Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems | School of Computing and Information Systems

12-2022

QuoTe: Quality-oriented Testing for deep learning systems

Jialuo CHEN

Jingyi WANG

Xingjun MA

Youcheng SUN

Jun SUN
Singapore Management University, junsun@smu.edu.sg

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the Software Engineering Commons

Citation

CHEN, Jialuo; WANG, Jingyi; MA, Xingjun; SUN, Youcheng; SUN, Jun; ZHANG, Peixin; and CHENG, Peng. QuoTe: Quality-oriented Testing for deep learning systems. (2022). *ACM Transactions on Software Engineering and Methodology*. 32, (5), 1-33.
Available at: https://ink.library.smu.edu.sg/sis_research/7785

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Author

Jialuo CHEN, Jingyi WANG, Xingjun MA, Youcheng SUN, Jun SUN, Peixin ZHANG, and Peng CHENG

This journal article is available at Institutional Knowledge at Singapore Management University: https://ink.library.smu.edu.sg/sis_research/7785

QuoTe: Quality-oriented Testing for Deep Learning Systems

JIALUO CHEN, Zhejiang University, China
JINGYI WANG*, Zhejiang University, China
XINGJUN MA, Fudan University, China
YOUCHENG SUN, University of Manchester, UK
JUN SUN, Singapore Management University, Singapore
PEIXIN ZHANG, Zhejiang University, China
PENG CHENG*, Zhejiang University, China

Recently, there has been a significant growth of interest in applying software engineering techniques for the quality assurance of deep learning (DL) systems. One popular direction is deep learning testing, i.e., given a property of test, defects of DL systems are found either by fuzzing or guided search with the help of certain testing metrics. However, recent studies have revealed that the neuron coverage metrics, commonly used by most existing DL testing approaches, are not necessarily correlated with model quality (e.g., robustness, the most studied model property), and are also not an effective measurement on the confidence of the model quality after testing. In this work, we address this gap by proposing a novel testing framework called QuoTe (i.e., **Qu**ality-**o**riented **Te**sting). A key part of QuoTe is a quantitative measurement on 1) the value of each test case in enhancing the model property of interest (often via retraining), and 2) the convergence quality of the model property improvement. QuoTe utilizes the proposed metric to automatically select or generate valuable test cases for improving model quality. The proposed metric is also a lightweight yet strong indicator of how well the improvement converged. Extensive experiments on both image and tabular datasets with a variety of model architectures confirm the effectiveness and efficiency of QuoTe in improving DL model quality, i.e., robustness and fairness. As a generic quality-oriented testing framework, future adaptions can be made to other domains (e.g., text) as well as other model properties.

CCS Concepts: • **Software and its engineering → Software testing and debugging**; • **Computing methodologies → Neural networks**.

Additional Key Words and Phrases: deep learning, testing, robustness, fairness.

1 INTRODUCTION

Deep learning (DL) [38] has been the core driving force behind the unprecedented breakthroughs in solving many challenging real-world problems such as object classification [65], speech recognition [23] and natural language processing [11]. However, DL systems are known to have defects in terms of robustness or fairness. For instance, a self-driving car killed the driver as the object detector failed to recognize the white truck against a bright sky [5], and a photo-tagging app tagged pictures of dark-skinned people as gorillas [24]. Along with its

*Both corresponding authors.

Authors' addresses: Jialuo Chen, Zhejiang University, China, chenjialuo@zju.edu.cn; Jingyi Wang, Zhejiang University, China, wangjyee@zju.edu.cn; Xingjun Ma, Fudan University, China, xingjunma@fdu.edu.cn; Youcheng Sun, University of Manchester, UK, youcheng.sun@manchester.ac.uk; Jun Sun, Singapore Management University, Singapore, junsun@smu.edu.sg; Peixin Zhang, Zhejiang University, China, pxzhang94@zju.edu.cn; Peng Cheng, Zhejiang University, China, lunarheart@zju.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
1049-331X/2023/2-ART $15.00
https://doi.org/10.1145/3582573

Just Accepted

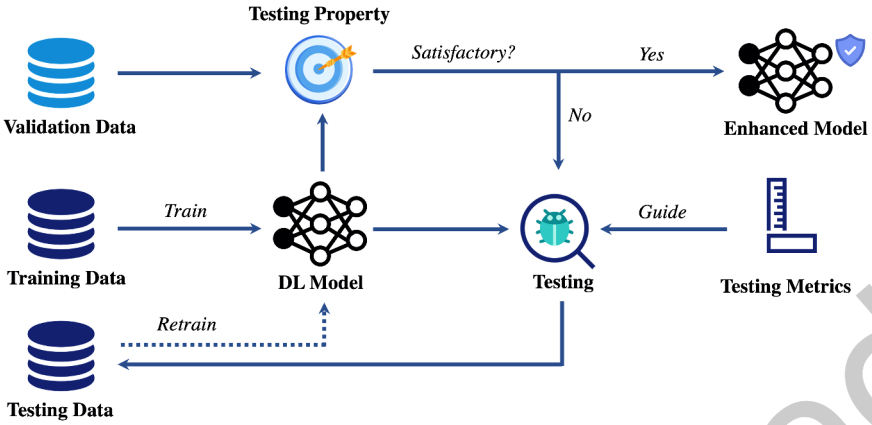

Fig. 1. The overview of QuoTe testing framework. It starts from an initial DL model trained from the training dataset in a standard way. Then, it applies a testing algorithm to generate a new set of test cases for retraining the model to improve the specified testing property. QuoTe iteratively generates the test suite and retrains the model at each iteration. It checks whether the property of the retrained model is satisfactory using an independent quality validation dataset at the end of iterations. If the answer is yes, it terminates and outputs the enhanced model; otherwise, QuoTe continues until the requirement is satisfied or the testing budget is exhausted.

prevalent applications, such DL defects naturally raise great concerns about its trustworthiness, especially when deployed in safety- and ethic-critical applications such as face recognition [54], autonomous driving [15, 41] and medical diagnosis [18, 47].

Significant efforts have been made in the software engineering community to enhance the quality of DL systems with a focus on threats like adversarial examples (on robustness) [19, 55, 63, 70]. Adversarial examples are inputs that are formed intentionally by applying small perturbations to the normal examples, aiming to fool the DL systems to give wrong results (see Fig. 4 for an example). Among the works, formal verification aims to prove that no adversarial examples exist in the neighborhood of a given input. Substantial progress has been made using approaches like abstract interpretation [63, 83] and reachability analysis [67]. However, these formal verification techniques are in general computationally expensive and only scale to limited model structures and properties (e.g., local robustness [29]).

Another popular line of work is deep learning testing, which aims to generate test cases that can expose the defects of DL models. For example, DeepXplore [55], a robustness testing technique for deep learning, revealed thousands of incorrect corner case behaviors in the autonomous driving DL systems; Themis [19], a fairness testing technique, detected significant model discrimination towards sensitive features like gender, marital status and race; ADF [87] utilizes gradient information to guide the fairness testing process and is more efficient in finding discriminatory samples. These test cases can then be used to repair the model (e.g., through retraining), potentially improving the quality of DL systems. However, this should not be taken for granted, as recent studies have shown that test cases generated based on existing testing metrics have limited correlation to model robustness and robustness improvement after retraining [13, 26].

There are two key ingredients when it comes to testing DL systems for a given model property of interest such as robustness and fairness. The first key ingredient is the test adequacy metrics [57] (which we call testing metrics for simplicity) used to evaluate the quality of a test case or a test suite. Multiple robustness testing metrics have been proposed, including neuron coverage [55], multi-granularity neuron coverage [45], MC/DC coverage [64],

Just Accepted

combinatorial coverage [44] and surprise adequacy [35]. The common idea is to explore as much diversity as possible in a certain subspace defined based on different abstraction levels of a neuron network model, e.g., neuron activation [55], neuron activation pattern [45], neuron activation conditions [64] and neuron activation vector [35]. The second is the method adopted for generating test cases, which is often done by mutating a set of seed inputs with the guidance of the testing metric. Existing test case generation techniques such as DeepXplore [55], DeepConcolic [64], DeepHunter [81], DeepCT [44] and ADAPT [40] are mostly designed to improve the neuron coverage metrics. While existing testing approaches help expose the defects of DL systems to some extent, recent studies have unfortunately found that neuron coverage metrics are not helpful for improving model robustness [13, 26, 43]. As a result, unlike in the case of traditional software testing where the software quality is more certainly improved after fixing defects revealed through testing, one may not easily improve the quality of a DL system after retraining with the generated test cases.

In this work, we address the above-mentioned limitations of existing DL testing approaches by proposing a novel DL testing framework called QuoTe (i.e., **Qu**ality-**o**riented **T**esting), which is designed with DL model quality enhancement in mind, aiming to bridge the gap between testing and model quality enhancement. As illustrated in Fig. 1, QuoTe distinguishes itself from existing neuron coverage guided testing works in the following important aspects. First, QuoTe is quality-oriented. For a given property (e.g., robustness or fairness), QuoTe takes a user-defined requirement with respect to the property as input and integrates the retraining process into the testing pipeline. The requirement is provided in prior for quality assurance purposes. For instance, the user may require "the enhanced model should mitigate 80% of adversarial examples (potentially from an independent validation)". QuoTe then iteratively improves the model quality by generating test cases and retraining the model to satisfy the requirement. Second, in QuoTe, we propose a novel set of lightweight metrics that are strongly correlated with the property for testing. The metrics can quantitatively measure the relevance of each test case for model enhancement, and are designed to favor test cases that can significantly improve the model quality. Extensive experimental results further confirm the effectiveness and efficiency of QuoTe in improving model robustness and fairness, by up to 67.69% and 58.90% increase on average respectively.

In a nutshell, we make the following main contributions:

- We propose a quality-oriented testing framework (QuoTe) for DL systems as an extension of the RobOT framework [69]. QuoTe provides an end-to-end solution for enhancing the quality of DL systems in terms of certain model properties.
- We propose a set of lightweight testing metrics that quantify the importance of each test case with respect to the testing properties, which are demonstrated to be stronger indicators than existing coverage metrics.
- We implement in QuoTe, a set of fuzzing strategies guided by the proposed metrics to automatically select and generate high-quality test cases for improving the model quality.
- We evaluate QuoTe for two important DL model properties (i.e., robustness and fairness) on 8 benchmark datasets (including 5 image datasets and 3 tabular datasets). Our experiments show that QuoTe is more effective in improving model quality than state-of-the-art test case selection and generation works. Besides, we release our implementation and experimental data for future research [1].

This paper is a significant extension of our previous work published in ICSE 2021 [69], by extending the original idea for robustness testing to a more challenging model property, i.e., fairness. We define the problem and also provide extensive experiments on both tabular and image domain data, to demonstrate the effectiveness of our approach for fairness testing. As a generic quality-oriented testing framework, future adaptions can be made to extend QuoTe to other data domains (e.g., text/NLP) and more testing properties to improve the trustworthiness of DL systems from different dimensions.

[1] https://github.com/Testing4AI/QuoTe

Just Accepted

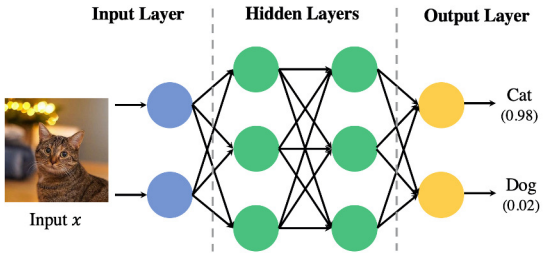

Fig. 2. An example DNN to predict cat or dog.

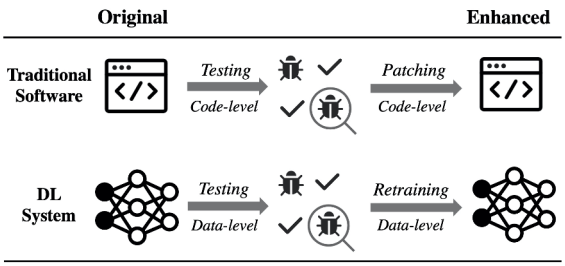

Fig. 3. Comparison between traditional software and DL system quality assurance after testing.

The remainder of the paper is organized as follows. We first provide the background of DL testing in Section 2. We then present the details of QuoTe in Section 3. Extensive experimental results on multiple benchmark datasets are in Section 4. Finally, we discuss related works in Section 5 and conclude in Section 6.

2 BACKGROUND

2.1 Deep Neural Networks

In this work, we focus on deep learning models, e.g., deep neural networks (DNNs) for classification. We introduce a conceptual deep neural network (DNN) as an example in Fig. 2 for simplicity and remark that our approach is applicable for state-of-the-art DNNs in our experiments such as ResNet [27] and VGG [62].

A DNN classifier is a decision function $f : X \rightarrow Y$, which maps an input $x \in X$ (often preprocessed into a vector) into a discrete class label $y \in Y = \{1, 2, \cdots, C\}$, where C is the total number of classes. A typical DNN classifier is composed of L layers: $\{f^1, f^2, \cdots, f^{L-1}, f^L\}$, where f^1 is the input layer, f^L is the output layer[2], and $f^2, \cdots, f^{L-1}$ are the hidden layers, as shown in Fig. 2. We use θ to denote the parameters of f which assign weights to each connected edge between neurons. Given an input x, we can obtain the output of each neuron ne on x, i.e., $\phi(x, ne)$, by calculating the weighted sum of the outputs of all the neurons in its previous layer and then applying an activation function (e.g., Sigmoid, Tanh, or Relu). Given a dataset $D = \{(x_i, y_i)\}_{i=1}^N$, a DNN is often trained by solving the following optimization problem:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f^L(x_i), y_i), \tag{1}$$

where $\mathcal{L}$ is a loss function that calculates a loss by comparing the model output $f^L(x_i)$ with the ground-truth label y_i. The most commonly used loss function for multi-class classification tasks is categorical cross-entropy [50]. The DNN is then trained by computing the gradient w.r.t. the loss for each sample in the dataset D and updating the model parameters θ accordingly.

