White-box Fairness Testing through Adversarial Sampling

Peixin Zhang
Zhejiang University
pxzhang94@zju.edu.cn

Jingyi Wang*
National University of Singapore
wangjy@comp.nus.edu.sg

Jun Sun
Singapore Management University
junsun@smu.edu.sg

Guoliang Dong
Zhejiang University
dgl-prc@zju.edu.cn

Xinyu Wang
Zhejiang University
wangxinyu@zju.edu.cn

Xingen Wang
Zhejiang University
newroot@zju.edu.cn

Jin Song Dong
National University of Singapore
dcsdjs@nus.edu.sg

Ting Dai
Huawei International Pte. Ltd.
daiting2@huawei.com

ABSTRACT

Although deep neural networks (DNNs) have demonstrated astonishing performance in many applications, there are still concerns on their dependability. One desirable property of DNN for applications with societal impact is fairness (i.e., non-discrimination). In this work, we propose a scalable approach for searching individual discriminatory instances of DNN. Compared with state-of-the-art methods, our approach only employs lightweight procedures like gradient computation and clustering, which makes it significantly more scalable than existing methods. Experimental results show that our approach explores the search space more effectively (9 times) and generates much more individual discriminatory instances (25 times) using much less time (half to 1/7).

ACM Reference Format:
Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box Fairness Testing through Adversarial Sampling. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep neural networks (DNNs) are gradually adopted in a wide range of applications, including fraud detection [9], facial recognition [22], self-driving [5], and medical diagnosis [27]. Although DNNs have demonstrated astonishing performance in many applications, there are still concerns on their dependability. One desirable property of DNN for applications with societal impact is fairness (i.e., non-discrimination) [18]. Since there are often societal bias in the training data, the resultant DNNs might introduce discrimination unintentionally. This has been demonstrated in [25]. Discrimination in DNNs is often more 'hidden' than that of traditional decision-making software since it is still an open problem on how to interpret DNNs. Therefore, it is crucial to have systematical methods for automatically identifying potential discrimination in a given DNN.

Various forms of discrimination exist in the machine learning literature, including but not limited to group discrimination [8] and individual discrimination [7]. Discrimination is often defined over a set of protected attributes[1], such as age, race, gender and etc. Intuitively, discrimination happens when a machine learning model tends to make different decisions for different *individuals* (individual discrimination) or *subgroups* (group discrimination) differentiated only by one/multiple protected attributes. Note that the set of protected attributes is often application-dependent and given in advance.

In this work, we focus on the problem of developing a systematic and scalable approach for generating individual discriminatory instances for DNNs. Note that it is often insufficient to identify one instance demonstrating the existence of individual discrimination in a given DNN. It is desirable to generate as many as possible such instances so that the DNN can be retrained with the generated instances to reduce discrimination. In the literature, there have been multiple relevant attempts on the problem [2, 10, 26]. In [10], Galhotra *et al.* proposed THEMIS to measure the occurrence frequency of discrimination by randomly sampling each attribute within its domain and identifying those biased instances. In [26], Udeshi *et al.* developed AEQUITAS which consists of a global search and a local search. That is, AEQUITAS first searches the input space by random sampling (a.k.a. global search), and then applies local search based on results of the global search, by perturbing the identified individual discriminatory instances with selected attributes along random directions to identify as many as possible instances evidencing discrimination. In [2], Agarwal *et al.* proposed a method called Symbolic Generation, which first generates a local explanation decision tree using an existing method [21] to approximate the model decision and then performs symbolic execution based on the decision tree to generate test cases. Like AEQUITAS, it also combines a global search which aims to maximize path coverage based on the decision tree with a local search which aims to maximize the number of discriminatory instances.

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 DNN, experiment results show that their performances on DNN are far worse than that on traditional models, and are far from being effective. Furthermore, there are additional shortcomings for each approach, as we discuss in Section 3.3.

In this work, we propose a scalable gradient-based algorithm called *A*dversarial *D*iscrimination *F*inder (ADF) for generating individual discriminatory instances, which is specifically designed

*Corresponding authors: Jingyi Wang and Xinyu Wang.
[1]We use 'protected'/'sensitive' and 'attribute'/'feature' interchangeably.

