Information and Software Technology 150 (2022) 106982

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Can test input selection methods for deep neural network guarantee test diversity? A large-scale empirical study

Chunyu Zhao [a], Yanzhou Mu [b], Xiang Chen [a,c,d,*], Jingke Zhao [a,b], Xiaolin Ju [a,**], Gan Wang [b]

[a] School of Information Science and Technology, Nantong University, Nantong, China
[b] College of Intelligence and Computing, Tianjin University, Tianjin, China
[c] Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology, Nanjing, China
[d] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:
Deep neural network testing
Test input selection
Test diversity
Empirical study

ABSTRACT

Context: Recently, various methods on test input selection for deep neural network (TIS-DNN) have been proposed. These methods can effectively reduce the labeling cost by selecting a subset from the original test inputs, which can still accurately estimate the performance (such as accuracy) of the target DNN models.
Objective: Previous studies on TIS-DNN mainly focused on the performance on all the classes. However, the selected subset may miss the coverage of some classes or decrease the performance on some classes, which will reduce the test diversity of the original test inputs.
Methods: Therefore, we conducted a large-scale empirical study to investigate whether previous TIS-DNN methods can guarantee test diversity in the subset. In our study, we selected five state-of-the-art TIS-DNN methods: SRS, CSS, CES, DeepReduce and PACE. Then we selected 18 pairs of DNN models and the corresponding test inputs from seven popular DNN datasets.
Results: Our experimental results can be summarized as follows. (1) Previous TIS-DNN methods can guarantee the performance on all the classes. However, these methods have a negative impact on the test diversity and the performance on each class is not satisfactory. (2) Reducing the performance estimation error on each class can help reduce the estimation error on the test adequacy of the original inputs based on DNN-based coverage criteria (especially for the criterion NC and the criterion TKNC). (3) There still exists great room for performance improvement (i.e., 7.637% improvement on all the classes and 12.833% improvement on each class) after comparing the TIS-DNN method PACE with approximately optimal solutions.
Conclusion: The above experimental findings implicate there is still a long way for the TIS-DNN issue to go. Given this, we present observations about the road ahead for this issue.

1. Introduction

With the increasing adoption of deep neural networks (DNNs) in various application domains (such as computer vision [1], natural language processing [2], software engineering [3–9]), more and more attention has been paid to the quality assurance of the DNNs [10]. Similar to traditional software, DNN-based applications also contain faults. Specifically, due to the complex internal structure, minor disturbances in the data will cause the DNN models to make incomprehensible wrong predictions. Some of the faults can even threaten personal safety. For example, in 2018, a woman was killed by one of Uber's self-driving cars in Arizona.[1] In the same year, a semi-autonomous Tesla Model S crashed into a parked firetruck on California freeway.[2] Therefore, the security of DNNs has attracted more concerns, and many software testing methods have been applied to ensure its quality and reliability [11–13].

Recently, DNN model testing has been very costly due to numerous test inputs. The test inputs can be different, such as images, voice, text, and other data. Labeling these inputs is time-consuming and heavily relies on domain knowledge [14–16]. For some domains (such as disease diagnosis based on pathological images), labeling these inputs even requires multiple domain experts to guarantee the label's correctness.

* Corresponding author at: School of Information Science and Technology, Nantong University, Nantong, China.
** Corresponding author.
E-mail addresses: 2010320004@stmail.ntu.edu.cn (C. Zhao), 2019218009@tju.edu.cn (Y. Mu), xchencs@ntu.edu.cn (X. Chen), ke1206371563@gmail.com (J. Zhao), ju.xl@ntu.edu.cn (X. Ju), wg_98@tju.edu.cn (G. Wang).

[1] https://www.vice.com/en/article/9kga85/uber-is-giving-up-on-self-driving-cars-in-california-after-deadly-crash.
[2] https://www.newsweek.com/autonomous-tesla-crashes-parked-fire-truck-california-freeway-789177.

https://doi.org/10.1016/j.infsof.2022.106982
Received 19 October 2021; Received in revised form 7 June 2022; Accepted 8 June 2022
Available online 16 June 2022

To reduce the labeling cost, test input selection for deep neural network (TIS-DNN) aims to select a small subset from the original test inputs to estimate the performance (such as accuracy) of the target DNN model. Then experts only need to label the test inputs in this small-scale subset, which is helpful to reduce the labeling cost. Moreover, executing the test inputs in this subset can also effectively save model performance evaluation time. Nowadays, researchers have proposed different novel TIS-DNN methods. For example, Li et al. [16] proposed a stratified sampling method CSS which is based on the model prediction confidence, and a cross-entropy-based sampling method CES. Later, Zhou et al. [17] proposed a two-phase TIS-DNN method DeepReduce. Recently, Chen et al. [14] proposed a practical accuracy estimation method PACE. Most of the prior TIS-DNN methods only examine the performance difference between the selected subset and the original test input. However, after our manual analysis, the selected subset may lose some test properties. For example, when the selection ratio of the TIS-DNN method PACE [14] is set to 1.5% for the dataset CIFAR-100, the accuracy estimation error on all the classes is only 3.9%, but 28% of the classes in the original test inputs are missed, which means that the selected subset cannot test the performance of the target DNN model on these missed classes.

Therefore, in our study, we aim at covering the classes of the original test input as much as possible in the resulting selected subset. Moreover, the performance estimation of each class should be close to the original test input as much as possible. We call this test property the test diversity. To our best knowledge, whether previous TIS-DNN methods can guarantee the test diversity has not been investigated in previous studies. Our large-scale empirical study first selected 18 pairs of DNN models and the corresponding test inputs from seven popular DNN datasets as our experimental subjects. Then we consider four state-of-the-art methods: CSS [16], CES [16], DeepReduce [17], and PACE [14]. Moreover, we also consider a simple random sampling method SRS as a reference method.

In summary, the contributions of our empirical study can be summarized as follows.

- To the best of our knowledge, we are the first to investigate the test diversity issue in the subsets selected by the state-of-the-art TIS-DNN methods.
- We conducted a large-scale empirical study to investigate this issue. In our empirical study, we select 18 pairs of DNN models and corresponding test inputs. Then we consider five state-of-the-art TIS-DNN methods. Our empirical results verified that the test diversity is not satisfactory for current TIS-DNN methods. Moreover, we find that concerning the test diversity can improve the coverage rate estimation accuracy base on the DNN-based coverage criterion.
- Since there is still much room for performance improvement, researchers need to design more novel effective TIS-DNN methods in the future. In this study, we design a multi-objective optimization-based method to explore one of the feasible directions and show promising results.
- For other researchers to follow our study, we shared our scripts, our proposed multi-objective optimization-based approach, and the detailed results in our project homepage.[3]

The remainder of this paper is organized as follows. Section 2 analyzes the related work and emphasizes the novelty of our study. Section 3 shows the motivation and problem formalization of TIS-DNN. Section 4 illustrates our three research questions (RQs) and their design motivation. Section 5 shows the case study design, including experimental subjects, our chosen five state-of-the-art TIS-DNN methods, performance measures, and the implementation details. Section 6 performs result analysis for each RQ. Section 7 investigates whether considering the test diversity can guarantee performance estimation on all the classes and introduces our proposed multi-objective optimization-based TIS-DNN method. Section 8 discusses potential threats for our empirical results. Section 9 concludes our study.

[3] https://github.com/SwilderY/TIS-DNN.

2. Related work

In this section, we first analyze the related work for deep neural network and quality assurance studies for them. Then we focus on test optimization for deep neural network testing. After analyzing the related studies, we emphasize the novelty of our study.

2.1. Deep neural network testing

Different from traditional software [18], the deep neural network is a special software artifact (i.e., the DNN models automatically learn the knowledge from the data). Since DNN models consist of multiple layers and each layer contains multiple neurons, researchers have proposed different coverage criteria by analyzing the structure of the DNN models. For example, Pei et al. [19] first proposed a white-box framework for testing real-world DNN models and then proposed neuron coverage criteria. Ma et al. [20] proposed a range of finer-grained adequacy criteria DeepGauge, which includes neuron-level coverage criteria and layer-level coverage criteria. Kim et al. [21] proposed surprise adequacy for DNN models, which can quantitatively measure the relative surprise of each input for the training data (i.e., surprise adequacy). Moreover, they also proposed surprise coverage, which can measure the coverage of neurons with specific activation traits. Du et al. [22] mainly focused on RNN models and designed five coverage criteria, which can perform quantitative analysis on RNN models. Except for the above DNN-based coverage criteria, researchers also studied test generation techniques. For example, Sun et al. [12] first applied concolic testing for DNN models. Xie et al. [11] proposed a coverage-guided fuzz testing framework DeepHunter. Later, Zhang et al. [23] proposed a simple and effective fuzzing-based black box attack method DeepSearch. Finally, researchers also focused on adversarial samples [24]. DNN models are vulnerable to even a small perturbation of input data (i.e., adversarial samples). Until now, different attack techniques have been proposed (such as FGSM [25], JSMA [26], DeepFool [27]). Then researchers focused on the robustness improvement of the DNN models. For example, Papernot et al. [28] introduced defensive distillation to reduce the effectiveness of adversarial samples.

