

Received November 18, 2019, accepted December 17, 2019, date of publication December 27, 2019, date of current version January 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962695

ARTDL: Adaptive Random Testing for Deep Learning Systems

MIN YAN¹, LI WANG², (Senior Member, IEEE), AND AIGUO FEI¹

¹School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

²School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Li Wang (liwang@bupt.edu.cn)


This work was supported in part by the National Natural Science Foundation of China under Grant 61871416, in part by the Beijing Science and Technology Nova Program under Grant xx2018083, in part by the Fundamental Research Funds for the Central Universities under Grant 2018XKJC03, in part by the Beijing Municipal Natural Science Foundation under Grant L172010, and in part by the BUPT Excellent Ph.D. Students Foundation under Grant CX2019114.

ABSTRACT With recent breakthroughs in Deep Learning (DL), DL systems are increasingly deployed in safety-critical fields. Hence, some software testing methods are required to ensure the reliability and safety of DL systems. Since the rules of DL systems are inferred from training data, it is difficult to know the implementation rules about each behavior of DL systems. At the same time, Random Testing (RT) is a popular testing method and the knowledge about software implementation is not needed when we use RT. Therefore, RT is very suitable for the testing of DL systems. And the existing mechanisms for testing DL systems also depend heavily on RT by the labeled test data. In order to increase the effectiveness of RT for DL systems, we design, implement and evaluate the Adaptive Random Testing for DL systems (ARTDL), which is the first Adaptive Random Testing (ART) method to improve the effectiveness of RT for DL systems. ARTDL refers to the idea of ART. That is, fewer test cases are needed to detect failures by selecting the test case with the furthest distance from non-failure-causing test cases. Firstly, we propose the Feature-based Euclidean Distance (FED) as the distance metric that can be used to measure the difference between failure-causing inputs and non-failure-causing inputs. Secondly, we verify the availability of FED by presenting the failure pattern of DL models. Finally, we design ARTDL algorithm to generate the test cases that are more likely to cause failures based on the FED. We implement ARTDL to test top performing DL models in the field of image classification and automatic driving. The results show that, on average, the number of test cases used to find the first bug is reduced by 62.74% through ARTDL, compared with RT.

INDEX TERMS Deep learning testing, adaptive random testing, distance metric, metamorphic testing.

I. INTRODUCTION

In the past few years, Deep Learning (DL) systems have demonstrated amazing performance in various domains such as image classification [1], [2], speech recognition [3], and playing games [4]. Based on these advances, DL systems are increasingly deployed in safety-critical fields such as autonomous vehicles [5], medical diagnostics [6] and aircraft collision avoidance [7]. Unfortunately, DL systems, despite their significant capabilities, are known to demonstrate unexpected or incorrect behaviors. Specially, such incorrect behaviors can lead to some fatal crashes when DL systems are deployed in safety-critical domain [8], [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Minh Jo .

This naturally raises an urgent need to test and verify behaviors of DL systems to ensure their reliability and safety in the real-world deployment.

Random Testing (RT), as one of the most fundamental and popular software testing methods, is a possible strategy to test DL systems. The benefit of RT is simple in concept and easy to implement [10]. Most importantly, it may be the only practically feasible technique if we have no knowledge about the implementation of software [11]. Obviously, this characteristic also is an important benefit for the testing of DL systems because of the shift of development paradigm induced by DL [12]. Traditional software is constructed deductively by writing down the rules as program code and the behavior of system can be governed by these known rules. However, for DL systems, the implementation rules are inferred inductively

from training data and these rules are not exactly known to us. The paradigm shift makes DL systems do not have source code corresponding to some of their critical behaviors. Therefore, RT is a very suitable test method for DL systems. In practice, the standard approach for testing DL systems is to gather and manually label as much test data as possible and these test data is used to test DL systems by RT method [13], [14].

Furthermore, in order to increase the effectiveness of RT for DL systems while maintaining the benefits of RT, we propose a new test method denoted as Adaptive Random Testing for DL systems (ARTDL). ARTDL refers to the basic idea of the Adaptive Random Testing (ART) [15] which is proposed as an enhancement to RT. ART is based on the intuition that fewer test cases are needed to detect failures by analyzing the failure pattern. Failure pattern is the distribution and geometry of the failure-causing inputs in the input space [11]. Existing empirical observations show that failure-causing inputs of many numerical programs tend to form contiguous failure regions [16]. Based on such failure pattern, the test case with the furthest distance from the executed non-failure-causing test cases is more likely to cause failure behavior.

However, a major challenge will arise on how to define the distance metric that can be used to measure the difference between failure-causing inputs and non-failure-causing inputs of DL systems, if we want to improve the effectiveness of RT for DL systems by ART method. In this paper, we focus on the DL systems where the inputs are images. Existing distance metrics of ART methods include intuitive distance of numerical values [15], object distance for object-oriented software [17] and the distance based on the concepts of categories and choices [18], [19], [20]. However, these distance metrics cannot be directly applied to DL systems. Therefore, it is necessary to propose a new distance metric so that ART can be applied to test DL systems.

Given above discussions, in this paper, we design, implement and evaluate an automated black-box test method for DL systems: ARTDL, which can be used to improve the effectiveness of RT. Firstly, we define the Feature-based Euclidean Distance (FED) as the distance metric to measure the difference between failure-causing inputs and non-failure-causing inputs. FED measures the difference between inputs by calculating the distance between the feature vectors extracted from input images by VGGNet [21]. Secondly, we present the failure pattern of DL models by visualizing feature vectors of failure-causing inputs and non-failure-causing inputs in the three-dimension space. The results verify that FED can be used to estimate the difference between failure-causing inputs and non-failure-causing inputs. Therefore, it is feasible that the ART is used to test DL models based on the FED. Finally, based on the FED, we design the ARTDL algorithm to generate the test cases that are more likely to cause failure. We implement and evaluate ARTDL to test four DL models using two real-world datasets including MNIST [22] and Udacity self-driving car challenge data [23]. The results show that ARTDL can improve the effectiveness of RT. The main

contribution of this paper is that we propose, to the best of our knowledge, the first ART method for DL systems, denoted as ARTDL. We implement and evaluate the ARTDL,¹ on average, which reduces 62.74% of required test cases.

