eta_1) = [Q^{\boldsymbol{\theta}_1}(s_t, k_t, v_{k_t}) - y_t]^2 + ||\hat{f}(s_t, \eta_1) - f(s_t)||^2,$$

$$\mathcal{L}'_\pi(\boldsymbol{\theta}_2, \eta_2) = -\sum_{k_t \in \mathcal{F}} Q(s_t, k_t, v_{k_t}^{\boldsymbol{\theta}_2}(s_t)) + ||\hat{g}(s_t, a_t, \eta_2) - g(s_t, a_t)||^2.$$

It is worth remarking that a_t is a function of $\boldsymbol{\theta}_2$. By alternately updating $\boldsymbol{\theta}_2$ and η_2, the policy network $v_{k_t}^Q$ will learn to balance the action's potential value and novelty.

The overview of our proposed RELAX framework is depicted in Fig. 2.

D. Global vs. Local Policy

So far, we have considered one single agent as responsible for generating the CFs for *all* the instances of a given dataset, and hereinafter we refer to it as RELAX-GLOBAL. This allows us to learn a generalized policy that is able to produce counterfactuals, which are not tailored to a specific input sample. Algorithm 1 describes the training of RELAX-GLOBAL with the hierarchical curiosity-driven exploration technique discussed in Section IV-C above.

Algorithm 1 Training RELAX-GLOBAL Agent with Curiosity

$\boldsymbol{\theta}_1 \leftarrow$ *initialize the deep Q-network* $Q^{\boldsymbol{\theta}_1}$
$\boldsymbol{\theta}_2 \leftarrow$ *initialize the deterministic policy network* $v_{k_t}^{\boldsymbol{\theta}_2}$
$\eta_1 \leftarrow$ *initialize the RND state modules* $\hat{f}, f$
$\eta_2 \leftarrow$ *initialize the RND action modules* $\hat{g}, g$
$\mathcal{M} \leftarrow$ *initialize the replay buffer*
$i \leftarrow 1$
while $i \leq$max_epochs **do**
  $\boldsymbol{x} \sim \mathcal{D}$ ▷ Sample a training instance $\boldsymbol{x}$ from $\mathcal{D}$
  $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
  $s_0 = (\boldsymbol{x}_0, \boldsymbol{f}_0)$ ▷ Initial state
  $t \leftarrow 0$
  for $t \leq T$ **do** ▷ Maximum agent steps for each sample (T=50,000)
    $v_{k_t} \leftarrow v_{k_t}^{\boldsymbol{\theta}_2}(s_t)$ ▷ Compute the continuous parameter
    $a_t \leftarrow (k_t, v_{k_t})$ ▷ Select the discrete action by ε-greedy
    $r_t^{i,a} \leftarrow ||\hat{g}(s_t, a_t, \eta_2) - g(s_t, a_t)||^2$
      ▷ Generate action curiosity $r_t^{i,a}$
    $r_t, s_{t+1} \leftarrow \mathcal{T}(s_t, a_t)$
    ▷ The agent gets the reward and observes the next state
    $r_t^{i,s} \leftarrow ||\hat{f}(s_t, \eta_1) - f(s_t)||^2, \quad r'_t = r_t^{i,s} + r_t$
      ▷ Generate state curiosity $r_t^{i,s}$ and modified reward r'_t
    $p_t \leftarrow$ *compute the importance* p_t
    $\mathcal{M} \leftarrow (\{s_t\}, \{a_t\}, \{r'_t\}, \{s_{t+1}\}, \{p_t\})$
      ▷ Store transition into the replay buffer
    $B \sim \mathcal{M}$ ▷ Randomly sample batch B from $\mathcal{M}$
    $\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_1 - \gamma_1 \nabla \mathcal{L}'_Q(\boldsymbol{\theta}_1, \eta_1), \quad \eta_1 \leftarrow \eta_1 - \gamma_2 \nabla \mathcal{L}'_Q(\boldsymbol{\theta}_1, \eta_1)$
    $\boldsymbol{\theta}_2 \leftarrow \boldsymbol{\theta}_2 - \gamma_3 \nabla \mathcal{L}'_\pi(\boldsymbol{\theta}_2, \eta_2), \quad \eta_2 \leftarrow \eta_2 - \gamma_4 \nabla \mathcal{L}'_\pi(\boldsymbol{\theta}_2, \eta_2)$
      ▷ Update the parameters of both networks via SGD
    $t \leftarrow t + 1$
  end for
  $i \leftarrow i + 1$
end while
return $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ ▷ Optimal parameters of both networks

However, in some cases, the CF generation process should in fact capture the peculiarities of each individual original instance. To accomodate such a need, we introduce a variant to our proposed method, called RELAX-LOCAL.

RELAX-LOCAL trains a dedicated agent for crafting the optimal CF for a *single* target example. It starts by initializing RELAX-LOCAL's policy with one pretrained RELAX-GLOBAL's policy. Then, using a standard transfer learning approach [46], RELAX-LOCAL is fine-tuned on the target samples. This step consists of randomly generating synthetic training data points around the target example, using a method similar to [5]. Specifically, we uniformly sample data points whose Euclidean distance from the target example is at most equal to one.[3]

V. DISTILL INTERPRETABLE AGENT'S POLICY

The complex neural network structure of a DRL policy learned by RELAX for generating CFs poses a challenge for understanding and reasoning about the decision logic of the agent. To explain the decision process of the agent, we distill knowledge from the learned policy to a much simpler decision tree (DT) model. The DT approximates the policy's operation and resembles its internal logic, yet it is naturally interpretable as its tree-based structure reveals simple but comprehensive decision rules made by the DRL agent.[4]

Extracting a decision tree to interpret a complex neural network is an established practice, often conducted by a teacher-student training process [35], [36], [47]. However, we cannot simply adopt this strategy because the hybrid action space of our CF generation policy consists of both discrete actions and continuous parameters. Our approach, detailed in Algorithm 2 and depicted in Fig. 3, addresses this challenge and consists of the following two key steps.

1) Trajectories Collection. We sample trajectories of state-action pairs generated by a teacher DRL policy to train a student DT. The student model may encounter states that are not possible under the teacher's induced state distribution due to imperfect approximation. As in previous work [38], our solution to this problem is data augmentation. We iteratively sample a set of additional states by simulating the current DT model, and add the teacher's decision on these samples to the training dataset.

2) Decision Tree Training. At each training step of a student DT, following [36], we resample state-action pairs from the current (augmented) training dataset with probability proportional to state importance:

$$p((s,a)) \propto (Q^{\pi^*}(s, \pi^*(s)) - \min_{a' \in \mathcal{A}} Q^{\pi^*}(s, a')) \mathbb{1}_{\mathcal{D}}[(s,a)],$$

where π^* is the DRL teacher policy and $\mathcal{A}$ is the policy's parameterized action space. An action $a \in A$ includes both a discrete action and its continuous parameter. A state-action pair has a higher priority to be sampled if the action on the state significantly outweighs other actions, deemed by the Q-value function of the teacher policy π^*.

[3]Notice that all our input samples are normalized unit vectors.

[4]We remark that directly learning such tree models is infeasible as a tree structure cannot be updated by gradient-based optimization.