In our study, we resorted to DNN-based coverage criteria [19,20] to investigate whether reducing the accuracy estimation error on each class of the selected test subset can help to decrease the coverage rate estimation error based on DNN-based coverage criteria.

2.2. Test optimization for deep neural network

There are two kinds of optimization methods, which can reduce the labeling cost of the gathered test inputs. The first kind of methods focused on test input selection. The purpose of these TIS-DNN methods is to select a small-scale subset from the original inputs, which can still accurately estimate the performance of the DNN models under test. Then users only need to label the test inputs in the subset, which can effectively save the labeling cost. Li et al. [16] first proposed a confidence-based stratified sampling method and a cross-entropy-based sampling method. Later, Zhou et al. [17] proposed a two-phase test input reduction method DeepReduce. Recently, Chen et al. [14] proposed a practical accuracy estimation-based method PACE.

The second kind of methods focused on test input prioritization. The purpose of these methods is to label the test inputs that are more valuable to test the DNN model under the guidance of the predefined prioritization rules, which can finally improve the test efficiency. Compared with the first kind of method, the second kind of method does

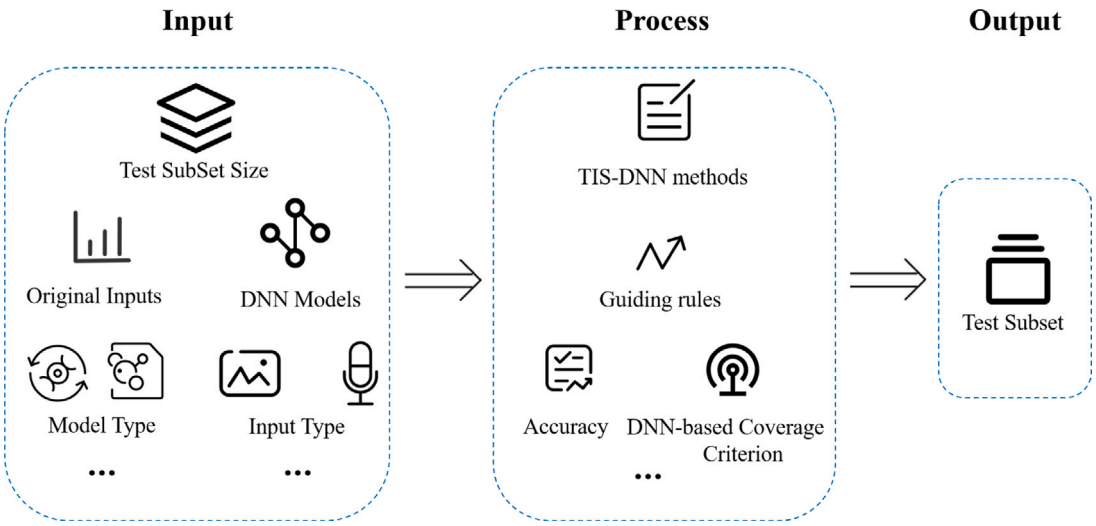


Fig. 1. The overview of our case study design.

not need to discard any test inputs, and the inputs that will cause the model to produce error predictions can be found as early as possible. Compared with the first kind of methods, the second kind of methods paid more attention to finding the test inputs, which could lead to the wrong prediction of the model or improve its performance after retraining. Byun et al. [29] first investigated this issue. They conduct the test input prioritization under the guidance of sentiment metrics. Feng et al. [30] proposed a test prioritization method DeepGini based on the statistical perspective. Recently, Wang et al. [15] proposed a test input prioritization method PRIMA via intelligent mutation analysis.

2.3. Novelty of our study

In this study, we mainly focus on the first kind of methods (i.e., test input selection methods) for deep neural network. After our in-depth analysis, we find that the selected subset by the previous TIS-DNN method [14,16,17] may miss the coverage of some classes in the original test input (i.e., test diversity on the select subsets) in our manual analysis. This means that the selected subset cannot test the performance of the target DNN model on these missed classes. To the best of our knowledge, this issue has not been studied in previous studies. Therefore, we conducted a large-scale empirical study based on popular DNN models and state-of-the-art TIS-DNN methods to investigate whether previous TIS-DNN methods can guarantee test diversity on the select subsets. Moreover, we also explored how much room there is for the improvement of previous TIS-DNN methods, and provided implications for future studies on the TIS-DNN problem.

3. Motivation and problem formalization of TIS-DNN

Accurate performance (such as accuracy) estimation is important for quality assurance of the target DNN models. If the performance is not satisfactory for the current application, the developers can alter the structure of the DNN models (e.g., adding or deleting a layer) or further tune the hyper-parameters to improve the model performance. However, labeling all the test inputs is time-consuming and heavily relying on expert knowledge. For example, when preparing ImageNet, the researchers resorted to Amazon Mechanical Turk, which is an online platform for users to complete the task and get paid [31]. Test input selection is an effective way to reduce the labeling efforts. Specifically, TIS-DNN methods aim to accurately estimate the performance of the target DNN models by selecting a subset from the original test inputs. The formal definition of TIS-DNN problem can be defined as follows.

Definition 1 (*Test Input Selection for DNN*)**.** Given original unlabeled test set T, the DNN model M to be tested, and the evaluation function f, Suppose the selected test subset is denoted as T', the test input selection for DNN aims to make T' meet the following conditions:

$\|T'\| \ll \|T\|$: $f(T, M) \approx f(T', M)$

In Definition 1, the evaluation function can be defined based on test properties (such as accuracy on all the classes).

4. Research questions

In our empirical study, we aim to answer the following three research questions (RQs).

RQ1: Can previous TIS-DNN methods guarantee the accurate performance estimation on each class?

Motivation. Previous TIS-DNN methods [14,16,17] mainly used the performance on all the classes to evaluate the performance difference between the selected subset and the original test inputs. However, the selected subset may reduce the test diversity of the original test. For example, some classes of the original inputs may not be selected by TIS-DNN methods. Moreover, there will exist a huge performance difference in predicting some classes. Therefore, in this RQ, we first want to verify whether the previous TIS-DNN methods can guarantee accurate performance estimation on all the classes. Then, we want to further investigate whether the previous TIS-DNN methods can ensure the accurate performance estimation on each class.

RQ2: Can the accurate performance estimation on each class help to achieve accurate test adequacy estimation (i.e., coverage estimation based on DNN-based coverage criteria)?

Motivation. In traditional software testing, the coverage criteria (such as control-flow coverage, data-flow coverage) [18] are used to evaluate the adequacy of the test suite. However, these coverage criteria cannot be directly applied to test deep neural networks, which are data-driven programming paradigm. Therefore, researchers proposed different DNN-based coverage criteria (such as DeepXplore [19], DeepGauge [20]) to measure the test adequacy of the test inputs. In this RQ, we aim to investigate whether accurate performance estimation on each class can help to achieve accurate coverage estimation for the test adequacy of the original inputs based on DNN-based coverage criteria.

RQ3: Can current TIS-DNN methods be improved in terms of performance?

Motivation. Since we know the labels of all the test inputs for each dataset in our controlled empirical study, we can design a method to obtain the approximate optimal subset for each pair of the model and the corresponding test inputs respectively. Then, after comparing

the approximate optimal subsets, we can compute the performance improvement room of the current TIS-DNN methods in this RQ. If the room for performance improvement is large, it means that more advanced effective TIS-DNN methods need to be proposed in the future. Moreover, since the size of the subset is an important parameter for TIS-DNN methods, we also want to investigate whether exists the suggest value range of the subset for previous TIS-DNN methods, which can help to achieve the minimum performance estimation error on all the classes and each class.

5. Case study design

In this section, we describe our case study design to address our research questions. We show the overview of our case study design in Fig. 1. This figure mainly shows the process of constructing test subsets by the TIS-DNN method. The inputs are the number of the selected subset, the target DNN model and its corresponding unlabeled original inputs, while the output is the selected subset for experts to label the class. In our empirical study, we consider different DNN model types and different input types to guarantee the generalization of our empirical findings. During the process of the TIS-DNN method, we first measure the accuracy and the coverage rate based on a DNN-based coverage criterion of the original inputs and the subset. Then we compare the performance difference between these two sets.

In the rest of this section, we first introduce the experimental subjects. Then we show the state-of-the-art TIS-DNN methods. Later, we illustrate the performance measures based on accuracy and the coverage rate used in our case study. Finally, we give the implementation details and running platform information.

