Fairness Testing of Machine Learning Models Using Deep Reinforcement Learning

Wentao Xie and Peng Wu ✉
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
Email: {xiewt,wp}@ios.ac.cn

***Abstract*—Machine learning models play an important role for decision-making systems in areas such as hiring, insurance, and predictive policing. However, it still remains a challenge to guarantee their trustworthiness. Fairness is one of the most critical properties of these machine learning models, while individual discriminatory cases may break the trustworthiness of these systems severely. In this paper, we present a systematic approach of testing the fairness of a machine learning model, with individual discriminatory inputs generated automatically in an adaptive manner based on the state-of-the-art deep reinforcement learning techniques. Our approach can explore and exploit the input space efficiently, and find more individual discriminatory inputs within less time consumption. Case studies with typical benchmark models demonstrate the effectiveness and efficiency of our approach, compared to the state-of-the-art black-box fairness testing approaches.**

***Index Terms*—fairness testing; black-box testing; deep reinforcement learning; test data generation; machine learning models**

I. Introduction

In recent years, machine learning has been widely applied in various safety-critical areas, such as autopilot [1], medical diagnosis [2], face recognition [3], and financial decision-making [4]. Although machine learning has achieved great success and even surpassed the performance of humans beings in these areas, it is still challenging to ensure the trustworthiness of machine learning models. Fairness is one of the most important aspects of a dependable system. However, unfortunately, machine learning models may make unfair decisions due to the possibly biased training data, and hence result in unintended social injustices. Therefore, it is a highly-demanding task to study an efficient approach for systematically exposing the potential fairness violations in machine learning models.

Two general categories of criteria have been presented to evaluate the fairness of a machine learning model: *group fairness* and *individual fairness*. Group fairness means that groups of inputs with different values over the protected attributes (such as race, gender, or age) are treated equally fair by the machine learning model. Typical group fairness criteria include equal opportunity [5] and disparate treatment [6]. Individual fairness [7] is defined on individual inputs, and asserts that similar outcomes shall be decided for similar inputs (i.e, the inputs that differ only on the protected attributes). From the perspective of software testing, the discriminately treated individual inputs witness exactly the individual fairness defects in the machine learning model. The motivation of this paper is to investigate a cost-effective way to generate test inputs automatically to detect such individual fairness violations. As indicated in [8], the fairness of a machine learning model can be significantly improved through retraining with the detected individual discriminatory inputs.

A few efforts have been devoted on fairness testing of machine learning models using black-box testing approaches [9]–[11], or a white-box testing approach [8]. Galhotra *et al*. proposed THEMIS [9] to evaluate the fairness score of a machine learning model with random test inputs. Udeshi *et al*. proposed Aequitas [10] to improve the efficiency of fairness testing by searching the neighborhoods of the randomly found individual discriminatory inputs. A *neighbour* of a test input is constructed randomly by perturbing the test input itself. Aggarwal *et al*. proposed SG to learn decision trees as local interpretable model-agnostic explanations (LIME [12]) of a machine learning model through random sampling, and then generate test inputs through symbolic execution along the paths of the local decision trees [11]. In contrast, Zhang *et al*. proposed a white-box fairness testing approach ADF for deep neural networks [8], which combines global and local search strategies to generate individual discriminatory inputs through gradient-guided adversarial sampling. The experiment results show that ADF can explore the search space more effectively with more individual discriminatory inputs detected than the previous black-box fairness testing approaches. But the gradient information is not available in the black-box setting, hence limits the applicability of this white-box fairness testing approach.

In this paper we intend to further improve the effectiveness and efficiency of black-box fairness testing for general machine learning models, within the limited test resources (e.g., time consumption). As shown in the above related work, random sampling may not lead a way close to individual discriminatory inputs. An instructive guidance is necessary to help systematically search for individual discriminatory inputs in the black-box setting.

Inspired by the success of reinforcement learning [13] in real-word applications (e.g., game play [14], [15], recommendation systems [16], robotics [17], [18]) and in software testing [19], we propose to rely on reinforcement learning to achieve an optimal black-box search strategy for individual fairness testing. Indeed, the individual fairness test generation problem can be formulated as a reinforcement learning

978-1-6654-0392-4/20/$31.00 ©2020 IEEE
DOI 10.1109/TrustCom50675.2020.00029

problem, where the machine learning model under test (MUT) is regarded as a part of the environment. The reinforcement learning agent takes an action to the environment, which results in a test input for the MUT, and then observe the state and the feedback from the environment in terms of a reward. Through iterations of such interactions, the agent learns an optimal policy to generate individual discriminatory inputs with high efficiency, and without accessing the internal dynamics of the MUT.

The contribution of this paper is three-fold as follows.