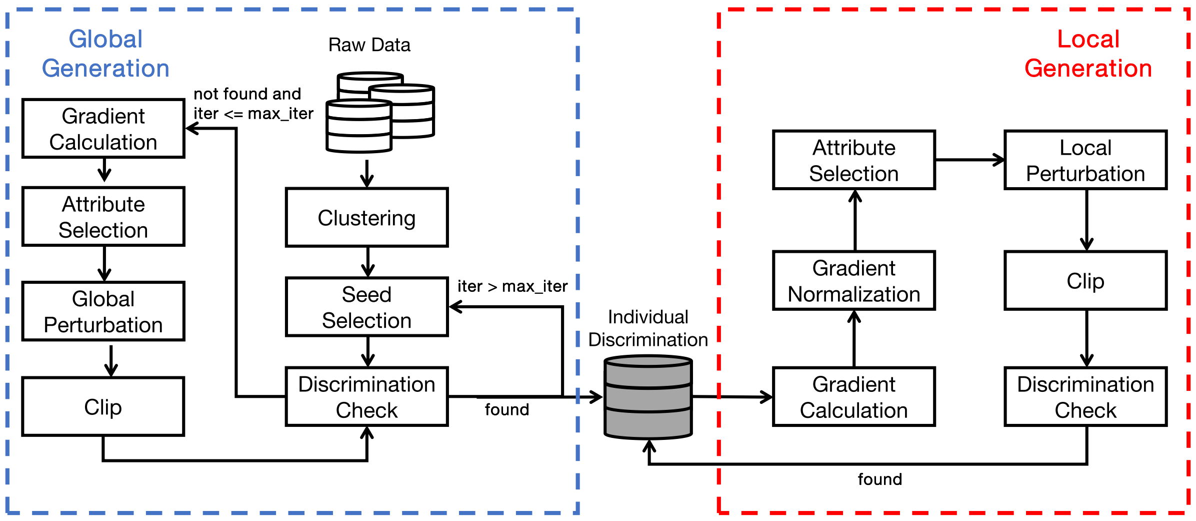


Figure 1: An overview of ADF .

for DNN. Gradient is an effective tool to craft test inputs for DNN. It can be computed efficiently for large DNN and it offers intuitive guidance on how a model prediction changes with respect to certain attributes. It is useful in many DNN related tasks. It is noticeably used in recent works to generate adversarial samples [12, 13, 19, 20], i.e., instances which are only slightly difference from some existing instances in the training set yet result in very different model prediction. Inspired by these works, we use gradient as an effective way for searching individual discriminatory instances in DNN.

An overview of ADF is presented in Figure 1. ADF has two parts, i.e., global generation (i.e., the left part) and local generation (i.e., the right part). During global generation, the samples in the original dataset are clustered and seed instances from each cluster are selected in a round-robin fashion. The goal of the global generation is to increase diversity in the generated individual discriminatory instance. Gradients are used in the global generation to guide the crafting of individual discriminatory instances, by maximizing the difference between the DNN outputs of two similar instances. The global generation stops if a certain number of individual discriminatory instances have been successfully generated or it times out. The individual discriminatory instances identified are then taken as inputs for local generation. The idea is to search neighbors of the individual discriminatory instances for more discriminatory instances. Gradients are used in local generation in a different way as guidance, i.e., we utilize the gradient's absolute values which represent the importance of each attribute to identify individual discriminatory instances which are minimally different from the seeds while maintaining their model predictions. Note that ADF is generative, i.e., it may generate samples which are not in the original dataset. In order to make sure the generated instances are valid, a Clip function is used in both global generation and local generation.

ADF has been implemented as in a self-contained toolkit. Our experiments on multiple real-word benchmarks show that ADF explores 8 times more input space and generates 24 times more individual discriminatory instances on average than AEQUITAS. Comparing with Symbolic Generation, ADF has an average of 324% and 32% higher success rate in global generation and local generation. Furthermore, ADF is around 2 and 7 times more efficient than AEQUITAS and Symbolic Generation. Note that ADF only relies on lightweight procedures like clustering and gradients, which makes it much more effective and scalable than existing methods.

In summary, We make the following main contributions.

- We present an efficient and effective approach ADF for generating individual discriminatory instances of DNN based on gradient.
- We implement and publish ADF as a self-contained toolkit[2] on-line.
- We evaluate ADF with 6 benchmarks on 3 datasets. Our experiment shows ADF is significantly more effective and efficient in generating individual discriminatory instances than state-of-the-art methods.

The remainder of the paper is organized as follows. Section 2 presents the necessary background on individual discrimination and DNN. In Section 3, we present ADF in detail. In Section 4, we discuss our experimental setup and our results. We review related works in Section 5 and conclude in Section 6.

2 BACKGROUND

In this section, we briefly review relevant background, including Deep Neural Network (DNN), individual discrimination, gradient-based adversarial attack, and then define our problem.

DNN. A DNN $\mathcal{D}$ often contains an input layer, multiple hidden layers and an output layer. We denote these layers as NL =

[2] https://github.com/pxzhang94/ADF

$\{NL_j | j \in \{0, \ldots, J\}\}$ and assume the j-th layer has s_j neurons. For each neuron, it first calculates the weighted sum of the outputs of all the neurons in its previous layer to get the output $v_{j,k}$ (as shown in Equation 1) and then applies an activation function (e.g., Sigmoid, hyperbolic tangent (tanh), or rectified linear unit (relu) [17]) ϕ.

$$v_{j,k} = \phi(\sum_{l=1}^{s_{j-1}} \omega_{j-1,k,l} \cdot v_{j-1,l}) \tag{1}$$

In the following, we use $\theta = \{\omega_{j,k} | 1 \leq j \leq J, 0 \leq k \leq s_j\}$ to denote the set of parameters of $\mathcal{D}$. In this work, we focus on DNN classifiers $\mathcal{D} : X \rightarrow Y$, i.e., for a given instance $x \in X$, a DNN outputs a predicted label $y \in Y$ which has the highest probability.

Individual discrimination. We denote X as a dataset and its set of attributes by $A = \{A_1, A_2, \ldots, A_n\}$. Assume each attribute A_i has a valuation domain $\mathbb{I}_i$, the input domain is then $\mathbb{I} = \mathbb{I}_1 \times \mathbb{I}_2 \times \cdots \times \mathbb{I}_n$, which denote all the possible combinations of attribute valuations. 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. A DNN model $\mathcal{D}$ trained on X may contain discrimination. In this work, we focus on individual discrimination.

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 instance in $\mathbb{I}$. We say that x is an individual discriminatory instance of a model $\mathcal{D}$ if there exists another data instance $x' \in \mathbb{I}$ 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 an individual discriminatory instance pair.

