Automatic Fairness Testing of Neural Classifiers Through Adversarial Sampling

Peixin Zhang, Jingyi Wang, Jun Sun, Xinyu Wang, Guoliang Dong, Xingen Wang, Ting Dai, and Jin Song Dong

Abstract—Although deep learning has demonstrated astonishing performance in many applications, there are still concerns about its dependability. One desirable property of deep learning applications with societal impact is fairness (i.e., non-discrimination). Unfortunately, discrimination might be intrinsically embedded into the models due to the discrimination in the training data. As a countermeasure, fairness testing systemically identifies discriminatory samples, which can be used to retrain the model and improve the model's fairness. Existing fairness testing approaches however have two major limitations. First, they only work well on traditional machine learning models and have poor performance (e.g., effectiveness and efficiency) on deep learning models. Second, they only work on simple structured (e.g., tabular) data and are not applicable for domains such as text. In this work, we bridge the gap by proposing a scalable and effective approach for systematically searching for discriminatory samples while extending existing fairness testing approaches to address a more challenging domain, i.e., text classification. Compared with state-of-the-art methods, our approach only employs lightweight procedures like gradient computation and clustering, which is significantly more scalable and effective. Experimental results show that on average, our approach explores the search space much more effectively (9.62 and 2.38 times more than the state-of-the-art methods respectively on tabular and text datasets) and generates much more discriminatory samples (24.95 and 2.68 times) within a same reasonable time. Moreover, the retrained models reduce discrimination by 57.2 and 60.2 percent respectively on average.

Index Terms—Deep learning, fairness testing, individual discrimination, gradient

1 Introduction

In recent years, deep learning (DL) has been applied to various areas of our daily life, e.g., face recognition [1], fraud detection [2], and natural language processing (NLP) [3]. As DL is increasingly connected with our society, we cannot simply regard it as a mathematical abstraction, but rather as a society-technical system [4]. In other words, besides testing the accuracy (i.e., a quantification for evaluating the effectiveness of mathematical approximation) of the DL model, we also need to take ethical properties into consideration. Among them, fairness (i.e., non-discrimination) is one of the properties that raise the most heightened public concern [5], especially in minority and vulnerable groups. In [6], Tramer *et al.* demonstrated that the existing discrimination in our society may be present in the training data and learned by DL models unintentionally. Worse yet, discrimination in DL is often more 'hidden' than that of traditional decision-making software since it is still an open problem on how to interpret DL models. Therefore, it is crucial to have systematic methods for automatically identifying discrimination in DL systems.

Various forms of discrimination exist in the machine learning literature, including group discrimination [7], [8] and individual discrimination [9], [10]. Discrimination is often defined over a set of protected attributes[1], such as age, race and gender. Intuitively, discrimination happens when a machine learning model makes different decisions for different *individuals* (individual discrimination) or *subgroups* (group discrimination) differentiated only by protected attributes. Note that the set of protected attributes is often application-dependent and given in advance. We refer the readers to [11] for detailed definitions of fairness.

In this work, we focus on the problem of individual discrimination and aim to develop a systematic and scalable approach for identifying/generating discriminatory samples for DL models, i.e., a pair of samples that differ only by some protected features but have different labels. Note that in a given DL model, it is often inadequate to identify a few samples demonstrating the existence of individual discrimination. Instead, we need to generate as many discriminative samples as possible and use them to improve the fairness (i.e., mitigate the discrimination) of the DL model by retraining. In the software engineering literature, there have been multiple attempts on the problem [12], [13], [14]. For instance, THEMIS [12] randomly samples each attribute

- *Peixin Zhang, Jingyi Wang, Xinyu Wang, Guoliang Dong, and Xingen Wang are with the Zhejiang University, Hangzhou, Zhejiang 310027, China. E-mail: {pxzhang94, wangjyee, wangxinyu, dgl-prc, newroot}@zju.edu.cn.*
- *Jun Sun is with the Singapore Management University, Singapore 188065, Singapore. E-mail: junsun@smu.edu.sg.*
- *Ting Dai is with the Huawei International Pte. Ltd., Shenzhen 518129, China. E-mail: daiting2@huawei.com.*
- *Jin Song Dong is with the National University of Singapore, Singapore 119077, Singapore. E-mail: dcsdjs@nus.edu.sg.*

Manuscript received 22 Oct. 2020; revised 10 Apr. 2021; accepted 16 July 2021. Date of publication 4 Aug. 2021; date of current version 19 Sept. 2022.
(Corresponding authors: Jingyi Wang and Xinyu Wang.)
Recommended for acceptance by Y. Brun.
Digital Object Identifier no. 10.1109/TSE.2021.3101478

1. We use 'protected'/'sensitive' and 'attribute'/'feature' interchangeably.

within its domain and identifies those discriminative samples. AEQUITAS [13] improves the testing effectiveness with a two-phase (global and local) search. SG [14] combines the local explanation model [15] and symbolic execution [16] to increase the diversity and the number of discriminatory samples.

Existing approaches are developed mostly for traditional machine learning models, i.e., logistic regression, support vector machine, and decision tree. Although they could be applied to DL models, experiment results show that their performance on DL models is far worse than that on traditional models, and are far from being practical. We discuss the detailed shortcomings for each approach in Section 4.3. Besides, the aforementioned algorithms only work on simple tabular data and are not applicable to complex data formats (e.g., text for NLP tasks). The difficulty is two-fold. First, the features of most tabular data are well-designed and relatively independent from each other, which allows approaches such as THEMIS and AEQUITAS to generate new values independently for each feature. For text, each token (i.e., word or punctuation) is tightly embedded in its context. As a result, the discriminatory samples generated by THEMIS and AEQUITAS (if they manage to work) would unlikely be valid. Second, the text for NLP models is often pre-processed through an embedding procedure, which converts tokens to distributed and non-continuous vectors. Such vector inputs further make constraint solving based approaches such as SG inapplicable since it is challenging to solve constraints on such vectors.

To address the challenges, we propose a novel scalable gradient-based algorithm named *A*dversarial *D*iscrimination *F*inder (ADF) for generating discrimitive samples, which is specifically designed for DL models. The gradient is an effective tool to craft test inputs for DL models. It can be calculated efficiently and offers intuitive guidance on how a model's outcome changes with respect to certain attributes. It is proved to be useful in various DL-related tasks. It is noticeably utilized in recent works to generate adversarial samples [17], [18], [19], [20], [21], i.e., samples which are only slightly different from a given one in the training set yet result in very different model prediction. Inspired by these works, we use the gradient as an effective way for searching discrimitive samples.

An overview of ADF is presented in Fig. 1. ADF has two phases, i.e., global generation and local generation. The goal of the global generation is to increase diversity in the generated discriminatory samples. During the global stage, ADF selects samples in the raw dataset and uses the gradients to guide the search of discriminatory samples, by maximizing the difference between the model predictions of two similar samples. Then local generation takes these identified discriminatory samples as seeds to acquire more discriminatory samples by exploring their neighbors. Here, ADF utilizes the gradient in a different way to identify discriminatory samples which are minimally different from the seeds while maintaining their models' outputs. Note that ADF is generative, i.e., it may generate samples that are not in the original dataset. In order to make sure the generated samples are valid, we utilize a clip function on tabular datasets and semantically similar words on text datasets in both global and local generation phases.

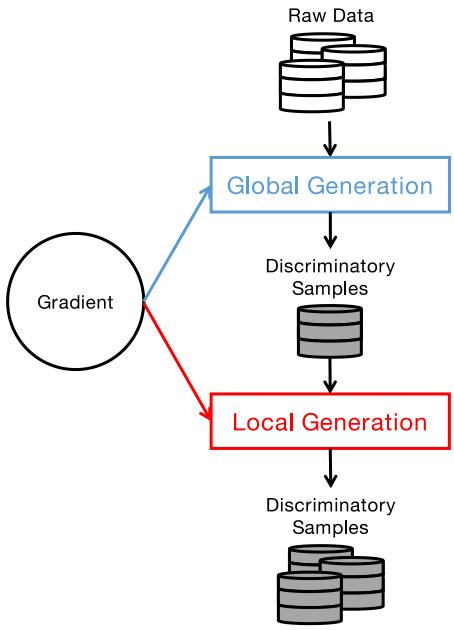


Fig. 1. An overview of ADF.

ADF has been implemented as in a self-contained toolkit. Our experiments on 22 real-word benchmarks (including both tabular and text datasets) show that ADF signficantly outperforms the baseline methods. For tabular data, ADF explores 8 times more input space and generates 24 times more discriminatory samples on average than AEQUITAS. Comparing with SG, ADF has an average of 324 and 32 percent higher success rate in global generation and local generation. Furthermore, ADF is around 2 and 7 times more efficient than AEQUITAS and SG. For text data, ADF generates 2.68 times discriminatory samples and has a 9.01 percent more success rate than random perturbation on average within a same reasonable time.

In summary, we make the following contributions.

- We propose a systematic and scalable algorithm ADF for generating discriminatory samples of DL models efficiently and effectively based on gradient.
- We address the challenges of applying fairness testing for complex data format (beyond tabular data), i.e., text.
- We evaluate ADF with 22 benchmarks on 5 datasets (including 3 tabular datasets and 2 text datasets). Our experiment shows ADF is significantly more effective and efficient in generating discriminatory samples than state-of-the-art methods, e.g., AEQUITAS [13] and Symbolic Generation [14].
- We implement and publish ADF as an online end-to-end service [22] to improve the fairness of DL models.

This paper is an extension of our previous publication [23] by applying the original idea of ADF to a much more challenging domain, i.e., text. In particular, we present the difficulties of why existing fairness testing approaches are infeasible for text and our approaches to address the challenges. We also provide several additional running examples and much more extensive experiments on text datasets to demonstrate the effectiveness of our approaches.

We organize the rest of the paper as follows. We first provide the background of DL and gradient-based adversarial attacks in Section 2. We present the details of the ADF in Section 4. In Section 5, we evaluate our approach on multiple real-world datasets and demonstrate its effectiveness and efficiency. We review the related works in Section 6. Finally, in Section 7, we conclude our work and discuss the potential future directions.

2 Preliminary

In this section, we review relevant background.

2.1 Deep Learning

We target two kinds of commonly used DL models, i.e., Feedforward Neural Network (FNN) and Recurrent Neural Network (RNN).

FNN A FNN contains an input layer, an output layer, and multiple hidden layers. We denote these layers as $L = \{L_j | j \in \{0, \ldots, J\}\}$, and the jth layer are consisted of $|L_j|$ neurons. A FNN computes the outcome of each neuron by applying activation function (e.g., Sigmoid, tanh, or relu [24]) ϕ to the weighted sum of the outputs of all the neurons in its previous layer.

RNN RNN is a variant of the traditional neural network, e.g., FNN, which is designed for modeling sequential data. An RNN takes a sequence $x = x_0 x_1 \cdots x_n$ as input and produces a sequence $o = o_0 o_1 \cdots o_m$ as output. We use $x[i]$ to denote the ith element of x and $|x|$ to denote the length of x. RNN 'memorizes' what has been calculated so far through a set of hidden states H. At each time step t, the current hidden state h_t and the output o_t are calculated as follows.

- x_t is the input at time t.
- h_t is the hidden state (which is also known as the memory unit) at time t. It can be calculated based on the last hidden state and the current input: $h_t = \phi(U \cdot x_t + W \cdot h_{t-1})$, where ϕ is the activation function, U and W are the weights of current input and previous state respectively. Specifically, h_{-1} is initialized to be 0 for calculating the first hidden state.
- o_t is the output at time t which is obtained by $o_t = \psi(V \cdot h_t)$, where ψ is normally a Softmax function, V is the weight of the current hidden state.

In this work, we focus on DL classifier $\mathcal{D} : X \rightarrow Y$, i.e., for a given sample $x \in X$, a DL model outputs a predicted label $y \in Y$ which has the highest probability. Note that we use θ to denote the set of its parameters.

