**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 and then find the optimal CFs via deep reinforcement learning (DRL) with discrete-continuous hybrid action space. Extensive experiments conducted on several 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, to demonstrate the usefulness of our method in a real-world use case, we leverage CFs generated by ReLAX to suggest actions that a country should take to reduce the risk of mortality due to COVID-19. Interestingly enough, the actions recommended by our method correspond to the strategies that many countries have actually implemented to counter the COVID-19 pandemic.

**CCS CONCEPTS**

- Computing methodologies → Reinforcement learning: Sequential decision making. Machine learning: Artificial intelligence.

**KEYWORDS**

machine learning explainability, explainable AI, counterfactual explanations, deep reinforcement learning.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM ’22, October 17–21, 2022, Atlanta, GA, USA © 2022 Association for Computing Machinery. ACM ISBN 978-1-4503-9236-5/22/10 ... $15.00 https://doi.org/10.1145/3511808.3557429

1 We voluntarily left this notion of “closeness” underspecified here; a more precise definition of it is provided in Section 3.
model’s internals (e.g., random forests or neural networks), or they work only with differentiable models, or finally, they depend on each sample’s neighborhood. Thus, they are hard to generalize for more complex models and inefficient for large datasets.

To overcome these limitations, we introduce a novel CF generation method called 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). 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.

Our proposed method 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.

We validate our method on five datasets (four 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 show that CFs generated by ReLAX can be leveraged to suggest actions that a country should take to reduce the risk of mortality due to the COVID-19 pandemic.

Overall, the main contributions of this work are as follows:

- ReLAX is the first method for generating model-agnostic counterfactual examples 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).
- 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.
- To overcome the sparse reward problem due to the large state and action space, we integrate a hierarchical curiosity-driven exploration mechanism into our DRL agent.

- We further assess the power of ReLAX on a real-world use case. More specifically, we show that the actions suggested by ReLAX to reduce the risk of mortality due to the COVID-19 pandemic correspond to the strategies that many countries have actually put in practice.
- The source code implementation of ReLAX and the datasets (including the one used for the COVID-19 risk of mortality task) are made publicly available.2

The remainder of the paper is organized as follows. In Section 2, we review related work. In Section 3, we formalize the problem of generating counterfactual explanations, while in Section 4 we present ReLAX, our proposed method to solve this problem using deep reinforcement learning. We validate our approach and discuss the main findings of our work in Section 5. In Section 6, we further demonstrate the practical impact of ReLAX on a real-world use case. Finally, Section 7 concludes the paper.

2 https://github.com/Mewtwo1996/ReLAX.git
sequence of satisfiability problems, where both the distance function (objective) and predictive model (constraints) are represented as logic formulae. In addition, Mothilal et al. [28] present DiCE, a framework for generating and evaluating a diverse and feasible set of counterfactual explanations, which assumes knowledge of the model’s gradients is available, thus not fully model-agnostic.

Our ReLAX method is totally model-agnostic and aims to be as general and flexible as possible. Moreover, different from existing model-agnostic methods, ReLAX is much more efficient to generate 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, in contrast with ReLAX, both LORE and MACE do not consider nor control over the sparsity of the generated CFs; moreover, LORE does not even take into account their actionability.

2.2 Reinforcement Learning with Parameterized Action Space

Many real-world reinforcement learning (RL) problems require complex controls with discrete-continuous hybrid action space. For example, in Robot Soccer [25], 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 [35], or convert a discrete action into a continuous action-parameter space can be represented as: \( \mathcal{A} = \bigcup_{k \in V_k} \{(a_k, q_k) \mid a_k \in A, q_k \in V_k\} \). Masson et al. [25] 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. [19] 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. [45] 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. [49] introduce P-DQN, which seamlessly combines and integrates both DQN [26] and DDPG [21]. Empirical study indicates that P-DQN is efficient and robust.