5.1. Experimental subjects

In our empirical studies, we select 18 pairs of DNN models and the corresponding test inputs as our experimental subjects. The characteristics of the DNN models and the corresponding datasets can be found in Table 1. The selected DNN models are trained on seven datasets (i.e., MNIST,[4] CIFAR-10,[5] CIFAR-100,[6] SVHN,[7] ImageNET,[8] Fashion-MNIST,[9] and Speech-Commands[10]), which have been widely used in previous DNN quality assurance studies [14,16,17]. The corresponding test inputs also have a certain diversity. From the perspective of the test input types, these test inputs include image data and audio data. Since adversarial input generation is a popular method to test DNN models, we also consider adversarial inputs for some datasets. Specifically, for four datasets (i.e., Autogen-MNIST, Autogen-SVHN, Autogen-Fashion, and Autogen-CIFAR-10), we use these mixed inputs (i.e., 5000 original test inputs and 5000 adversarial inputs), which were constructed for the performance evaluation of the method DeepGini [30] and the method PACE [14]. The adversarial inputs are created by a basic iterative method [32].

The characteristics of these datasets, the corresponding trained DNN models and test inputs are illustrated as follows.

- **MNIST.** MNIST is a handwritten digit dataset, which contains a total of 70,000 input data, of which 60,000 data are training data and 10,000 data are test data. This data is divided into 10 classes (i.e., categories). On the dataset MINIST, we first consider three classical models proposed by LeCun et al. [33] for MNIST (i.e., LeNet-1, LeNet-4, LeNet-5). Moreover, we also consider the mutated models to simulate low-accuracy models, whose performance is less than 0.8 by following the previous study [16]. In our study, we directly adopted their three mutated models (i.e., LeNet-5-M1, LeNet-5-M2, and LeNet-5-M3).
- **CIFAR-10.** The dataset CIFAR-10 contains a total of 60,000 32×32 color images in 10 different classes (i.e., airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks). Each class contains 6000 images. Since the images in CIFAR-10 are low-resolution (32×32), this dataset can allow researchers to quickly try different DNN models. For this dataset, 50,000 images are used for the training set, and the remaining 10,000 images are used for the test set. On the dataset CIFAR-10, we consider two models (i.e., the high-accuracy model ResNet-20 and the low-accuracy model VGG-16) by following the previous studies [14,30].
- **CIFAR-100.** The dataset CIFAR-100 is similar to the dataset CIFAR-10, except that it contains 100 classes. Each class contains 600 images. Therefore, this dataset contains 60,000 images. For each class, 500 images are used as the training set and the remaining 100 images are used as the test set. The 100 classes in CIFAR-100 are grouped into 20 superclasses. Each image has a "fine" label (i.e., the class to which it belongs) and a "coarse" label (i.e., the superclass to which it belongs). On the dataset CIFAR-100, we only consider the model ResNet-20 by following the previous study [14].
- **SVHN.** The dataset SVHN was gathered from the house number in the Google street view images. Its style is similar to MNIST (i.e., the images are of small cropped digits), but it contains an order of magnitude more labeled data, which contains more than 600,000 digital images. These images are divided into 10 categories. This dataset is used to solve a more challenging problem (i.e., identifying characters and numbers in natural scene images) [34]. On the dataset SVHN, we only consider the model LeNet-5 by following the previous studies [14,30].
- **Fashion-MNIST.** The dataset Fashion-MNIST is a dataset of Zalando's article images and serves as a drop-in replacement for the original MNIST dataset. Fashion-MNIST, which was provided by a fashion company from Germany, covers 70,000 different product images from 10 classes. It keeps the same image size and structure of training and testing splits with the dataset MNIST. On the dataset Fashion-MNIST, we only consider the model LeNet-5 by following the previous study [14].
- **ImageNET.** The dataset ImageNET is a large-scale image dataset, which was established to promote the development of computer image recognition. ImageNET collected more than 1.4 million images as the training data, 50,000 images as the validation data, and 50,000 images as the test data. These images are divided into 1000 classes. In our study, we consider two models (i.e., ResNet-50 and VGG-19) by following the previous study [14].
- **Speech-Commands.** The dataset Speech-Commands is used for speech recognition. Each set of data contains a single English word. These words are spoken by different speakers. These audio files are organized into different folders according to the words they contain. This dataset contains a total of 64,727 audio files and these audio files are divided into 30 classes. On the dataset Speech-Commands, we consider the model DeepSpeech by following the previous study [14].

In summary, to build a comprehensive test benchmark to alleviate the external threat of our empirical study, we guarantee the diversity of the selected experimental subjects from the following four perspectives.

- From the perspective of model accuracy, these models can be divided into high-precision models (i.e., the accuracy is over 0.8) and low-precision models (i.e., the accuracy is not over 0.8).

[4] http://yann.lecun.com/exdb/mnist.
[5] http://www.cs.toronto.edu/~kriz/CIFAR.html.
[6] https://www.cs.toronto.edu/~kriz/cifar.html.
[7] http://ufldl.stanford.edu/housenumbers.
[8] http://www.image-net.org.
[9] https://github.com/zalandoresearch/fashion-mnist.
[10] https://github.com/bjtommychen/Keras_DeepSpeech2_SpeechRecognition.

Table 1
The characteristics of the DNN models and corresponding original test inputs.

ID	Dataset	Model	Model size (KB)	Input size	Accuracy (%)	# Class	Input type
1	MNIST	LeNet-1	113	10,000	94.86	10	Original
2		LeNet-4	947	10,000	96.79	10	Original
3		LeNet-5	1093	10,000	98.68	10	Original
4		LeNet-5-M1	1093	10,000	79.53	10	Original
5		LeNet-5-M2	1093	10,000	77.27	10	Original
6		LeNet-5-M3	1093	10,000	79.14	10	Original
7	CIFAR-10	VGG-16	21,814	10,000	78.71	10	Original
8		ResNet-20	3507	10,000	91.45	10	Original
9	CIFAR-100	ResNet-20	10,615	10,000	71.42	100	Original
10	SVHN	LeNet-5	522	26,032	87.90	10	Original
11	Speech-Commands	DeepSpeech	6734	6471	94.53	30	Original
12	Fashion-MNIST	LeNet-5	385	10,000	89.88	10	Original
13	ImageNet	VGG-19	562,176	50,000	64.73	1000	Original
14		ResNet-50	100,352	50,000	68.27	1000	Original
15	Autogen-MNIST	LeNet-5	1093	10,000	49.35	10	Generated
16	Autogen-SVHN	LeNet-5	522	10,000	43.69	10	Generated
17	Autogen-Fashion	LeNet-5	385	10,000	45.31	10	Generated
18	Autogen-CIFAR-10	ResNet-20	3507	10,000	42.92	10	Generated

- From the perspective of model structure, these models can be divided into convolution neural networks (CNNs) and recurrent neural networks (RNNs).
- From the perspective of the test input types, these inputs include image and audio data.
- From the perspective of the test input generation methods, these inputs include original inputs and adversarial inputs.

5.2. TIS-DNN methods

In our study, we mainly consider five state-of-the-art TIS-DNN methods [14,16,17]. All of these methods have been proposed in the past three years, and the related studies were published in high-quality journals and conferences. Therefore, we can ensure that our chosen TIS-DNN methods are representative and stand for the state of the art in this research topic.

A brief introduction of these TIS-DNN methods can be found as follows.

- **SRS (Simple Random Sampling).** SRS is the simplest method to select the subset from the original test inputs. Specifically, this method randomly selects a required number of test inputs without replacement from the original test inputs with the same probability.
- **CSS (Confidence-based Stratified Sampling).** CSS [16] first divides the confidence value, which is provided by the classification model when predicting the label for the input, into different intervals. Then it selects test inputs based on their confidence values in different intervals, which can guarantee the distribution similarity between the selected subset and the original inputs.
- **CES (Cross-Entropy based Sampling).** CES [16] extracts its last hidden-layer features as the feature vector for each test input, which can be viewed as the learned representation of the test input. Then CES selects a required number of test inputs by minimizing the cross entropy between the selected subset and the original inputs to guarantee the distribution similarity.
- **DeepReduce.** DeepReduce [17] adopts a two-phase strategy. In the first phase, it uses a greedy method HGS [35] to select the minimized subset with the same testing adequacy of the original inputs based on neuron coverage criteria. In the second phase, it uses a heuristic-based method to add more testing data based on relative-entropy minimization, which can approximate the output distribution between the original inputs and the select subset. To represent output distribution, they use the outputs of the neurons in the last layer of a DNN model, which is more related to the behavior of a DNN model.
- **PACE (Practical ACcuracy Estimation).** To guarantee the selected subset can cover different testing capabilities (i.e., different functionalities of a DNN model) and maintain the original distribution of these testing capabilities, PACE [14] first clusters the original test inputs into different groups, which can discriminate the test inputs with different testing capabilities. Since the sizes of groups can reflect the distribution of different testing capabilities, PACE then selects the most representative test inputs from each group according to the proportion of different group sizes by utilizing an example-based explanation algorithm (i.e., the MMD-critic algorithm [36]). Finally, there may exist some test inputs, which do not belong to any group. However, each of them is likely to have a unique testing capability and the space of these test inputs can be called the minority space. PACE utilizes adaptive random testing [37] to select the representative test inputs from the minority space.