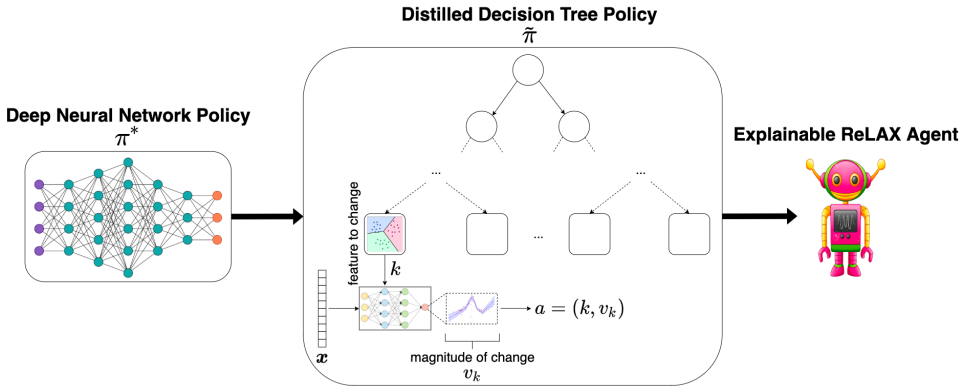

Fig. 3: Overview of our proposed RELAX agent's policy distillation framework.

We train a student DT in two steps to approximate discrete actions and their continuous parameters of a teacher policy, separately. Firstly, we learn a DT to predict DRL policy's discrete actions, where each leaf node indicates the feature k to change. Then, we train a two-layer Multi-Layer Perceptron for Regression (MLP-REG) to learn a function at each leaf node that takes as input the current state (the input $\boldsymbol{x}$ and the selected feature k) and predicts the continuous parameter for the discrete action indicated by the leaf node, i.e., the magnitude of the feature change v_k.

Algorithm 2 Extracting rules from the RELAX agent with hybrid action space

$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize dataset with empty set
$\hat{\pi}_0 \leftarrow \pi^*$ ▷ Initialize the policy with the agent's policy
for $i \leq N$ **do**
  $\mathcal{D}_i \leftarrow \{(s, \pi^*(s)) \sim d^{\hat{\pi}_{i-1}}\}$ ▷ Sample B trajectories
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ ▷ Update the dataset
  $\mathcal{D}' \leftarrow \{(s, k) \sim p(s, k)\}$ ▷ Resample from the dataset, where k is the discrete action
  Train decision tree $\hat{\pi}_i$
end for
$\tilde{\pi} \leftarrow$ `best_policy`$(\{\hat{\pi}_1, \ldots, \hat{\pi}_N\})$
for each leaf l in $\tilde{\pi}$ **do**
  Train a two-layer MLP-REG m_l to fit the continuous action v_k with s as the input.
end for
return $\tilde{\pi}, \{m_l : l \in \tilde{\pi}\}$ ▷ Return the decision tree $\tilde{\pi}$ for discrete action and a set of $\{m_l\}$ for continuous action.

As shown in Section VII-B, our experiments demonstrate that a decision tree distilled with Algorithm 2 can produce interpretable policies at the cost of a small performance degradation, which ranges from 5% to 11% of the original teacher policy's validity.

VI. EXPERIMENTS

In this section, we describe the experiments we conduct to validate our CF generation method. To demonstrate the effectiveness, efficiency, and flexibility of RELAX, we assess the quality of explanations extracted from several models learned either for classification or regression tasks.

A. Setup

Datasets and Tasks. We test with six public tabular datasets: *Breast Cancer*, *Credit Card Fraud*, *Diabetes*, *Sonar*, *Wave*, and *Boston Housing*. The first five are associated with classification tasks, whereas the last one with regression. A complete overview of the main properties of these data collections is available in Table I.

Dataset	N. of Instances	N. of Features	Task
Breast Cancer [48]	699	10 (numerical)	class.
Credit Card Fraud [49]	1,000,000	7 (mixed)	class.
Diabetes [50]	768	8 (numerical)	class.
Sonar [51]	208	60 (numerical)	class.
Wave [52]	5,000	21 (numerical)	class.
Boston Housing [53]	506	14 (mixed)	regr.

TABLE I: Main characteristics of the six public datasets used.

Predictive Models. Each dataset is randomly split into 70% training and 30% test portions. For each task and the associated dataset, we train the suitable set of predictive models chosen amongst the following: Random Forest (RF), Adaptive Boosting (ADABOOST), Gradient Boosting (XGBOOST), Multi-Layer Perceptron (MLP), and Multi-Layer Perceptron for Regression (MLP-REG). Both MLP and MLP-REG are fully-connected feed-forward neural networks with two hidden layers; MLP includes also a logistic (i.e., sigmoid) activation function at the last output layer. Notice that some combinations do not apply, e.g., MLP-REG is only trained on the *Boston Housing* dataset. We perform 10-fold cross validation on the training set portion of each dataset to fine-tune the hyperparameters of all the trainable models. Hence, for each task/dataset pair, we re-train all the applicable models with the best hyperparameters on the whole training set, and we measure their performance on the test set previously hold out. To assess the quality of the predictive models, we use accuracy for classification and RMSE for regression. Eventually, we consider only the best performing model(s) for each task/dataset pair. For example, for *Breast Cancer* we take into account two models, i.e., RF and MLP, which result the most accurate ones and behave similarly to each other. In Table II, we summarize the characteristics of each predictive model used in combination with the benchmarking datasets.

Counterfactual Generator Baselines. We compare RELAX with all the CF generator baselines described in Sec-

Dataset [Best Model]	**Structure**	**Acc.=▲/RMSE=●**
Breast Cancer [RF]	#trees=100	0.99 (▲)
Credit Card Fraud [XGBOOST]	#trees=100	0.98 (▲)
Diabetes [ADABOOST]	#trees=100	0.79 (▲)
Wave [XGBOOST]	#trees=100	0.95 (▲)
Breast Cancer [MLP]	L1=64; L2=128	1.00 (▲)
Sonar [MLP]	L1=256; L2=256	0.90 (▲)
Wave [MLP]	L1=100; L2=200	0.97 (▲)
Boston Housing [MLP-REG]	L1=50; L2=128	3.36 (●)

TABLE II: Model structure and performance for each dataset/task pair.

tion II. NEAREST-CT is considered the simplest approach. Furthermore, we distinguish between *model-specific* (tree-specific: FEATTWEAK and FOCUS; NN-specific: DEEPFOOL and GRACE), *gradient-aware* (DiCE), and *model-agnostic* (LORE and MACE).

Methodology. We compare the set of CF generation methods that are suitable for each dataset/target model in Table II. In particular, model-agnostic techniques (including both variants of our RELAX) clearly apply to every setting, whereas model-specific approaches can be tested only when the target model matches (e.g., FOCUS can be used only in combination with tree-based models). Eventually, we generate a separate CF for each input sample in the test set of every dataset above, using all the CF generators relevant to the setting at hand, i.e., all the model-agnostic methods, gradient-aware (if applicable), and model-specific techniques that apply.