3 PROBLEM FORMULATION

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

The idea of counterfactual explanations is to reveal the rationale behind predictions made by \( h_\omega \) on individual inputs \( x \) by means of counterfactual examples (CFs). More specifically, for an instance \( x \), a CF \( \tilde{x} \neq x \) according to \( h_\omega \) is found by perturbing a subset of the features of \( x \), chosen from the set \( F \subseteq \{1, \ldots, n\} \). The general goal of such modification is to transform \( x \) into \( \tilde{x} \) so to change the original prediction, i.e., \( h_\omega(x) \neq h_\omega(\tilde{x}) \) [43]. In particular, this depends on whether \( h_\omega \) is a classifier or a regressor. In the former case, the objective would be to transform \( x \) into \( \tilde{x} \) so to change the original predicted class \( c = h_\omega(x) \) into another \( \tilde{c} = h_\omega(\tilde{x}) \), such that \( \tilde{c} \neq c \). Notice that \( \tilde{c} \) can be either specified upfront (i.e., targeted CF) or it can be any \( \tilde{c} \neq c \) (i.e., untargeted CF). In the case \( h_\omega \) is a regressor, instead, the goal is trickier: one possible approach to specifying the validity of a counterfactual example \( \tilde{x} \) is to set a threshold \( \delta \in \mathbb{R} \setminus \{0\} \) and \( |h_\omega(\tilde{x}) - h_\omega(x)| \geq \delta \). However, CFs found via such a thresholding are sensitive to the choice of \( \delta \) [36].

Either way, as long as the CF classification or regression goal is met, several CFs can be generally found for a given input \( x \). 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 \( \tilde{x} \) as the “closest” one to \( x \). Intuitively, this is to favor CFs that require the minimal perturbation of the original input.

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

\[
L_{CF}(g_\theta; D, h_\omega) = \frac{1}{|D|} \sum_{x \in D} \ell_{pred}(x, g_\theta(x); h_\omega) + \lambda \ell_{dist}(x, g_\theta(x)).
\]

(1)

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

\[
\ell_{pred}(x, g_\theta(x); h_\omega) = \mathbb{I}_C(x),
\]

(2)

where \( \mathbb{I}_C(x) \) is the well-known 0-1 indicator function, which evaluates to 1 if \( x \in C \), 0 otherwise.

The second component \( \ell_{dist} : X \times X \mapsto \mathbb{R}_{\geq 0} \) is any arbitrary distance function that discourages \( \tilde{x} \) to be too far away from \( x \). For example, \( \ell_{dist}(x, g_\theta(x)) = ||x - g_\theta(x)||_p \), where \( || \cdot ||_p \) is the \( L^p \)-norm. In this work, inspired by previous approaches, we set \( \ell_{dist} \) equal to \( L^1 \)-norm [13].
In addition, $\lambda$ serves as a scaling factor to trade off between $\ell_{dist}$ and $\ell_{pred}$. 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. Plus, the total number of perturbed features must also be limited, i.e., $|\mathcal{F}| \leq m$ for some value $1 \leq m \leq n$.

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

$$
\theta^* = \arg \min_{\theta} \mathcal{L}_{CF}(g_{\theta}; D, h_{\omega})
$$

subject to: $|\mathcal{F}| \leq m$.

This in turn allows us to generate the optimal CF $\bar{x}^*$ for any $x$, as $\bar{x}^* = g^*(x)$. Finally, the resulting optimal counterfactual explanation can be simply computed as $e_o = \bar{x}^* - x$ [39]. The overview of a generic counterfactual explanation framework is shown in Figure 1.

![Figure 1: Overview of a generic counterfactual explainer.](image)

4 PROPOSED FRAMEWORK: ReLAX

In this work, we propose to find the optimal CF generator for any arbitrary model – i.e., to solve the constrained optimization problem defined in Equation 3 – via deep reinforcement learning (DRL). We call our method Reinforcement Learning Agent xplainer (ReLAX).

4.1 Markov Decision Process Formulation

We consider the problem of computing the optimal counterfactual example $\bar{x}^*$ from $x \in \mathcal{D}$ – i.e., the optimal CF generator $g^*$ defined in Equation 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 $x$ to tweak and the magnitude of such change) and (ii) receives an observation (i.e., the prediction output by $h_{\omega}$ 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 $\bar{x}^*$ is found.

