International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2022, 12(4): 403–436, doi: 10.21655/ijsi.1673-7288.00282
©2022 by Institute of Software, Chinese Academy of Sciences. All rights reserved.

Research Article

Deep Learning Test Optimization Method Using Multi-objective Optimization

Yanzhou Mu (沐燕舟)[1], Zan Wang (王赞)[1,3], Xiang Chen (陈翔)[2],
Junjie Chen (陈俊洁)[1], Jingke Zhao (赵静珂)[3], Jianmin Wang (王建敏)[4]

[1] (College of Intelligence and Computing, Tianjin University, Tianjin 300350, China)
[2] (School of Information Science and Technology, Nantong University, Nantong 226019, China)
[3] (School of New Media and Communication, Tianjin University, Tianjin 300072, China)
[4] (Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China)
Corresponding author: Zan Wang, wangzan@tju.edu.cn

Abstract With the rapid development of deep learning technology, research on its quality assurance is raising more attention. Meanwhile, it is no longer difficult to collect test data owing to the mature sensor technology, but it costs a lot to label the collected data. To reduce the cost of labeling, the existing studies attempt to select a test subset from the original test set. The test subset, however, only ensures that the overall accuracy (the accuracy of the target deep learning model on all test inputs of the test set) of the test subset is similar to that of the original test set; it cannot maintain other test properties similar to those of the original test set. For example, it cannot fully cover all kinds of test input in the original test set. This study proposes a method based on multi-objective optimization called Deep Multi-Objective Selection (DMOS). It firstly analyzes the data distribution of the original test set by Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Then, it designs multiple optimization objectives given the characteristics of the clustering results and then carries out multi-objective optimization to find out the appropriate selection solution. Massive experiments are carried out on eight pairs of classic deep learning test sets and models. The results reveal that the best test subset selected by the DMOS method (the test subset corresponding to the Pareto optimal solution with the best performance) can not only cover more test input categories in the original test set but also estimate the accuracy of each test input category extremely close to that of the original test set. Meanwhile, it can also ensure that the overall accuracy and test adequacy are close to those of the original test set: the average error of the overall accuracy estimation is only 1.081%, which is 0.845% lower than that of Practical ACcuracy Estimation (PACE), an improvement of 43.87%. The average error of the accuracy estimation of each test input category is only 5.547%, which is 2.926% less than that of PACE, an improvement of 34.53%. The average estimation error of the five test adequacy measures is only 8.739%, which is 7.328% lower than that of PACE, an improvement of 45.61%.

This is the English version of Chinese article “采用多目标优化的深度学习测试优化方法. 软件学报, 2022, 33(7): 2499-2524. doi: 10.13328/j.cnki.jos.006583”

Funding items: National Natural Science Foundation of China (61872263); Basic Strengthening Program on Technical Fields (2020-JCJQ-JJ-490); 2020 Tianjin Special Programs for Intelligent Manufacturing of China (20201180)
Received 2021-09-05; Revised 2021-10-14; Accepted 2022-01-10; IJSI published online 2022-12-28

Keywords deep learning; software testing; test input selection; multi-objective optimization; genetic evolution

Citation Mu YZ, Wang Z, Chen X, Chen JJ, Zhao JK, Wang JM. Deep learning test optimization method using multi-objective optimization, *International Journal of Software and Informatics*, 2022, 12(4): 403–436. http://www.ijsi.org/1673-7288/282.htm

With the rapid development of Deep Learning (DL), Deep Neural Network (DNN) models trained on the basis of massive data have achieved excellent performance in more and more applications[11], such as automatic driving[1], face recognition[2], speech recognition[3, 4], medical diagnosis[5], aircraft collision avoidance systems[6], and software engineering[7–10]. However, like traditional software systems, DNN models still have defects. Due to their complex internal structures, such models may make incomprehensible mispredictions under any small data disturbance. This will lead to serious consequences in security-critical areas. For example, in 2018, a pedestrian was killed by Uber's self-driving car in Tempe, Arizona (https://www.vice.com/en/article/9kga85/uber-is-giving-up-on-self-driving-cars-in-California-after-deadly-crash). A Tesla Model S in the autopilot mode crashed into a fire truck with flashing lights on a freeway in California in 2018 (https://www.newsweek.com/autonomous-tesla-crashes-parked-fire-truck-california-freeway-789177). Therefore, it is crucial to ensure the quality of DNN models. As a conventional quality assurance method, software testing can effectively detect inherent defects in a system to be tested. Thus, designing an effective testing method for DNN models has been a research hotspot in the fields of deep learning and software testing in recent years.

Consistent with traditional software testing, DL testing also aims to efficiently and adequately expose defects in the DNN model to be tested. However, as a DNN model is data-driven, its final predictions are determined by the weight of each neuron obtained during training, as well as test input. Therefore, for DL testing, maintaining a high-quality test set is an important requirement to achieve the above goal. With the rapid development of sensor technology, it is no longer difficult to collect massive data in a short time, but this also greatly increases the testing cost of a DNN model. For example, Facebook's face recognition system DeepFace[12] used about 220,000 face images for testing; DeepTest[13] tested a CNN-based self-driving model with 254,221 images; the popular dataset ImageNet[14] contains 100,000 test images (namely, a total of 1,000 categories, each containing 100 images) for testing various image classification models. Although massive test data can ensure complete testing of a DNN model, a great amount of testing time will be consumed. In addition, massive raw data collection need to be correctly labeled before being used for testing. At present, labeling is mainly done manually, and to ensure the correctness of labeling, multiple users are generally required to cooperate to complete labeling, which consumes a lot of manpower and material resources. Particularly, domain-specific test data (such as images from the medical field) need to be labeled by experts in this regard, which further raises the labeling cost. Thus, redundant test data will not only waste colossal labeling costs but also greatly reduce the efficiency of testing. In cooperation with enterprises, we find that the labeling of test sets has become one of the serious challenges burdening them in testing DNN models. Therefore, maintaining a small-scale test set with a good test capability has become an important research issue in DL testing. The DL test input selection method aims to select representative test input (namely, data able to maintain properties of original test sets) in original unlabeled test sets and then label constructed test subsets to reduce the costs of overall labeling and testing.

Research on this problem is still in its infancy. Li *et al.*[15] first proposed a Confidence-based Stratified Sampling (CSS) method. The method adopts a DNN model to divide the predicted confidence of each test input into intervals and then conducts sampling in each interval according to a certain proportion until reaching the labeled number specified by users. However, the CSS method is only suitable for models with high accuracy. Later, Li *et al.*[15] proposed a Cross Entropy-based Sampling (CES) method, which first randomly samples data in an original test set and then iterates the process continuously. With the optimization objective of reducing cross entropy between the test subset and the original test set, the iteration terminates when the specified threshold is reached. Compared with CSS, CES has better generalization performance; however, affected by randomness, it is of poor stability and thus can hardly be applicable to actual scenarios. Zhou *et al.*[16] presented the two-stage selection method DeepReduce based on two objectives: firstly, on the basis of Neuron Coverage (NC), a test subset is selected by a greedy strategy, and the neuron output of the last layer of the DNN model to be tested is extracted as the feature representation of each input. Then, with the optimization objective of minimizing relative entropy between the test subset and the original test set, samples are heuristically selected from the original test set and added to the test subset until the specified threshold is reached. Chen *et al.*[17] developed a Practical ACcuracy Estimation (PACE) method. In this method, an original test set is first clustered by Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)[18]. Then, the clustered results are sampled according to corresponding proportions by MMD-critic-based prototype sampling[19] and adaptive random selection[20] separately. Finally, the samples are merged to form the test subset. Tests showed that PACE can effectively reduce the size of the original test set and accurately estimate the overall accuracy of the DNN model to be tested. In other words, PACE can achieve the overall accuracy close to that of the original test set using only about 2% of the test input samples selected from the original test set, with an average error of only 1.181%–2.302%. However, it fails to guarantee that the selected test subset can cover different categories of test input in the original test set, and it produces a huge error in estimating the accuracy of the test input of each category. For example, according to the test results, in the test set CIFAR100 with 100 categories, under a sampling rate of 1.5%, the test subset selected by PACE will miss 28% of the test-input categories in the original test set. In addition, for the DNN model ResNet20 to be tested, the accuracy estimation of the test input of each category in the test subset selected by PACE has an average error as high as 43%.

If a selected new test subset is only close to the original test set in terms of overall accuracy, it will probably fail to cover all categories of test input in the original test set and lose other test properties of the original test set on the DNN model to be tested. The quality evaluation of a test subset needs to examine whether the test subset can fully possess multiple attributes and characteristics of the original test set. Therefore, we hope that a selected test subset should not only cover all test-input categories in the original test set but also enable precise accuracy estimation of test input in corresponding categories (namely, reaching an accuracy level close to that of the original test set). To this end, this paper proposes a Deep Multi-Objective Selection (DMOS) method. Specifically, DMOS first extracts the features of each test input in an original test set through the DNN model to be tested and conducts hierarchical clustering through HDBSCAN[18] to ensure sampling quality as much as possible. Then, with the objective of minimizing the difference between the original test set and the test subset in terms of category distribution in each cluster generated by clustering, the classical multi-objective optimization algorithm NSGA-II[21] is employed for solutions. Given the number of selections specified by users, if the selected test-input samples are biased to some cluster, samples in other clusters will be reduced, which enlarges the difference between the proportions of other clusters in the selected test subset and those in the original test set. Clearly, there is a conflict among these optimization objectives. Finally, DMOS returns a non-dominated solution set, and each solution in this set represents a specific selection scheme, the constituent element of which represents the weight of selecting the test input at this location. Users can choose appropriate solutions according to their preference for test requirements to form a final subset for labeling, so as to save labeling and testing costs. On the basis of ensuring overall accuracy, this method enables the selected test subset to cover different categories of test input in the original test set more completely and estimate the accuracy of test input in each category more precisely.

We carried out empirical research on eight test objects constructed by the combination of classical DL test sets and DNN models. DNN models in the eight test objects are all used for classification. In terms of overall accuracy, these models can be classified into high-precision (namely that overall accuracy is more than 0.8) and low-precision (namely that overall accuracy is not more than 0.8) models. In terms of the hierarchical structure of networks, these models can be classified into Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models. Data sets in the test objects can be divided into speech and image data in terms of sample types. The test results reveal that test subsets selected by DMOS are significantly better than those selected by the latest method PACE regarding overall-accuracy estimation and accuracy estimation of the test input of various categories[17]. In addition, the test adequacy of test subsets selected by DMOS is more similar to that of the original test sets. Specifically, the average error of DMOS in overall-accuracy estimation is only 1.081%, which is 0.845% lower than that of PACE, an improvement of 43.87%. The average error of DMOS in accuracy estimation of test input of various categories is only 5.547%, which is 2.926% lower than that of PACE, an improvement of 34.53%. The average error of DMOS in estimating five test adequacy measures is only 8.739%, which is 7.328% lower than that of PACE, an improvement of 45.61%.

In conclusion, the main contributions of this paper can be summarized as follows:

(1) We model DL test input selection as a multi-objective optimization problem for the first time and propose DMOS for solutions. DMOS solves the problem by the clustering method HDBSCAN[18] and the classical multi-objective optimization algorithm NSGA-II[21]. According to the test results, non-dominated solution sets returned by DMOS can ensure that the selected test subsets can cover test-input categories in original test sets as many as possible and accurately estimate the accuracy of the test input of each category. This method can effectively reduce the cost of DNN model testing and improve testing efficiency.

(2) We conduct in-depth experiments on eight groups of classical DNN models and DL test sets and verify the effectiveness of DMOS in different types of data sets and models.

(3) We implement DMOS on top of the Keras 2.3.1 and the TensorFlow 1.15.0 frameworks and shared it on internet (https://github.com/EzioQR/ DMOS2021) to facilitate follow-up research by other researchers.

1 Background Knowledge

1.1 DL testing and optimization

A DNN model consists of multiple layers, and each layer contains many neurons[22]. Interlayer neurons are connected by weighted links, and the weights of these links are calculated according to the training data and training processes. Given these calculated weights, the DNN model maps input to output. At present, DNN models are mainly classified into CNN and RNN models. Involving convolution calculation, a CNN model is usually used to process data with mesh topology (such as images)[23], while an RNN model usually makes processing decisions on the basis of information calculated by existing data, which is more suitable to deal with varying time-series information, such as speech, natural language, and other data containing sequence information[24]. Software testing is a common method used to ensure the quality of a DNN model[25, 26]; similarly to traditional software testing, DNN model testing also concerns test input and test oracles. By test input in DNN-model testing is meant various data collected to test model performance, such as speech, images, and texts, and test oracles are mainly manual category labels. In other words, each test input requires manual labeling of its basic facts. By comparing labeled actual types with predicted ones, researchers can judge whether a DNN model correctly predicts test input.

Nevertheless, it is costly to label massive test input collected. To improve the efficiency of DNN model testing, researchers have proposed two kinds of optimization methods.

1.1.1 *The first kind starts with test input selection*

Methods of this kind aim to select a small-scale test subset to accurately estimate the overall accuracy of the test set. Users can only label such test input to complete testing, effectively saving labeling costs. In the earlier research, Li *et al.*[15] proposed CSS and CES. The CSS method sorts the prediction of all test input in an original test set through a DNN model to be tested and divides it into intervals accordingly. Then, given the specified number of selections, the method samples data hierarchically and merges the samples to form the final test subset. The CES method selects the test input by continuously reducing the cross entropy between the test subset and the original test set on the basis of the output of the last hidden layer of the DNN model to be tested. Zhou *et al.*[16] proposed a two-stage selection method of sequential sampling, DeepReduce. Guided by NC, this method first selects a test subset from the original test set by a greedy strategy; then, it continuously reduces the relative entropy between the test subset and the original test set according to the output of the last layer of the DNN model to be tested until the specified number of selections is reached. Recently, Chen *et al.*[17] proposed the PACE method based on clustering. This method clusters test input according to their features extracted by the DNN model to be tested. Then, test input in normal-point and abnormal-point clusters obtained by clustering is sampled by MMD-critic-based prototype sampling[19] and adaptive random selection[20], respectively. Finally, the samples from various clusters are merged to form the final test subset.