Evaluation Metrics. We evaluate the quality of generated CFs according to the following four standard metrics [6]: *Validity*, *Proximity*, *Sparsity*, and *Generation Time*. Validity measures the ratio of CFs that actually meet the prediction goal to the total number of CFs generated;[5] the higher the validity the better. Proximity computes the distance of a CF from the original input sample; in this work, we use L^1-norm to measure proximity, and therefore the smaller it is the better. Sparsity indicates the number of features that must be changed according to a CF, and therefore is equivalent to the L^0-norm between a CF and the corresponding original input sample. The smaller it is the better, as sparser CFs likely lead to more human-interpretable explanations. Finally, Generation Time computes the time required to generate CFs, which, clearly, should be as small as possible. All the metrics above are averaged across all the test input samples. Moreover, experiments were repeated 5 times and results are expressed as the mean $\pm$ standard deviation.

B. Explainability Evaluation

Sparsity-Validity Trade-off. In Fig. 4, we plot the number of perturbed features (i.e., sparsity) versus the validity of counterfactuals obtained with different CF generation methods, when applied to classification tasks. More specifically, we fix a threshold on the maximum number of features that each CF generator is allowed to perturb and we show: (*i*) the *actual* sparsity; and (*ii*) the validity of the generated CFs. The rationale of this analysis is to show which method is able to achieve the best trade-off between two contrasting metrics: sparsity and validity. Intuitively, the larger is the number of perturbed features, the higher is the chance of obtaining a valid counterfactual. On the other hand, we should privilege sparser CFs, i.e., CFs that modify as less features as possible, since those require less effort and therefore are possibly more interpretable and feasible to implement.

Results show that both RELAX-GLOBAL and RELAX-LOCAL achieve the best balance between sparsity and validity. That is, our method outperforms all the baselines in terms of validity and, more importantly, it obtains these results even when we set a very restrictive threshold on sparsity, i.e., when few features are modified. As expected, though, if we soften such a cap on the number of features allowed to change, other methods like LORE may eventually match the performance of RELAX or even reach higher validity scores. In fact, not controlling for sparsity will make all CF generation methods behave similarly. This is what we observe when we test with DEEPFOOL or the simplest NEAREST-CT baseline, which by design tend to generate valid CFs only if a large fraction of the input features get modified. Due to this behavior, both DEEPFOOL and NEAREST-CT cannot be visualized in the plots, as they fall outside the range of sparsity values imposed in our experiments. Furthermore, although MACE is model-agnostic, its applicability to neural network target models is problematic due to its large computational cost [10]. Besides, the only implementation of MACE remaining available works only in combination with RF target models, and this is why we used it only in the first setting (*Breast Cancer* [RF]).

A similar analysis on the sparsity vs. validity trade-off for the *Boston Housing* regression task is shown in Table III. In this case, we compare only our two variants of RELAX since none of the CF generation baselines considered is designed to operate in a regression setting. As expected, the larger is the threshold δ used to determine if the prediction goal of the counterfactual example is met the harder is for RELAX to find a valid counterfactual.

	Validity (Sparsity)	
Thr. (δ)	RELAX-GLOBAL	RELAX-LOCAL
0.20	0.81 ± 0.09 ($\mathbf{3.02} \pm 0.17$)	$\mathbf{0.87} \pm 0.05$ (3.10 ± 0.18)
0.40	0.74 ± 0.06 ($\mathbf{3.09} \pm 0.16$)	$\mathbf{0.81} \pm 0.05$ (3.18 ± 0.16)
0.60	0.70 ± 0.06 ($\mathbf{3.21} \pm 0.12$)	$\mathbf{0.77} \pm 0.03$ (3.28 ± 0.09)

TABLE III: *Sparsity* vs. *Validity* of counterfactuals generated by RELAX for the *Boston Housing* regression task.

Finally, analogous conclusions can be drawn if we compare proximity vs. validity: RELAX is able to strike the best balance also between those two conflicting metrics.

Generation Time. Although validity and sparsity (proximity) are crucial to measure the quality of a CF generation method, efficiency is pivotal as well. Therefore, we compare the average generation time for each model-agnostic CF generator, namely LORE, MACE, and our RELAX in its two variants. We focus on model-agnostic methods because we want this comparison to be as general as possible. Table IV shows that our method takes up to 42% less time than other model-agnostic baseline to produce valid counterfactuals, which happen to be also closer to the original instances. This result

[5]Some work consider the complementary metric, which is known as *Fidelity* and is equal to (1-*Validity*).

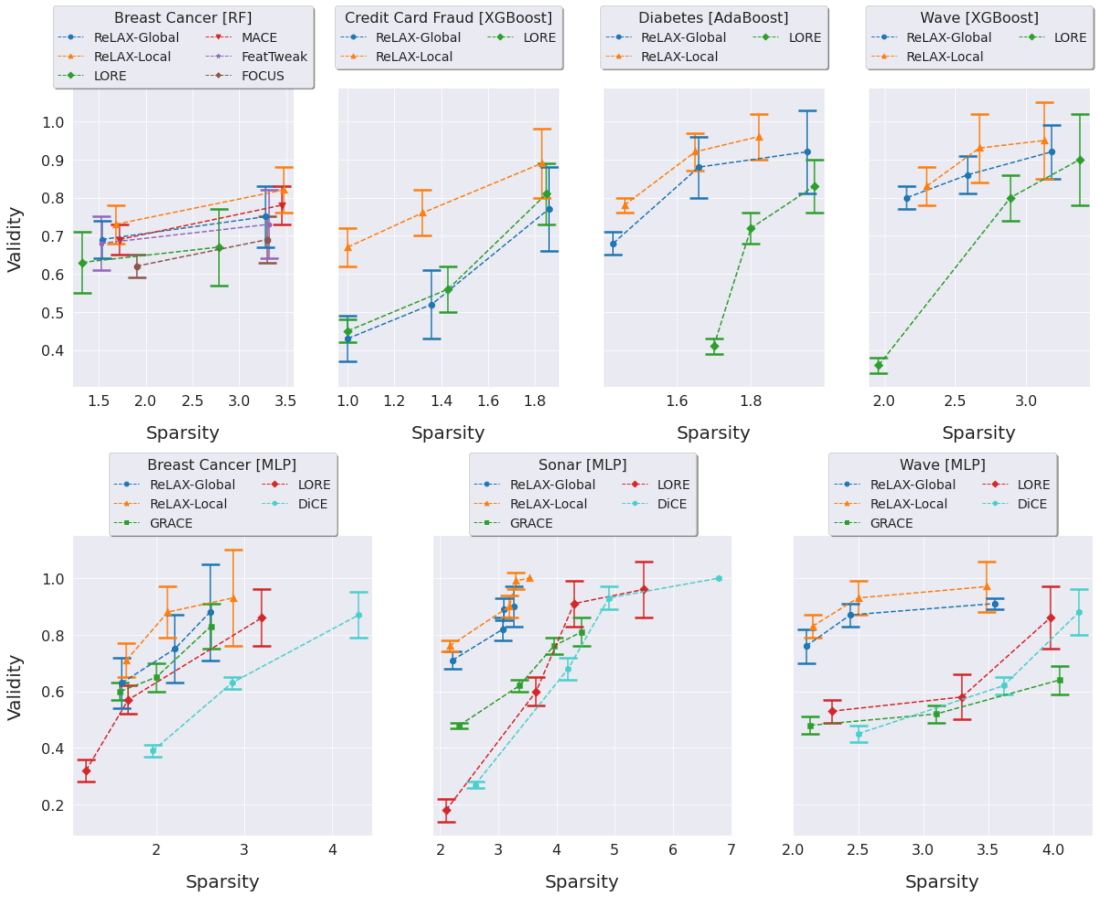

Fig. 4: *Sparsity* vs. *Validity* of counterfactuals generated by RELAX and other baselines for classification tasks.

is even more remarkable if we consider that the counterfactual generation time of RELAX includes the training time of the DRL agent.