II. RELATED WORK

A. ADAPTIVE RANDOM TESTING

Adaptive Random Testing (ART) is an attempt to improve the failure-detection effectiveness of RT [11], [30]. The first ART was proposed aiming at numerical program [15]. ARTOO [17] proposed the object distance for object-oriented applications. Kuo *et al.* [18] and Merkel *et al.* [19] proposed a difference measure that can be applied to a broad range of software input types, based on the concepts of categories and choices. Furthermore, a linear-order ART algorithm was proposed [20], which takes advantage of the properties of the distance measure that is based on the concepts of categories and choices to achieve linear-time test case selection overhead. However, existing ART methods cannot be directly applied for DL systems.

B. TESTING OF DEEP LEARNING

With the tremendous progress of DL, the concerns about trustworthiness of DL systems are aroused and many researchers focus on the techniques for testing DL-enable systems [31], [32]. The testing methods of DL systems include white-box testing and black-box testing. As shown in Table 1, we summarized the related work and compare ARTDL with others. Some coverage-guided white-box testing methods towards DL system have been proposed to guide the generation of inputs. Specifically, the DeepXplore [24] first proposed neuron coverage criteria to drive test generation. More failures of DL models will be exposed by increasing the neuron coverage of test inputs. DLFuzz [25] overcame the trouble that DeepXplore needs to rely on multiple DL systems of the similar functionality by combining the fuzzing testing and neuron coverage. DeepGauge [26] proposed a set of testing criteria based on multi-level and multi-granularity coverage for testing Deep Neural Networks (DNN). DeepTest [27] maximized the neuron coverage by applying different realistic transformations on a set of seed images to find many erroneous behaviors of the DL-based autonomous vehicles. Different from these criteria that only consider the activation of individual neuron, Sun *et al.* [28] proposed MC/DC coverage criterion for DL, which took into account the causal relation between features in DNNs. However, Li *et al.* [33] argued that these proposed structural coverage criteria for DL systems could be misleading and they suggested that reported fault-detection “capabilities” conjectured from high coverage testing are more likely due to the adversary-oriented search but not the real “high” coverage. Overall, although these white-box testing methods play an important role in promoting the research of DL system testing in infancy, it is not clear whether these criteria are directly related to the

¹<https://github.com/yan-min/ARTDL>

TABLE 1. The summary of related software testing methods for DL systems.

Type	Method	The internal implementation can be unknown	Guided method is used to improve the effectiveness of testing
white-box	DeepXplore [24]		✓
	DLFuzz [25]		✓
	DeepGauge [26]		✓
	DeepTest [27]		✓
	Sun et al. [28]		✓
black-box	Random Testing [13], [29]	✓	
	ARTDL	✓	✓

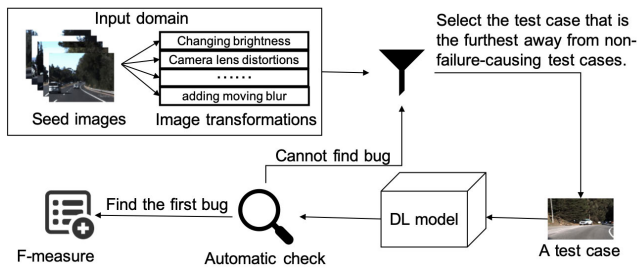


FIGURE 1. The workflow of testing a DL model through the ARTDL.

system decision logic, due to the black-box nature of a DL system.

Besides, black-box testing methods are also needed, which test from the functional level, without knowing the internal implementation of the software. In practice, black-box testing methods and white-box testing methods are often used together to ensure the reliability and safety of the software. The RT, as a popular black-box software testing method, has been widely used to evaluate DL systems by measuring their accuracy on test cases randomly drawn from manually labeled datasets [13] and ad hoc simulations [29]. Furthermore, ART as an enhancement to RT is often used to improve the effectiveness of RT [11]. In this paper, we are interested in the problem that when RT has been chosen as a viable testing method for a DL system, is it worthwhile to use ART instead and how to apply ART in DL systems?

III. METHODOLOGY

A. OVERVIEW

In this subsection, we provide an overview for the testing of DL systems by the ARTDL. The workflow of the testing is shown in Figure 1. Firstly, ARTDL takes the combination of seed images and image transformations as the input domain of DL systems. Seed images are small quantity of labeled test data. Image transformations are different ways to add noises to seed images, which also exist in practice (e.g., changing brightness, camera lens distortions, adding moving blur, etc.). Secondly, ARTDL selects the test case that is more likely to detect failure based on the idea of ART. Finally, we estimate the performance of ARTDL by the metric: F-measure, which is used to evaluate the effectiveness of testing method in the field of software testing and it is

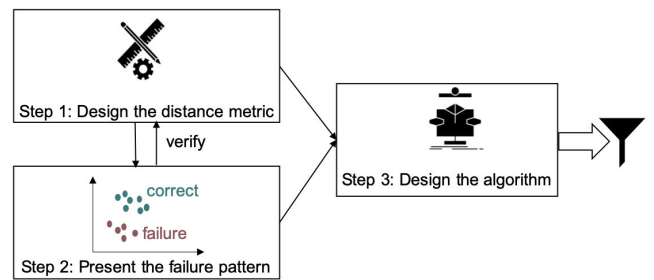


FIGURE 2. Overview of the ARTDL method.

defined as the expected number of test cases generated until the first fault is detected [15]. We leverage metamorphic relations [34] to automatically decide whether the output of the program is correct, which will be described in detail in the Section III.D.

Importantly, the ARTDL should select the test cases that are more likely to detect failure from input domain. In our method, as shown in Figure 2, three steps are required to achieve this goal. Firstly, an appropriate distance metric should be defined to reflect the difference between non-failure-causing inputs and failure-causing inputs. Secondly, the failure pattern of DL systems should be presented to verify the availability of proposed distance metric. Finally, based on the distance metric and the failure pattern, an algorithm should be designed to generate the test cases that are more likely to detect failures. These three steps will be described in detail in following subsections.