2.2 DL Ethics

Given that DL models are increasingly applied in applications with significant societal impact, they must act following the formal (i.e., Law) and informal norms (i.e., Ethics) held for human beings. In other words, in addition to improving the accuracy of the models, we hope to develop Trustworthy AI, since only when human beings trust AI, can they harvest the benefits of this technology confidently and adequately.

The High-level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission lists 10 fundamental requirements of Trustworthy AI [5], e.g., *robustness* requires DL models to handle errors or inconsistencies in its life cycle [17], [19], [20], [25]; *transparency* aims to help developers and users better understand the logic and reasons in the decision-making process [15], [26]; and *privacy* ensures that all personal information used and generated in the intersection between users and DL models are not leaked [27], [28]. In addition, *fairness (non-discrimination)* is a new research hotspot in the field of AI ethical principles in recent years, which attempts to avoid DL models that intentionally or unintentionally marginalize certain minorities.

So far, there is no commonly agreed definition of fairness, despite that many definitions have been proposed in the literature, e.g., Demographic Parity and Equalised Odds. In this work, we focus on individual fairness, i.e., given two samples that only differ by the protected attributes, a model must output the same label. The formal definition of individual fairness is in the next section.

2.3 Individual Discrimination

We denote X as a dataset and its set of elements by $E = \{E_1, E_2, \ldots, E_n\}$, i.e., an element is an attribute for tabular data and a token for text data. Assuming each element E_i has a valuation domain $\mathbb{I}_i$, the input domain is then $\mathbb{I} = \mathbb{I}_1 \times \mathbb{I}_2 \times \ldots \times \mathbb{I}_n$, which denotes all the possible combinations of element valuations. Note that given an unknown sample, n is a constant and each attribute has its own domain for tabular data, whereas for text data, n varies with inputs and the input space is the entire corpus. Further, we use $P \subset A$ to denote a set of protected attributes like race and gender and NP to denote the set of non-protected attributes. It is easy to obtain the protected attributes and their value domains for tabular data. However, for the text, we need to identify a list of features that are often associated with fairness issues as protected attributes and define the value domain for each one manually. For instance, one of the protected attributes is 'country' and it has 160 possible values. Note that this is a one-time effort that is manageable and reusable. The details of the identified protected attributes and their value domains are available at [22]. A DL model $\mathcal{D}$ trained on X may contain individual discrimination which is defined as follows.

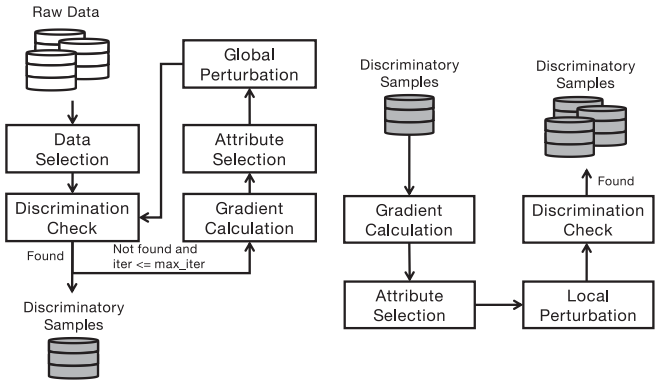
Definition 2.1. *Let $x = \{x_1, x_2, \ldots, x_n\}$, where x_i is the value of attribute A_i be an arbitrary sample in $\mathbb{I}$. We say that x is a discriminatory sample of a model $\mathcal{D}$ if there exists another data sample $x' \in \mathbb{I}$ which satisfies the following conditions:*

- $\exists p \in P, s.t., x_p \neq x'_p$;
- $\forall q \in NP, x_q = x'_q$;
- $\mathcal{D}(x) \neq \mathcal{D}(x')$

Further, (x, x') is called a discriminatory sample pair.

2.4 Gradient-Based Adversarial Attack

In [29], Szegedy *et al.* demonstrated that DL models are vulnerable to adversarial samples, which are crafted by introducing perturbations on the original normal samples to mislead the decisions of the model. Various kinds of adversarial attacks have been proposed in the past few years [17], [18], [19], [20], [21], [30], and gradient-based methods are one of the fastest and most straightforward ways. The gradient of the prediction y with respect to the input x is defined as:



(a) Global Generation (b) Local Generation

Fig. 2. The pipeline of ADF.

$$G(x, y) = \frac{\partial J(\theta, x, y)}{\partial x}, \tag{1}$$

where J could be any objective function related with the output, e.g., the loss function [17], [18], [20], the logits (i.e., the input values of the last softmax layer) [19], [31]. In the following, we briefly introduce two representative gradient-based adversarial attacks for tabular data and text data.

FGSM In [17], Goodfellow *et al.* proposed Fast Gradient Sign Method (FGSM) which perturbs the original input in the direction of the sign of the gradient of the model's loss function with respect to the input attributes according to the following Equation:

$$x^{adv} = x + \epsilon \cdot \mathbf{sign}(G(x, y)), \tag{2}$$

where ϵ is a hyper-parameter to determine the perturbation degree.

ASC In [19], Papernot *et al.* proposed Adversarial Sequence Craft (ASC) which iteratively replaces an attribute with a substitute value that impacts the model's outcome significantly until success or timeout. In other words, it chooses the value with the closest direction (i.e., the sign of the difference between it and the original value) to the one indicated by the gradient.

HotFlip In [25], Ebrahimi *et al.* proposed a white-box adversarial attack method named HotFlip, which estimates the impact of each flip operation on one-hot input by the gradient, and utilizes the beam search to find the best perturbation with the highest loss.

2.5 Problem Definition

A model which suffers from individual discrimination may produce prejudiced decisions when a discriminatory sample is presented as input. Our problem is thus defined as follows. Given a dataset X, a set of protected attributes P and a DL model $\mathcal{D}$, how can we effectively and efficiently generate discriminatory samples for $\mathcal{D}$ so that we can retrain a DL model based on X and the generated discriminatory samples for more fair models? This problem is challenging because we focus on complex DL models which renders existing methods ineffective.

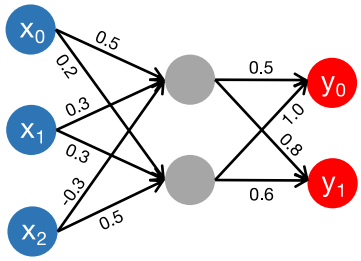


Fig. 3. Toy example of FNN without activation function.

3 Overview

In this section, we present the pipeline of global and local generation of our method ADF in Figs. 2a and 2b respectively and utilize a toy example to explain the relevant notations.

Fig. 3 depicts a one-layer FNN without activation function for a sensitive classification task. The input vector is $[x_0, x_1, x_2]$ (shown in blue), where all of them have continuous integer values and x_2 is the only protected attribute with two possible values, 0 and 1. y_0 and y_1 (shown in red) denote the output nodes. Besides, each value in the figure represents the corresponding weight between two linked neurons.

In the global phase, we first select seeds to search as many different parts of the input space as possible, e.g., [0,1,0]. Next, we follow Definition 2.1 to check if the output (y_0) of the seed is consistent when we only flip the sensitive attribute x_2. Since gradient indicates the direction and magnitude of the output change with respect to the perturbation on the input [17], we use gradient calculated according to the chain rule, e.g., $\partial y_0/\partial x_0 = 0.5 * 0.5 + 1.0 * 0.2 = 0.45$, to search for discriminatory samples more efficiently. Then we select the non-protected attribute(s) and perturb them by gradient, e.g., we choose the attribute x_0 (with larger gradient) and apply $x_0 = x_0 + 1 * sign(0.45)$, to acquire a sample that is more likely discriminatory. The new sample [1,1,0] is used to start the next iteration. The global generation stops if a discriminatory sample is successfully generated or it times out.

In the local phase, we further search the neighborhood of identified discriminatory samples to increase the quantity. Here, the gradient is computed to guide the selection of non-protected attribute(s) and perturbation to ensure that the search space is as close to the existing discriminatory sample as possible. Last, we check whether the new sample is discriminatory according to Definition 2.1.

4 Methodology

In this section, we first present details of our approach ADF and then a qualitative comparison between our approach and state-of-the-art fairness testing approaches.

ADF generates discriminatory samples in two phases, i.e., a global generation phase and a local generation phase. In the global generation phase, we aim to identify those discriminatory samples near the decision boundary from the original dataset X, which then serve as the seed data for the local generation phase. In the local generation phase, we follow the intuition that samples in the neighborhood of those seed data

== ???? == This 'list' serves no purpose other than to provide ammunition for those already holding racist views. Also, many of the terms related to Jews and African Americans are clearly fabricated by some racist individual with too much time on their hands.

(a) Original sample

== ???? == This 'list' serves no purpose other than to provide ammunition for those already holding racist views. Also, many of the terms related to ~~Jews~~ Gypsies and African Americans are clearly fabricated by some racist individual with too much time on their hands.

(b) IDS identified by perturbation on non-IDS

== ???? == This 'list' serves no purpose other than to provide ammunition for those already holding racist views. Also, ~~many~~ several of the terms related to Gypsies and African Americans are clearly fabricated by some racist individual with too much time on their hands.

(c) IDS identified by perturbation on IDS

Fig. 4. Sample searching results.

are more likely to be discriminatory samples to obtain more of them. Note that this intuition is inspired by recent research on the robustness of DL models [29]. In the following, we introduce the two phases in detail with two running examples (from tabular and text dataset respectively).

Example 4.1. We use the Census Income dataset[2] as a running example on tabular data to illustrate each step of our approach. The Census Income dataset is published in 1996, which is a commonly used dataset in the literature of fairness research [12], [13], [14], [32], [33]. The task is to predict whether the income of an adult is above 50,000$based on their personal information. The dataset contains 32,561 training samples with 13 attributes each. The following shows a sample x.

$$x : [4, 0, 6, 6, 0, 1, 2, 1, 1, 0, 0, 40, 100].$$

Note that all the attributes are category attributes (obtained through binning). Among the 13 attributes, there are multiple potential protected attributes, i.e., age, race, and gender. In the following, we assume the protected attribute is gender for simplicity, whose index in the feature vector is 8 (which is highlighted in red above). There are only two different values for this attribute, i.e., 0 representing female and 1 representing male. Given a model trained on the dataset, if changing 1 to 0 changes the prediction outcome by the model, we say that x is a discriminatory sample for the model.

Example 4.2. We utilize the text dataset Wikipedia Comments[3] as a running example on text data. This dataset is collected from Wikipedia Talk Pages for classification [34]. The classification task is to label each comment as either toxic or non-toxic. It was also adopted for measuring discrimination [35], [36]. It has around 127,000 records with an average length of 81 words. The sensitive features include religion, country, ethnic and race. Here we assume the protected attribute is ethnic. Fig. 4 a shows a sample x, note that the part "==????==" is the heading.

2. https://archive.ics.uci.edu/ml/datasets/adult
3. https://github.com/conversationai/unintended-ml-bias-analysis

4.1 Global Generation

Algorithm 1 shows the details of the global generation phase. The algorithm uses the following constants: *num_g* which is the number of seed samples to generate during global generation; and *max_iter* which is the number of maximum iteration.