1) We present a reinforcement learning framework for black-box fairness testing of machine learning models. To the best of our knowledge, this is the first time that reinforcement learning is adopted for black-box testing of general machine learning models.
2) We implement our reinforcement-learning based black-box fairness testing approach using the state-of-the-art Deep Q-Network [14] as the underlying learning engine.
3) We evaluate our implementation through detailed case studies with typical fairness benchmarks from the machine learning literature. The experimental results show that our approach can detect more individual discriminatory inputs within less time consumption. Indeed, our approach can generate 4 to 10 times more individual discriminatory inputs than the state-of-the-art black-box fairness testing approaches within the same amount of time. Our approach can also detect on average 3 times more individual discriminatory inputs than the state-of-the-art black-box fairness testing approaches with the same number of tests.

We conjecture that our reinforcement-learning based framework can be flexibly extended for testing other aspects of machine learning models, and thus open a way of research and practices in the field of machine learning testing.

The rest of the paper is organized as follows. Section 2 introduces the definition of individual discrimination and the basic concepts of reinforcement learning. Section 3 presents our reinforcement-learning based framework for black-box fairness testing of machine learning models. Then, our implementation and experiments are reported in Section 4 with detailed analysis. We discuss the related work in Section 5. The paper is concluded in Section 6 with future work.

II. Preliminaries

In this section, we briefly introduce the background and basic concepts used throughout the paper, including individual discrimination and reinforcement learning.

A. Individual Discrimination

Let X denote an n-dimensional dataset with $n > 1$, and $A = \{a_1, a_2, \ldots, a_n\}$ be its set of attributes. Let $\mathcal{I}_k$ denote the set of values of each attribute a_k for $1 \leq k \leq n$. Then, the input domain is $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_n$, and $X \subseteq \mathcal{I}$. Assume $P \subset A$ is the set of protected attributes, such as gender, race, and age; while the attributes in $A \setminus P$ are non-protected.

Considering a machine learning model $M : \mathcal{I} \to \mathcal{O}$ that accepts an input $x \in \mathcal{I}$ and decides an output $M(x) \in \mathcal{O}$. Herein, any decision made by model M shall not depend on any protected attribute of the input. This is characterized by *individual discrimination*, which is formally defined as follows.

Definition 1 (Individual Discrimination): For any $x = (x_1, x_2, \ldots, x_n) \in \mathcal{I}$ and $x' = (x'_1, x'_2, \ldots, x'_n) \in \mathcal{I}$, (x, x') is an *individual discriminatory pair* of model M with respect to the set P of protected attributes, if

- there exists an attribute $a_p \in P$ such that $x_p \neq x'_p$;
- for any attribute $a_q \in A \setminus P$, $x_q = x'_q$;
- $M(x) \neq M(x')$.

M is *individually fair* with respect to the set P of protected attributes if there does not exist an individual discriminatory pair of M with respect to P.

When the context is clear, (x, x') is simply referred to as an individual discriminatory pair, and x and x' are also known as individual discriminatory inputs (or instances).

Note that by making no explicit use of the protected attributes in training M does not necessarily prevent the model from making unfair decisions. This is also known as fairness through unawareness (FTU) [20]. FTU does not take into account the potential of certain non-protected attributes containing discriminatory information analogous to the protected ones, and hence highly predictive of the true outcome of the model. Such non-protected attributes and their relationships with the protected ones may not be obvious in the first place [7], [21].

B. Reinforcement Learning

Reinforcement learning [13] is a classical machine learning paradigm for an agent (or learner) to make a sequence of decisions in an uncertain and possibly complex environment. At each step $t \geq 0$, the agent observes the state s_t of the environment, and takes an appropriate action α_t to the environment in response. Then, the agent receives a reward r_t from the environment and observes its follow-up state s_{t+1}. Along these interactions with the environment, the goal of the agent is to maximize the expected cumulative reward $\mathbf{E}(\sum_{t=0}^{\infty} \gamma^t r_t)$, where $\gamma \in [0, 1]$ is a discount factor. A policy π of the agent maps the states to the actions.

The agent uses the action-value function $Q^{\pi}(s_t, \alpha_t) = \mathbf{E}[\sum_{t'=t}^{\infty} \gamma^{t'} r_{t'} | s_t, \alpha_t]$ to estimate how appropriate it is for the agent to take action α_t at state s_t under policy π. The optimal action-value function is defined as the maximum expected cumulative reward achievable under any policy. Usually, the agent initially sets a random action-value function, and then update it iteratively based on the *Bellman equation* [13] until the action-value function converges to the optimal one. As a result, the agent follows a greedy strategy to achieve the optimal policy. For finite sets of states and actions, tables are often used to store the action-value functions. Due to the curse of dimensionality, neural networks are used to fit the action-value functions.

By introducing deep neural networks into the reinforcement learning paradigm, deep reinforcement learning [22] well integrates function approximation and target optimization, and can scale to decision-making problems that were previously intractable [15], [16], [18]. In this work, we use Deep Q-Network [14] as the underlying learning engine to help generate more individual discriminatory inputs within less test consumption.

III. Methodology

We present in this section our reinforcement-learning based black-box fairness testing approach for general machine learning models.