We report the table with the complete results of all experiments in our GitHub repository.[6]

C. Hyperparameter Tuning

Our CF generation method is associated with a number of controlling parameters. To understand the impact of these parameters, we analyze the behavior of RELAX on the *Sonar* dataset with MLP as the target classifier and the maximal sparsity limited to 5 features.

The scaling factor λ**.** We first investigate the effect of the scaling factor λ on the generated CFs, picking it from $0.001, 0.01, 0.1, 1, 10$. As shown in Fig. 5, λ controls the balance between sparsity and validity. Indeed, larger values of λ force the agent to prefer sparser CFs at the expense of lower validity (see Fig. 5 – left), whereas smaller values of λ result in the opposite behavior (see Fig. 5 – right).

The target model's architecture. We assess the robustness of our CF generation method when applied to different target neural network architectures. More specifically, Table V shows the impact of different MLP architectures on the validity and sparsity of counterfactuals generated by RELAX in

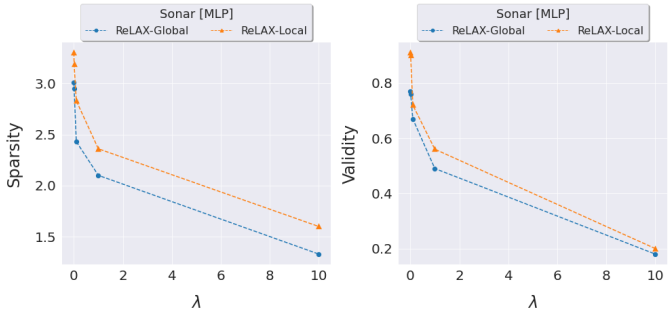

Fig. 5: The impact of λ on *Sparsity* (left) and *Validity* (right).

comparison with three competitors: GRACE (NN-specific), DiCE (gradient-aware), and LORE (model-agnostic). We may observe that GRACE, DiCE, and LORE are all sensitive to the MLP size, i.e., when the target neural network is getting large the validity and sparsity of CFs generated with those two methods deteriorate significantly. In the case of GRACE and DiCE, the reason for that detrimental effect is due to the fact that they leverage the gradient of the function approximated by the neural network, which is obviously correlated with the complexity of the MLP structure. Despite model-agnostic, LORE requires to accurately learn a locally-interpretable surrogate model of the target neural network, which may be hard when this becomes too complex. Eventually, RELAX is more robust toward different MLP structures, and its performance

[6] https://github.com/Mewtwo1996/ReLAX.git/eval/results.md

Metric	**Dataset [Models]**	**CF Generation Methods**			
		RELAX-GLOBAL	RELAX-LOCAL	LORE	MACE
Proximity	*Breast Cancer* [RF,MLP]	[**4.46**, 5.92]	[4.49, 5.87]	[4.63, **5.63**]	[4.47, N/A]
	Credit Card Fraud [XGBOOST]	[1.43]	[**1.38**]	[1.42]	[N/A]
	Diabetes [ADABOOST]	[**4.41**]	[4.50]	[4.76]	[N/A]
	Sonar [MLP]	[**7.32**]	[7.66]	[7.36]	[N/A]
	Wave [XGBOOST, MLP]	[**5.93**, **6.38**]	[6.02, 6.50]	[6.60, 6.41]	[N/A, N/A]
	Boston Housing [MLP-REG]	[**5.10**]	[5.36]	[N/A]	[N/A]
***Generation Time* (secs.)**	*	1500	**1320**	2100	2280

TABLE IV: Comparison of *Proximity* and *Generation Time* for model-agnostic CF generation methods.

		Validity (Sparsity)				
MLP size	**Accuracy**	RELAX-GLOBAL	RELAX-LOCAL	GRACE	DiCE	LORE
[256, 256]	0.88	0.76 (**2.95**)	**0.90** (3.19)	0.62 (3.32)	0.68 (4.20)	0.60 (3.65)
[128, 128]	0.85	0.80 (**2.69**)	**0.90** (2.88)	0.73 (2.85)	0.77 (2.71)	0.76 (3.21)
[64, 64]	0.79	0.90 (**1.88**)	**0.95** (1.93)	0.86 (1.90)	0.92 (2.11)	0.90 (2.52)

TABLE V: The impact of different MLP architectures on the *Validity* (*Sparsity*) of counterfactuals.

is quite consistent independently of the MLP size. Indeed, the agent underneath RELAX *truly* treats the target model as a black box regardless of its internal complexity.

The efficiency of pretraining. Finally, we show how pretrained RELAX-GLOBAL improves the performance of RELAX-LOCAL. More specifically, we train RELAX-LOCAL as described in Section IV-D, and compare it with another agent learned from scratch. Unsurprisingly, initializing RELAX-LOCAL with pretrained RELAX-GLOBAL reduces its CFs generation time, as shown in Table VI.

Setting	***Validity* (*Sparsity*)**	***Generation Time* (secs.)**
without pretraining	0.80 (3.27)	1132
with pretraining	**0.90** (**3.19**)	**500**

TABLE VI: Performance of RELAX-LOCAL with/without pretrained RELAX-GLOBAL.

VII. PRACTICAL IMPACT

In this section, we further validate the power of our CF generation method on two practical downstream tasks: *(i)* providing actionable recommendations and *(ii)* interpreting agent's policy explanations. For both tasks we consider two real-world use cases: ***Diabetes*** and ***COVID-19***. The former attempts to identify diabetic patients from a dataset of healthcare records. The latter aims to predict the risk of mortality due to the COVID-19 pandemic from a dataset of demographic information collected by country-level reports.

A. Providing Actionable Recommendations

Similarly to previous work on counterfactual explanations [19], [40], we show how RELAX-LOCAL (hereinafter, RELAX) can be used to generate meaningful *actionable* recommendations in the two real-world scenarios considered.

Before digging into details, we want to distinguish between correlation and causality in our target black box models to explain. Most ML models focus on accurately learning correlations between input features and output predictions, but lack in reasoning about cause-effect relations. In order to generate faithful explanations with RELAX, we follow the pipeline to build ML models without considering causality, which is a common practice adopted by other explanation methods. However, we will extend our method to capture causal relationships in the future work.

Use Case 1: Diabetes. We use the popular dataset collected from the National Institute of Diabetes and Digestive and Kidney Diseases. This contains 768 instances, each one labeled either as 1 (positive for diabetes) or 0 (negative for diabetes).

