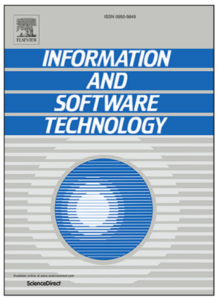

Stratified random sampling for neural network test input selection

Zhuo Wu, Zan Wang, Junjie Chen, Hanmo You, Ming Yan,
Lanjun Wang

PII:	S0950-5849(23)00186-6
DOI:	https://doi.org/10.1016/j.infsof.2023.107331
Reference:	INFSOF 107331

To appear in: *Information and Software Technology*

Received date :	15 February 2023
Revised date :	17 July 2023
Accepted date :	8 September 2023

Stratified Random Sampling for Neural Network Test Input Selection

Zhuo Wu[a], Zan Wang[a,b], Junjie Chen[b], Hanmo You[b], Ming Yan[b], Lanjun Wang[a,*]

[a] *School of New Media and Communication, Tianjin University, Tianjin, China*
[b] *College of Intelligence and Computing, Tianjin University, Tianjin, China*

Abstract

Context: Testing techniques to ensure the quality of deep neural networks (DNNs) are essential and crucial. However, the testing process can be inefficient due to a large number of test cases and the manual effort of labeling them. Recent work tackles the above challenge by selecting a small but representative subset of the tests. Such an approach allows us to quickly estimate the accuracy of a DNN with reduced effort, because only a small set of tests are to be manually labeled. However, existing approaches cannot guarantee unbiased results or provide an accurate estimation.

Objectives: In this work, we leverage a statistical perspective on providing an unbiased estimation of the model accuracy with the smallest estimation variance, named Stratified random Sampling with Optimum Allocation (SSOA).

Methods: Our approach first divides the unlabeled test set into strata based on predictive confidences. Then, we design two stratum accuracy variance estimation methods to allocate the given budget assigned to each stratum based on the optimum allocation strategy. Finally, we conduct multiple experiments to evaluate the effectiveness and stability of SSOA by comparing it with baseline methods.

Results: The results show that SSOA significantly outperforms all compared

*Corresponding author
Email addresses: wuzhuo@tju.edu.cn (Zhuo Wu), wangzan@tju.edu.cn (Zan Wang), junjiechen@tju.edu.cn (Junjie Chen), youhanmo@tju.edu.cn (Hanmo You), yanming@tju.edu.cn (Ming Yan), wang.lanjun@outlook.com (Lanjun Wang)

approaches with average improvements over 26.14% in terms of Mean Squared Errors (MSE) of estimated accuracy. In addition, the MSE shows a steady downward trend as the budget grows.

Conclusion: SSOA can assist testers in estimating the accuracy of DNNs, lowering labeling costs, and enhancing the efficiency of DNN testing.

Keywords:
Deep Neural Network Testing, Test Input Selection, Test Optimization

1. INTRODUCTION

Deep neural network (DNN) has become an indispensable tool for a wide range of applications such as image classification [1], speech recognition [2], and natural language processing [3]. Like traditional software, DNN also contains bugs, which might threaten human lives or properties in safety-critical scenarios, e.g., autonomous driving [4], and thus must be adequately tested. However, testing DNN-based software is different from traditional software because traditional software are programmed manually based on the business logic, whereas DNNs are constructed based on a data-driven optimization process [5]. In general, with the development of sensing technology, collecting sufficient amounts of real-world test inputs is no longer a bottleneck. For example, we can easily take millions of photos. However, labeling data to obtain the ground truth of the images is tedious and expensive [6].

In order to reduce the cost of manual labeling, researchers have defined the problem of DNN test input selection [7]. Specifically, this problem is to select a subset of test cases under a given sample size budget and label the selected test cases only to estimate the model performance (i.e., accuracy). We divide the existing DNN test input selection methods into two categories, i.e., heuristic-based approaches [8, 9] and sampling-based approaches [7, 10].

Heuristic-based selection methods [8, 9] usually use a specific criterion as the optimization goal and then sample the test cases under budget constraints to reach the corresponding goal with a greedy strategy. The limitations of heuristic-based methods are two-fold. Firstly, the estimation error of heuristic-based methods may not decrease with the increasing budget. This is because that fundamentally those samples selected by the greedy sampling strategy are neither identical nor independent, and thus the law of large numbers cannot be applied. As a result, the distribution of the selected test cases may not match that of the original test set. For example, we conducted a pilot study on ResNet-50 based on ImageNet with a budget of 50. The p-value of Kolmogorov-Smirnov test [11] between the label distributions of samples selected by PACE [9] and the original test set is 0.03 (less than 0.05), which demonstrates the inconsistency between the two distributions. Secondly, existing heuristic-based selection methods rely on the inner information from the model, e.g., neuron coverage [8] and the output of the last hidden layer [9]. However, since the model is often considered a valuable property, and releasing inner information might cause model inversion attacks and alike [12], such details are seldom released; thus, black-box testing is considerably more practical.

Sampling-based selection methods [7, 10] do not have the above limitations. However, sampling-based methods still suffer from other drawbacks. First, when the number of samples is small, especially less than 100, the variance of the accuracy estimation of the sampling-based selection methods is much higher than those heuristic-based methods. Taking RestNet-50 trained by ImageNet as an example, we evaluate the effectiveness of accuracy estimation with a budget of 50. For PACE [9] (i.e., a heuristic-based approach), the square root of MSE is 0.04, while for SRS [7] (i.e., a sampling-based approach), the square root of MSE is 0.07. Second, existing sampling-based methods lost sight of theoretical statistics analysis of test inputs (e.g., minimum estimation variance [7, 10]), which is important in accuracy estimation [13].

To alleviate the above limitations, we propose a method called *Stratified random Sampling and the corresponding Optimum Allocation* (SSOA) to solve the test input selection task for DNNs. According to sampling theory [14], **stratified random sampling** involves dividing a population into smaller groups (a.k.a. strata). **The optimum allocation** is a procedure for assigning the sampling budget to each stratum. The allocation procedure is called "optimum" because, in stratified random sampling, it produces the smallest variance for estimating the mean of populations, which is the model test accuracy in the test input selection task for DNNs, given a fixed budget. The end-to-end SSOA approach contains five steps: 1) divide the large test set into strata; 2) compute the stratum accuracy variance; 3) calculate the quota of each stratum; 4) apply simple random sampling to obtain the corresponding quota in each stratum; and 5) label selected samples and compute the overall accuracy estimation by the weighted mean of strata sample sizes. In this procedure, there are two main challenges. The first one is how to stratify the original test set into strata to minimize the intra-group confidence variance and maximize the inter-group variance. The second one is that the variance of the accuracy in each stratum cannot be observed due to the lack of labels, but the optimum allocation relies on it to obtain the budget assignment for each stratum.

Regarding the first challenge, we divide the whole test set based on the predictive confidence of the DNN. Further, we propose two approaches to stratify the test set into small groups. The first one is to cluster the tests such as the intra-group confidence variance is small and the inter-group variance is large. The second one is to stratify the test set based on pre-set rules due to the prior knowledge of the confidence stratification in [7]. Regarding the second challenge, we propose two ways to estimate the stratum accuracy variance. The first one is to leverage training data as auxiliary information. However, the first variance estimation method is based on the assumption that the training dataset is accessible and the training dataset shares the distribution of the operation dataset. In a case of either of these assumptions is not established, an alternative approach, pre-sampling, is designed to estimate the stratum accuracy variance by conducting simple random sampling and labeling of a small set of tests.

We compare SSOA with the state-of-the-art sampling-based test input selection methods [7, 10] by the performance of accuracy estimation based on the selected subset. Following previous studies [9, 10, 7], the evaluation metric is the Mean Square Error (MSE) between the accuracy of the selected subset and the accuracy of the original large-scale test set. The experimental results demonstrate that SSOA significantly outperforms all the four baseline methods (i.e., Simple Random Sampling [7], Cross Entropy-based Sampling [7], Confidence-based Stratified Sampling [7], and DeepEST [10]) with an average improvements across different budgets of 26.14%, 35.63%, 39.26%, and 48.62%, respectively. Besides evaluating the feasibility of the solutions to the above two challenges, i.e. stratification strategies and stratum accuracy variance estimation approaches, we also illustrate that the stability of SSOA outperforms heuristic-based methods.

In summary, our contributions are as follows:

- We propose an unbiased DNN test input selection framework, SSOA, based on stratified random sampling and optimum allocation. It achieves the minimum estimation variance theoretically.
- We design stratification methods and stratum variance estimation methods to implement SSOA.
- We conduct experiments on five common scenarios to evaluate the effectiveness of SSOA. Meanwhile, we also show the stability of SSOA by comparing it with heuristic-based methods. We have released the implementation of the framework on [15] for future usages.