A reinforcement learning framework can be formally represented as 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ is the non-empty set of states of the environment; $\mathcal{A}$ is the non-empty set of actions; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function; and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The behaviour of the agent interacting with the environment can be modelled as a Markov decision process. A policy $\pi : \mathcal{S} \to \mathcal{A}$ of the agent decides the next action to be taken based on the observation of the state of the environment. Then, the goal of the agent is to learn an optimal policy π to maximize the expected cumulative reward $\mathbf{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(x_t, \pi(x_t)]$.

A. Reinforcement-Learning Framework for Black-Box Fairness Testing

The black-box fairness test generation problem of a machine learning model can be characterized within a reinforcement learning framework, where the machine learning model under test (MUT) is the core part of the environment, and the agent plays the role of a tester with the intention to learn an optimal online test generation strategy for the MUT. Fig. 1 shows the overall architecture of the reinforcement learning framework for black-box fairness testing.

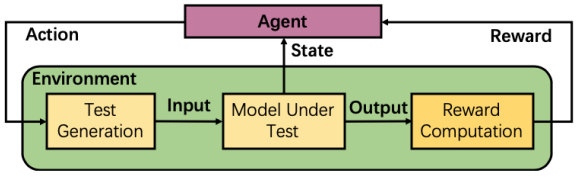

Fig. 1. Reinforcement Learning for Black-Box Fairness Testing

Informally, at each step $t \geq 0$, the agent observes the state s_t of the environment, i.e., the test input executed on the MUT, and takes to the environment the action α_t with the maximal action value so far. The action is then interpreted in the environment to generate a follow-up test input s_{t+1} and its counterpart inputs, which are the same as the follow-up test input except in their protected attributes. These inputs are then executed on the MUT one-by-one, and the outputs of the MUT are evaluated to determine whether an individual discriminatory pair is detected. Based on the evaluation result, the environment responds to the agent with a corresponding reward r_{t+1}. The reinforcement learning framework for black-box fairness testing is depicted below in details.

States A test input that is currently executed on the MUT is regarded as the current state of the environment (or the black-box testing process). Thus, the state space $\mathcal{S}$ is exactly the input domain $\mathcal{I}$ of the MUT, i.e., $\mathcal{S} = \mathcal{I}$ in the context of black-box testing. Especially, an initial test input that the testing process starts with constitutes the initial state x_0 of the environment, and is often pre-determined randomly.

Actions One of the key issues of machine learning testing is the test generation problem, that is, how to generate effective test inputs that can expose the potential defects (e.g., the individual discriminatory pairs) of the MUT. Previous work [8]–[11] indicate that random sampling is very inefficient for this purpose. Moreover, the individual discriminatory inputs are often located in contiguous regions in the input domain, which explains the performance improvement of a local search method in detecting the individual fairness defects. This suggests to exploit the neighbourhoods of the already executed test inputs for follow-up search.

Let $\|x' - x\|_0$ denote the L_0 distance between x and x' in $\mathcal{I}$, i.e., the number of their non-protected attributes whose values are different. Then, a neighbourhood of an input x can be directly defined as $\{x' \mid \|x' - x\|_0 = 1\}$, i.e., the inputs that differ from x only in one non-protected attribute. Hence, we define a systematic action space $\mathcal{A} = \{+\delta_k, -\delta_k \mid a_k \in A \setminus P, \delta > 0\}$, where each action $+\delta_k$ (respectively, $-\delta_k$) mutates only one non-protected attribute a_k of a (source) test input with offset $+\delta$ (respectively $-\delta$). In this way, each action can lead to a follow-up test input that is marginally close to the source one. Note that for each test input x, we need to run the MUT with x and all its counterparts x' that differ from x only on one or more protected attributes, in order to check if (x, x') constitutes an individual discriminatory pair. Thus, an action $+\delta_k$ or $-\delta_k$ on a protected attribute $a_k \in P$ can only result in redundant test inputs that have already been executed in previous steps.

Transition Function The transition function characterizes the state change effect of the actions on the environment. With the above settings of states and actions, the transition function $\mathcal{P}$ can be defined as follows: for any $x = (x_1, \ldots, x_k, \ldots, x_n) \in S$, $a_k \in A \setminus P$, $\pm \in \{+, -\}$,

$$\mathcal{P}(x, \pm\delta_k) = \begin{cases} (x_1, \ldots, x_k \pm \delta, \ldots, x_n) & \text{if } x_k \pm \delta \in \mathcal{I}_k \\ x & \text{otherwise} \end{cases}$$

A sequence $x_0, x_1, \cdots$ of states is an execution of the environment observed by the agent, if for any $t \geq 0$, there exists $\alpha_t \in \mathcal{A}$ such that $x_{t+1} = \mathcal{P}(x_t, \alpha_t)$, denoted $x_t \xrightarrow{\alpha_t} x_{t+1}$. With the limited test resource, we consider only finite executions $x_0, x_1, \cdots, x_T$, where $T > 0$ is a constant and x_T is the termination state. Then, the sequence of state-action pairs $e = (x_0, \alpha_0), (x_1, \alpha_1), \cdots, (x_{T-1}, \alpha_{T-1})$ constitutes an episode e, representing one round of reinforcement learning. The length of each episode is fixed, i.e., the number T of steps in the episode. Thus, the reinforcement-learning based black-box fairness testing can be performed round-by-round to gradually reach an optimal search strategy for individual

discriminatory inputs.