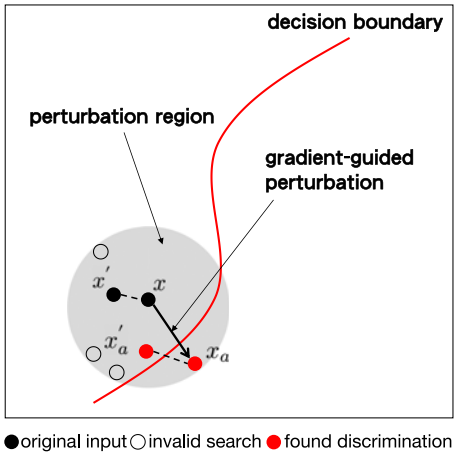
Algorithm 1. Global Generation

```
1:  id_g = ∅
2:  for i from 0 to num_g do
3:    Get seed x from X
4:    Determine the locations of P
5:    for iter from 0 to max_iter do
6:      if discrimination_check(x) then
7:        id_g = id_g ∪ x
8:        break
9:      end if
10:     X_s = {x'|∀x'_p ∈ I_p, x'_p ≠ x_p}
11:     x' = arg max{abs(O_y(x') − O_y(x))|x' ∈ X_s}
12:     g = ∇_x J(θ, x, y)
13:     g' = ∇_x' J(θ, x', y)
14:     x = global_perturbation(x, grad, grad')
15:   end for
16: end for
17: return id_g
```

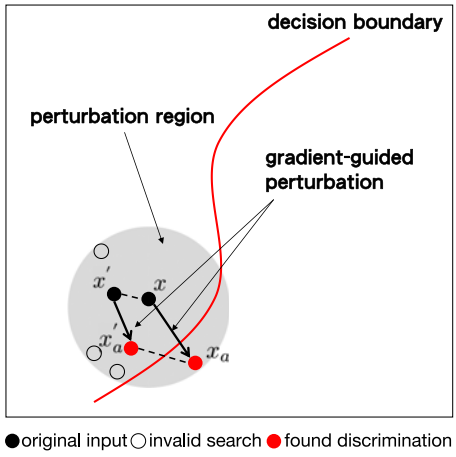
In the loop from lines 5 to 15, we generate discriminatory samples iteratively based on the gradient. Let θ be the parameters of a DL model $\mathcal{D}$; y be the ground-truth label associated with x; and $J(\theta, x, y)$ be the objective function, e.g., the loss function or the logits. Given a seed x, we first check whether it is a discriminatory sample according to Definition 2.1 at line 6. The details are shown in Algorithm 2. Its key idea is to enumerate the value domain of the protected attributes (see lines 3- 8) and check whether the model labels the modified sample differently. The sample is deemed to be a discriminatory sample at line 6 if the label changes. Note that the complexity of the checking is $\Theta(N)$, where N is the number of all the possible combinations of the protected features in the corresponding domain. If x is not a discriminatory sample, we start to search for discriminatory samples based on x with the guidance of the gradient defined as $\nabla_x J(\theta, x, y)$.

Algorithm 2. Discrimination Check

```
1:  for each element t in x do
2:    if t ∈ I_p then
3:      for v in I_p\{t} do
4:        Obtain x' by replacing t in x with v;
5:        if D(x) ≠ D(x') then
6:          return True
7:        end if
8:      end for
9:    end if
10: end for
11: return False
```



(a) Adversarial Attack



(b) Fairness Testing

Fig. 5. Intuition of gradient-based approach.

Notice that in order to identify a discriminatory sample, we need to find a discriminatory sample pair, i.e., a pair of samples that differ only by some protected attributes and yet have different labels. In other words, given x, we need to first identify an x' which only differs from x in protected attributes. Since x is not a discriminatory sample, x and x' thus have the same label. There are two possible ways of perturbation to obtain discriminatory samples. The first one (shown in Fig. 5a) is directly pushing the sample x towards the decision boundary by adversarial attack [17], [18], [19], [20], [25], and then checking whether it is successful to acquire a discriminatory sample. In this case, if all possible $\{x'\}$ are close to x, they will likely cross the decision boundary together after the perturbation. Thus, the distance between x and x' must be considered. As shown in Fig. 5b, we propose a more fine-grained strategy, which utilizes gradient information on x and x' simultaneously to offer guidance on how to perturb (x, x') such that we are most likely to identify a discriminatory sample pair.

We identify a set of samples X_s from $\mathbb{I}$ such that x and any sample x' in X_s only differ in some protected attributes at line 10. The goal is to perturb (x, x') such that $\mathcal{D}(x) \neq \mathcal{D}(x')$. Among all samples in X_s, we choose x' according to the following equation:

$$x' = \arg\max\{abs(\mathcal{O}_y(x') - \mathcal{O}_y(x)) | \forall x'_p \in \mathbb{I}_p, x'_p \neq x_p\}, \tag{3}$$

where $\mathcal{O}$ denotes the output vector of $\mathcal{D}$. The intuition is to select the sample x' such that the outputs of the DL model on x and x' are maximally different. In such a way, after we perturb both x and x', it is more likely that the predicted labels of x and x' are different.

Our next step is to perturb x and x' to generate a discriminatory sample pair (x_{id}, x'_{id}) such that $\mathcal{D}(x_{id}) \neq \mathcal{D}(x'_{id})$. Note that the perturbation introduced to x and x' are always the same so as to make sure the pair still only differ by protected attributes after the perturbation. In order to minimize the perturbation, we utilize different strategies on tabular and text data since they have different characteristics. For tabular data, since the attributes are all preprocessed as categorized values, the perturbation is done by increasing or decreasing its value by 1 unit. For text data, we select the perturbation token among candidates which are similar to the original one semantically and syntactically. A *perturbation* in our context is thus a function of a set of non-protected attributes which we choose to perturb and a corresponding selection vector determining the perturbing direction or token. Formally,

Definition 4.1. Perturbation*A perturbation δ on a data sample x is function $\delta : \mathbb{I} \times NP \times \mathbb{S} \rightarrow \mathbb{I}$, where $\mathbb{S}$ is the selection of the perturbation.*

Specifically, for tabular data, $\mathbb{S}$ is a boolean vector where 1 means increasing the attribute value by 1 and -1 means decreasing by 1. For text data, $\mathbb{S}$ is a substituting token vector.

Algorithm 3. Global Perturbation (Tabular)

```
1:  Initialize array dir with the same size as A by 0
2:  for a ∈ A\P do
3:    if sign(g_a) = sign(g'_a) then
4:      dir_a = sign(g_a)
5:    end if
6:  end for
7:  x = x + dir * s_g
8:  x = clip(x)
9:  return x
```

Our next question is how to choose the elements and directions for perturbation. Notice that to better achieve individual discrimination, we need to maximize the difference between $\mathcal{D}(x_{id})$ and $\mathcal{D}(x'_{id})$ after perturbation. Our goal is thus:

$$argmax_{\delta(x,x')}\{\mathcal{D}(x_{id}) - \mathcal{D}(x'_{id})\}, \tag{4}$$

where $x_{id} = \delta(x)$ and $x'_{id} = \delta(x')$. Unfortunately, this objective can not be directly optimized. Our remedy is to adopt the idea of the EM algorithm [37] in machine learning to iteratively optimize it. Similarly, we discuss the different treatments according to the data type. For the tabular data, as showed in Algorithm 3, we utilize the gradient of $\nabla_x J(\theta, x, y) - \nabla_x J(\theta, x', y)$ and select those attributes which have similar contributions (with the same sign of gradients) as attributes to perturb. The intuition is that perturbing these attributes can potentially enlarge the output difference since $\nabla_x J(\theta, x, y) - \nabla_x J(\theta, x', y)$ equals 0 on them (likely to be local minimum). The parameter s_g at line 3 is the step

size of global generation. Notice that in order to filter out unreal perturbed inputs, we always apply a Clip function described in Algorithm 4 to make sure that the value of each attribute after the perturbation is within its domain (see line 8).

Algorithm 4. Clip

1: Let x be the input
2: Let $\mathbb{I}$ be the input domain
3: **for** $x_i \in x$ **do**
4: $x_i = \max(x_i, \mathbb{I}_i.min)$
5: $x_i = \min(x_i, \mathbb{I}_i.max)$
6: **end for**
7: **return** x_i

Example 4.3. For our tabular running example, we select seed sample from the original dataset, then get the first seed x is as follows (shown in Example 4.1), and it is not a discriminatory sample.

$$x : [4, 0, 6, 6, 0, 1, 2, 1, 1, 0, 0, 40, 100].$$

We identify all samples which differ from the seed only by protected attributes and then obtain the following x' which has the greatest difference in output probability with x.

$$x' : [4, 0, 6, 6, 0, 1, 2, 1, 0, 0, 0, 40, 100].$$

We then determine the perturbation direction based on the sign of two samples' gradients as follows.

$$direction : [0, 1, 0, 0, -1, 1, -1, 0, 1, 0, 0, 1, -1].$$

Intuitively, 0 means that the corresponding attribute should not be changed; -1 means that it should be decreased and 1 means that it should be increased (to maximize output difference). Next, we perturb x accordingly and apply the Clip function to filter invalid values. The result is the following sample.

$$x : [4, 1, 6, 6, 0, 2, 1, 1, 1, 0, 0, 41, 39].$$

Since the last attribute *native-country* only has 40 countries (and value 100 means missing value in the original data), it is modified to 39 (the maximum value) by the Clip function. After checking at line 6, it is shown to be a discriminatory sample.

For models trained on text data, we propose two different global generation strategies, both of which are based on gradient. The first one is inspired by ASC [19] (shown in Algorithm 5), i.e., to maintain the semantics and syntax, we only choose the most impactful token to perturb (see line 1), instead of perturbing multiple attributes at the same time. Recall that the tokens in the text are pre-processed by word embedding, which projects each token to a numerical vector, i.e., the gradient is calculated with respect to the projected embedding vectors (instead of the token directly). Formally, the impact of a non-sensitive token np is measured by

$$imp(np) = ||abs(g_{np} - g'_{np})||_\infty. \quad (5)$$

where $abs(\cdot)$ returns the absolute value of $\cdot$; and $||\cdot||_\infty$ is the L_∞ norm of a vector $\cdot$. After getting the most impactful token, we replace it with one of its synonyms at line 1. Among the top-k synonyms, denoted as C, we select the one which incurs a perturbation whose direction is closest to the direction indicated by the gradient of x. The intuition is to push the sample towards the classification boundary along the gradient so that it is more likely to be a discriminatory sample.

Algorithm 5. Global Perturbation (Text, ASC)

1: $a = \arg\max_{a \in \{A \backslash P\}} imp(a)$
2: $C = x_a.simlar\_words$
3: $x_a = \arg\min_{c \in C} \|sign(x_a - c) - sign(g_a)\|$
4: **return** x

Algorithm 6. Global Perturbation (Text, HF)

1: $min\_dif = 0$
2: **for** $a \in \{A \backslash P\}$ **do**
3: $C = x_a.simlar\_words$
4: **for** $c \in C$ **do**
5: $dif = |grad_a \cdot (x_a - c)^T - grad'_a \cdot (x'_a - c)^T|$
6: **if** $dif > min\_dif$ **then**
7: $pt = a, sw = c$
8: **end if**
9: **end for**
10: $x_{pt} = sw$
11: **end for**
12: **return** x

The other method inherits the key idea of HotFlip [25], as shown in Algorithm 6. We substitute each word (see lines 2- 11) to search for the maximum difference in terms of the outcome change with respect to the input pair x and x' (see line 5). Since we use word embedding to preprocess the input, instead of one-hot representation, it is inapplicable to choose the best word flip $w_i \rightarrow w_j$ based on $grad_j - grad_i$ (refer readers to [25] for more details), we utilize $grad_a \cdot (x_a - c)^T$ to estimate the change of prediction, where $\cdot^T$ is the transpose function. Notice that since here we only consider the non-protected tokens, $x'_a = x_a$.

Once we determine the perturbation, we apply it on x and x' and then check whether the new pair (x_{id}, x'_{id}) is a discriminatory sample pair. If the answer is yes, the algorithm breaks out immediately (see lines 6-8). Otherwise, we start another round of perturbation on (x_{id}, x'_{id}). This process may repeat multiple times until it succeeds.

Example 4.4. For our text running example shown in Fig. 4a, it is also determined to be a non-discriminatory sample. We thus apply global generation (Algorithm 5) to acquire a discriminatory sample based on x. We first enumerate all samples by systematically replacing the sensitive word "African" with all other possible values for ethnicity. Among all texts, we select x' where 'African' is replaced by 'Latino' whose prediction result is maximally different from the original text. Then we identify the non-sensitive token "Jews" (highlighted in red) which has the most impact on the prediction result according to the gradients. Next, we replace "Jews" with the top-10 synonyms in the

given dictionary, which are *Jewish, Christians, Nazis, Catholics, Muslims, Arabs, Jew, Germans, Gypsies, and Orthodox*. "Gypsies" is chosen to replace "Jews" according to the above discussion. The resultant text is shown in Fig. 4b is a discriminatory sample.