2. Related Work

There are two main lines of related works, i.e., test input selection and test input prioritization.

2.1. DNN Test Input Selection

Multiple test input selection techniques [7, 9, 8, 10] have been proposed to solve this problem. In general, we categorize the existing methods into sampling-based and heuristic-based methods.

2.1.1. *Sampling-based Selection*

The simplest sampling-based test input selection method is simple random sampling (SRS). SRS randomly selects the specified number of test inputs from the original test set. The probability of each test input being selected is the same. As a result, SRS is a method of unbiased sampling.

Li et al. [7] proposed two sampling-based selection methods, which are Confidence-based Stratified Sampling (CSS) and Cross Entropy-based Sampling (CES). CSS applies the method of stratified random sampling [14]. More details of stratified random sampling will be introduced in Section 3.2. CSS first divides the original test set into different strata according to their confidences. Then, CSS is randomly sampled from each stratum according to a manually decided fixed proportion. CES leverages the idea of clustering sampling [14]. More specifically, CES conducts selection by minimizing the cross-entropy between the selected set and the whole test set. According to [7], both CSS and CES are unbiased.

Furthermore, Guerriero et al. [10] leveraged adaptive sampling to select inputs and proposed a test input selection method, DeepEST. Adaptive sampling has also been used in the software reliability assessment task [16]. At each step, DeepEST applies SRS with a fixed probability r and applies weight-based sampling (WBS) with a probability $1-r$. [10] provided four approaches to calculate the weights. Besides, as observed in [10], WBS prefers to select the samples with wrong predictions, which makes the estimated accuracy low. Thus, Guerriero et al [10] propose to apply Hansen-Hurwitz [17] to correct the systematic bias in sampling and obtained an unbiased estimator. DeppEST has a slightly different goal, as it tries to improve the tradeoff between having unbiased low-variance estimate with exposing more mis-predictions.

2.1.2. *Heuristic-based Selection*

Heuristic-based methods often start with a certain criterion as the optimization goal, then select the least number of samples to meet the criterion through a greedy strategy, and finally obtains the desired test inputs.

However, heuristic-based methods do not consider whether the estimation is unbiased, and thus their results provide no guarantee.

Chen et al. [9] proposed a cluster-based method, PACE. Specifically, PACE first utilizes HDBSCAN [18] to cluster the original large-scale test set into several groups representing different test capabilities. Then, PACE uses MMD-critic [19] to select key test inputs which represent the cluster and performs adaptive random selection [20] among the unique test inputs. Similarly, Zhou et al. [8] proposed a two-phase heuristic-based method, DeepReduce. In the first stage, DeepReduce uses structural neuron coverage criteria [21, 22] and the HGS [23] algorithm to select a subset of test inputs. Then, DeepReduce designs to use relative entropy (i.e., KL divergence) as a guide for the selection of test inputs in the second stage.

2.1.3. Comparison Analysis

Compared with heuristic-based methods, sampling-based methods have three major advantages: 1) based on existing sampling theory [14], unbiased test input selection methods provide theoretical guarantees. Since the error of the estimator is controllable, the generalization ability of sampling-based selection is better than heuristic-based selection. 2) Owing to the law of large numbers, when we use a sampling-based method for test input selection, the MSE between the estimated value of accuracy and the ground truth steadily decreases as the sample size budget increases. However, for the heuristic-based method, a lower MSE might be achieved with a small budget, and thus it is hard to decide whether to fully use the given budget. More details can be referred to Section 6.4. 3) The sampling-based selection can rely on the model outputs but does not need to extract the output of the middle layer of the model as auxiliary information as in PACE [24] and DeepReduce [8], which indicates that the sampling-based test input selection is a black-box process.

We focus on developing a sampling-based DNN test input selection framework. More specifically, our study is based on stratified random sampling and its optimum allocation, which will be introduced next.

2.2. DNN Test Input Prioritization

Regarding test input prioritization, many approaches [5, 25, 26, 27, 28, 29, 30, 6, 31] based on the bug-revealing capability of test inputs have been proposed to optimize DNN testing. Feng et al. [5] proposed DeepGini, a test prioritization technique based on a statistical perspective of DNN. DeepGini borrowed the idea of Gini impurity in information theory and applied it to the analysis of deep neural networks. The main argument is that the more evenly the model predicted an input as each class, the more possibly the model would misclassify, indicating that test input is more likely to be a bug-revealing input. Wang et al. [6] proposed PRIMA, a test prioritization technique based on mutation analysis. When testing resources are limited, priority execution of important test cases can improve testing efficiency. PRIMA mutates the deep learning model and test input separately. Then, PRIMA measures the degree of exploration of the model and itself by the test input, finds test cases with a strong ability to uncover errors, and ranks them at the front of the test sequence. Deep neural network software developers prioritize these top-ranked test inputs for final testing. Shen et al. [31] proposed MCP, which improves the accuracy of the DNN model by using test inputs near the decision boundary for retraining.

The main difference between test input prioritization and test input selection is that test input selection is to make a precise estimate of the accuracy of the model by selecting samples, and test input prioritization is to preferentially select test inputs that will be predicted wrongly. Please note that the definition of "Test input selection" mentioned in some papers is different from that in our paper, and the goal of the task is also different [32, 33]. In these papers [32, 33], the goal of "Test input selection" is the same as test input prioritization, which is to select test inputs with strong bug-revealing ability.

3. PRELIMINARIES

In this section, we describe the problem of DNN test input selection. Moreover, we provide necessary background on stratified random sampling, which we will use in Section 4 to solve the problem.

3.1. Problem Formulation

Definition 1 (DNN test input selection)**.** *Given a trained DNN model $\mathcal{M}$, and an unlabeled test sample set $\mathbb{S}$ with N samples, the task of* ***DNN test input selection*** *is to select a subset $\mathbb{T}$, with a given size budget $n < N$, such that the model test accuracy is accurately estimated by $\mathbb{T}$.*

Due to the huge manual effort required to label the whole test set $\mathbb{S}$, the ground-truth model test accuracy, denoted as θ, cannot be observed. The task of test input selection aims to reduce the workloads by selecting a subset $\mathbb{T}$. We denote the model test accuracy estimated with the selected test cases $\mathbb{T}$ as $\hat{\theta}$. The estimation error is measured using mean square error (MSE), which is defined as:

$$MSE(\hat{\theta}) = \mathbf{E}\{(\hat{\theta} - \theta)^2\} \tag{1}$$

The MSE can also be written as the sum of the variance of the estimator and the squared bias of the estimator:

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias^2(\hat{\theta}) \tag{2}$$

where $Var(\hat{\theta})$ is the variance of the estimation and $Bias(\hat{\theta})$ is the system bias of the estimation. If the estimator is unbiased, i.e. $\mathbf{E}\{\hat{\theta}\} = \theta$, then MSE and variance are asymptotically equivalent.

3.2. Stratified Random Sampling and Optimum Allocation

In this study, we apply stratified random sampling and its optimum allocation to the task of test input selection.

In stratified random sampling [14], a full set $\mathbb{S}$ formed by N samples is divided into L strata $\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_L$ having $N_1, N_2, \ldots, N_L$ samples respectively. These strata do not overlap, i.e. $\mathbb{S}_k \cap \mathbb{S}_l = \emptyset$, $k \neq l$ and together form the full set, i.e. $\cup_{l=1}^{L} \mathbb{S}_l = \mathbb{S}$. In addition, there also is $\sum_{l=1}^{L} N_l = N$.

Once the strata are defined, and given a sample size budget n, an independent sample of size n_l is selected from N_l samples in stratum $\mathbb{S}_l (l = 1, \ldots, L)$ such that $\forall l$, $n_l \leq N_l$, and $\sum_{l=1}^{L} n_l = n$.

In this study, as Definition 1, the observation variable is the accuracy, thus we define the accuracy on each sample x as

$$\theta(x) = \begin{cases} 1 & x \text{ is correctly predicted} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Then the test accuracy on the full test set is defined as $\theta = \frac{1}{N}\sum_{i=1}^{N}\theta(x_i)$. Meanwhile, the test accuracy on the l-th stratum is defined as $\theta_l = \frac{1}{N_l}\sum_{i=1}^{N_l}\theta(x_i)$. We select n_l samples randomly to observe the accuracy, the estimation of test accuracy on the l-th stratum is as:

$$\hat{\theta}_l = \frac{1}{n_l}\sum_{i=1}^{n_l}\theta(x_i) \tag{4}$$

Since SRS achieves an unbiased estimation, there is $\mathbf{E}\{\hat{\theta}_l\} = \theta_l$. Therefore, the overall estimation $\hat{\theta}$ is the unbiased estimator of θ, which is:

$$\theta = \frac{1}{N}\sum_{l=1}^{L} N_l\theta_l = \frac{1}{N}\sum_{l=1}^{L} N_l\mathbf{E}\{\hat{\theta}_l\} = \mathbf{E}\{\hat{\theta}\} \tag{5}$$

As the task of test input selection aims to obtain the minimum MSE of the accuracy estimation as well as the estimator being unbiased, we only need to obtain the minimum variance of the estimation based on Equation 2. Then, according to [14], we achieve this goal by applying optimum allocation to choose the sample size in each stratum, $n_1, n_2, \ldots, n_L$, which is:

$$n_l = n\frac{N_l\sigma_l}{\sum\limits_{h=1}^{L} N_h\sigma_h} \tag{6}$$

where $\sigma_l^2 = Var(\theta_l)$ represents the variance of test accuracy on the l-th stratum.