1.1.2 *The second kind starts with test input sorting*

Methods of this kind aim to sort all test input according to prediction error probabilities of DNN models to be tested. Compared with the first kind, this kind does not require the discarding of any test input, and users can find test input that will cause mispredictions of models earlier. Feng *et al.*[27] presented a test input sorting method called Deepgini. According to the confidence of the DNN model to be tested with regard to test input predictions, this method calculates Gini purity[28] for sorting. Zhang *et al.*[29] proposed sorting test input by calculating its noise sensitivity. Their findings indicated that added with the same noise, test input with high noise sensitivity is more likely to deceive a DNN model than that with low noise sensitivity. Furthermore, Ma *et al.*[30] proposed a group of indicators for model confidence on the basis of specific test input to guide the selection of test input that is more likely to be misclassified, which is similar to the goal of the test sorting mentioned above.

The method of this paper belongs to the first kind, namely that it attempts to select a new test subset to replace the original test set for DNN model testing.

1.2 Multi-objective optimization

Multi-objective Optimization Problems (MOPs) can be seen in many scenarios of daily life, such as job planning[31] and financial portfolio management[32], which involve the simultaneous optimization of multiple objectives. Due to conflicts among some optimization objectives, a MOP may not have an optimal solution but a group of compromised solutions (namely, a Pareto optimal solution set). Its formal definition is expressed as[33]

$$\underset{x\in S\subset R}{\mathrm{opt}}\, f(x)=\mathrm{opt}\left(f_1(x),f_2(x),\cdots,f_M(x)\right) \tag{1}$$

where x is an S-dimensional decision vector in the search space; $f_m(x)$ is the mth objective to be optimized ($m=1,\cdots,M$), and it may seek maximization ($\mathrm{opt}=\max$) or minimization ($\mathrm{opt}=\min$); M is the total number of optimization objectives. In addition, for MOPs, some related terms[34] are defined as follows.

Definition 1 (Pareto dominance)**.** For two different solutions x_1 and x_2, if none of the objectives achieved by x_1 is inferior to those achieved by x_2, and x_1 has at least one objective achieved superior to objectives achieved by x_2, then, x_1 dominates x_2, or x_1 has Pareto dominance over x_2.

Definition 2 (Pareto optimal solution set)**.** The set in which all global Pareto optimal solutions are included is called the Pareto optimal Solution set (PS).

Definition 3 (Pareto optimal front)**.** In a Pareto optimal solution set, the image corresponding to each solution in the objective space is called the Pareto optimal Front (PF).

Pareto-based Multi-Objective Evolutionary Algorithms (MOEAs) are generally used to solve MOPs. These algorithms first select non-dominated solutions with good convergence through the Pareto-dominance principle and then further select non-dominated solutions according to density to ensure the quality of the final solution set. In the past 20 years, researchers have proposed many MOEAs to solve MOPs, including the Non-dominated Sorting Genetic Algorithm II (NSGA-II)[21], the Strength Pareto Evolutionary Algorithm 2 (SPEA2)[35], and the decomposition-based MOEA (MOEA/D)[36]. In this paper, an MOEA is used for solutions in that MOEA aims not only to find a solution close to the PF (i.e., convergence performance) but also to find a uniformly and widely distributed solution.

2 DMOS

2.1 Research motivation

When measuring the difference in testing capabilities between a selected test subset and the original test set, existing methods usually only consider the overall accuracy of the test sets, which will make the selected test subset lose other test properties of the original test set. According to the test results, although the test subsets selected by the existing methods can accurately estimate the overall accuracy of original test sets under different numbers of selections, they are likely to miss some categories of test input in the original test sets and have large errors in accuracy estimation of the test input of various categories covered. For example, when PACE[17] runs on the test object constructed by the CIFAR100 test set and the ResNet20 model, in the case of the sampling quantity accounting for 1.5% of the original test set, the overall accuracy estimation error is only 3.9%. However, in such a case, the selected test subset omits 28% of the test input categories in the original test set, and the average error in the accuracy estimation of the test input of various categories reaches 43%. Therefore, under the only consideration of overall accuracy, the selected test subsets probably fail to effectively represent the original test sets. Feasible test set replacement means that selected subsets should have multiple attributes and characteristics of the original test sets, including similar sample diversity. In recent years, many techniques for measuring the diversity of test sets have emerged in traditional software testing[37], and these studies show that the effectiveness of test sets benefits from the diversity of samples to a great extent. Thus, from the point of view of ensuring sample diversity and model testing effects, we believe that the selected new test subsets should not only cover test input categories in original test sets as many as possible but also ensure approximate accuracy to original test sets in each category.

Given the above analysis, we hope that test subsets selected by DMOS can cover as many test input categories in original test sets as possible and precisely estimate the accuracy of the test input of various categories. In order to obtain such properties, DMOS first estimates the distributions of the test input of various categories in original test sets through a clustering algorithm. Then, with the optimization objective of minimizing the difference between various clusters in terms of distribution of data in various categories, DMOS uses a multi-objective optimization algorithm for solutions and finally obtains approximate optimal selection schemes close to expected objectives.

2.2 Problem modeling

The existing methods generally assess the test capability difference between original test sets and test subsets on the basis of the overall accuracy of the test sets. DeepReduce proposed by Zhou *et al*.[16] attempts to use multiple optimization objectives as the basis for quality assessment of a selected subset. The research problem is modeled below.

Definition 4 (modeling of DL test input selection based on multi-objective optimization)**.** For a DNN model M to be tested and an original unlabeled DL test set T, we denote $f_1(T,M), f_2(T,M), \cdots, f_s(T,M)$ as s objective functions to assess the performance of M over T; let T' be the selected test subset. Then, the DL test input selection aims to make T', with $\|T'\| \ll \|T\|$:

$$f_1(T,M) \approx f_1(T',M), f_2(T,M) \approx f_2(T',M), \cdots, f_s(T,M) \approx f_s(T',M).$$

During the specific implementation of DMOS, as the original test data are unlabeled, the labels predicted by the DNN model to be tested are used to distinguish test input categories. After the clustering of the original test set, iterative problem solving is conducted with the sum of errors in the proportions of the different categories of data in the corresponding clusters of both the original test set and the test subset as the optimization objective. This aims to reduce the distribution difference of the same cluster between the two sets to ensure consistent distributions of the two sets. If optimization objectives are designed directly according to model-predicted label categories without the use of the clustering method, the accuracy of the DNN model to be tested will affect the performance of the method (a low-precision model will have a large deviation in estimated categories by test input in the original test set). The clustering method can preliminarily estimate the samples of each category in the original test set without labels. Thus, the purpose of using such a method is to correct deviations caused by DNN models of different precision so that subsequent multi-objective optimization can be carried out with relatively reliable data. The multi-objective optimization through DMOS can be expressed by the following MOP:

$$\begin{cases} \min & y_k = \mathit{Ratio}_{\mathrm{diff}}(x, \mathit{cluster}_k) & \forall k \in \{1, \ldots, m\} \\ \text{s.t.} & x = (x_1, \ldots, x_n) & \\ & x_i \in [1, 100] & \forall i \in \{1, \ldots, n\} \end{cases} \tag{2}$$

where x is an n-dimensional vector (n is the size of the original test set); it represents a specific selection scheme for the original test set, and each element of the vector represents the weight of selecting the test input at the corresponding position, which is a real number between 1 and 100. According to the number of selections specified by users, several elements with the largest weights are selected by sorting to form a new test subset. $\mathit{cluster}_k$ represents the kth cluster after the clustering of the original test set:

$$Ratio_{\text{diff}}(x, cluster_k) = \sum_{i=1}^{L} |l_i - l_i'| \tag{3}$$

Suppose that m clusters are formed after the clustering; y_k represents the sum of the differences between the test subset selected by the selection scheme x and the original test set in proportions of the test input of each category in Cluster k. It can be calculated by Eq. (3), where L is the total number of different categories in the original test set, predicted by the DNN model to be tested; l_i represents the proportion of the samples belonging to Category l to the total samples of Cluster k in the original test set, while l_i' represents that in the test subset. Specifically, assume that the DNN model to be tested predicts 10 categories in the original test and that there are 100 test input samples in Cluster 1 of the original test set after clustering; the number of samples belonging to the 10 model-predicted categories in Cluster 1 of the original test set is 65, 5, 3, 1, 5, 1, 5, 5, 5, and 5, with the corresponding proportions being 65%, 5%, 3%, 1%, 5%, 1%, 5%, 5%, 5%, and 5%, respectively. Cluster 1 in the test subset formed according to population individuals only contains 10 test input samples, and the number of samples belonging to the 10 model-predicted categories in Cluster 1 of the test subset is 0, 0, 1, 9, 0, 0, 0, 0, 0, and 0 (namely that only two categories of test input samples are selected), with the corresponding proportions being 0%, 0%, 1%, 9%, 0%, 0%, 0%, 0%, 0%, and 0%, respectively. Thus, the calculated optimization objective of Cluster 1 is 1.06, i.e., the sum of the absolute values of differences between the two sets in proportions of data of various categories. Optimization objectives of other clusters can be calculated similarly.

During the optimization, we hope that the multi-objective optimization algorithm can effectively coordinate the relationship between corresponding optimization objectives of each cluster so that the obtained selection scheme can make the data distribution of the generated test subset as close as possible to that of the original test set in each cluster. Intuitively, DMOS is to estimate the data distribution of the original test set by clustering and then employ multi-objective optimization to continuously reduce the distribution difference between the selected test subset and the original test set, so as to ensure the similarity of their properties.

2.3 Algorithm process

2.3.1 *Overall framework: a brief introduction to input and output parameters and the overall process*

The input of DMOS includes a DNN model D to be tested, an original test set T containing a total of t test input samples, and the number n of test input selections specified by users. First, DMOS uses the test input features extracted by D from T as the basis for clustering. Then, it iteratively finds a selection solution with Pareto optimality by using the predicted results of D as the category labels of the test input and optimizing the difference between the test subset and the original test set in the proportions of data of various categories in each cluster. Finally, the method returns a selected test subset X from T according to users' test requirements. The specific process of DMOS is shown in Algorithm 1, and the workflow is illustrated in Fig. 1.

2.3.2 *Data preprocessing: feature extraction, feature dimension reduction, feature value standardization, etc.*

DMOS first extracts the feature representation of each test input sample (Lines 2–4 in Algorithm 1) and then executes preprocessing operations such as feature dimension reduction and feature value standardization (namely, Min-Max normalization) (Line 5). As data in T is not labeled, to distinguish categories among different test input samples during selection, DMOS uses predicted results of D as category labels of test input samples in T (Line 6).

Algorithm 1. Process of DMOS.

```
Input: a DNN model $D$ to be tested;
       an original test set $T$ from which samples are to be selected, with its size denoted as $st$;
       the number of samples selected from $T$ specified by users, denoted as $n$.
Output: a new sample set $X$ formed by selection of size $n$.
1.  $X \leftarrow \varnothing$
2.  foreach $t_i$ in $T$ do
3.      $f_i \leftarrow extractFeatures(t_i, D)$ //Extract feature vectors of test input
4.  end foreach
5.  $\{f'_1, f'_2, \cdots, f'_{st}\} \leftarrow Preprocess(f_1, f_2, \cdots, f_{st})$ //Preprocess feature vectors of test input
6.  predictLabels $\leftarrow D.predict(T)$ //Obtain predicted labels of all test input samples in $T$
7.  $G \leftarrow \{g_1, g_2, \cdots, g_m\} \leftarrow cluster(f'_1, f'_2, \cdots, f'_{st})$ //Classify all test input samples into $m$
    clusters according to their feature vectors
8.  totalLabelsProportion $\leftarrow \varnothing$ //Store the proportions of samples of each category in all clusters to
    the total samples of the current cluster
9.  foreach $g_k$ in $G$ do
10.     singleClusterProportion $\leftarrow \varnothing$ //Store the proportions of samples of each category in a single
        cluster to the total samples of the current cluster
11.     foreach $l$ in Set(predictLabels) do
12.         singleClusterNums $\leftarrow$ Where$(g_k == l)$.size()
13.         singleClusterProportion.append (singleClusterNums$|g_k|$)
14.     end foreach
15.     totalLabelsProportion.append (singleClusterProportion)
16. end foreach
17. Iters $\leftarrow 50$, NIND $\leftarrow 50$, $M \leftarrow m$ //Set the number of evolutionary iterations, population size,
    and number of optimization objectives
18. Population$\leftarrow$Random.round(NIND, $st$) //Initialize the population matrix randomly
19. NDSet$\leftarrow$NSGA-II (Population, Iters, $M$, @Fitness) //Execute multi-objective optimization and
    storing the Pareto solution set into NDSet
20. $X$.add(Convert(NDSet, $n$)) //Users determine the selection solution according to their needs and
    add selected test input samples into $X$
21. return $X$
```

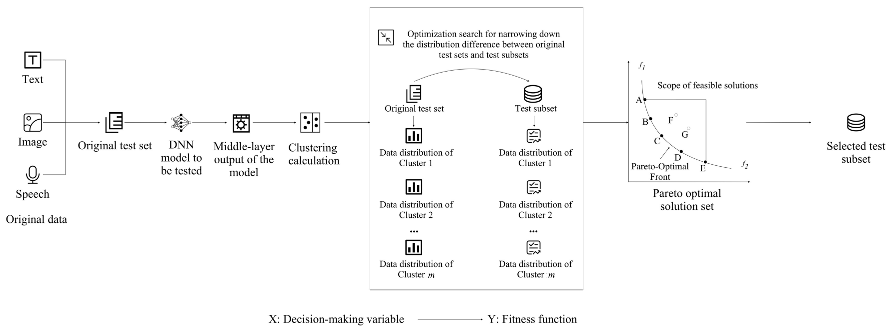
Text
Image
Speech
Original data
Original test set
DNN model to be tested
Middle-layer output of the model
Clustering calculation
Optimization search for narrowing down the distribution difference between original test sets and test subsets
Original test set
Test subset
Data distribution of Cluster 1
Data distribution of Cluster 1
Data distribution of Cluster 2
Data distribution of Cluster 2
...
Data distribution of Cluster m
Data distribution of Cluster m
f1
Scope of feasible solutions
A
B
F
G
C
D
E
Pareto-Optimal Front
f2
Pareto optimal solution set
Selected test subset
X: Decision-making variable
Y: Fitness function

Figure 1 Flow of DMOS

2.3.3 *HDBSCAN: hierarchical clustering of data for sampling*

DMOS employs HDBSCAN[18] to cluster the original test set to evaluate its data distributions preliminarily, thus preparing for later sampling. The reasons for using HDBSCAN can be summarized as follows.

(1) As it is difficult to predict the real categories of test input in a test set, clustering algorithms requiring the known number of clusters in advance are not applicable, such as the K-means algorithm[38]. HDBSCAN does not require a preset number of clusters, and its

clustering is not density-based.