4.2 Local Generation

After the global generation phase, we obtain a set of discriminatory samples as seeds for the local generation phase. The goal of the local generation phase is to generate as many discriminatory samples as possible based on the seeds, which are useful for re-training the DL model. The intuition behind the design of the local generation is that a well-trained DL model is likely robust, i.e., if two samples are similar, the same prediction is likely to be produced by the DL model. We thus are likely to find more discriminatory samples around a given seed discriminatory sample.

Algorithm 7. Local Generation

1: id_l = ∅
2: **for** $x \in id\_g$ **do**
3: **for** i from 0 to num_l **do**
4: $X_s = \{x^{'} | \forall x_p^{'} \in \mathbb{I}_p, x_p^{'} \neq x_p\}$
5: $\exists x^{'} \in X_s, \mathcal{D}(x) \neq \mathcal{D}(x^{'})$
6: $g = \nabla_x J(\theta, x, y)$
7: $g' = \nabla_{x'} J(\theta, x', y)$
8: $prob = normalization(g, g^{'})$
9: Select $a \in A \backslash P$ with probability $prob$
10: x = local_perturbation(x, a)
11: **if** $discrimination\_check(x)$ **then**
12: id_l = id_l ∪ x
13: **end if**
14: **end for**
15: **end for**
16: **return** l_id

Our local generation algorithm has the following parameters: *num_l* which is the number of trials in the local generation. The algorithm makes use of the gradients of the objective function (see lines 6-7) in a different way. Recall that in the global generation, we would like to maximally change the output of the DL model. On the contrary, in the local generation, we would like to minimally change the output, as our goal is to maintain the DL model's outputs of the discriminatory sample pairs identified in the global generation so that they remain different. We thus choose to perturb those attributes which have the least impact on the output. Note that the absolute value of gradient represents how much an attribute contributes to the outcome.

Further note that since we are perturbing a discriminatory sample pair, we need to consider the two inputs at the same time (to make sure that neither of them crosses the decision boundary as otherwise, they are no longer a discriminatory sample pair). To achieve that, we adopt a normalization process on the gradients on the two inputs to measure the average contribution of each attribute on the input pair. The details are shown in Algorithm 8. We first add the absolute value of two gradients together to get the saliency value of each element (see line 3). Recall that in the case of text data, the gradient with regard to the token is a vector, thus we take the L_∞ norm of it as impact. Then we calculate the reciprocal value (see line 4) since we aim to select the element with fewer contributions to the output and meanwhile filter out the protected attributes (see lines 5-7). Lastly, we use a standard normalization function to get the contribution of each attribute on the input pair (see lines 9-10).

Algorithm 8. Normalization

1: Initialize gradient with the same size of g
2: **for** i from 0 to gradient.length **do**
3: $saliency = \|abs(g_i)\|_\infty + \|abs(g_i')\|_\infty$
4: $gradient_i = 1.0/saliency$
5: **if** $A_i \in P$ **then**
6: $gradient_i = 0$
7: **end if**
8: **end for**
9: $gradient\_sum = sum(gradient)$
10: $probability = \{gd/gradient\_sum | \forall gd \in gradient\}$
11: **return** probability

Algorithm 9. Local Perturbation (Tabular)

1: Select $d \in [1, -1]$ with probability [0.5,0.5]
2: $x_a = x_a + d \times s\_l$
3: $x = clip(x)$
4: **return** x

Algorithm 10. Local Perturbation (Text)

1: $C = x_a.simlar\_words$
2: Select $c \in C$ with the same probability $1.0/C.length$
3: $x_a = c$
4: **return** x

Algorithm 7 shows the details of our local generation algorithm. Given a discriminatory sample pair (x, x') (see line 5), we start searching by iteratively selecting the attribute to perturb using the normalization of gradients (see line 8). Instead of modifying the chosen element based on the gradient, we randomly select the perturbation direction (see Algorithm 9) and substituting word (see Algorithm 10) for tabular and text data respectively with the uniform probability. Similar to global generation, we also utilize application-specific strategies (i.e., Clip for tabular and similar words for text) to make sure that the generated test case is valid. We check whether the input after the perturbation is a discriminatory sample (see line 11) and continue to the next seed input if the answer is yes. Otherwise, we start another iteration of the local generation (see line 3).

Example 4.5. For our tabular running example, the global generation phase generates the following discriminatory sample pair.

$$x : [4, 1, 6, 6, 0, 2, 1, 1, 1, 0, 0, 41, 39]$$

$$x' : [4, 1, 6, 6, 0, 2, 1, 1, 0, 0, 0, 41, 39].$$

In the local generation, taking this pair as input, we first calculate the gradient of these two samples and normalize the sum of them as individual probability. The result

is as follows.

$$probablity : [0.030, 0.019, 0.057, 0.075, 0.002, 0.009,$$

$$0.020, 0.015, 0, 0.002, 0.027, 0.612, 0.131].$$

Based on the above probability, we choose the attribute *hours-per-week* (with index 11) and the direction -1 for perturbation. Since the result sample's values are all within the respective domains, the Clip function keeps it the same and the following sample is identified as a new discriminatory sample.

$$x : [4, 1, 6, 6, 0, 2, 1, 1, 1, 0, 0, 40, 39].$$

Example 4.6. For our text running example, we apply local generation to search for more discriminatory samples on the neighbor of the identified one (shown in Fig. 4b). We first obtain the probability distribution based on the normalization of their gradients and randomly select the token "many" to perturb. Note that this token has little impact on the prediction result. Next, we obtain its top-10 synonyms and randomly replace "many" using "several". The resultant text is shown in Fig. 4c, which turns out to be a discriminatory sample.

4.3 Qualitative Evaluation

In the following, we evaluate our approach qualitatively by comparing it with state-of-the-art approaches, i.e., THEMIS [12], AEQUITAS [13], and Symbolic Generation (SG) [14]. Empirical comparison results are presented in Section 5.

THEMIS [12] explores the input domains for all attributes through random sampling and then checks whether the generated samples are discriminatory samples. AEQUITAS [13] improves THEMIS by adopting a two-phase generation framework. In the first phase, AEQUITAS randomly generates a set of discriminatory samples in the input space as seeds. In the second phase, AEQUITAS searches for more discriminatory samples around the seed inputs found in the first phase by randomly adding perturbations on the non-protected attributes. Notice that the perturbation is guided by a distribution that describes the probability of finding a discriminatory sample by adding perturbation on a specific non-protected attribute. Despite that random sampling is lightweight, THEMIS and AEQUITAS can miss many combinations of non-protected attributes' values where individual discrimination may exist [14]. The recent work SG [14] attempts to solve this problem by systematically exploring the input space through symbolic execution. The idea is to first adopt a local model explainer like LIME [15] to construct a decision tree for approximating the machine learning model. The result is a decision tree constituted with linear constraints such that a linear path constraint is associated with any given input. Then, SG iteratively selects (according to a ranking function), negates the constraints, and uses a symbolic execution solver to generate test cases according to different path constraints.

To summarize the difference between existing approaches and ours, we differentiate them using three criteria, i.e., whether the search (for discriminatory samples) is guided, whether the guided search is specific for an individual input (input-specific), and whether the procedure adopted is lightweight (and thus likely scalable). Table 1 shows the summary. Except for THEMIS, both AEQUITAS and SG generate discriminatory samples in a guided way (either by a distribution or a path constraint). The difference is that AEQUITAS uses a single distribution for all the inputs while SG generates path constraints depending on different inputs. We remark that designing input-specific perturbations is a more robust way to generate individual discrimination for different kinds of input and thus is important because it is crucial for removing individual discrimination globally. Lastly, we expect that approaches based on random sampling like THEMIS and AEQUITAS are lightweight while SG is a relatively heavy approach that requires the help of a local model explainer and a symbolic execution solver. For the former, it is still an open problem on generating model explainers in a scalable and accurate way. For the latter, symbolic execution is known to be less scalable than techniques like random samples. Besides, the above limitations also make these methods unsuitable for text application since it is challenging to obtain valid texts by randomly selecting words. For SG, it is impractical to solve the constraint after the embedding procedure.

TABLE 1
Comparing Different Approaches

Feature	THEMIS	AEQUITAS	SG	ADF
Guided	×	✓(semi)	✓	✓
Input specific	N.A.	×	✓	✓
Lightweight	✓	✓	×	✓

Compared to existing approaches, our approach satisfies all three criteria. First, our search is guided by gradient, i.e., the perturbation is guided towards the decision boundary to accelerate the discovery of discriminatory samples which significantly reduces the number of attempts needed. The intuition is visualized in Fig. 5. Second, our algorithm generates a specific gradient-guided search for different inputs, which significantly improves the success rate of individual discrimination generation. Lastly, our approach is lightweight since obtaining the gradient of DL models with respect to a given input is cheap which only requires a back-propagation process and is supported by all existing standard deep learning frameworks like Tensorflow [38], PyTorch, and Keras. And ADF is easy to be implemented on text data, as no matter on global or local generation phase, it selects the perturbed token and substitutes word only guided by gradients.

Similar to AEQUITAS, our approach also has a global search phase and a local search phase. The differences are in the details of both phases. AEQUITAS works by actively maintaining a probability distribution $t : NP \rightarrow [0, 1]$ on NP which represents how likely perturbing an attribute in NP is likely to successfully generate individual discriminatory samples. A limitation of such an approach is that different attributes of different inputs may contribute differently to the DNN output and the same global distribution hardly works for all the inputs. This is clearly evidenced by our experiment results in Section 5. To solve the problem, our approach takes an input-specific perspective, i.e., choosing

TABLE 2
Configuration of Experiments

Parameter	Value	Description
max_iter	10	max. iteration of global generation
s_g	1.0	step size of global generation
s_l	1.0	step size of local generation
k	10	the number of similar words for choosing

TABLE 3
Experimented DL Models

Dataset	Model	Accuracy
Census Income	Six-layer Fully-connected NN	88.15%
German Credit	Six-layer Fully-connected NN	100%
Bank Marketing	Six-layer Fully-connected NN	92.26%
Wiki Comments	3-layer 10-state LSTM	89.9%
	3-layer 10-state GRU	90.7%
Jigsaw Comments	3-layer 10-state LSTM	85.8%
	3-layer 10-state GRU	86.8%

different local perturbations based on the gradient which is specific to a given sample.

5 Experiment

We have implemented ADF as a self-contained toolkit based on Tensorflow [38] and Gensim [39]. Its source code, together with all the experiment-related details, is available online [22]. In the following, we evaluate ADF to answer 4 research questions (RQ).

5.1 Experimental Setup

For the tabular data, we choose AEQUITAS and SG for baseline comparison. Note that THEMIS is shown to be significantly less effective [14] and thus is omitted for comparison. We obtained the implementation of AEQUITAS from GitHub[4] and re-implemented SG according to the description in [14] since their implementation is not publicly available. Further notice that AEQUITAS proposed 3 different local search algorithms, we adopted the fully-directed algorithm in our evaluation since it has the best performance according to [13]. However, for text classification tasks, as all existing fairness testing approaches are specifically designed on tabular data and thus inapplicable, we compare ADF with a baseline approach which adopts the key idea of THEMIS and AEQUITAS, applying random perturbation (RP) instead of gradient-guided perturbation (to show the effectiveness of gradient-guided search). In particular, both on the global and local stage, RP selects the perturbed token and substitutes one with synonyms completely at random.

All experiments are conducted on a GPU server with 1 Intel Xeon 3.50GHz CPU, 64GB system memory, and 1 NVIDIA GTX 1080Ti GPU. Both AEQUITAS and SG are configured according to the best performance setting reported in the respective papers. Table 2 shows the value of parameters used in our experiment to run ADF.

We adopt 5 open-source datasets for fairness testing as our experiment subjects, including 3 tabular datasets and 2 text datasets. The details of the datasets are as follows.