Here, we want to show that RELAX is able to provide suggestions of which features should be perturbed in order to turn a patient with diabetes into a healthy one. To achieve this goal, we first randomly split the dataset into two parts, 70% for training and 30% for testing. Then, we train the following models to learn the binary classifier for predicting diabetic patients: SVM, RF, ADABOOST, and XGBOOST. After running 10-fold cross validation, our best-performing classifier is ADABOOST with 100 trees, which is able to reach about 79% accuracy on the test set. As our task is to find the optimal counterfactual explanation for patients with diabetes, we focus on converting positive samples into negative ones. We first rank the features according to the importance score induced by the ADABOOST model, as depicted in Fig. 6 (left). Thus, we generate the CFs for 55 diabetic patients in the test set using our RELAX algorithm. It is worth recalling that we enforce our method to perturb *actionable* features only. Specifically, features like *Pregnancy*, *Pedigree*, and *Age* cannot be modified, since they are historical properties of the patient. Eventually, we consider the set of actionable features along with their associated constraints ($\uparrow$ = *increase*, $\downarrow$ = *decrease*), if any, as follows:

- ***Plasma Glucose (Plas)*** ($\downarrow$): Normal plasma glucose levels are defined as under 100 mg/dL during fasting and less than 140 mg/dL 2-hours postprandial. In practice, values from 100 mg/dl to 126 mg/dl are diagnostic of pre-diabetes.
- ***Skin Thickness (Skin)*** ($\downarrow$): Triceps skin fold thickness reflects the level of body fat. Higher values may lead to obesity, in turn, increasing the chances of diabetes.
- ***Insulin (Insu)***: 2-hour serum insulin (mu U/ml). According to the research, skin thickness is significantly related to duration of diabetes.
- ***Body Mass Index (BMI)***: BMI computed as

(weight in kg)/(height in m)^2. A higher BMI raises the risk of having uncontrolled diabetes and complications related to it.

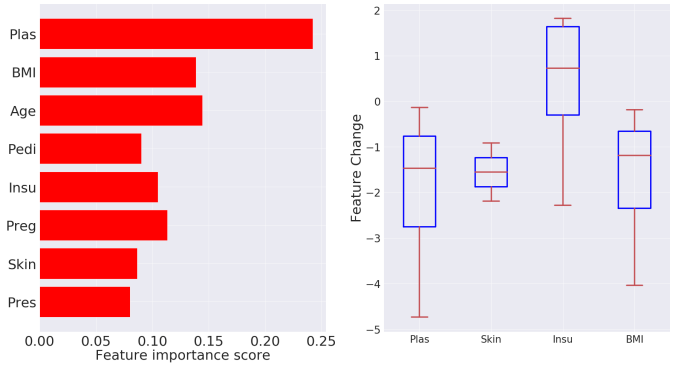

Fig. 6: Left: The ranking of features according to their importance. Right: Feature changes for transforming a diabetic patient into a healthy one.

The RELAX agent tweaks 1.49 features on average, and the average proximity (i.e., L^1-norm) between the original sample and the corresponding counterfactual example is 2.18. Fig. 6 (right) shows the direction of feature changes as suggested by CFs. Generally speaking, these recommendations (e.g., lowering the BMI) are compliant with those typically provided by domain knowledge experts [54] [55].

Use Case 2: COVID-19. In this scenario, the goal is to demonstrate that counterfactual explanations provided by RELAX may help countries that suffer from high COVID-19 mortality taking effective actions, economically and medically, to reduce such risk. In other words, we aim to show that our method can suggest which attributes should be changed (and to what extent) so that high-risk labeled instances can be converted to normal. To achieve that, we first need to learn a binary classifier that predicts whether the country's death risk due to COVID-19 is high or not, given the country-level features. This, in turn, requires collecting and labeling a dataset to train predictive models on.

Inspired by previous work addressing a similar task [56], we gather 17 country-level demographic features from 156 countries across 7 continents [57]–[60]. Furthermore, we label each instance with a binary class denoting the COVID-19 risk of mortality ("high" vs. "normal"), as in [56]. This dataset is made publicly available.[7]

We randomly split our collected dataset into two parts, i.e., 70% used for training and 30% for test. Therefore, we train the following models to learn the binary classifier for predicting COVID-19 mortality risk: SVM, RF, ADABOOST, and XGBOOST. After running 10-fold cross validation, the best-performing model turns out to be XGBOOST with 500 trees, which achieves 85% accuracy on the test set. We sort the features according to the ranking induced by the XGBOOST model, as shown in Fig. 7 (left). Then, we generate with our RELAX algorithm the CFs for the 16 high-risk countries in the test set. It is worth remarking that, although the target binary classifier is learned considering *all* the features from the whole training set, we force RELAX to tweak only the subset of *actionable* features so that effective suggestions can be found by observing the generated CFs. Besides, in order to avoid bizarre recommendations, we constrain the change of (actionable) features toward a "plausible" direction (e.g., it would not make any sense to suggest increasing the unemployment rate of a country). Below is the set of actionable features along with their associated constraints (↑ = *increase*, ↓ = *decrease*), if any:

- ***Death Rate*** (↓): Measures the number of overall deaths per 1,000 mid-year population before 2020 (i.e., it does *not* include deaths due to COVID-19).
- ***Unemployment Rate (UR)*** (↓): Represents the number of unemployed people as a percentage of the labor force; high unemployment is usually associated with lower living standards.
- ***Doctors per 10,000*** (↑): Indicates the number of physicians per 10,000 people.
- ***Nurses per 10,000*** (↑): Counts the number of nursing and midwifery personnel per 10,000 people.
- ***Urban Population Rate:*** Measures the ratio of people who live in urban areas over the total population, which could be adjusted via a series of rational policies.
- ***Obesity Prevalence:*** Denotes the percentage of a country's population considered to be obese.

The RELAX agent tweaks 1.67 features on average, and the average proximity (i.e., L^1-norm) between the original sample and the corresponding counterfactual example is 1.18. Fig. 7 (right) shows the direction of feature changes as suggested by the generated CFs.

Generally speaking, the CFs provide reasonable actions to take by exploiting the domain knowledge learned by the black-box model. In particular, we observe that the CFs suggest five main changes: *(i)* Decreasing the death rate;[8] *(ii)* Decreasing the unemployment rate; *(iii)* Increasing the nurse rate per 10,000 people; *(iv)* Decreasing the urban population rate; and *(v)* Decreasing the obesity prevalence. Not only those recommendations look sensible, as much as straightforward or even obvious, but their accuracy is also confirmed by the fact that many countries have indeed adopted similar strategies to counter the impact of COVID-19. For example, US approved the visa for more than 5,000 international nurses to strengthen the health workforce.[9] Moreover, according to the investigation made by the US Center for Disease Control and Prevention,[10] obese patients with COVID-19 aged 18 years and younger are associated with a 3.07 times higher risk of hospitalization and a 1.42 times higher risk of severe illness. Thus, reducing the obesity prevalence can indeed lower mortality risk. Finally, reducing the unemployment rate, as well as the urban population rate, may allow a wider range of people to enhance their social community awareness, take

[7] https://github.com/Mewtwo1996/ReLAX.git/use-cases/covid-19/dataset/covid-19.csv

[8] Albeit it may sound odd, reducing the death rate can subsume a broader suggestion for improving the life quality of a country.