4. APPROACH

The sampling-based approaches have been claimed to be more advantageous than heuristic-based approaches in Section 2.1.3. However, the existing

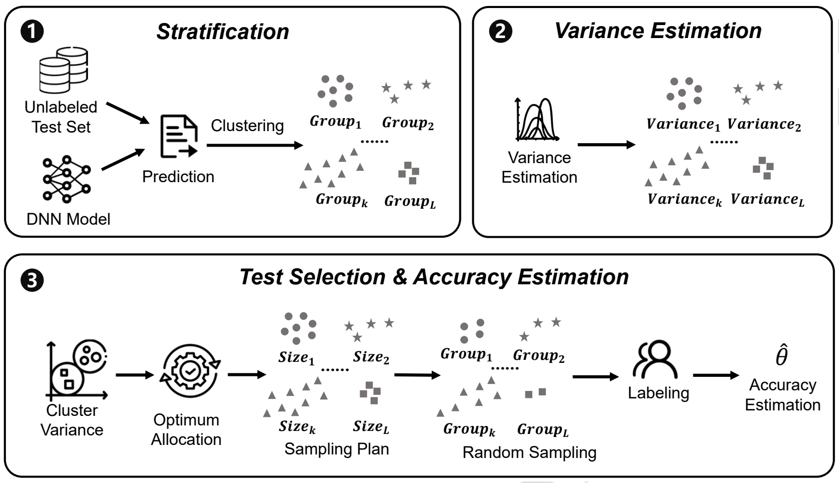

Figure 1: Overview of SSOA

sampling-based approaches [7, 10] do not utilize a more efficient allocation strategy and can introduce vastly large variance with a small budget. Motivated by this, we propose an unbiased test input selection framework based on Stratified random Sampling and Optimum Allocation, called SSOA. More specifically, we randomly sample test inputs from each stratum and adopt optimum allocation to determine the sampling size, which has been proven optimal allocation [14].

Figure 1 presents the overview of our framework. Firstly, to keep the distribution of the sampled test cases consistent with the distribution of the original test set, we stratify the whole test set, so that similar test inputs are stratified into the same group. Secondly, we obtain the accuracy variance of each stratum. Thirdly, we use the optimum allocation to determine the number of samples to select from each stratum based on Equation 6. Fourthly, we randomly sample in each stratum according to the results of the optimum allocation to obtain the final selected test set. Finally, we label the selected subset and estimate the accuracy.

In the above process, two key questions must be answered:

1. How to stratify the whole test dataset $\mathbb{S}$?

2. How to obtain the accuracy variance in each stratum, i.e. σ_l in Equation

6?

To answer question 1), as our method is designed to work in a black-box setting, we use the confidence of each test input as the basis for stratification. Two stratification methods are proposed in Section 4.1, which are automatic stratification by clustering and pre-set rules.

To answer question 2), since it is impossible to obtain the actual accuracy variance in each stratum without labeling. We propose two methods to estimate the accuracy variance in each stratum. The first method is to put training data into strata and estimate the accuracy variance of each stratum based on the training accuracy. The second method is pre-sampling, which is to randomly select a small number of test inputs h for labeling to estimate the variance of each stratum.

In this section, we first describe the stratification methods in Section 4.1, and then present the variance estimation methods in Section 4.2. Then, we summarize the usage of our SSOA in Section 4.3. Finally, we analyze that SSOA provides an unbiased sampling.

4.1. Stratification

Based on the sampling theory [14], an effective stratification should partition the samples such that the intra-stratum difference is minimized whereas the inter-stratum difference is maximized. Since we assume that the model, and its architecture, are unknown, we use the confidence as the index to divide the group, where confidence of a test input is the largest value in the output layer of the model. The confidence represents the credibility of the predicted label given by the model. Test cases with high confidence are more likely to be predicted correctly by the model, while test cases with low confidence are more likely to be predicted wrongly [34, 35].

Next, we propose two stratification methods: automatic confidence interval division by clustering and confidence interval bining based on experience. Both of them can effectively distinguish test inputs with different test capabilities.

The first stratification method is clustering. When the intra-stratum variance is small, and the inter-stratum variance is large, stratified random sampling is effective. To achieve this goal, we adopt the K-Means algorithm [36] to cluster all test inputs. There are two reasons to adopt the algorithm. Firstly, the K-Means clustering algorithm assigns every sample a group; that is to say, none of the samples are treated as the noise outside clusters. Alternative clustering algorithms (such as HDBSCAN [18]) may leave behind some samples as noises. Although we can assign the remaining samples into one stratum, the variance of this stratum will be huge, which damages the performance of optimum allocation. However,when we use K-means, the remaining samples are not all assigned into the same group. Instead, each remaining sample will be divided into as close a group as possible, and the variance of that group will still be smaller than that of the group full of noises. Secondly, the K-Means algorithm is a widely used and efficient clustering algorithm.

The second stratification method is stratification based on confidence. As introduced, this rule-based setting option is inspired by the fact that CSS [7] has used manually set dataset partition and achieved better performance than SRS in most scenarios. However, as CSS does not apply optimum allocation, its performance is not as good as SSOA, even with the same partition. More detailed comparisons are elaborated in Section 6.2.

In addition, in the presence of corner cases on which human understanding is more reliable, a rule-based division setting is more effective than clustering. For example, suppose 50% of the test inputs have confidence as 1.0, rest of them are distributed in the range of $[0, 1)$; the clustering method inclines to cluster the top 50% together with some other high confidence samples (e.g., more than 0.98) into one stratum. However, based on the effectiveness of the optimum allocation, a better setting is to set a stratum only containing samples with confidence 1.0. This is because the accuracy estimation of this stratum is 1, and the accuracy variance estimation is 0; then, based on Equation 6, there is no need to sample from this stratum. That is, the other strata can be allocated more budgets and have better estimations.

4.2. Variance Estimation

We use optimum allocation described in Section 3.2 to determine the number of samples in each stratum. As shown in Equation 6, we need to calculate the standard deviation (or variance) of the test inputs in each stratum. Then, we can obtain the quantity that needs to be selected in each stratum. As described in Section 3.2, this form of allocation is the optimal allocation, which has been proven[14]. However, the original test inputs collected are unlabeled. Without labels, the accuracy variance of the test inputs in each stratum cannot be calculated. Thus, we propose two methods for estimating the accuracy variance of each stratum.

In the first method, we leverage the corresponding training data as auxiliary information to estimate the variance of each stratum. After determining the strata, we put the training data into the corresponding stratum. Then we use the variance of the training data to estimate the variance of the test data in the stratum. Moreover, in Section 6.3, we use experiments to verify that there is no significant difference between the variance of the training data and the variance of the test data in the same stratum when the training set and test set are from the same distribution. In a nutshell, the training data can be put into different strata by two stratification approaches (mentioned in Section 4.1), clustering and rule-based confindence setting, respectively.

For cluster stratification, we present the process of putting training data into different strata in Algotithm 1. Specifically, C represent a list of L clusters $[\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_L]$, where $\mathbb{C}_l$ is a set of test inputs in the l-th cluster, and Ψ_l is the cluster centroid. C is the clustering result produced by cluster stratification. Firstly, we create a new list E for stratifying training data, formed by L empty set $\mathbb{E}_l$ (Line 1). For each training input d in D, we calculate its distance from each cluster centroid Ψ_i $Dist(d, \Psi_i)$, and we put it into the stratum to which the cluster centroid closest to this training data belongs (Lines 2-5). Then, for each stratum l, we calculate the accuracy variance of the training data in it (denoted as $\hat{\sigma}_l^2$) as the variance estimation (Lines 6-9).