Gradient-based adversarial attack. Deep neural networks are shown to be vulnerable to adversarial samples [24]. In recent years, many adversarial attack methods have been proposed to craft adversarial samples, which deliberately perturb the original normal input subtly and yet able to fool the DNN model. In the following, we briefly introduce gradient-based adversarial attacks which inspire our work. The intuition is to perturb the original input in the gradient direction so that the DNN will change its output to the largest extent.

FGSM 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 DNN's loss function with respect to the input features according to the following Equation 2:

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

where J is the loss function of the DNN $\mathcal{D}$, $y = \mathcal{D}(x)$ is the predicted class of x, $\nabla_x J(x, y)$ is the gradient of J on x with respect to the label y and ϵ is a hyper-parameter to determine perturbation degree. FGSM is highly effective and efficient to obtain adversarial samples in practice.

Later, several other gradient-based attack methods are proposed to extend FGSM. For instance, instead of attacking only once, Basic Iterative Method (BIM) [13] employs perturbations based on gradient multiple times with smaller step size. Another method is Jacobian-based Saliency Map Attack (JSMA) [19] which only selects two most important features to perturb according to the saliency map. We omit the details and remark that they share the same spirit of utilizing gradient information of a DNN.

Problem definition. A model which suffers from individual discrimination may produce biased decision when an individual discriminatory instance is presented as input. Our problem is thus defined as follows. Given a dataset X (with a set of attributes A and a set of protected attributes P) and a DNN model $\mathcal{D}$, how can we effectively and efficiently generate individual discriminatory instances for $\mathcal{D}$ so that we can retrain a DNN model based on X and the generated individual discriminatory instances for better fairness? This problem is challenging because we focus on complex DNNs which renders existing methods ineffective.

3 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 approaches.

ADF generates individual discriminatory instances in two phases, i.e., a global generation phase and a local generation phase. In the global generation phase, we aim to identify those individual discriminatory instances near the decision boundary from the original dataset X, which serve as the seed data for the local generation phase. In the local generation phase, we follow the intuition that instances nearby those seed data are likely to be individual discriminatory instances to find more of them. Note that this intuition is inspired by recent research on the robustness of DNNs [24]. In the following, we introduce the two phases in details.

Example 3.1. We use the Census Income dataset[3] as a running example 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 [2, 4, 10, 11, 26]. The task is to predict whether the income of an adult is above \$50,000 based on their personal information. The dataset contains 32561 training instances with 13 attributes each. The following shows a sample instance 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 an individual discriminatory instance for the model.

3.1 Global generation

Algorithm 1 shows the details of the global generation phase. The algorithm uses the following constants: c_num which is the size of clusters; g_num which is the number of seed instances to generate

[3]https://archive.ics.uci.edu/ml/datasets/adult

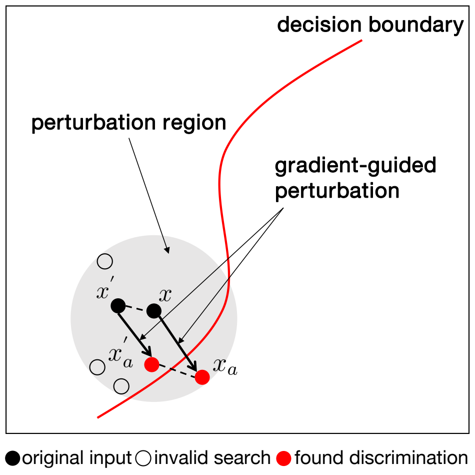

decision boundary
perturbation region
gradient-guided perturbation
x'
x
x'_a
x_a
● original input ○ invalid search ● found discrimination

Figure 2: Intuition of gradient-based approach.

during global generation; *max_iter* which is the number of maximum iteration number; and *s_g* which is the step size of global generation.

First, we cluster the original dataset using a standard popular clustering algorithm K-Means [15] at line 2. Afterwards, we obtain seed instances from each cluster in a round-robin fashion (see line 4). The goal of clustering is to improve the diversity of the seeds.

In the loop from line 5 to 22, we generate individual discriminatory instances iteratively based on the gradient. Let θ be the parameters of a DNN $\mathcal{D}$; y be the ground-truth label associated with x; and $J(\theta, x, y)$ be the loss function used to train the model $\mathcal{D}$. Given a seed x selected from a cluster (see line 4), we first check whether it is an individual discriminatory instance according to Definition 2.1 at line 6. 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. For the running example 3.1, $N = 2$. If x is not an individual discriminatory instance, we start to search for an individual discriminatory instance based on x with the guidance of the gradient defined as $\nabla_x J(\theta, x, y)$.

Notice that in order to identify an individual discriminatory instance, we need to find an individual discriminatory instance pair, i.e., a pair of instances which 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 with x in protected attributes. Since x is not an individual discriminatory instance, x and x' thus has the same label. As shown in Figure 2, we then utilize gradient information on x and x' to offer guidance on how to perturb (x, x') such that we are most likely to identify an individual discriminatory instance pair.