5.3. Performance measures

Hereafter, we present the different performance measures that we used in our study. They are divided into two categories: accuracy based measures and coverage-rate based measures.

5.3.1. Accuracy-based performance measures

Accuracy performance measure computes the proportion of correctly predicted test inputs to all the test inputs, which has been commonly used to evaluate the performance of the DNN models under test. For previous TIS-DNN studies, researchers mainly focused on accuracy estimation error on all the classes $AccEE_{all}$. Supposing the accuracy on the subset is Acc_{sub} and the accuracy on the original set is Acc_{org}, $AccEE_{all}$ for the DNN model under test can be computed as follows.

$$AccEE_{all} = |Acc_{sub} - Acc_{org}| \quad (1)$$

Then, we further investigate the test diversity in the subset by the performance measure $AccEE_{avg}$, which focuses on the average value of the accuracy over all the classes. Supposing the original test inputs include m classes. For the ith class, the accuracy on the subset is $Acc_{sub}[i]$ and the accuracy on the original test inputs is $Acc_{org}[i]$. Then $AccEE_{avg}$ for the DNN model under test can be computed as follows.

$$AccEE_{avg} = \frac{\sum_{i=1}^{m} |Acc_{sub}[i] - Acc_{org}[i]|}{m} \quad (2)$$

5.3.2. Coverage rate-based performance measure

Finally, we consider the coverage rate difference $CovDiff$ between the subset and the original test inputs when given a DNN-based coverage criterion [19,20].

The brief introduction of our considered five coverage criteria is shown as follows.

- **NC (Neuron Coverage).** NC criterion was first proposed by Pei et al. [19]. This is the first criterion to measure the adequacy of the deep learning test inputs. This criterion measures the proportion of activated neurons in total neurons. The neuron is activated when its output value is greater than the pre-established threshold.

Later, Ma et al. [20] proposed DeepGauge, which includes the following four coverage criteria with different granularities. Specifically, KMNC, NBC, and SNAC are designed in the neuron-level granularity. TKNC is designed in the layer-level granularity.

- **KMNC (K-multisection Neuron Coverage).** KMNC first divides the output range of each neuron into K sections from the training data. If the output value of a neuron (suppose that the output value interval of neuron n as $[a, b]$) falls into a section after executing the next test input, it is regarded as the coverage interval, so it calculates the proportion of all neuron coverage intervals.
- **NBC (Neuron Boundary Coverage).** Unlike KMNC, NBC considers whether the external areas of neuron output values (i.e., $(-\infty, a)$ and $(b, +\infty)$) are covered after executing the test input.
- **SNAC (Strong Neuron Activation Coverage).** SNAC only considers whether the external area of the upper bound (i.e., $(b, +\infty)$) is covered after the test input is executed.
- **TKNC (Top-k Neuron Coverage).** Researchers conjecture that the characteristics of neurons in the same layer are similar, while the active neurons in different layers are the main components that determine the characteristic of DNN model [20]. Therefore, the TKNC metric measures the proportion of the top k most active neurons in each layer to the total number of neurons.

Notice some of our considered criteria are controversial and their practicality was challenged by researchers. For example, Fabrice et al. [38] proved that NC and its variants were useless under some conditions. Later Chen et al. [39] also found that the above neuron coverage criteria were useless for finding the data, which can make the model generate wrong predictions. However, in this study, we only take these criteria to measure the test adequacy difference between the selected subsets and the original test inputs. Therefore, we do not consider the practicability in DNN testing.

Supposing the coverage rate on the subset is Cov_{sub} and the coverage rate on the original test inputs is Cov_{org}, $CovDiff$ for the DNN model under test can be computed as follows.

$$CovDiff = |Cov_{sub} - Cov_{org}| \quad (3)$$

Table 2
The subset size setting method used in result analysis for RQ1.

# Class	Minimum size	Maximum size	Step size
10	5	205	10
30	22	1590	64
100	50	2050	100
1000	500	20,500	1,000

5.4. Implementation details and running platform

All the TIS-DNN methods and experimental scripts were implemented on two deep learning frameworks (i.e., Tensorflow-1.15.0 and Keras-2.3.1) by using Python programming language (i.e., Python 3.6). All the deep learning models and the original test inputs shown in Table 1 are gathered from Github. We conducted our empirical study on the server (CPU: Xeon E5-2640 v4, hard disk: 4T, memory: 125 GB).

6. Result analysis

6.1. Result analysis for RQ1

Method. To answer this RQ, we first analyze the overall accuracy on all the classes $AccEE_{all}$ and the accuracy on each class $AccEE_{avg}$ by controlling the size of the selected subset. During our empirical study, we find the test diversity problem for TIS-DNN will be obvious when the subset size is within a specific value range. Moreover, we find the value range of different datasets has a certain correlation with the class number of the dataset, but not the instances in the dataset. The subset size setting method used in RQ1 can be found in Table 2. For example, for the datasets with 10 classes, the subset size is increased from 5 to 205, and the step size is 10. For the datasets with 30 classes, the subset size is increased from 22 to 1590, and the step size is 64. Given the subset size, we can run each TIS-DNN method to collect $AccEE_{all}$ and $AccEE_{avg}$. Since SRS is a random method, we run SRS 10 times independently with different random seeds and obtain the average value. For the remaining TIS-DNN methods, we only run them once.

Result. In this RQ, we first show the results on four DNN models From Figs. 2 to 5. For each subfigure, the x-axis denotes the size of the selected subset. the y-axis denotes the accuracy estimation error on all the classes and each class. In terms of accuracy estimation error, all the TIS-DNN methods can keep the low value of $AccEE_{all}$. For example, in Fig. 2, the value of $AccEE_{all}$ is almost no larger than 0.1. These results of our replication study can verify the effectiveness of previous TIS-DNN methods. However, after considering the test diversity in the subset, we find the accuracy estimation error on each class cannot be guaranteed (accuracy estimation error can be even up to 0.5 in some cases) especially when the size of the subset is small. For example, in Fig. 2, when the size of the subset is larger than 400 (i.e., 4% of the original test inputs), the value of $AccEE_{avg}$ will decrease significantly. In Fig. 3, when the size of the subset is larger than 3500 (i.e., 7% of the original test inputs), the value of $AccEE_{avg}$ will decrease significantly. We can achieve similar findings for other DNN models and the detailed results can be found in our project homepage.

Finally, we aim to comprehensively compare different TIS-DNN methods. For the ith pair of DNN model and corresponding test inputs, we compute the average $AccEE_{all}$ value ($AccEE_{all}[i]$) and $AccEE_{avg}$ value ($AccEE_{avg}[i]$) when considering different subset sizes for each TIS-DNN method. Notice we cannot gather the result of the method CSS on the Speech-Commands dataset, since the confidence between each alphabet and a word is impractical to confirm. Therefore, we show the value distribution of $AccEE_{all}$ and $AccEE_{avg}$for 17 pairs in Fig. 6 via boxplot. When considering the value $AccEE_{all}$ and $AccEE_{avg}$, we

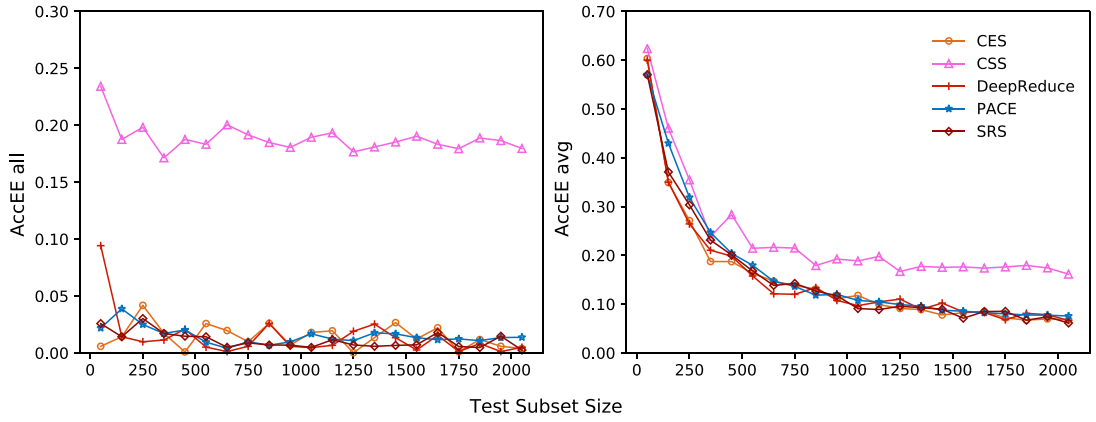


Fig. 2. Accuracy estimation error on ResNet-20 model for the CIFAR-100 dataset when considering different subset size.