[9] https://www.npr.org/sections/health-shots/2022/01/06/1069369625/short-staffed-and-covid-battered-u-s-hospitals-are-hiring-more-foreign-nurses

[10] https://www.cdc.gov/obesity/data/obesity-and-covid-19.html

consciousness of the risks of the pandemic, and ultimately adopt a safe lifestyle.

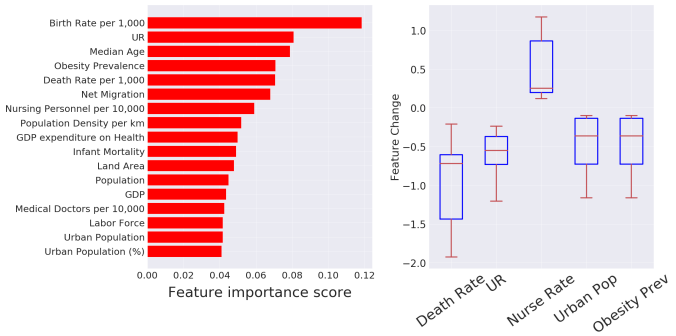

Fig. 7: Left: The ranking of features according to their importance. Right: Feature changes for transforming a high mortality risk country into a normal one.

B. Interpreting Agent's Policy Explanations

Using Algorithm 2 described in Section V, we distill decision rules to interpret the counterfactual explanations learned by the RELAX-GLOBAL (hereinafter, RELAX) agent's policy on the two use cases considered.

Use Case 1: Diabetes. We first train a specific DRL agent via RELAX to generate CFs for positive samples (i.e., patients with diabetes). The validity of this method is 0.90, whereas sparsity measures 1.86. Then, we distill the agent to a hybrid decision tree to generate CFs. The validity degradation of the tree model is within 5% of the original neural network policy. Fig. 8 displays the top-4 layers of the distilled decision tree.

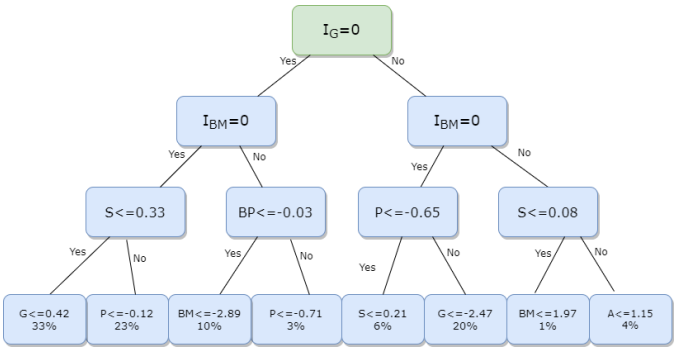

Fig. 8: Top-4 layers of the distilled decision tree (***Diabetes***).

The decision variable of each node contains *Glucose* (G), *Blood Pressure* (BP), *Skin Thickness* (S), *Body Mass Index* (BM), *Pedigree* (P), *Age* (A), and their indicator variables: I_G, I_{BP}, I_S, I_{BM}, I_P and I_A, which denote if the corresponding variable has been used. Also, we show the frequency of final decisions of each tree path on the fourth layer.

Thus, we are able to interpret the logic behind counterfactual explanations generated by RELAX using the distilled decision tree, along two directions: *(i) capturing existing knowledge* and *(ii) discovering new rules*. Concerning *(i)*, we observe that the decision variables used to split the branch at each node in the first three layers (i.e., G, BM, S, and BP) are known to be indeed relevant for determining diabetic condition, as reported in previous work [19]. With the information provided by these variables, our DRL agent will select the target variable to modify. Regarding *(ii)*, instead, we notice that the decision tree takes multiple variables into consideration, rather than relying on a single variable like [19]. On the top four layers, 6 variables collectively classifies input into 16 branches. For instance, when $I_G = 0$ and $I_{BM} = 0$, the condition of *Skin Thickness* and *Glucose* jointly exert strong impact on the severity of diabetes. Moreover, although only modifying actionable features, the tree model also illustrates how non-actionable features affect a final decision. In this tree, the value of *Pedigree* is also considered as a decision node on the third layer since it affects the Q-value function significantly.

Use Case 2: COVID-19. As with the previous use case, we first train a specific DRL agent via RELAX to generate CFs for positive samples (i.e., countries with high mortality rate due to COVID-19). The validity of the method is 0.78, and its sparsity measures 1.98. Then, a hybrid decision tree is adopted to distill the agent's policy. The validity degradation of the tree model is within 11% of the original neural network policy. Fig. 9 depicts the top-4 layers of the distilled decision tree.

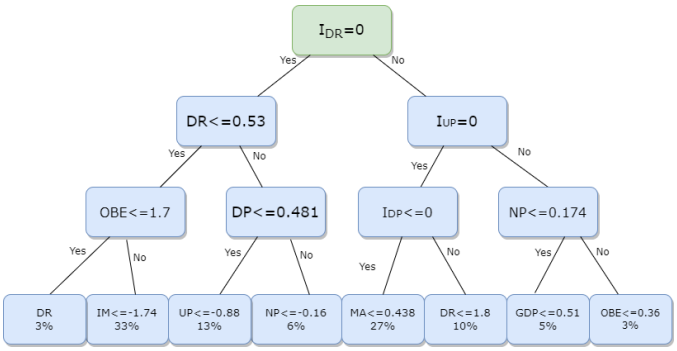

Fig. 9: Top-4 layers of the distilled decision tree (***COVID-19***).

The decision variable of each node contains *Death Rate* (DR), *Unemployment Rate* (UR), *Obesity Prevalence* (OBE), *Urban Population Rate* (UP), *Doctor per 10,000* (DP), *Nurse per 10,000* (NP), *Infant Mortality* (IM), *Median Age* (MA), *GDP*, along with their indicator variables: I_{DR}, I_{UR}, I_{DP}, which denote if the corresponding variable has been used. Also, we show the frequency of final decisions of each tree path on the fourth layer.

We interpret the logic behind the RELAX agent along with the same two axes considered for the previous use case: *(i) capturing existing knowledge* and *(ii) discovering new rules*. Concerning *(i)*, as depicted in the decision tree, many influential factors are incorporated. Specifically, within each decision node of the first three layers, *Unemployment Rate*, *Death Rate*, *Obesity Prevalence*, *Nurse per 10,000*, and *GDP* are used to split the branches. According to [56], we find that these geopolitical and demographic attributes at the country-level exert strong impact on generating the CFs. Regarding *(ii)*, the decision tree serves as a strong tool to discover interaction among independent variables. Variables that appear together in a traversal path are interacting with one another. For instance, on the top 4 layers, *Death Rate* (DR), *Obesity Prevalence* (OBE) and *Doctor per 10,000* (DP) jointly work to split the

branches. In other words, whether OBE and DP affect the CFs depend on the condition of DR. Based on this observation, we recommend the institutions or governments could decrease the COVID-19 risk of mortality by focusing on these three specific factors, and further investigating their relation.