We identify a set of instances X from $\mathbb{I}$ such that x and any instance x' in X only differs 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 instances in X, we choose x' according to the following equation:

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

Algorithm 1 Global Generation

```
1:  g_id = ∅
2:  clusters = KMeans(data, c_num)
3:  **for** i from 0 to g_num **do**
4:      Get seed $x$ from clusters in a round-robin fashion
5:      **for** iter from 0 to max_iter **do**
6:          **if** $x$ is an individual discriminatory input **then**
7:              g_id = g_id ∪ x
8:              **break**
9:          **end if**
10:         $X = \{x' | \forall x'_p \in \mathbb{I}_p, x'_p \neq x_p\}$
11:         $x' = \arg\max\{abs(\mathcal{G}_y(x') - \mathcal{G}_y(x)) | x' \in X\}$
12:         $grad = \nabla J(x)$
13:         $grad' = \nabla J(x')$
14:         Initialize array dir with the same size as $A$ by 0
15:         **for** $a \in A \backslash P$ **do**
16:             **if** $sign(grad_a) = sign(grad'_a)$ **then**
17:                 $dir_a = sign(grad_a)$
18:             **end if**
19:         **end for**
20:         $x = x + dir * s\_g$
21:         $x = Clip(x)$
22:     **end for**
23: **end for**
24: **return** g_id
```

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

Our next step is to perturb x and x' to generate an individual discriminatory instance pair (x_a, x'_a) such that $\mathcal{D}(x_a) \neq \mathcal{D}(x'_a)$. 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 our running example, since the attributes are all preprocessed as categorized values, the perturbation is done by increasing or decreasing its value by 1 unit (i.e., the minimal perturbation). A *perturbation* in our context is thus a function of a set of non-protected attributes which we choose to perturb and a corresponding boolean vector where 1 means increasing the attribute value by 1 and 0 means decreasing by 1. Formally,

Definition 3.2. ***Perturbation*** A perturbation δ on a data instance x is function $\delta : \mathbb{I} \times NP \times \mathbb{B} \longrightarrow \mathbb{I}$, where $\mathbb{B}$ is the direction of the perturbation.

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

$$\text{argmax}_{\delta(x,x')}\{\mathcal{D}(x_a) - \mathcal{D}(x'_a)\}, \quad (4)$$

where $x_a = \delta(x)$ and $x'_a = \delta(x')$. Unfortunately, this objective can not be directly optimized. Our remedy is to adopt the idea of EM algorithm [6] in machine learning to iteratively optimize it. Here, we utilize the gradient of $\nabla J(x) - \nabla J(x')$ and select those attributes which have similar contributions (with the same sign of

Algorithm 2 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 $x_i$
```

gradients) as attributes to perturb. The intuition is that perturbing these attributes can potentially enlarge the output difference since $\nabla J(x) - \nabla J(x')$ equals 0 on them (likely to be local minimum). Once we find the perturbation, we apply it on x and x' and then check whether the new pair (x_a, x'_a) is an individual discriminatory instance pair. If the answer is yes, the algorithm will break out immediately (see line 6-8). Otherwise, we start another round of perturbation on (x_a, x'_a). This process may repeat multiple times until it succeeds. Notice that in order to filter out unreal test inputs, we always apply a Clip function described at Algorithm 2 to make sure that each attribute value after perturbation is within its domain (see line 21).

Example 3.3. For our running example, we cluster the raw training data into 4 clusters and select seed instances from each cluster in a round-robin fashion. The first selected seed x is as follows (shown in Example 3.1), and it is not an individual discriminatory instance.

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

We identify all instances 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 instances' 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 the following instance.

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

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 an individual discriminatory instance.

3.2 Local generation

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

Our local generation algorithm has the following parameters: *l_num* which is the number of trials in local generation; and *s_l* is the step size of local generation. The algorithm makes use of the gradients of loss (see lines 6-7) in a different way. Recall that in global generation, we would like to change the DNN output maximally. On the contrary, in local generation, we would like to change the output as minimally as possible, as our goal is to maintain the DNN outputs of the individual discriminatory instance pairs identified in the global generation so that they remain different. We thus choose to perturb those attributes which have least effect on the output. Note that the absolute value of gradient represents how much an attribute contributes to the output.

Further note that since we are perturbing an individual discriminatory instance pair, we need to consider the two inputs at the same time (to make sure that neither of them will cross the decision boundary as otherwise they are no longer an individual discriminatory instance 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 4. We first add the absolute value of two gradients together to get the saliency value of each attribute (see line 3). Then we calculate the reciprocal value (see line 4) since we aim to select the attributes with less 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 3 shows the details of our local generation algorithm. Given an individual discriminatory instance pair (x, x') (see line 5), we start searching by iteratively selecting the attributes to perturb using the normalization of gradients (see line 8). Instead of modifying all the attributes with the same sign, we randomly select the perturbation direction (sign) with a uniform probability [0.5, 0.5]. Similar to global generation, we also utilize the Clip function (see Algorithm 2) to make sure that the generated test case is real (see line 12). We check whether the input after perturbation is an individual discriminatory instance (see line 13) and continue to the next seed input if the answer is yes. Otherwise, we start another iteration of local generation (see line 3).

Example 3.4. For our running example, the global generation phase generates the following individual discriminatory instance 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 instances 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

Algorithm 3 Local Generation