(2) The effectiveness of HDBSCAN has been verified[18], and thus better clustering results can be obtained by this method.

(3) Only few parameters need to be set in HDBSCAN[18].

Given the previously preprocessed feature vectors of test input, DMOS employs HDBSCAN for soft clustering, dividing T into m clusters of different sizes (Line 7). Then, it calculates the proportions of data of various categories in these clusters in the original test set to prepare for later calculation of optimization objectives (Lines 9–16). Specific operations are as follows: traverse the cluster set G, and every time a new cluster g_k is traversed, calculate the proportions of data of different categories in g_k (Lines 10–14); store results in *totalLabelsProportion* (Line 15).

2.3.4 *Multi-objective optimization: operation parameter initialization and solution processing*

After the parameters of the original test set are calculated, basic parameters of multi-objective optimization are set (Line 17), including the number of evolutionary iterations *Iters*, the population size *NIND*, and the number M of optimization objectives (as m clusters are formed by HDBSCAN in this method, m optimization objectives are produced). Then, *Population* (Line 18) is initialized randomly. In DL test input selection, any individual in the *Population* is an st-dimensional vector, representing a specific selection solution for the original test set. Any element in this vector is a real number between 1 and 100, representing the weight of selecting the test input in this position. Upon the determination of the final selection solution, the individual vector elements are sorted in descending order, and the test input samples in the first n positions with the largest weight are taken to construct the new test subset. The classical algorithm NSGA-II[21] in multi-objective optimization is used to solve the above problems. Similarly to other multi-objective genetic evolutionary algorithms, NSGA-II first randomly generates an initial population of a certain size. Then, it obtains the first generation of the child population through the three basic operators of selection, crossover, and mutation after the non-dominated sorting of individuals in the population. From the second generation, the parent and child populations are merged into a new generation for fast non-dominated sorting; meanwhile, the crowding degree of individuals in each non-dominated layer is calculated, and appropriate individuals are selected according to the non-dominance relationship and individual crowding degree to form a new parent population. After that, a new child population is generated by the basic operators. When the specified number of iterations or other specified conditions are reached, the iteration will be terminated, and a PS is returned. The reasons for using NSGA-II in DMOS are summarized as follows: (1) the elitist strategy is introduced to process individuals, where parent and child populations compete together to produce new individuals. This expands the sampling space and ensures the quality of individuals; (2) fast non-dominated sorting is proposed to reduce the time complexity of algorithm running; (3) the crowding degree and its comparison operator are introduced, which makes up for the defect of requiring manual designation of shared parameters and enables uniform convergence of population individuals, effectively ensuring population diversity. Once NSGA-II terminates, the obtained PS is stored in *NDSet* (Line 19 in Algorithm 1). In the end, users can determine the final selection solution according to their test needs, and the selected test subset corresponding to the Pareto optimal solution is stored in X (Line 20) and then returned (Line 21).

2.3.5 *Fitness function design: establishing the function relationship between optimization objectives and populations*

Fitness function design is a key step in solving multi-objective optimization, namely, the calculation of optimization-objective values corresponding to individuals in a population. In DL test input selection, the fitness function of DMOS is shown in Algorithm 2, whose input

Algorithm 2. Fitness (calculation of fitness function).

```
Input: the total population to be optimized, Population;
       the proportion of each category in each cluster to the current cluster after clustering of the
       original test set $T$, totalLabelsProportion;
       user-specified number of samples to be selected from the original test set, $n$;
       category of each sample in the original test set $T$ predicted by the model to be tested,
       predictLabels.
Output: optimization-objective values Objvalues corresponding to all individuals in a population,
        with a size of $NIND \times M$.
1.  $clusterIndex \leftarrow 0$
2.  $labelNums \leftarrow predictLabels.size()$
3.  $Objvalues \leftarrow \varnothing$
4.  foreach Individual in Population do
5.      $T' \leftarrow argsort(Individual)[:n]$ //By sorting, select the samples in the first $n$ positions with the
          largest weight to form a new test subset $T'$
6.      $totalProportionErrors \leftarrow \varnothing$ //Store the optimization-objective value of each individual
7.      $G' \leftarrow \{g_1, g_2, \cdots, g_t\} \leftarrow GetNew\{T', G\}$ //Obtain clusters corresponding to the selected
          subset $T'$, where $t \leq m$
8.      foreach $i$ in $range(0, |G|)$ do
9.          if $G[i]$ not in $G'$ then //If no sample in Cluster $G[i]$ is selected in $G'$
10.             $totalProportionErrors.append(sum(totalLabelsProportion[clusterIndex]))$
11.             $clusterIndex += 1$
12.         else
13.             $s \leftarrow \varnothing$
14.             foreach $l$ in $Set(predictLabels)$ do //Calculate the proportion of samples of each
                  category in $G'[i]$ to total samples of $G'[i]$
15.                 $singleClusterNums \leftarrow where(G'[i] == l).size()$
16.                 $s.append(singleClusterNums/|G'[i]|)$
17.             end foreach
18.             //Calculate the errors in proportions of each category in the same cluster of the original
                  set and the subset
19.             $e \leftarrow [abs(totalLabelsProportion[clusterIndex][j] - s[j])$ for $j$ in $range(0, labeNums)]$
20.             $clusterIndex += 1$
21.             $totalProportionErrors.append(sum(e))$ //Calculate the sum of errors in proportions of
                  all categories in the current cluster of the original set and the subset
22.         end if
23.     end foreach
24.     $Objvalues.append(totalProportionErrors)$
25. end foreach
26. return Objvalues
```

parameters include the total population *Population* to be optimized, the original test set T, the proportion *totalLabelsProportion* of data of various categories in each cluster after clustering, the number n of selections specified by users, and the category label set *predictLabels* predicted by the DNN model D to be tested. It finally returns the set *Objvalues* of optimization-objective values of all individuals in the population. The specific solving process is as follows. The population is traversed (Line 4), and its elements are sorted according to their values. The top n test input samples with the largest weight are selected to form a new test subset T' (Line 5 in Algorithm 2), where the clustering information of test input in T' is denoted as $G' = \{g'_1, g'_2, \cdots, g'_t\}$, with $t \leq m$ (Line 7). Then, each cluster in G is traversed (Line 8 in Algorithm 2). If G' contains no samples of $G[i]$, the optimization-objective value of Cluster $G[i]$ is directly denoted as the sum of the proportions of the various categories in $G[i]$ in the original test set (Lines 9–11); otherwise, the proportions of samples of various categories in Cluster $G'[i]$ are computed (Lines 14–17). After that, the differences between Clusters $G'[i]$

and $G[i]$ in the proportions of samples of various categories are calculated, the absolute values of which are recorded in e (Lines 18–19). Finally, the sum of all elements in e is recorded as the optimization-objective value corresponding to Cluster $G[i]$ (Line 20). Each optimization-objective value of an individual after each solving is stored in *Objvalues* (Line 23), which will be returned after all individuals have been considered (Line 25).

3 Test Design and Result Analysis

3.1 Introduction to test objects

The test objects studied in this paper includ DL test sets and DNN models to be tested. We use eight test objects, the combinations of the DL test sets and the DNN models of classification tasks, as the smallest units of testing and evaluation analysis. Table 1 lists the details of the models and test sets used. In this table, the last four columns show the size of the DNN models to be tested, the size of the DL test sets (including the number of test input samples), the overall accuracy of models over the test sets, and the number of different test-input categories contained in the test sets. As the actual application of DL is complex, we attempted to construct a comprehensive test benchmark during our research.

Table 1 DNN models and relevant test sets

ID	Test set	Model	Model size (KB)	Number of test input samples	Overall accuracy (%)	Number of categories
1	MNIST	LeNet1	113	10,000	94.86	10
2		LeNet4	947	10,000	96.79	10
3		LeNet5	1,093	10,000	98.68	10
4	CIFAR10	VGG16	21 814	10,000	78.71	10
5		ResNet20	3,507	10,000	91.45	10
6	CIFAR100	ResNet20	10,615	10,000	71.42	100
7	ImageNet	VGG19	562,176	50,000	64.73	1,000
8	Speech-Commands	DeepSpeech	6,734	6,471	94.53	30

Note: The precision of the ImageNet-based model in our study is lower than that in Ref. [41] in that the preprocessing method used in our study is slightly different from that in Ref. [41]. More specifically, to obtain the input images with a fixed size of 224×224 from ImageNet, Simonyan *et al.*[41] randomly cropped the images from the rescaled ones, while we resized the images according to the method provided by the official examples of Keras without image cropping.

3.1.1 *DL test sets*

DNN models used in the testing were trained by five popular datasets (namely, MNIST, CIFAR10, CIFAR100, ImageNet, and Speech-Commands). These datasets have widely been used in existing studies[11, 27, 39] and have also been used to generate adversarial data[40] to test DNN models in some studies. Specifically, MNIST (http://yann.lecun.com/exdb/mnist/) is a dataset on the identification of handwritten digits (numbers 0–9, with a total of 10 categories). CIFAR10 (http://www.cs.toronto.edu/~kriz/CIFAR.html) is a dataset containing 10 categories of universal objects (airplanes, cars, birds, etc.) in the real world. CIFAR100 is similar to CIFAR10, but it contains more real objects than CIFAR10, with 100 categories of real objects in total. ImageNet (http://www.image-net.org) is an image dataset organized according to the WordNet hierarchy (containing 1,000 different categories of images). Speech-Commands[39] is a dataset of speech recognition, which contains a set of WAV-formatted audio files with a length of about 1 s collected by crowd-sourcing; each file contains only one spoken English word.

3.1.2 *DNN models to be tested*

As for the types of DNN models to be tested, this paper first considers two types of DNN models with different precision, namely, high-precision and low-precision models. Here, models with overall accuracy of more than 80% are called high-precision models; otherwise, they are called low-precision models. We use two low-precision models of the real world, namely, the CIFAR100-based ResNet20 model and the ImageNet-based VGG19 model. In addition, this paper mainly focuse on DNN models that are based on classification tasks. For example, DeepSpeech is a multi-label classification model (its predicted results are not definite category labels but a sequence of phonemes), while others are single-label classification models (their predicted results are determined category labels). Most of the DNN models used in the testing are CNN models, and only DeepSpeech (https://github.com/bjtommychen/Keras_DeepSpeech2_SpeechRecognition) is an RNN model.

3.2 Introduction to evaluation indicators and contrast methods

3.2.1 *Evaluation indicators*

• Accuracy estimation error (overall accuracy and accuracy of test input of each category)

Accuracy refers to the proportion of samples correctly classified by a model to be tested to total samples in the test set. In the previous methods for DL test input selection, researchers usually focus on whether the overall accuracy between test subsets and original test sets is similar to measure the similarity between their test capabilities. In this paper, the focus is also given to category information, namely, the accuracy of the test input of various categories. Thus, absolute values of the differences between original test sets and test subsets in overall accuracy and accuracy of the test input of various categories are taken as the criteria to measure the similarity between the test properties of the two sets.

• Coverage estimation error

In recent years, researchers have proposed many test coverage indicators to measure the test adequacy of DNN models[25, 26, 42]. In this study, we use five neuron-coverage indicators to measure that. The first indicator is DeepXplore's NC (DNC) proposed by Pei *et al.*[26], which can be calculated by the ratio of activated neurons (if the output value of a neuron is greater than a preset threshold after test input execution, the neuron is considered to be activated) to total neurons in a DNN model to be tested. The remaining four indicators are multi-granularity NC metrics proposed by Ma *et al.*[25], i.e., top-K NC (TKNC), K-multisection NC (KMNC), neuron boundary coverage (NBC), and strong neuron activation coverage (SNAC). Specifically, TKNC is a metric of layer-level NC, which can be calculated by the proportion of the first k neurons in each layer arranged in descending order of output values after execution of test input to the total neurons. KMNC first divides the output range of each neuron into k sections from the training data. If the output value of a neuron (the output range of the neuron n is denoted as $[a, b]$) falls into a specific section after execution of the next test input, this section is considered to be covered. Thus, this metric calculates the proportions of covered sections by all neurons. Unlike KMNC, NBC considers the coverage of the outer regions of neuron output (i.e., $(-\infty, a)$ and $(b, +\infty)$) after the execution of test input. SNAC only considers the coverage of the outer region (i.e., $(b, +\infty)$) of upper bounds after the execution of test input. In this paper, we take the absolute values of the differences between original test sets and selected test subsets in these five test coverage indicators as criteria to measure the similarity of their test properties.

• Inverted generational distance (IGD)

Accuracy and coverage mainly assess performance under the scenario of DL test input selection. For an effective assessment of Pareto solutions returned from DMOS, IGD[36] is used as an evaluation indicator. It calculates the average Euclidean distance between all solutions in the theoretical PF and those in the Pareto front obtained by the multi-objective optimization algorithm to be evaluated (namely that the difference between solutions from the method to be evaluated and theoretically optimal ones). A smaller IGD indicates that the PS obtained by the algorithm to be evaluated is closer to the theoretically optimal solution, with a more uniform distribution. It also indicates that this algorithm has better convergence and diversity.

- Statistical test method

To verify whether there is a statistically significant difference in test results of various selection methods, we use the Wilcoxon signed-rank test method[43] to analyze the test results. It is a nonparametric hypothesis test method for testing whether median deviations of paired data satisfy the null hypothesis. In this paper, given the confidence of 0.05, the null hypothesis is rejected when the calculated p is less than 0.05, which means a statistically significant difference between the two groups of test results, and the null hypothesis is accepted when it is greater than 0.05, which implies that the difference between the two groups of test results can be ignored. On this basis, we also use the "Win/Tie/Loss" test method to compare the performance of different selection methods. This test method has been widely used in many traditional studies[44–46], where "Win" means that our method is significantly better than PACE under the confidence of 95%; "Tie" means that there is no statistically significant difference between our method and PACE, and "Loss" refers to other cases. In addition, the Scott-Knott ESD test[47] is also employed, which is widely used to analyze the superiority of some methods over others and can rank these methods globally. It ensures that there is no statistically significant performance difference between methods of the same group but a statistically significant performance difference between methods of different groups. Results of the Scott-Knott ESD test are plotted as box plots (shown in Figs. 2–7). In these figures, method groups with a statistically significant difference are separated from others by a dotted line, and methods in groups on the same side of the dotted line have no statistically significant difference.