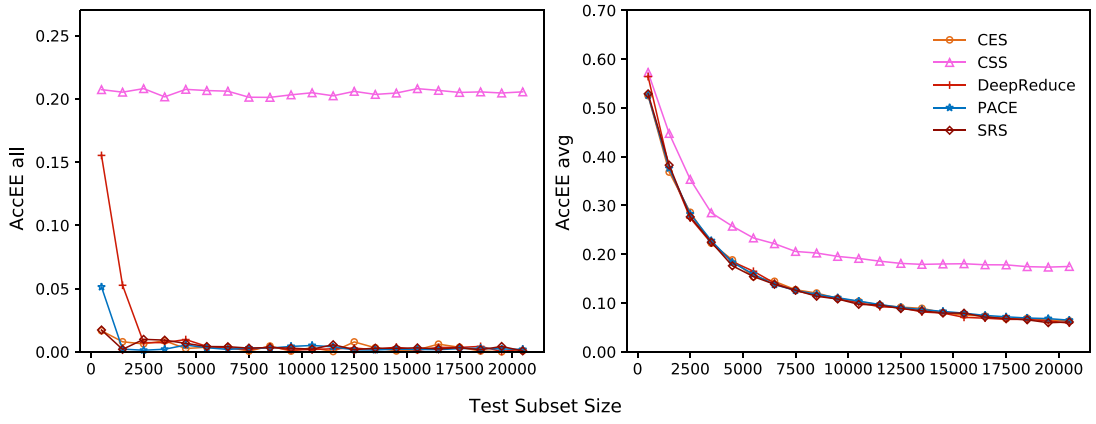


Fig. 3. Accuracy estimation error on VGG-19 model for the ImageNet dataset when considering different subset size.

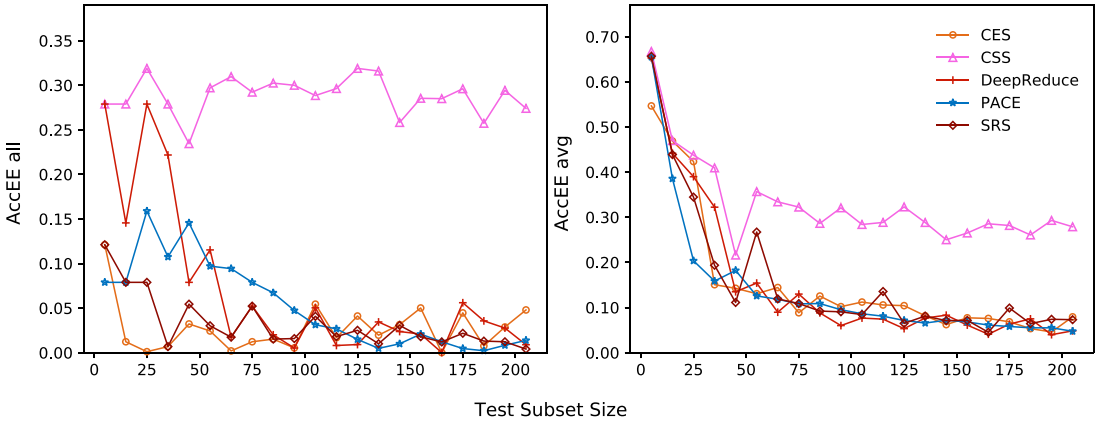


Fig. 4. Accuracy estimation error on LeNet-5 model for the SVHN dataset when considering different subset size.

can find that except for the method CSS, the other TIS-DNN methods can achieve similar performance. Similar to the findings in previous studies [14,16], the method CSS performs even worse than the simple random method SRS, since the method CSS is not robust to the divergence between the original inputs and the selected subset.

Summary for RQ1: The previous TIS-DNN methods can achieve low accuracy estimation error on all the classes. However, the estimation error on each class is still high especially when the subset size is small, which can influence the test diversity of the subset.

6.2. Result analysis for RQ2

Method. To investigate RQ2, we first aim to construct subsets, which can cover different classes and the scale is fixed. We extract the same number of test inputs each time according to the specified number of classes to build a test subset. The subset construction method used in result analysis for RQ2 can be found in Table 3. For example, for the datasets with 10 classes, the size of the subset is fixed to 100. Then the number of the covered classes is increased from 1 to 10 and the step size is 1. For the dataset with 30 classes, the size of the subset is fixed to 300. Then the number of the covered classes is increased from 3 to 30 and the step size is 3. Since the existing results show that

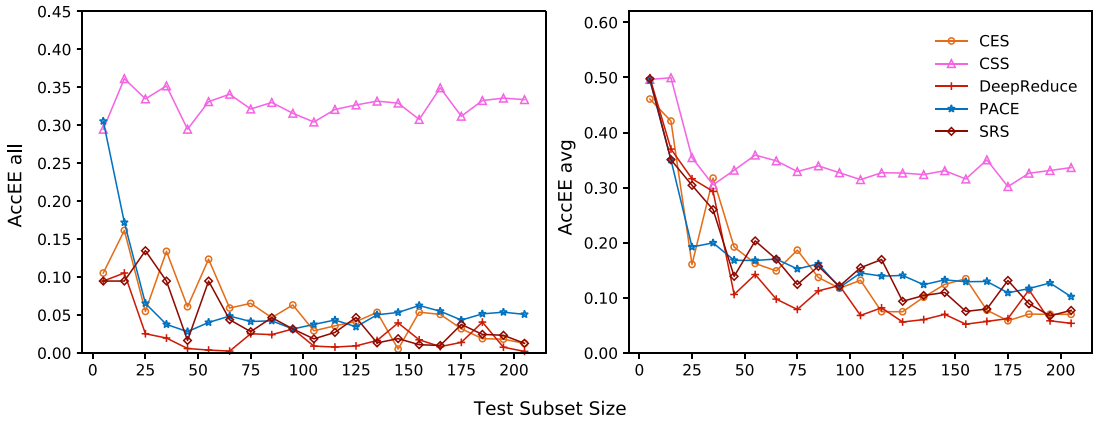


Fig. 5. Accuracy estimation error on LeNet-5 model for the Autogen-MNIST dataset when considering different subset size.

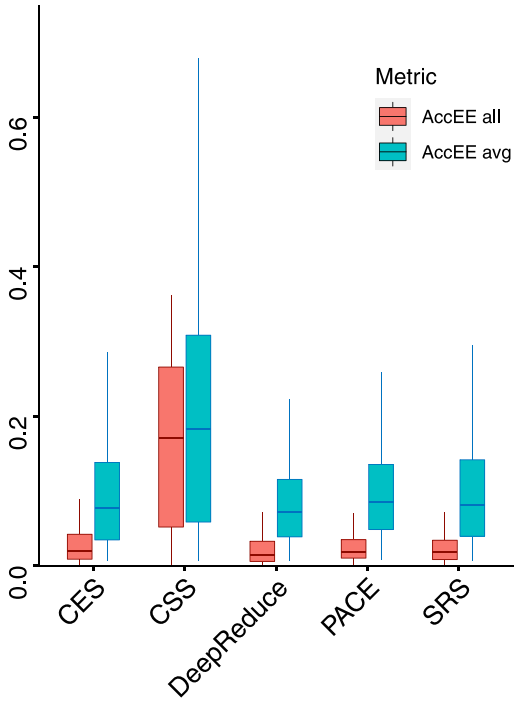


Fig. 6. $AccEE_{avg}$ and $AccEE_{all}$ value distribution on all the pairs via boxplot for different TIS-DNN methods.

the more classes are covered, the smaller value of $AccEE_{avg}$ is under the same test subset size. In this RQ, we want to analyze the correlation between $CovDiff$ and $AccEE_{avg}$, which means whether decreasing the value of $AccEE_{avg}$ can help to improve the coverage rate difference between the subset and the original test inputs when given a coverage criterion. Here we consider the five popular deep neuron network-based coverage criteria (i.e., NC, NBC, SNAC, TKNC, and KMNC) to calculate the test adequacy of deep learning testing. A brief introduction of these coverage criteria can be found in Section 5.3. Notice we do not consider the experimental protocol for subset construction in RQ1, since the motivations of RQ1 and RQ2 are different. In RQ1, we want to explore the performance differences between the existing TIS-DNN methods under different test subset sizes. While in RQ2, we want to explore the relationship between $AccEE_{avg}$ and five DNN-coverage estimation errors, which has little relationship with the test subset size. Moreover, the time cost of calculating KMNC metric on experiment subject 13 is very expensive. Therefore, we choose to conduct the experiment in RQ2 with a smaller and fixed test subset size.

Table 3
The subset construction method used in result analysis for RQ2.