```
1:  l_id = ∅
2:  for $x \in g\_id$ do
3:      for i from 0 to l_num do
4:          $X = \{x^{'} | \forall x^{'}_p \in \mathbb{I}_p, x^{'}_p \neq x_p\}$
5:          $\exists x^{'} \in X, \mathcal{D}(x) \neq \mathcal{D}(x^{'})$
6:          $grad = \nabla J(x)$
7:          $grad^{'} = \nabla J(x^{'})$
8:          $p = Normalization(grad, grad^{'})$
9:          Select $a \in A \backslash P$ with probability $p_a$
10:         Select $d \in [1, -1]$ with probability $[0.5, 0.5]$
11:         $x_a = x_a + d \times s\_l$
12:         $x = Clip(x)$
13:         if $x$ is a individual discriminatory input then
14:             l_id = l_id ∪ x
15:         end if
16:     end for
17: end for
18: return l_id
```

Algorithm 4 Normalization(gradient1, gradient2)

```
1:  Initialize gradient with the same size of gradient1
2:  for i from 0 to gradient.length do
3:      $saliency = |gradient1_i| + |gradient2_i|$
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: $probability = \{g/gradient\_sum | \forall g \in gradient\}$
11: return probability
```

the result instance's values are all within the respective domains, the Clip function keeps it the same and the following instance is identified as a new individual discriminatory instance.

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

3.3 Qualitative Evaluation

In the following, we evaluate our approach qualitatively by comparing it with state-of-the-art approaches, i.e., THEMIS [10], AEQUITAS [26] and Symbolic Generation (SG) [2]. Empirical comparison results are presented in Section 4.

THEMIS [10] explores the input domains for all attributes through random sampling and then checks whether the generated instances are individual discriminatory instances. AEQUITAS [26] improves THEMIS by adopting a two-phase generation framework. In the first phase, AEQUITAS randomly searches for a set of individual discriminatory instances in the input space as seeds. In the second phase, AEQUITAS searches for more individual discriminatory instances 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 which describes the probability of finding an individual discriminatory instance 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 [2]. The most recent work SG [2] 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 [21] 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.

Table 1: Comparing different approaches.

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

To summarize the difference between existing approaches and ours, we differentiate them using three criteria, i.e., whether the search (for individual discriminatory instances) is guided, whether the guided search is specific for an individual input (input-specific), and whether the procedure adopted is light-weight (and thus likely scalable). Table 1 shows the summary. Except THEMIS, both AEQUITAS and SG generate individual discriminatory instances 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 which 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.

Compared to existing approaches, our approach satisfies all the three criteria. First, our search is guided by gradient, i.e., the perturbation is guided towards the decision boundary to accelerate the discovery of individual discriminatory instances which significantly reduces the number of attempts needed. The intuition is visualized in Figure 2. 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 DNN 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 [1], PyTorch and Keras.

Similar to AEQUITAS , our approach also has a global search phase and a local search phase. The differences are in the details of

Table 2: Configuration of experiments.

Parameter	Value	Description
c_num	4	cluster count
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

Table 3: Experimented DNN 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%

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 instances. 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 4. To solve the problem, our approach takes an input-specific perspective, i.e., choosing different local perturbation based on gradient which are specific to a given instance.

4 EXPERIMENT

We have implemented ADF as a self-contained toolkit based on Tensorflow [1] and scikit-learn. Its source code, together with all the experiment related details, are available online. In the following, we evaluate ADF to answer multiple research questions (RQ).

4.1 Experimental Setup

We choose AEQUITAS and SG for baseline comparison. Note that THEMIS is shown to be significantly less effective [2] and thus is omitted for comparison. We obtained the implementation of AEQUITAS from GitHub[4] and re-implemented SG according to the description in [2] since their implementation is not publicly available. 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 [26]. We conducted our experiments 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 3 open-source datasets for fairness testing as our experiment subjects. The details of the datasets as follows.

- *Census Income* The details of this dataset have been introduced in Example 3.1. It is used as a benchmark by AEQUITAS and SG [2].
- *German Credit*[5] This is a small dataset with 600 data and 20 attributes. It was used to evaluate several existing works [2, 10]. The attributes *age* and *gender* are protected attributes. The original aim of dataset is to give an assessment of individual's credit based on personal and financial records. It is used as a benchmark by SG [2].
- *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 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 [2].

We use the binning method to pre-process the numerical attributes. The details of the models used in the experiments are shown in Table 3. Since these datasets are relatively simple, we train models in the form of fully-connected DNNs and perform clustering based on the standard K-Means algorithm.

Table 4: Comparison with AEQUITAS.

Dataset	Protected 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

4.2 Research Questions

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

RQ1: How effective is our algorithm in finding individual discriminatory instance?

We first compare ADF with AEQUITAS. 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 1000 instances in the global generation phase (except for credit data, which is set to be 600 due to its small size), and then generate 1000 instances during local generation for each successfully identified individual discriminatory instance in the global phase.

The details of the comparison are presented in Table 4. Note that the maximum number of the two-phase searched instances is thus 1001000 (1000 global and 1000000 local instances). We further filter out duplicate instances. Column #GDiff is the number of non-duplicate instances generated after the two-phase search. Column #ID shows the number of individual discriminatory instances identified. It can be observed that ADF is significantly more effective than AEQUITAS in finding individual discriminatory instances. *On average, ADF generates 8.6 times more non-duplicate instances.* One of the reasons why AEQUITAS explores a much smaller space is that it often generates duplicate instances (as was observed in [2]) since a global sampling distribution is used for all the inputs, while ADF perturbs a specific input according to the

[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

Table 5: Comparison with SG in 500 seconds.

Dataset	Protected 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

Table 6: Number of individual discriminatory instances 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

guidance of an instance-specific gradient. More importantly, *ADF generated nearly 25 times more individual discriminatory instances on average.* A close investigation shows that the reason is that gradient provides a much better guidance in identifying individual discriminatory instances. This is clearly evidenced by the average success rate which is calculated by #ID / #GDiff. *AEQUITAS has a success rate of 18.22%, whereas ADF achieves a success rate of 40.89%, which is more than 2 times of that of AEQUITAS.*

Although SG similarly has two phases, it works differently from ADF or AEQUITAS. That is, SG maintains a priority queue, pops an instance and apply global search iteratively. If the instance is an individual discriminatory instance, local search is employed. Afterwards, 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 with the same time limit, i.e., 500 s. The results are shown in Table 5. We observe that on average: *ADF 1) explores 6.6 times more instances, 2) generates 6.5 times more individual discriminatory instances and 3) has 42.8% success rate (whereas SG has a success rate of 41.5%).* 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 DNN.

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 global generation is to identify diversified individual discriminatory instances. For a fair comparison, we generates 1000 instances in global generation (except for *credit data*, which is set to be 600), and count how many individual discriminatory instances are identified by each method in this stage. Note that the same seed instances are used for SG and ADF.

Table 7: Number of individual discriminatory instances 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

The results are shown in Table 6. It can be observed that ADF generates the most number of individual discriminatory instances, *with an average improvement of 794% and 324% 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.

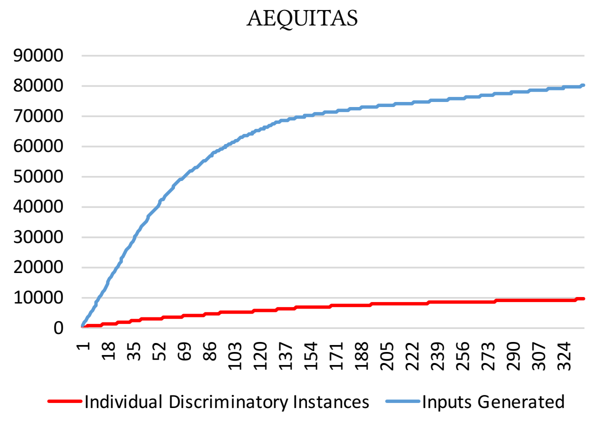
Local Generation Local generation aims to further craft more individual discriminatory instances based on the results of global generation. To make a fair comparison between the three strategies for local generation, we seed each method with the same set of individual discriminatory instances and apply the three strategies to generate 1000 instances for each seed. 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.

The results are shown in Table 7. It can be observed that the local generation strategy of our method ADF performs the best among the three. Specifically, *ADF generates 153% more individual discriminatory instances than AEQUITAS, and 32% 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 individual discriminatory instances by changing certain attributes. The difference is that AEQUITAS's probability is global, i.e., the same probability is used for all instances, whereas ADF's probability is based on gradient and thus specific to certain instance. We conduct a further experiment to evaluate whether ADF's approach is more effective or not. We feed these approach the same set of seed instances and then measure the relationship between the number of seed instances explored and the number of new individual discriminatory instances identified.

The result is shown in Figure 3 where the x-axis is the number of seeds explored; the blue line represents the number of instances generated and the red line represents the number of individual discriminatory instances identified. It can be observed that for ADF, both lines grow steadily with the number of seeds explored, which suggests that the instance-specific probability used in ADF works reliably. In comparison, the increase of both the number of instances and the number of individual discriminatory instances drops with an increasing number of seeds for AEQUITAS. This is possibly due to the ineffective global probability and the duplication in the generated instances.

We thus have the following answer to RQ1:



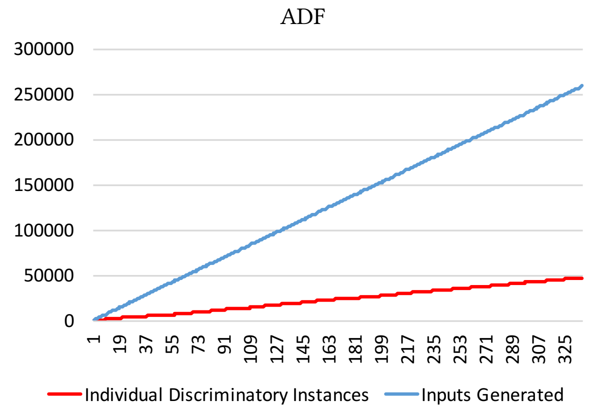


Figure 3: Effectiveness of local generation.

Table 8: Time (s) taken to generate 1000 individual discriminatory instances.

Dataset	Protected 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

Answer to RQ1: ADF outperforms the state-of-the-art methods AEQUITAS and SG. Compared to AEQUITAS, ADF searches 9.6 times input space, generates 25 times individual discriminatory instances and has more than 2 times success rate. Compared to SG, ADF searches 6.6 times input space and generates 6.5 individual discriminatory instances and has a slightly higher success rate given same time limit. Gradient provides effective guidance during both global generation and local generation.

RQ2: How efficient is our algorithm in finding individual discriminatory instances?

Table 9: Number of iteration for global generation in ADF.

Dataset	Protected Attr.	ADF
census	age	4.415 (1.530)
census	race	6.427 (2.268)
census	gender	7.759 (3.451)
bank	age	2.859 (1.811)
credit	age	1.479 (1.310)
credit	gender	5.295 (3.741)

Besides effectiveness, efficiency is also important. We thus conduct an experiment to compare the efficiency of these three approaches. Table 8 presents how much time each method takes to generate 1000 individual discriminatory instances. Note that for all methods we measure the total time. 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% of the time required by AEQUITAS and SG respectively.* Combined with the results shown in Figure 4, it implies that AEQUITAS and ADF have similar efficiency in generating instances (and ADF has much higher success rate in finding individual discriminatory instances). Considering that AEQUITAS performs random sampling whereas ADF requires to calculate the gradient, it suggests that the overhead of calculating gradient in ADF is negligible. SG takes significantly more time to generate instances based on a seed instance. Its efficiency is thus much worse, as expected. To further understand the time cost in the global generation phase of ADF, we utilize 1000 seed instances (600 for *credit data*) to calculate the average number of iterations needed to identify an individual discriminatory instance and present the results in Table 9. The numbers A(B) in the table means that, on average it takes A iterations for all the instances (with the maximum iteration set to 10) and B iterations for successfully identified individual discriminatory instances. We could observe that for successfully identified individual discriminatory instances, it often requires few iterations (around 2 on average), which suggests that the main cost is on those unsuccessful search (10 iterations). The implication is that we could further improve the efficiency of ADF by choosing an appropriate *max_iter* for different applications.