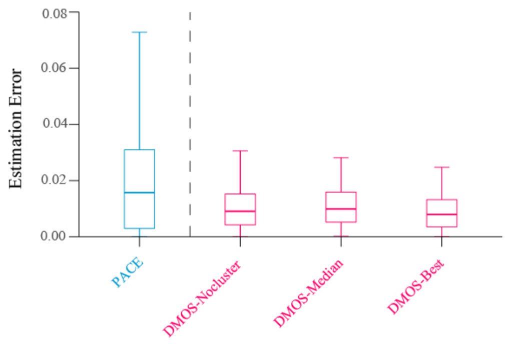

Figure 2 Scott-Knott ESD test results of overall accuracy estimation error

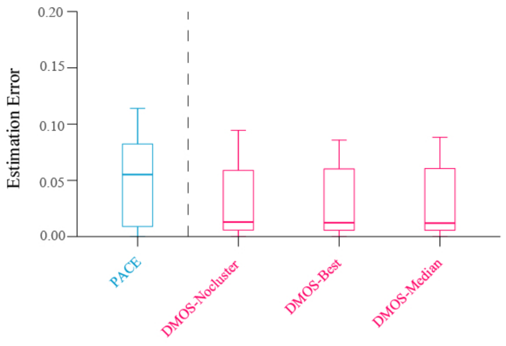

Figure 3 Scott-Knott ESD test results of NC estimation error

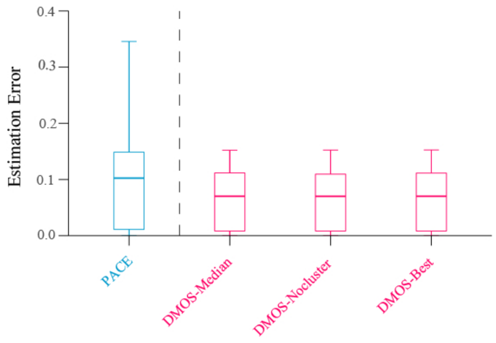

Figure 4 Scott-Knott ESD test results of NBC estimation error

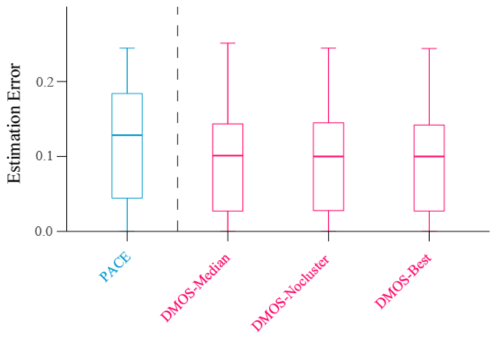

Figure 5 Scott-Knott ESD test results of SNAC estimation error

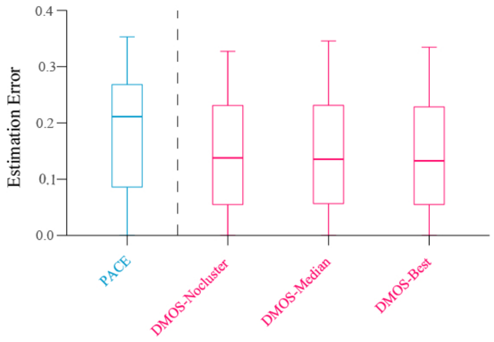

Figure 6 Scott-Knott ESD test results of TKNC estimation error

3.2.2 *Comparison methods*

PACE[17] is a newly proposed method for DL test input selection. As it has been comprehensively compared with the existing methods (CES, etc.) in previous studies, this study use it as a contrast method. The input of PACE includes a DNN model to be tested, a test set to be labeled, and the user-specified number of selections. For the whole test set, some test input samples have similar test capabilities, while others have different ones. Therefore, the test subset selected by PACE should cover various types of test capabilities of the original test set, so as to better represent the original test set. First, PACE converts each test input sample into a feature vector through feature extraction and pre-processes the vector through feature

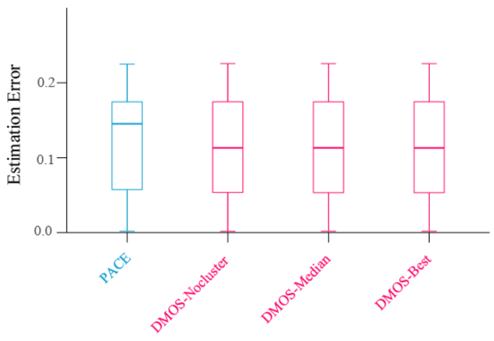

Figure 7 Scott-Knott ESD test results of KMNC estimation error

value normalization or feature dimension reduction. Then, these test input samples are divided into several normal-point and abnormal-point clusters by clustering. After that, PACE defines a threshold to determine the proportions of sampling from these clusters and then determines the number of samples. On this basis, prototype selection based on MMD-Critic[19] and adaptive random selection[20] is used for sampling in normal-point clusters and outlier space, respectively. Finally, a new test subset is constructed, which is the output of PACE. Developers can just label this set of data to complete model testing, thus greatly saving testing costs.

To comprehensively evaluate the performance of DMOS from multiple perspectives, we also design several variants of DMOS for research.

(1) DMOS-Nocluster. Without clustering, this method directly takes the difference between the original test and the selected test subset in the proportions of the test input of different categories as the optimization object for solutions. In the obtained PS, the solution with the minimum average error in the accuracy estimation of test input of various categories is taken as the final selection solution. Similarly, we also evaluated the performance of the final selection solution through indicators such as overall accuracy and test coverage to analyze the contribution of clustering to DMOS.

(2) DMOS-Median. In this method, the median of errors in the accuracy estimation of the test input of various categories in the PS obtained by DMOS is selected as the result.

(3) DMOS-Best. In this method, the solution with the minimum average error in accuracy estimation of the test input of various categories in the PS obtained by DMOS is selected as the result.

(4) RandomSearch. In this method, a random search strategy is adopted for the evaluation and selection functions to make decisions (which means its random search directions of optimization). Then, the final selection solution is obtained after a specified number of evolutions. This method is mainly designed to evaluate the performance of DMOS from the view of multi-objective optimization, and hence, IGD is mainly employed to evaluate the optimization effect and solution-set quality of this method and DMOS. On this basis, the contribution of multi-objective optimization to DMOS is analyzed.

(5) EA-Best. This method directly takes the accuracy estimation errors of test input in each real category of the original test set and the selected test subset as optimization objectives, which tries several groups of parameters and uses NSGA-II for solutions. EA-Best is designed to explore the limit cases of the accurate coverage of test input in various categories under multi-objective optimization, and thus, the solution with the minimum average error in accuracy estimation of the test input of various categories is selected every time as the result. Although solutions obtained by EA-Best are not theoretically optimal,

results reveal that there is little performance difference under different test parameters. Therefore, we can believe that solutions obtained by EA-Best are close to theoretically optimal ones and can be used as benchmarks for the comparison with DMOS.

3.3 Experimental setup

DMOS was implemented in Python 3.7.4, and features were extracted from DL test sets in test objects by Keras 2.3.1[48] and TensorFlow 1.15.0[49]. In addition, algorithms for feature dimension reduction, clustering, and multi-objective optimization were used during the implementation, which are provided by Python's mature frameworks Scikit-learn 0.23.1[50], HDBSCAN 0.8.26[51], and Geatpy 2.6.0[52], respectively. Parameters involved in DMOS mainly include those of clustering, dimension reduction, feature types, and multi-objective optimization. The first three groups of parameters jointly affect the clustering effect, while the last group of parameters directly affects the quality of the final Pareto solutions. It is expected that clustering can effectively distinguish different categories of test input samples and that each cluster should have as few different categories as possible. For different test objects, both clustering and dimension-reduction parameters were set according to recommendations in PACE. The optimal setting of feature-type parameters was –1, which means they are the last output-layer parameters of the model. Parameters of multi-objective optimization mainly include the number of evolutionary iterations and the total number of populations, in which the former determines the timing of convergence, and the latter determines the scope of search space, and both were set to 50. For the implementation and parameter setting of other selection methods, we followed the recommendations of the existing work[17].

To present obvious errors in accuracy estimation of the test input of various categories in different test objects under different numbers of selections, we made the following settings. For all test objects with 10 categories, the number of selections varied from 55 to 205, with an iteration step size of 10. For test objects with 100 categories, the number of selections varied from 350 to 2,050, with an iteration step size of 100. For test objects with 1,000 categories, the number of selections varied from 500 to 20,500, with an iteration step size of 1,000. For the Speech-Commndstest set, the number of selections varied from 22 to 1,590, with a step size of 64.

All the tests were carried out on an Intel Xeon E5-2640 server Ubuntu 18.04.2 as operating system and 128 GB of memory.

3.4 Research questions

Starting with the following research questions, we carried out corresponding analytical tests to verify the effectiveness of DMOS.

- RQ1: Can DMOS ensure that the selected test subsets are similar to the original test sets regarding the accuracy estimation of the test input of various categories?

A selected test subset should not only cover as many categories of its original test set as possible but also ensure the similarity with the original test set in terms of the accuracy of the test input of various covered categories, which is of great significance for the subset to maintain test properties of the original test set. The existing methods give priority to the difference between test subsets and original test sets in overall accuracy during selection. As a result, they are very likely to ignore the accuracy of the selected test subset on the test input of various categories. Thus, this research question is expected to analyze whether DMOS, compared with the existing methods, can effectively ensure that the selected test subsets can estimate the accuracy of the test input of various categories more precisely.

- RQ2: Can the test subsets selected by DMOS have test capabilities similar to those of the original test sets in terms of overall accuracy and test coverage?

Apart from the accuracy estimation of the test input of various categories, in previous studies, overall accuracy and some test coverage indicators[16] were often used to measure the difference between selected test subsets and original test sets in test properties. Therefore, a high-performance selection method should not only ensure that selected test subsets are close to original test sets in terms of the accuracy estimation of the test input of various categories but also produce small estimation errors of the overall accuracy and test-coverage indicators. Regarding this research question, we want to analyze whether DMOS can maintain the test capabilities of the selected test subsets closer to those of the original test sets more effectively than other methods from the perspective of other evaluation indicators.

- RQ3: How much do clustering and multi-objective optimization contribute to DMOS?

The two most critical steps of DMOS are clustering and multi-objective optimization. The former aims to estimate the distribution proportions of various data in the original test sets preliminarily when no real labels are available, thereby laying a good foundation for later optimization sampling. The latter endeavors to continuously make the data distribution of the selected test subsets closer to that of the original test sets on the basis of clustering. Therefore, this research question is expected to analyze the contribution degree of the two critical parts to the final performance of DMOS.

- RQ4: How much is the runtime overhead of DMOS?

The existing DL test input selection methods are developed to reduce the original unlabeled datasets and thus cut the costs of labeling and testing by researchers. Compared with the time overhead required for these tasks, execution costs of the existing selection methods are negligible. To fully study the performance of DMOS, however, we propose this question to test this method in terms of runtime overhead and compare it with the existing methods.

3.5 Result analysis

- RQ1: Can DMOS ensure that the selected test subsets are similar to the original test sets regarding the accuracy estimation of the test input of various categories?

(1) Design

To evaluate whether the selected test subsets could contain as many categories in the original test sets as possible and whether various categories of the test subsets and original test sets have similar accuracy for the DNN models to be tested, we mainly analyze the test results of DMOS-Best, DMOS-Median, EA-Best, and PACE from the following two aspects. Firstly, we observe the variations of average errors in the accuracy estimation of the test input of various categories in test objects under different selection methods as the number of selections increased. On this basis, we analyze whether these methods could make average errors converge to a smaller range as early as possible, as well as their performance stability in terms of standard deviations. Secondly, we use Wilcoxon signed-rank test and Win/Tie/Loss analysis to evaluate whether each selection method could significantly reduce the errors in accuracy estimation of any category for the same test object under different numbers of selections and then compare the performance of the various methods.

(2) Results

Tables 2–9 list the variations of average errors in accuracy estimation of the test input of various categories with the number of selections under the eight test objects. In each table, the first column refers to the number of selections, and the elements in parentheses for the second to sixth column are the mean and the standard deviation of errors in the accuracy estimation of the test input of various categories by the selection methods under different numbers of selections. The elements with a dark gray background indicate that the performance of the corresponding method is the best under the same number of selections, while those with a light gray background

Table 2 Variations of average errors in accuracy estimation of test input of various categories for test object CIFAR10-ResNet20

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
55	(0.058, 0.029)	(0.191, 0.272)	(0.056, 0.024)	(0.054, 0.022)	(0.096, 0.049)
65	(0.050, 0.022)	(0.131, 0.126)	(0.040, 0.025)	(0.048, 0.021)	(0.084, 0.036)
75	(0.051, 0.028)	(0.121, 0.116)	(0.049, 0.032)	(0.044, 0.020)	(0.083, 0.023)
85	(0.047, 0.017)	(0.110, 0.121)	(0.044, 0.026)	(0.040, 0.023)	(0.078, 0.021)
95	(0.048, 0.023)	(0.116, 0.117)	(0.046, 0.034)	(0.041, 0.020)	(0.076, 0.022)
105	(0.038, 0.027)	(0.108, 0.068)	(0.045, 0.036)	(0.028, 0.021)	(0.078, 0.033)
115	(0.039, 0.034)	(0.120, 0.115)	(0.034, 0.021)	(0.033, 0.023)	(0.063, 0.014)
125	(0.039, 0.031)	(0.107, 0.115)	(0.039, 0.022)	(0.026, 0.024)	(0.063, 0.017)
135	(0.032, 0.026)	(0.085, 0.066)	(0.027, 0.024)	(0.032, 0.027)	(0.060, 0.021)
145	(0.042, 0.031)	(0.076, 0.054)	(0.037, 0.032)	(0.029, 0.024)	(0.062, 0.014)
155	(0.043, 0.036)	(0.072, 0.057)	(0.037, 0.037)	(0.024, 0.021)	(0.061, 0.017)
165	(0.036, 0.025)	(0.065, 0.049)	(0.025, 0.019)	(0.021, 0.019)	(0.052, 0.019)
175	(0.030, 0.022)	(0.051, 0.044)	(0.028, 0.033)	(0.023, 0.022)	(0.053, 0.015)
185	(0.027, 0.030)	(0.047, 0.046)	(0.027, 0.020)	(0.021, 0.018)	(0.045, 0.019)
195	(0.037, 0.043)	(0.047, 0.041)	(0.031, 0.019)	(0.023, 0.017)	(0.055, 0.017)
205	(0.037, 0.027)	(0.044, 0.042)	(0.019, 0.022)	(0.017, 0.016)	(0.046, 0.010)

Table 3 Variations of average errors in accuracy estimation of test input of various categories for test object MNIST-LENET1