Original class nums	Smallest class nums	Step	Fixed subset size
10	1	1	100
30	3	3	300
100	10	10	1000
1000	100	100	10000

Finally, we compute the correlation coefficient via the Pearson correlation coefficient between the accuracy estimation error on each class $AccEE_{avg}$ (denoted by the one-dimensional vector X) and the coverage difference $CovDiff$ when given a DNN-based coverage criterion (denoted by the one-dimensional vector Y). The Pearson correlation coefficient can be computed as follows.

$$r(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var[X] \cdot Var[Y]}} \tag{4}$$

where $Cov(X,Y)$ means the covariance of X and Y, and $Var[Y]$ denotes the variance of the vector Y.

Result. Due to the extra high complexity of the neuron network model (i.e., ResNet-50 trained by ImageNet), the cost of gathering the coverage rate of all these DNN models is unacceptable. Therefore, we only list the results of 17 pairs of the DNN models and corresponding test inputs in Table 4. In this table, ID corresponds to the specific pair of the DNN model and corresponding test inputs shown in Table 1. Then the correlation coefficients for different DNN-based coverage criteria between the accuracy estimation error on each class $AccEE_{avg}$ and the coverage difference between the subset and the original test inputs $CovDiff$ are shown in the remaining five columns respectively. The closer the absolute value of the correlation coefficient is to 1, the higher the correlation degree is. Here a positive value means a positive correlation, and a negative value means a negative correlation. In the last row, we gather the number of pairs, whose absolute value of the correlation coefficient is larger than 0.8 (i.e., strong correlation suggested by Mukaka et al. [40]). In this table, we can find among these five criteria, the criterion NC and the criterion TKNC shows the strong correlation with the estimation error on each class. These two criteria can achieve the strong correlation on 13 experimental subjects. That means for the subset, reducing accuracy estimation error on each class can help to accurate coverage rate estimation of the original inputs for these two criteria.

Table 4
The correlation coefficient between the accuracy estimation error on each class $AccEE_{avg}$ and the coverage difference between the subset and the original test set.

ID	NC	NBC	SNAC	TKNC	KMNC
1	**0.965**	N/A	**0.838**	0.745	0.770
2	**0.867**	−0.054	**0.807**	0.732	−0.258
3	**0.935**	−0.202	**0.876**	**0.865**	0.032
4	**0.845**	0.411	**0.838**	0.786	0.005
5	**0.919**	0.050	**0.906**	**0.947**	−0.147
6	**0.922**	0.032	**0.925**	**0.949**	−0.347
7	**0.990**	0.765	0.778	**0.904**	0.237
8	0.754	−0.412	−0.093	**0.917**	**0.883**
9	**0.961**	−0.043	0.378	**0.885**	0.049
10	**0.945**	**0.929**	**0.934**	**0.969**	0.542
11	0.724	0.202	0.219	0.796	0.676
12	**0.932**	0.759	0.762	**0.909**	0.588
13	0.535	−0.669	−0.489	**−0.852**	0.099
15	0.574	0.478	0.024	**0.883**	0.372
16	**0.861**	0.345	0.313	**0.828**	**−0.901**
17	**0.941**	−0.393	**0.894**	**0.983**	0.753
18	**0.956**	**0.949**	**0.956**	**0.948**	0.757
Sum	13/17	2/17	9/17	13/17	2/17

*Notice we do not show the result of the 14th experimental subject due to the high computational cost in gathering coverage information.

Summary RQ2: Reducing the accuracy estimation error on each class on the subset can help to achieve accurate coverage rate estimation of the original inputs based on DNN-based coverage criteria (especially for the criterion NC and the criterion TKNC).

6.3. Result analysis for RQ3

Method. In this RQ, we want to compute the performance improvement room for previous TIS-DNN methods in terms of accuracy-based performance measures. To answer this RQ, we first need to know the theoretically approximate optimal solution for different experimental subjects. In practice, during the usage of the TIS-DNN methods, the practitioners do not know the labels of the original test inputs and cannot find this optimal solution. However, in our controlled experimental study, we know the ground-truth labels of all the test inputs. Therefore, we can construct this optimal solution by a simple method **Best**. Specifically, given the size of the subset, this method first determines the sampling number $sampNum[i]$ for the ith class according to the proportion of each class in the original test inputs. Then for the ith class, it computes the sampling number of test inputs $sampNum_c[i]$ which are predicted correctly, and the sampling number of test inputs $sampNum_w[i]$ which are predicted wrongly based on the accuracy of the ith class of the original test inputs. Later, for the ith class, it samples $sampNum_c[i]$ test inputs with the correct predictions, and $sampNum_w[i]$ test inputs with the wrong predictions from the original inputs in the ith class. Finally, after iterating all the classes, it can determine the approximate optimal subset.

Based on the previous analysis on RQ1, we can find that except for the method CSS, the other TIS-DNN methods can achieve similar performance. Moreover, the method PACE can achieve better performance than other TIS-DNN methods in their original study [14]. Therefore, we mainly investigate the performance improvement room by comparing the method Best and the method PACE in this RQ.

Result. Final results (i.e., average value and standard deviation) can be found in Table 5. Taking the performance measure $AccEE_{all}$ as an example, the performance improvement room is obtained by subtracting the $AccEE_{all}$ value of the method PACE from the $AccEE_{all}$ value of the method Best. In this table, in terms of $AccEE_{all}$, the method PACE has 0.103% to 7.673% performance improvement room. In terms of $AccEE_{avg}$, the method PACE has 1.190% to 12.833% performance improvement.

Then, we analyze the effect of the class number on the performance improvement room. Different from the DNN models for image data type, the output of the DeepSpeech model is the probability distribution of the basic letters that may be included in the test instance. This makes the feature structure of the audio data is quite different from the image data. Therefore, We compute the average value and the standard deviation grouped by the class number when considering different data types and the results can be found in Table 6. Specifically, for the image data type, we can find with the number of the classes increases, the performance improvement room for $AccEE_{all}$ decreases, while the performance improvement room for $AccEE_{avg}$ generally increases.

Finally, we analyze the performance turning point (i.e., the performance will significantly decrease) of the method PACE on $AccEE_{avg}$ and its corresponding subset size for each pair. Specifically, when considering the image data type, for the datasets with 10 classes, the size of the corresponding subset is about 25 to 35. For the datasets with 100 classes, the size of the corresponding subset is about 250. For the datasets with 1000 classes, the size of the corresponding subset is about 2500. When considering the audio data type, for the datasets with 30 classes, the size of the corresponding subset is about 250. These findings can provide subset size recommendation for the practitioners, which can help to achieve the optimal $AccEE_{avg}$ when using PACE.

Summary for RQ3: The previous TIS-DNN methods have a large performance improvement room on all the classes and each class. For example, the method PACE has at most 7.673% and 12.833% performance improvement room in terms of $AccEE_{all}$ and $AccEE_{avg}$ respectively. Moreover, the performance improvement room and the performance turning point of the method PACE are related to the class number for the image data type.

7. Discussion & implications

In this section, we first want to investigate whether considering the test diversity can guarantee performance estimation on all the classes. We second introduce our proposed multi-objective optimization-based TIS-DNN method and show the preliminary results to verify the feasibility of this exploration direction.

7.1. Considering test diversity can guarantee performance estimation on all the classes

Based on our large-scale experimental study on 18 pairs of DNN models and the corresponding test inputs, we find although the previous state-of-the-art TIS-DNN methods can guarantee the accurate accuracy estimation of the target DNN models on all the classes, while the accuracy estimation on each class (i.e., test diversity) is still not satisfactory. Moreover, we find that accurately estimating the accuracy on each class will not significantly decrease the coverage rate of the original inputs based on recently proposed DNN-based coverage criteria, since the relevant performance measure $AccEE_{avg}$ can evaluate the similarity of test diversity between the selected subset and the original test inputs in a fine-grained way. As shown in Figs. 2–5, with the size of the subset increases, the changing trend of $AccEE_{All}$ is basically consistent with the trend of $AccEE_{avg}$ no matter how the test subset is constructed by the selection method.

Table 5
The performance improvement room after comparing the method Best and the method PACE on each experimental subject.

Measure	Subject ID	Improvement room (Average, Standard Deviation)					
Room for $AccEE_{all}$	1–6	(2.121%, 1.910%)	(0.870%, 0.693%)	(0.103%, 0.311%)	(1.139%, 4.024%)	(3.117%, 7.125%)	(0.126%, 0.514%)
	7–12	(2.780%, 4.142%)	(1.124%, 1.664%)	(1.349%, 0.608%)	(3.997%, 4.527%)	(0.514%, 1.029%)	(0.282%, 0.978%)
	13–18	(0.479%, 0.959%)	(0.719%, 0.549%)	(3.189%, 5.018%)	(7.673%, 8.416%)	(1.005%, 1.134%)	(1.660%, 1.696%)
Room for $AccEE_{avg}$	1–6	(3.728%, 4.367%)	(1.190%, 0.615%)	(1.218%, 2.040%)	(1.815%, 4.043%)	(2.153%, 6.220%)	(1.519%, 4.099%)
	7–12	(9.633%, 3.072%)	(7.939%, 2.847%)	(10.963%, 6.071%)	(7.230%, 4.466%)	(3.490%, 7.854%)	(3.540%, 2.639%)
	13–18	(9.713%, 4.710%)	(9.408%, 4.591%)	(12.833%, 2.832%)	(10.971%, 6.765%)	(11.423%, 4.756%)	(7.214%, 4.632%)

Table 6
The performance improvement room after comparing the method Best and the method PACE grouped by the class number.