We thus have the following answer to RQ2:

Answer to RQ2: ADF is more efficient than AEQUITAS and SG in generating individual discriminatory instances, with an average speedup of 104% and 588%.

RQ3: How useful are the identified individual discriminatory instances for improving the fairness of the DNN?
To further show the usefulness of our generated individual discriminatory instances, we evaluate whether we can improve the fairness of the DNN model by retraining it with data augmented with the generated individual discriminatory instances. We remark that AEQUITAS also uses retraining to improve the fairness of the original models and SG does not have such discussions. Further note that we label the generated individual discriminatory instances using

Table 10: Fairness improvement.

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

the idea of majority voting [14, 26] from decisions of multiple models (which can be obtained by model mutation [16, 28] or training multiple models). Besides, we need a systematic way of evaluating the fairness of a given model. For this, we adopt the method proposed and used by AEQUITAS [26]. The idea is to randomly sample a large set of instances and evaluate the model fairness by the percentage of individual discriminatory instances in the set.

The results are shown in Table 10, where column *Before* and *After* are the estimated fairness of the model before and after retraining using the generated discriminatory instances. The smaller the number is, the more fair the model is. Since we randomly select 5% of generated individual discriminatory instances for data augmentation and retraining, we repeated the procedure 5 times and present the average improvement to avoid the effect of randomness. It can be observed that retraining with the individual discriminatory instances can significantly improve the model fairness, and *ADF achieves better fairness improvement (with more identified individual discriminatory instances), i.e., 57.2% on average, versus existing approaches, i.e., 45.1% for AEQUITAS and 49.1% for SG.*

We thus have the following answer to RQ3:

Answer to RQ3: The individual discriminatory instances generated by ADF are useful to improve the fairness of the DNN through retraining, i.e., with an average improvement of 57.2%.

4.3 Threats to Validity

Limited datasets We evaluated ADF with only 3 datasets. Although they are the most common public benchmarks used in the fairness testing literature, we cannot conclude the effectiveness and efficiency on other datasets. It is however easy to extend our evaluation if additional datasets are available in the future since ADF is dataset-independent and has been made available online.

Limited model structures We only used the basic fully-connected deep neural networks in the experiments since the data is relatively simple (i.e., with a maximum of 20 features). However, the key idea of ADF is generic which can be easily implemented for more complex neural networks like convolutional neural networks (CNNs), as it is shown that gradient works well for generating adversarial samples of CNN [12, 13, 19].

Access to model ADF is a white-box algorithm which generates individual discriminatory instances based on gradient with regard to the loss function used for training, which means it needs access to the model. It is widely accepted that DNN testing could have the full knowledge of the target model.

Step-size parameters The step-size parameters of ADF depend on the training dataset. For datasets with only categorized attributes (like the ones we tested in our experiments), 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 individual discriminatory instance during its perturbation, especially for local generation. If the step-size is too small, it is hard to generate individual discriminatory instance in global generation.

5 RELATED WORK

Fairness testing. This work is closely related to fairness testing of machine learning models. Galhotra *et al.* proposed THEMIS [3, 10] which firstly 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 individual discriminatory instances' 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 [26] which inherits and improves THEMIS, and focuses on the individual discriminatory instance generation. AEQUITAS is a systematic generating algorithm. It first explores the input domain randomly to discover individual discriminatory instances in the global search phase. During the local generation, AEQUITAS searches the neighbors of individual discriminatory instances identified in the global phase, by perturbing them. For 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 individual discriminatory instances, AEQUITAS also design an automated iterative retraining method to obtain a more fair model. Later, Agarwal *et al.* proposed Symbolic Generation (SG) [2] which integrates symbolic execution and local model explanation techniques to craft individual discriminatory instances. 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) [21]. 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 are presented in Section 3.3.

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 DNN by crafting adversarial samples. Gradient-based adversarial attack is one kind of most effective method. Goodfellow *et al.* proposed the first attacking algorithm Fast Gradient Sign Method (FGSM) [12] to generate adversarial samples by perturbing the original input with the linearization of the loss function used in 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) [13] employs perturbations based on gradients multiple times (often with smaller step sizes), and applies a function which performs per-attribute clipping to make sure the sample after each iteration is located in the neighborhood of the original sample. Papernot *et al.* introduced Jacobian-based Saliency Map Attack (JSMA) [19], an iterative targeted attack method, which attempts to force the DNN model to output the attacker-desired label with minimal perturbation by utilizing the backward derivative (gradient). It works by 1) first collecting the saliency map [23] based on the Jacobian matrix, and 2) selecting a pair of features which would cause the most significant change on the desired class, followed by 3) increasing the value of the selected two features to the maximum, lastly 4) repeat the above three steps until either the number of features perturbed exceeds the bound or achieving a successful adversarial sample. Besides, Pei *et al.* [20] designed an algorithm for maximizing the coverage of neurons as well as model outputs of multiple DNNs, and solve the optimization function using gradient.