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

4.1.1 States ($S$). At each time step $t$, the agent’s state is $S_t = s_t$, where $s_t = (x_t, f_t) \in S$ represents the current modified sample ($x_t$) along with the set of features changed so far ($f_t$). More specifically, $f_t \in \{0, 1\}^{|\mathcal{F}|}$ is an $|\mathcal{F}|$-dimensional binary indicator vector, where $f_t[k] = 1$ iff the actionable feature $k$ has been modified in one of the actions taken by the agent before time $t$. Initially, when $t = 0$, $x_0 = x$ and $f_0 = 0^{|\mathcal{F}|}$.

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

4.1.2 Discrete-Continuous Hybrid Actions ($A$). To mimic the two-step behavior discussed above, we consider a discrete-continuous hybrid action space with a two-tier hierarchical structure.

For an arbitrary step $t$, we maintain the set of feature identifiers that the agent is allowed to modify $F_t$, i.e., $F_t = F$. 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 $F_t \subset F = \mathcal{F} \setminus \bigcup_{j < t} k_j$. This is to allow each feature to be selected at most in one action. Upon choosing $k_t \in F_t$, 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 we define our discrete-continuous hybrid action space as follows:

$$
A_t = \{(k_t, v_{k_t}) | k_t \in F_t, v_{k_t} \in \mathbb{R}\}.
$$

4.1.3 Transition Function ($T$). Let $a_t = (k_t, v_{k_t}) \in A_t$ 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 $x_t$ and $f_t$ as follows:

$$
T((x_t, f_t), a_t, (x_{t+1}, f_{t+1})) = \begin{cases} 1, & \text{if } x_t \xrightarrow{a_t} x_{t+1} \land f_t \xrightarrow{a_t} f_{t+1} \\ 0, & \text{otherwise}. \end{cases}
$$

The statements $x_t \xrightarrow{a_t} x_{t+1}$ and $f_t \xrightarrow{a_t} f_{t+1}$ are shorthand for $x_{t+1}[k_t] = x_t[k_t] + v_{k_t}$ and $f_{t+1}[k_t] = 1$, respectively. This corresponds to increasing the value of feature $k_t$ by the magnitude $v_{k_t}$, and updating the binary indicator vector $f_t$ accordingly.

4.1.4 Reward ($r$). The reward is expressed in terms of the objective function defined in Equation 3 and has the following form:

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

where $\ell_{\text{dist}}^t = \ell_{\text{dist}}(x_t, x)$ and $\lambda \in \mathbb{R}_{>0}$ is a parameter that controls how much weight to put over the distance between the current modified instance ($x_t$) and the original sample ($x$). 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 $x$ as lower as possible.

4.1.5 Policy ($\pi_{\theta}$). We define a parameterized policy $\pi_{\theta}$ to maximize the expected reward in the MDP problem. Our ultimate goal, though, is to find an optimal policy $\pi^* = \pi_{\theta^*}$ that solves Equation 3. It is worth noticing that finding the optimal policy $\pi^*$ that maximizes the expected return in this environment is equivalent to minimizing Equation 1, and thereby finding the optimal CF generator $g^*$. The
We use the P-DQN framework [49] to find the optimal policy \( Q \).

\[
\theta^* = \arg \min_{\theta} \frac{1}{|D|} \sum_{x \in D} \ell_{\text{pred}}(x, g(x); h_\omega) + \lambda \ell_{\text{dist}}(x, g(x))
\]

\[
= \arg \max_{\theta} \frac{1}{|D|} \sum_{x \in D} (1 - \lambda \ell_{\text{dist}}(x, g(x)) - \lambda \ell_{\text{dist}}(x, g(x)), \text{if } h_\omega(x) \neq h_\omega(x))
\]

\[
= \arg \max_{\theta} \frac{1}{|D|} \sum_{x \in D} \sum_{t=1}^T r(s_t, a_t, s_{t+1})
\]

Here, \( T \) defines the maximum steps taken by the agents for each sample \( x \) and is set to 50,000.