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
55	(0.039, 0.024)	(0.078, 0.058)	(0.037, 0.018)	(0.035, 0.018)	(0.076, 0.033)
65	(0.038, 0.029)	(0.071, 0.054)	(0.038, 0.023)	(0.037, 0.021)	(0.057, 0.033)
75	(0.041, 0.021)	(0.076, 0.082)	(0.037, 0.018)	(0.033, 0.021)	(0.061, 0.033)
85	(0.029, 0.020)	(0.075, 0.087)	(0.035, 0.017)	(0.036, 0.018)	(0.064, 0.033)
95	(0.036, 0.019)	(0.073, 0.088)	(0.038, 0.025)	(0.030, 0.023)	(0.053, 0.024)
105	(0.033, 0.023)	(0.070, 0.089)	(0.031, 0.017)	(0.029, 0.019)	(0.061, 0.033)
115	(0.035, 0.027)	(0.067, 0.090)	(0.030, 0.018)	(0.030, 0.013)	(0.050, 0.020)
125	(0.034, 0.022)	(0.074, 0.089)	(0.031, 0.024)	(0.022, 0.016)	(0.049, 0.021)
135	(0.031, 0.024)	(0.076, 0.088)	(0.027, 0.021)	(0.022, 0.015)	(0.046, 0.020)
145	(0.030, 0.024)	(0.082, 0.105)	(0.028, 0.026)	(0.022, 0.027)	(0.041, 0.020)
155	(0.028, 0.023)	(0.079, 0.106)	(0.025, 0.017)	(0.019, 0.013)	(0.045, 0.021)
165	(0.023, 0.026)	(0.069, 0.085)	(0.029, 0.025)	(0.016, 0.013)	(0.045, 0.022)
175	(0.027, 0.022)	(0.065, 0.076)	(0.024, 0.019)	(0.012, 0.008)	(0.039, 0.016)
185	(0.021, 0.017)	(0.064, 0.061)	(0.014, 0.010)	(0.014, 0.012)	(0.039, 0.023)
195	(0.020, 0.018)	(0.064, 0.060)	(0.019, 0.012)	(0.016, 0.016)	(0.043, 0.015)
205	(0.020, 0.023)	(0.063, 0.072)	(0.025, 0.024)	(0.013, 0.016)	(0.041, 0.019)

indicate that the performance of the corresponding method is the second best. In terms of the average errors in the accuracy estimation of the test input of each category, for test objects with 10 categories (see Tables 3, 4, and 6), as too few test-input categories are in the test sets, the average estimation errors of the selected test subsets have begun to converge to a small range when the samples of DMOS-Best and DMOS-Median account for 0.7% of the original test sets. Although PACE has also begun to converge, it has a significantly higher average estimation error than other methods. As shown in Tables 2–5, DMOS-Median converges to a small error range earlier than PACE regarding test objects No. 3 and No. 5, but its advantages are no longer obvious with the increase in the number of selections. For the test object with 100 categories (the test object No. 6), as shown in Table 7, the test subsets selected by DMOS-Best and DMOS-Median before convergence can always maintain lower average estimation errors than that by PACE, and DMOS-Best and DMOS-Median reach convergence ranges earlier (when the number of samples is about 8% of the original test set). Although the estimation error range of DMOS-Median after convergence is of little difference from that of PACE, DMOS-Median is still superior to PACE

Table 4 Variations of average errors in accuracy estimation of test input of various categories for test object MNIST-LENET4

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
55	(0.032, 0.028)	(0.052, 0.048)	(0.025, 0.012)	(0.026, 0.015)	(0.046, 0.028)
65	(0.023, 0.015)	(0.050, 0.045)	(0.022, 0.016)	(0.046, 0.055)	(0.041, 0.028)
75	(0.022, 0.016)	(0.042, 0.038)	(0.022, 0.016)	(0.023, 0.016)	(0.049, 0.028)
85	(0.023, 0.014)	(0.048, 0.053)	(0.019 0.011)	(0.022, 0.014)	(0.040, 0.028)
95	(0.022, 0.016)	(0.043, 0.043)	(0.022, 0.016)	(0.024, 0.014)	(0.042, 0.023)
105	(0.025, 0.014)	(0.041, 0.042)	(0.024, 0.013)	(0.020, 0.009)	(0.037, 0.027)
115	(0.027, 0.015)	(0.038, 0.038)	(0.022, 0.016)	(0.018, 0.011)	(0.037, 0.015)
125	(0.023, 0.018)	(0.044, 0.034)	(0.018, 0.008)	(0.020, 0.007)	(0.034, 0.016)
135	(0.019, 0.008)	(0.041, 0.031)	(0.022, 0.010)	(0.017, 0.007)	(0.035, 0.017)
145	(0.024, 0.013)	(0.045, 0.035)	(0.019, 0.010)	(0.018, 0.011)	(0.035, 0.015)
155	(0.017, 0.009)	(0.043, 0.032)	(0.020, 0.009)	(0.018, 0.013)	(0.034, 0.017)
165	(0.020, 0.012)	(0.045, 0.040)	(0.018, 0.009)	(0.018, 0.008)	(0.027, 0.015)
175	(0.021, 0.011)	(0.048, 0.051)	(0.020, 0.011)	(0.015, 0.009)	(0.032, 0.017)
185	(0.021, 0.011)	(0.045, 0.047)	(0.016, 0.009)	(0.013, 0.009)	(0.027, 0.016)
195	(0.018, 0.013)	(0.045, 0.046)	(0.014, 0.007)	(0.014, 0.008)	(0.031, 0.016)
205	(0.019, 0.012)	(0.050, 0.046)	(0.017, 0.012)	(0.016, 0.008)	(0.030, 0.015)

Table 5 Variations of average errors in accuracy estimation of test input of various categories for test object MNIST-LENET5

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
55	(0.013, 0.007)	(0.025, 0.038)	(0.013, 0.007)	(0.013, 0.007)	(0.020, 0.007)
65	(0.013, 0.007)	(0.023, 0.032)	(0.013, 0.007)	(0.013, 0.007)	(0.013, 0.007)
75	(0.013, 0.007)	(0.023, 0.032)	(0.013, 0.007)	(0.013, 0.007)	(0.016, 0.007)
85	(0.013, 0.007)	(0.022, 0.028)	(0.013, 0.007)	(0.013, 0.007)	(0.020, 0.007)
95	(0.013, 0.007)	(0.021, 0.025)	(0.013, 0.007)	(0.013, 0.007)	(0.025, 0.007)
105	(0.013, 0.007)	(0.021, 0.025)	(0.013, 0.007)	(0.013, 0.007)	(0.018, 0.007)
115	(0.013, 0.006)	(0.019, 0.020)	(0.013, 0.007)	(0.013, 0.007)	(0.020, 0.007)
125	(0.013, 0.007)	(0.019, 0.018)	(0.013, 0.007)	(0.013, 0.007)	(0.019, 0.007)
135	(0.013, 0.007)	(0.018, 0.015)	(0.013, 0.007)	(0.020, 0.016)	(0.015, 0.007)
145	(0.013, 0.007)	(0.017, 0.014)	(0.013, 0.007)	(0.013, 0.007)	(0.017, 0.007)
155	(0.012, 0.005)	(0.019, 0.016)	(0.013, 0.007)	(0.014, 0.008)	(0.018, 0.007)
165	(0.013, 0.007)	(0.018, 0.014)	(0.013, 0.006)	(0.013, 0.006)	(0.019, 0.007)
175	(0.012, 0.005)	(0.018, 0.013)	(0.013, 0.006)	(0.012, 0.005)	(0.022, 0.008)
185	(0.016, 0.011)	(0.017, 0.012)	(0.012, 0.005)	(0.012, 0.005)	(0.019, 0.007)
195	(0.013, 0.006)	(0.016, 0.011)	(0.012, 0.005)	(0.012, 0.005)	(0.015, 0.007)
205	(0.012, 0.005)	(0.016, 0.010)	(0.011, 0.004)	(0.012, 0.005)	(0.019, 0.007)

after the sampling quantity reaches 16%, and its average estimation error range (6.9%–12.4%) after convergence is still superior to that (7.5%–13%) of PACE. DMOS-Best can always maintain its advantages over PACE, with its convergence range (5.9%–10%) of estimation errors being much lower than that (7.5%–13%) of PACE. For the test object No. 8 (see Table 8), although DMOS-Best outperforms PACE, DMOS-Median always falls behind PACE with the increase in the number of selections in terms of the performance of selected test subsets. The reason is as follows: clustering features of DMOS in the testing are the output of the last layer of the DNN model to be tested, while the output of the last layer of the model used by the test object No. 8 is the prediction probabilities of individual letters. These probabilities should be further transformed to produce final prediction categories, namely, words composed of letters with maximum probabilities. In other words, unlike other test objects, the test object No. 8 lacks a direct mapping relationship between its final test-input categories and clustering features, which makes the clustering method fail to effectively estimate the distribution of the test input. As a result, each test subset obtained by DMOS for the test object No. 8 performs unstably. In

Table 6 Variations of average errors in accuracy estimation of test input of various categories for test object CIFAR10-VGG16

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
55	(0.075, 0.081)	(0.159, 0.105)	(0.079, 0.055)	(0.075, 0.076)	(0.151, 0.046)
65	(0.070, 0.041)	(0.160, 0.123)	(0.060, 0.042)	(0.052, 0.051)	(0.131, 0.040)
75	(0.079, 0.056)	(0.145, 0.112)	(0.061, 0.062)	(0.046, 0.034)	(0.121, 0.025)
85	(0.066, 0.064)	(0.155, 0.101)	(0.066, 0.057)	(0.051, 0.035)	(0.123, 0.037)
95	(0.061, 0.031)	(0.140, 0.085)	(0.066, 0.041)	(0.051, 0.048)	(0.109, 0.027)
105	(0.056, 0.069)	(0.147, 0.090)	(0.060, 0.055)	(0.042, 0.026)	(0.097, 0.027)
115	(0.052, 0.039)	(0.152, 0.109)	(0.053, 0.071)	(0.048, 0.039)	(0.100, 0.026)
125	(0.059, 0.045)	(0.130, 0.100)	(0.042, 0.020)	(0.041, 0.041)	(0.086, 0.022)
135	(0.067, 0.057)	(0.118, 0.092)	(0.056, 0.047)	(0.043, 0.039)	(0.096, 0.036)
145	(0.053, 0.044)	(0.103, 0.090)	(0.031, 0.012)	(0.035, 0.025)	(0.086, 0.018)
155	(0.047, 0.045)	(0.108, 0.097)	(0.045, 0.029)	(0.037, 0.026)	(0.085, 0.029)
165	(0.047, 0.028)	(0.091, 0.093)	(0.048, 0.041)	(0.030, 0.032)	(0.080, 0.018)
175	(0.057, 0.061)	(0.093, 0.083)	(0.042, 0.044)	(0.032, 0.028)	(0.081, 0.021)
185	(0.039, 0.035)	(0.095, 0.069)	(0.028, 0.024)	(0.041, 0.029)	(0.070, 0.018)
195	(0.053, 0.055)	(0.085, 0.065)	(0.032, 0.021)	(0.027, 0.021)	(0.071, 0.020)
205	(0.043, 0.033)	(0.080, 0.050)	(0.029, 0.018)	(0.028, 0.023)	(0.072, 0.015)

Table 7 Variations of average errors in accuracy estimation of test input of various categories for test object CIFAR100-ResNet20

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
350	(0.208, 0.163)	(0.256, 0.202)	(0.190, 0.162)	(0.184, 0.145)	(0.220, 0.063)
450	(0.165, 0.137)	(0.214, 0.158)	(0.154, 0.131)	(0.157, 0.135)	(0.185, 0.051)
550	(0.143, 0.122)	(0.182, 0.147)	(0.131, 0.110)	(0.130, 0.097)	(0.164, 0.041)
650	(0.127, 0.099)	(0.160, 0.100)	(0.117, 0.090)	(0.116, 0.096)	(0.146, 0.037)
750	(0.121, 0.088)	(0.140, 0.098)	(0.106, 0.074)	(0.104, 0.083)	(0.129, 0.034)
850	(0.114, 0.089)	(0.130, 0.095)	(0.099, 0.094)	(0.092, 0.076)	(0.124, 0.030)
950	(0.095, 0.073)	(0.111, 0.090)	(0.097, 0.079)	(0.088, 0.073)	(0.111, 0.027)
1,050	(0.097, 0.084)	(0.105, 0.082)	(0.087, 0.076)	(0.086, 0.080)	(0.109, 0.026)
1,150	(0.086, 0.070)	(0.097, 0.081)	(0.079, 0.067)	(0.081, 0.063)	(0.101, 0.023)
1,250	(0.090, 0.068)	(0.091, 0.074)	(0.083, 0.064)	(0.074, 0.064)	(0.094, 0.022)
1,350	(0.084, 0.070)	(0.084, 0.076)	(0.079, 0.065)	(0.071, 0.062)	(0.091, 0.019)
1,450	(0.075, 0.063)	(0.084, 0.068)	(0.075, 0.059)	(0.070, 0.064)	(0.087, 0.021)
1,550	(0.073, 0.057)	(0.081, 0.065)	(0.072, 0.058)	(0.067, 0.062)	(0.083, 0.021)
1,650	(0.077, 0.062)	(0.085, 0.063)	(0.069, 0.066)	(0.061, 0.053)	(0.079, 0.016)
1,750	(0.070, 0.059)	(0.083, 0.061)	(0.062, 0.052)	(0.058, 0.048)	(0.079, 0.022)
1,850	(0.064, 0.053)	(0.077, 0.060)	(0.065, 0.045)	(0.062, 0.049)	(0.076, 0.018)
1,950	(0.061, 0.052)	(0.076, 0.060)	(0.059, 0.048)	(0.058, 0.052)	(0.071, 0.014)
2,050	(0.058, 0.046)	(0.075, 0.059)	(0.059, 0.047)	(0.058, 0.047)	(0.069, 0.019)

future research, we will pay more attention to feature analysis of speech datasets and explore the relationship between categories and features in such datasets to improve the performance of DMOS. Noteworthily, in Tables 2–9, the average estimation errors of DMOS-Best change almost the same as those of EA-Best, which indicates that there is a solution highly close to the theoretically optimal one in the PS obtained by DMOS. Moreover, DMOS-Median outperforms the latest selection method PACE on most of the test objects (except that it has no significant advantage over PACE with regard to test objects No. 3 and No. 5 and is inferior to PACE on the test object No. 8), which fully proves that DMOS can effectively ensure that its selected test subsets can estimate the accuracy of the test input of various categories precisely. For each selection method, without real label information, increasing the number of categories will raise the difficulty in accurate estimation of each category. As a result, all selection methods produce higher errors in the accuracy estimation of the test input of various categories in the case of 100