2.2 DL Model Properties

In this work, we focus on two important and most commonly studied testing properties for DL models, i.e., *robustness* and *fairness*. Here, we provide a brief introduction and formal definitions of them.

2.2.1 Robustness Property. Intuitively speaking, robustness is the degree to which a system can function correctly in the presence of invalid inputs or stressful environmental conditions [60]. For a DL system, we often use

[2]The output is a probability vector and the predicted class label is $\arg\max f^L$.

Just Accepted

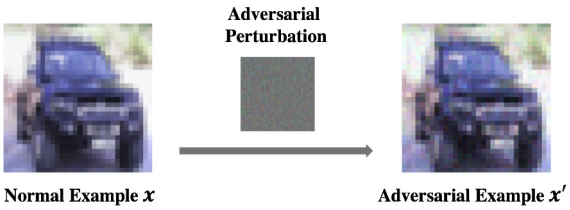

Fig. 4. An adversarial example (x' from x) to fool the DL system to output a wrong label.

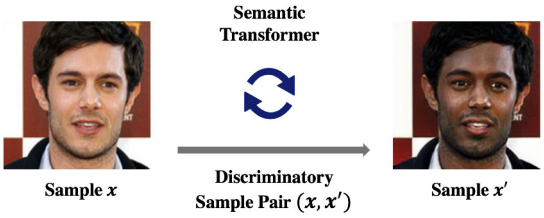

Fig. 5. An discriminatory sample pair (x, x') (on race) make the DL system output inconsistent results.

robustness to measure the resilience towards perturbations (e.g., adversarial noise). Mathematically, the global robustness of a DL model is defined as follows.

Definition 1. ***Global Robustness.*** *Given a DL model $f : X \rightarrow Y$, model f is called (ϵ, δ)-globally-robust iff $\forall x, x' \in X, ||x - x'||_p \leq \epsilon \Rightarrow ||f(x) - f(x')|| \leq \delta$, where ϵ and $||\cdot||_p$ are the perturbation bound and the p-norm for distance measurement respectively, while δ represents the model tolerance.*

Global robustness is theoretically sound, and yet extremely challenging for testing or verification [33]. To mitigate the complexity, multiple attempts have been made to constrain the robustness into the local input space, such as Local Robustness [29], CLEVER [77] and Lipschitz Constant [82]. These local versions of robustness are not ideal either, i.e., shown to have their own limitations [31, 33]. For instance, CLEVER relies on the extreme value theory, making it extremely costly to calculate (e.g., costs an hour for one seed). Nowadays, adversarial robustness (i.e., accuracy against adversarial examples) is the most popular robustness evaluation method in the machine learning security community due to its high efficiency and flexibility. The definition of adversarial example is as follows.

Definition 2. ***Adversarial Example.*** *Given a DL model $f : X \rightarrow Y$, a normal sample $x \in X$ and its ground truth output $g(x)$, x' is an adversarial example of model f by adding a slight perturbation on x ($||x - x'||_p \leq \epsilon$) that satisfies the condition: $f(x') \neq g(x)$, i.e., a wrong prediction.*

Fig. 4 shows an adversarial example x' crafted by the most representative Projected Gradient Descent (PGD) attack [48] (sampled from the CIFAR-10 dataset [37]), which fools the DL system to output a wrong label 'ship' while it is still an 'automobile' in human eyes by adding small perturbations. However, x and x' are totally different in the perspective of the model, thus it usually gives a wrong answer with high confidence. Naturally, a robust model should give consistent results for the neighbours of normal examples.

2.2.2 Fairness Property. Fairness is another desirable property that introduces great social impacts and has gained increasing interests recently. However, there still lacks commonly agreed definition of fairness. In this work, we focus on individual fairness/discrimination, which has been mostly studied in the fairness testing literature [87, 89]. We first consider tabular (structural) data that each input consists of a set of attributes. The intuition is that, given two samples that only differ by certain protected (or sensitive) attributes (e.g., race, gender and age), a fair DL model should output the same label. The definition of individual discrimination is as follows.

Definition 3. ***Individual Discrimination.*** *Let A be the set of protected attributes and Z be the set of non-protected attributes. Given a DL model $f : X \rightarrow Y$, $x \in X$ is a discriminatory sample of model f if there exists another sample $x' \in X$ where x only differs from x' on A attributes (the remaining attributes keep same) that satisfies the condition: $f(x) \neq f(x')$. Further, (x, x') is called a discriminatory sample pair.*

Just Accepted

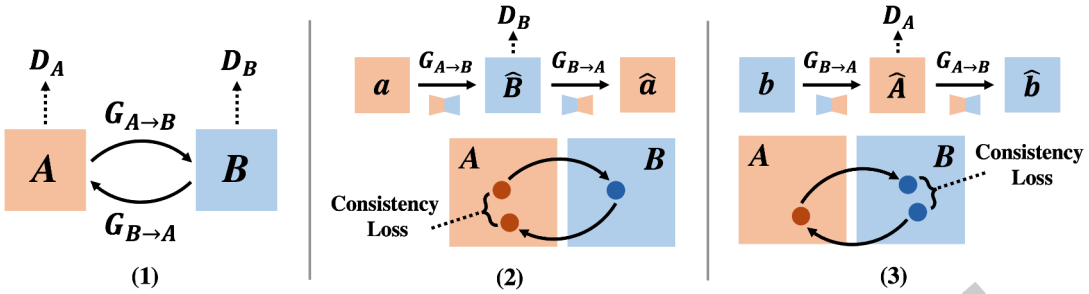

Fig. 6. A simplified view of CycleGAN architecture. $G_{A\to B}$ and $G_{B\to A}$ are the generators that map input from domain A to target domain B and B to A respectively. D_A and D_B are the discriminators that distinguish whether the input is 'real'. The orange and blue solid circles represent the data in domain A and B respectively. The hat symbol indicates the data is synthesized. Objective (2) and (3) will be optimized jointedly in the training process of CycleGAN.

Here, we use the Census Income dataset [36] as a running example for a better illustration. The task is to predict whether the income of an adult is above 50,000$ based on their personal information. The dataset contains more than 30K training samples with 13 attributes each. The following shows two samples x and x':

$$x : [4, 0, 6, 6, 0, 1, 2, 1, \mathbf{1}\ (male), 0, 0, 40, 100] \rightarrow x' : [4, 0, 6, 6, 0, 1, 2, 1, \mathbf{0}\ (female), 0, 0, 40, 100]$$

Among the 13 attributes, there are multiple potential protected attributes, i.e., age, race, and gender. Here, we assume the protected attribute is gender for simplicity, whose index in the feature vector is 8 (highlighted with bold). There are only two different values for this attribute, i.e., 1 representing male and 0 representing female. Given a DL model trained on the dataset, if changing male to female (denoted as x') changes the prediction by the model, we say that x is a discriminatory sample for the model, and (x, x') is a discriminatory sample pair.

Note that tabular data is now still the main study object for the fairness testing problem while image data is much less explored. Rather than explicitly manipulating the discrete value for tabular data as above, the fairness testing becomes much more challenging for the high-dimensional image domain, since manipulating the protected attributes (e.g., race and gender) for images is not intuitive as they are implicit in the input feature space and need to be handled in the semantic level. To address this problem, we adopt an image-to-image transformation technology CycleGAN [90] to help transform images across the protected attributes while preserving other information according to Def. 3. As shown in Fig. 6, CycleGAN provides a mechanism to transfer from domain A to domain B and B to A respectively with two generative models, i.e., $G_{A\to B}$ and $G_{B\to A}$. Similar to traditional GAN [21], CycleGAN contains two discriminators D_A and D_B to distinguish whether the input is 'real', i.e., generated by the generative model or from the original dataset in the corresponding domain. To ensure that the output of generator $G_{A\to B}$ (i.e., $G_{A\to B}(a)$) is in the data distribution of the target domain B, the consistency loss is introduced to make the synthesized inputs more realistic. Thus, the total loss consists of three parts, the loss of generators, the loss of discriminators and the cycle consistency loss, and the four models will be optimized jointedly during the training process.

Fig. 5 shows a discriminatory sample pair (x, x') generated by manipulating the person's race (from domain 'Caucasian' to 'African') in a natural way, which makes the model output inconsistent predictions. There are three key factors influencing the generation results. The first is the protected attribute to cross. Attributes such as gender can be hard to be transformed well since the related features are complex (often entangled). The second is the data in two domains used for training GANs. We use a large-scale high-quality dataset (BUPT-Transferface [71]) on which is costly to train models (more than one day with a GTX 1080 Ti), while adopting other small datasets

Just Accepted

inevitably leads to poor transformation performance due to that the features can not be fully extracted (i.e., underfitting condition). The third can be the training setting, and we adopt the recommended configuration in [90] to achieve balanced transformation results.

2.3 Deep Learning Testing

Deep Learning testing (DL testing) refers to activities designed to reveal the differences between existing and required behaviors of a DL system w.r.t. different kinds of model properties [86]. In the following, we briefly introduce representative testing methods for the two properties and provide pointers for more details.

2.3.1 Robustness Testing. Most existing robustness testing works are based on neuron coverage [55] or its variants [45], aiming to generate test cases to achieve a higher neuron coverage. Simply speaking, a neuron ne is covered if there exists at least one test case x where $\phi(x, ne)$ is larger than a threshold and thus has been activated. DeepXplore [55] is the first testing work for DL, which proposes the first testing metric, i.e., neuron coverage, and a differential testing framework to generate test cases to improve the neuron coverage. DLFuzz [25] and DeepHunter [81] improve DeepXplore under the guidance of neuron coverage defined in [55] and multi-granularity neuron coverage metrics defined in [45] respectively. DeepCT [44] proposes a novel coverage metric that considers the combination of different neurons at each layer. ADAPT [40] adopts multiple adaptive strategies to generate test cases that could significantly improve the neuron coverage metrics defined in [45], and achieves state-of-the-art performance. Besides the above robustness testing methods, popular adversarial attacks in the machine learning security community such as FGSM [22], PGD [48], JSMA [53] and C&W [10] are also used to generate test cases in multiple works [17, 35, 75].

2.3.2 Fairness Testing. Compared to robustness, fairness testing is much less explored (still lacking specifically designed testing metrics). In terms of test case generation, Themis [19] is the first fairness testing method, which explores the input domains for all attributes through random sampling and then checks whether the generated samples are discriminatory. AEQUITAS [68] improves Themis by adopting a two-phase (global and local) generation framework to systematically search the input space, which is guided by a distribution that describes the probability of finding a discriminatory sample. ADF [87] improves the search strategy through adversarial sampling and has been demonstrated to be more effective in generating discriminatory samples. Despite the considerable progress, existing fairness testing works mainly focus on structural data, while complex image data is rarely investigated. DeepFAIT [88] is a recent work that designs specifically for high-dimensional image data, which adopts GAN [21] for crossing sensitive attributes and proposes several strategies to generate a variety of unfair sample pairs.

2.4 Test Case Selection

Test case selection is crucial for improving the model properties with a limited retraining budget and reducing the labeling effort [28, 46]. The key of the selection is to quantitatively measure the value of each test case. DeepGini [17] was recently proposed to select the most informative test cases that are more likely to be misclassified by the model to guide the model retraining process, which is demonstrated to be more effective than neuron coverage-based metrics on improving model robustness. MCP [61] is another uncertainty-based metric, which selects the test cases close to the decision boundaries by the top-2 probabilities. Surprise Adequacy [35] metrics including LSA (likelihood-based) and DSA (distance-based) are utilized to select test cases with larger surprise values [7]. PRIMA [75] was proposed to select test cases via intelligent mutation analysis based on designed mutation rules. A recent work DAT [28] makes an empirical study on the performance of several selection metrics for model enhancement under different data distributions.

Just Accepted

2.5 Model Retraining

Model retraining is demonstrated to be an effective way to improve the quality of a model [3, 16, 35], by rectifying the prediction results of a model on the test cases generated by testing algorithms (e.g., DeepXplore [55] and DeepTest [66]). Retraining is usually done by adding the vulnerability-exposing test cases to the original training dataset, while the training settings such as model structure and loss function keep the same. We make a brief comparison between traditional software testing and DL testing in Fig. 3. Although many testing methods (e.g., random testing [52], symbolic execution [8], concolic testing [72] and fuzzing [20]) can be applied to identify vulnerabilities or bugs for both the traditional software and the DL systems, the workflow differs after testing is done, i.e., the quality of traditional software is enhanced by patching the found bugs, whereas DL systems are often improved by utilizing the found buggy inputs in different ways, e.g., retraining [3, 16, 17] or repairing [42, 84]. Such improvement is guaranteed by patching bugs identified through testing in traditional software (assuming regression bugs are not frequent), i.e., the usefulness of a bug-revealing test for traditional software requires no justification. While for DL systems, the usefulness of a test case can only be judged by taking into account the retraining step. That is, the contribution of each test case to the quality improvement through retraining can be quite different.

2.6 Problem Definition

Unlike existing DL testing methods which focus on finding error cases, our goal is to design an end-to-end quality-oriented testing framework to improve a certain model property via testing. We consider two important properties in this paper, i.e., robustness and fairness. Naturally, two key problems need to be answered: 1) how can we design testing metrics which are strongly correlated with model robustness and fairness? 2) how can we automatically generate test cases favoring the proposed testing metrics?

3 THE QUOTE FRAMEWORK

In this section, we present QuoTe, a novel quality-oriented framework for testing and enhancing (with respect to robustness and fairness) DL systems automatically. The overall framework of QuoTe is shown in Fig. 1. We assume that a requirement on the specified testing property (introduced in Section 2.2) is provided in prior for quality assurance purposes. Note that the requirement is likely application-specific, i.e., different applications may have different requirements. For instance, the user may require the retrained model should mitigate 80% of adversarial examples (potentially from an oracle). We consider such empirical evaluation to be practical.