Reward Function The reward function is designed to guide the agent to make appropriate actions towards accomplishing the goal of reinforcement learning. In the context of fairness testing, we prefer the actions that can deliver as many fresh individual discriminatory inputs as possible. Let $d(x)$ denote that x is an individual discriminatory input, and d_set be the set of the individual discriminatory inputs generated and executed previously. The reward function can then be defined as follows.

$$\mathcal{R}(x,\alpha) = \begin{cases} r_1 & \text{if } d(\mathcal{P}(x,\alpha)) \text{ and } \mathcal{P}(x,\alpha) \notin d\_set \\ r_2 & \text{if } d(\mathcal{P}(x,\alpha)) \text{ and } \mathcal{P}(x,\alpha) \in d\_set \\ r_3 & \text{if not } d(\mathcal{P}(x,\alpha)) \end{cases}$$

where r_1, r_2, r_3 are reward constants.

In order to maximize the expected cumulative reward, we set $r_1 \geq r_2 \geq r_3$, $r_1 \geq 0$ and $r_3 \leq 0$ particularly. Thus, the actions that can lead to unknown individual discriminatory inputs are rewarded positively, while those lead to non-discriminatory inputs are punished with negative rewards.

Let $e(\pi)$ be the episode derived by policy π, i.e., $e = (x_0, \alpha_0), (x_1, \alpha_1), \cdots, (x_{T-1}, \alpha_{T-1})$ with $\alpha_t = \pi(x_t)$ and $x_{t+1} = \mathcal{P}(x_t, \alpha_t)$ for each $0 \leq t < T$. Then, the expected cumulative reward $\mathcal{E}_{e(\pi)}$ after the agent finishes the episode $e(\pi)$ is $\mathcal{E}_{e(\pi)} = \mathbf{E}[\sum_{t=0}^{T-1} \gamma^t \mathcal{R}(x_t, \alpha_t)]$. Thus, the black-box fairness test generation problem of a machine learning model with the input domain $\mathcal{I}$ can be transformed into a reinforcement learning framework $\langle \mathcal{I}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ that aims to find an optimal policy π to maximize the expected cumulative reward $\mathcal{E}_{e(\pi)}$.

B. Black-Box Fairness Testing Through Deep Q-Network

We implement the above black-box fairness testing framework with the state-of-the-art Deep Q-Network (DQN) as the underlying learning engine. Algorithm 1 depicts the main algorithm of our DQN-based black-box fairness testing approach for a given model learning model (encapsulated within the environment M).

The inputs of the main algorithm include the environment M, the DQN agent Q initialized with a random policy π, and an initial test input x_0 that can be random or come from the training data. Through reinforcement learning, the agent updates its policy iteratively to guide the fairness testing process towards individual discriminatory inputs. Considering the limited test resources, the agent repeats only N finite episodes, and each episode runs for T steps.

At each step $step$ with $0 \leq step < T$, the agent observes the state $state$ of the environment. At Line 7, the agent follows the typical $\epsilon-$greedy exploration strategy to select the action $action$, i.e., the agent may select a random action with probability $\epsilon \in (0, 1]$, or otherwise select the action $action \in \mathcal{A}$ with the maximal approximate action value $Q^\pi(state, action)$.

Then, at Line 8, the environment responds the agent with the next state $next\_state$ and the reward r, as defined in the previous section. If the next state $next\_state$ constitutes an individual discriminatory input not yet met before (Line 9), it is then included into the set d_set of individual discriminatory inputs (Line 10).

Algorithm 1: Black-box Fairness Testing Through Deep Q-Network

Input: Environment M, agent Q, initial test input x_0
Output: Set d_set of individual discriminatory inputs

```
1  Initialize a replay memory D of capacity C;
2  d_set ← ∅, num_episode ← 0;
3  while num_episode < N do
4      step ← 0, r ← 0, state ← x_0;
5      num_episode ← num_episode + 1;
6      while step < T do
7          action ← Q.act(state);
8          next_state, r ← M.step(action, state);
9          if d(next_state) and next_state ∉ d_set then
10             d_set ← d_set ∪ {next_state};
11         D.append(state, action, r, next_state);
12         state ← next_state;
13         step ← step + 1;
14         batch ← D.sample(BATCHSIZE);
15         Q.replay(batch);
16 return d_set
```

At Line 11, the execution step $(state, action, r, next\_state)$ is pooled into the replay memory D to refresh the agent's experiences. At Line 14, a minibatch $batch$ of size $BATCHSIZE$ is drawn randomly from the replay memory. Then, at Line 15, the agent performs a prioritized experience replay [23] with the minibatch to revise the policy π, by updating the deep Q-network through the stochastic gradient decent to minimize the square error between the approximated action-values and the target action-values from the Bellman function, i.e., for each execution step $(s_j, \alpha_j, r_j, s'_j)$ in the minibatch, performing a gradient decent step on

$$(y_j - Q^\pi(s_j, \alpha_j))^2$$

where $y_j = r_j + \gamma \max_{\alpha \in \mathcal{A}} Q^\pi(s_{j+1}, \alpha)$ is the target action-value according to the Bellman equation.