B. DISTANCE METRIC

In this paper, we define the Feature-based Euclidean Distance (FED) as the distance metric to measure the distance between inputs. FED aims to estimate the difference between failure-causing inputs and non-failure-causing inputs and make failure-causing inputs far away from non-failure-causing. Empirical observations showed that failure-causing inputs of numeric programs tend to form contiguous failure regions [16], since adjacent numeric inputs are likely to result in similar computations. It is known that DNNs of DL model have amazing ability to automatically identify and extract the relevant features from raw inputs without any human guidance besides labeled training data [35]. It is critical that DNN can automatically extract features from the input

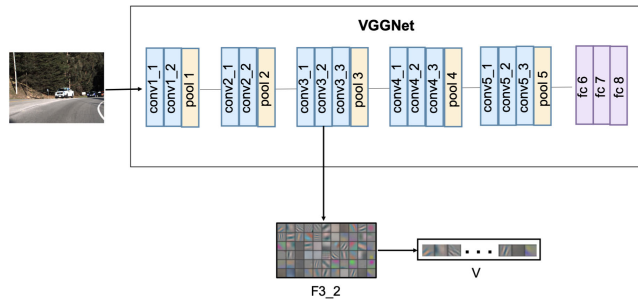


FIGURE 3. The workflow for extracting the features of the convolutional layer conv3_2 of VGGNet from an input image.

images to obtain the output. Hence, in the DL systems of image classification and image-based autonomous vehicles, adjacent features extracted from input images are likely to result in similar output. Therefore, the difference between failure-causing inputs and non-failure-causing inputs can be measured by the distance between the features extracted from images instead of measuring the distance between images directly.

Empirical research on visualization of the extracted features of Convolutional Neural Network (CNN) has shown that the features extracted from each layer represent the hierarchical nature of the features in the network [36]. The features of low layer represent corners and other edge/color conjunctions. The mid layer capture similar features of textures and background. The features of high layer represent entire objects with significant pose variation and are more class-specific [36]. However, we don't know which features can better distinguish the difference between failure-causing inputs and non-failure-causing inputs. In this paper, we find the features that can be used to distinguish the difference between failure-causing inputs and non-failure-causing inputs through the experiments of testing multiple models on different datasets. The results will be described in detail in Section V.

To be specific, as shown in Figure 3, we extract the feature from the image by VGGNet [21], which is a widely used DNN to extract features from each image. We take the feature of convolutional layer conv3_2 of VGGNet as an example. For one image, we can get a feature matrix F_{3_2} with size (x_i, y_i) by the convolutional layer conv3_2, where x_i and y_i are the width and height of the feature maps respectively. The feature vector of the given image is V that is the vector flattened from feature matrix F_{3_2} . In summary, the process of extracting features is defined as follows:

$$V_p = h(I_p), \tag{1}$$

where I_p denotes a input image, $h(\cdot)$ denotes the method of extracting features as mentioned above, and V_p denotes the feature extracted from I_p by conv3_2 layer of VGGNet.

Based on the feature extracted by the convolutional layer conv3_2 of VGGNet, we propose FED as the distance metric to measure the distance between features of input images through the Euclidean Distance (ED). Specifically, the feature

vectors V_p and V_q are represented as $V_p = \{p_1, p_2, \dots, p_m\}$ and $V_q = \{q_1, q_2, \dots, q_m\}$ respectively. FED is defined as follows:

$$\begin{aligned} FED(I_p, I_q) &= ED(h(I_p), h(I_q)) \\ &= ED(V_p, V_q) \\ &= \sqrt{\sum_{i=1}^m (p_i - q_i)^2}, \end{aligned} \tag{2}$$

where I_p and I_q denote two images respectively, V_p and V_q denote two feature vectors extracted from I_p and I_q respectively, $ED(\cdot, \cdot)$ denotes the Euclidean Distance between two vectors.

C. FAILURE PATTERN

Failure pattern is the distribution and geometry of the failure-causing inputs. Essentially, the goal of testing is to generate test cases with a view to maximize the number of failures revealed. In order to achieve this goal, the distribution and geometry of the failure-causing inputs should be considered. Empirical studies have shown that the failure patterns in numerical software are categorized into three patterns [37]: block pattern, strip pattern, and point pattern. The block pattern and the strip pattern denote that the failure regions of input domain is contiguous and the ART method can be used to improve the performance of RT. Therefore, the failure pattern of DL models should be presented to determine whether ART can be used for the testing of DL models.

In this paper, we present the failure pattern of DL models by Principle Component Analysis (PCA) technique [38]. PCA is a dimension-reduction tool that can be used to reduce a large set of variables to a small set that still contains most of the information of the large set. We apply PCA to reduce the dimension of feature vector extracted from input image by VGGNet as follows:

$$L_m = V_m \times P_m, \tag{3}$$

where V_m denotes a k -dimension feature vector, P_m denotes the projection matrix with size (k, l) , and L_m denotes a target l -dimension vector. We set l as 3 and the feature vector of image can be shown in three-dimension space. Through this way, the distribution of failure-causing inputs and non-failure-causing inputs can be showed intuitively in three-dimension space.

D. ARTDL ALGORITHM

We design the ARTDL algorithm to generate test cases that are more likely to cause failures for DL systems. The most popular existing ART algorithm is Fixed-Sized-Candidate-Set ART (FSCS-ART) [15]. In FSCS-ART algorithm, a fixed-size candidate set of random inputs is first generated whenever a new test case is needed. For each candidate set, the test case with the furthest distance from non-failure-causing test cases will be selected as a new test case. The ARTDL algorithm shares the idea of fixed-sized candidate

Algorithm 1 ARTDL**Input:**

executed_data: executed non-failure-causing inputs
input_data: the set of combination of all seed images and image transformations
DL: the DL model under test

Output:

F : F-measure

```

1: // main procedure
2: initial  $F := 0$ , counter := 0
3: reveal_failure := False
4: while not reveal_failure do
5:   executed_set = random sample  $m$  data from
   executed_data
6:   candidate_set = random sample  $n$  data from
   input_data
7:   test_data = SELECT_BEST_TEST_DATA
   (executed_set, candidate_set)
8:   if DL(test_data) is incorrect then
9:     reveal_failure := True
10:  else
11:    executed_set = executed_data union test_data
12:    counter = counter + 1
13:  end if
14:  input_set.remove(test_data)
15: end while
16: return  $F$ ;
17: // utility function for select best test case
18: function SELECT_BEST_TEST_DATA(executed_set,
   candidate_set)
19:   best_distance :=  $-1.0$ 
20:   for each Image in candidate_set do
21:     dis = average of FED(Image, elements in exe-
   cuted_set)
22:     if best_distance < dis then
23:       best_data = data
24:       best_distance = dis
25:     end if
26:   end for
27: end function

```

set of FSCS-ART. As shown in Algorithm 1, the inputs of ARTDL algorithm include the DL model under test and two sets of data including executed data and input data. Executed data include the input that has been tested and it doesn't cause failure. Input data includes all possible inputs that are generated by image transformations of seed images. Firstly, we get the candidate set by sampling n data samples from input data and the executed set is obtained by sampling m data samples from executed data. Secondly, ARTDL selects the best test case from candidate set by comparing the FED between the test data of candidate set and the test data of executed set (Algorithm 1 line 18-28). Specially, we calculate the average

value of FED between the test data and elements of executed set, which is different from the calculated minimum in the original FSCS-ART. The test data with the largest FED value will be the best test case. Finally, the best test case is fed into the DL model to determine whether the test case can expose the failure of the model. Repeat above steps until the first failure is detected. After the first failure is detected, we get the value of F-measure of ARTDL algorithm, which can be used to estimate the effectiveness of ARTDL method. F-measure is defined as the expected number of test cases generated until the first fault is detected [15], which reflects the effectiveness of a testing method.

Besides, there is an oracle problem [39] in testing DL systems, that is, how to automatically determine whether the software program performs as expected upon the given test inputs (Algorithm 1 line 8). Many studies have shown that metamorphic testing is a popular solution to this problem [34], [40] and it has been widely used in the testing of self-driving systems. DeepTest [27] tested three different DNN models for autonomous driving by a set of metamorphic relations based on image transformations. Zhou and Sun [41], [42] introduced an innovative testing strategy that combines MT with fuzzing and detected previously unknown fatal errors in the real-life LiDAR Obstacle Perception system of the Baidu Apollo self-driving software. The key to metamorphic testing is a set of metamorphic relations [43], which refer to the relations among multiple inputs and their expected outputs. In this paper, we define metamorphic relations by different image transformations and leverage metamorphic relations to automatically decide whether the output of the program is correct. For example, the results of handwritten numeral recognition should not be changed for the same image under any lighting conditions, blurring, or any affine transformations with small parameter values.

IV. IMPLEMENTATION

We implement ARTDL using TensorFlow 1.12.0 [44] and Keras 1.2.2 [45] DL frameworks. All the experiments are run on the Ubuntu 16.04 (Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 16 cores, 32GB of memory)

A. TEST DATASETS AND DL MODELS

We adapt two popular public datasets including MNIST and Driving to evaluate ARTDL. For each dataset, we evaluate ARTDL on two different DL models. We provide a summary of the two datasets and the corresponding DL models in Table 1. All the evaluated DL models are pre-trained (i.e., we use public weights reported by previous researchers).

MNIST [22] is a dataset for handwritten digits recognition, which contains 28×28 pixel images with class labels from 0 to 9. The dataset includes 70,000 input data in total, of which 60,000 are training data and 10,000 are test data. On MNIST, two different DL models (LeNet-1, LeNet-4) based on the LeNet family [46], [47] are used to test.

Driving is the dataset released by Udacity self-driving car challenge [23]. The dataset has 5614 labeled testing samples.

TABLE 2. Details of DL models and datasets used to evaluate the ARTDL.

Dataset	Dataset Description	DL models
MNIST	Hand-written digits	LeNet-1 [46], [47]
		LeNet-4 [46], [47]
Driving	Driving video frames	Rambo [48]
		Autumn [49]

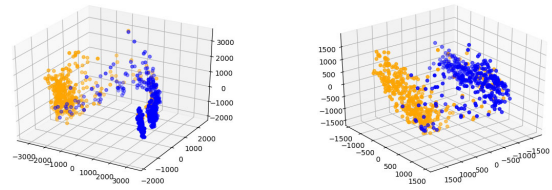
TABLE 3. Image transformations and parameters used in ARTDL and the details of these transformations can be found in [27].

Transformation	Parameters	Parameter ranges	
Translation	(x, y)	(10, 10) to (100, 100) with step (10, 10)	
Brightness	bias	1 to 10 with step 1	
Blur	Moving	(degree, angle)	(1, 45) to (12, 45) with step (1, 0)
	Averaging	Kernel size	$3 \times 3, 4 \times 4, 5 \times 5$
	Gaussian	Kernel size	$3 \times 3, 4 \times 4, 5 \times 5$
	Median	aperture linear size	3, 5
	Bilateral	diameter, sigma-Color, sigmaS-space	9, 75, 75

Each sample contains an image and the corresponding labeled steering angle. The images were captured by a camera mounted behind the windshield of a driving car and the simultaneous steering wheel angles were obtained by the human driver for each image. We evaluate ARTDL on top performing DL models released by Udacity driving challenge: Rambo [48] and Autumn [49]. Rambo model is consisted of three CNNs whose outputs are merged at the final layer [48]. One of the CNNs is inspired by comma.ai's steering model [50], and two of the CNNs are inspired by the NVIDIA'S self-driving architecture [5]. Autumn model is composed of a data preprocessing process and a CNN model [49]. Specifically, Autumn computes the optical flow of images, which is fed into the CNN model. The architecture of the CNN model includes three 5×5 conv layers with stride 2, plus two 3×3 conv layers and followed by five fully-connected layers with dropout.