Table 8 Variations of average errors in accuracy estimation of test input of various categories for test object Speech-Commands-DeepSpeech

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
22	(0.420, 0.448)	(0.472, 0.454)	(0.324, 0.418)	(0.361, 0.433)	(0.535, 0.403)
86	(0.097, 0.175)	(0.077, 0.084)	(0.053, 0.026)	(0.058, 0.041)	(0.150, 0.035)
150	(0.058, 0.043)	(0.061, 0.048)	(0.056, 0.028)	(0.050, 0.025)	(0.086, 0.031)
214	(0.060, 0.032)	(0.050, 0.031)	(0.050, 0.031)	(0.045, 0.030)	(0.065, 0.026)
278	(0.046, 0.029)	(0.047, 0.030)	(0.044, 0.026)	(0.041, 0.025)	(0.061, 0.027)
342	(0.040, 0.028)	(0.045, 0.030)	(0.037, 0.027)	(0.036, 0.030)	(0.057, 0.025)
406	(0.037, 0.029)	(0.039, 0.025)	(0.035, 0.032)	(0.027, 0.020)	(0.052, 0.022)
470	(0.031, 0.028)	(0.039, 0.024)	(0.034, 0.026)	(0.030, 0.026)	(0.044, 0.021)
534	(0.029, 0.023)	(0.034, 0.021)	(0.025, 0.021)	(0.027, 0.020)	(0.040, 0.013)
598	(0.029, 0.022)	(0.035, 0.028)	(0.028, 0.022)	(0.024, 0.018)	(0.037, 0.015)
662	(0.024, 0.020)	(0.032, 0.024)	(0.023, 0.014)	(0.023, 0.016)	(0.035, 0.011)
726	(0.027, 0.020)	(0.031, 0.025)	(0.024, 0.020)	(0.021, 0.013)	(0.034, 0.010)
790	(0.023, 0.014)	(0.032, 0.025)	(0.022, 0.018)	(0.020, 0.014)	(0.031, 0.012)
854	(0.027, 0.022)	(0.030, 0.026)	(0.023, 0.018)	(0.020, 0.016)	(0.031, 0.012)
918	(0.022, 0.019)	(0.029, 0.025)	(0.019, 0.021)	(0.019, 0.015)	(0.028, 0.008)
982	(0.022, 0.016)	(0.028, 0.026)	(0.022, 0.016)	(0.019, 0.015)	(0.027, 0.010)
1,046	(0.022, 0.019)	(0.027, 0.024)	(0.022, 0.016)	(0.018, 0.012)	(0.026, 0.008)
1,110	(0.018, 0.013)	(0.026, 0.023)	(0.020, 0.017)	(0.016, 0.013)	(0.026, 0.009)
1,174	(0.017, 0.013)	(0.027, 0.027)	(0.019, 0.018)	(0.018, 0.012)	(0.027, 0.009)
1,238	(0.019, 0.017)	(0.027, 0.027)	(0.018, 0.012)	(0.014, 0.014)	(0.026, 0.008)
1,302	(0.016, 0.013)	(0.023, 0.023)	(0.019, 0.014)	(0.016, 0.012)	(0.024, 0.008)
1,366	(0.017, 0.010)	(0.023, 0.022)	(0.018, 0.015)	(0.017, 0.014)	(0.022, 0.006)
1,430	(0.017, 0.012)	(0.022, 0.021)	(0.018, 0.015)	(0.011, 0.011)	(0.022, 0.007)
1,494	(0.016, 0.015)	(0.021, 0.019)	(0.016, 0.010)	(0.013, 0.009)	(0.020, 0.006)
1,558	(0.016, 0.013)	(0.020, 0.019)	(0.018, 0.013)	(0.014, 0.011)	(0.021, 0.005)

Table 9 Variations of average errors in accuracy estimation of test input of various categories for test ImageNet-VGG19

Number of selections	DMOS-Nocluster	PACE	DMOS-Best	EA-Best	DMOS-Median
500	(0.525, 0.241)	(0.526, 0.241)	(0.517, 0.241)	(0.515, 0.242)	(0.531, 0.216)
1,050	(0.360, 0.237)	(0.376, 0.248)	(0.356, 0.245)	(0.361, 0.235)	(0.372, 0.177)
2,050	(0.272, 0.204)	(0.283, 0.215)	(0.266, 0.202)	(0.267, 0.207)	(0.278, 0.124)
3,050	(0.212, 0.171)	(0.228, 0.183)	(0.209, 0.173)	(0.208, 0.167)	(0.220, 0.077)
4,050	(0.177, 0.148)	(0.183, 0.156)	(0.173, 0.141)	(0.170, 0.136)	(0.182, 0.063)
5,050	(0.150, 0.122)	(0.159, 0.133)	(0.151, 0.124)	(0.149, 0.127)	(0.157, 0.057)
6,050	(0.137, 0.111)	(0.138, 0.114)	(0.133, 0.111)	(0.135, 0.113)	(0.143, 0.044)
7,050	(0.119, 0.096)	(0.126, 0.109)	(0.117, 0.095)	(0.121, 0.099)	(0.127, 0.037)
8,050	(0.112, 0.092)	(0.118, 0.098)	(0.108, 0.088)	(0.111, 0.091)	(0.116, 0.034)
9,050	(0.103, 0.085)	(0.111, 0.092)	(0.101, 0.082)	(0.101, 0.083)	(0.108, 0.030)
10,500	(0.098, 0.079)	(0.104, 0.086)	(0.095, 0.078)	(0.095, 0.080)	(0.100, 0.030)
11,500	(0.092, 0.075)	(0.097, 0.078)	(0.090, 0.074)	(0.090, 0.073)	(0.094, 0.030)
12,500	(0.083, 0.069)	(0.091, 0.075)	(0.084, 0.068)	(0.086, 0.069)	(0.089, 0.025)
13,500	(0.081, 0.066)	(0.087, 0.070)	(0.081, 0.066)	(0.080, 0.064)	(0.085, 0.022)
14,500	(0.077, 0.063)	(0.083, 0.067)	(0.076, 0.060)	(0.075, 0.061)	(0.080, 0.021)
15,500	(0.074, 0.060)	(0.079, 0.064)	(0.072, 0.058)	(0.070, 0.057)	(0.076, 0.021)
16,500	(0.067, 0.055)	(0.075, 0.062)	(0.070, 0.057)	(0.070, 0.056)	(0.073, 0.020)
17,500	(0.066, 0.053)	(0.072, 0.060)	(0.066, 0.053)	(0.066, 0.054)	(0.069, 0.020)
18,500	(0.064, 0.052)	(0.069, 0.058)	(0.062, 0.049)	(0.062, 0.052)	(0.066, 0.019)
19,500	(0.061, 0.050)	(0.068, 0.057)	(0.060, 0.048)	(0.059, 0.048)	(0.063, 0.018)
20,500	(0.059, 0.048)	(0.065, 0.056)	(0.056, 0.044)	(0.060, 0.048)	(0.061, 0.018)

categories (the test object No. 6) and 1,000 categories (the test object No. 7) than in the case of 10 categories (namely that the variation ranges of the average estimation errors of various methods in Tables 7–9 are higher than those in Tables 2–6), although they have much more selections in the former case than in the latter one. In addition, each selection method covers very few categories and quantities when no real label is available. With such an insufficient selection, all methods have high errors in accuracy estimation of the test input of various categories (in Tables 7–9, EA-Best that can produce solutions closest to theoretically optimal ones still performs poorly under a small number of selections), and their performance is barely different. Nevertheless, Table 9 shows that DMOS-Best and DMOS-Median still outperform PACE by a narrow margin, and under the same number of selections, the test subsets selected by the two can still have smaller average errors in the accuracy estimation of various test-input categories than those selected by PACE. Lastly, in terms of the standard deviations of errors in accuracy estimation of the test input of various categories, with the rise in the number of samples, the estimation errors of all selection methods tend to be gentle, and the standard deviations are gradually decreased. Moreover, the ranges of the standard deviations and their variations of both DMOS-Best and DMOS-Median are smaller than those of PACE. As indicated above, test subsets selected by DMOS can precisely estimate the accuracy of the test input of various categories more evenly and comprehensively than those selected by PACE, with more stable performance.

Table 10 presents the results of the Wilcoxon signed-rank test combined with Win/Tie/Loss analysis for each selection method. Specifically, Table 10 demonstrates the comparison results of DMOS-Nocluster, DMOS-Best, EA-Best, and DMOS-Median with PACE in terms of the accuracy estimation of the test input of various categories for the eight test objects (for each test object, it compares the number of categories in which the four methods are significantly better than PACE in accuracy estimation). Similar to Tables 2–9, a dark-gray shaded result in Table 10 indicates the method with the best performance on the current test object, while a light-gray shaded one indicates the method with the second-best performance. According to Table 10, DMOS-Best outperforms PACE in the accuracy estimation of about 50% of the categories on all the 10-category test objects, and DMOS-Median can accurately estimate more categories than PACE. In addition, for the 10-category test objects except No. 1 and No. 5, DMOS-Best and DMOS-Median are not significantly inferior to PACE in terms of accuracy estimation. For the test object No. 8 using speech data, however, DMOS-Median performs worse than PACE: it outperforms PACE only in estimating seven categories, and its estimation performance is poorer than that of PACE in the other 10 categories. By contrast, DMOS-Best keeps its advantages over PACE, capable of accurately estimating more categories. This also indicates that we should attach more importance to speech datasets in future work to improve the overall quality of the final Pareto solution set. For test objects with 100 and 1,000 categories, as mentioned in the previous section, the increase in categories reduces the performance difference between various selection methods on the test objects No. 6 and No. 7. As DMOS-Median uses medians of all solutions in the Pareto solution set, it performs more stably than DMOS-Best under different

Table 10 Win/Tie/Loss analysis of four multi-objective optimization methods versus PACE as to various categories of eight test objects

ID	DMOS-Nocluster VS PACE	DMOS-Best VS PACE	EA-Best VS PACE	DMOS-Median VS PACE
1	6/3/1	8/1/1	8/1/1	4/5/1
2	3/7/0	5/5/0	6/4/0	4/6/0
3	4/6/0	4/6/0	4/6/0	3/7/0
4	7/3/0	7/3/0	7/3/0	3/7/0
5	5/4/1	4/6/0	6/4/0	2/6/2
6	23/65/12	23/64/13	23/67/10	31/42/27
7	195/655/150	208/642/150	199/653/148	277/493/230
8	7/16/7	11/12/7	11/14/5	7/13/10

numbers of selections. Therefore, DMOS-Median can outstandingly outperform PACE in more categories than DMOS-Best. Lastly, the performance of DMOS-Best is of little difference from that of EA-Best, which also illustrates the effectiveness of DMOS.

Table 11 lists the average errors of various methods under different evaluation indicators. The average errors in the accuracy estimation of test input of various categories listed in the first row show that such errors of DMOS-Nocluster, DMOS-Best, DMOE-Median, and PACE in a total of 144 tests of the eight test objects are 5.954%, 5.547%, 7.589%, and 8.473%, respectively. Compared with PACE, the three methods have respective average errors reduced by 2.519%, 2.926%, and 0.884%, with a respective improvement (calculated by (PACE-DMOS-X)/PACE) of 29.73%, 34.53%, and 10.43%. This also shows the performance superiority of DMOS to the classical method PACE in the current field.

Table 11 Average errors of various methods under seven evaluation indicators (%)

Indicator	DMOS-Nocluster	DMOS-Best	DMOS-Median	PACE
AvgAcc	5.954	**5.547**	7.589	8.473
TotalAcc	1.238	**1.081**	1.211	1.926
KMNC	10.520	**10.516**	10.517	11.721
NBC	**6.412**	**6.412**	6.435	11.527
NC	3.157	3.149	**3.143**	11.471
SNAC	9.258	**9.223**	9.285	18.938
TKNC	14.526	**14.394**	14.438	26.676

Summary. (1) The optimal solutions in the PS obtained by DMOS are close to the theoretically optimal ones and outperform the ones obtained by PACE on both speech and image test sets. (2) In the PS obtained by DMOS, solutions with ordinary performance still outperform the ones obtained by PACE on image test sets but perform poorly on speech test sets. (3) When more categories are included in the original test sets, it is more difficult for selection methods to effectively estimate the accuracy of the test input of various categories. However, on these test sets, solutions with ordinary performance from the PS obtained by DMOS can still maintain advantages over the ones obtained by PACE.

- RQ2: Can test subsets selected by DMOS have test capabilities similar to those of original test sets in terms of overall accuracy and test coverage?

(1) Design

For this research question, we conduct the Scott-Knott ESD test analysis on the results of DMOS-Best, DMOS-Medium, DMOS-Nocluster, and PACE on the eight test objects (a total of 144 groups of data, including 25 tests on the test object No. 8, 16 tests on each of the test objects No. 1–No. 5, 18 tests on the test object No. 6, and 21 tests on the test object No. 7). The analysis is carried out from six aspects, i.e., errors in overall accuracy estimation and estimation errors of the five test-coverage indicators of NC, NBC, SNAC, TKNC, and KMNC, to explore whether the test subsets selected by DMOS have more test properties similar to those of the original test sets in terms of other evaluation indicators than those selected by other methods.

(2) Results

As shown in Figs. 2–7, the errors of the various selection methods in estimating overall accuracy and test coverage should be as small as possible. Therefore, when a method stays more to the right in the ranking, it has better performance. The results reveal that the errors of DMOS-Best and DMOS-Median in estimating four test-coverage indicators are significantly smaller than those of PACE (in terms of KMNC, although DMOS-Best and DMOS-Median do not outperform PACE significantly, they still maintain a slight advantage), and both methods outperform PACE significantly in overall accuracy estimation. From the aspect of the upper bounds of the various methods in the boxplots, DMOS-Best and DMOS-Median are tied with and sometimes outperform PACE; however, the medians and lower bounds of the boxplots show that DMOS has lower estimation errors and more stable performance under the six indicators than PACE. According to Lines 3–7 in Table 11, the average overall-accuracy estimation errors of DMOS-Nocluster, DMOS-Best, DMOE-Median, and PACE in 144 tests on eight test objects are 1.238%, 1.081%, 1.211%, and 1.926% respectively. Compared with PACE, the other three methods have their respective errors reduced by 0.688%, 0.845%, and 0.715% on average, an average improvement of 35.72%. 43.87%, and 37.12%, respectively. The average estimation errors of DMOS-Nocluster, DMOS-Best, DMOE-Median, and PACE in terms of the five test-coverage indicators in 144 tests on the eight test objects are 8.775%, 8.739%, 8.763%, and 16.067%, respectively. Compared with PACE, the other three methods have their respective errors reduced by 7.292%, 7.328%, and 7.304% on average, an average improvement of 45.39%, 45.61%, and 45.46% respectively. This also indicates the performance superiority of DMOS.