IV. Implementation and Experiments

We implement the above black-box fairness testing algorithm in Python 3.7.2, based on an open-source implementation of Deep Q-Network [24]. We evaluate and compare our approach with the state-of-the-art black-box fairness testing tools Aequitas [10] and SG [11], of which the implementations are from [8] and configured for its best performance according to the literature. Recall that Aequitas and SG have been briefly introduced in the first section.

A. Research Questions

The experiments are designed to evaluate the effectiveness and efficiency of our black-box fairness testing approach in

generating individual discriminatory inputs. Specially, through the experiments, we intend to investigate the following research questions.

RQ1: *How efficiently can our approach generate individual discriminatory inputs in an effective and adaptive way?*

RQ2: *How do the parameters (e.g., the reward constants, the initial test input) in the main algorithm influence the effectiveness and efficiency of our approach?*

B. Effectiveness and Efficiency

Table I shows the datasets used in our experiments. These datasets are commonly used as the benchmarks in the fairness testing literature [8], [10], [11]. For each dataset, we train a six-layer fully-connected neural network (Multilayer Perceptron, MLP) model using tensorflow 1.14.0 [25] and a logistic regression (LR) model using sklearn 0.20.3. The training results are reported in Table II.

TABLE I
DATASETS

Benchmark	Size	Number of Attributes	Domain Size
German Credit	1000	21	6.32×10^{17}
Bank Marketing	45211	16	3.30×10^{24}
Census Income	15360	13	1.74×10^{18}

TABLE II
MACHINE LEARNING MODELS UNDER TEST

Dataset	Model	Accuracy
German Credit	Multilayer Perceptron	100%
German Credit	Logistic Regression	70%
Bank Marketing	Multilayer Perceptron	93.41%
Bank Marketing	Logistic Regression	89.87%
Census Income	Multilayer Perceptron	86.28%
Census Income	Logistic Regression	80.87%

The following experiments are run on a macOS Catalina system with a 2.7GHZ quad-core Intel Core i7 processor and 16GB memory. Each experiment is repeated three times for statistical average.

The parameters used in the main algorithm is set as follows: the capacity of the replay memory is $C = 10000$; the size of a minibatch is $BATCHSIZE = 32$; the discount factor is $\gamma = 1$ because, as in [19], we treat each reward equally important and ignore the disturbance of the time it is received. In each episode, the probability ϵ of taking a random action is decreased from 1 to 0.1 in the first 2556 steps, and fixed at 0.1 thereafter. This is to warm up the agent for further interactions.

The following metrics are used to quantitatively evaluate the effectiveness and efficiency of our approach and others,

- $\#Tests$: the number of test inputs executed;
- $\#DTests$: the number of individual discriminatory inputs detected, among all the executed test inputs;
- $Time$: time consumption in seconds for executing all the test inputs;
- $G{-}Ratio$: the generation ratio that is the ratio of the individual discriminatory inputs among all the executed test inputs, i.e., $G{-}Ratio = \#DTests/\#Tests$;
- $G{-}Rate$: the generation rate that is the number of the individual discriminatory inputs detected per second, i.e., $G{-}Rate = \#DTest/Time$.

TABLE III, TABLE IV and TABLE V report the experimental results of Aequitas, SG and our approach, respectively. The protected attribute of each dataset is shown in Column Dataset.

For SG, its implementation terminates by default when 1000 test inputs are generated. For our approach, the reward constants are set as follows: $r_1 = 100, r_2 = -3, r3 = -4$, and the DQN agent runs 200 episodes, each taking 500 steps.

In Columns $G{-}Ratio$ and $G{-}Rate$ of TABLE V, "(kX)" means that the value of the corresponding metric of our approach is k times the maximum value of the same metric of both Aequitas and SG approaches.

It can be seen that our approach outperforms greatly Aequitas and SG in the generation rates for all the six models tested. Indeed, our approach can generate 4 to 10 times more individual discriminatory inputs than Aequitas and SG within the same amount of time. Our approach also outperforms Aequitas and SG in the generation ratios for all the models, except the Bank Marketing and German Credit MLP models. For these two models, our approach can generate 7 to 10 times more individual discriminatory inputs than SG within the same amount of time, while the generation of local decision trees in SG is very time-consuming. On average, the generation ratio of our approach is about 3 times higher than that of Aequitas and of SG.

C. Reward Function

In the reward function presented in Section 3.1, there are three reward constants r_1, r_2 and r_3. We further investigate the effects of these three constants in the defect detection efficiency of our approach.