For rule-based stratification, we first compute the confidence of all training

```
Algorithm 1: Variance Estimation via Training Data
Input  : $C$: a list of the clusters with the cluster centroids $\Psi_i, i \in [1, \ldots, L]$
         $\mathbb{D}$: the training set
Output: $Z$: a list of the estimated accuracy variance in strata
/* Create a new list for stratifying training data                          */
1  $E \leftarrow [\mathbb{E}_1, \mathbb{E}_2, \ldots, \mathbb{E}_L]$
2  foreach $d$ in $\mathbb{D}$ do
3      $i \leftarrow \arg\min_{i \in [1,\ldots,L]} Dist(d, \Psi_i)$
4      $\mathbb{E}_i.append(d)$
5  end
6  foreach $\mathbb{E}_l$ in $E$ do
7      $\hat{\sigma}_l^2 \leftarrow$ the accuracy variance of $\mathbb{E}_l$
8      $Z.append(\hat{\sigma}_l^2)$
9  end
10 return $Z$
```

data. Then, we put each training data into the corresponding stratum according to the confidence.

However, as training data may not be accessible, as well as the scenarios in which the distributions of training data and testing data are different, we propose a second method to estimate the variance of each stratum by a step of pre-sampling, which is to randomly select a small amount of data from test inputs in each stratum. Specifically, we randomly sample h test inputs from each stratum. After labeling these h test inputs, we calculate the accuracy variance of these h test inputs as an estimate of the variance of this stratum. Next, we use these estimated variances to calculate the optimal number n_l of samples for each stratum based on Equation 6. Finally, we randomly sample the stratum with a size of $n_l - h$. As a result, the final number of samples in each stratum is still n_l, and the sample size from the original test input set is still n.

4.3. Usage of SSOA

In this section, we present the usage of our approach, and Algorithm 2 shows the high-level pseudocode of our DNN test input selection framework based on stratified random sampling with optimum allocation.

```
Algorithm 2: SSOA
  Input  : $\mathcal{M}$: the DNN model under test
           $\mathbb{S}$: the whole testing set, whose size is $N$
           $n$: the number of selected test inputs from $\mathbb{S}$
           $\mathbb{D}$: the training set (optional)
           $n\_clusters$: the number of clusters in the clustering stratification
  (optional)
           $h$: the number of pre-samples in the variance estimate (optional)
  Output: $\mathbb{T}$: a set of selected test inputs, whose size is $n$
1 $\mathbb{T} \leftarrow \emptyset$
2 Calculate the confidence of each test input in $\mathbb{S}$ with $\mathcal{M}$
3 Stratification based on Section 4.1
4 foreach $\mathbb{S}_l$ in $\mathbb{S}$ do
5     $\hat{\sigma}_l^2 \leftarrow$ Estimate the accuracy variance of $\mathbb{S}_l$ based on Section 4.2
6 end
7 foreach $\mathbb{S}_l$ in $\mathbb{S}$ do
      /* optimum allocation                                    */
8     $n_l \leftarrow n \cdot \frac{N_l \cdot \hat{\sigma}_l}{\sum_{h=1}^{L} N_h \cdot \hat{\sigma}_h}$
      /* simple random selection                               */
9     $selectedInput \leftarrow SRS(\mathbb{S}_l, n_l)$
10    $\mathbb{T}.add(selectedInput)$
11 end
12 return $\mathbb{T}$
```

The inputs include a DNN under test denoted as $\mathcal{M}$, a whole test set denoted as $\mathbb{S}$ whose size is denoted as N, a whole training set denoted as $\mathbb{D}$, and a required number representing the size of the small set of selected test inputs denoted as n. When using clustering stratification (Line 3), we can set the number of clusters $n\_clusters$ in the input. When estimating the variance using the pre-sampling method (Line 5), we need to input the number of pre-sampled test inputs in each stratum. Since both of our proposed stratification methods need to use confidence as the basis for stratification, we first calculate the confidence of all test inputs (Line 2). Next, We stratify all test inputs through a stratifying approach (e.g., rule-based stratification or cluster-based stratification), shown in Line 3.

Then, we estimate the variance of each stratum through the information from the training data or pre-sampling method (Lines 4-6). After determining the stratification and variance estimation of each stratum, we select test inputs in each stratum (Lines 7-11). We calculate the number of test inputs that should be selected in each stratum by Equation 6 (Line 8). Finally, we randomly select the specified number of test inputs in each stratum (Line 9). These selected test inputs together form the final test subset. After completing the sampling of each stratum, the estimation of the original whole test set can be calculated through the sampled test inputs subset.

4.4. Analysis on Unbiasedness of SSOA

In this section, we analyze why SSOA is unbiased. Traditional stratified sampling has been shown to be unbiased [14]. The main difference between SSOA and original stratified sampling is that the stratum variance used to determine the number of samples in each stratum (i.e., Equation 6) is not the real stratum variance. As introduced in Section 4.2, we provide two estimation approaches: by training data and by pre-sampling.

Next, we analyze the unbiasedness of these two approaches respectively. We only use the training set to estimate the variance under the condition that the training set and test set are independent and identically distributed [37] (a.k.a. i.i.d., which is a common assumption in the machine learning). Due to the i.i.d. property, variances of the training set and test set are equivalent, and so no bias is introduced. When the i.i.d. condition is not established, we use pre-sampling to estimate the variance. Please note that, some statistical methods can be used to analyze the data feature distribution of the training data and the test data to determine whether the i.i.d. condition is established in advance. To avoid bias in pre-sampling, we apply SRS to select test data, where the estimation based on SRS is unbiased. After the sampling number of each stratum is determined by Equation 6, we still execute SRS to obtain the specified number of test inputs in each stratum, and as a result, the overall process is also unbiased.

5. Implementation

5.1. Datasets and DNN Models

Table 1: DNN models and datasets

ID	**Model**	**Dataset**	**#Classes**	**#Trains**	**#Tests**	**Accuracy(%)**
S1	CN-12	CIFAR-10	10	50,000	10,000	80.64
S2	VGG-16					93.75
S3	VGG-16	CIFAR-100	100	50,000	10,000	69.44
S4	VGG-19	ImageNet	1,000	1,200,000	5,000	71.46
S5	ResNet-50					75.14

In our experiments, we evaluate our approach and baseline approaches with four model architectures and three publicly-available datasets. The details of the datasets and models are shown in Table 1. For datasets, CIFAR-10 [38] is a 10-class ubiquitous object classification dataset. CIFAR-100 [38] is a 100-class finer-grained object classification dataset. ImageNet [39] is a 1000-class large-scale realistic object classification dataset. Since the original training set is too large, in our experiments, we sample 50 images for each class in the original training set when estimating variance with the training data. Furthermore, since the test set of ImageNet has no labels, we randomly sample 5 images from each class in the validation set, forming a test set of 5,000 images in total. For models, CN-12 [40] represents a twelve-layer convolutional neural network. It has six convolutional layers with a kernel size 3×3, three max-pooling layers with 2×2 filters and stride 2, as well as three dropout layers with a dropout rate of 0.5. The structure and training code of the model is the same as those of [10]. We train the models ourselves with the code given by Guerriero et al. [41]. We use the pre-trained models VGG-19 and ResNet-50 provided by Keras [42].

Note that a recent study, DeepEST [10] also uses MNIST dataset, together with CN-5 model and LeNet-5 model in the experiment. We do not consider them because these scenarios cannot show the effectiveness of different test input selection methods. As reported in [10], comparing DeepEST with SRS under the conditions that the model is CN-5 and the sampling size budget is 200, the MSE of SRS and DeepEST are 3.6×10^{-5} and 1.5×10^{-5}, respectively. By considering the ground truth test accuracy is 0.993, the benefit from the better test input selection method is very small. As a result, we consider more complex datasets, CIFAR and ImageNet, as well as more well-known models, VGG and ResNet, in our experiments.

5.2. Baseline Approaches

Regarding effectiveness, we compare our approach with four sampling-based test selection methods SRS, CES[7], CSS[7], and DeepEST[10] as introduced in Section 2.1.1. We chose these methods as compared methods in our experiments because they are all designed for the same task. More importantly, sampling techniques are used in these methods. Although neither CES nor DeepEST have a sampling allocation strategy, there are fewer existing solutions to the test input selection problem. More specifically, for DeepEST, which has four variations, we implement $DeepEST_C$ as our baseline because it has the best performance among all variants [10]. Regarding stability, we compare SSOA with the heuristic-based selection method PACE [9].

5.3. Parameter Settings

For stratification methods, we adopt the implementations of the K-Means cluster algorithm provided by sklearn [43]. The parameters in K-Means are set as *n_clusters* to be 3 for all the subjects in our experiments. We also investigate the impact of the *n_clusters* value in our selection framework. We consider the *n_clusters* value to be 2, 3, 4, 5, and 6, respectively (presented in Section 6.6). In the rule-based stratification method, we first rank all test inputs in descending order of confidence. Then, we divide the first 80% with the highest confidence in the test set into the first stratum. The middle 10% is the second stratum, and the last 10% is the third stratum. This stratification setup is consistent with CSS [7].