B. IMAGE TRANSFORMATIONS.

Generating input images that are likely to cause fault may be very useful if the inputs seem to appear in the real-world. Therefore, we focus on generating realistic synthetic images by applying image transformations [27] on seed images and mimic different real-world phenomena like camera lens distortions, object movements, different brightness, etc. In this paper, we define eight realistic image transformations including changing brightness, translation, scaling and blurring, where the blurring includes moving blurring and four different types of blurring filters: averaging, Gaussian, median, and bilateral [29]. The detail descriptions of parameters for each transformation are shown in Table 2. We implement these image transformations by OpenCV [51].



(a) conv3-2

(b) conv4-2

FIGURE 4. Results of the failure pattern of Driving dataset, which are obtained by the features of conv3-2 and conv4-2 layer of VGGNet respectively. The yellow dots represent the non-failure-causing inputs and the blue dots represent the failure-causing inputs.

V. RESULTS

A. FAILURE PATTERN

In this subsection, we present the failure pattern of DL models and verify that the FED can be used to estimate the difference between failure-causing inputs and non-failure-causing inputs. In order to present the failure pattern of DL models, we have to get the non-failure-causing input set and the failure-causing input set. We generate synthetic images by transforming seed images. The failure patterns are obtained by the transformed test cases. These transformed test cases are fed into the DL model and we judge whether the output of the model is correct by metamorphic relation [34]. For the Driving dataset, after a synthetic image is fed into the DL model, if the difference between the output of model and the output of corresponding seed image is less than the parameter δ , the synthetic image belongs to the non-failure-causing input set. Otherwise, it belongs to the failure-causing input set. For the MNIST dataset, if the output of synthetic image is different from the output of corresponding seed image, the synthetic image belongs to the failure-causing input set. Otherwise, it belongs to the non-failure-causing input set.

In order to verify which feature of convolutional layer of VGGNet is the most suitable feature to distinguish the difference between failure-causing inputs and non-failure-causing inputs, we present the failure patterns obtained from each convolutional layer of VGGNet. Firstly, we get the feature vectors of input images by every convolutional layer respectively. Then, we apply PCA to reduce the dimension of feature vectors and visualize them in three-dimension space. To reduce the computational complexity, we random sample 400 input images from failure-causing input set and non-failure-causing input set respectively. As shown in Figure 4, when the failure pattern is obtained by the feature of conv3_2 or conv4_2, non-failure-causing inputs are in contiguous areas and absolute majority failure-causing inputs are far away from non-failure-causing inputs although minority failure-causing inputs are close to the non-failure-causing inputs. Besides, other results of failure pattern are shown in Figure 5. From Figure 5(a), we present the failure pattern that is obtained directly by images instead of the feature extracted from images by VGGNet. That is, the pixel vectors of image are directly visualized in three-dimension

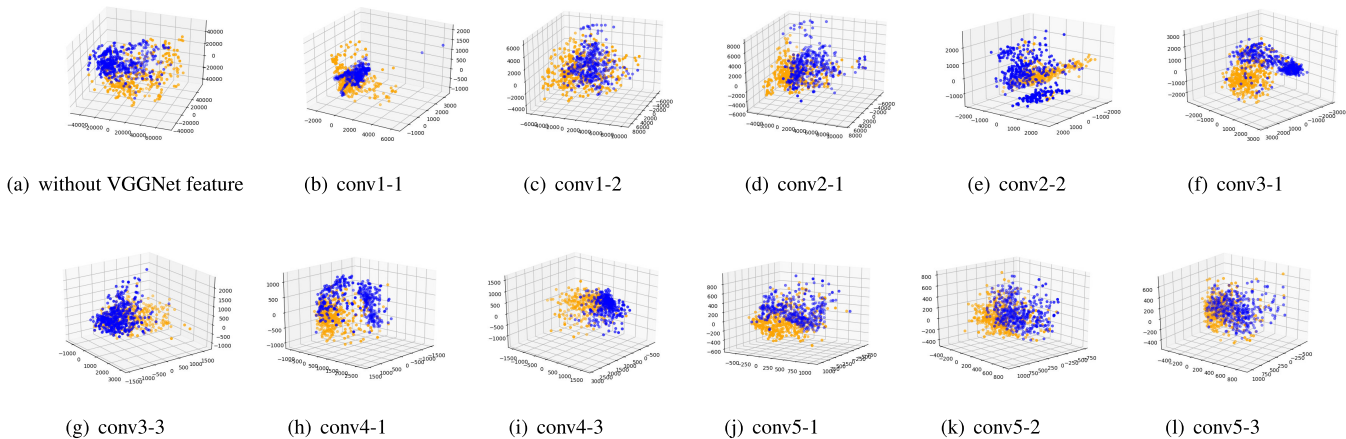


FIGURE 5. Results of the failure pattern of Driving dataset. (a) shows the failure pattern that is directly obtained by input images and the failure patterns obtained by the features of each convolutional layer of VGGNet (except for conv3-2 and conv4-2) are presented from (b) to (l). The yellow dots represent the non-failure-causing inputs and the blue dots represent the failure-causing inputs.

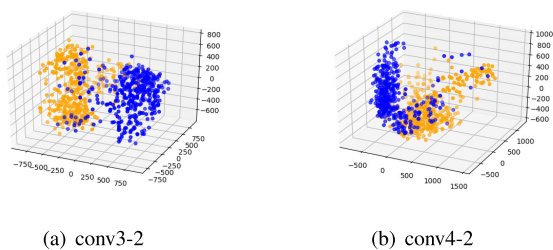


FIGURE 6. Results of the failure pattern of MNIST dataset, which are obtained by the features of conv3-2 and conv4-2 layer of VGGNet respectively. The yellow dots represent the non-failure-causing inputs and the blue dots represent the failure-causing inputs.

space by the PCA technical. From Figure 5(b) to Figure 5(l), we present the failure patterns obtained by the feature extracted by each convolutional layer of VGGNet as shown in Figure 2 (except for conv3-2 and conv4-2). The results show that the failure patterns obtained by these methods can not distinguish the difference between failure-causing inputs and non-failure-causing inputs. Furthermore, we also present the failure pattern of MNIST dataset. Since MNIST dataset is used for classification, we calculate the distance for the inputs of same classification. In this experiment, we take classification 1 as an example. The results of the failure pattern of MNIST dataset, as shown in Figure 6 and 7, are similar with the results of Driving dataset.