### 4.2 Policy Optimization

We use the P-DQN framework [49] to find the optimal policy \( \pi^* \). At each time step, the agent uses a hybrid action \( a_t \in \mathcal{A}_t \) to perturb the currently modified \( x_t \), obtained from the original input \( x \). 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 \( v_{\theta_k} \) is associated with the discrete action \( k_t \), which means \( v_{\theta_k} \) is the optimal action given state \( s_t \) and \( k_t \). We therefore cast this as a function \( v_{\theta_k}^O(s_t) \). Thus, the Bellman equation can be written as:

\[
Q(s_t, k_t, v_{\theta_k}) = \mathbb{E}_{r_t, s_{t+1}}[r_t + y \max_{k_t} Q(s_{t+1}, k_t, v_{\theta_k}^O(s_{t+1}))(s_t, a_t = (k_t, v_{\theta_k}))].
\]

As with DQN, a deep neural network (DNN) \( Q^\theta(s_t, k_t, v_{\theta_k}) \) is used to approximate the Q-value function, and we fit \( v_{\theta_k}^O \) with another deterministic policy network \( \pi^O_{\theta_k}(s_t) \), where \( \theta_1 \) and \( \theta_2 \) are the parameters of the two DNNs. To find the optimal \( \theta_1^*, \theta_2^* \), we minimize the loss functions below via stochastic gradient descent:

\[
L_Q(\theta_1) = \|Q^\theta(s_t, k_t, v_{\theta_k}) - y_t\|^2, \quad \mathcal{L}_o(\theta_2) = \sum_{k_t \in \mathcal{K}} \sum_{v_{\theta_k}} Q(s_t, k_t, v_{\theta_k})^O(\theta_2) - y_t^O(\theta_2)^2,
\]

where the \( y_t \) is the n-step target [38]. 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 samples transition pairs uniformly from the replay buffer. Therefore, we adopt prioritized experience replay [34] 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:

\[
p_t \propto |R_j + yQ_{\text{target}}(s_j, \pi_j) - Q(s_{j-1}, a_{j-1})|^{\beta},
\]

where \( \beta \) is a parameter to determine how much prioritization is used. During training, the transition pairs are stored in the replay buffer with their priority; a sum tree data structure is then used to efficiently update the priority and sample pairs from the replay buffer.

### 4.3 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 [15][11]; 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 adopt a Random Network Distillation (RND) module [7], i.e., a curiosity-driven approach to generate state curiosity \( r_t^s \) by quantifying the next-state novelty, which will be further combined with the external rewards \( r_t \) to generate the modified reward \( r_t^s = r_t^s + r_t \). RND transforms the exploring procedure into a supervised learning task by minimizing the mean squared error:

\[
r_t^s = \|\hat{f}(s_t, \eta_t) - f(s_t)\|^2,
\]

where \( f(s_t) \) is the fixed target network and \( \hat{f}(s_t, \eta_t) \) 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, \( r_t^s \) 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 \( a_t \) = \( (k_t, v_{\theta_k}^O(s_t)) \) is given by:

\[
r_t^s = \|\hat{g}(s_t, a_t, \eta_2) - g(s_t, a_t)\|^2,
\]

where \( g \) and \( \hat{g} \) have the same role as \( f \) and \( \hat{f} \), respectively. Here, we leverage state curiosity \( r_t^s \) to enhance high-level index planning in discrete spaces, and \( r_t^s \) 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 4.2:

\[
L'_Q(\theta_1, \eta_1) = \|Q^\theta(s_t, k_t, v_{\theta_k}) - y_t\|^2 + \|\hat{f}(s_t, \eta_t) - f(s_t)\|^2, \quad L'_o(\theta_2, \eta_2) = \sum_{k_t \in \mathcal{K}} \sum_{v_{\theta_k}} Q(s_t, k_t, v_{\theta_k})^O(\theta_2) + (\hat{g}(s_t, a_t, \eta_2) - g(s_t, a_t))^2
\]

It is worth remarking that \( a_t \) is a function of \( \theta_2 \). By alternatively updating \( \theta_2 \) and \( \eta_2 \), the policy network \( \pi^O_{\theta_k} \) will learn to balance the action’s potential value and novelty.