The number of pre-selection h in each stratum is a hyperparameter of the variance estimation methods. In a nutshell, the setting of h can influence the performance of variance estimation. If h is too small, it can lead to imprecise variance estimation; however, if h is too high, it indicates that the number of test inputs used in optimum allocation is less, which indicates SSOA degrades to random sampling. Thus, we have to choose a proper setting of h due to the actual sampling budget. In our experiments, as the minimum sample budget is 50, we set h as 10 to achieve both an accurate variance estimation and enough space to apply optimum allocation.

5.4. Hardware and Runtime Environments

We implement our methods using Python and train all DNN models based on Keras 2.7.0 with TensorFlow 2.4.1. Our experiments are conducted on the Intel Xeon Silver 4214 machine with 256GB RAM, Ubuntu 18.04.4. All the code and data in our work are available at the project homepage [15].

6. Evaluation

In this section, we address the following research questions:

- RQ1: Compared with baselines, how does SSOA perform?
- RQ2: Does optimum allocation contribute to SSOA?
- RQ3: Is the variance of each stratum estimated by the training data and pre-sampling consistent with the variance of the original test data in each stratum?
- RQ4: Does SSOA perform more stability than the heuristic-based selection methods?
- RQ5: Is SSOA still effective when the test and training data are from different distributions?

- RQ6: How does $n\_clusters$ influence the performance?

6.1. Effectiveness of SSOA (RQ1)

Setup. As we propose two methods for stratification (Section 4.1) and two methods for variance estimation (Section 4.2), in subsequent experiments, SSOA framework specifically has four implementations **SSOA-R-T, SSOA-C-T, SSOA-R-P, and SSOA-C-P**, where **R** represents the **R**ule-based stratification and **C** represents the **C**luster stratification; meanwhile, **T** represents the variance estimation via **T**rainning set, and **P** represents the variance estimation via **P**re-sampling.

To answer RQ1, we utilize the Mean Squared Errors (MSE) to evaluate the effectiveness of the test selection methods as in the previous studies [7, 10, 9]. The smaller MSE, the more precise the model accuracy is estimated, which means that the corresponding selection method is more effective. Due to the randomness in the sampling-based methods, we repeat all experiments K times under different selection budget n. As a result, MSE can be calculated as:

$$MSE(\hat{acc}) = \frac{1}{K}\sum_{i=1}^{K}(\hat{acc}_i - acc)^2 \tag{7}$$

where $\hat{acc}_i$ is the estimated accuracy at the i-th repetition and acc is the actual accuracy of the original large-scale test set on the corresponding model. In the experiments, as [7], we set K as 50. In addition, since the value of MSE is too small to be observed easily, still the same as [7], we normalize the results as the square root of MSE. Specifically, we first use each sampling-based test selection method to select a subset of test inputs, under the size budget n ranging from 50 to 200 with an interval of 10. Then, we use the selected test samples to calculate the overall accuracy estimation. For the same selection number n, we obtain K estimated values $\hat{acc}$. Finally, we use Equation 7 to evaluate the effectiveness of the test input selection method.

Results. In order to illustrate the performance clearly, we separate our four SSOA methods into two groups to compare with baselines based on the stratum variance estimation. The resutls from Table 2 and the first row of Figure 2 are related to stratum variance estimation by training data. Meanwhile, the results from Table 3 and the second row of Figure 2 are related to stratum variance estimation by pre-sampling.

Figure 2 demonstrates the effectiveness of methods at selected sample budgets from 50 to 200. This figure shows that SSOA-R-T and SSOA-C-T outperform baseline methods overall.

To further analyze the results, we also sum up average improvements in Table 2. Given a budget, a baseline method and an SSOA method, the relative improvement is $(\sqrt{MSE}_{baseline} - \sqrt{MSE}_{SSOA})/\sqrt{MSE}_{baseline}$, where $\sqrt{MSE}_{baseline}$ and $\sqrt{MSE}_{SSOA}$ are the corresponding results from Figure 2. The reported average improvement is the average over different budgets. We also detect the significant difference between our selection method and other baselines. A one-sided sample Wilcoxon signed-rank test [44] is applied to check whether our selection framework outperforms other compared methods. In Table 2, the bold values indicate that our selection framework outperforms the compared methods with the significance level as 0.05.

Table 2: Average improvements across budgets from 50 to 200 of SSOA-R-T and SSOA-C-T over baselines (%)

App.	**SSOA-R-T**					**SSOA-C-T**				
	S1	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
SRS	**16.10**	**45.91**	**6.47**	**30.63**	**28.63**	**13.87**	**17.32**	**13.31**	**34.20**	**30.50**
CES	**18.56**	**57.49**	**34.79**	**31.84**	**32.79**	**16.87**	**34.59**	**40.36**	**34.88**	**34.91**
CSS	**38.73**	**31.69**	**38.96**	**38.62**	**51.55**	**37.69**	-2.91	**43.06**	**41.93**	**52.97**
DeepEST	0.18	**57.79**	**34.90**	**75.21**	**73.93**	-2.38	**35.73**	**39.69**	**76.44**	**74.30**

* S1:{Model: CN-12, Dataset: CIFAR-10}; S2:{Model: VGG-16, Dataset: CIFAR-10}; S3:{Model: VGG-16, Dataset: CIFAR-100}; S4:{Model: VGG-19, Dataset: ImageNet}; S5:{Model: ResNet-50, Dataset: ImageNet}.

For S1, we observe that DeepEST and SSOAs have comparable performance, and much better than the other baselines. According to Table 4, we understand even passed the hypothesis test, the stratum test accuracy variance estimations of S1 are not precise by training data. Therefore, the primary factor contributing to the absence of advantages in S1 for SSOAs, when compared to DeepEST, is the inadequate accuracy of variance estimation. For S2, SSOA-R-T outperforms

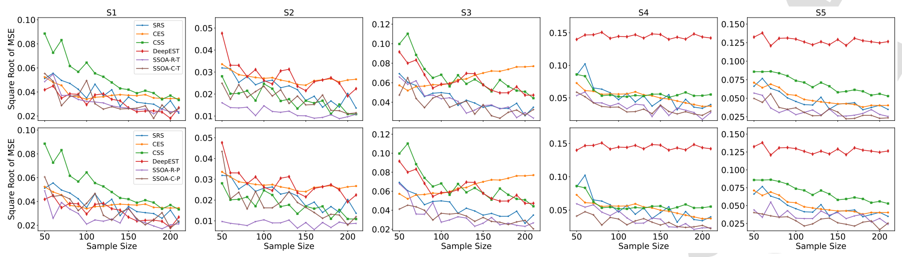

Figure 2: Estimation performance under budgets from 50 to 200 by comparing SSOA-R-T, SSOA-C-T, SSOA-R-P and SSOA-C-P with baselines (the first row of diagrams: the results of SSOA-R-T and SSOA-C-T; the second row of diagrams: the results of SSOA-R-P and SSOA-C-P).

the all compared methods. Meanwhile, although SSOA-C-T is much better than the other baselines, it is only comparable with CSS. The reason for such comparable result is the clustering method does not achieve a high quality of stratification on S2, because the average improvement of SSOA-R-T over CSS is quite huge as shown in Table 2. For S3, obviously SSOA-R-T, SSOA-C-T, as well as SRS perform better than the other three baselines, as shown in Figure 2. The experimental results from [10] are the same, i.e., SRS is better than DeepEST in the same scenario. Furthermore, as illustrated in Figure 2, SSOA-C-T outperforms SRS in 76.47% (13 out of 17) sampling size budgets in the range from 50 to 200. For S4 and S5, SSOA-R-T and SSOA-C-T perform better than the other baselines. This shows that SSOAs are also effective on complex datasets and models. Furthermore, as shown in Figure 2 and Table 3, the trend of SSOA-R-P is consistent with that of SSOA-R-T, and the trend of SSOA-C-P is similar to that of SSOA-C-T.