Data type	# Class	Room for $AccEE_{all}$	Room for $AccEE_{avg}$
Image	10	(2.085%, 1.819%)	(5.886%, 2.333%)
	100	(1.349%, 0.608%)	(10.963%, 6.071%)
	1000	(0.599%, 0.654%)	(9.561%, 4.624%)
Audio	30	(0.514%, 1.029%)	(3.490%, 7.854%)

Table 7
The correlation between $AccEE_{avg}$ and $AccEE_{All}$ by adopting the **Best** method with the increasing of the subset size on 18 groups of experimental subjects.

Subject ID	Pearson correlation		
1–3	**0.943**	0.660	0.414
4–6	0.185	0.640	0.319
7–9	0.658	**0.962**	**0.948**
10–12	**0.958**	0.525	**0.967**
13–15	**0.985**	**0.919**	**0.963**
16–18	**0.940**	**0.933**	**0.947**

In this section, we quantitatively compute the correlation between $AccEE_{avg}$ and $AccEE_{All}$ by adopting the **Best** method (introduced in Section 6.3) with the increasing of the subset size (the detailed setting can be found in Section 6.1) on 18 groups of experimental subjects. The experimental results are shown in Table 7. In this table, we can find $AccEE_{avg}$ and $AccEE_{All}$ show a positive correlation on all of the experimental subjects and there is a strong positive correlation on 11 experimental subjects. These results show that reducing the accuracy estimation error on each class can help to achieve accurate accuracy estimation on all the classes. However, as analyzed in RQ1, only considering the reduction of the accuracy estimation error on all the classes cannot guarantee the test diversity of the subset. Therefore, keeping test diversity in the subset should be considered in future TIS-DNN studies.

7.2. Improving test diversity via multi-objective optimization

In our empirical studies, we find that there is still a lot of performance improvement room after analyzing the TIS-DNN method PACE [14]. Therefore, we should encourage researchers to design more advanced TIS-DNN methods, which can achieve accurate performance estimation both on all the classes and on each class.

In this section, we wish to design a novel TIS-DNN method via multi-objective optimization, which has been successfully used in some software engineering tasks (such as defect prediction [41,42], vulnerability identification [43]), to guarantee the test diversity in the subset. Before introducing this method, we first formalize TIS-DNN problem from the perspective of multi-objective optimization.

Definition 2 (*Multi-objective Optimization Based Test Input Selection for DNN*)**.** Given original test inputs T, the DNN model M to be tested, and the evaluation function $f_1, f_2, \ldots, f_s$, Suppose the selected test subset is denoted as T', the Multi-objective optimization based test input selection for DNN aims to make T' meet the following conditions:

$\|T'\| \ll \|T\|$: $f_1(T, M) \approx f_1(T', M)$, $f_2(T, M) \approx f_2(T', M)$, $\cdots$, $f_s(T, M) \approx f_s(T', M)$

In most of previous TIS-DNN methods, there is usually only one evaluation function f, that is, the overall accuracy of all test inputs (Notice, DeepReduce [17] takes neuron coverage as the second evaluation function). To ensure the test diversity, We resort to multi-objective optimization method and propose a novel method $MultiSelect$. Specifically, we first label T according to the prediction result of M, then we cluster T hierarchically and take the proportion of different classes in each cluster as the evaluation function $f_1, f_2, \ldots, f_s$. Later we continuously reduce the data distribution difference of each cluster between T and T' through multi-objective optimization. We hope T' can cover the classes of the original test inputs as much as possible and achieve the similar accuracy estimation on all the classes when compared to T.

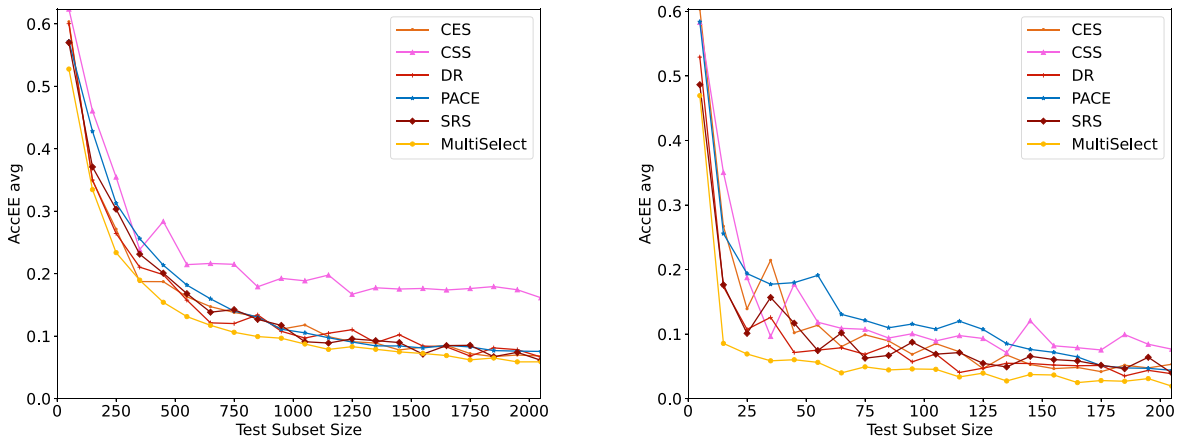
We conducted experiments on all the experimental subjects. The results of the subject 8 and the subject 9 are shown in Fig. 7. Notice that our proposed method $MultiSelect$ will generate multiple non-dominated solutions due to the nature of the multi-objective optimization. We choose the solution with the best performance as the final selection result in terms of the performance measure $AccEE_{avg}$. In this figure, we can find our proposed method can better guarantee the test diversity than state-of-the-art TIS-DNN baselines. More comparison results on other experimental subjects can be found in our project homepage.

8. Threats to validity

Internal threats. The first threat of internal validity lies in the implementation correctness of TIS-DNN baselines. For the baseline methods, we use the source code shared by previous TIS-DNN studies and use the optimal hyperparameter value suggested by these studies [14,16,17]. The second threat is the theoretically approximate optimal solution for different experimental subjects for result analysis for RQ3. Since we know the ground-truth labels of all the test inputs in our controlled experimental study, we can construct this optimal solution by our designed simple method **Best**.

External threats. The threat of external validity is related to the representative of our chosen experimental subjects. To alleviate this threat, we mainly consider the experimental subjects used to evaluate the effectiveness of the recently proposed TIS-DNN method PACE [14]. Since the test diversity property investigated in our empirical study is based on the classification task, we consider all the experimental subjects for the classification tasks. These experimental subjects can cover different model structures, different types of test inputs, different test input generation methods, and models with different performances. Therefore, the generalization of our experimental results can be guaranteed.

Construct threats. The construct threat in this study is the performance measure used to measure the test diversity of the selected subset



(a) The dataset CIFAR-100 with ResNet-20 model

(b) The dataset CIFAR-10 with ResNet-20 model

Fig. 7. Comparison between our proposed method *Multi-select* and the state-of-the-art TIS-DNN baselines in terms of $AccEE_{avg}$ performance measure.

Table 8
The brief textual definitions for acronyms and abbreviations.

Acronym/Abbreviation	Definition
DNN	Deep Neural Network
TIS	Test Input Selection
SRS	Simple Random Sampling
CSS	Confidence-based Stratified Sampling
CES	Cross-Entropy based Sampling
PACE	Practical Accuracy Estimation
NC	Neuron Coverage
KMNC	K-Multisection Neuron Coverage
NBC	Neuron Boundary Coverage
SNAC	Strong Neuron Activation Coverage
TKNC	Top-K Neuron Coverage
Acc_{sub}	the ACCuracy on the SUBset
Acc_{org}	the ACCuracy on the ORiGinal test inputs
$AccEE_{all}$	the ACCuracy Estimation Error on ALL the classes
$AccEE_{avg}$	the average value of ACCuracy Estimation Error on each class
Cov_{sub}	the COVerage rate on the SUBset
Cov_{org}	the COVerage rate on the ORiGinal test inputs
$CovDiff$	the COVerage rate DIFFerence between the selected subset and the original test inputs

by TIS-DNN methods. To alleviate this threat, we used $AccEE_{avg}$ as the simple proxy for test diversity since high accuracy difference in some classes can make the value of $AccEE_{avg}$ high, while the value of $AccEE_{all}$ can still low. Notice $AccEE_{all}$ and $AccEE_{avg}$ may have some redundancy when each class has the same number of instances and the same accuracy in the original set and the subset. However, this scenario rarely happens in our empirical study. In our study, for the subsets returned by the TIS-DNN methods, the number of instances contained in different classes may vary greatly. In the worst cases, instances in some classes do not even appear in subsets especially when the dataset has many different classes. Moreover, the accuracy of different classes may also vary greatly in our empirical study.