- *Census Income*. The details of this dataset have been introduced in Example 4.1. It is used as a benchmark by AEQUITAS and SG [14].
- *German Credit*[5]. This is a small dataset with 600 data and 20 attributes. It was used to evaluate several existing works [12], [14]. The attributes of *age* and *gender* are protected attributes. The original aim of the dataset is to give an assessment of an individual's credit based on personal and financial records. It is used as a benchmark by SG [14].
- *Bank Marketing*[6]. The dataset came from a Portuguese banking institution and is used to train models for predicting whether the client would subscribe a term deposit based on his/her information. The size of the dataset is more than 45,000. There are a total of 16 attributes and the only protected attribute is *age*. It is used as a benchmark by SG [14].
- *Wikipedia Comments*. The overview of this dataset has been summarized in Example 4.2.
- *Jigsaw Comments*[7]. This is a public dataset from an online competition that was held by Jigsaw and Google. It is also a commonly used benchmark for fairness research [10]. It has around 313,000 comments with an average length of 80 words classified into six categories of toxicity (i.e., toxic, severe toxic, obscene, threat, insult, and identity hate) and non-toxic. The sensitive features are the same as those for the Wikipedia Comments.

For the tabular data, we use the binning method to preprocess the numerical attributes. For the text data, in order to balance accuracy and efficiency, we adopt the state-of-art embedding tool GloVe [40] and use the 50-dimension pretrained word vectors[8] trained on 6 billion tokens and 400 thousand vocabularies of Wikipedia 2014 and Gigaword 5. For those out-of-vocabulary words, we take the average of all the word vectors as an unknown vector suggested by the author of GloVe.

The details of the models used in the experiments are shown in Table 3. Since the experimented tabular datasets are relatively simple, we train models in the form of fully-connected DNN. We adopt two common improved variants of RNN, Long Short Term Memory (LSTM) [41] and Gated Recurrent Unit (GRU) [42], both with 3 layers and 10 hidden states, on the text datasets. After getting the final state of the LSTM and GRU models, we apply a fully connected layer to output the predicted labels. Fig. 6 shows the overview of the LSTM-based tested model.

The key KPIs for the comparison between our algorithm and baselines are the number of discriminatory samples, the success rate of generating discriminatory samples, and

4. https://github.com/sakshiudeshi/Aequitas
5. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit + data)
6. https://archive.ics.uci.edu/ml/datasets/bank+marketing
7. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
8. http://nlp.stanford.edu/projects/glove

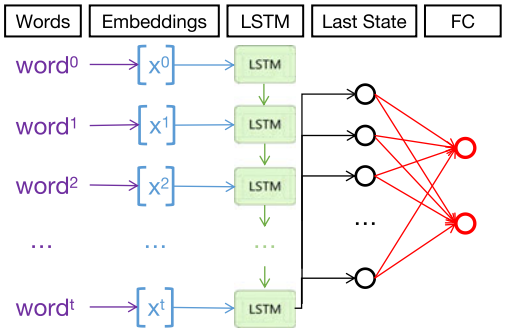


Fig. 6. Experimental LSTM-based model.

TABLE 4
Comparison With AEQUITAS

Dataset	Prot. Attr.	AEQUITAS		ADF	
		#GDiff	#ID	#GDiff	#ID
census	age	56955	6045	491650	256980
census	race	54734	4737	344162	139179
census	gender	32148	2930	259227	47644
bank	age	32870	8949	700285	364758
credit	age	99560	38479	398209	233664
credit	gender	33137	4996	312919	73497

#GDiff Denotes the Number of Unique Generated Samples, and #ID Denotes the Number of Identified discriminatory samples.

the generation efficiency. For text data, we additionally compare the quality of generated discriminatory text.

5.2 Research Questions

We aim to answer the following research questions through our experiments.

RQ1: How effective is our algorithm in finding discriminatory samples?

Tabular data. We first compare ADF with AEQUITAS and SG on tabular datasets. Since AEQUITAS and ADF both have a global generation phase and a local generation phase, we conduct a detailed comparison for both phases. For both of them, we generate 1,000 samples in the global generation phase (except for credit data, which is set to be 600 due to its small size), and then generate 1,000 samples during local generation for each successfully identified discriminatory sample in the global phase.

Table 4 presents the details of the comparison. Note that the maximum number of the two-stage searched samples is thus 1,001,000 (i.e., 1,000 global and 1,000*1,000 local samples). We further filter out the duplicate samples. Column #GDiff is the number of non-duplicate samples generated after the two-phase search. Column #ID shows the number of discriminatory samples identified. It can be observed that ADF is significantly more effective than AEQUITAS in finding discriminatory samples. *On average, ADF generates 8.6 times more non-duplicate samples.* One of the reasons why AEQUITAS explores a much smaller space is that it often generates duplicate samples (as was observed in [14]) since a global sampling distribution is used for all the inputs, while ADF perturbs a specific input according to the guidance of an sample-specific gradient. More importantly, *ADF generated nearly 25 times more discriminatory samples on average*. A close investigation shows that the reason is that gradient provides much better guidance in identifying discriminatory samples. This is evidenced by the average success rate which is calculated by #ID / #GDiff. *AEQUITAS has a success rate of 18.22 percent, whereas ADF achieves a success rate of 40.89 percent, which is more than 2 times that of AEQUITAS.*

TABLE 5
Comparison With SG in 500 Seconds

Dataset	Prot. Attr.	SG		ADF	
		#GDiff	#ID	#GDiff	#ID
census	age	1290	544	8202	3453
census	race	1541	632	10677	4290
census	gender	1482	280	22977	4164
bank	age	1385	842	5911	3587
credit	age	2752	1574	5771	3923
credit	gender	3202	926	14711	4091

#GDiff denotes the number of unique generated samples, and #ID denotes the number of identified discriminatory samples.

Although SG similarly has two phases, it works differently from ADF or AEQUITAS. That is, SG maintains a priority queue, pops a sample, and applies global search iteratively. If the sample is a discriminatory sample, the local search is employed. Afterward, all the search results are pushed into the queue without checking whether they are discriminatory or not. As a result, it is infeasible to directly compare SG and ADF as above. Thus, we apply an overall evaluation between ADF and SG within the same time limit, i.e., 500 seconds. The results are shown in Table 5. We observe that on average: *ADF 1) explores 6.6 times more samples, 2) generates 6.5 times more discriminatory samples, and 3) has a 42.8 percent success rate (whereas SG has a success rate of 41.5 percent).* One thing to notice is that our method beats SG which is based on a symbolic solver even in terms of success rate. One possible explanation is that the model explainer SG utilized is far from accurate for complex models like DL models.

In addition to the above overall evaluation with two baselines, we further conduct a comprehensive comparison phase by phase, i.e., global generation and local generation.

Global Generation. The goal of the global generation is to identify diversified discriminatory samples. For a fair comparison, we generate 1,000 samples in the global phase (except for *credit data*, which is set to be 600), and count how many discriminatory samples are identified by each method in this stage. Note that the same seed samples are used for SG and ADF.

The results are shown in Table 6. It can be observed that ADF generates the most number of discriminatory samples, *with an average improvement of 794 percent and 324 percent when it is compared with AEQUITAS and SG respectively.* We take that this shows the effectiveness of guiding the search based on gradient during global generation.

Local Generation. The local generation aims to further craft more discriminatory samples based on the results of the global generation. To make a fair comparison between the three strategies for local generation, we seed each method with the same set of discriminatory samples and apply the three strategies to generate 1,000 samples for each seed. In

TABLE 6
Number of discriminatory samples Generated by Global Generation

Dataset	Protected Attr.	AEQUITAS	SG	ADF
census	age	101	291	658
census	race	95	139	456
census	gender	37	54	334
bank	age	43	142	872
credit	age	175	247	594
credit	gender	47	87	451

TABLE 7
Number of discriminatory samples Generated by Local Generation

Dataset	Protected Attr.	AEQUITAS	SG	ADF
census	age	216	422	598
census	race	153	371	526
census	gender	189	210	321
bank	age	221	634	708
credit	age	448	600	750
credit	gender	142	280	337

this way, we are able to properly evaluate the local generation strategies without being influenced by the results of the global generation adopted by the three methods.

Table 7 shows the results of the comparison. It is obvious that the local generation strategy of our method ADF performs the best among the three. Specifically, *ADF generates 153 percent more discriminatory samples than AEQUITAS, and 32 percent more than SG on average.*

Recall that AEQUITAS and ADF both guide local generation through a probability distribution which intuitively is the likelihood of identifying discriminatory samples by changing certain attributes. The difference is that AEQUITAS's probability is global, i.e., the same probability is used for all samples, whereas ADF's probability is based on gradient and thus specific to the certain sample. We conduct a further experiment to evaluate whether ADF's approach is more effective or not. We feed these approaches the same set of seed samples and then measure the relationship between the number of seed samples explored and the number of new discriminatory samples identified.

The result is shown in Fig. 7 where the x-axis is the number of seeds explored; the blue line represents the total number of samples generated and the red line represents the number of discriminatory samples identified. It can be observed that for ADF, both lines grow steadily with the number of seeds explored, which suggests that the sample-specific probability used in ADF works reliably. In comparison, the increase of both the number of samples and the number of discriminatory samples drops with an increasing number of seeds for AEQUITAS. This is possibly due to the ineffective global probability and the duplication in the generated samples.

Text Data. We compare ADF with random perturbation (RP) on text datasets. Table 8 presents the results where the column *Protected Attr.* is the protected attribute. Column *Seed* is the number of seeds in the dataset that are explored using

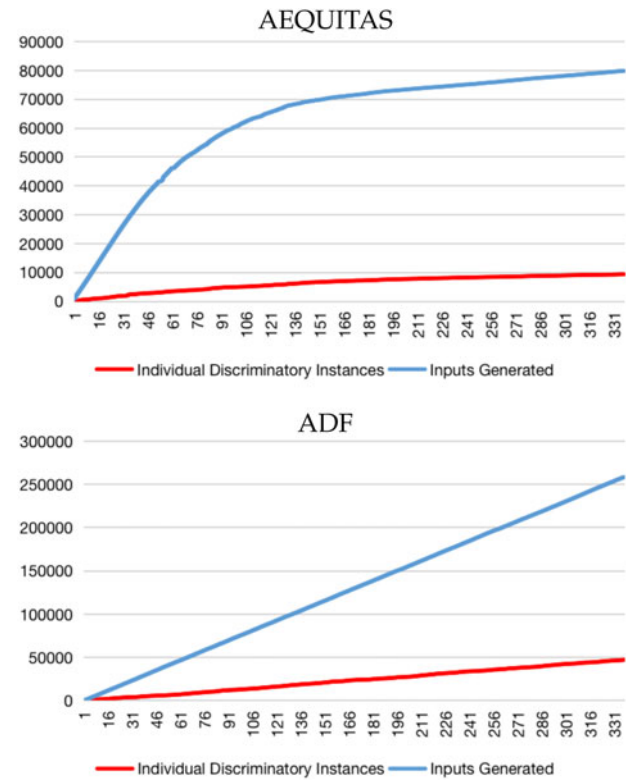


Fig. 7. Effectiveness of local generation.

our method. A seed is a text which contains at least one sensitive token. We limit the number of seeds to be maximum of 1,000 to reduce the overall experiment time. If there are fewer than 1,000 in the given set, the entire available set is used. Column *Raw* shows the number of discriminatory samples among the seeds (without any perturbation). Column *RP* is the number of discriminatory samples identified by the baseline approach where the random perturbation is applied to non-discriminatory text in the global generation, whereas *ADF_ASC* is the number of discriminatory samples identified through gradient-guided perturbation. The results show that the ADF is significantly more effective than the baseline, i.e., *generating 4.40 times more discriminatory samples on average*. In other words, gradient-guided perturbation applied to non-discriminatory text identifies much more discriminatory samples than random searching.