If $r_1 = r_2 = r_3 = 0$, our approach presumably downgrades close to a random test generation approach. The green curve in Fig. 2 shows the G-ratios of individual discriminatory inputs generated with $r_1 = r_2 = r_3 = 0$ within 1000 episodes for the Census Income MLP model, while the blue one shows the G-ratios of individual discriminatory inputs generated by the baseline approach, i.e., generating test inputs through only random actions. It can be seen that both curves get very close to each other after about 200 episodes. This indicates that non-zero rewards play the key role in optimizing the search strategy towards individual discriminatory inputs.

1) r_1: To evaluate the effect of r_1, we set $r_2 = r3 = 0$ to avoid the interventions from r_2 and r_3. Fig. 3 reports the results of 200 episodes with $r_1 \in [0, 100]$ for the Census Income MLP model. It can be seen that in most cases, the larger r_1 is, the higher the G-ratio of individual discriminatory inputs is. This indicates the incentive effect of positive r_1 rewards on generating fresh individual discriminatory inputs.

Due to the nonlinear nature of the underlying problem, it is expected that an unnecessarily large r_1 would not help detect more discriminatory defects, but just increase the computational burden.

TABLE III
AEQUITAS

Dataset	Model	$\#Tests$	$\#DTests$	$Time$	$G-Ratio$	$G-Rate$
Census Income (Gender)	MLP	3609	608	66.3	16.8%	9.17
Census Income (Gender)	LR	65539	3760	366	5%	10.27
Bank Marketing (Age)	MLP	4579	1893	380	41.3%	4.98
Bank Marketing (Age)	LR	36259	937	692	2.58%	1.35
German Credit (Gender)	MLP	3864	667	158	17.3%	4.22
German Credit (Gender)	LR	11738	545	117	4%	4.65

TABLE IV
SG

Dataset	Model	$\#Tests$	$\#DTests$	$Time$	$G-Ratio$	$G-Rate$
Census Income (Gender)	MLP	1000	200	613	20%	0.32
Census Income (Gender)	LR	1000	82	1070	0.82%	0.07
Bank Marketing (Age)	MLP	1000	702.33	743	70.2%	0.94
Bank Marketing (Age)	LR	1000	92.33	1114	9.23%	0.08
German Credit (Gender)	MLP	1000	703	757.6	70.3%	0.93
German Credit (Gender)	LR	1000	74	1035	0.74%	0.07

TABLE V
OUR APPROACH BASED ON DEEP Q-NETWORK

Dataset	Model	$\#Tests$	$\#DTests$	$Time$	$G-Ratio$	$G-Rate$
Census Income (Gender)	MLP	86176	39171	883	**45.4%(2.27X)**	**44.36(4.83X)**
Census Income (Gender)	LR	85029	37689	753	**44.3%(8.87X)**	**50.05(4.87X)**
Bank Marketing (Age)	MLP	84241	48946	1029	58.1%(0.82X)	**47.56(9.55X)**
Bank Marketing (Age)	LR	80205	12659	834	**15.78%(1.70X)**	**15.17(11.23X)**
German Credit (Gender)	MLP	79597	23978	871	30.12%(0.43X)	**27.52(6.52X)**
German Credit (Gender)	LR	84858	30801	761	**36.29%(9.07X)**	**40.47(8.70X)**

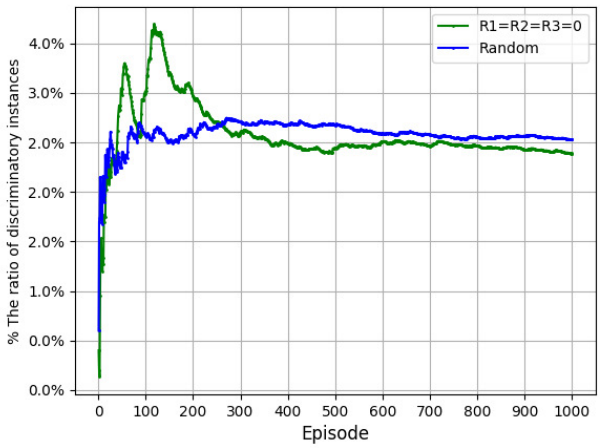

Fig. 2. Zero Rewards (Census Income MLP, $r_1 = r_2 = r_3 = 0$)

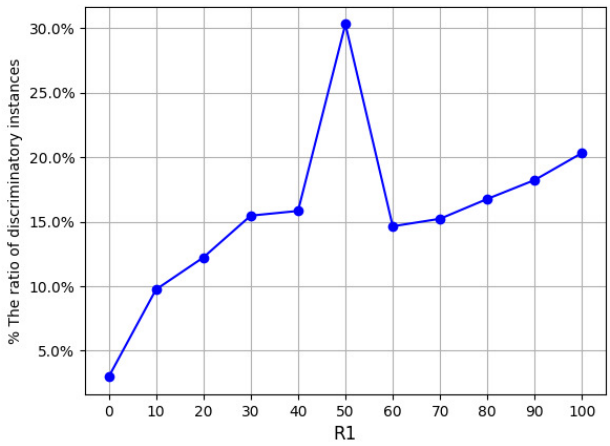

Fig. 3. r_1 (Census Income MLP, $r_2 = r_3 = 0$)

2) r_3: Similarly, we set $r_1 = 0, r_2 = 0$ to evaluate the effect of r_3. Fig. 4 reports the results of 200 episodes with $r_3 \in [-20, 0]$ for the Census Income MLP model. It can be seen that negative r_3 rewards can guide the agent away from the nondiscriminatory instances, while towards the discriminatory ones.