9. Conclusion and future work

Test input selection methods for deep neural networks have been proposed to reduce labeling costs. However, the selected subset may miss some classes of the inputs, which will reduce the test diversity of the original test inputs. We conducted a large-scale empirical study based on 18 DNN models and five state-of-the-art TIS-DNN baselines to investigate this issue. Empirical results show that the accuracy estimation on each class (i.e., test diversity) is still not satisfactory. Moreover, maintaining the test diversity in the subset will help to achieve accurate coverage rate estimation of the original inputs based on some DNN-based coverage criteria (especially for the criterion NC [19] and the criterion TKNC [20]). Since there has a large performance improvement room for current TIS-DNN methods, we hope that more effective TIS-DNN methods which can perform accurate performance estimation on all classes and each class can be designed in the future. In this study, we introduce our proposed multi-objective optimization-based TIS-DNN method and show the preliminary results to verify the feasibility of this research direction.

In the future, we first want to investigate the generalization of our empirical results by considering more DNN models from other application domains (such as text mining). We second want to design more effective methods, which can guarantee the test diversity of TIS-DNN. Finally, we want to analyze the test diversity problem for TIS-DNN in the finer granularity. For example, we can retrieve a vector of accuracy (one element for each class) and then analyze the vector component by component.

CRediT authorship contribution statement

Chunyu Zhao: Methodology, Software, Writing – original draft. **Yanzhou Mu:** Methodology, Software, Writing – original draft. **Xiang Chen:** Methodology, Conceptualization, Supervision, Writing – review & editing. **Jingke Zhao:** Software, Validation. **Xiaolin Ju:** SupervisTest Diversition, Writing – review & editing. **Gan Wang:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Chunyu Zhao and Yanzhou Mu have contributed equally to this work and they are co-first authors. This work is supported in part by the National Natural Science Foundation of China (Grant No. 61872263), The Open Project of Key Laboratory of Safety-Critical Software for Nanjing University of Aeronautics and Astronautics, Ministry of Industry and Information Technology, China (Grant No. NJ2020022), The Open Project of State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), China (Grant No. 2020-MS-07), and the Nantong Application Research Plan, China (Grant No. JC2021124).

Appendix

In the appendix, we introduce the brief textual definitions for acronyms and abbreviations in our article for the convenience of readers (see Table 8).

References

[1] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, Deep learning for computer vision: A brief review, Comput. Intell. Neurosci. 2018 (2018).
[2] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, Ieee Comput. Intell. Mag. 13 (3) (2018) 55–75.
[3] Y. Yang, X. Xia, D. Lo, J. Grundy, A survey on deep learning for software engineering, 2020, arXiv preprint arXiv:2011.14597.
[4] K. Cao, C. Chen, S. Baltes, C. Treude, X. Chen, Automated query reformulation for efficient search based on query logs from stack overflow, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering, ICSE, IEEE, 2021, pp. 1273–1285.
[5] X. Chen, C. Chen, D. Zhang, Z. Xing, Sethesaurus: Wordnet in software engineering, IEEE Trans. Softw. Eng. (2019).
[6] W. Zheng, J. Gao, X. Wu, F. Liu, Y. Xun, G. Liu, X. Chen, The impact factors on the performance of machine learning-based vulnerability detection: A comparative study, J. Syst. Softw. 168 (2020) 110659.
[7] D. Chen, X. Chen, H. Li, J. Xie, Y. Mu, Deepcpdp: Deep learning based cross-project defect prediction, IEEE Access 7 (2019) 184832–184848.
[8] T. Zhou, X. Sun, X. Xia, B. Li, X. Chen, Improving defect prediction with deep forest, Inf. Softw. Technol. 114 (2019) 204–216.
[9] Z. Li, Y. Wu, B. Peng, X. Chen, Z. Sun, Y. Liu, D. Yu, SeCNN: A semantic CNN parser for code comment generation, J. Syst. Softw. 181 (2021) 111036.
[10] J.M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, IEEE Trans. Softw. Eng. (2020).
[11] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, S. See, 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, 2019, pp. 146–157.
[12] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Concolic testing for deep neural networks, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 109–119.
[13] S. Wang, N. Shrestha, A.K. Subburaman, J. Wang, M. Wei, N. Nagappan, Automatic unit test generation for machine learning libraries: How far are we? in: 2021 IEEE/ACM 43rd International Conference on Software Engineering, ICSE, IEEE, 2021, pp. 1548–1560.
[14] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, M. Yan, Practical accuracy estimation for efficient deep neural network testing, ACM Trans. Softw. Eng. Methodol. (TOSEM) 29 (4) (2020) 1–35.
[15] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, W. Zhang, Prioritizing test inputs for deep neural networks via mutation analysis, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering, ICSE, IEEE, 2021, pp. 397–409.
[16] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, J. Lü, Boosting operational DNN testing efficiency through conditioning, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 499–509.
[17] J. Zhou, F. Li, J. Dong, H. Zhang, D. Hao, Cost-effective testing of a deep learning model through input reduction, in: 2020 IEEE 31st International Symposium on Software Reliability Engineering, ISSRE, IEEE, 2020, pp. 289–300.
[18] H. Zhu, P.A. Hall, J.H. May, Software unit test coverage and adequacy, Acm Comput. Surv. (Csur) 29 (4) (1997) 366–427.
[19] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, in: Proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 1–18.
[20] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, et al., Deepgauge: Multi-granularity testing criteria for deep learning systems, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 120–131.
[21] J. Kim, R. Feldt, S. Yoo, Guiding deep learning system testing using surprise adequacy, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, 2019, pp. 1039–1049.
[22] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, J. Zhao, Deepstellar: Model-based quantitative analysis of stateful deep learning systems, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 477–487.
[23] F. Zhang, S.P. Chowdhury, M. Christakis, Deepsearch: A simple and effective blackbox attack for deep neural networks, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 800–812.
[24] B. Biggio, F. Roli, Wild patterns: Ten years after the rise of adversarial machine learning, Pattern Recognit. 84 (2018) 317–331.
[25] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
[26] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: 2016 IEEE European Symposium on Security and Privacy, EuroS&P, IEEE, 2016, pp. 372–387.
[27] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: A simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.
[28] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy, SP, IEEE, 2016, pp. 582–597.
[29] T. Byun, V. Sharma, A. Vijayakumar, S. Rayadurgam, D. Cofer, Input prioritization for testing neural networks, in: 2019 IEEE International Conference on Artificial Intelligence Testing, AITest, IEEE, 2019, pp. 63–70.
[30] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, Z. Chen, 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, 2020, pp. 177–188.
[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.
[32] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, 2016, arXiv preprint arXiv:1607.02533.
[33] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, et al., Comparison of learning algorithms for handwritten digit recognition, in: International Conference on Artificial Neural Networks, Vol. 60, Perth, Australia, 1995, pp. 53–60.
[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, in: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[35] M.J. Harrold, R. Gupta, M.L. Soffa, A methodology for controlling the size of a test suite, ACM Trans. Softw. Eng. Methodol. (TOSEM) 2 (3) (1993) 270–285.
[36] B. Kim, R. Khanna, O.O. Koyejo, Examples are not enough, learn to criticize! criticism for interpretability, Adv. Neural Inf. Process. Syst. 29 (2016).
[37] T.Y. Chen, H. Leung, I. Mak, Adaptive random testing, in: Annual Asian Computing Science Conference, Springer, 2004, pp. 320–329.
[38] F.Y. Harel-Canada, L. Wang, M.A. Gulzar, Q. Gu, M. Kim, Is neuron coverage a meaningful measure for testing deep neural networks? in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020.
[39] J. Chen, M. Yan, Z. Wang, Y. ning Kang, Z. Wu, Deep neural network test coverage: How far are we? 2020, arXiv, arXiv:2010.04946.
[40] M.M. Mukaka, A guide to appropriate use of correlation coefficient in medical research, Malawi Med. J. 24 (3) (2012) 69–71.
[41] X. Chen, Y. Zhao, Q. Wang, Z. Yuan, MULTI: Multi-objective effort-aware just-in-time software defect prediction, Inf. Softw. Technol. 93 (2018) 1–13.
[42] C. Ni, X. Chen, F. Wu, Y. Shen, Q. Gu, An empirical study on pareto based multi-objective feature selection for software defect prediction, J. Syst. Softw. 152 (2019) 215–238.
[43] X. Chen, Y. Zhao, Z. Cui, G. Meng, Y. Liu, Z. Wang, Large-scale empirical studies on effort-aware security vulnerability prediction methods, IEEE Trans. Reliab. 69 (1) (2019) 70–87.