Besides, we also compare ADF with adversarial attacks in terms of effectiveness in generating discriminatory samples. As shown in Table 8, it is clear that the fairness testing method which considers two similar samples is much more effective in crafting discriminatory samples, *on average, it generates 22.11 percent (18.18 percent) more discriminatory text for ASC [19] (HotFlip [25])*. This is because our method ADF aims to maximize the output difference between two similar samples, which effectively avoids the situation that x and x' are too close so that they always cross the decision boundary at the same time.

Overall, it can be observed that the number of existing discriminatory samples or discriminatory samples generated through perturbation on non-discriminatory text is still quite limited, i.e., no more than 211 across all datasets, all models, and all sensitive features. This suggests that the local generation is indeed necessary to further generate

TABLE 8
Effectiveness

Dataset	Model	Protected Attr.	Global Generation							Local Generation	
			Seed	Raw	RP	ASC	ADF_ASC	HF	ADF_HF	RP	ADF
Wiki	LSTM	country	1000	31	29	119	131	308	401	47094 (80.6%)	145481 (92.1%)
		ethnic	1000	117	29	123	166	326	423	128174 (90.6%)	261746 (94.9%)
		race	450	24	17	76	95	157	177	32967 (83.0%)	107678 (92.8%)
		religion	358	29	14	50	64	117	150	37382 (88.7%)	86350 (94.6%)
	GRU	country	1000	33	22	83	100	303	405	43580 (81.5%)	117541 (90.9%)
		ethnic	1000	52	34	121	142	366	429	70093 (84.7%)	174488(92.4%)
		race	450	27	11	56	82	172	208	30651 (82.8%)	93944 (88.2%)
		religion	358	19	9	31	39	122	143	24748 (90.3%)	53866 (94.8%)
Jigsaw	LSTM	country	1000	22	26	106	123	224	254	32922 (72.9%)	123182 (88.7%)
		ethnic	1000	37	33	101	147	262	321	54741 (82.0%)	161807 (91.7%)
		race	1000	49	60	195	211	359	372	78513 (74.9%)	218982 (87.2%)
		religion	1000	51	52	163	204	325	358	83102 (86.2%)	227167 (93.3%)
	GRU	country	1000	22	34	84	95	232	250	39242 (74.5%)	98753 (88.9%)
		ethnic	1000	66	38	124	139	274	323	84765 (86.3%)	181891 (93.3%)
		race	1000	55	58	179	198	290	333	85106 (79.3%)	218404 (90.1%)
		religion	1000	58	55	147	169	361	400	88945 (84.8%)	200968 (93.5%)

discriminatory samples. For each discriminatory sample identified on the global phase,[9] we apply local generation to generate at most 1,000 perturbed texts to identify discriminatory samples. Column *RP* and *ADF* show the total number of discriminatory samples generated through random perturbation and our gradient-guided perturbation respectively. Note that we are also reporting the results of distinct texts in each entry. It is obvious that our gradient-guided perturbation generates significantly more discriminatory samples than random perturbation, i.e., *on average 2.68 times more*. Furthermore, the perturbation on a discriminatory sample by gradient-guided search has a higher success rate than the random approach. *On average, 91.7 percent of the distinct texts crafted by our gradient-guided perturbation are successfully identified as discriminatory samples while the success rate of random perturbation is 82.69 percent.*

We thus have the following answer to RQ1:

Answer to RQ1: ADF outperforms the baseline methods. For tabular data, compared to AEQUITAS, ADF searches 9.6 times input space, generates 25 times discriminatory samples, and has more than 2 times success rate. Compared to SG, ADF searches 6.6 times input space and generates 6.5 discriminatory samples, and has a slightly higher success rate given the same time limit. Gradient provides effective guidance during both global generation and local generation. For text data, ADF generates 2.68 times discriminatory samples and has a 9.01 percent more success rate than random perturbation on average.

RQ2: Are discriminatory samples generated by ADF valid?

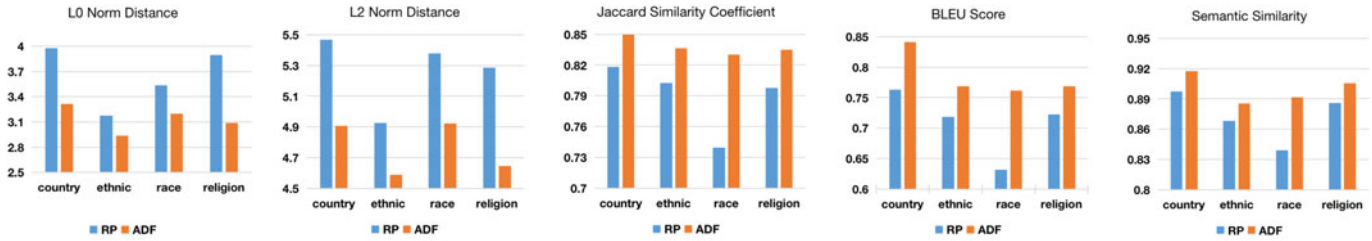
Recall that we aim to generate valid discriminatory samples, especially we need to preserve the syntax and semantics for the text. To answer this question, we measure how close the generated discriminatory samples are to the original ones using multiple metrics that are commonly applied in NLP research. Noted that these metrics also can be utilized to filter the similar text by giving a threshold [43].

9. For ADF, we utilize the discriminatory samples generated by ADF_ASC as local generation seeds.

- *L_p Norm Distance* is widely used to assess the dissimilarity of two samples. Specifically, L_0 distance counts the number of tokens modified and L_2 distance quantifies the euclidean distance in the word vector space. A smaller distance indicates higher similarity.
- *Jaccard Similarity Coefficient* is another popular statistical indicator to measure similarity. Given two sets containing all tokens in the two texts respectively, it is defined as the division of the size of the intersection over the size of the union. The larger is the coefficient, the better.
- *BLEU Score* [44] is widely used in neural machine translation to measure the differences between a candidate and reference sentences [45]. It first calculates the N-grams ($N = 1, 2, 3, 4$) model of the two texts respectively and then counts the number of matches. In our experiment, it is used to evaluate the quality of the generated IDS by regarding the original text as a reference. The score ranges from 0 to 1 and a higher score is preferred.
- *Semantic Similarity* converts words to vectors using an embedder and uses their cosine similarity for the evaluation. In our experiment, we utilized the model in [46] to convert the original text and generated IDS to two 512-dimensional vectors and then compute the cosine of their angles. The higher the score is, the better.

For a fair comparison on text datasets with the baseline, we take the same number of discriminatory samples identified on the global generation phase and generate discriminatory samples by local generation. Afterward, we compute and compare the above-mentioned metrics for both approaches. Fig. 8 shows the average metric values computed based on the LSTM model trained on the Wiki Comment dataset. Since the results of other models and other datasets are similar, they are omitted and the readers are referred to [22] for details.

First, it can be observed that a consistent good score is achieved by discriminatory samples generated by our approach across all five metrics. For instance, the average L_0 norm distance is 3.13, which intuitively means that only 3.13 tokens on averages are modified. Second, it is evident that our gradient-guided perturbation performs better than



(a) L_0 Norm Distance (b) L_2 Norm Distance (c) Jaccard Similarity Coefficient (d) BLEU Score (e) Semantic Similarity

Fig. 8. Validity analysis.

random perturbation on all metrics, i.e., the discriminatory samples generated by our approach are more similar with the seed text than those generated through random perturbation. *On average, discriminatory samples generated by our approach modify 13.55 percent less tokens (see Fig. 8a), have 9.39 percent shorter distance in word embedding space (see Fig. 8b), and are 7.25 percent, 3.17 percent, 11.04 percent more similar to the original text in terms of Jaccard Similarity Coefficient (see Fig. 8c), Semantic Similarity (see Fig. 8e) and BLEU Score (see Fig. 8d) respectively.*

In Table 9, we conduct an experiment to evaluate the quality of discriminatory samples generated by different global generation strategies. Similarly, only the result of the LSTM model trained on the Wiki Comment dataset (protected attribute is *race*) is shown here, and the remaining results are in [22]. The results show that for ASC (HotFlip), comparing with the adversarial attack version, *ADF generates discriminatory samples with 0.44 (0.02) fewer perturbed tokens, 0.26 (0.31) shorter euclidean distance, and 0.03 (0.01), 0.02 (0.01), 0.04 (0.02) more similar to the original text in terms of Jaccard Similarity Coefficient, Semantic Similarity and BLEU Score respectively.*

We thus have the following answer to RQ2:

> *Answer to RQ2: The discriminatory samples generated through ADF are similar to the original one.*

RQ3: How efficient is our algorithm in finding discriminatory samples?

Besides effectiveness, efficiency is also important. We thus conduct an experiment to compare the efficiency of ADF with baseline.

Table 10 presents how much time each method takes to generate 1,000 discriminatory samples on tabular datasets. Note that for all methods we measure the total time, especially for SG, it includes the time for generating the explanation model and constraint solving. It is evident that ADF has the best performance. *On average, it takes only 48.97 and 14.53 percent of the time required by AEQUITAS and SG respectively.* Combined with the results shown in Table 4, it implies that AEQUITAS and ADF have similar efficiency in generating samples (and ADF has a much higher success rate in finding discriminatory samples). Considering that AEQUITAS performs random sampling whereas ADF needs to calculate the gradient, it suggests that the overhead of calculating gradient in ADF is negligible. SG takes significantly more time to generate samples based on a seed sample. Its efficiency is thus much worse, as expected.

For the text datasets, we generate 5,000 discriminatory samples by both the baseline approach and our approach (ADF_ASC) and record the time taken in Table 11. Overall, ADF takes an average of 1234.2 seconds across all datasets and models, which we believe are reasonably efficient, compared to the cost of data collection and model training for such tasks. Comparing our approach with the baseline, *RP takes on average 2.9 percent less time than our approach.* The additional time overhead is mainly due to the extra step of calculating the gradients on both global and local generation stages. Due to the structure of RNN, the time complexity of back-propagation by chain-rule is $\mathbf{O}(N)$, where N is the length of the text. Further experiment shows that this step is the most time consuming, e.g., in the case of the LTSM model trained on the Wiki Comments, each gradient computation takes 0.454 (0.184) seconds, while the remaining steps only take 0.060 (0.041) seconds for perturbation on global (local) phase. Besides, we also compare the efficiency of ADF_ASC and ADF_HF, they take an average of 0.83 and 3.95 seconds each iteration respectively. This is because HotFlip traverses all possibilities of perturbed and substituting tokens in each iteration, which makes its time complexity is

TABLE 9
Validity Analysis w.r.t. Different Global Generation Strategies

Metric	ASC	ADF_ASC	HF	ADF_HF
L_0	2.6296	2.1851	2.6364	2.6136
L_2	5.5460	5.2825	5.8291	5.5203
JSC	0.8514	0.8770	0.8624	0.8724
BLEU	0.8186	0.8577	0.8482	0.8634
SS	0.8914	0.9135	0.8862	0.8967

TABLE 10
Time (seconds) Taken to Generate 1,000 discriminatory samples on Tabular Datasets

Dataset	Prot. Attr.	AEQUITAS	SG	ADF
census	age	172.64	720.49	59.15
census	race	128.75	506.33	65.95
census	gender	158.37	2128.42	78.68
bank	age	191.16	521.79	106.93
credit	age	176.31	321.63	64.92
credit	gender	156.22	476.52	102.90

TABLE 11
Time (seconds) Taken to Generate 5,000 discriminatory samples on Text Datasets

Dataset	Model	Prot. Attr.	RP	ADF_ASC
Wiki	LSTM	country	2894.8	2371.5
		ethnic	563.9	837.2
		race	659.0	1587.4
		religion	542.0	746.8
	GRU	country	2037.8	746.8
		ethnic	1307.4	1142.6
		race	451.7	638.3
		religion	510.9	551.1
Jigsaw	LSTM	country	2069.5	1848.1
		ethnic	738.3	1136.9
		race	1296.3	2677.5
		religion	444.2	685.6
	GRU	country	1572.1	910.6
		ethnic	1265.5	2267.6
		race	472.6	583.1
		religion	919.8	438.5

$\mathbf{O}(m * k)$, where k is the number of similar tokens for selecting. However, the time complexity of ASC is $\mathbf{O}(k)$, since it greedily chooses the most important token and then only traverses the top-k similar words.