Summary. (1) The test subsets selected by DMOS can be closer to the original test sets than those selected by PACE in terms of other test evaluation indicators. (2) Ensuring the precise accuracy estimation of the test input of various categories by test subsets is of great significance for ensuring the similarity of other properties between test subsets and original test sets.

- RQ3: How much do clustering and multi-objective optimization contribute to DMOS?

(1) Design

To investigate the contribution of clustering and multi-objective optimization to DMOS, we first evaluate the performance difference between DMOS-Nocluster and DMOS-Best that employs clustering in the accuracy of the test input of various categories, overall accuracy, and test coverage. Here, as DMOS-Nocluster also takes the solution with the smallest average error in the accuracy estimation of the test input of various categories in a PS as the result, we only compare DMOS-Nocluster with DMOS-Best for this RQ to explore the contribution of clustering to DMOS. Then, we compare the Pareto solution set obtained by DMOS with that obtained by RandomSearch in IGD to evaluate the convergence and diversity of the obtained solution sets from the perspective of multi-objective optimization.

(2) Results

Regarding the comparison with DMOS-Nocluster, we select the solution with the smallest average error in the accuracy estimation of the test input of various categories in the solution set obtained by DMOS-Nocluster as the final solution representing the performance of DMOS-Nocluster. Tables 2–6 indicate that DMOS-Nocluster also enjoys much better performance than PACE and is able to reach an estimation level of a smaller error earlier under fewer selections, but it is still slightly inferior to DMOS-Best. Considering the overall accuracy and test-coverage indicators, DMOS-Nocluster and DMOS-Best have no significant performance difference, but DMOS-Best still has slight advantages. The underlying logic behind clustering is that DMOS does not blindly trust final labels predicted by the DNN model to be tested but retains more properties by extracting the middle-level output of the model as the feature representation of the original test set. Thus, on this basis, when clustering is used to evaluate the data distribution of the original test set (proportions of test input of different categories), reference information of sampling for multi-objective optimization will be more accurate. Therefore, we suggest that clustering should be applied to estimate the distribution of each category of the original test sets before sampling in practical use.

Regarding the comparison with RandomSearch, as shown in Fig. 8, RandomSearch has unstable performance in IGD, and DMOS outperforms RandomSearch significantly. This shows that the search direction of RandomSearch fluctuates greatly during the optimization, with no obvious downward trend of the overall fluctuation, and it fails to find a suitable optimization direction. The comparison with RandomSearch demonstrates that multi-objective optimization of DMOS has better convergence, and the accordingly obtained solution set has better diversity. Moreover, it is necessary to design appropriate optimization objectives to solve DL test input selection, and it is impossible to find suitable solutions under random search directions.

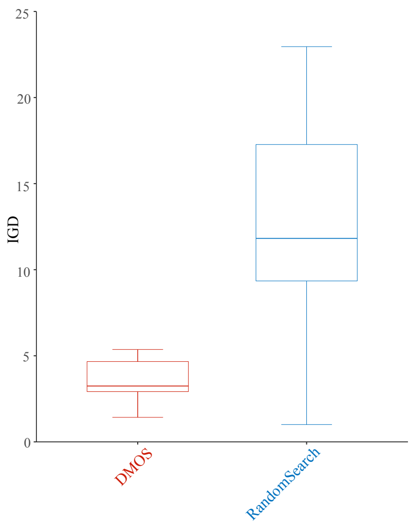

Figure 8 Evaluation results of IGD

Summary. (1) Clustering can effectively correct deviations caused by category labels predicted by DMOS using DNN models to be tested for better estimation of the data distribution of the original test sets. This can lay a good foundation for multi-objective optimization and ensure the performance of DMOS. (2) Designing an appropriate optimization objective is highly conducive to obtaining desired Pareto optimal selection solutions. The desired performance of DL test input selection cannot be easily achieved just through random search decisions.

- RQ4: How much is the runtime overhead of DMOS?

(1) Design

The runtime overhead of DMOS is mainly concentrated in the extraction of test-input features, dimension reduction, clustering, and multi-objective optimization. For the eight test objects, both DMOS and PACE select test subsets containing 1,000 test input samples from original test sets, and time is counted from model and data reading to completion of test subset selection. Finally, the runtime overhead of DMOS is compared with that of PACE.

(2) Results

Table 12 lists the test results. Test-object IDs in the first column correspond to relevant information in Table 1. The numbers in the second column represent the time overhead of DMOS in selecting a test subset of 1,000 test input samples under the current test object, while those in the third column stand for the time overhead of PACE in doing so. The bold elements indicate that the corresponding method has lower time overhead and better performance. It should be noted that the results also indicate that the time overhead (in seconds) of the existing test input selection methods is much less than that of manual labeling. The time overhead of DMOS on six test objects is less than that of PACE, which clearly suggests the high efficiency of DMOS. Meanwhile, DMOS is only slightly better than PACE on the test object No. 5; on the test object

Table 12 Time overhead analysis of DMOS and PACE (s)

ID	DMOS	PACE
1	**85**	203
2	**86**	113
3	86	**48**
4	**93**	202
5	**106**	120
6	**146**	233
7	4,683	**3,591**
8	**132**	1,967

No. 7, excessive categories (1,000 categories) increase the time overhead of each evolutionary iteration in multi-objective optimization, and hence, the performance of DMOS is poorer than that of PACE. For the test object No. 8, high complexity of the model middle-layer representation extracted by PACE according to recommended optimal parameters greatly increases the time overhead of MMD-critic-based prototype sampling, which results in a poorer performance of PACE than that of DMOS. The time overhead analysis of DMOS and PACE in selecting 1,000 test input samples from the eight test objects is shown in Table 12.

Summary. (1) The time overhead of existing DL test input selection methods is much less than that of manual labeling. Therefore, it is of high practical significance to design an effective DL test input selection method. (2) DMOS with multi-objective optimization is highly superior to the latest selection method PACE in terms of runtime overhead.

4 Discussions

4.1 Research on quality of Pareto solution sets obtained by DMOS

Test results of the previous section sufficiently show that selection solutions obtained by DMOS can effectively construct test subsets with test capabilities close to those of original test sets, but the quality of selection solutions obtained under the same number of selections is different (in terms of average errors in accuracy estimation of the test input of variable categories). From the perspective of medians, solutions obtained by DMOS are still better than those obtained by the existing selection methods. To further improve the quality of the solution sets obtained by DMOS in future research, we further analyzed the test results and drew the following conclusions.

(1) Increasing the number of selections will reduce performance fluctuations of solutions. With very few selections, all solutions perform poorly, with little difference in performance, but when the number grows, the performance difference increases accordingly. When the number reaches a certain value, repeated samples selected among selection solutions corresponding to various solutions increase, and thus, the performance difference between various solutions starts to decrease. For example, the test results show that for an original test set, a 10-category or 100-category dataset containing 10,000 samples, the performance of the obtained PS tends to stabilize when the number of selections is about 800.

(2) More categories contained in original test sets result in smaller performance fluctuations of solutions. More categories contained in original test sets produce higher difficulty in accurate estimation of all the categories, thus resulting in smaller performance differences between solutions. Thus, in the case of test sets with richer test-input categories, although the performance variations of solutions under different numbers of selections follow the laws described above, relevant fluctuation ranges are much smaller than those in the case of test sets with fewer categories.

(3) DMOS can produce Pareto solution sets of higher quality on test sets without any mutation. According to the test results, DMOS performs poorly for test sets containing adversarial test input samples that slightly disturb features to make models erroneous. The reason is as follows: although features of the adversarial test input samples are oflittle difference from those of the original test input samples, model-predicted results differ greatly. This makes it impossible for clustering to accurately estimate the data distribution of test sets, thus worsening the performance of multi-objective optimization.

4.2 Effectiveness threats

4.2.1 *Internal effectiveness threat*

Internal effectiveness threats mainly come from DMOS implementation, implementation of selection methods used in contrast tests, and script implementation for analyzing and evaluating all test results. To effectively reduce these threats, we relied on encapsulation algorithms in some existing mature frameworks in Python to implement DMOS. For other contrast methods, we used the latest versions from open-source links shared by these methods and adopted optimal parameters recommended in the original texts for testing. As to all the scripts for analyzing and evaluating test results, by coding, we realized multiple ways of mutual script checking to ensure the correctness of results. In addition, during the implementation, we examined all the code involved carefully.

4.2.2 *External effectiveness threats*

External effectiveness threats mainly come from the test objects studied, namely, DL test sets and DNN models to be tested. DNN models used are those trained by popular datasets (including MNIST, CIFAR-10, CIFAR-100, ImageNet, and Speech-Commands). In view of optimization objectives, we only considered the classification models. To reduce relevant threats, we considered different types of DNN models to be tested from the aspects of high-precision and low-precision models, as well as CNN and RNN models (i.e., DeepSpeech). For the DL test sets, we diversified their types to reduce external effectiveness threats, starting from different types of test input samples including image and speech ones. At present, no tests have been carried out on text data and related models in that the number of categories of data in some common text classification tasks (usually two categories, for example, spam recognition) is far less than that of image data (up to hundreds or even thousands of categories). As a result, the effects of various selection methods cannot be demonstrated fully. In addition, simple text classification tasks are similar to classification tasks of image data. Specifically, the middle-layer output of DNN models to be tested in learning text data are used as features for clustering; after data labeling, DMOS could carry out multi-objective optimization. It can also design optimization objectives for iterative solutions given the proportions of the test input of different categories in various clusters formed by clustering. Thus, we did not consider this kind of text data and related models temporarily. In future research, we will continue to collect test objects with more complex tasks and conduct more extensive research.

4.2.3 *Structural effectiveness threats*

Structural effectiveness threats mainly lie in specified parameters during method running in tests. In this study, these parameters mainly include clustering parameters, dimension-reduction parameters, feature type parameters (parameters to find out the output of what layer of the DNN model to be tested are selected as clustering features), multi-objective optimization parameters (population size and the number of evolutionary iterations). As to clustering and dimension-reduction parameters, we used the optimal parameters recommended by Chen *et al.*[17]. As to feature type parameters, we tried four parameters (-1, -2, 0, and 1) and finally selected a robust one as the recommended optimal parameter, which was introduced in Section 3.3. In addition, multi-objective optimization parameters mainly determine the scope of search space and the time of evolutionary iterations. According to parameter recommendations in other related work, we tried four groups of combinations (20–50, 50–100, 100–200, and 200–400) and selected the group of parameters with the best comprehensive performance in terms of both time cost and results. In future work, we will continue to study the influence of these parameters on the performance of the method.

5 Related Work

5.1 DL testing

Credibility problems in DNN models urge researchers to develop various technologies for effective and complete testing. In addition to the high efficiency of DL testing, test adequacy of DNN models to be tested is also a research focus in DL testing. Extensive studies have proposed many test-adequacy metrics to evaluate test methods, and test-input augmentation has become the major way to improve test adequacy.

In recent years, the most deeply studied metric of DL test adequacy is NC[25, 26, 54]. For example, DeepXplore[26] introduced NC to measure ratios of neurons with activation values higher than preset thresholds. Similarly, DeepGauge[25] introduced a series of test-adequacy criteria based on activation values of neurons. Recent studies have also proposed test standards and techniques driven by symbolic execution[55], coverage-guided fuzzing[56], and deformation and transformation[13]. Gerasimou *et al.*[58] proposed a test-adequacy criterion based on coverage of important neurons, i.e., Importance-Driven Coverage (IDC), of DNN models. Important neurons refer to those with core contributions to decision-making, namely, neurons with the greatest effect on results in the case of test input passing through neural networks. Related tests have proved that this criterion can comprehensively evaluate the adequacy of test sets effectively from the perspective of neuron activation.

Test-input augmentation aims to generate new test data on the basis of original test sets, so as to expose errors that cannot be exposed originally and then improve test adequacy. The research on test-input augmentation for DL systems can be divided into the following two types.

- Guided by test coverage, the first type uses specific mutation operators to augment test sets by transforming random seeds, so as to make the augmented test sets with higher test coverage. For example, Guo *et al.*[59] proposed using NC[26] as the final objective of DLFuzz to guide the mutation to generate new test inputs. During an iteration, DLFuzz introduces a small disturbance to test input samples and retains those able to increase NC as mutated test input samples of the next iteration.
- Focusing on the characteristics of DNNs, the second type produces adversarial samples. In other words, the features of the original test input samples are slightly disturbed and transformed so that the new test input samples are close to the original ones but able to trigger defects that the original ones fail to do so. For example, Szegedy *et al.*[60] used loss functions between adversarial samples and test results as objective functions to guide the generation of adversarial samples. Xiao *et al.*[61] proposed a method for pixel transformation in high-dimensional space and proved that adversarial samples produced by this method are of higher smoothness, smaller variations, and higher authenticity.

In contrast, our work endeavors to help developers reduce labeling costs by selecting small test subsets that can maintain the properties of the original test sets. Li *et al.*[15] first proposed CSS, which divides model-predicted confidence into intervals and then conducts sampling in each interval according to a certain proportion to form the final test subsets. Later, they proposed CES, which uses cross entropy to measure the difference between test subsets and original test sets and reduces the difference by multiple rounds of iterative sampling to form the final test subsets. Zhou *et al.*[16] proposed a two-stage sampling method, DeepReduce. The method first selects a small test subset according to NC and then adds more test data to the subset by using heuristic rules until the number of samples specified by users is reached. Chen *et al.*[17] proposed PACE for selective sampling. This method first conducts hierarchical clustering of data in original test sets and then samples data in normal and abnormal clusters through MMD-critic-based sampling[19] and adaptive random selection[20], respectively, so as to form test subsets through data merging. Test subsets generated by the existing methods are very close to original test sets in overall accuracy. However, given a small number of selections, they fail to cover the test input of all categories in the original test sets, and the accuracy estimation of such test input has large errors. Innovatively starting from multi-objective optimization, by clustering, this paper estimates the data distribution of the original test sets and designs optimization objectives to constantly reduce the distribution difference between test subsets and original test sets. This ensures that test subsets can fully cover and accurately estimate test input of different categories in the original test sets.