QuoTe integrates the DL (re)training into the testing framework. It starts from the initial training dataset D_0, and trains an initial DNN f_0 in the standard way. Then, it applies a fuzzing algorithm (see Section 3.5) which is guided by our proposed testing metrics (see Section 3.3) to generate a new set of test cases D_t, for retraining the model f_0 to improve the specified property P. The retraining step distinguishes QuoTe from existing DL testing work and it places a specific requirement on how the test cases in D_t are generated and selected, i.e., the test cases should be helpful in improving f_0's property P after retraining. QuoTe iteratively generates the test suite D_t and retrains the model f_n at each iteration. Afterwards, it checks whether the property of the new model f_n is satisfactory using an independent quality validation dataset D_v, probably subject to an acceptable degrade of the model's accuracy on normal data. If the answer is yes, it terminates and outputs the enhanced model f_n; otherwise, QuoTe continues until the requirement is satisfied or the testing budget is exhausted.

Notice that in this work, we consider mainly faults related to training data quality among all the possible DL faults [30] and one possible fix scenario (i.e. model retraining by enriching training data rather than changing the DL model architecture or hyper-parameters), thus QuoTe is a complementary to testing works focus on fixing fundamental issues such as network design or implementation. In the following, we illustrate each component of QuoTe in detail.

Just Accepted

3.1 Empirical Evaluation for Model Properties

In this work, we focus on two important testing properties for DL systems, i.e., *robustness* and *fairness*. Although many DL testing works in the literature claim a potential improvement on the DL model property by retraining using the generated test suite, such a conjecture is often not rigorously examined. This is partially due to the ambiguity associated with terms such as robustness. For instance, the evaluations of [55, 64, 66, 81] are based on the empirical accuracy of the validation set [86].

Based on the definitions in Section 2.2, we adopt the following empirical definitions for robustness and fairness in QuoTe, which are commonly used for model quality evaluation in the machine learning community with respect to the testing properties [9, 48, 73, 74, 85, 87].

Definition 4. ***Empirical Robustness (ER).*** *Given a DL model $f : X \rightarrow Y$ and a validation set $D_v = \{(x_i, y_i)\}_{i=1}^{M}$, we define its empirical robustness $\mu : (f, D_v) \rightarrow [0, 1]$ as $\gamma_R = \frac{1}{M}\sum_i^M \mathbb{1}(f(x_i) = y_i)$, i.e., the accuracy of f on D_v.*

Definition 5. ***Empirical Fairness (EF).*** *Given a DL model $f : X \rightarrow Y$ and a validation set $D_v = \{(x_i, x_i')\}_{i=1}^{M}$, we define its empirical fairness $\mu : (f, D_v) \rightarrow [0, 1]$ as $\gamma_F = \frac{1}{M}\sum_i^M \mathbb{1}(f(x_i) = f(x_i'))$, i.e., the consistency (or non-discrimination) of f on D_v.*

Note that the selection of D_v is crucial for the practical evaluation of the model quality and it is still an open research problem on how to form D_v. In practice, users often adopt different adversarial attacks (e.g., PGD for robustness [48] and ADF for fairness [87]) to construct such a validation set using adversarial/discriminatory samples. This empirical view is testing-friendly and facilitates QuoTe to efficiently compare the quality of models before and after testing (retrained). Moreover, it is also practical, as it connects the evaluation with many existing adversarial attacks and testing techniques.

3.2 A General View of QuoTe

Alg. 1 presents the high-level algorithmic design of QuoTe for the workflow of DL testing on a specific model property P (e.g., R stands for *Robustness* and F for *Fairness*) based on Def. 4 and Def. 5. The initially trained model f_0 is given as an input to the algorithm and the testing and retraining iterations in QuoTe are conducted within the main loop (Line 2-6). The loop continues until the user-specified empirical quality requirement r is satisfied (Line 2) or the computational budget is exceeded. QuoTe aims to bridge the gap between DL testing and retraining. Let *Test* (Line 3) denote a fuzzing algorithm (guided by certain metrics) to generate test cases to find defects on property P. The objective of quality-oriented testing is to improve the property P through testing. Formally, given a deep learning model f, the goal of QuoTe at each iteration is to improve the following:

$$EP\Big(\min_{\theta} \frac{1}{|D|}\sum_{i=1}^{|D|} \mathcal{L}_{(x_i,y_i)\in D}(f^L(x_i), y_i), D_v\Big), \tag{2}$$

where EP represents ER (Def. 4) when the target model property P is *Robustness*. Intuitively, the testing metric should be designed in such a way that after retraining with the generated test cases, objective (2) is improved. This objective directly links the testing metric to the model property.

In the remaining section, we realize the testing method in Line 3 by answering two questions: 1) How should we design test metrics that are strongly correlated with the model properties? and 2) How can we select and generate valuable test cases guided by the proposed metrics for model enhancement?

3.3 Property-correlated Testing Metrics

Our first goal is to design testing metrics that are strongly correlated with model properties for testing. Let's begin with the robustness property.

Just Accepted

Algorithm 1 QuoTe (f_0, D, D_v, r)

```
Initial model f = f_0
while EP(f, D_v) < r and not exceed the computational budget do
    D_t ← Test(f, D)
    D ← D ∪ D_t
    Update f by further training the model with D_t
end while
return Enhanced model f on property P
```

3.3.1 Robustness-oriented Metrics. We note that there have been some efforts to modify the standard training procedure (Obj. 2) in order to obtain a more robust model. For instance, the most effective and successful approach so far is robust training [48, 73], which incorporates an adversary in the training process so that the robustness of the trained model can be enhanced by minimizing the loss of adversarial examples iteratively (also known as the Min-Max optimization problem):

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \max_{||x_i' - x_i||_p \leq \epsilon} \mathcal{L}(f^L(x_i'), y_i). \tag{3}$$

At the heart of robust training is to identify a strong (ideally worst-case) adversarial example x' around[3] a normal example x and train the model so that the loss on the strong adversarial example can be minimized. This inspires us to consider DL testing analogously in terms of how we generate test cases (around a normal example) and retrain the model with the test cases to improve the model robustness. The key implication is that when we design robustness-oriented testing metrics to guide testing, we should evaluate the usefulness of a test case from a loss-oriented perspective.

Let x_0 be the seed for testing. We assume that a test case x^t is generated in the neighborhood $\epsilon-$ball around x_0, i.e., $\{x|\ ||x - x_0||_p \leq \epsilon\}$ using a given testing method. The main intuition is that a test case which induces a higher loss is a stronger adversarial example, which is consequently more helpful in training robust models [48] and relevant to improving the model robustness through retraining. Based on this intuition, we propose two levels of testing metrics on top of the loss as follows.

1) Zero-Order Loss (ZOL). The first metric directly calculates the loss of a test case with respect to the DL model. Formally, given a test case x^t (generated from seed x), a DL model f, the loss of x^t on f is defined as:

$$ZOL(x^t, f^L) = \mathcal{L}(f^L(x^t), y), \tag{4}$$

where y is the ground-truth label of x, and $\mathcal{L}$ is the categorical cross-entropy loss function. For test cases generated from the same seed, we prefer test cases with higher losses, which are supposed to be more helpful in improving the model robustness via retraining.

2) First-Order Loss (FOL). The loss of the generated test cases can be quite different for different seeds and models It is generally easier to generate test cases with high losses around seeds, which unfortunately do not generalize well. Thus, ZOL is unable to measure the value of the test cases in a unified way. To address this problem, we propose a more fine-grained metric that could help us measure to what degree we have achieved the highest loss in the seed's neighborhood. The intuition is that, given a seed input, the loss around it often first increases and eventually converges if we follow the gradient direction to modify the seed [48]. Thus, a criterion that measures how well the loss converges can serve as the testing metric. A test case with better convergence

[3] A $\epsilon-$ball defined according to a certain L_p norm.

Just Accepted

quality corresponds to a higher loss than its neighbors. Next, we introduce First-Order Stationary Condition (FOSC) to provide a measurement on the loss convergence quality of the generated test cases.

Formally, given a seed input x_0, its neighborhood area $\mathcal{X} = \{x| \ ||x - x_0||_p \leq \epsilon\}$, and a test case x^t, the FOSC value of x^t is calculated as:

$$c(x^t) = \max_{x \in \mathcal{X}} \langle x - x^t, \nabla_x h(\theta, x^t) \rangle, \tag{5}$$

where $h(\theta, x^t) = \mathcal{L}(f^L(x^t), y)$. In [73], it is proved that the above problem has the following closed form solution if we take $\infty-$norm for $\mathcal{X}$:

$$c(x^t) = \epsilon ||\nabla_x h(\theta, x^t)||_1 - \langle x^t - x_0, \nabla_x h(\theta, x^t) \rangle. \tag{6}$$

However, many existing DL testing works generate test cases from the L_2 norm neighborhood, making the above closed-form solution for L_∞ unavailable. We thus consider solving the formulation in Eq. 5 with L_2 norm and obtain the solution as follows:

$$c(x^t) = \epsilon ||\nabla_x h(\theta, x^t)||_2. \tag{7}$$

Proof. According to Cauchy–Schwarz inequality:

$$\begin{aligned} |\langle x - x^t, \nabla_x h(\theta, x^t) \rangle|^2 \leq \\ \langle x - x^t, x - x^t \rangle \cdot \langle \nabla_x h(\theta, x^t), \nabla_x h(\theta, x^t) \rangle \leq \\ \epsilon^2 \cdot (||\nabla_x h(\theta, x^t)||_2)^2. \end{aligned}$$

Since there must exist $x \in \mathcal{X}$ such that $x - x^t$ and $\nabla_x h(\theta, x^t)$ are in the same direction, we thus have:

$$\max |\langle x - x^t, \nabla_x h(\theta, x^t) \rangle|^2 = \epsilon^2 \cdot (||\nabla_x h(\theta, x^t)||_2)^2.$$

Finally, we have:

$$c(x^t) = \max |\langle x - x^t, \nabla_x h(\theta, x^t) \rangle| = \epsilon \cdot ||\nabla_x h(\theta, x^t)||_2.$$

□

Note that FOSC (or FOL) (in both Eq. 6 and Eq. 7) can be calculated efficiently, whose main cost is a one-time gradient computation (easy to obtain with all the DL frameworks). The FOL value represents the first-order loss of a given test case. The loss of a test case converges and achieves the highest value if its FOL value equals zero. Thus, a smaller FOL value means a better convergence quality and a higher loss relatively.

3.3.2 Fairness-oriented Metrics. Different from the robustness-oriented metrics, we need to evaluate the usefulness of a discriminatory sample pair using joint loss for the fairness property. Intuitively, a discriminatory sample pair which induces a higher joint loss is more helpful in reducing the discrimination.

Let x_0 be the seed for testing. We assume that a test case (i.e., discriminatory sample) x_1^t is generated around x_0 and (x_1^t, x_2^t) is an identified discriminatory sample pair (see Def. 3) through fairness testing techniques like [19, 68]. We adapt the ZOL metric for joint loss of fairness property as follows:

$$ZOL((x_1^t, x_2^t), f) = -\sum_i^C f_i^L(x_1^t) \cdot \log f_i^L(x_2^t), \tag{8}$$

where $f_i^L(x)$ is the probability output of label i and C is the total number of classes. This objective aims to impose the model to output similar predictions for (x_1^t, x_2^t) for reducing the individual discrimination towards to sample x_1^t. Based on Eq. 8, we could obtain the corresponding FOL metric for the fairness property according to Eq. 6 and Eq. 7 straightforwardly.

Just Accepted

Algorithm 2 KM-ST(D_t, k, n)

1: Let $D_s = \emptyset$
2: Let FOL_{max} and FOL_{min} be the maximum and minimum FOL values of D_t respectively
3: Equally divide range $[FOL_{min}, FOL_{max}]$ into k sections $KR = [R_1, R_2, \cdots, R_k]$
4: **for** each range $R \in KR$ **do**
5: Randomly select n/k samples D_r from D_t whose FOL values are in R
6: $D_s = D_s \cup D_r$
7: **end for**
8: **return** D_s

3.3.3 Comparison with Neuron Coverage Metrics. Compared to existing neuron coverage metrics [45, 55], our proposed loss-based metrics have the following main advantages. First, both our ZOL and FOL testing metrics are strongly correlated to the adversarial strength of the generated test cases regarding on the certain model property. Thus, our metrics can serve as strong indicators of the degree of satisfying the property after retraining. Second, our metrics are also able to measure the value of each test case in retraining, which further helps us select valuable ones from a large number of test cases to reduce the heavy retraining cost. Moreover, our metrics are structure-unaware, which work without the need to keep a heavy map of neuron activations as in the case of existing coverage metrics, and thus, are lightweight and easy to calculate.

3.4 FOL-Guided Test Case Selection

In the following, we show the usefulness of the proposed metrics through an important application, i.e., test case selection from a massive amount of generated test cases. Note that by default, we use the FOL metric hereafter due to the limitation of ZOL as described above.

Test case selection is crucial for improving the model with respect to the property with a limited retraining budget. The key of the selection is to quantitatively measure the value of each test case. So far, this problem remains an open challenge. Prior work like DeepGini [17] has been proposed to calculate a Gini index of a test case x from the model's output probability distribution:

$$DeepGini(x, f) = 1 - \sum_{i}^{C} f_i^L(x^t)^2. \tag{9}$$

DeepGini's intuition is to favor those test cases with the most uncertainty (i.e., a more flat probability distribution) under the current model's prediction. The larger the DeepGini value is, the higher the uncertainty.

Compared to DeepGini, our FOL metric contains fine-grained information at the loss level and is strongly correlated with certain model testing properties. Given a set of test cases D_t, we introduce the following two test case selection strategies based on the FOL distribution to select a subset $D_s \subset D_t$ for retraining the model. Let $D_t = [x_1, x_2, \cdots, x_M]$ be a ranked list in descending order according to the FOL value, i.e.,$\forall i \in [1, M-1]$, we have $FOL(x_i) \geq FOL(x_{i+1})$.