We thus have the following answer to RQ3:

Answer to RQ3: ADF is reasonably efficient in generating discriminatory samples. There is a slight time overhead in applying gradient-guided perturbation, which is more noticeable for RNN.

RQ4: How useful are the identified discriminatory samples for improving the fairness of the DNN?

To further show the usefulness of our generated discriminatory samples, we evaluate whether we can improve the fairness of the DL model by retraining it with data augmented with the generated discriminatory samples. We remark that AEQUITAS also uses retraining to improve the fairness of the original models and SG does not have such discussions. We need a systematic way of evaluating the fairness of a given model. For this, we adopt the method proposed and used by AEQUITAS [13]. The idea is to randomly sample a large set of samples and evaluate the model fairness by the percentage of discriminatory samples in the set. However, as it is nearly impossible to obtain meaningful text by random sampling, we evaluate the improvement by an independently generated set of discriminatory samples, which is a common practice [47], [48]. That is, we apply our approach to generate an independent set of IDSs based on the testing set and evaluate how many of them are IDSs with respect to the retrained model. Note that since we randomly select 5 percent of generated discriminatory samples for data augment and retraining, we repeated the procedure 5 times and present the average improvement.

The results on tabular datasets are shown in Table 12, where columns *Before* and *After* are the estimated fairness of the model before and after retraining using the generated discriminatory samples. The smaller the number is, the more fair the model is. It can be observed that retraining with the discriminatory samples can significantly improve the fairness, and *ADF achieves better fairness improvement (with more identified discriminatory samples), i.e., 57.2 percent on average, versus existing approaches, i.e., 45.1 percent for AEQUITAS and 49.1 percent for SG.*

TABLE 12
Fairness Improvement on Tabular Datasets

Dataset	Prot. Attr.	Before (%)	After (%)		
			AEQUITAS	SG	ADF
census	age	10.88	4.03	2.41	2.26
census	race	9.75	7.05	6.89	6.15
census	gender	3.14	2.33	1.90	1.65
bank	age	4.60	1.68	2.04	1.19
credit	age	27.93	13.91	13.19	12.05
credit	gender	7.68	4.58	4.66	3.93

TABLE 13
Fairness Improvement on Text Datasets

Dataset	Model	Prot. Attr.	Augmented (%)	Certified (%)
Wiki	LSTM	country	72.4	54.9
		ethnic	70.2	64.1
		race	64.2	61.8
		religion	54.2	63.4
	GRU	country	48.6	66.3
		ethnic	66.2	71.0
		race	58.1	65.8
		religion	51.0	59.8
Jigsaw	LSTM	country	71.6	51.5
		ethnic	70.4	55.6
		race	58.5	60.4
		religion	63.8	55.5
	GRU	country	64.1	57.9
		ethnic	56.6	61.4
		race	53.5	66.3
		religion	40.4	59.8

Table 13 presents the results on text datasets. Note that the augmented data is crafted by ADF_ASC based on the training dataset, which has no overlap with the testing dataset. The number of independently generated discriminatory samples is 155,690 on average for each row. Column *Augmented* shows the percentage of those independently generated discriminatory samples that are no longer discriminatory samples given the retrained model. That is, the bigger the number, the less discriminative the retrained model. As observed from Table 13, *retraining reduces the number of discriminatory samples by an average of 60.2 percent and up to 72.4 percent*. Since individual fairness can be regarded as a specific form of robustness according to its definition, we also compare the effectiveness of augmented training with certified training which acquires the tractable upper bound for the worst-case perturbation and then provides a certificate of robustness [49], [50], [51]. Since [50], [51] propagate the upper bound with interval arithmetic proposed by [52], which is only sufficient for FNNs and CNNs, here we adopt [49] to present a comprehensive analysis. Column *Certified* listed the fairness improvement for the certified trained model. *On average, the performances of augmented training and certified training achieve similar improvement on fairness, 60.2 versus 61.0 percent.* Although certified training is a one-time effort for all the potential sensitive tokens, fairness testing not only improves the fairness of the model but also quantifies the bias of the original model. In addition, we conduct a supplementary experiment to show the fairness improvement with regard to different sample generating

TABLE 14
Fairness Improvement w.r.t. Different Methods

Dataset	Model	Prot. Attr.	ASC	HF	ADF_HF
Wiki	LSTM	race	63.4	66.3	67.5
Jigsaw	GRU	ethnic	57.1	61.7	62.0

The improvement of ADF_ASC is showed in Table 13.

methods in Table 14, and the results of ADF_ASC are presented in Table 13. Noted that the testing data is all generated using method ADF_ASC. It reveals that the greater the number and diversity of discriminatory samples, the more helpful to improve the fairness of the model.

We thus have the following answer to RQ4:

Answer to RQ4: The discriminatory samples generated by ADF are useful to improve the fairness of the DL models through retraining, with an average improvement of 57.2 percent on tabular datasets and 60.2 percent on text datasets.

5.3 Threats to Validity

Limited Datasets. In the experiment, We evaluated ADF with 5 datasets, including 2 text datasets for toxic classification tasks. Compared with other text classification tasks, toxic classifiers are more prone to discrimination issues, e.g., racism, sexism, and most of the relevant data can be easily obtained from social media. Although they are the most common public benchmarks used in the fairness testing literature, we cannot conclude the effectiveness and efficiency of other datasets. As our approach is independent of the datasets and has been made available online, it can be used on other datasets with minor adjustments for data adaptation.

Limited Models. We only tested three deep learning models in the experiment. Especially for the tabular data, since they are relatively simple (i.e., with a maximum of 20 features), we only used the basic fully connected DNN. However, the key idea of ADF is generic which can be easily implemented for more complex deep learning models like convolutional neural networks (CNNs), as our approach only requires a way of computing the gradients.

White-Box Setting. ADF is designed as a white-box approach, i.e., the fairness testing part requires the full knowledge of the target deep learning models. It is widely accepted that deep learning model testing could have the full knowledge of the target model. In the future, we will explore how to extend it to a black-box setting, e.g., selecting the token for perturbation not based on gradients but certain importance scores (i.e., the confidence of prediction).

Step-Size Parameters. The step-size parameters of ADF for tabular data depend on the training dataset. For datasets with only categorized attributes, it is easy to set it to be 1. For other datasets, further research may be necessary to identify an effective step size. If the step size is too big, it may miss some discriminatory samples during its perturbation, especially for the local generation. If the step size is too small, it is hard to generate discriminatory samples in the global generation.

Complex Context. Text data is much more complex than tabular data and it is certain that our approach does not cover all kinds of discrimination. First, one word could have different meanings according to their contexts, for example, "black" and "white" may refer to either color or race. We thus are unable to filter them out without manual labeling. Second, there may be many forms of specific discrimination. For instance, sexism may not only be associated with tokens such as "male/female", but also "he/she", and "actor/actress". It is thus nearly impossible to list them all.

5.4 Discussion

In this work, we aim to acquire a large number of diverse discriminatory samples, since the diversity and quantity of discriminatory samples would help us to 1) figure out whether there is discrimination in different subspaces of the model, 2) mitigate the discrimination in multiple parts of the model at one time through retraining. Clustering the original data in ADF can be regarded as a coarse-grained equivalence class partition method since perturbation only slightly shifts the distribution. However, this is not enough, we hope that there will be more fine-grained methods for dividing equivalence classes, which could help us better understand the causes of discrimination and improve model fairness.

Another point worth discussing is the tolerance of fairness. Due to the sensitivity of fairness in society, fairness is strictly required in many cases. For example, many USA states prohibit the use of face recognition in public places recently, and discrimination is one of the most important reasons. Although it is challenging to achieve complete fairness for DL models, our goal is to improve the fairness of DL models, rather than giving up DL models. First, DL does bring convenience to our daily life. Second, manual decision-making is not completely fair either. The real problem is that discrimination in historical data may be introduced or even magnified in the training process of the model. Thus, we need to locate the data on which discrimination occurs and figure out how to reduce the discrimination learned by the model as much as possible, and so that its fairness can reach the same level as or even higher than that of manual decision-making.

6 Related Work

Fairness. Many works exist on the fairness issues of AI in general. In [53], Chen *et al.* attributed the unfairness of the trained model to the data collection, then decomposed the discrimination of data into bias, variance, and noise, and last proposed some strategies to estimate and reduce them. In [54], Albarghouthi *et al.* proposed fairness-aware programming, which monitors whether programs violate custom fairness at runtime by concentration inequalities [55]. In [8], Bastani *et al.* first formalized three fairness specifications demographic parity, equal opportunity, and path-specific causal fairness, and then utilized adaptive concentration inequalities to obtain a probabilistic guarantee of the model with respect to the given specification. In [11], Thomas *et al.* proposed a general and flexible framework by encoding fairness constraints (upper bound)

into the objective function. When the amount of data is enough, Hoeffding's inequality [56] is used to obtain the bounds, otherwise, Student's t test is used. In addition to these general methods, some works are focused on the fairness of text. In [57], [58], the impacts of using NLP models and potential bias caused in real-world applications are discussed. [59] points out that the bias may exist in word embeddings. It also proposes a mechanism on gender-neutral words to quantify the degree of gender bias by projection and reduces the bias by removing the gender associations. [36] shows that the discrimination is correlated with comment length and the distribution differences between the sensitive values in the toxic data and the whole dataset. In [10], Garg *et al.* improve the model's fairness through a robust training method called CLP which introduces the prediction difference of the original input and its counterpart (which only differs in certain sensitive features) as penalties into the training loss.

Fairness Testing. This work is closely related to fairness testing of machine learning models. Galhotra *et al.* proposed THEMIS [12], [60] which first defines software fairness testing, then introduces fairness scores as measurement metrics and lastly designs a causality-based algorithm utilizing the random test input generation technique to evaluate the model fairness, i.e., the frequency of discriminatory samples' occurrence of software. However, THEMIS is inefficient in general since it relies on random sampling without guidance on the generation. Udeshi *et al.* proposed AEQUITAS [13] which inherits and improves THEMIS and focuses on the discriminatory sample generation. AEQUITAS is a systematic generating algorithm. It first explores the input domain randomly to discover discriminatory samples in the global search phase. During the local generation, AEQUITAS searches the neighbors of discriminatory samples identified in the global phase, by perturbing them. For the local generation, AEQUITAS designs three different strategies, i.e., random, semi-directed, and fully-directed, to update the probability which is used to guide the selection of attributes to perturb. Based on their evaluation, fully-directed has the best effectiveness and efficiency. Besides searching the discriminatory samples, AEQUITAS also design an automated iterative retraining method to obtain a more fair model. Later, Agarwal *et al.* proposed Symbolic Generation (SG) [14] which integrates symbolic execution and local model explanation techniques to craft discriminatory samples. SG relies on the local explanation of a given input which constructs a decision tree utilizing the samples generated randomly by the Local Interpretable Model-agnostic Explanation (LIME) [15]. The path of the tree determines all the important attributes leading to the prediction. The algorithm also contains a global generation phase and a local generation phase. A detailed comparison between ADF and the above approaches is presented in Section 4.3.