3) r_2: The reward constant r_2 is designed to avoid duplicate individual discriminatory inputs. Fig. 5 (respectively, Fig. 6) reports the results of 200 episodes with $r_2 \in [-5, 5]$ and $r_1 = 50, r_3 = -2$ (respectively $r_1 = 100, r_3 = -4$) for the Census Income MLP model. It can be seen that the positive r_2 rewards tend to result in more duplicate individual discriminatory inputs than the negative ones. Moreover, in both reward settings, the minimum G-ratio of duplicate individual discriminatory inputs is achieved with $r_3 < r_2 < 0$.

Moreover, the reward setting with $r_1 = 100, r_2 = -3$ and $r_3 = -4$ delivers less duplicate individual discriminatory inputs than the reward setting with $r_1 = 50, r_2 = -1$ and

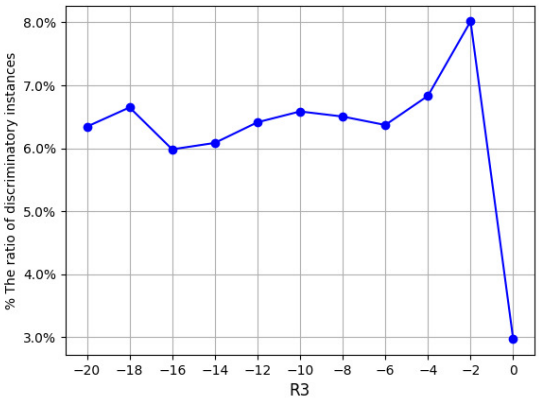

Fig. 4. r_3 (Census Income MLP, $r_1 = r_2 = 0$)

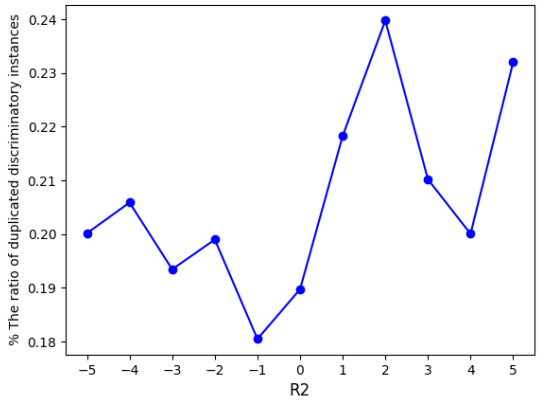

Fig. 5. r_2 (Census Income MLP, $r_1 = 50$, $r_3 = -2$)

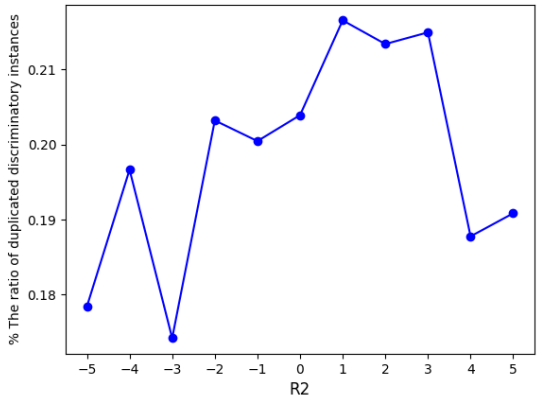

Fig. 6. r_2 (Census Income MLP, $r_1 = 100$, $r_3 = -4$)

$r_3 = -2$.

D. Initial Test Inputs

To evaluate the effect of initial test inputs, we randomly choose 10 inputs from the Census Income dataset as the initial test inputs, and run the main algorithm with $r_1 = 100, r_2 = -3, r_3 = -4$ until 100000 or 150000 test inputs are executed. TABLE VI reports the means and variances of the G-ratios of individual discriminatory inputs for the Census Income MLP model. It can be seen that with the increasing number of test inputs executed, the G-ratio of individual discriminatory inputs also increases, while preserving the very low variances. This indicates that our approach is rather stable and insensitive to the given initial test input.

TABLE VI
INITIAL TEST INPUTS (CENSUS INCOME MLP)

Number of Test Inputs	Mean	Variance
100000	21.01%	0.4%
150000	27.43%	0.5%

With the above experimental analysis, it can be concluded that the reward constants play the key role in fairness test generation, independent of the given initial test input. Specially, a reward setting with $r_1 > 0 > r_2 > r_3$ can guide the agent to hit more individual discriminatory inputs in an efficient manner.