VIII. Conclusion and Future Work

In this work, we presented ReLAX, the first method for generating model-agnostic counterfactual examples based on deep reinforcement learning with hierarchical curiosity-driven exploration. We implemented two variants of it: ReLAX-Global and ReLAX-Local. The former learns a generalized agent's policy from a whole set of training instances, whereas the latter trains a dedicated agent's policy for crafting the optimal counterfactual of a single target example via transfer learning from a pretrained ReLAX-Global. Extensive experiments run on six public tabular datasets demonstrated that ReLAX significantly outperforms all the considered counterfactual generation baselines in every standard quality metric. Our method is scalable with respect to the number of features and instances, and can explain *any* black-box model, regardless of its internal complexity and prediction task (i.e., classification or regression). Moreover, we proposed a distillation algorithm that extracts decision rules from the ReLAX agent's policy in the form of an interpretable decision tree, thereby making the counterfactuals generation process itself explainable. Finally, we showed the ability of ReLAX to provide actionable recommendations and distill interpretable policy explanations in two practical, real-world use cases.

In future work, we plan to investigate how ReLAX can be extended to generate explanations for non-tabular input data, such as images and text, and how to enforce counterfactual examples laying on the original data manifold.

Acknowledgment

This research was supported by the Italian Ministry of Education, University and Research (MIUR) under the grant "Dipartimenti di Eccellenza 2018-2022" and the XJTLU Research Development Fund under RDF-21-01-053, TDF21/22-R23-160, and AI Empowerment Tech. Inc. Research Fund under RDS10120220021.

References

[1] EU, "Regulation (EU) 2016/679 of the European Parliament (GDPR)," *Official Journal of the European Union*, vol. L119, pp. 1–88, 2016.

[2] P. Bracke, A. Datta, C. Jung, and S. Sen, "Machine Learning Explainability in Finance: An Application to Default Risk Analysis," Aug. 2019.

[3] M. A. Ahmad, C. Eckert, and A. Teredesai, "Interpretable Machine Learning in Healthcare," in *Proc. of BCB, Washington, DC, USA*. ACM, 2018, pp. 559–560.

[4] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[5] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local Rule-Based Explanations of Black Box Decision Systems," *arXiv preprint arXiv:1805.10820*, May 2018.

[6] S. Verma, J. Dickerson, and K. Hines, "Counterfactual Explanations for Machine Learning: A Review," *arXiv preprint arXiv:2010.10596*, 2020.

[7] I. Stepin, J. M. Alonso, A. Catala, and M. Pereira-Fariña, "A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence," *IEEE Access*, vol. 9, pp. 11 974–12 001, 2021.

[8] K. Kanamori, T. Takagi, K. Kobayashi, and H. Arimura, "DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization," in *Proc. of IJCAI*. IJCAI.org, 2020, pp. 2855–2862.

[9] A. Karimi, B. J. von Kügelgen, B. Schölkopf, and I. Valera, "Algorithmic Recourse under Imperfect Causal Knowledge: A Probabilistic Approach," in *Proc. of NeurIPS*, 2020, pp. 265–277.

[10] A. Karimi, G. Barthe, B. Balle, and I. Valera, "Model-Agnostic Counterfactual Explanations for Consequential Decisions," in *Proc. AISTATS, online, Palermo, Sicily, Italy*, vol. 108. PMLR, 2020, pp. 895–905.

[11] C. Russell, "Efficient Search for Diverse Coherent Explanations," in *Proc. of FAT**. ACM, 2019, pp. 20–28.

[12] B. Ustun, A. Spangher, and Y. Liu, "Actionable Recourse in Linear Classification," in *Proc. of FAT**. ACM, 2019, pp. 10–19.

[13] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR," *Harvart Journal of Law & Tech.*, vol. 31, p. 841, 2017.

[14] M. Pawelczyk, K. Broelemann, and G. Kasneci, "Learning Model-Agnostic Counterfactual Explanations for Tabular Data," in *Proc. of TheWebConf*. ACM/IW3C2, 2020, pp. 3126–3132.

[15] R. McGrath, L. Costabello, C. L. Van, P. Sweeney, F. Kamiab, Z. Shen, and F. Lécué, "Interpretable Credit Application Predictions With Counterfactual Explanations," *CoRR*, vol. abs/1811.05245, 2018. [Online]. Available: http://arxiv.org/abs/1811.05245

[16] A. V. Looveren and J. Klaise, "Interpretable Counterfactual Explanations Guided by Prototypes," *CoRR*, vol. abs/1907.02584, 2019. [Online]. Available: http://arxiv.org/abs/1907.02584

[17] S. Dandl, C. Molnar, M. Binder, and B. Bischl, "Multi-Objective Counterfactual Explanations," in *Proc. of PPSN*, ser. LNCS, vol. 12269. Springer, 2020, pp. 448–469.

[18] T. Le, S. Wang, and D. Lee, "GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction," in *Proc. of KDD*. ACM, 2020, pp. 238–248.

[19] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas, "Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking," in *Proc. of KDD*. ACM, 2017, pp. 465–474.

[20] A. Lucic, H. Oosterhuis, H. Haned, and M. de Rijke, "FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles," in *Proc. of AAAI*, ser. AAAI-22, 2022.

[21] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks," in *Proc. of CVPR*. IEEE Computer Society, 2016, pp. 2574–2582.

[22] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, "CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks," in *Proc. of AISTATS*, vol. 151. PMLR, 2022, pp. 4499–4511.

[23] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations," in *Proc. of FAT**. ACM, 2020, pp. 607–617.

[24] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement Learning with Parameterized Actions," in *Proc. of AAAI*. AAAI Press, 2016, pp. 1934–1940.

[25] A. A. Sherstov and P. Stone, "Function Approximation via Tile Coding: Automating Parameter Choice," in *International Symposium on Abstraction, Reformulation, and Approximation*. Springer, 2005, pp. 194–205.

[26] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork," in *Proc. of AAMAS-ALA Workshop*, 2016.

[27] M. Khamassi, G. Velentzas, T. Tsitsimis, and C. Tzafestas, "Active Exploration and Parameterized Reinforcement Learning Applied to a Simulated Human-Robot Interaction Task," in *Proc. of IRC*. IEEE, 2017, pp. 28–35.

[28] E. Wei, D. Wicke, and S. Luke, "Hierarchical Approaches for Reinforcement Learning in Parameterized Action Space," in *AAAI Spring Symposium Series*, 2018, pp. 358–364.

[29] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu, "Parametrized Deep Q-networks Learning: Reinforcement Learning with Discrete-Continuous Hybrid Action Space," *CoRR*, vol. abs/1810.06394, 2018. [Online]. Available: http://arxiv.org/abs/1810.06394

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," in *Proc. of ICLR*, 2016.

[32] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, "Programmatically Interpretable Reinforcement Learning," in *Proc. of ICML*. PMLR, 2018, pp. 5052–5061.

[33] A. Verma, H. M. Le, Y. Yue, and S. Chaudhuri, "Imitation-Projected Programmatic Reinforcement Learning," in *Proc. of NeurIPS*, 2019, pp. 15 726–15 737.