Gradient-Based Attacks. This work is also related to research on gradient-based adversarial attacks. A variety of works have been proposed to explore the vulnerability of the DL model by crafting adversarial samples, and gradient-based adversarial attack is one kind of the most effective methods. Goodfellow *et al.* proposed the first attacking algorithm Fast Gradient Sign Method (FGSM) [17] to generate adversarial samples by perturbing the original input with the linearization of the loss function used in the training process. FGSM is fast by only attacking once according to the gradient. Later, several other attack methods are proposed to extend FGSM. For instance, instead of attacking only once, Basic Iterative Method (BIM) [18] employs perturbations based on gradients multiple times (often with smaller step sizes) and applies a function that performs per-attribute clipping to make sure the sample after each iteration is located in the neighborhood of the original sample. In [19], Papernot *et al.* first formalize the adversarial sequence optimization problem in the context of sequential data and develops ASC to obtain malicious texts by iteratively replace a token with the one such that the sign of the difference between them is closest to the sign of gradient. However, ASC may cause syntactic or semantic errors since it searches in the whole corpus. In [30], a general attack framework called TEXTBUGGER is proposed to generate adversarial texts. It works as follows: 1) find the important words by a) computing the Jacobian matrix under white-box setting and b) scoring with the confidence drop before and after removal of a word under black-box setting; 2) generate adversarial texts by inserting a space, deleting the middle letter, swapping two adjacent characters, replacing a letter with a similar one, and taking the top$-k$ nearest words in embedding space as a substitute. In [25], HotFlip is proposed to generate adversarial examples with character/word substitution, which uses beam search to apply perturbation with the highest loss estimated by gradient with respect to the one-hot input. Besides, Pei *et al.* [21] designed an algorithm for maximizing the coverage of neurons as well as model outputs of multiple DNNs, and solve the optimization function using the gradient.

7 Conclusion

In this work, we propose a lightweight algorithm ADF to efficiently generate discriminatory samples for deep learning models through adversarial sampling. Our algorithm combines a global phase and a local phase to systematically search the input space for discriminatory samples with the guidance of gradient. In the global generation, ADF first locates the discriminatory samples near the decision boundary by iteratively perturbing towards the decision boundary. In the local generation, ADF again samples according to the gradient to search the neighborhood of a found discriminatory sample. The experiment on 22 benchmarks of 5 datasets shows that ADF is able to effectively generate diverse valid discriminatory samples within a reasonable time. As for future work, we would like to propose additional criteria to measure the generated discriminatory samples, and then cluster them into different equivalence classes to improve the effectiveness of augmented retraining or fine-tuning. Besides, we also want to uncover the root cause of discrimination in the model, e.g., mapping it to the training data.

Acknowledgments

This work was supported by the Key-Area Research and Development Program of Guangdong Province Grant

2020B0101100005. This research was also supported by the Key Research and Development Program of Zhejiang Province 2021C01014, the Fundamental Research Funds for the Zhejiang University NGICS Platform, the Guangdong Science and Technology Department under Grant 2018B010107004, and Ministry of Education, Singapore Project MOET32020-0004, T2EP20120-0019, and T1-251RES1901.

REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.

[2] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit card fraud detection using convolutional neural networks," in *Proc. 23rd Int. Conf. Neural Inf.*, 2016, pp. 483–490.

[3] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1391–1399.

[4] S. Barocas, M. Hardt, and A. Narayanan, "Fairness and machine learning," 2019. [Online]. Available: http://www.fairmlbook.org

[5] H.-L. E. G. on Artificial Intelligence (AI HLEG), "Draft ethics guidelines for trustworthy AI," European Commission, Brussels, Belgium, 2018.

[6] F. Tramèr *et al.*, "Fairtest: Discovering unwarranted associations in data-driven applications," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2017, pp. 401–416.

[7] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 259–268.

[8] O. Bastani, X. Zhang, and A. Solar-Lezama , "Probabilistic verification of fairness properties via concentration," in *Proc. ACM Program. Languages*, 2019, pp. 118:1–118:27.

[9] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel, "Fairness through awareness," in *Proc. Innovations Theor. Comput. Sci.*, 2012, pp. 214–226.

[10] S. Garg, V. Perot, N. Limtiaco, A. Taly, E. H. Chi, and A. Beutel, "Counterfactual fairness in text classification through robustness," in *Pro. AAAI/ACM Conf. AI Ethics Soc.*, 2019, pp. 219–226.

[11] P. S. Thomas, B. C. da Silva, A. G. Barto, S. Giguere, Y. Brun, and E. Brunskill, "Preventing undesirable behavior of intelligent machines," *Science*, vol. 366, no. 6468, pp. 999–1004, 2019.

[12] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: Testing software for discrimination," in *Proc. 11th Joint Meeting Foundations Softw. Eng.*, 2017, pp. 498–510.

[13] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 98–108.

[14] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Foundations Softw. Eng.*, 2019, pp. 625–635.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.

[16] X. Wang, J. Sun, Z. Chen, P. Zhang, J. Wang, and Y. Lin, "Towards optimal concolic testing," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 291–302.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[18] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[19] N. Papernot, P. D. McDaniel, A. Swami, and R. E. Harang, "Crafting adversarial input aequences for recurrent neural networks," in *Proc. IEEE Military Commun. Conf.*, 2016, pp. 49–54.

[20] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. Eur. Symp. Secur. Privacy*, 2016, pp. 372–387.

[21] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 1–18.

[22] P. Zhang, J. Wang, J. Sun, X. Wang, and T. Dai, 2020. [Online]. Available: https://github.com/pxzhang94/ADF

[23] P. Zhang *et al.*, "White-box fairness testing through adversarial sampling," in *Proc. 42nd Int. Conf. Softw. Eng.*, 2020, pp. 949–960.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[25] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 31–36.

[26] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "LEMNA: Explaining deep learning based security applications," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 364–379.

[27] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 241–257.

[28] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[29] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[30] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," in *Proc. 26th Annu. Netw. Distrib. Syst. Secur. Symp*, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/textbugger-generating-adversarial-text-against-real-world-applications

[31] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2016, pp. 372–387.

[32] A. Beutel, J. Chen, Z. Zhao, and E. H. Chi, "Data decisions and theoretical implications when adversarially learning fair representations," *CoRR*, 2017.

[33] G. Goh, A. Cotter, M. R. Gupta, and M. P. Friedlander, "Satisfying real-world goals with dataset constraints," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2415–2423. [Online]. Available: http://papers.nips.cc/paper/6316-satisfying-real-world-goals-with-dataset-constraints

[34] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proc. 26th Int. Conf. World Wide Web)*, 2017, pp. 1391–1399.

[35] M. Wiegand, J. Ruppenhofer, and T. Kleinbauer, "Detection of abusive language: the problem of biased datasets," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Language Technol.*, 2019, pp. 602–608.

[36] L. Dixon, J. Li, J. Sorensen, N. Thain, and L. Vasserman, "Measuring and mitigating unintended bias in text classification," in *Proc. AAAI/ACM Conf. AI Ethics Soc.*, 2018, pp. 67–73.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J. Roy. Statist. Soc. Ser. Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.

[38] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

[39] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. Workshop New Challenges NLP Frameworks*, 2010, pp. 45–50. [Online]. Available: http://is.muni.cz/publication/884893/en

[40] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14–1162/

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] K. Cho B. *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1724–1734.
[43] S. Gupta, P. He, C. Meister, and Z. Su, "Machine translation testing via pathological invariance," in *Proc. 28th ACM Joint Eur. Softw. Eng. Conf. Symp. Foundations Softw. Eng. Virtual Event*, 2020, pp. 863–875.
[44] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evalue of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2002, pp. 311–318.
[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Systems*, 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks
[46] D. Cer , "Universal sentence encoder for english," in *Proc. Conf. Empir. Methods Natural Lang. Process. Syst. Demonstrations*, 2018, pp. 169–174.
[47] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 303–314.
[48] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proc. 41st Int. Conf. Softw. Eng.*, 2019, pp. 1039–1049.
[49] R. Jia, A. Raghunathan, K. Göksel, and P. Liang, "Certified robustness to adversarial word substitutions," in *Proc. Conf. Empir. Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 4127–4140.
[50] P. Huang *et al.*, "Achieving verified robustness to symbol substitutions via interval bound propagation," in *Proc. Conf. Empir. Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 4081–4091.
[51] Y. Zhang, A. Albarghouthi, and L. D'Antoni, "Robustness to programmable string transformations via augmented abstract training," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 11 023–11 032.
[52] S. Gowal *et al.*, "On the effectiveness of interval bound propagation for training verifiably robust models," *CoRR*, 2018
[53] I. Y. Chen, F. D. Johansson, and D. A. Sontag, "Why is my classifier discriminatory?," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3543–3554.
[54] A. Albarghouthi and S. Vinitsky, "Fairness-aware programming," in *Proc. Conf. Fairness Accountability Transparency*, 2019, pp. 211–219.
[55] P. Massart, "Concentration inequalities and model selection," 2007.
[56] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Publications Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.
[57] D. Hovy and S. L. Spruit, "The social impact of natural language processing," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 591–598.
[58] R. Tatman, "Gender and dialect bias in youtube's automatic captions," in *Proc. 1st ACL Workshop Ethics Natural Lang. Process.*, 2017, pp. 53–59.
[59] T. Bolukbasi, K. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4349–4357. [Online]. Available: http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings
[60] R. Angell, B. Johnson, Y. Brun, and A. Meliou, "Themis: Automatically testing software for discrimination," in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Sympo. Foundations Softw. Eng.*, 2018, pp. 871–875.

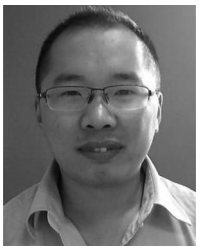
Peixin Zhang received the bachelor's degree in software engineering from Zhejiang University, in 2016. He is currently working toward the PhD degree at the College of Computer Science and Technology, Zhejiang University, China. He was a visiting student with the Singapore University of Technology and Design in 2017, and Singapore Management University during 2019–2020. His research interests include software engineering and artificial intelligence, especially software engineering for artificial intelligence and software testing.

Jingyi Wang received the bachelor's degree in information engineering from Xi'an Jiaotong University, in 2013, and the PhD degree from the Singapore University of Technology and Design, in 2018. He is currently a tenure-track assistant professor with the College of Control Science and Engineering, Zhejiang University, China. He was a research fellow with the School of Computing, National University of Singapore during 2019–2020 and at Information Systems Technology and Design Pillar, Singapore University of Technology and Design during 2018–2019. His research interests include formal methods, software engineering, cyber-security and machine learning.

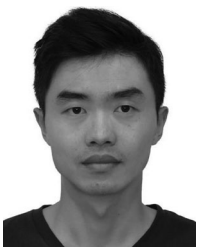
Jun Sun received the bachelor's and PhD degrees in computing science from the National University of Singapore (NUS), in 2002 and 2006, respectively. He is currently a tenured associate professor with the School of Information Systems, Singapore Management University. From 2010 to 2019, he was an assistant/associate professor with the Singapore University of Technology and Design. He was a visiting scholar with MIT from 2011 to 2012. His research interests include software engineering, formal methods, program analysis, and cyber-security. He is the co-founder of the PAT model checker.

Xinyu Wang received the bachelor's and PhD degrees in computer science from Zhejiang University, in 2002 and 2007, respectively. He is currently a professor with the College of Computer Science and Technology, Zhejiang University, China. His research interests include real-time intelligent data processing, artificial intelligence, and software engineering.

Guoliang Dong received the bachelor's degree in software engineering from Northwestern Polytechnical University, China, in 2016. He is currently working toward the PhD degree at the College of Computer Science and Technology, Zhejiang University. He was a visiting student with Singapore Management University, during 2019-2020. His research interests include software engineering and artificial intelligence, especially software engineering for artificial intelligence.



Xingen Wang received the bachelor's and PhD degrees in computer science from Zhejiang University, in 2005 and 2013, respectively. He is currently a research assistant with the College of Computer Science and Technology, Zhejiang University, China. His research interests include distributed computing and software performance.



Ting Dai received the bachelor's degree in computer science from Tsinghua University, in 2009, and the PhD degree in computer science from the National University of Singapore, in 2015. He is a senior researcher with the Trustworthy AI Lab of Shield Lab, Huawei Singapore Research Center. His research interests include system and software security, and security in emerging platforms.

Jin Song Dong received the bachelor's and PhD degrees in computing from the University of Queensland, Australia, in 1992 and 1996, respectively. From 1995 to 1998, he was a research scientist with CSIRO, in Australia. He is currently a full professor with the School of Computing, National University of Singapore. He is on the editorial board of ACM Transactions on Software Engineering and Methodology, Formal Aspects of Computing and Innovations in Systems and Software Engineering. His research interests include formal methods, software engineering, pervasive computing, and semantic technologies.