V. RELATED WORK

The related work on fairness testing of machine learning models has been discussed and evaluated comparatively in the previous sections. In this section we further discuss other related work in machine learning testing and reinforcement learning.

Adaptive Testing of Machine Learning Models Adaptive test generation features in generating test cases based on the previously generated ones so that the generated test cases can fulfil certain diversity criteria. This technique has been applied in machine learning testing. In [26], Spieker & Gotlieb adopted contextual bandits to guide the selection of metamorphic relations for testing machine learning models. Reinforcement learning can be regarded as an extension of contextual bandits. In our work, the property of individual fairness serves as the unique metamorphic relation. We use reinforcement learning to guide the selection of test cases through assigning a series of rewards to separate actions.

Software Testing Using Reinforcement Learning Reinforcement learning has also been applied to test Android applications, although in a different way. In [27]–[30], reinforcement learning is used to abstract the GUI behavior models of Android applications, while our work use reinforcement learning to optimize the test generation process. The reinforcement learning scheme presented in [19] is similar to ours, but uses an on-policy learner for property-based testing of traditional software. In [19], rewards are determined at the end of each episode based on the diversity of test inputs. Instead, we uses an off-policy learner (Deep Q-Network) for testing machine learning models, and rewards are determined step-by-step in

order to improve the defect detection efficiency of the testing process.

VI. Conclusion

In this paper, we have presented a black-box fairness testing approach based on deep reinforcement learning. This new approach rephrases the individual fairness test generation problem from the perspective of reinforcement learning. It not just follows the principle of combining global and local search strategies as suggested in previous work, but also integrates the power of target optimization from reinforcement learning to achieve an optimal strategy for individual fairness test generation. Comparative experiments with the typical fairness benchmark models have shown that our reinforcement-learning based black-box fairness testing approach outperforms greatly the state-of-the-art black-box fairness testing approaches in its performance of generating individual discriminatory inputs. As future work, we would like to study other definitions of reward functions in the context of black-box testing, and to extend our reinforcement learning based testing framework for other aspects of machine learning models. Furthermore, it is still an open problem how to determine the ideal G-ratio of individual discriminatory inputs for fairness testing of a machine learning model.

Acknowledgment

We thank the anonymous referees for their valuable comments. This work is partially supported by the National Key R&D Program of China under the Grant No. 2017YFB0801900, and the CAS-INRIA Joint Research Program under the Grant No. GJHZ1844.

References

[1] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[2] M. Fatima, M. Pasha *et al.*, "Survey of machine learning algorithms for disease diagnostic," *Journal of Intelligent Learning Systems and Applications*, vol. 9, no. 01, p. 1, 2017.

[3] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep face recognition: A survey," in *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2018, pp. 471–478.

[4] F. D. Paiva, R. T. N. Cardoso, G. P. Hanaoka, and W. M. Duarte, "Decision-making for financial trading: A fusion approach of machine learning and portfolio selection," *Expert Systems with Applications*, vol. 115, pp. 635–655, 2019.

[5] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[6] M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," in *Artificial Intelligence and Statistics*, 2017, pp. 962–970.

[7] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[8] P. ZHANG, J. WANG, J. SUN, G. DONG, X. WANG, X. WANG, J. S. DONG, and D. TING, "White-box fairness testing through adversarial sampling," in *Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020)*, 2020, pp. 1–12.

[9] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 498–510.

[10] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 98–108.

[11] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 625–635.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[16] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1040–1048.

[17] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[18] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[19] S. Reddy and C. Lemieux, "Quickly generating diverse valid test inputs with reinforcement learning icse 2020," in *International conference on software engineering (ICSE'2020)*, 2020.

[20] N. Grgić-Hlača, M. B. Zafar, K. P. Gummadi, and A. Weller, "The case for process fairness in learning: Feature selection for fair decision making," in *Symposium on Machine Learning and the Law at the 29th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016, p. 11 p.

[21] M. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," in *the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4069–4079.

[22] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv*, pp. arXiv–1511, 2015.

[24] MorvanZhou, *Reinforcement Learning Methods and Tutorials*, 2020. [Online]. Available: https://github.com/MorvanZhou/Reinforcement-learning-with-tensorflow

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[26] H. Spieker and A. Gotlieb, "Adaptive metamorphic testing with contextual bandits," *Journal of Systems and Software*, p. 110574, 2020.

[27] D. Adamo, M. K. Khan, S. Koppula, and R. Bryce, "Reinforcement learning for android gui testing," in *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 2018, pp. 2–8.

[28] Y. Koroglu, A. Sen, O. Muslu, Y. Mete, C. Ulker, T. Tanriverdi, and Y. Donmez, "Qbe: Qlearning-based exploration of android applications," in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018, pp. 105–115.

[29] T. A. T. Vuong and S. Takada, "A reinforcement learning based approach to automated testing of android applications," in *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 2018, pp. 31–37.

[30] M. Pan, A. Huang, G. Wang, T. Zhang, and X. Li, "Reinforcement learning based curiosity-driven testing of android applications," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 153–164.