In addition, by comparing stratification methods based on Tables 2 and 3, the clustering stratification method (C) performs worse than the rule-based setting (R) in S1 and S2, but better in S3, S4 and S5. This indicates that even though the same allocation method and the same variance estimation method are used, the stratification method affects the final selection. Furthermore, we observe that the SSOA is more effective in the cases: 1) The variance differences among strata are large. For example, the performance of SSOAs with rule-based

Table 3: Average improvements across budgets from 50 to 200 of SSOA-R-P and SSOA-C-P over baselines (%)

App.	SSOA-R-P					SSOA-C-P				
	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
SRS	**28.71**	**58.20**	**22.51**	**29.20**	**21.62**	**11.06**	**18.10**	**22.04**	**36.46**	**38.00**
CES	**31.04**	**67.97**	**45.49**	**30.31**	**26.06**	**13.32**	**33.98**	**47.37**	**38.16**	**41.74**
CSS	**47.78**	**48.06**	**49.13**	**36.74**	**47.12**	**35.50**	-3.15	**49.29**	**44.32**	**58.12**
DeepEST	**14.14**	**67.93**	**45.81**	**74.43**	**71.62**	-5.72	**36.76**	**46.45**	**77.71**	**77.42**

* S1:{Model: CN-12, Dataset: CIFAR-10}; S2:{Model: VGG-16, Dataset: CIFAR-10}; S3:{Model: VGG-16, Dataset: CIFAR-100}; S4:{Model: VGG-19, Dataset: ImageNet}; S5:{Model: ResNet-50, Dataset: ImageNet}.

stratification on S2 is better than those of S1 and S3, and the improvements relative to baselines are more than 30%. The reason is that after rule-based stratification on S2, the ground truth test variances of the three strata are 0.49, 0.34, and 0.09, respectively. It can be seen that the variances in strata are significantly different. 2) When the sample sizes in each stratum are appropriate (not too many or too few). For example, the performance of SSOAs with clustering stratification on S3 is better than that of S1 and S2. Comparing the stratum sample sizes of S3 and S2, the number of test inputs in each stratum in S3 is 7261, 1531, and 1208, respectively, meanwhile the number of test inputs in each stratum in S2 is 9250, 438, and 312, respectively. The first stratum of S2 occupies over 90% samples in the test set.

Answer to RQ1: SSOA outperforms all the baseline approaches in most cases (i.e., 93.75% cases). Therefore, SSOA is a more reliable choice for DNN test input selection.

6.2. Effectiveness of Optimum Allocation (RQ2)

Setup. We implement the rule-based stratification same as CSS [7], as introduced in Section 5.3. Then their difference is only related to the allocation method, where CSS manually allocates the sampling ratio of each stratum, while SSOA-R uses the optimum allocation method as Equation 6 to allocate the sampling ratio of each stratum. Like RQ1, the MSE is used as the index to compare the effectiveness.

Results. Our proposed test input selection method is built on the optimum allocation of stratified sampling. We investigated the contribution of optimum allocation by comparing the effectiveness of CSS with SSOA-R-T and SSOA-R-P, since the rule-based stratification settings in SSOA-R-T and SSOA-R-P are the same as CSS, except the allocation method. That is, in CSS, the number of samples for each stratum is manually set through experience, while in SSOA-R-T/SSOA-R-P, the number of samples for each stratum is calculated by Equation 6. The comparison results are in Figure 2. We observe that both SSOA-R-T and SSOA-R-P outperform CSS on all sample size budgets in all five scenarios.

Answer to RQ2: The optimum allocation contributes to the selection framework. Therefore, SSOA can both save the manual effort in tuning hyperparameters used in CSS and gain the optimal sampling results when given the specific stratification.

6.3. Feasibility of Variance Estimation (RQ3)

Setup. To answer RQ3, for each experiment scenario, we separately record the estimated variance of training data, the estimated variance of pre-sampling and the actual variance of test data in each stratum. A paired sample Wilcoxon signed-rank test [44] is applied to check whether the estimated variance is matched with the actual variance in each stratum. During the hypothesis test, we input the estimated variances and the actual variances from the test data, and then calculate the p-value. Finally, we judge whether to accept the hypothesis, that is, whether there is no significant difference between the variance estimated by our methods and the actual variance of the test data in each stratum. In our experiments, the significance level of the test is set as 0.05. In addition, we have conducted experiments on both two stratification methods through the above steps.

Results. From Table 4, we can observe that it is feasible to estimate the variance by the training data and pre-sampling data. More specifically, all the p-value in Table 4 are greater than 0.05 and some of them are even 1. Thus, we can conclude that there is no significant difference between the estimated variance (i.e., by the training data and the pre-sampling data) and the variance of the test data, which demonstrates the feasibility of leveraging estimated variance to represent the actual variance of test data.

Table 4: Significant difference test for estimated variance and actual variance in the same stratum

Strat.	**Training**					**Pre-sampling**				
	S1	**S2**	**S3**	**S4**	**S5**	**S1**	**S2**	**S3**	**S4**	**S5**
Rule-based	0.423	0.181	0.789	1.0	0.789	0.789	1.0	0.423	0.789	0.789
Cluster	0.423	0.181	0.789	0.423	0.423	0.423	1.0	0.423	0.181	1.0

[*] S1:{Model: CN-12, Dataset: CIFAR-10}; S2:{Model: VGG-16, Dataset: CIFAR-10}; S3:{Model: VGG-16, Dataset: CIFAR-100}; S4:{Model: VGG-19, Dataset: ImageNet}; S5:{Model: ResNet-50, Dataset: ImageNet}.

Answer to RQ3: Leveraging the estimated variance to represent the actual variance is feasible since there is no significant difference between them.

6.4. Stability Comparison (RQ4)

Setup. To answer RQ4, we expand the number of samples to 4000 in the five experimental scenarios. For SSOA, we use the implementation of SSOA-R-T to illustrate the stability. The performance of SSOA-R-T with a size budget of up to 4000 is compared with a typical heuristic-based method, PACE. We also use SRS as a reference. The same as RQ1, the MSE is used as the index to compare the effectiveness.

Results. It is also interesting to investigate the stability of the DNN test input selection methods, including heuristic-based as introduced in Section 3. We compare the stability of SRS, SSOA-R-T and one heuristic-based method, PACE, by sampling from 50 to 4000, whose results are shown in Figure 3. For S1, we observe that the MSE of the accuracy estimated by PACE increases when the number of samples is greater than 2500. For S3, we also observe that the

MSE of the accuracy estimated by PACE increases when the number of samples exceeds 3000. Although the heuristic-based sampling method PACE can achieve extremely small estimation errors at certain budget, PACE's performance is not stable, and it is nontrivial, if not impossible, to estimate which sample size the PACE performs well. However, sampling-based selection methods present a steady downward trend with increasing budget sizes.

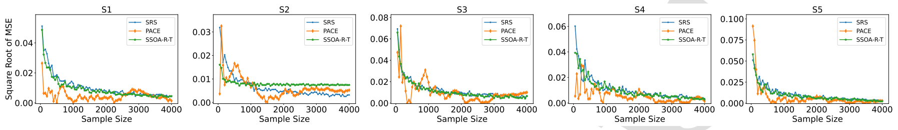

Figure 3: Comparison of heuristic-based selection and sampling-based selection.

Answer to RQ4: Our method is more stable than the heuristic-based selection methods. In practice, for SSOA, to fully utilize the sample size budget can obtain the optimal estimation. However, for heuristic-based methods, like PACE, to determine the selected sample size within the budget for the best performance is critical but tricky.

6.5. Effectiveness on Different Distributions (RQ5)

Setup. In some real-world scenarios, the test and training sets are not collected under the identical and independent condition. We follow the existing work [9] to construct test and training sets from different distributions by introducing adversarial test inputs. In the experiments, we use three methods for adversarial sample generation, which are C&W (Carlini&Wagner) [45], FGSM (Fast Gradient Sign Method) [46] and BIM (Basic Iterative Methods) [47]. These three adversarial sample generation methods have been widely used in the existing works [29, 22]. We keep 5000 test inputs from the original test set and apply the adversarial example generation method on the rest 5000 test inputs, which is to construct a new combined test set for S2 and S3, respectively. Since test and training sets are from different distributions, we cannot use the training data to estimate the variance of each stratum. As a result, we compare two

SSOA implementations based on pre-sampling variance estimation, SSOA-R-P and SSOA-C-P, with baselines on those new combined test sets. In addition, the same as RQ1, the MSE is used as the index to compare the effectiveness.

Results. From Figure 4, we observe that the overall MSE is higher than that of Figure 2. This shows that it is more difficult to precisely estimate the accuracy of the test set whose distribution is different from training because the model is not designed for such test scenario. Furthermore, Figure 4 also shows that compared with the baselines, SSOA-R-P and SSOA-C-P perform better. This demonstrates that SSOA is still effective when the test and training data are from different distributions. In particular, SSOA-C-P performs better than SSOA-R-P. The reason is that with adversarial samples, the distribution of confidence in the new combined test set differs from the original test set. As a result, the previously manually divided stratification [7] is not the most reasonable. Therefore, when the test and training sets are from different distributions, SSOA-C-P is the proper implementation.

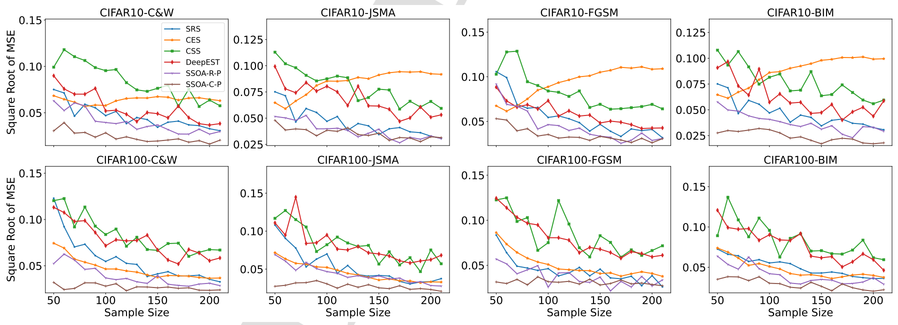

Figure 4: Square root of MSE under the combined test set.