The overview of our ReLAX framework is depicted in Figure 2.

### 4.4 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 4.3 above. However, in some cases, the CF generation process should capture the peculiarities of each individual original instance. To accommodate such a need, we introduce a variant 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 a pretrained ReLAX-Global’s policy. Then, using a standard transfer learning approach [50], 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 [13]. Specifically, we uniformly sample data points whose $L^2$-norm from the target example is at most $1.3$.

Notice that all our input samples are normalized unit vectors.

5 EXPERIMENTS

5.1 Setup

Datasets and Tasks. We test with five public tabular datasets described in Table 1, used for classification and regression tasks.

| Dataset             | N. of Instances | N. of Features | Task          |
|---------------------|-----------------|----------------|---------------|
| Breast Cancer [5]   | 699             | 10 (numerical) | classification|
| Diabetes [2]        | 768             | 8 (numerical)  | classification|
| Sonar [3]           | 208             | 60 (numerical) | classification|
| Wave [4]            | 5,000           | 21 (numerical) | classification|
| Boston Housing [1]  | 506             | 14 (mixed)    | regression    |

Table 1: Main characteristics of the five 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 held 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. In Table 2, we summarize the main 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 Section 2. Nearest-CT is considered the simplest approach. Furthermore, we distinguish between model-specific and model-agnostic methods. The former include: FeatTweak and FOCUS (tree-specific); DeepFool and GRACE (NN-specific). The latter are: LORE, MACE, and DiCE.
Table 2: Model structure and performance for each dataset/task pair.

| Dataset [Best Model] | Structure | Acc. (±) | RMSE (±) |
|----------------------|-----------|----------|----------|
| Breast Cancer [RF]   | [trees=100] | 0.99 (±) |          |
| Diabetes [AdaBoost]  |           |          |          |
| Wave [XGBoost]       |           |          |          |
| Boston Housing [MLP] | [MLP-8, MLP-128] | 1.00 (±) |          |
| Sonar [MLP]          | [MLP-256, MLP-256] | 0.90 (±) |          |
| Wave [MLP]           | [MLP-100, MLP-250] | 0.97 (±) |          |
| Boston Housing [MLP] | [MLP-60, MLP-128] | 1.36 (±) |          |

Table 3: Sparsity vs. Validity of counterfactuals generated by RelAX for the Boston Housing regression task.

| Threshold (δ) | RelAX-Global | RelAX-Local |
|--------------|--------------|-------------|
| 0.20         | 0.81 ± 0.09 (0.92 ± 0.17) | 0.87 ± 0.05 (0.90 ± 0.18) |
| 0.40         | 0.74 ± 0.06 (0.99 ± 0.10) | 0.81 ± 0.05 (0.98 ± 0.16) |
| 0.60         | 0.70 ± 0.06 (0.91 ± 0.12) | 0.77 ± 0.03 (0.92 ± 0.09) |

5.2 Results

Sparsity-Validity Trade-off: In Figure 3, 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 are 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 [17]. 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 3. 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 tolerance δ used to determine if the prediction goal of the counterfactual example is met the harder is for RelAX to find a valid counterfactual.

Finally, analogous conclusions can be drawn if we compare proximity vs. validity (see Table 4): 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 4 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 is even more remarkable if we consider that the counterfactual generation time of RelAX includes the training of the DRL agent.

5.3 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 Figure 4, 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 Figure 4 – left), whereas smaller values of λ result in the opposite (see Figure 4 – right).
Figure 3: Sparsity vs. Validity of counterfactuals generated by ReLAX and other baselines for classification tasks.