[34] H. Zhu, Z. Xiong, S. Magill, and S. Jagannathan, "An Inductive Synthesis Framework for Verifiable Reinforcement Learning," in *Proc. of PLDI*. ACM, 2019, pp. 686–701.

[35] N. Frosst and G. E. Hinton, "Distilling a Neural Network Into a Soft Decision Tree," in *Proc. Workshop on Comprehensibility and Explanation in AI and ML*, vol. 2071. CEUR-WS.org, 2017.

[36] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable Reinforcement Learning via Policy Extraction," *CoRR*, vol. abs/1805.08328, 2018. [Online]. Available: http://arxiv.org/abs/1805.08328

[37] M. Vasic, A. Petrovic, K. Wang, M. Nikolic, R. Singh, and S. Khurshid, "MoËT: Interpretable and Verifiable Reinforcement Learning via Mixture of Expert Trees," *CoRR*, vol. abs/1906.06717, 2019. [Online]. Available: http://arxiv.org/abs/1906.06717

[38] S. Ross, G. J. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Proc. of AISTATS*, vol. 15. JMLR.org, 2011, pp. 627–635.

[39] T. Spooner, D. Dervovic, J. Long, J. Shepard, J. Chen, and D. Magazzeni, "Counterfactual Explanations for Arbitrary Regression Models," *CoRR*, vol. abs/2106.15212, 2021. [Online]. Available: https://arxiv.org/abs/2106.15212

[40] G. Tolomei and F. Silvestri, "Generating Actionable Interpretations from Ensembles of Decision Trees," *IEEE TKDE*, vol. 33, no. 4, pp. 1540–1553, 2021.

[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[42] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *CoRR*, vol. abs/1511.05952, 2015. [Online]. Available: https://arxiv.org/abs/1511.05952

[43] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping," *Advances in NeurIPS*, vol. 33, pp. 15 931–15 941, 2020.

[44] M. Grzes and D. Kudenko, "Theoretical and Empirical Analysis of Reward Shaping in Reinforcement Learning," in *Proc. of ICMLA*. IEEE, 2009, pp. 337–344.

[45] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation," *CoRR*, vol. abs/1810.12894, 2018. [Online]. Available: https://arxiv.org/abs/1810.12894

[46] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *Proc. of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[47] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting CNNs via Decision Trees," in *Proc. of CVPR*. IEEE, 2019, pp. 6261–6270.

[48] "Breast Cancer Dataset," [Online]. Available from: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic), 1995.

[49] "Credit Card Fraud," [Online]. Available from: https://www.kaggle.com/datasets/dhanushnarayananr/credit-card-fraud, 2022.

[50] "Diabetes Dataset," [Online]. Available from: https://www.kaggle.com/uciml/pima-indians-diabetes-database, 1988.

[51] "Sonar Dataset," [Online]. Available from: https://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks), 1988.

[52] "Wave Dataset," [Online]. Available from: https://archive.ics.uci.edu/ml/datasets/waveform+database+generator+(version+1), 1988.

[53] "Boston Housing Dataset," [Online]. Available from: https://www.kaggle.com/vikrishnan/boston-house-prices, 1978.

[54] J. G. Derraik, M. Rademaker, W. S. Cutfield, T. E. Pinto, S. Tregurtha, A. Faherty, J. M. Peart, P. L. Drury, and P. L. Hofman, "Effects of Age, Gender, BMI, and Anatomical Site on Skin Thickness in Children and Adults with Diabetes," *PLoS One*, vol. 9, no. 1, p. e86637, 2014.

[55] A. Tirosh, I. Shai, D. Tekes-Manova, E. Israeli, D. Pereg, T. Shochat, I. Kochba, and A. Rudich, "Normal Fasting Plasma Glucose Levels and Type 2 Diabetes in Young Men," *New England Journal of Medicine*, vol. 353, no. 14, pp. 1454–1462, 2005.

[56] J. J. Bird, C. M. Barnes, C. Premebida, A. Ekárt, and D. R. Faria, "Country-level Pandemic Risk and Preparedness Classification based on COVID-19 Data: A Machine Learning Approach," *PLoS One*, vol. 15, no. 10, p. e0241332, 2020.

[57] Worldometers, "Current World Population," 2013.

[58] Central Intelligency Agency, "The CIA World Factbook 2020," 2020.

[59] World Health Organization, "Global Health Workforce Statistics," 2020.

[60] ——, "Prevalence of Obesity among Adults," 2017.

Ziheng Chen is currently a Ph.D. candidate in the Department of Applied Mathematics and Statistics at Stony Brook University, USA. He received his B.Sc. degree in statistics from the Renmin University of China in 2016. His research interests include explainable machine learning, reinforcement learning, recommender systems, tree structure models, ensemble learning, and healthcare applications.

Fabrizio Silvestri is a full professor at DIAG of the Sapienza University of Rome. His research interests lie in artificial intelligence, and in particular, machine learning applied to web search problems and natural language processing. He authored more than 150 papers in international journals and conference proceedings. It holds nine industrial patents. At Facebook AI, Fabrizio Silvestri has directed research groups to develop artificial intelligence techniques to combat malicious actors who use the Facebook platform for malicious purposes (hate speech, misinformation, terrorism, and so on). Fabrizio Silvestri has a Ph.D. in computer science from the University of Pisa.

Gabriele Tolomei is an associate professor at the Department of Computer Science of the Sapienza University of Rome. He received his M.Sc. in Computer Science in 2005 from the University of Pisa and his Ph.D. in Computer Science in 2011 from the Ca' Foscari University of Venice. He was a research scientist at Yahoo Labs in London and an assistant professor at the Department of Mathematics at the University of Padua. His main research interests are in (human-)explainable, robust, and collaborative machine learning. He authored more than 40 papers on peer-reviewed international conferences and journals, and he is the inventor of four US patents.

Jia Wang is a lecturer at the Department of Intelligent Science of the Xi'an Jiaotong-Liverpool University. She received her B.S. in Communication Engineering in 2012 from Beijing Jiao Tong University, her M.Sc. in 2015 from KTH Royal Institute of Technology, and her Ph.D. in Computer Science in 2020 from The Hong Kong Polytechnic University. She was a visiting student at the University of Southern California in 2019. Her research interests are in data mining, machine learning, game theory, and multi-agent systems.

He Zhu is an assistant professor at the Department of Computer Science of Rutgers University. He received his Ph.D. in Computer Science in 2016 from Purdue University. His main research interests are program synthesis, neurosymbolic programming, reinforcement learning, automated reasoning, and formal methods.

Hongshik Ahn is a full professor at the Department of Applied Mathematics and Statistics of Stony Brook University. He received his Ph.D. in Statistics in 1992 from the University of Wisconsin-Madison, MA in Statistics in 1987 from the University of California at Berkeley, and BS in Mathematics from Seoul National University, South Korea. He was a statistician at the National Center for Toxicological Research, US FDA, from 1992 to 1996. His main research interests are machine learning, classification of high-dimensional data, survival analysis, and bioinformatics. He authored two books, three book chapters, and 75 papers in peer-reviewed international journals.