1) K-Multisection Strategy (KM-ST). The idea of KM-ST is to uniformly sample the FOL space of D_t, and Alg. 2 shows the details. Assume we need to select n test cases from D_t. We first equally divide the range of FOL into k sections (KR) at line 3. Then for each range $r \in KR$, we randomly select the same number of test cases at line 5. We also consider the pure random strategy (RAND) without the range segmentation for comparison. An extreme case is that the FOL values of D_t follow a constant distribution, and there would be no difference between the two strategies from the loss convergence perspective.

Just Accepted

Algorithm 3 FOL-Fuzz($f, seeds\_list, \epsilon, lr, k, \lambda, iters$)

1: Let $fuzz\_result = \emptyset$
2: **for** $seed \in seeds\_list$ **do**
3: Maintain two lists $s\_list_1 = s\_list_2 = \{seed\}$
4: Obtain $FOL_{max} = FOL_{min} = FOL(seed)$
5: **while** s_list_1 is not empty **do**
6: Obtain a seed $x = s\_list_1.pop()$
7: Let $x' = x$
8: **for** $iter = 0$ to $iters$ **do**
9: Set optimization objective Obj_A as Eq. 10
10: Obtain $grads = \frac{\nabla Obj_A}{\nabla x'}$
11: Obtain $perb = processing(grads, lr)$
12: Obtain $x' = x' + perb$
13: Obtain $dis = Dist(x, x')$
14: **if** $FOL(x') \geq FOL_{max}$ and $dis \leq \epsilon$ **then**
15: $FOL_{max} = FOL(x')$
16: $s\_list_1.append(x')$
17: **if** satisfactory(x') **then**
18: $fuzz\_result.append(x')$
19: **end if**
20: **end if**
21: **end for**
22: **end while**
23: **while** s_list_2 is not empty **do**
24: Obtain a seed $x = s\_list_2.pop()$
25: Let $x' = x$
26: **for** $iter = 0$ to $iters$ **do**
27: Set Obj_A as Eq. 10 (opposite to line 9)
28: Repeat line 10 to 13
29: **if** $FOL(x') \leq FOL_{min}$ and $dis \leq \epsilon$ **then**
30: $FOL_{min} = FOL(x')$
31: $s\_list_2.append(x')$
32: Repeat line 17 to 19
33: **end if**
34: **end for**
35: **end while**
36: **end for**
37: **return** $fuzz\_result$

Just Accepted

2) Bi-End Strategy (BE-ST). The idea of BE-ST is to form D_s by equally combining test cases with small and large FOL values. This strategy mixes test cases of strong and weak adversarial strength in the context of the model loss landscape, which is inspired by a recent work on improving standard robust training [34]. Given a ranked D_t, we hope to take a number of test cases from the two ends of the list to compose D_s.

Algorithm 4 DiscriminationCheck(x)

```
1: for each element t in x do
2:     if t ∈ 𝕀_A then                          ▷ 𝕀_A is the valuation domain of the protected attributes.
3:         for v in 𝕀_A\{t} do
4:             Obtain x' by replacing t in x with v
5:             if f(x) ≠ f(x') then
6:                 return True
7:             end if
8:         end for
9:     end if
10: end for
11: return False
```

3.5 FOL-Guided Fuzzing

Next, we introduce a simple yet efficient fuzzing strategy to generate test cases based on FOL. Note that since we have no prior knowledge of the FOL distribution, we are not able to design a fuzzing strategy for KM-ST. Instead, we design a fuzzing algorithm for the BE-ST strategy. The idea is to greedily search for test cases in two directions, i.e., with both small and large FOL values.

Alg. 3 presents the details. The inputs include the model f, the list of seeds to fuzz *seeds_list*, the perturbation bound ϵ, the learning rate lr, the number of labels for optimization k, the weight λ on how much we favor FOL during fuzzing and lastly the number of iterations *iters* for one seed. For each seed in the *seeds_list*, we maintain two lists of seeds s_list_1 and s_list_2 at line 3 for two directions. After obtaining a seed x from s_list_1 (line 5), we iteratively add perturbation on it from line 7 to line 21 in the increasing direction guided by FOL (opposite direction for s_list_2 at line 24 to 34). We set the following generic objective Obj_A for optimization (line 9, 27):

$$Obj_A = obj_P \pm \lambda \cdot FOL(x') \tag{10}$$

For the robustness property, we set obj_P as an adversarial objective as $obj_P = \sum_{i=2}^{k} f_{c_i}^L(x') - f_{c_1}^L(x')$, where c_i is the label with the i^{th} largest softmax probability of $f^L(x)$ (e.g., c_1 is the maximum), $f_{c_i}^L(x)$ is the softmax output of label c_i. The idea is to guide the perturbation towards changing the original label (i.e., generating an adversarial example) whilst increasing or decreasing the FOL value conditionally. While for fairness, there are two strategies for finding discriminatory sample pairs. AEQUITAS [68] uses random perturbations, while ADF [87] applies adversarial perturbations. Our empirical evaluation shows that the latter strategy is more effective in finding discrimination samples (more than 5× on average). Thus, we follow [87] to set $objP = |f_{c_1}^L(x') - f_{c_1}^L(x)|$ as an adversarial objective similarly for finding the target sample x_1^t, and then checking whether there exists a discriminatory sample pair (x_1^t, x_2^t) following Alg. 4.

We then obtain the gradient of the final objective (line 9) and calculate the perturbation based on the gradient by multiplying the learning rate lr and a randomized coefficient to reduce the chance to have duplicate perturbations (line 11). We run two kinds of checks to achieve the BE-ST strategy respectively (line 14 and 29). If the FOL value of the new sample after perturbation (x') is either larger (line 14) or smaller (line 29) than the threshold, we add x' to the seed list (line 16 and 31). Furthermore, we add x' to the fuzzing result if it satisfies the check (line 17 and 32). For robustness, the check is satisfied if x' has a different label with the original seed x, while for fairness, the check is satisfied if x' is a discriminatory sample (i.e., there exists another x'' such that (x', x'') makes a discriminatory sample pair according to Def. 3). Note that compared to coverage-guided fuzzing algorithms which need to profile and update neuron coverage information [40, 81], our FOL-guided fuzzing algorithm is much more lightweight, i.e., whose main cost is to calculate a gradient at each step.

Just Accepted

Hyper-parameters. Our algorithm (Alg. 3) involves 5 hyper-parameters totally, ϵ, lr, k, λ and $iters$. Assigning appropriate values to them is important for the fuzzing process. The first two hyper-parameters ϵ and lr influence the adding perturbation $perb$ (line 11) directly. For instance, with small ϵ and lr values often fails to make notable perturbations on the test cases, while with large values makes the generated inputs far away from the oracle. We set ϵ to 0.5 (after $Dist$ calculation) and lr to 0.1 respectively following ADAPT [40]. Hyper-parameter k is the number of labels in obj_P which controls the diversity of target labels. For example, if k is 2, the target label during optimization will be always the label with the second largest score. We set k to 5 following existing testing works, i.e., DeepXplore [55] and DLFuzz [25]. Hyper-parameter $iters$ is the maximum number of iterations for one seed to find the satisfying input, we set it to 3 to perform multiple times within the testing budget. Lastly, hyper-parameter λ in Eq. 10 controls how much the FOL metric is favoured during the joint optimization process. Intuitively, if λ is very small (e.g, 0.1), the diversity of the generated test cases will be limited due to the lack of search. We set λ to 1.0 to better balance the guidance of the testing metric and the adversarial objective. In experiments, we manually fix the values as above and leave further exploration on tuning these hyper-parameters for boosting the performance as future work.

4 EXPERIMENTAL EVALUATION

In the following, we evaluate QuoTe through multiple experiments.

4.1 Experiment Settings

4.1.1 Properties, Datasets and Models. In this work, we focus on two main properties for DL testing, i.e., robustness and fairness. In total, we adopt 8 benchmark datasets widely used in the literature for the evaluation, including 5 image datasets (4 for robustness and 1 for fairness) and 3 tabular datasets (all for fairness). We first briefly introduce the five image datasets. MNIST [39] is a handwritten digits (from 0 to 9) dataset, consisting of 70,000 images with size $28 \times 28 \times 1$. FASHION [80] is a fashion products (e.g., coat, shirt) dataset, consisting of 70,000 images with size $28 \times 28 \times 1$. SVHN [51] is a house number dataset that contains 99,289 natural scene images with size $32 \times 32 \times 3$. CIFAR-10 [37] has 60,000 images with size $32 \times 32 \times 3$. FairFace [32] is a collection of 108,501 race-balanced face images with size $224 \times 224 \times 3$. We train an age classifier and take it as a challenging benchmark for the fairness research due to the high complexity. For the three tabular datasets, Census Income [36] has been introduced in Section 2.2.2. German Credit [14] consists of 600 items with 20 attributes, aiming to give an assessment of individual's credit based on personal records. Bank Marketing [49] contains 45,211 items with 16 attributes, which is used to predict whether the client would subscribe to a term deposit.

For each image dataset, we use two different widely used DNN model structures for evaluation. We only consider one standard model structure for tabular data with low input dimensions. The details of all the datasets and models are summarized in Table 1. Note that 'ConvNet-1/2' are mainly composed of multiple convolutional layers and 'MLP-6' is composed of fully-connected layers only.

4.1.2 Test Case Generation. To generate test cases for model quality evaluation in practical, we adopt 2 adversarial attacks [22, 48] and 3 testing approaches [25, 40, 55] for the robustness experiments and 3 testing approaches for fairness [68, 87, 88]. We summarize all the configurations in Table 2. Specifically, for DeepXplore [55], two extra models are used for differential testing, which are trained independently with the same structure but under different training settings, i.e., different optimizers (SGD and Adam) and training epochs (0.5× and 2×). DeepFAIT [88] is a systematic fairness testing framework proposed recently for the image domain. It utilizes GANs to realize semantic transformations across different protected attributes, and designs three strategies (i.e., Random Generation, Gaussian Injection and Gradient-based Generation) to help generate a variety of unfair sample pairs. Following the original paper's configuration, we use BUPT-Transferface [71] as an auxiliary dataset

Just Accepted

Table 1. Testing properties, datasets and models.

Testing Property	Dataset	Domain	Model	#Layers	#Parameters	Accuracy
Robustness	MNIST [39]	Image	LeNet-5	7	107.8K	99.02%
			LeNet-1	5	7.2K	98.59%
	FASHION [80]	Image	LeNet-5	7	107.8K	90.70%
			ConvNet-1	8	594.9K	91.83%
	SVHN [51]	Image	LeNet-5	7	136.9K	88.84%
			ResNet-20	20	274.4K	92.30%
	CIFAR-10 [37]	Image	ConvNet-2	12	2.91M	84.84%
			ResNet-20	20	274.4K	90.39%
Fairness	Census Income [36]	Tabular	MLP-6	6	3.7K	88.15%
	German Credit [14]		MLP-6	6	4.1K	100.0%
	Bank Marketing [49]		MLP-6	6	3.9K	92.26%
	FairFace [32]	Image	ResNet-50	50	23.53M	92.36%
			VGG-16	16	134.30M	84.65%

to train the semantic transformer on race, and then generate mixed test cases through different generation strategies.

4.1.3 Test Case Selection. We adopt a recent work DeepGini [17] (GINI) and the pure random strategy (RAND) as the baselines of the test case selection. Recall that DeepGini calculates a Gini-index (Eq. 9) for each test case according to the output probability distribution of the model (details in Section 3.4). While for RAND, each test case has the same probability of being chosen from the data pool.

4.1.4 Empirical Evaluation. We adopt Def. 4 and Def. 5 to empirically evaluate the robustness and fairness of a model before and after testing (and retraining) respectively. For robustness, we compose the quality validation set D_v for each dataset by combining the adversarial examples generated by FGSM [22] and PGD [48]. For fairness, we compose the validation set D_v by combining the generated discriminatory sample pairs using AEQUITAS [68] and ADF [87] for tabular data, while using DeepFAIT [88] for image data.

4.2 Research Questions

RQ1: What is the correlation between our FOL metric and model testing properties?

For each dataset/model pair (e.g., MNIST/LeNet-5), we first prepare three models with different robustness or fairness levels. The first model (Model 1) is the original trained model. The second model (Model 2) is an enhanced model which is retrained by augmenting 5% of the generated test cases D_t and is more robust/fairer than Model 1 through empirical evaluation. The third model (Model 3) is an enhanced model which is retrained by augmenting 10% of the generated test cases and is considered to be the most robust/fair.

Robustness property. For each model, we conduct attacks to obtain the same number of test cases (i.e., adversarial examples) which are independent to the set used for model retraining. We then calculate and show the FOL distribution of the generated adversarial examples for different models in Fig. 7. We observe that there is a strong correlation between the FOL distribution of these adversarial examples and the model robustness. Specifically, the adversarial examples of a more robust model have smaller FOL values. This is clearly evidenced by Fig. 7, i.e., for every dataset/model pair, the probability density is intensively distributed around zero for Model 3 (the most robust model) while steadily expanding to larger FOL values for Model 2 and Model 1 (with

Just Accepted

Table 2. Test case generation details.

Testing Method	Parameter	MNIST	FASHION	SVHN	CIFAR-10
FGSM [22]	Step Size	0.3	0.3	0.03	0.01
PGD [48]	Steps	10	10	10	10
	Step Size	0.3/6	0.3/6	0.03/6	0.01/6
DeepXplore [55]	Relu Threshold	0.5	0.5	0.5	0.5
DLFuzz [25]	Distance Bound	0.5	0.5	0.5	0.5
ADAPT [40]	Fuzzing Time Per Seed	10s	10s	10s	20s

Testing Method	Parameter	Census	Credit	Bank
AEQUITAS [68]	Max Iteration	10	10	10
ADF [87]	G/L Step Size	1.0/1.0	1.0/1.0	1.0/1.0
	Protected Attribute	Age, Race, Gender	Age, Gender	Age

Testing Method	Parameter	FairFace
DeepFAIT [88]	Max Iteration	10
	G/L Step Size	5.0/1.0
	Protected Attribute	Race

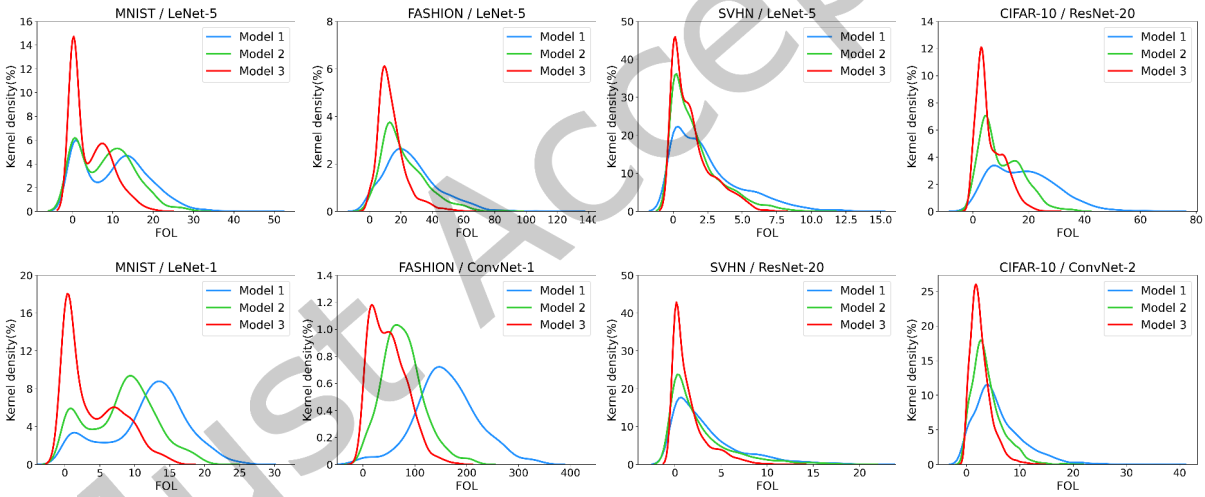

Fig. 7. The FOL distribution of adversarial examples for models with different robustness levels.

Just Accepted

Model 1 larger than Model 2). The underlying reason is that a more robust model in general has a more flat loss distribution and thus a smaller FOL value (which is calculated based on the loss gradient).

Table 3 further validates our observation. Specifically, for each dataset/model pair, six models with different robustness (or fairness) levels are used for calculating the Pearson correlation coefficients [4] with the FOL values. The absolute coefficient value is larger than 0.8 (a high correlation) for 15 times out of 16, and 10 times for the value is larger than 0.9 (a very high correlation). Moreover, the FOL distribution is also influenced by model structures and datasets to a large extent generally. For instance, the FOL range of CIFAR-10/ConvNet-2 is more

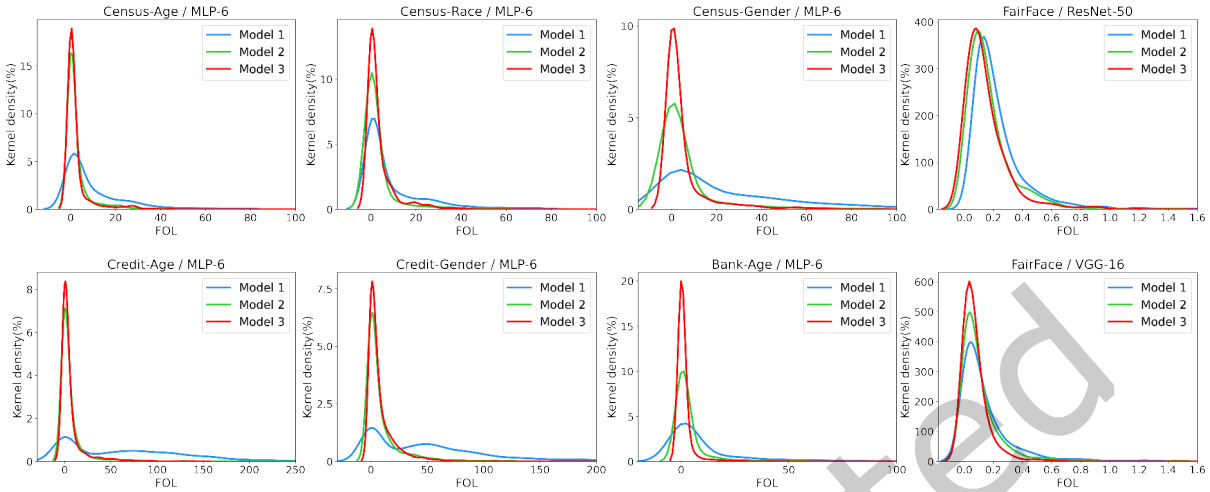

Fig. 8. The FOL distribution of discriminatory sample pairs for models with different fairness levels.

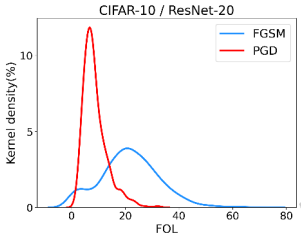

Fig. 9. The FOL distribution of adversarial examples from FGSM and PGD for the CIFAR-10 model.

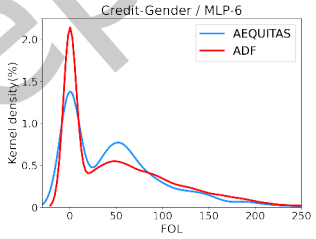

Fig. 10. The FOL distribution of discrimination sample pairs from ADF and AEQUITAS for the Credit model.

narrow than that of CIFAR-10/ResNet-20. While comparing across datasets, FASHION has the widest FOL value range (multiple times larger than others), showing a much more flat FOL distribution.

In addition, we find that adversarial examples crafted by stronger attacks have smaller FOL values. Fig. 9 shows the FOL distribution of adversarial examples from attacking the CIFAR-10 model with FGSM and PGD respectively (see Fig. 14 for other models in Appendix A). We observe that adversarial examples from PGD have significantly smaller FOL values than FGSM. The reason is that stronger attacks like PGD are generating adversarial examples that have better loss convergence quality and induce relatively higher loss.

Fairness property. Similarly, we obtain the same number of discriminatory sample pairs for each model with respect to certain protected attributes. Fig. 8 shows the FOL distribution of discriminatory sample pairs for different models for tabular data (left 3 columns) and image data (right 1 column). The general observation is consistent with that of the robustness property. Due to a much larger model capacity, the FairFace model's fairness converges more slowly (e.g., the difference between Model 1 and Model 2) compared to that of tabular models. The absolute Pearson coefficient values between FOL and fairness as shown in Table 3 are all larger than 0.8. Therefore, there is still a strong correlation between the FOL distribution of discriminatory sample pairs and the model fairness, i.e., the discriminatory sample pairs of a fairer model have smaller FOL values.

Just Accepted

Table 3. Pearson correlation coefficients [4] (absolute value in range [0,1]) between the FOL metric and the empirical testing properties on different models, and a larger value stands for a higher correlation [2, 59].

Dataset	Model	FOL/Robustness Coef. Abs.
MNIST	LeNet-5	0.8243
	LeNet-1	0.8809
FASHION	LeNet-5	0.8908
	ConvNet-1	0.9087
SVHN	LeNet-5	0.9803
	ResNet-20	0.7869
CIFAR-10	ConvNet-2	0.9939
	ResNet-20	0.9912

Dataset	Model	Proc. Attr.	FOL/Fairness Coef. Abs.
Census	MLP-6	Age	0.9927
Census	MLP-6	Race	0.9677
Census	MLP-6	Gender	0.9951
Credit	MLP-6	Age	0.9756
Credit	MLP-6	Gender	0.9898
Bank	MLP-6	Age	0.8518
FairFace	ResNet-20	Race	0.9825
	VGG-16	Race	0.8194

Moreover, for the two testing algorithms used for generating discriminatory sample pairs, i.e., ADF and AEQUITAS, there is a clear difference between their FOL distribution as shown in Fig. 10 (see Fig. 15 for other attributes in Appendix A). Specifically, ADF generates more discriminatory sample pairs with low FOL values, while AEQUITAS has a relatively flat FOL distribution, which indicates that ADF might be a stronger fairness attack than AEQUITAS.

We thus have the following answer to RQ1:

Answer to RQ1: FOL is strongly correlated with model robustness/fairness, and a more robust/fairer model has smaller FOL values for adversarial examples/discriminatory sample pairs.

RQ2: How effective is our proposed FOL metric for test case selection?

To answer the question, we first generate a large set of test cases using different methods, and then adopt different test case selection strategies (i.e., BE-ST, KM-ST, RAND and GINI) to select a subset of test cases with the same size to retrain the model. A selection strategy is considered to be more effective if the retrained model with the selected test cases shows better robustness or fairness. For adversarial attacks, we adopt FGSM (weak) and PGD (strong) attacks to generate a combined set of test cases (ATTACK). For testing algorithms including DeepXplore, DLFuzz, ADAPT, AEQUITAS, ADF and DeepFAIT, we generate a set of test cases for each of them. The parameters used are consistent with Table 2. Then, we apply the four strategies respectively to select ***p%*** (ranging from 1% to 10%) test cases from each test set, then we obtain a retrained model by continuing learning with the augmented data and evaluate its empirical robustness or fairness. To reduce the randomness, we repeat the process 5 times and report the average and deviation results (in the form of $a \pm b$).

Robustness property. Fig. 11a and Fig. 11b show the robustness evaluation results with different percentages of test cases with different model structures. We observe that for all the selection strategies, the retrained model obtained improved robustness to some extent, and the model robustness steadily improves as we augment more test cases (from 1% to 10%) for retraining. By traversing the subfigures, we could find that FOL-guided strategies (both BE-ST and KM-ST) have significantly better performance than Random and DeepGini, with KM-ST wins on 20 out of the 32 cases (dataset/model pairs) and BE-ST wins on 9 out of 32. Specifically, our method achieves 8.27%, 14.11%, 14.79% and 10.83% more robustness improvement than Random on average for the four different sets of test cases, and 28.14%, 37.63%, 44.47% and 34.48% more improvement than DeepGini. The reason behind is that the FOL-based strategy is able to select test cases which have higher and more diverse loss convergence than others, which are better correlated with model robustness. With a close look, we also find that the model property

Just Accepted

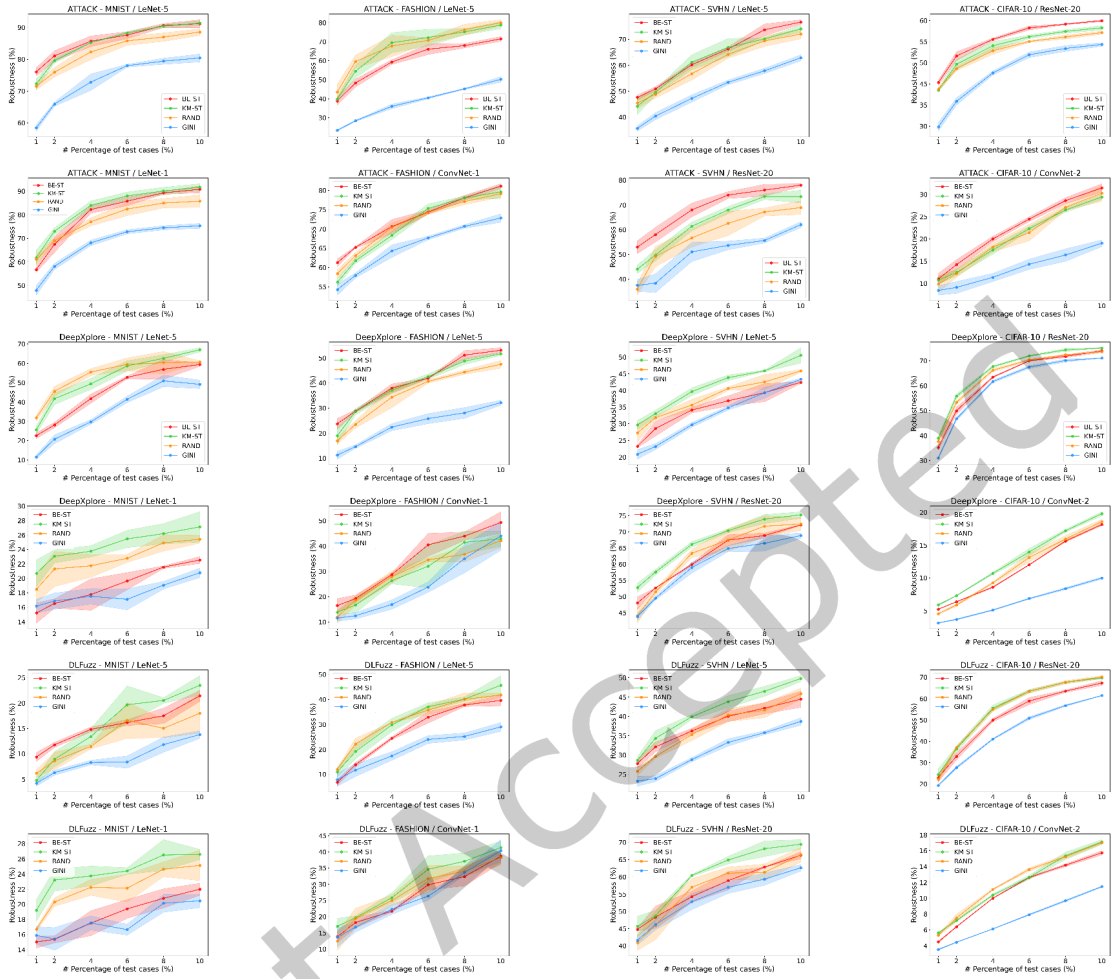

Fig. 11a. Robustness improvement with different selection strategies (with confidence interval of 95% significance level).

Just Accepted

(robustness) is indeed influenced by the model structure. For instance, ConvNet-2 on the CIFAR-10 dataset shows much lower robustness performance compared to ResNet-20 in Fig. 11a. A popular explanation in the machine learning security community is that the large model is easier to be fooled due to the underfit condition (training stage). Nevertheless, all the models still get improved through data enriching and retraining. Note that our evaluation here is not iterative, but is done by retraining the model once (not from scratch). We conduct an iterative retraining experiment on the MNIST/CIFAR-10 datasets to showcase the trend of improvements (see Fig.16 in Appendix A). As expected, the model robustness is steadily improved and the improvement decays along the iterations. The whole retraining process will be early terminated if the quality requirement is met. In the meanwhile, we observe that the retrained models keep the utility as well, i.e., maintaining high accuracy on the clean test set as shown in Table 4.

It is noteworthy that the Random strategy (orange line) performs surprisingly well in some cases, which is close to KM-ST (e.g., ATTACK for FASHION). As Alg. 2 shows, KM-ST samples test cases from a list of ranges

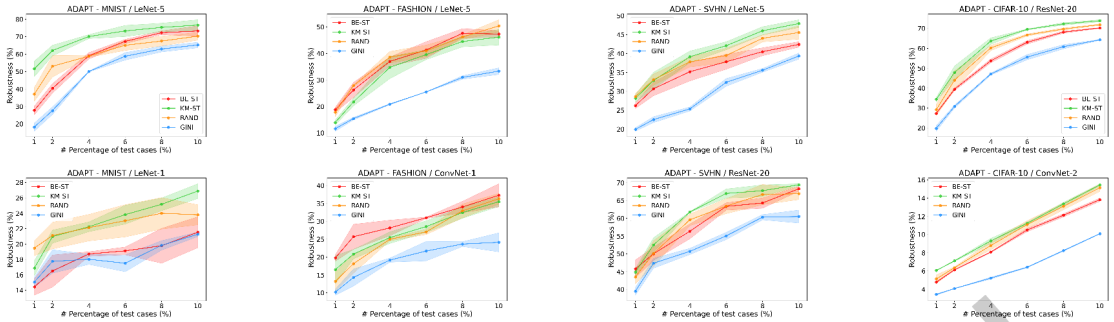

Fig. 11b. Robustness improvement with different selection strategies (with confidence interval of 95% significance level).

Table 4. Clean test accuracy of models before and after retraining with 10% of generated test cases (with confidence interval of 95% significance level).

Dataset	Model	Original	Retrained
MNIST	LeNet-5	99.02%	98.99±0.07%
	LeNet-1	98.59%	98.53±0.03%
FASHION	LeNet-5	90.70%	90.51±0.20%
	ConvNet-1	91.83%	91.91±0.35%
SVHN	LeNet-5	88.84%	88.58±0.54%
	ResNet-20	92.30%	93.81±0.68%
CIFAR-10	ConvNet-2	84.84%	85.23±0.19%
	ResNet-20	90.39%	89.46±0.22%

Dataset	Model	Proc. Attr.	Original	Retrained
Census	MLP-6	Age	88.15%	88.00±0.08%
Census	MLP-6	Race	88.15%	88.10±0.14%
Census	MLP-6	Gender	88.15%	88.02±0.12%
Credit	MLP-6	Age	100.0%	99.50±0.60%
Credit	MLP-6	Gender	100.0%	98.83±1.25%
Bank	MLP-6	Age	92.26%	91.97±0.38%
FairFace	ResNet-20	Race	92.36%	92.64±0.56%
	VGG-16	Race	84.65%	85.33±1.15%

based on the FOL distribution. Considering the extreme case that FOL is a constant distribution, and there will be no difference between these two strategies. This is further validated by Fig. 7 where the FOL distribution of FASHION dataset is much more flat than others (with lower kernel densities). Nevertheless, KM-ST still holds a clear advantage over Random overall (by up to 42.54%).

Moreover, we also observe that different testing algorithms obtain different robustness improvements. ADAPT and DLFuzz have the highest (47.43% on average) and lowest (37.55% on average) robustness improvement respectively while DeepXplore is in the middle (45.94% on average). Adversarial attacks often achieve higher robustness improvement than all three neuron coverage-guided fuzzing algorithms for simpler datasets such as MNIST, Fashion and SVHN. This casts a shadow on the usefulness of the test cases (adversarial examples) generated by neuron-coverage-guided fuzzing algorithms in improving model robustness and is consistent with recent empirical studies [13, 26, 43].

We further conduct experiments to evaluate and compare how robust the retrained models are when using adversarial examples generated in different ways. We use test cases generated by different DL testing algorithms (the parameters are consistent with Table 2) to evaluate the resilience of models to unseen threats. We summarize the results in Table 5. We observe that the robustness drops noticeably (especially so for CIFAR-10), i.e., 20.45%, 28.34% and 22.95% for DeepXplore, DLFuzz, and ADAPT each on average compared to the results in Fig. 11a. Table 6 shows the results under one recent adversarial attack [12] where the robustness also drops to some extent. Intuitively, using adversarial examples from single generation method may be insufficient to face diverse and complex threats in the real-world application scenario. Nevertheless, our test case selection strategies still outperform Random and DeepGini in most cases, and BE-ST shows the best generalization ability to unseen

Just Accepted

Table 5. Robustness performance of models (retrained using adversarial examples from attack algorithms) against test cases generated by DL testing algorithms (with confidence interval of 95% significance level).

Dataset	Model	DeepXplore				DLFuzz				ADAPT			
		BE-ST	KM-ST	Random	DeepGini	BE-ST	KM-ST	Random	DeepGini	BE-ST	KM-ST	Random	DeepGini
MNIST	LeNet-5	86.52±2.01%	83.36±1.58%	80.34±1.49%	71.10±1.99%	82.59±2.98%	79.64±4.25%	76.26±5.68%	69.21±1.46%	76.57±2.78%	74.83±0.72%	70.13±2.23%	62.30±1.91%
	LeNet-1	73.49±6.08%	67.46±3.54%	61.77±6.91%	55.52±4.23%	73.35±5.89%	67.28±3.60%	61.82±6.84%	55.36±4.18%	72.81±6.07%	67.30±3.97%	61.71±6.88%	55.29±4.21%
FASHION	LeNet-5	61.62±3.79%	65.10±3.33%	64.83±1.47%	51.86±1.06%	56.45±3.71%	58.69±3.69%	59.53±1.98%	41.16±2.59%	58.47±1.99%	60.24±2.72%	61.39±2.34%	43.73±1.17%
	ConvNet-1	15.35±0.35%	15.22±0.98%	15.49±1.20%	17.41±1.27%	17.51±1.16%	18.64±1.51%	18.31±0.92%	18.48±1.14%	21.13±1.80%	21.17±1.64%	21.45±1.59%	22.88±1.91%
SVHN	LeNet-5	33.58±1.14%	33.30±1.32%	33.24±1.82%	23.91±0.43%	31.52±1.68%	31.20±1.80%	30.54±0.87%	23.91±0.52%	26.98±2.11%	27.37±1.42%	27.72±1.26%	18.63±1.17%
	ResNet-20	48.51±2.86%	45.43±2.55%	43.96±1.37%	41.65±1.09%	45.28±2.06%	43.64±1.28%	43.25±1.59%	42.10±1.72%	45.67±2.53%	41.18±1.48%	40.20±1.62%	37.23±0.93%
CIFAR-10	ConvNet-2	10.87±0.11%	8.59±0.12%	8.74±0.21%	5.76±0.04%	13.77±0.28%	10.65±0.08%	10.19±0.26%	7.65±0.16%	14.44±0.16%	11.77±0.10%	11.65±0.25%	7.34±0.08%
	ResNet-20	36.48±0.62%	32.25±1.23%	30.24±1.57%	21.43±0.94%	31.32±0.32%	29.03±0.79%	27.21±0.53%	20.01±1.32%	37.63±1.12%	34.45±1.30%	31.78±1.38%	24.59±0.72%
	Average	**45.80%**	**44.21%**	**42.32%**	**36.08%**	**43.97%**	**42.35%**	**40.89%**	**34.74%**	**44.21%**	**42.49%**	**40.75%**	**34.00%**

Table 6. Robustness performance of models under Auto-Attack [12] (with confidence interval of 95% significance level).

Dataset	Model	Auto-Attack			
		BE-ST	KM-ST	Random	DeepGini
MNIST	LeNet-5	88.53±0.37%	87.89±0.53%	85.74±0.81%	77.75±0.25%
	LeNet-1	90.06±1.17%	90.24±1.80%	84.99±0.56%	72.27±0.92%
FASHION	LeNet-5	70.35±1.89%	73.76±1.67%	73.63±2.52%	42.55±1.28%
	ConvNet-1	72.49±0.71%	70.57±1.36%	69.63±0.60%	61.40±1.48%
SVHN	LeNet-5	69.58±0.41%	68.37±0.69%	67.39±0.58%	62.98±0.64%
	ResNet-20	73.56±1.07%	68.07±1.94%	65.89±1.90%	63.02±0.98%
CIFAR-10	ConvNet-2	27.20±0.08%	24.46±0.11%	24.83±0.19%	13.89±0.07%
	ResNet-20	55.60±0.33%	52.99±0.64%	51.21±0.70%	46.26±0.58%
	Average	**68.42%**	**67.04%**	**65.41%**	**55.02%**

attacks (by up to 35.13% enhancement than Random), indicating that test cases with high convergence (hard examples) can help improve model overall robustness. Thus, it is necessary to improve the diversity of test cases for retraining from a perspective that is well correlated with model robustness.

Fairness property. Fig. 12a and Fig. 12b show the fairness evaluation results for tabular data and image data respectively. The main observations are consistent with that of robustness above. For all the strategies, the model's fairness steadily improves as more test cases (discriminatory sample pairs) are used for retraining.

For tabular data, KM-ST has better performance than Random and DeepGini on average, achieving 4.23% and 4.72% more fairness improvement than Random for the two different sets of test cases respectively, with 33.38% and 37.46% more improvement for DeepGini. Interestingly, BE-ST performs best at the early stage in 8 out of 12 cases where the improvement is up to 23.64% more than Random. However, as more test cases are selected for retraining, BEST falls behind KM-ST and Random fast. This phenomenon may be due to the size of the dataset/model being small so that the model converges fast. While for image data with high dimensions, both our strategies perform well, achieving 7.07% more improvement than Random and 11.61% more than DeepGini on average. Moreover, the overall performance is impacted by the protected attribute to some extent as well. For instance, the improvement on the Age attribute is less than that on the Gender, as shown in Fig. 12a.

We further evaluate and compare how fair the retrained models are with an independently-generated discriminatory set, i.e., test cases generated through random sampling in [19]. We summarize the results in Table 7 (for tabular data only). Different datasets or sensitive attributes cause model discrimination at different levels. Overall, our test case selection strategies still outperform Random and DeepGini baselines in most of the cases. In addition, we observe that the average fairness improvement of BE-ST is the highest, which is also consistent

Just Accepted

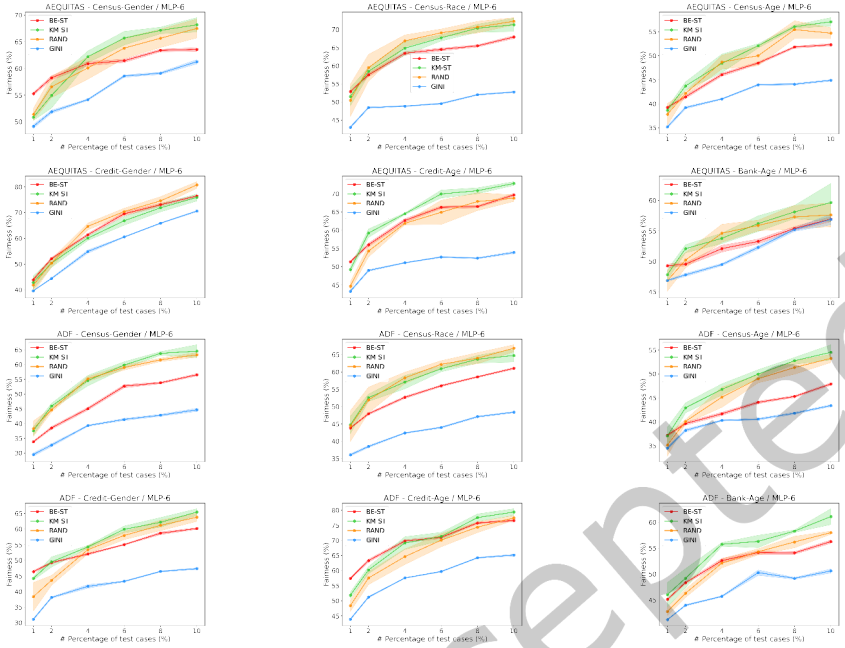

Fig. 12a. Fairness improvement with different strategies (tabular) (with confidence interval of 95% significance level).

Fig. 12b. Fairness improvement with different strategies (image) (with confidence interval of 95% significance level).

Just Accepted

with the results of the robustness property shown in Table 5,6, further validating the potential of our adapted FOL metric in the test case selection scenario.

We thus have the following answer to RQ2:

> **Answer to RQ2:** FOL-guided test case selection is able to select more valuable test cases to improve the model robustness or fairness through testing and retraining.

RQ3: How effective and efficient is our FOL-guided fuzzing algorithm?

To answer the question, we compare our FOL-guided fuzzing algorithm (Alg. 3) with two state-of-the-art testing methods, i.e., ADAPT [40] and ADF [87] on model robustness and fairness respectively, to show the

Table 7. Fairness performance of models against the test cases generated through random sampling (with confidence interval of 95% significance level).

Dataset	Model	Proc. Attr.	AEQUITAS				ADF			
			BE-ST	KM-ST	Random	DeepGini	BE-ST	KM-ST	Random	DeepGini
Census	MLP-6	Age	91.67±0.13%	91.40±0.25%	90.16±0.30%	88.74±0.15%	91.78±0.23%	89.01±0.34%	88.35±0.29%	86.56±0.08%
Census	MLP-6	Race	92.72±0.17%	90.45±0.38%	89.67±0.26%	88.43±0.14%	90.19±0.07%	89.93±0.23%	89.56±0.12%	87.62±0.02%
Census	MLP-6	Gender	97.97±0.08%	97.90±0.26%	98.14±0.19%	98.08±0.21%	97.95±0.15%	98.37±0.12%	98.16±0.11%	98.04±0.05%
Credit	MLP-6	Age	89.68±0.06%	87.33±0.21%	85.72±0.19%	84.74±0.07%	89.30±0.12%	89.04±0.10%	88.05±0.17%	86.89±0.13%
Credit	MLP-6	Gender	95.58±0.25%	94.47±0.14%	94.58±0.34%	94.72±0.11%	95.33±0.10%	94.15±0.18%	94.30±0.33%	94.18±0.04%
Bank	MLP-6	Age	96.64±0.17%	95.93±0.16%	96.25±0.40%	96.12±0.08%	96.61±0.03%	96.17±0.22%	96.31±0.15%	95.80±0.13%
		Average	**94.04%**	**92.91%**	**92.42%**	**91.81%**	**93.53%**	**92.78%**	**92.45%**	**91.52%**

Table 8. Comparison of FOL-fuzz and ADAPT [40] (with confidence interval of 95% significance level). a/b: a is the result of FOL-fuzz and b is the result of ADAPT.

Dataset	Model	10 mins		20 mins	
		# Test case	Robustness↑	# Test case	Robustness↑
MNIST	LeNet-5	3284±206 / 4543±118	47.37±1.94% / 37.71±1.31%	7062±329 / 9227±273	65.88±1.87% / 54.93±1.12%
	LeNet-1	2228±136 / 2705±45	20.60±0.77% / 16.31±0.32%	4171±157 / 5086±28	35.36±1.45% / 20.19±0.84%
FASHION	LeNet-5	9023±265 / 10722±441	33.06±0.84% / 15.44±1.70%	18339±286 / 21947±101	49.16±2.12% / 29.65±3.78%
	ConvNet-1	3303±279 / 2533±236	23.06±0.89% / 13.85±1.43%	6586±304 / 4510±267	25.22±1.13% / 19.00±1.39%
SVHN	LeNet-5	12467±345 / 16763±663	29.86±0.90% / 28.78±1.81%	24903±256 / 33885±520	39.96±0.99% / 36.61±2.07%
	ResNet-20	1369±91 / 3029±43	41.67±0.56% / 42.03±1.20%	2708±155 / 5699±327	47.11±0.68% / 48.72±2.19%
CIFAR-10	ResNet-20	2041±36 / 3575±256	22.82±0.59% / 19.21±1.21%	4023±107 / 6740±287	29.32±3.03% / 23.82±3.44%
	ConvNet-2	4531±162 / 2791±285	7.43±0.49% / 6.48±0.33%	9225±204 / 5772±316	9.02±0.55% / 9.13±0.72%
	Average	**4791 / 5833**	**28.23% / 22.48%**	**9628 / 11608**	**37.63% / 30.26%**

Table 9. Robustness improvement of models after retraining using test cases with different densities (with confidence interval of 95% significance level).

Dataset	Model	Robustness↑ (# Test cases for each seed)			
		1	10	50	100
MNIST	LeNet-5	48.15±2.16%	64.62±3.58%	70.55±2.46%	72.33±2.24%
	LeNet-1	27.62±2.31%	57.94±2.52%	64.09±0.84%	64.80±1.22%
FASHION	LeNet-5	15.81±0.97%	31.72±0.72%	45.26±0.54%	48.51±1.14%
	ConvNet-1	22.92±1.10%	26.24±0.71%	27.93±0.74%	27.63±0.58%
SVHN	LeNet-5	27.21±0.59%	35.57±1.01%	40.24±0.65%	38.98±0.49%
	ResNet-20	36.54±1.44%	45.47±1.91%	49.81±0.85%	50.46±1.53%
CIFAR-10	ConvNet-2	7.75±0.12%	8.35±0.15%	9.17±0.22%	9.86±0.08%
	ResNet-20	24.48±0.34%	27.75±0.89%	30.83±0.46%	31.42±0.87%
	Average	**26.31%**	**37.21%**	**42.24%**	**43.00%**

effectiveness and efficiency of improving the model quality with respect to a certain testing property. To reduce the randomness, we repeat the process 5 times and report the average and deviation results.

Robustness property. We compare FOL-fuzz with the state-of-the-art neuron coverage-guided fuzzing algorithm ADAPT as follows. We run FOL-fuzz and ADAPT on the same set of seeds to generate test cases within

Just Accepted

the same amount of time (i.e., 10 minutes and 20 minutes). Then we retrain the model with all the generated test cases to compare their robustness improvement. The hyper-parameters for FOL-Fuzz are set as follows: $k = 5,\ \lambda = 1.0,\ iters = 3,\ lr = 0.1$, and the parameters for ADAPT are consistent with Table 2.

Table 8 summarizes the results. We observe that ADAPT generates more test cases on average, i.e., 8721 compared to 7210 of FOL-Fuzz. Nevertheless, using test cases founded by FOL-Fuzz (less than ADAPT) to retrain the model significantly improves the model's robustness than ADAPT, i.e., 32.93% compared to 26.37% of ADAPT on average. With a closer look, we could find that ADAPT usually generates much fewer test cases than FOL-Fuzz for large models (e.g., CIFAR-10/ConvNet-2). The reason behind is that ADAPT needs to calculate and keep the global coverage map in each iteration, which is extremely costly for the model with a large number of neurons, while the main cost of FOL-Fuzz lies on the gradient calculation. Fig. 13 further shows several samples generated by the two methods respectively, which reveals that ADAPT tends to generate a lot of test cases around a seed (with little difference) with the core goal of improving certain neuron coverage metric. However, not all these tests are meaningful to improve model robustness. On the other hand, FOL-Fuzz is able to discover more diverse and valuable test cases relatively.

In addition, we explore how the presence of multiple test cases belonging to the same seed affects the retraining performance. We control the generation of FOL-Fuzz to output a different number of test cases for each seed, i.e., 1, 10, 50 and 100, and then retrain models on the generated test cases to evaluate the robustness improvement respectively. Table 9 summarizes the results. We observe that robustness increases as more test cases are generated for each seed, from 26.31% to 43.00% on average. However, the improvement is limited after the number grows to a certain threshold. We only get 0.76% more improvement by adding the number from 50 to 100, and the improvement even drops a little in some cases (e.g., SVHN/LeNet-5). Thus, the size of test cases is not the larger the better, the diversity of test cases is of great importance.

Fairness property. We compare FOL-fuzz with the state-of-the-art fairness fuzzing algorithm ADF for tabular data as follows. We run FOL-fuzz and ADF (global generation) on the same set of seeds to generate test cases (i.e., discriminatory sample pairs). The hyper-parameters for FOL-Fuzz are set as follows: $k = 1,\ \lambda = 1.0,\ iters = 10,\ lr = 1.0$, and parameters for ADF are consistent with Tab. 2.

Tab. 10 shows the results. We observe that ADF generates slightly more test cases on average, i.e., 561 compared to 504 of FOL-Fuzz. Still, the generation process varies with the dataset and protected attribute, which may be caused by the limited search space (tabular data with discrete values). Not surprisingly, we observe that using test cases founded by FOL-Fuzz (although less than ADF) to retrain the model improves the model's fairness more than ADF, which wins in 4 out of 6 cases and achieves 46.56% improvement compared to 43.92% of ADF on average. The improvement of FOL-fuzz on fairness is not significant when compared to that on robustness shown in Tab. 8. One possible reason is that the size of the search space is different (the dimension of tabular data is much less than image data) and the fuzzing process is further limited.

We thus have the following answer to RQ3:

> **Answer to RQ3:** FOL-guided fuzzing is able to efficiently generate more valuable test cases to improve the model robustness and fairness.

4.3 Threats to Validity

Our experimental results demonstrate the effectiveness of QuoTe. However, we acknowledge some threats to the validity of our approach and experiments. We discuss threats according to the guideline for experimental studies in software engineering [58, 79].

Just Accepted

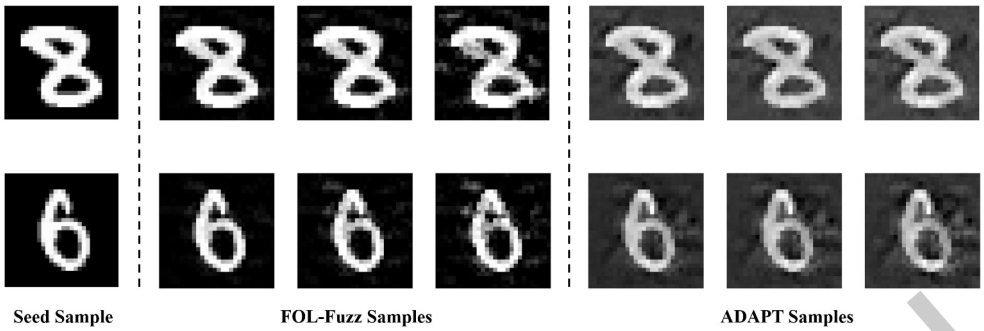

Fig. 13. Generated test cases from FOL-Fuzz and ADAPT testing algorithms on the MNIST dataset.

Table 10. Comparison of FOL-fuzz and ADF [87] (with confidence interval of 95% significance level). a/b: a is the result of FOL-fuzz and b is the result of ADF.

Dataset	Model	Proc. Attr.	1000 seeds	
			# Test case	Fairness↑
Census	MLP-6	Age	821±9 / 656±4	42.90±0.40% / 39.85±0.31%
Census	MLP-6	Race	653±3 / 456±3	42.85±0.17% / 44.02±0.13%
Census	MLP-6	Gender	371±4 / 337±2	47.50±0.19% / 38.13±0.22%
Credit	MLP-6	Age	575±6 / 594±4	53.52±0.15% / 48.88±0.22%
Credit	MLP-6	Gender	235±1 / 449±2	44.75±0.04% / 44.07±0.11%
Bank	MLP-6	Age	366±3 / 873±5	47.84±0.21% / 48.59±0.14%
		Average	**504 / 561**	**46.56% / 43.92%**

The internal threat could be caused by the implementation of QuoTe testing framework and baselines in comparison. To reduce this threat, we borrow the available implementations of the compared methods from released codes by their authors, and carefully check the correctness of our code.

The external threat to validity mainly comes from the datasets and model structures. Regarding the datasets, we consider 8 public datasets in the testing literature, including 5 image datasets and 3 tabular datasets. To reduce the threat from the models, we employ multiple well-known architectures for each dataset (including 8 structures in total) to limit the impact of model dependency to some extent. Since QuoTe is generic and independent of datasets or model structures, it can be adapted with minor adjustments for further use.

The construct threat lies in the empirical evaluation and quality requirement in QuoTe, the hyper-parameter setting, and the randomness in the training process. Following DeepGini [17], We adopt an empirical approach to evaluate properties (robustness and fairness), which might vary with the pre-generated quality validation dataset D_v. So far, it is still an open problem as to how to efficiently measure the robustness (or other properties) of DL models. Our QuoTe testing framework requires a quality requirement on the certain testing property as the input, that could be application-specific and relevant to the model as well. In practice, users could adjust the requirement dynamically regarding the actual scenarios. For the hyper-parameters, we follow the standard setting in existing works. For our method, further research for other datasets may be necessary to identify the hyper-parameters to achieve a more effective and efficient testing. Regarding the randomness, we repeat each experiment 5 times (have retrained more than 6,000 models) and we reported the results with statistical significance evaluation.

Just Accepted

5 RELATED WORK

This work is mainly related to the following lines of works on building trustworthy DL systems.

5.1 Deep Learning Testing

Extensive DL testing works are focused on designing testing metrics to expose the robustness vulnerabilities of DL systems including neuron coverage [55], multi-granularity neuron coverage [45], neuron activation conditions [64], surprise adequacy [35] and mutation adequacy [56]. Along with the testing metrics, many test case generation algorithms are also proposed including gradient-guided perturbation [55, 87], black-box [78] and metric-guided fuzzing [25, 40, 81]. However, these testing works lack rigorous evaluation of their effectiveness in improving the model robustness (although most of them claim so) and have been shown to be ineffective in multiple recent works [13, 26, 43]. Multiple metrics have been proposed in the machine learning community to quantify the robustness of DL models as well [6, 76, 77, 82]. However, most of them are used to evaluate local robustness and are hard to calculate, thus are not suitable to test directly. Compared to the robustness property, the fairness property is much less explored so far. Several works [1, 19, 68] are proposed to find discriminatory samples of machine learning models (deep learning models in [87, 89]). Our work bridges the gap and integrates model retraining into the testing pipeline for better quality assurance. Moreover, our QuoTe testing framework is generic, i.e., it could be extended to other model properties by incorporating specific testing metrics.

5.2 Adversarial Training

The key idea of adversarial training is to improve the robustness of the DL models by considering adversarial examples in the training phase. There are plenty of works on conducting adversarial attacks on DL models (of which we are not able to cover all) to generate adversarial examples such as FGSM [22], PGD [48] and C&W [10]. Adversarial training, in general, may overfit to the specific kinds of attacks that generate the adversarial examples for training [48] and thus can not guarantee robustness on new kinds of attacks. Later, robust training [48] is proposed to train robust models by solving a saddle point problem described in Sec. 3. Motivated by adversarial training, we propose the FOL metric based on loss objective (e.g., pair-loss for fairness), which is strongly correlated with model property and could be used to select more valuable test cases for model retraining to enhance the model property. We believe that adversarial training and QuoTe (like other testing approaches) are complementary, and DL testing is important as testing can continuously generate new testing cases to enhance the model's property more efficiently.

6 CONCLUSION

In this work, we propose a novel quality-oriented testing framework QuoTe for deep learning systems towards improving a specific model property (e.g., robustness and fairness), which focuses on fixing mainly faults related to training data. The core of QuoTe is a set of lightweight metrics to quantify both the value of each test case in improving model property (via retraining) and the convergence quality of the improvement. We also utilize the proposed metric to select and automatically fuzz for more valuable test cases for model quality enhancement. We implemented QuoTe as a self-contained open-source toolkit for future research. Extensive experiments on multiple benchmark datasets verify the effectiveness and efficiency of QuoTe in improving DL model robustness and fairness, which outperforms several state-of-the-art works.

Future Work. In this work, we evaluate the effectiveness and efficiency of QuoTe on two main DL properties, i.e., robustness and fairness. Extending to other less explored properties such as safety and privacy is also worthy of future study, although it is still challenging to achieve complete trustworthiness for DL models. Our work focuses on fixing faults (or bugs) in the training dataset and improving model quality through data augmentation and retraining. We are also aware that there are other fundamental faults in model structure, implementation or

Just Accepted

training setting in reality, which impacts model property to a certain extent. In the future, we hope to explore the influence of other possible faults on model quality enhancement. So far, our approach mainly focuses on image and structural (tabular) data with feed-forward neural networks (e.g., CNN and MLP), we hope to extend experiments to the text/NLP domain where input features are word vectors and recurrent neural networks (RNN) are more used. There are several hyper-parameters involved in the used fuzzing and generation algorithms, and we manually set them following existing works or through simple ablation studies, which may not be the optimal setting. In the future, we plan to investigate automatic parameter tuning.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (Grant No. 2020YFB2010901), the Key R&D Program of Zhejiang (Grant No. 2022C01018), the NSFC Program (Grant No. 62102359, 61833015, 62293511, U1911401) and the Fundamental Research Funds for Central Universities (Zhejiang University NGICS Platform).

Just Accepted

REFERENCES

[1] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 625–635.

[2] Haldun Akoglu. 2018. User's guide to correlation coefficients. *Turkish journal of emergency medicine* 18, 3 (2018), 91–93.

[3] Mohammed Oualid Attaoui, Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2022. Black-box Safety Analysis and Retraining of DNNs based on Feature Extraction and Clustering. *arXiv preprint arXiv:2201.05077* (2022).

[4] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 1–4.

[5] Neal E Boudette. 2017. Tesla's self-driving system cleared in deadly crash. *The New York Times* 19 (2017).

[6] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. 2019. Accurate, reliable and fast robustness evaluation. In *Advances in Neural Information Processing Systems*. 12861–12871.

[7] Taejoon Byun, Vaibhav Sharma, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. 2019. Input prioritization for testing neural networks. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 63–70.

[8] Cristian Cadar, Daniel Dunbar, Dawson R Engler, et al. 2008. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs.. In *OSDI*, Vol. 8. 209–224.

[9] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[10] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. IEEE, 39–57.

[11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12, ARTICLE (2011), 2493–2537.

[12] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[13] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jinsong Dong, and Ting Dai. 2020. An Empirical Study on Correlation between Coverage and Robustness for Deep Neural Networks. In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 73–82.

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[15] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A Kai Qin, and Yun Yang. 2020. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1000–1008.

[16] Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2022. HUDD: A tool to debug DNNs for safety analysis. In *2022 IEEE/ACM 44st International Conference on Software Engineering*.

[17] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual Event, USA) *(ISSTA 2020)*. Association for Computing Machinery, New York, NY, USA, 177–188. https://doi.org/10.1145/3395363.3397357

[18] Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. 2019. Adversarial attacks on medical machine learning. *Science* 363, 6433 (2019), 1287–1289.

[19] 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*. 498–510.
[20] Patrice Godefroid, Adam Kiezun, and Michael Y Levin. 2008. Grammar-based whitebox fuzzing. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 206–215.
[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[22] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6572
[23] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 6645–6649.
[24] Loren Grush. 2015. Google engineer apologizes after Photos app tags two black people as gorillas. *The Verge* 1 (2015).
[25] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 739–743.
[26] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. 2020. Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?. In *Proceedings of the Joint Meeting on Foundations of Software Engineering (FSE)*.
[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[28] Qiang Hu, Yuejun Guo*, Maxime Cordy, Xiaofei Xie, Lei Ma, Mike Papadakis, and Yves Le Traon. 2022. An empirical study on data distribution-aware test selection for deep learning enhancement. *ACM Transactions on Software Engineering and Methodology* (2022).
[29] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 3–29.
[30] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2020. Taxonomy of real faults in deep learning systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1110–1121.
[31] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. 2019. Limitations of the Lipschitz Constant as a Defense Against Adversarial Examples. In *ECML PKDD 2018 Workshops*, Carlos Alzate, Anna Monreale, Haytham Assem, Albert Bifet, Teodora Sandra Buda, Bora Caglayan, Brett Drury, Eva García-Martín, Ricard Gavaldà, Irena Koprinska, Stefan Kramer, Niklas Lavesson, Michael Madden, Ian Molloy, Maria-Irina Nicolae, and Mathieu Sinn (Eds.). Springer International Publishing, Cham, 16–29.
[32] Kimmo Karkkainen and Jungseock Joo. 2021. Fairface: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1548–1558.
[33] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Towards proving the adversarial robustness of deep neural networks. *arXiv preprint arXiv:1709.02802* (2017).
[34] Marc Khoury and Dylan Hadfield-Menell. 2019. Adversarial training with voronoi constraints. *arXiv preprint arXiv:1905.01019* (2019).
[35] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1039–1049.
[36] Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *Kdd*, Vol. 96. 202–207.
[37] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[38] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.
[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[40] Seokhyun Lee, Sooyoung Cha, Dain Lee, and Hakjoo Oh. 2020. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 165–176.
[41] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. 2011. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 163–168.
[42] Yu Li, Muxi Chen, and Qiang Xu. 2022. HybridRepair: towards annotation-efficient repair for deep learning models. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 227–238.
[43] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. 2019. Structural coverage criteria for neural networks could be misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 89–92.
[44] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. 2019. Deepct: Tomographic combinatorial testing for deep learning systems. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 614–618.
[45] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 120–131.

Just Accepted

[46] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–22.
[47] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. 2020. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognition* (2020), 107332.
[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJzIBfZAb
[49] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62 (2014), 22–31.
[50] Kevin P. Murphy. 2012. *Machine learning - a probabilistic perspective*. MIT Press.
[51] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning. (2011).
[52] Carlos Pacheco and Michael D Ernst. 2007. Randoop: feedback-directed random testing for Java. In *Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*. 815–816.
[53] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.
[54] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. (2015).
[55] 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.
[56] Vincenzo Riccio, Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. DeepMetis: Augmenting a Deep Learning Test Set to Increase its Mutation Score. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 355–367.
[57] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering* 25, 6 (2020), 5193–5254.
[58] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 2 (2009), 131–164.
[59] Patrick Schober, Christa Boer, and Lothar A Schwarte. 2018. Correlation coefficients: appropriate use and interpretation. *Anesthesia & Analgesia* 126, 5 (2018), 1763–1768.
[60] Ali Shahrokni and Robert Feldt. 2013. A systematic review of software robustness. *Information and Software Technology* 55, 1 (2013), 1–17.
[61] Weijun Shen, Yanhui Li, Lin Chen, Yuanlei Han, Yuming Zhou, and Baowen Xu. 2020. Multiple-boundary clustering and prioritization to promote neural network retraining. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 410–422.
[62] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[63] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.
[64] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic Testing for Deep Neural Networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (Montpellier, France) *(ASE 2018)*. Association for Computing Machinery, New York, NY, USA, 109–119. https://doi.org/10.1145/3238147.3238172
[65] Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.
[66] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314.
[67] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T Johnson. 2019. Star-based reachability analysis of deep neural networks. In *International Symposium on Formal Methods*. Springer, 670–686.
[68] 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*. 98–108.
[69] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. RobOT: Robustness-oriented testing for deep learning systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 300–311.
[70] 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*. IEEE Press, 1245–1256.
[71] Mei Wang, Weihong Deng, Jiani Hu, Xunqiang Tao, and Yaohai Huang. 2019. Racial faces in the wild: Reducing racial bias by information maximization adaptation network. In *Proceedings of the ieee/cvf international conference on computer vision*. 692–702.
[72] Xinyu Wang, Jun Sun, Zhenbang Chen, Peixin Zhang, Jingyi Wang, and Yun Lin. 2018. Towards optimal concolic testing. In *Proceedings of the 40th International Conference on Software Engineering*. 291–302.
[73] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. 2019. On the Convergence and Robustness of Adversarial Training. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning*

Just Accepted

Research, Vol. 97), Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 6586–6595. http://proceedings.mlr.press/v97/wang19i.html
[74] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.
[75] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 397–409.
[76] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. *arXiv preprint arXiv:1804.09699* (2018).
[77] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. 2018. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BkUHlMZ0b
[78] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. 2018. Feature-guided black-box safety testing of deep neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 408–426.
[79] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.
[80] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
[81] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Beijing, China) *(ISSTA 2019)*. Association for Computing Machinery, New York, NY, USA, 146–157. https://doi.org/10.1145/3293882.3330579
[82] Huan Xu and Shie Mannor. 2012. Robustness and generalization. *Machine learning* 86, 3 (2012), 391–423.
[83] Pengfei Yang, Renjue Li, Jianlin Li, Cheng-Chao Huang, Jingyi Wang, Jun Sun, Bai Xue, and Lijun Zhang. 2020. Improving Neural Network Verification through Spurious Region Guided Refinement. *arXiv preprint arXiv:2010.07722* (2020).
[84] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. 2021. Deeprepair: Style-guided repairing for deep neural networks in the real-world operational environment. *IEEE Transactions on Reliability* (2021).
[85] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*. PMLR, 7472–7482.
[86] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).
[87] 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 the ACM/IEEE 42nd International Conference on Software Engineering*. 949–960.
[88] Peixin Zhang, Jingyi Wang, Jun Sun, and Xinyu Wang. 2021. Fairness Testing of Deep Image Classification with Adequacy Metrics. *arXiv preprint arXiv:2111.08856* (2021).
[89] Peixin Zhang, Jingyi Wang, Jun Sun, Xinyu Wang, Guoliang Dong, Xingen Wang, Ting Dai, and Jin Song Dong. 2021. Automatic Fairness Testing of Neural Classifiers through Adversarial Sampling. *IEEE Transactions on Software Engineering* (2021).
[90] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

A ADDITIONAL FIGURES

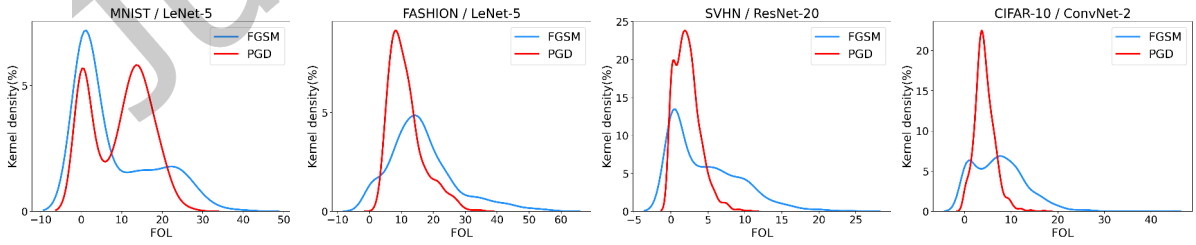

Fig. 14. The FOL distribution of FGSM and PGD attacks for different models.

Just Accepted

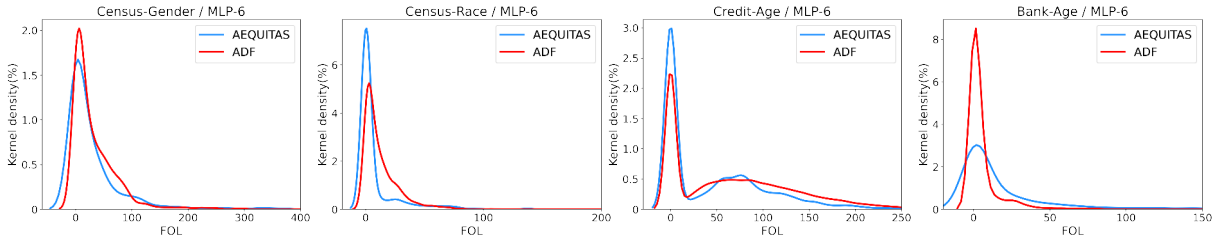

Fig. 15. The FOL distribution of AEQUITAS and ADF testing algorithms for different models.

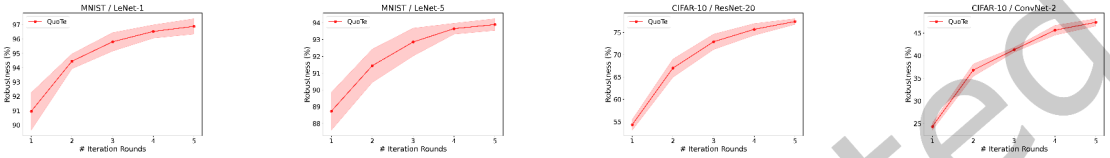

Fig. 16. The trend of robustness improvement (ATTACK) in iterations on the MNIST and CIFAR-10 datasets (with confidence interval of 95% significance level).

Just Accepted