Table 4: Comparison of Proximity and Generation Time for model-agnostic CF generation methods.

| Metric              | Dataset [Models] | CF Generation Methods | RELAX-Global | RELAX-Local | LORE | MACE |
|---------------------|------------------|-----------------------|--------------|-------------|------|------|
| Proximity           | Breast Cancer [RF, MLP] | RELAX-Global | [4.46, 5.92] | [4.49, 5.87] | [4.63, 5.63] | [4.47, N/A] |
|                     | Diabetes [AdaBoost]     | RELAX-Global | [4.31] | [4.30] | [4.78] | [N/A] |
|                     | Sonar [MLP]             | RELAX-Local | [7.32] | [7.66] | [7.36] | [N/A] |
|                     | Wave [XGBoost, MLP]     | RELAX-Local | [5.93, 6.38] | [6.02, 6.50] | [6.68, 6.41] | [N/A] |
|                     | Boston Housing [MLP-Reg] | RELAX-Local | [5.10] | [5.36] | [N/A] | [N/A] |
|                     | Generation Time (secs.) |             | 1500 | 1320 | 2100 | 2280 |

Table 5: The impact of different MLP architectures on the Validity (Sparsity) of counterfactuals.

| MLP size           | Acc. | RELAX-Global | RELAX-Local | GRACE | LORE |
|--------------------|------|--------------|-------------|-------|------|
| [256, 256]         | 0.88 | 0.76 (2.95) | 0.90 (3.19) | 0.62 (3.32) | 0.60 (3.65) |
| [128, 128]         | 0.85 | 0.80 (2.69) | 0.90 (2.88) | 0.73 (2.85) | 0.76 (3.21) |
| [64, 64]           | 0.79 | 0.90 (1.88) | 0.95 (1.93) | 0.86 (1.90) | 0.90 (2.52) |

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 5 shows the impact of different MLP architectures on the validity and sparsity of counterfactuals generated by ReLAX in comparison with two competitors: GRACE (NN-specific) and LORE (model-agnostic). We may observe that both GRACE and LORE are quite 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, the reason for that detrimental effect is due to the fact that it leverages 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. Instead, ReLAX is more robust toward different MLP structures, and its performance is 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 4.4, 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 6.

| Setting          | Validity (Sparsity) | Gen. Time (secs.) |
|------------------|---------------------|-------------------|
| without pretraining | 0.80 (3.29)         | 1132              |
| with pretraining  | 0.90 (3.19)         | 500               |

Table 6: Performance of ReLAX-LOCAL with/without pretrained ReLAX-Global.

6 CASE STUDY: COVID-19 MORTALITY

The goal of this case study 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.

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. Inspired by previous work addressing a similar task [6], we gather 17 country-level demographic features from 156 countries across 7 continents [8, 46–48]. Furthermore, we label each instance with a binary class denoting the COVID-19 risk of mortality (“high” vs. “normal”), as in [6]. This dataset is made publicly available.

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 Figure 5 (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).

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. Figure 5 (right) shows the direction of feature changes as suggested by the CFs.

We observe that the CFs suggest five main changes: (i) Decreasing the death rate; (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 healthcare workforce. Moreover, according to the investigation made by the US Center for Disease Control and Prevention, 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 and the urban population rate may allow a broader range of people to enhance their social community awareness, take consciousness of the risks of the pandemic, and thus adopt a safe lifestyle.

7 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 five public tabular datasets demonstrated that ReLAX significantly outperforms all the considered CF 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). Finally, we show that CFs generated by ReLAX are useful in practice, as they can be used to suggest actions that a country should take to reduce the risk of COVID-19 mortality.

Several research directions are worth exploring in future work. For example: extend ReLAX to generate explanations for non-tabular input data like images, and compare our method against non-counterfactual explanation approaches like SHAP [24] or LIME [31].

ACKNOWLEDGMENTS

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 RDS10122020021. We thank the support from Casual AI Reading Group sponsored by Swarma Club and XOrder. We also thank Prof. Jianye Hao and Dr. Tianpei Yang.

1https://github.com/Mewtwo1996/ReLAX.git
2https://www.npr.org/sections/health-shots/2020/01/06/1069369625/short-staffed-and-covid-battered-u-s-hospitals-are-hiring-more-foreign-nurses
3https://www.cdc.gov/obesity/data/obesity-and-covid-19.html