5.2 Test input selection

Traditional software testing faces increasing test costs due to redundant test input. Thus, it is necessary to screen original test sets to improve test efficiency, namely, to satisfy the same test functions of the original test sets under a minimized number. In traditional software testing, common algorithms for test-input reduction include greedy algorithms and heuristic algorithms[62–67], which mainly aim to remove redundant test input related to some test-capability metrics in the original test sets. Harrold *et al.*[68] proposed a method, Harrold-Gupta-Soffa (HGS), that combines the greedy strategy with heuristic search, which iteratively selects test input samples in the original test sets by referring to designed test requirements. Hua *et al.*[69] first used a genetic algorithm for evolutionary iterations of an original test set to obtain the optimal solution set of test input samples. Then, they employed an ant colony algorithm to further reduce the result to obtain an optimal test subset after minimization. Nie *et al.*[70] first analyzed the relationship between test-capability metrics and divided an original test set, and then, they sampled data in the divided small sets on the basis of a heuristic algorithm and a greedy algorithm separately to obtain a screened small-scale test subset. However, as DL systems are developed on the basis of data-driven programming paradigms, DL test input selection aims to test DL systems by selecting small-scale test subsets that can represent the test capabilities of the original test sets, which is distinctly different from test reduction in traditional software testing. Thus, it is impossible to directly use evaluation criteria and algorithms in traditional software test reduction for DL test input selection.

6 Conclusion

This paper modeled DL test input selection as a multi-objective optimization problem and innovatively proposed DMOS. It designed optimization objectives considering the accuracy of the test input of various categories and solved the problem by using a multi-objective genetic evolutionary algorithm to obtain an efficient small-scale test subset. We tested the performance of the method on eight test objects composed of DNN models and test sets. According to the results, DMOS can not only ensure the precise accuracy estimation of the test input of various categories in test sets but also make sure that the newly generated test subsets and the original test sets have other similar test properties (such as overall accuracy and test adequacy). Thus, it is significantly superior to the latest selection method PACE. In future research, we will attempt to measure the quality of test subsets from more evaluation aspects and extend DMOS to DNN models and test sets of regression tasks.

References

[1] Chen CY, Seff A, Kornhauser AL, Xiao JX. Deepdriving: Learning affordance for direct perception in autonomous driving. Proc. of the 2015 IEEE Int'l Conf. on Computer Vision. 2015. 2722–2730.

[2] Sun Y, Chen YH, Wang XG, Tang XO. Deep learning face representation by joint identification-verification. Advances in Neural Information Processing Systems 27: Annual Conf. on Neural Information Processing Systems. 2014. 1988–1996.

[3] Goodfellow IJ, Bengio Y, Courville AC. Deep Learning. Adaptive Computation and Machine Learning. MIT Press, 2016.

[4] LeCun Y, Bengio Y, Hinton GE. Deep learning. Nature, 2015, 521(7553): 436–444.

[5] Obermeyer Z, Emanuel EJ. Predicting the future—Big data, machine learning, and clinical medicine. The New England Journal of Medicine, 2016, 375(13): 1216.

[6] Julian KD, Lopez J, Brush JS, Owen MP, Kochenderfer MJ. Policy compression for aircraft collision avoidance systems. Proc. of the 35th IEEE/AIAA Digital Avionics Systems Conf. 2016. 1–10.

[7] Chen JJ, He XT, Lin QW, Xu Y, Zhang HY, Hao D, Gao F, Xu ZW, Dang YN, Zhang DM. An empirical investigation of incident triage for online service systems. Proc. of the 41st Int'l Conf. on Software Engineering: Software Engineering in Practice. 2019. 111–120.

[8] Chen JJ, He XT, Lin QW, Zhang HY, Hao D, Gao F, Xu ZW, Dang YN, Zhang DM. Continuous incident triage for large-scale online service systems. Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. 2019. 364–375.

[9] Li X, Li W, Zhang YQ, Zhang LM. DeepFL: Integrating multiple fault diagnosis dimensions for deep fault localization. In: Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. 2019. 169–180.

[10] Zhang X, Xu Y, Lin QW, Qiao B, Zhang HY, Dang YN, Xie CY, Yang XS, Cheng Q, Li Z, Chen JJ, He XT, Yao R, Lou JG, Chintalapati M, Shen FR, Zhang DM. Robust log-based anomaly detection on unstable log data. Proc. of the 2019 ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. 2019. 807–817.

[11] Wang Z, You H, Chen J, Zhang Y, Dong X, Zhang W. Prioritizing test inputs for deep neural networks via mutation analysis. Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering. 2021. 397–409.

[12] Taigman Y, Yang M, Ranzato MA, Wolf L. DeepFace: Closing the gap to human-level performance in face verification. Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition. 2014. 1701–1708.

[13] Tian YC, Pei KX, Jana S, Ray B. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. Proc. of the 40th Int'l Conf. on Software Engineering. 2018. 303–314.

[14] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang ZH, Karpathy A, Khosla A, Bernstein MS, Berg AC, Li FF. Imagenet large scale visual recognition challenge. Int'l Journal of Computer Vision, 2015, 115(3): 211–252.

[15] Li ZN, Ma X, Xu C, Cao C, Xu J, Lu J. Boosting operational DNN testing efficiency through conditioning. Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. 2019. 499–509.

[16] Zhou JY, Li F, Dong JH, Zhang H, Hao D. Cost-effective testing of a deep learning model through input reduction. Proc. of the 31st IEEE Int'l Symp. on Software Reliability Engineering. 2020. 289–300.

[17] Chen JJ, Wu Z, Wang Z, You HM, Zhang L, Yan M. Practical accuracy estimation for efficient deep neural network testing. ACM Trans. on Software Engineering and Methodology, 2020, 29(4): 1–35.

[18] McInnes L, Healy J, Astels S. HDBScan: Hierarchical density based clustering. The Journal of Open Source Software, 2017, 2(11): Article No.205.

[19] Kim B, Khanna R, Koyejo O. Examples are not enough, learn to criticize! Criticism for interpretability. Proc. of the 30th Int'l Conf. on Neural Information Processing Systems. 2016. 2288–2296.

[20] Chen TY, Kuo FC, Merkel RG, Tse TH. Adaptive random testing: The ART of test case diversity. Journal of Systems and Software, 2010, 83(1): 60–66.

[21] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002, 6(2): 182–197.

[22] Liu WB, Wang ZD, Liu XH, Zeng NY, Liu YR, Alsaadi F. A survey of deep neural network architectures and their applications. Neurocomputing, 2017, 234: 11–26.

[23] Gatys LA, Ecker AS, Bethge M. Image style transfer using convolutional neural networks. Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. 2016. 2414–2423.

[24] Lai SW, Xu L, Liu K, Zhao J. Recurrent convolutional neural networks for text classification. Proc. of the 29th AAAI Conf. on Artificial Intelligence. 2015. 2267–2273.

[25] Ma L, Juefei-Xu F, Zhang FY, Sun JY, Xue MH, Li B, Chen C, Su T, Li L, Liu Y, Zhao JJ, Wang YD. DeepGauge: Multi- granularity testing criteria for deep learning systems. Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering. 2018. 120–131.

[26] Pei KX, Cao YZ, Yang J, Jana S. DeepXplore: Automated whitebox testing of deep learning systems. Proc. of the 26th Symp. on Operating Systems Principles. 2017. 1–18.

[27] Feng Y, Shi QK, Gao XY, Wan J, Fang CR, Chen ZY. DeepGini: Prioritizing massive tests to enhance the robustness of deep neural networks. Proc. of the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis, Virtual Event. 2020. 177–188.

[28] Quinlan JR. Induction of decision trees. Machine Learning, 2004, 1: 81–106.

[29] Zhang L, Sun XC, Li Y, Zhang Z. A noise-sensitivity-analysis-based test prioritization technique for deep neural networks. arXiv:1901.00054, 2019.

[30] Ma W, Papadakis M, Tsakmalis A, Cordy M, Traon YL. Test selection for deep learning systems. ACM Trans. on Software Engineering and Methodology, 2021, 30(2): 1–22.

[31] Han Y, Gong DW, Jin Y, Pan QK. Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. IEEE Trans. on Cybernetics, 2019, 49(1): 184–197.

[32] Qi R, Yen G. Hybrid bi-objective portfolio optimization with pre-selection strategy. Information Sciences, 2017, 417: 401–419.

[33] Liu YP, Yen GG, Gong DW. A multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies. IEEE Trans. on Evolutionary Computation, 2019, 23(4): 660–674.

[34] Deb K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation, 1999, 7(3): 205–230.

[35] Zitzler E, Laumanns M, Thiele L. Spea2: Improving the strength Pareto evolutionary algorithm. Technical Report, 103, Computer Engineering and Networks Laboratory (TIK), 2001.

[36] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans. on Evolutionary Computation, 2007, 11(6): 712–731.

[37] Shi QK, Chen Z, Fang CR, Feng Y, Xu B. Measuring the diversity of a test set with distance entropy. IEEE Trans. on Reliability, 2016, 65(1): 19–27.

[38] Hartigan JA, Wong MA. Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society: Series C (Applied Statistics), 1979, 28(1): 100–108.

[39] Warden P. Speech commands: A public dataset for single-word speech recognition. arXiv:1804.03209, 2017.

[40] Kurakin A, Goodfellow IJ, Bengio S. Adversarial examples in the physical world. arXiv:1607.02533, 2016.

[41] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.

[42] Kim JH, Feldt R, Yoo S. Guiding deep learning system testing using surprise adequacy. Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering. 2019. 1039–1049.

[43] Wilcoxon F. Individual comparisons by ranking methods. Breakthroughs in Statistics, 1992: 196–202.

[44] Kocaguneli E, Menzies T, Keung JW, Cok D, Madachy R. Active learning and effort estimation:

Finding the essential content of software effort estimation data. IEEE Trans. on Software Engineering, 2013, 39(8): 1040–1053.

[45] Li M, Zhang H, Wu RX, Zhou Z. Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engineering, 2011, 19(2): 201–230.

[46] Valentini G, Dietterich TG. Low bias bagged support vector machines. Proc. of the 20th Int'l Conf. on Machine Learning. 2003. 752–759.

[47] Jelihovschi EG, Faria JC, Allaman I. ScottKnott: A package for performing the Scott-Knott clustering algorithm in R. Trends in Applied and Computational Mathematics, 2014, 15(3): 3–17.

[48] Keras. Accessed. 2021. https://keras.io

[49] Tensorflow. 2021. https://www.tensorflow.org/

[50] Fastica. 2021. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

[51] Hdbscan. 2021. https://pypi.org/project/hdbscan/

[52] Geatpy. 2021. http://geatpy.com/

[53] Zhang J, Harman M, Ma L, Liu Y. Machine learning testing: Survey, landscapes and horizons. arXiv:1906.10742, 2019.

[54] Sun Y, Huang X, Kroening D. Testing deep neural networks. arXiv:1803.04792, 2018.

[55] Gopinath D, Wang KY, Zhang MS, Pasareanu C, Khurshid S. Symbolic execution for deep neural networks. arXiv:1807.10439, 2018.

[56] Odena A, Goodfellow IJ. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. Proc. of the 36th Int'l Conf. on Machine Learning. 2019. 4901–4911.

[57] Xie XF, Ma L, Juefei-Xu F, Xue MH, Chen HX, Liu Y, Zhao JJ, Li B, Yin JX, See S. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. 2019. 146–157.

[58] Gerasimou S, Eniser HF, Sen A, Cakan A. Importance-driven deep learning system testing. Proc. of the 42nd IEEE/ACM Int'l Conf. on Software Engineering: Companion Proceeding. 2020. 322–323.

[59] Guo JM, Jiang Y, Zhao Y, Chen Q, Sun J. DLFuzz: Differential fuzzing testing of deep learning systems. Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. 2018. 739–743.

[60] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R. Intriguing properties of neural networks. arXiv:1312.6199, 2014.

[61] Xiao CW, Zhu JY, Li B, He W, Liu M, Song D. Spatially transformed adversarial examples. arXiv:1801.02612, 2018.

[62] Zhang L, Marinov D, Khurshid S. An empirical study of JUnit test-suite reduction. Proc. of the 22nd IEEE Int'l Symp. on Software Reliability Engineering. 2011. 170–179.

[63] Shi A, Yung T, Gyori A, Marinov D. Comparing and combining test-suite reduction and regression test selection. Proc. of the 10th Joint Meeting on Foundations of Software Engineering. 2015. 237–247.

[64] Rothermel G, Harrold MJ, von Ronne J, Hong C. Empirical studies of test-suite reduction. Software Testing, 2002, 12(4): 219–249.

[65] Chen J, Bai Y, Hao D, Zhang L, Xie B. How do assertions impact coverage-based test-suite reduction? Proc. of the 2017 IEEE Int'l Conf. on Software Testing. 2017. 418–423.

[66] Chen T, Lau M. A new heuristic for test suite reduction. Information and Software Technology, 1998, 40(5-6): 347–354.

[67] Cruciani E, Miranda B, Verdecchia R, Bertolino A. Scalable approaches for test suite reduction. Proc. of the 41st IEEE/ACM Int'l Conf. on Software Engineering. 2019. 419–429.

[68] Harrold MJ, Gupta R, Soffa M. A methodology for controlling the size of a test suite. ACM Trans. on Software Engineering and Methodology, 1993, 2(3): 270–285.

[69] Hua L, Wang CY, Gu Q, Cheng H. Test case set reduction based on genetic ant colony algorithm. Journal of Engineering Mathematics, 2012, 29(4): 486–492 (in Chinese with English abstract).

[70] Nie CH, Xu BW. A method of generating minimum test case set. Journal of Computer Science, 2003, 26(12): 1690–1695 (in Chinese with English abstract).

Yanzhou Mu, master. His research interests include machine learning, concurrent program analysis, and deep learning testing.

Junjie Chen, Ph.D., associate professor, doctoral supervisor. His research interests include software analysis and testing.

Zan Wang, Ph.D., professor, doctoral supervisor. His research interests include software testing and machine learning.

Jingke Zhao, master's degree candidate. His research interest is safety and quality assurance of deep learning.

Xiang Chen, Ph.D., associate professor. His research interests include software defect prediction, software defect location, regression testing, and combinatorial testing.

Jianmin Wang, Ph.D., assistant research fellow. His research interests include intelligent software testing and system simulation.