6 CONCLUSION

In this paper, we propose a lightweight algorithm ADF to efficiently generate individual discriminatory instances for deep neural network through adversarial sampling. Our algorithm combines a global phase and a local phase to systematically search the input space for individual discriminatory instances with the guidance of gradient. In the global generation, ADF first locates the individual discriminatory instances 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 individual discriminatory instance. We compare ADF with two state-of-the-art fairness testing methods in 6 benchmarks, the results show that ADF has significantly better performance both in terms of effectiveness and efficiency.

ACKNOWLEDGMENTS

This research was supported by NSFC Program (Grant No. 61972339). This research was also supported by Huawei (Grant No. MG19C08), the Guangdong Science and Technology Department (Grant No. 2018B010107004), the National Research Foundation, Prime Minister's Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd and under its National Cybersecurity R&D Program (Award No. NRF2014NCR-NCR001-30).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A System for Large-scale Machine Learning. In *12th Symposium on Operating Systems Design and Implementation*. 265–283.

[2] Aniya Agarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2018. Automated Test Generation to Detect Individual Discrimination in AI Models. *CoRR* (2018). http://arxiv.org/abs/1809.03260

[3] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: automatically testing software for discrimination. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE 2018), Lake Buena Vista, FL, USA*. 871–875. https://doi.org/10.1145/3236024.3264590

[4] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H. Chi. 2017. Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. *CoRR* (2017). http://arxiv.org/abs/1707.00075

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* (2016). http://arxiv.org/abs/1604.07316

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977), 1–38. https://www.jstor.org/stable/2984875

[7] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through Awareness. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA*. 214–226. https://doi.org/10.1145/2090236.2090255

[8] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia*. 259–268. https://doi.org/10.1145/2783258.2783311

[9] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. 2016. Credit Card Fraud Detection Using Convolutional Neural Networks. In *Neural Information Processing - 23rd International Conference (ICONIP 2016), Kyoto, Japan*. 483–490. https://doi.org/10.1007/978-3-319-46675-0_53

[10] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017), Paderborn, Germany*. 498–510. https://doi.org/10.1145/3106237.3106277

[11] Gabriel Goh, Andrew Cotter, Maya R. Gupta, and Michael P. Friedlander. 2016. Satisfying Real-world Goals with Dataset Constraints. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain*. 2415–2423. http://papers.nips.cc/paper/6316-satisfying-real-world-goals-with-dataset-constraints

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations*.

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Examples in the Physical World. In *5th International Conference on Learning Representations (ICLR 2017), Toulon, France*. https://openreview.net/forum?id=HJGU3Rodl

[14] Louisa Lam and Ching Y. Suen. 1997. Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *IEEE Trans. Systems, Man, and Cybernetics, Part A* 27, 5 (1997), 553–568. https://doi.org/10.1109/3468.618255

[15] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Information Theory* 28, 2 (1982), 129–136. https://doi.org/10.1109/TIT.1982.1056489

[16] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. DeepMutation: Mutation Testing of Deep Learning Systems. In *29th International Symposium on Software Reliability Engineering*. 100–111.

[17] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel*. 807–814. https://icml.cc/Conferences/2010/papers/432.pdf

[18] High-Level Expert Group on Artificial Intelligence (AI HLEG). 2018. *Draft Ethics Guidelines for Trustworthy AI.* Technical Report. European Commission.

[19] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *European Symposium on Security and Privacy*. 372–387.

[20] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 1–18.

[21] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA*. 1135–1144. https://doi.org/10.1145/2939672.2939778

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015), Boston, MA, USA*. 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[23] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *2nd International Conference on Learning Representations (ICLR 2014), Banff, AB, Canada*. http://arxiv.org/abs/1312.6034

[24] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing Properties of Neural Networks. In *2nd International Conference on Learning Representations (ICLR 2014), Banff, AB, Canada*. http://arxiv.org/abs/1312.6199

[25] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel J. Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering Unwarranted Associations in Data-Driven Applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P 2017), Paris, France*. 401–416. https://doi.org/10.1109/EuroSP.2017.29

[26] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated Directed Fairness Testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018), Montpellier, France*. 98–108. https://doi.org/10.1145/3238147.3238165

[27] Sandra Vieira, Walter H.L. Pinaya, and Andrea Mechelli. 2017. Using Deep Learning to Investigate the Neuroimaging Correlates of Psychiatric and Neurological Disorders: Methods and Applications. *Neuroscience & Biobehavioral Reviews* 74 (2017), 58–75. https://doi.org/10.1016/j.neubiorev.2017.01.002

[28] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. In *Proceedings of the 41st International Conference on Software Engineering (ICSE 2019), Montreal, QC, Canada*. 1245–1256. https://dl.acm.org/citation.cfm?id=3339661