Answer to RQ5: SSOA is still effective when the test and training data are from different distributions. Furthermore, SSOA-C-P is the proper implementation.

6.6. Imapct of Different $n\_clusters$ Value on SSOA (RQ6)

Setup. We design the experiment protocol as follows. We first run SSOA-C-T and SSOA-C-P with different $n\_clusters$ values, ranging from 2 to 6 with the interval of 1, to select test inputs under different sizes, respectively. We then calculated the square root of MSE for SSOA-C-T and SSOA-C-P with each $n\_clusters$ value. Finally, we show the selection results of different $n\_clusters$ values.

Results. We investigate the impact of different $n\_clusters$ values, whose results are shown in Figure 5. The line graph shows the square root of MSE at each sampling number for different $n\_clusters$ settings in different scenarios. Here, we study five different $n\_clusters$. For SSOA-C-T, Figure 5 shows that different $n\_clusters$ have little effect on the results, which indicates that SSOA-C-T is not sensitive to the setting of $n\_clusters$. The CSS [7] manually divides the test set into three strata. Therefore, for comparison with CSS [7], we use a consistent stratifying setting as $n\_clusters = 3$. For SSOA-C-P, for different scenarios, the best and worst $n\_clusters$ settings are different, probably because SSOA-C-P has more uncertainty in pre-sampling than SSOA-C-T. We observe that a smaller $n\_clusters$ usually indicates better performance than the larger one. This is because the larger the number of clusters is set, the larger the number of samples for pre-sampling is required. With the same size budget, there is not much space for sampling after estimating the variance. Meanwhile, we observe that the $n\_clusters$ =3 performs better among all the $n\_clusters$ values.

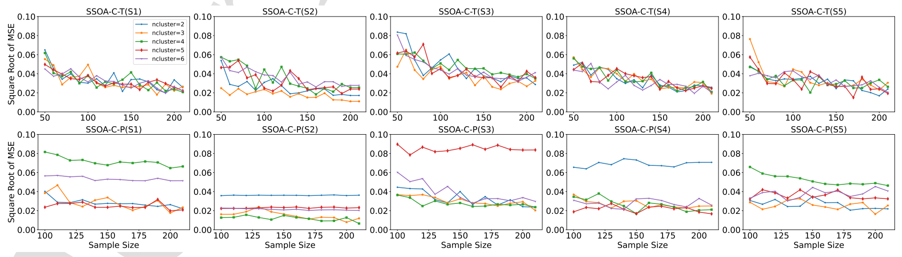

Figure 5: Square root of MSE under $n\_clusters$ from 2 to 6.

Answer to RQ6: As shown in Figure 5, setting $n\_clusters$ as 3 is a reasonable choice, and achieves the best effectiveness among all the studied $n\_clusters$ values in general.

7. Discussion

7.1. Scalability of SSOA

To check the capability of SSOA, Figure 6 illustrates the budget size up to 1000. These experiments are designed for some complex scenarios (e.g., extremely large-scale dataset to label) and sufficient budgets. The results show that due to the law of large numbers, the square root of MSE of all four SSOA methods are steadily decreasing while the budget increases. It further proves the stable effectiveness of SSOA when facing higher budgets.

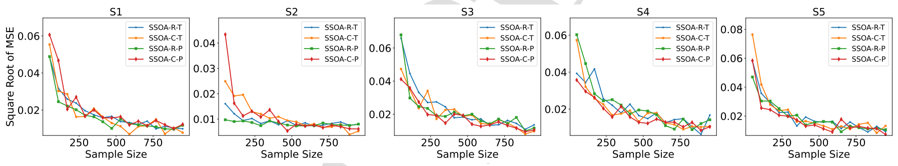

Figure 6: Square root of MSE under budgets from 50 to 1000.

7.2. Application Domain of SSOA

Table 5: Effectiveness comparison among SSOA-C-P, SRS, and CES in terms of the average MSE values (%) by taking two DNN regression models (i.e., Dave-orig and Dave-drop) with the testing set Driving as examples

Model.	**App.**	**Number of selected test inputs**													
		50	**60**	**70**	**80**	**90**	**100**	**110**	**120**	**130**	**140**	**150**	**160**	**170**	**180**
Dave-orig	SRS	2.894	3.215	2.812	2.136	3.941	2.011	0.834	0.822	1.440	0.820	1.384	1.746	1.563	1.353
	CES	1.599	**1.344**	1.220	1.195	1.101	1.013	1.066	1.082	1.022	0.975	1.011	1.022	0.949	0.879
	SSOA-C-P	**1.066**	1.656	**0.864**	**1.140**	**0.672**	**0.607**	**0.702**	**0.749**	**0.852**	**0.663**	**0.756**	**0.357**	**0.726**	**0.653**
Dave-drop	SRS	1.647	1.469	1.398	1.293	1.143	1.083	1.012	1.027	0.991	0.999	0.991	0.941	0.831	0.799
	CES	1.572	1.511	1.311	1.181	1.077	1.037	1.011	0.973	0.869	0.852	0.795	0.747	0.729	0.702
	SSOA-C-P	**0.323**	**1.131**	**0.860**	**0.800**	**0.826**	**0.769**	**0.551**	**0.627**	**0.811**	**0.503**	**0.751**	**0.113**	**0.258**	**0.343**

In our study, we evaluated the performance of SSOA in the domain of normal image classification. It is actually also interesting to investigate the effectiveness of SSOA in regression tasks. Here, we conducted an experiment by taking two DNN regression models (i.e., Dave-orig and Dave-drop [48]) with the testing set Driving to investigate it, whose results are shown in Table 5. The input of the two models Dave-orig and Dave-drop is the image captured by the camera on the driving vehicle, and the output is the predicted steering wheel angle. The Driving dataset [49] is the Udacity self- driving car challenge dataset containing 101,396 training and 5,614 testing examples. Dave-orig and Dave-drop are trained on the Driving datasets following the setup of [7]. Since the regression model does not have a confidence output, we cannot apply confidence rules for stratification, but we can use clustering stratification. Therefore, we use SSOA-C-P for test input selection on the regression task. Please note that, what we proposed is the test input selection framework. We can choose the applicable variant according to the usage scenario, and the previous experiment (RQ1) has also proved that all four variants are effective. Since CSS and DeepEST were not applied to the regression model in [7, 10], we only show the results of SRS, CES, and SSOA-C-P. We repeated the experiment 50 times for each method on each sample size. In the table, we highlighted the most effective approach (i.e the lowest MSE) for each number of selected test inputs. From Table 5, for all the 28 cases (2 subjects * 14 settings for the number of selected test inputs), SSOA performs the best in 96.43% (27 out of 28) cases. The results demonstrate that, regardless of classification task or regression task, SSOA is able to outperform all the compared approaches.

7.3. Future Works of SSOA

Our experiments have demonstrated that SSOA achieves great effectiveness for estimating the accuracy of the whole testing set by selecting a small number of test inputs. There are some possible directions to further improve SSOA.

First, in our study we evaluated the performance of SSOA in the input form of normal image. In the furture, we will try to extend SSOA to multimodal input scenarios, for example, speech, text, etc.

Second, the primary objective of SSOA is to accurately estimate the overall accuracy of the entire testing set, which constitutes a single-objective problem. However, in practical applications, there exist several additional objectives that need to be fulfilled, such as maintaining the same coverage as the entire testing set. In the future, we plan to extend SSOA to solve the problem of test input selection with multiple objectives.

Third, in SSOA-C the K-Means algorithm is adopted to perform clustering due to several reasons presented in Section 4.1. However, there are other clustering algorithms that meet the requirements mentioned in Section 4.1, and even some that do not require a pre-set number of clusters, such as Affinity propagation clustering. In the future, we may explore other clustering algoritms.

Fourth, SSOA involvers several parameters, such as the clustering parameters and the the number of pre-sampling h. For different subjects, the best parameter values may be different, and thus in the future we plan to propose to set the parameter value dynamically by considering the characteristics of the used subject.

7.4. Threats to Validity

The *internal* threat to validity mainly lies in the implementations of SSOA and baselines. To reduce this threat, we adopted implementations of baselines released by the authors, implemented SSOA based on some existing libraries (presented in Section 5), and carefully checked the code of our approach SSOA and experimental scripts.