Therefore, the results of Driving dataset and MNIST dataset all show that the feature of convolutional layer conv3_2 and conv4_2 of VGGNet are the suitable features to distinguish the difference between failure-causing inputs and non-failure-causing inputs. The reason may be the convolutional layer conv3_2 and conv4_2 of VGGNet can capture the features which causes failures of DL model. Empirical research on visualization of the extracted features of CNN has shown that the features extracted from each layer show the hierarchical nature of the features in the network and the mid layer capture similar features of textures and background [36]. At the same time, in the DL-based

autonomous vehicles, some fatal crashes happened because they were under unseen driving environment, such as rain fall [8] and bright sky [9], which is related to the features of textures and background captured by the mid layer. So, the features extracted by mid layer, such as conv3-2 and conv4-2, may be related to the failures of DL models.

In conclusion, when we use the feature of conv3_2 and conv4_2 as the metric to present the failure pattern of DL model, the results show that non-failure-causing inputs are in contiguous areas and absolute majority failure-causing inputs are far away from non-failure-causing inputs although minority failure-causing inputs are close to the non-failure-causing inputs. Therefore, the FED is available to estimate the difference between non-failure-causing inputs and failure-causing inputs. And we can improve the effectiveness of RT for DL models by ART method based on the FED.

B. THE PERFORMANCE OF ARTDL

We evaluate the performance of ARTDL with the metrics: F-measure, which is the expected number of test cases generated until the first failure is detected. It reflects the effectiveness of a testing method and has been the most frequently used metric to compare the effectiveness of ART and RT [11]. In practice, when a failure is detected, testing is normally stopped and debugging starts. Hence, the F-measure is not only more intuitively appealing but also more realistic from a practical perspective. In order to get the value of F-measure, we select the test case by ARTDL or RT method to test the DL model until the first failure is detected and we repeat this process many times. In our experiment, we set the values of m and n to 10 in the ARTDL algorithm. Then, we will get a set of the value of F-measure $M = \{u_1, u_2, \dots\}$. The central limit theorem [52] is used to determine when the process could be stopped so that we have enough samples to provide reliable statistic estimates.

Suppose we want to estimate the F-measure with an accuracy of $\pm r\%$ and a confidence level of $(1 - \alpha) \times 100\%$, where $1 - \alpha$ is the confidence coefficient. In order to achieve

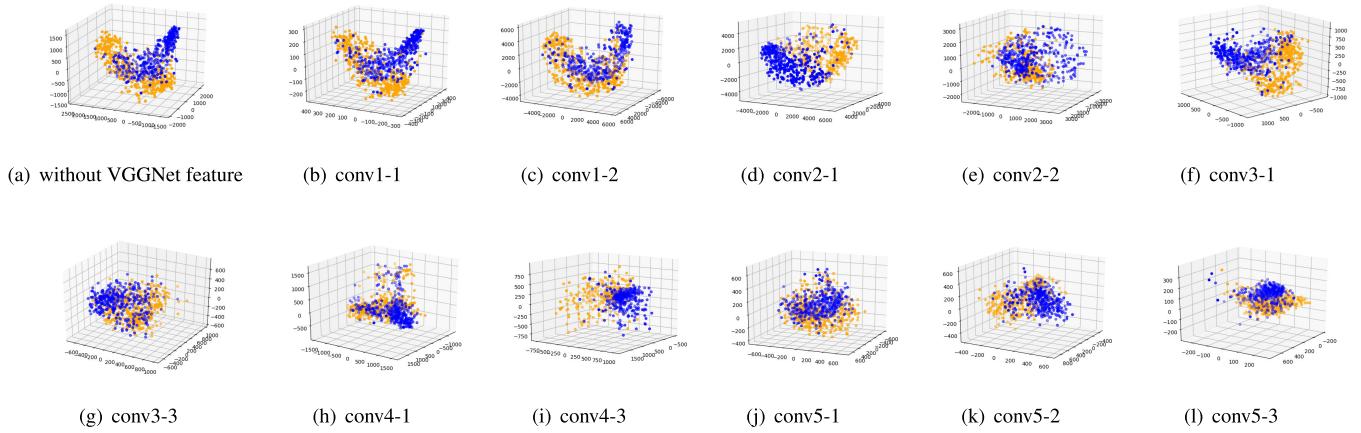


FIGURE 7. Results of the failure pattern of MNIST dataset. (a) shows the failure pattern that is directly obtained by input images and the failure patterns obtained by the features of each convolutional layer of VGGNet (except for conv3-2 and conv4-2) are presented from (b) to (l). The yellow dots represent the non-failure-causing inputs and the blue dots represent the failure-causing inputs.

TABLE 4. Comparing the value of F-measure between ARTDL and RT in four DL models with different failure rates. ARTDL(a_1) represents the ARTDL method whose distance metric is calculated by the features extracted by the conv3_2 layer of VGGNet. ARTDL(a_2) represents the ARTDL method whose distance metric is calculated by the features extracted by the conv4_2 layer of VGGNet. ARTDL(a_3) represents the ARTDL method whose distance metric is directly calculated by the input images. RT(r) represents the RT method.

DL models	F-measure				Reduction($\frac{r-a_1}{r} \times 100\%$)
	ARTDL(a_1)	ARTDL(a_2)	ARTDL(a_3)	RT(r)	
Rambo (failure rate=2.47%)	14.37	29.20	31.88	40.5	64.52%
Rambo (failure rate=0.52%)	57.11	162.58	149.3	167.2	65.84%
Autumn (failure rate=6.32%)	5.48	5.62	6.28	14.95	63.34%
Autumn (failure rate=2.93%)	14.31	16.14	18.26	40.54	64.70%
LeNet-1 (failure rate=7.65%)	6.82	8.10	6.90	12.94	47.30%
LeNet-1 (failure rate=2.90%)	15.89	19.94	18.26	33.19	52.12%
LeNet-4 (failure rate=1.41%)	17.24	21.93	18.93	71.21	75.79%
LeNet-4 (failure rate=0.85%)	28.67	38.72	45.75	103.86	72.40%
Average					62.74%

this goal, the size of M should be at least as

$$|M| = \left(\frac{100 \cdot z \cdot \sigma}{r \cdot \mu}\right)^2, \tag{4}$$

where z is the normal variate of the desired confidence level, μ is the population mean, and σ is the population standard deviation. In our experiment, the confidence level is set to 95% and r is set to 5. From the statistical tables, we know that for 95% confidence, $z = 1.96$. Thus, equation (4) becomes

$$|M| = \left(\frac{100 \cdot 1.96 \cdot \sigma}{5 \cdot \mu}\right)^2, \tag{5}$$

We compare the F-measure between ARTDL and RT in four DL models. Besides, we also compare three ARTDL methods with different distance metrics including the Euclidean distance between pixel vectors of input images, the FED between the feature vectors of conv3_2 layer and the FED between the feature vectors of conv4_2 layer. In order to ensure the universality of our conclusion, we conduct the experiments with different failure rates, to see how the value of F-measure varies. The results, as shown in Table 4, show that the values of F-measure of all ARTDL methods are lower

than RT. Besides, the ARTDL that calculates distance by using the features of conv3_2 layer has the best effect, that is, its value of F-measure is the smallest. Furthermore, when failure rates change, there is no significant change in the reduction ratio of the value of F-measure for all DL models. Hence, in this paper, we propose the ARTDL whose distance is calculated by the features extracted from input images by the conv3_2 layer of VGGNet. As shown in Table 4, the proposed ARTDL reduces the value of F-measure by 62.74% in average, compared with RT. The results show that the ARTDL has higher capability to detect fault than RT. Therefore, when we have chosen RT to test DL model, it is feasible to improve the effectiveness of RT by the ARTDL.

VI. THREATS TO VALIDITY

In this section, we will discuss the validity of ARTDL and the possible threats to the validity from four points: construct validity, external validity, internal validity, and reliability.

From the point of external validity, the ARTDL is suitable for DL systems whose inputs are images, but not for all DL systems. The ARTDL is based on the distance metric: FED. However, the FED is only suitable for calculating the distance

between two images because it relies on features extracted by VGGNet from images. So the ARTDL is only suitable for DL systems with image inputs, but not for the DL systems with other inputs such as speech, natural language and so on. From the point of internal validity, for some minority special cases, the conclusions of this paper may not be established. The results of the failure pattern of DL models show that absolute majority failure-causing inputs are far away from non-failure-causing inputs but there are minority failure-causing inputs that are close to the non-failure-causing inputs. Therefore, ARTDL may cease to be effective when the DL system contains only a few failures and they are all located very close to the correct inputs. From the point of construct validity, we measure the F-measure of ARTDL, which can validly evaluate the effectiveness of the ARTDL. From the point of reliability, our study can be easily repeated by others. In the experiment, there are no factors that may hinder the experiment from being repeated.

VII. CONCLUSION

In this work, we designed and implemented ARTDL, which can improve the effectiveness of RT for DL systems. We proposed a distance metric: FED, which is used to select random inputs by utilizing the fact that test cases with the furthest distance from non-failure-causing test cases are more likely to cause failures. Based on the guidance of the metric, the ARTDL reduced the expected number of test cases to detect the first failure by 62.74%, compared to RT. Therefore, we verified that it is worthwhile to replace RT with ARTDL when RT has been selected as a feasible test method for DL systems. And we presented potential usage of the ARTDL to expose the faulty behaviors of DL systems with less cost and ensure the reliability and safety of DL systems.

In the future, we will focus on how to make the ART method more applicable for more DL systems with different kinds of inputs (e.g., natural language and speech) to reduce the costs of RT for speech recognition system and machine translation system. Besides, we will further consider other distance metrics for DL systems and comprehensively analysis and compare the effectiveness of ART methods of DL systems with different distance metrics.

REFERENCES

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
- [3] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
- [4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
- [5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, "End to end learning for self-driving cars," Apr. 2016, *arXiv:1604.07316*. [Online]. Available: <https://arxiv.org/abs/1604.07316>
- [6] D. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2843–2851.
- [7] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *Proc. IEEE/AIAA 35th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2016, pp. 1–10.
- [8] (2014). *This is How Bad Self-Driving Cars Suck in the Rain*. [Online]. Available: <http://jalopnik.com/this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433>
- [9] *Tesla's Self-Driving System Cleared in deadly Crash*. [Online]. Available: <https://www.nytimes.com/2017/01/19/business/tesla-model-s-autopilot-fatal-crash.html>
- [10] J. W. Duran and S. C. Ntafos, "An evaluation of random testing," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 4, pp. 438–444, Jul. 1984.
- [11] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, P. Mcminn, A. Bertolino, J. Jenny Li, and H. Zhu, "An orchestrated survey of methodologies for automated software test case generation," *J. Syst. Softw.*, vol. 86, no. 8, pp. 1978–2001, Aug. 2013.
- [12] H. B. Braiek and F. Khomh, "On testing machine learning programs," Dec. 2018, *arXiv:1812.02257*. [Online]. Available: <https://arxiv.org/abs/1812.02257>
- [13] (2010). *Imagenet Crowdsourcing, Benchmarking & Other Cool Things*. [Online]. Available: <http://www.image-net.org/papers/ImageNet2010.pdf>
- [14] (2016). *Google Auto Waymo Disengagement Report for Autonomous Driving*. [Online]. Available: https://www.dmv.ca.gov/portal/wcm/connect/946b3502-c959-4e3b-b119-91319c27788f/GoogleAutoWaymo_disengage_report_2016.pdf?MOD=AJPERES
- [15] T. Y. Chen, H. Leung, and I. Mak, "Adaptive random testing," in *Proc. Annu. Asian Comput. Sci. Conf.* Berlin, Germany: Springer, 2004, pp. 320–329.
- [16] L. White and E. Cohen, "A domain strategy for computer program testing," *IEEE Trans. Softw. Eng.*, vol. SE-6, no. 3, pp. 247–257, May 1980.
- [17] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "ARTOO: Adaptive random testing for object-oriented software," in *Proc. 13th Int. Conf. Softw. Eng. (ICSE)*, 2008, pp. 71–80.
- [18] F.-C. Kuo, "On adaptive random testing," Ph.D. dissertation, Dept. Inf. Commun. Technol., Swinburne Univ. Technol., Melbourne, VIC, Australia, 2006.
- [19] R. G. Merkel, "Analysis and enhancements of adaptive random testing," Ph.D. dissertation, School Inf. Technol., Swinburne Univ. Technol., Melbourne, VIC, Australia, 2005.
- [20] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, R. Merkel, and G. Rothermel, "A cost-effective random testing method for programs with non-numeric inputs," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3509–3523, Mar. 2016.
- [21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [22] Y. Lecun and C. Cortes. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>
- [23] (2016). *Udacity self Driving Car Challenge 2*. [Online]. Available: <https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2>
- [24] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ. (SOSP)*, 2017, pp. 1–18.
- [25] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "DLFuzz: Differential fuzzing testing of deep learning systems," in *Proc. 26th ACM Joint Meeting Eur. Software Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, 2018, pp. 739–743.
- [26] L. Ma, Y. Liu, J. Zhao, Y. Wang, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, and L. Li, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng. (ASE)*, 2018, pp. 120–131.
- [27] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng. (ICSE)*, 2018, pp. 303–314.
- [28] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," Mar. 2018, *arXiv:1803.04792*. [Online]. Available: <https://arxiv.org/abs/1803.04792>

- [29] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2016.
- [30] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. Tse, "Adaptive random testing: The ART of test case diversity," *J. Syst. Softw.*, vol. 83, no. 1, pp. 60–66, Jan. 2010.
- [31] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," Jun. 2019, *arXiv:1906.10742*. [Online]. Available: <https://arxiv.org/abs/1906.10742>
- [32] S. Sherin and M. Z. Iqbal, "A systematic mapping study on testing of machine learning programs," Jul. 2019, *arXiv:1907.09427*. [Online]. Available: <https://arxiv.org/abs/1907.09427>
- [33] Z. Li, X. Ma, C. Xu, and C. Cao, "Structural coverage criteria for neural networks could be misleading," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., New Ideas Emerg. Results (ICSE-NIER)*, May 2019, pp. 89–92.
- [34] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.
- [35] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2011, pp. 24–29.
- [36] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *Proc. ECCV*, 2014.
- [37] F. Chan, T. Chen, I. Mak, and Y. Yu, "Proportional sampling strategy: Guidelines for software testing practitioners," *Inf. Softw. Technol.*, vol. 38, no. 12, pp. 775–782, Jan. 1996.
- [38] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006, p. 229.
- [39] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, May 2015.
- [40] T. Chen, T. Tse, and Z. Zhou, "Fault-based testing without the need of oracles," *Inf. Softw. Technol.*, vol. 45, no. 1, pp. 1–9, Jan. 2003.
- [41] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Commun. ACM*, vol. 62, no. 3, pp. 61–67, Feb. 2019.
- [42] C. Staff, "A case against mission-critical applications of machine learning," *Commun. ACM*, vol. 62, no. 8, p. 9, Jul. 2019.
- [43] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 4:1–4:27, 2018.
- [44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.
- [45] (2015). *Keras*. [Online]. Available: <https://github.com/keras-team/keras>
- [46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [47] C. C. Yann LeCun and C. J. Burges. (2010). *MNIST Handwritten Digit Database. AT&T LABS*. [Online]. Available: <http://yann.lecun.com/exdb/mnist>
- [48] (2016). *Rambo Model*. [Online]. Available: <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo>
- [49] (2016). *Autumn Model*. [Online]. Available: <https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/autumn>
- [50] (2016). *Comma.ai's Steering Model*. [Online]. Available: https://github.com/commaai/research/blob/master/train_steering_model.py
- [51] (2016). *Open Source Computer Vision Library*. [Online]. Available: <https://github.com/opencv/opencv>
- [52] G. A. Simon and J. F. Freund, *Modern Elementary Statistics*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.



MIN YAN received the bachelor's degree from the Wuhan University of Technology (WHUT), Wuhan, China, in 2015. She is currently pursuing the Ph.D. degree with the School of Software Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. Her research interests include safety and reliability of machine learning and software testing.



LI WANG (Senior Member, IEEE) received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2009. She held visiting positions with the School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA, USA, from December 2013 to January 2015, and the Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, from August 2015 to November 2015. She is currently

a Full Professor with the School of Electronic Engineering, BUPT, where she is also the Head of the High Performance Computing and Networking Laboratory. She is also with the Key Laboratory of the Universal Wireless Communications, Ministry of Education, China. Her current research interests include wireless communications, secure communications, cooperative networking, and distributed networking and storage. She has published two books for *Device-to-Device Communications* (Springer) and *Physical Layer Security* (Springer), respectively. She received the 2013 Beijing Young Elite Faculty for Higher Education Award, the Best Paper Award from ICCTA 2011, the Best Paper Runner Up from WASA 2015, the Best Paper Award from the IEEE ICC 2017, the Demo Award from the IEEE ICC 2018, and the Best Paper Award from the IEEE GLOBECOM 2018. She has been selected by the Beijing Nova Program, in 2018. She is the Symposium Chair of the IEEE ICC 2019 on Cognitive Radio and Networks Symposium, and the Chair of the Special Interest Group (SIG) on Social Behavior Driven Cognitive Radio Networks for the IEEE Technical Committee on Cognitive Networks. She served on the technical program committees for multiple the IEEE conferences, including the IEEE GLOBECOM, ICC, WCNC, and VTC over the years. She is serving as an Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and an Associate Editor for IEEE ACCESS.



AIGUO FEI is currently a member of the Chinese Academy of Engineering (CAE) and a Professor with the School of Software Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. He is also the President of the Chinese Command and Control Society. His current research interests include the Internet of Things and intelligent information system, big data and cloud computing, intelligent emergency communication system, and intelligence software development and testing.

...