The *external* threats to validity mainly lie in the DNN models and the test sets. For DNN models, we adopted the models trained based on popular datasets (including CIFAR-10 and CIFAR-100). To reduce the threats from DNN models and test sets, we considered using models and datasets widely-used in existing research [10]. Since SSOA can work with all kinds of inputs (e.g., text, voice), we consider extending SSOA to more domains in our future work.

The *construct* threats to validity mainly lie in the parameters in SSOA and the randomness in our study. SSOA involvers several parameters, such as the clustering parameters $n\_cluster$ and the the number of pre-sampling h. Regarding the clustering parameters, we set the number of clusters to be consistent with [7] and used the same parameters for all the scenarios. Meanwhile, we conducted a study to evaluate the impact of different clustering parameter $n\_cluster$, and the results demonstrate that the default setting (i.e., 3) is a good choice (presented in Section 6.6). Moreover, K-means is a commonly used clustering algorithm. Researchers in the field of data mining have also proposed some methods for determining the optimal number of clusters, which can also be applied to our test input selection framework. For example, the classical knee method [50], search-based method, contour coefficient method, etc [51]. Regarding the number of pre-sampling h, we set this parameter based on the test budget in our experiments and used the same setting for all the subjects. Moreover, we provide guidance on parameter h settings in the second paragraph of Section 5.3, i.e. how to adjust parameter h settings when the test budget changes. We presented the specific settings of the parameters in Section 5. To reduce the threat of randomness involved in our study (including SSOA and four baselines), we repeated each of them 50 times and calculated the effectiveness by Equation 7.

8. Conclusion

We integrate sampling theory into the task of deep learning test input selection and propose an unbiased test input selection framework based on stratified random sampling and optimum allocation, called SSOA. Experimental results demonstrate that SSOA is more effective than the state-of-art sampling-based selection methods. Besides, the experiment results also show that SSOA provides a reliable estimation of the model performance. These observations indicate that to make fully use of the budgets by SSOA can obtain an effective and reliable estimation of the model performance in practice.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (62202329).

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig, Achieving human parity in conversational speech recognition, arXiv preprint arXiv:1610.05256 (2016).

[3] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[4] M. Zhang, Y. Zhang, L. Zhang, C. Liu, S. Khurshid, Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018, ACM, 2018, pp. 132–142.

[5] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, Z. Chen, Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks, in: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2020, pp. 177–188.

[6] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, W. Zhang, Prioritizing test inputs for deep neural networks via mutation analysis, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 397–409.

[7] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, J. Lü, Boosting operational dnn testing efficiency through conditioning, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 499–509.

[8] J. Zhou, F. Li, J. Dong, H. Zhang, D. Hao, Cost-effective testing of a deep learning model through input reduction, in: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2020, pp. 289–300.

[9] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, M. Yan, Practical accuracy estimation for efficient deep neural network testing, ACM Transactions on Software Engineering and Methodology (TOSEM) 29 (4) (2020) 1–35.

[10] A. Guerriero, R. Pietrantuono, S. Russo, Operation is the hardest teacher: estimating dnn accuracy looking for mispredictions, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 348–358.

[11] F. J. Massey Jr, The kolmogorov-smirnov test for goodness of fit, Journal of the American statistical Association 46 (253) (1951) 68–78.

[12] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1322–1333.

[13] W. Deng, L. Zheng, Are labels always necessary for classifier accuracy evaluation?, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 15069–15078.

[14] S. K. Thompson, Sampling, Third Edition, John Wiley & Sons, Inc., 2012.

[15] Homepage, `https://github.com/AnonymousProjs26/AnonymousRep02` (Accessed: 2022).

[16] R. Pietrantuono, S. Russo, On adaptive sampling-based testing for software reliability assessment, in: ISSRE, IEEE Computer Society, 2016, pp. 1–11.

[17] M. H. Hansen, W. N. Hurwitz, On the theory of sampling from finite populations, The Annals of Mathematical Statistics 14 (4) (1943) 333–362.

[18] L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering, Journal of Open Source Software 2 (11) (2017) 205.

[19] B. Kim, R. Khanna, O. O. Koyejo, Examples are not enough, learn to criticize! criticism for interpretability, Advances in neural information processing systems 29 (2016).

[20] T. Y. Chen, F.-C. Kuo, R. G. Merkel, T. Tse, Adaptive random testing: The art of test case diversity, Journal of Systems and Software 83 (1) (2010) 60–66.

[21] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, in: proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 1–18.

[22] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, et al., Deepgauge: Multi-granularity testing criteria for deep learning systems, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 120–131.

[23] M. J. Harrold, R. Gupta, M. L. Soffa, A methodology for controlling the size of a test suite, ACM Transactions on Software Engineering and Methodology (TOSEM) 2 (3) (1993) 270–285.

[24] J. Chen, M. Yan, Z. Wang, Y. Kang, Z. Wu, Deep neural network test coverage: How far are we?, arXiv preprint arXiv:2010.04946 (2020).

[25] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, B. Xu, Adaptive test selection for deep neural networks, in: 44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022, ACM, 2022, pp. 73–85.

[26] Z. Liu, Y. Feng, Y. Yin, Z. Chen, Deepstate: Selecting test suites to enhance the robustness of recurrent neural networks, in: 44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022, ACM, 2022, pp. 598–609.

[27] C. Zhao, Y. Mu, X. Chen, J. Zhao, X. Ju, G. Wang, Can test input selection methods for deep neural network guarantee test diversity? A large-scale empirical study, Inf. Softw. Technol. 150 (2022) 106982.

[28] T. Byun, V. Sharma, A. Vijayakumar, S. Rayadurgam, D. D. Cofer, Input prioritization for testing neural networks, in: IEEE International Conference On Artificial Intelligence Testing, IEEE, 2019, pp. 63–70.

[29] J. Kim, R. Feldt, S. Yoo, Guiding deep learning system testing using surprise adequacy, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, 2019, pp. 1039–1049.

[30] Q. Hu, Y. Guo, M. Cordy, X. Xie, L. Ma, M. Papadakis, Y. L. Traon, An empirical study on data distribution-aware test selection for deep learning enhancement, ACM Trans. Softw. Eng. Methodol. 31 (4) (2022) 78:1–78:30.

[31] W. Shen, Y. Li, L. Chen, Y. Han, Y. Zhou, B. Xu, Multiple-boundary clustering and prioritization to promote neural network retraining, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020, pp. 410–422.

[32] Y. Li, M. Li, Q. Lai, Y. Liu, Q. Xu, Testrank: Bringing order into unlabeled test instances for deep learning tasks, Advances in Neural Information Processing Systems 34 (2021) 20874–20886.

[33] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, B. Xu, Adaptive test selection for deep neural networks, in: Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 73–85.

[34] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (5786) (2006) 504–507.

[35] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[36] J. A. Hartigan, M. A. Wong, Algorithm as 136: A k-means clustering algorithm, Journal of the royal statistical society. series c (applied statistics) 28 (1) (1979) 100–108.

[37] Independent and identically distributed random variables, https://en.wikipedia.org/wiki/Independent_and_identically_distributed_random_variables (Accessed: 2022).

[38] Cifar, http://www.cs.toronto.edu/~kriz/cifar.html (Accessed: 2022).

[39] Imagenet, https://www.image-net.org/. (Accessed: 2022).

[40] Github, https://github.com/coinse/sadl. (Accessed: 2022).

[41] Github, https://github.com/dessertlab/DeepEST (Accessed: 2022).

[42] Keras, https://keras.io/ (Accessed: 2022).

[43] K-means, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html (Accessed: 2022).

[44] F. Wilcoxon, Individual comparisons by ranking methods, in: Breakthroughs in statistics, Springer, 1992, pp. 196–202.

[45] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 ieee symposium on security and privacy (sp), IEEE, 2017, pp. 39–57.

[46] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: 3rd International Conference on Learning Representations, 2015.

[47] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: 5th International Conference on Learning Representations, 2017.

[48] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316 (2016).

[49] Driving, `https://udacity.com/self-driving-car` (Accessed: 2022).

[50] S. Salvador, P. Chan, Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms, in: 16th IEEE international conference on tools with artificial intelligence, IEEE, 2004, pp. 576–584.

[51] T. M. Kodinariya, P. R. Makwana, et al., Review on determining number of cluster in k-means clustering, International Journal 1 (6) (2013) 90–95.

Zhuo Wu: Data curation, Conceptualization, Methodology, Investigation, Visualization, Software, Writing - Original Draft

Zan Wang: Writing - Review & Editing, Methodology, Validation

Junjie Chen: Conceptualization, Methodology

Hanmo You: Writing - Review & Editing, Data curation

Ming Yan: Writing - Review & Editing, Visualization

Lanjun Wang: Conceptualization, Supervision, Methodology, Project administration, Writing - Review & Editing

Declaration of interests

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

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: