DeepGD: A Multi-Objective Black-Box Test Selection Approach for Deep Neural Networks

ZOHREH AGHABABAEYAN, University of Ottawa, Canada
MANEL ABDELLATIF, École de Technologie Supérieure, Canada
MAHBOUBEH DADKHAH, University of Ottawa, Canada
LIONEL BRIAND, University of Ottawa, Canada and Lero SFI Centre on Software Research and University of Limerick, Ireland

Deep neural networks (DNNs) are widely used in various application domains such as image processing, speech recognition, and natural language processing. However, testing DNN models may be challenging due to the complexity and size of their input domain. Particularly, testing DNN models often requires generating or exploring large unlabeled datasets. In practice, DNN test oracles, which identify the correct outputs for inputs, often require expensive manual effort to label test data, possibly involving multiple experts to ensure labeling correctness. In this paper, we propose *DeepGD*, a black-box multi-objective test selection approach for DNN models. It reduces the cost of labeling by prioritizing the selection of test inputs with high fault-revealing power from large unlabeled datasets. *DeepGD* not only selects test inputs with high uncertainty scores to trigger as many mispredicted inputs as possible but also maximizes the probability of revealing distinct faults in the DNN model by selecting diverse mispredicted inputs. The experimental results conducted on four widely used datasets and five DNN models show that in terms of fault-revealing ability: (1) White-box, coverage-based approaches fare poorly, (2) *DeepGD* outperforms existing black-box test selection approaches in terms of fault detection, and (3) *DeepGD* also leads to better guidance for DNN model retraining when using selected inputs to augment the training set.

CCS Concepts: • **Computing methodologies → Machine learning**; *Computer vision*; • **Software and its engineering → Software verification and validation**.

Additional Key Words and Phrases: Deep Neural Network, Test Case Selection, DNN Fault Detection, Multi-Objective Optimization, Deep Learning Model Evaluation, Uncertainty Metrics, Diversity, Model Retraining Guidance

ACM Reference Format:
Zohreh Aghababaeyan, Manel Abdellatif, Mahboubeh Dadkhah, and Lionel Briand. 2024. DeepGD: A Multi-Objective Black-Box Test Selection Approach for Deep Neural Networks. 1, 1 (March 2024), 24 pages.

1 INTRODUCTION

Deep Neural Networks (DNNs) have become widely used in a variety of application areas, including image processing [1], medical diagnostics [2], and autonomous driving [3]. However, DNNs may produce unexpected or incorrect results that could lead to significant negative consequences or losses. Therefore, effective testing of such models is crucial to ensure their reliability [4]. For testing and enhancing the performance of DNN-driven applications, a significant amount of labeled data is required. In many real-world scenarios, there are instances where the availability of labeled data for testing is either limited or absent. This situation becomes particularly pertinent when the trained and tested model is deployed in different operational settings and necessitates further evaluation using operational or real data, typically large and unlabeled. The same challenge extends to the testing of third-party pre-trained DNN models for example, where access to internal model information or training data is typically unavailable. In such cases, these models are commonly subjected to further testing using end-users' data, which is, in general, massive and unlabeled. However, obtaining the correct output labels for a large amount of unlabeled data, referred to as oracle information, can be a challenging and resource-intensive task [5]. This process often requires extensive manual effort by domain experts and can be a significant challenge when the volume of unlabeled data is large and resources are limited. For instance, in

Authors' addresses: Zohreh Aghababaeyan, University of Ottawa, Ottawa, Canada, Zohreh.a@uottawa.ca; Manel Abdellatif, École de Technologie Supérieure , Montreal, Canada, Manel.abdellatif@etsmtl.ca; Mahboubeh Dadkhah, University of Ottawa, Ottawa, Canada, mdadkhah@uottawa.ca; Lionel Briand, University of Ottawa, Ottawa, Canada and Lero SFI Centre on Software Research and University of Limerick, Ireland, Lbriand@uottawa.ca.

many real-world situations, DNN testing is usually bounded by a limited testing budget, especially when testing is performed on costly simulators such as flight simulators or real hardware (e.g., self-driving cars, medical equipment). These execution environments can be expensive and require substantial computing resources, which in turn limits the number of tests that can be performed [6]. This makes it more difficult to effectively test and improve the performance of DNN models. In order to address these challenges and make DNN testing feasible and cost-effective in practice, it is essential to select a small subset of test inputs that possess a high fault-revealing power. This approach reduces the number of inputs required for DNN testing, making it more cost-effective.

Several test selection approaches for DNN models have been proposed in recent years [7–14]. Some of them are white-box since they require access to the internals of the DNN models or the training datasets. Similar to code coverage in traditional software systems [15], researchers have proposed several neuron coverage metrics in the context of DNN test selection to prioritize the selection of test inputs with high coverage scores [7–11]. Studies have shown the effectiveness of these approaches for distinguishing adversarial inputs but failed to demonstrate a positive correlation between coverage and DNN mispredictions [16–19]. Furthermore, for some coverage metrics, reaching maximum coverage can be easily achieved by selecting a few test inputs [16, 20], thus putting into question the usefulness of such white-box DNN test selection approaches. Last, the use of white-box DNN test selection approaches is hindered by the requirement for access to the internal structure of the DNN model or the training dataset. This can be a significant limitation, particularly when the model is proprietary or provided by a third party [17].

To alleviate this latter problem, black-box test selection approaches have been proposed, which do not require access to the internal information of the DNN model under test or its training set. They generally rely on DNN output uncertainty metrics to guide the selection of test inputs. It has been observed that a test input is likely to be mispredicted by a DNN if the model is uncertain about its output classification, when similar probabilities are predicted for each class [21]. An empirical experiment conducted by Ma *et al.* [13] revealed that these metrics have medium to strong correlations with mispredicted inputs, while coverage-based metrics have weak or no correlation. One issue though is that if the model is not sufficiently trained, outputs cannot be trusted, thus leading to unreliable uncertainty results and test case selection [22].

Another black-box strategy that has been successfully applied in testing traditional software systems is to maximize diversity among test cases [23–25]. However, only a few studies have investigated its effectiveness in testing DNN models [12, 17, 26]. Zhao *et al.* [26] studied state-of-the-art (SOTA) test selection approaches for DNNs and found that they do not guarantee the selection of diverse test input sets. Furthermore, in our prior work [17], we compared the fault-revealing power of several diversity metrics with coverage criteria. We found that geometric diversity (GD) [27] outperforms coverage criteria in terms of fault-revealing power and computational time. However, we did not investigate how such a metric can be used for DNN test selection.

In this paper, we propose *DeepGD* (Deep Geometric Diversity), a black-box multi-objective search-based test selection approach. It relies on the Non-dominated Sorting Genetic Algorithm (NSGA-II) [28] to select test inputs that aim at revealing a maximal number of diverse faults in DNN models within a limited testing budget. The latter is mostly determined by test input labeling effort and, for larger models, their execution time. *DeepGD* can therefore be used to select a subset of test inputs to label from a large unlabeled dataset. The strategy of *DeepGD* is twofold: (1) trigger as many mispredicted inputs as possible by selecting inputs with high uncertainty scores, and (2) maximize the probability of revealing distinct faults in the DNN model by selecting diverse mispredicted inputs.

Similar to failures in traditional software systems, many mispredictions result from the same faults (causes) in the deep learning model and are therefore redundant [17, 29, 30]. Misprediction rates[1] can therefore be misleading when assessing a test selection approach, if a large number of mispredictions stem from the same fault. This is also why, for similar reasons, traditional software testing studies typically focus on faults, not failures [31–34]. Therefore, to validate the effectiveness of *DeepGD*, we use a fault estimation approach proposed in [17], which is based on clustering mispredicted inputs based on their features. Each cluster represents a distinct fault since it contains similar inputs that are mispredicted due to the same root causes. More specifically, we present an empirical evaluation of the effectiveness of *DeepGD* for selecting test input sets with high fault-revealing power. Since our fault estimation approach is tailored to image datasets and relies on clustering mispredicted inputs according to image features, we conduct our experiments on image classification models. They include five widely-used DNN models and four image recognition datasets. The

[1]Misprediction rate = 1 - Accuracy

results are compared with nine SOTA baseline selection approaches, both white-box and black-box, that have been recently published [7, 8, 10, 12, 13, 21].

The experimental results demonstrate that *DeepGD* yields a statistically significant and consistent improvement compared to SOTA approaches in terms of its ability to reveal DNNs faults. Specifically, results show that *DeepGD* is consistently the best approach, up to 4 percentage points (pp) better than the second-best alternative and 14 pp better than the worst black-box alternative, when excluding random selection (RS) since RS showed far inferior results in general. It is important to note that the ranking of alternatives varies across datasets, models, and test set sizes. Consequently, selecting any approach other than DeepGD may end up being the worst choice and lead to significant differences in performance. Further, we also investigate the effectiveness of *DeepGD* in guiding the retraining of DNNs and demonstrate that it consistently provides better results with that respect as well. *DeepGD* is therefore the only technique we can confidently recommend.

To summarize, the key contributions of this paper are as follows:

- We propose a black-box test selection approach (*DeepGD*) for DNNs that relies on a customized multi-objective genetic search and uses both diversity and uncertainty scores to guide the search toward finding test input sets with high fault-revealing power.
- Unlike existing test selection approaches, we consider in our validation a clustering-based approach to estimate faults in DNN models since test input sets typically contain many similar mispredicted inputs caused by the same problems (faults) in the model [17, 29]. We thus obtain more meaningful evaluation results, similar to common practice in testing research.
- We conduct a large-scale empirical evaluation to validate *DeepGD* by considering five DNN models, four different datasets, and nine SOTA test selection approaches for DNNs as baselines of comparison. We show that *DeepGD* provides better guidance than baselines for (1) selecting inputs with high fault-revealing power, and (2) improving the performance of the model through retraining based on an augmented training set.
- We study the diversity of inputs selected by *DeepGD* and SOTA baselines to assess whether our approach not only leads to test inputs with higher fault-revealing capabilities but also promotes greater diversity in the selected test input sets. We thus confirm that *DeepGD* indeed selects more diverse input sets compared to baselines.
- We analyze the execution time of *DeepGD* and other baselines and show that they are not computationally expensive. Though *DeepGD* exhibits longer execution times, these remain practical and will have little impact in practice. *DeepGD*'s improved fault detection and retraining guidance more than compensate for them.

The remainder of the paper is structured as follows. Section 2 presents our test selection approach. Section 3 presents our empirical evaluation. Section 4 outlines our results. Section 5 describes the threats to the validity of our study. Sections 6 and 7 contrast our work with related work and conclude the paper, respectively.

2 APPROACH: REFORMULATION AS AN NSGA-II SEARCH PROBLEM

A central problem in testing DNNs, especially when the labeling of test data is costly, is the selection of a small set of test inputs with high fault revealing power. In this paper, we aim to support the testing of DNN models by relying on *DeepGD*, a black-box search-based test selection approach using a genetic algorithm to select small sets of test inputs with high fault-revealing power in DNN models. Intuitively, testers should select a set of diverse test inputs with high failure probabilities in order to be more likely to detect as many diverse faults as possible [35]. Due to the combined high labeling cost of test inputs and large input space, we rely on NSGA-II to select test inputs with high fault-revealing capability. Such inputs are then selected for labeling and will be used to effectively test the DNN model. We choose NSGA-II since it is widely used in the literature and showed its performance to solve many search-based test selection problems [36, 37]. We also rely on NSGA-II since it is specifically adapted to our multi-objective search problem. It tries to find solutions with diverse trade-offs between fitness functions instead of covering all fitness functions separately (which is the case of MOSA, the Many Objective Sorting Algorithm [38] for example). Specifically, the search is driven by two objectives: (1) maximizing the uncertainty score of the test inputs to trigger a maximum number of mispredictions, and (2) maximizing the diversity of the test input set to trigger diverse mispredictions caused by distinct faults. To properly translate the process into a search problem using NSGA-II, we need to define the following elements.

2.1 Individuals and Initial Population

In genetic algorithms, individuals consist of a set of elements called genes. These genes are connected and form an individual that is also called a solution. In our approach, we assign a unique identifier id_i to each input i in the test dataset where $id_{1\leq i\leq n} \in [1, 2, .., n]$ and n is the size of the test dataset. Our test selection problem has a fixed testing budget $\beta < n$ which corresponds to the total number of inputs selected from the original test dataset to test the DNN model. In our search problem, an individual corresponds to a subset of inputs of size β. Each gene forming an individual corresponds to an *id* of a test input in the test dataset. In our context, each individual contains β distinct test inputs. We use random selection to build our initial population of individuals.

2.2 Fitness Functions

The main objective of our approach is to select unlabeled inputs that possess a strong ability to reveal diverse faults. These faults manifest themselves as diverse mispredicted inputs. To generate the latter, our search process is guided by two fitness functions. The first function aims to maximize the uncertainty score of the selected inputs, as it is moderately to strongly correlated with mispredictions [12, 13, 21, 39]. By maximizing uncertainty scores, we are thus more likely to trigger a larger number of mispredicted inputs [13]. The second function aims to maximize the diversity among the selected inputs. This approach increases the probability of identifying distinct faults. By simultaneously maximizing these two fitness functions, we aim to select a diverse set of mispredicted inputs that are due to distinct faults, thereby maximizing the fault-revealing power of the selected test input set. We will describe next the two fitness functions and detail how to compute them.

2.2.1 ***Gini Score.*** We consider the *Gini* score to estimate the likelihood of a test input being mispredicted. Feng *et al.* [21] proposed this metric to measure the classification uncertainty of DNN models and therefore identify potential failing test inputs. Intuitively, a test input is likely to be misclassified by a DNN if the model is uncertain about the classification and outputs similar probabilities for each class [21]. We choose this metric since it has been widely used in the literature, showed good performance in prioritizing test inputs for DNN models, and has a medium to strong correlation to misprediction rates [12, 13, 21, 39]. It is also a black-box metric that only requires the output probabilities of DNN models as we will describe in the following. Given a test input x and a DNN model that outputs $DNN(x) =< P_{x_1}, P_{x_2}, ..., P_{x_m} >$, where $P_{x_{i\in[1,..,m]}}$ is the probability that input x belongs to class C_i and m is the total number of classes, the *Gini* score of the test input x is defined as:

$$\xi(x) = 1 - \sum_{i=1}^{m} P_{x_i}^2 \tag{1}$$

The higher the *Gini* score, the higher the DNN's uncertainty. We compute the *Gini* score of a subset $S = \{s_1, s_2, ..., s_\beta\}$ of size β by computing the average *Gini* score of all inputs in the subset as follow:

$$Gini(S) = \frac{\sum_{i=1}^{\beta} \xi(s_i)}{\beta} \tag{2}$$

2.2.2 ***Geometric Diversity.*** We consider geometric diversity, one of the widely used metrics to measure the diversity of test input sets [17, 27, 40]. In a previous study [17], several coverage and diversity metrics were investigated and results showed that the GD metric is positively correlated to DNN faults and outperforms SOTA coverage metrics in terms of fault-revealing capabilities. In other words, when the geometric diversity of a test input set increases, its fault-revealing power increases as well since the diverse test input set will cover more faults in the DNN model. Furthermore, GD is a black-box diversity metric that requires neither knowledge about the model under test nor access to the training set. Consequently, we have relied on this metric as fitness function to guide the search towards finding a diverse test input set with high fault-revealing capability.

Feature Extraction. In order for diversity to account for the content of images, we need to first extract features from each input image and then compute the diversity based on those extracted features. In our work, we use VGG-16 [41] which is widely recognized as one of the SOTA models for image feature extraction. It is a pre-trained convolutional neural network model which was already trained on ImageNet [42], an extensive dataset including over 14 million labeled images. As a result, it has learned rich feature representations for a wide range of images and datasets [41]. VGG-16 has been extensively used in various image recognition problems and has been reported to be highly accurate [17, 43–45]. Despite its simple architecture, the VGG16 model has shown comparable performance to more complex feature extraction techniques such as ResNet50 and VGG19 [44]. We therefore rely on this feature extraction technique to produce high-quality feature representations, thereby enhancing the accuracy of our diversity measures.

We extract features of each image in the test input set S using VGG-16 and generate the corresponding feature matrix $F = (f_{ij}) \in R^{n*m}$ where n is the number of input images in S, and m is the number of features. Each row of this matrix represents the feature vector of an image, and each column (F_j) represents a feature. It is worth mentioning that, in order to ensure compatibility of certain image datasets with VGG-16, specific image pre-processing techniques are applied before feature extraction. These techniques involve operations such as image resizing and color channel duplication. After extracting the feature matrix, we normalize it as a pre-processing step to eliminate the dominance effect of features with large value ranges and to make the computation of the selected diversity metrics more scalable [17]. We apply the *Min-Max normalization* per feature, and transform the maximum and minimum values of that feature to one and zero, respectively. For every feature F_j in the feature matrix F where $F_j(i)$ is the value of feature number j for i^{th} input image in S, the normalized feature F'_j is calculated as follows:

$$F'_j(i) = \frac{F_j(i) - min(F_j)}{max(F_j) - min(F_j)} \tag{3}$$

Computation of Geometric Diversity Scores. After extracting the feature vectors, we calculate the geometric diversity of the test inputs as our second fitness function with the goal of selecting as many diverse test input sets as possible. Given a dataset $\mathbb{D}$, the normalized feature matrix F' where F'_S represents feature vectors of a subset $S \subseteq \mathbb{D}$, the geometric diversity of S is defined as:

$$GD(S) = det(F'_S * {F'_S}^T) \tag{4}$$

which corresponds to the squared volume of the parallelepiped spanned by the rows of F'_S, since they correspond to vectors in the feature space. The larger the volume, the more diverse is S in the feature space. We illustrate in Figure 1 an example of the geometric diversity of two input sets, *S1* and *S2*, including two inputs each. For the sake of simplicity, each input in this example is represented by its corresponding two-dimensional feature vector. The geometric diversity of an input set corresponds, to the squared volume of the parallelepiped spanned by the feature vectors of the input set, which is here a simple plane highlighted in blue (i.e., GD(S1) and GD(S2)). The larger the volume (i.e., the geometric diversity score), the more diverse the input set, as we can see in Figure 1. Moreover, Figure 2 includes two subsets of the MNIST dataset [46], each containing four images, evaluated with their respective GD scores. Subset 1, a low-diversity test set, includes images with greater similarity, reflected by its GD score of 13.20. In contrast, Subset 2 showcases a high-diversity test set, characterized by a wider range of image variations, as indicated by its higher GD score of 18.28.

2.2.3 *Multi-Objective Search.*

We need to maximize in our search the two aforementioned fitness functions to increase the fault-revealing capability of the selected test input set. This is therefore a multi-objective search problem that can be formalized as follows:

$$\max_{S \subset \mathbb{D}} \ Fitness(S) = (Gini(S), GD(S)) \tag{5}$$

where $\mathbb{D}$ is the test dataset, S is a subset of test inputs of size β, function $Fitness : S \longrightarrow \mathbf{R}^2$ consists of two real-value objective functions $(Gini(S), GD(S))$, and $\mathbf{R}^2$ is the objective space of our optimization problem. Given our two objectives, the "*max*" operator in Eq. 5 returns non-dominated solutions forming a Pareto front in the space defined by Gini and GD [28], i.e., solutions where at least one of the objective function values is the maximum and the other remains unsurpassed by other solutions.

2.3 Genetic Operators

We describe below the two genetic operators in our test selection approach. The first operator is crossover, which generates new offspring by cutting and joining high-fitness parents. The second operator is mutation which introduces small changes in individuals by mutating specific genes to (1) make the search more exploratory and (2) thus attempt

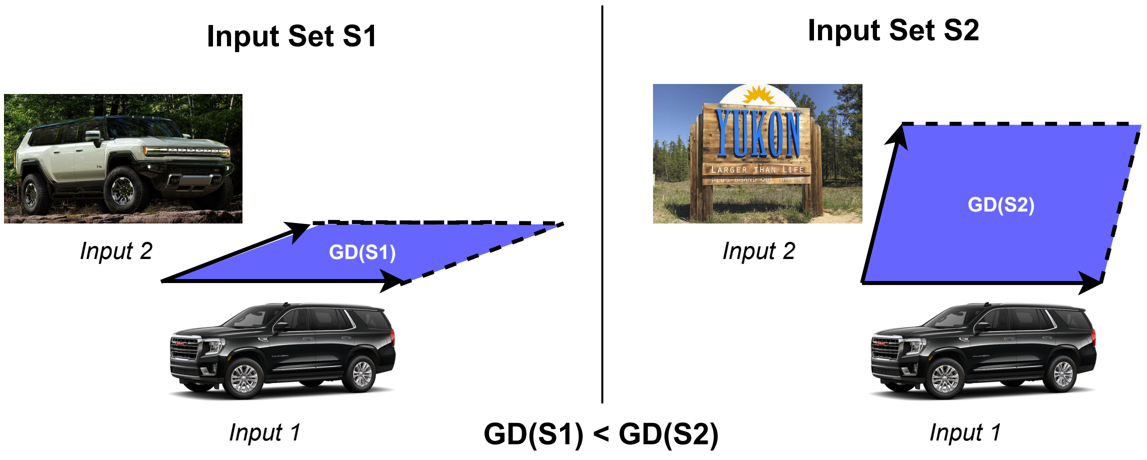

Fig. 1. Illustration of the geometric diversity metric [17]

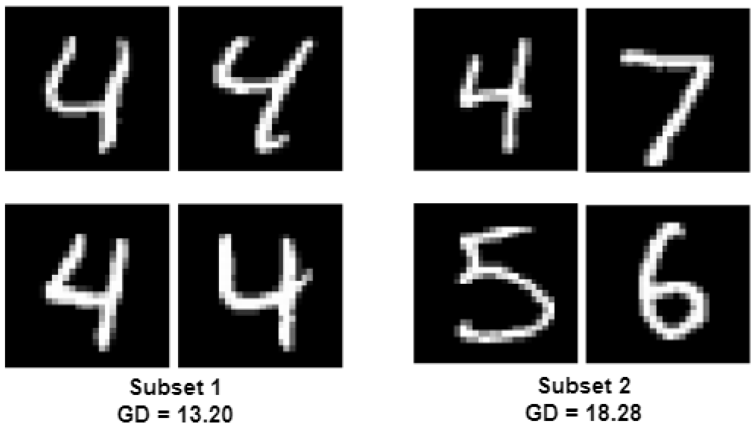

Fig. 2. Examples of geometric diversity scores for two different subsets

to increase the uncertainty score and the diversity among individuals in the population. We provide below a detailed description of how we customized these genetic operators to fit our test selection problem.

2.3.1 ***Crossover***. The crossover operator takes as input two parents (i.e., two subsets) and generates new offspring by slicing and joining the different parts of the selected parents. Let S_1 and S_2 be the two selected parents for crossover. We provide next an example of selected parents:

$$S_1 = \{Input_1, Input_2, Input_3, \cdots, Input_i^{S_1}, \cdots, Input_4, Input_5\}$$

$$S_2 = \{Input_6, Input_7, Input_8, \cdots, Input_i^{S_2}, \cdots, Input_9, Input_{10}\}$$

where (1) $Input_i^{S_1}$ and $Input_i^{S_2}$ denote the i^{th} input in S_1 and S_2, respectively and (2) $Input_{1\leq j\leq 10}$ denote all inputs in each individual. Before applying the crossover operator, we start by sorting the inputs forming each parent according to their *Gini* scores. Inputs with higher *Gini* scores are placed at the beginning of each corresponding parent, as in the example below.

$$S_1 = \{Input_3, Input_1, Input_5, \cdots, Input_i^{s_1}, \cdots, Input_4, Input_2\}$$

$$S_2 = \{Input_7, Input_{10}, Input_6, \cdots, Input_i^{s_2}, \cdots, Input_9, Input_8\}$$

Such reordering will help with the creation of potential high-fitness offspring with high uncertainty scores as we will explain in the following. After such sorting, we randomly select a crossover point using the uniform distribution. To form the first offspring, we slice and join the first parts of each parent based on the crossover point. Such offspring includes inputs with the highest *Gini* scores thanks to sorting. Finally, the second offspring is formed by joining the remaining parts of the selected parents. Since this offspring has inputs with the lowest *Gini* scores from both parents, it will be potentially discarded later by the selection operator or improved by the mutation operator as we will detail in the next section. Assuming that the crossover point is at position *i* in the example above, the generated offspring would be:

$$Offspring_1 = \{Input_3, Input_1, \cdots, Input_i^{s_1}, Input_7, Input_{10}, \cdots, Input_{n-i}^{s_2}\}$$

$$Offspring_2 = \{Input_{i+1}^{s_1}, \cdots, Input_4, Input_2, Input_{n-i+1}^{s_2}, \cdots, Input_9, Input_8\}$$

However, applying the crossover on the selected parents may lead to redundant inputs in the created offspring. Since one of our search goals is to maximize the diversity of the selected subsets, we remove such redundant inputs (and therefore increase the diversity of the offspring) by replacing them with random inputs that are not present in the created offspring. We should note that we do not use the GD metric to customize the crossover operator since it is computationally expensive. Instead, as described in the next section, we employ it in the mutation operator since mutations occur less frequently.

2.3.2 ***Mutation***. After completing the crossover, the offspring are selected for mutation to randomly change some genes that have (1) a low *Gini* score, and (2) a low contribution to the diversity of the selected offspring. Therefore, the mutation operator is considered in our search approach as a corrective operator with the goal of improving the offspring in terms of both fitness functions. As an example, let us assume that S_m is the test input set to mutate. We select 2% of the inputs from S_m that have the lowest *Gini* score. We mutate half of these inputs that have the least contribution to the diversity of S_m. Consequently, only 1% of its genes are mutated. This choice was inspired by recommendations for multi-point mutations in the literature [47–49]. We should recall that to achieve our desired final mutation rate of 1%, we initially select 2% of the genes. We compute the contribution to the diversity of the latter and retain the 1% of the genes that show the highest similarity. This allows us to contain the high cost of computing diversity associated with all genes while focusing mutation on genes contributing the least to diversity.

To measure the contribution of an $input_i$ to increasing the diversity in S_m, we measure the difference $GD_{diff}(S_m, i)$ between $GD(S_m)$ and $GD(S_m \setminus \{input_i\})$. The lower the difference, the more similar the input compared to the other ones in S_m. Inputs with low differences should therefore be mutated to accelerate our search process.

An example of offspring generated through mutation is provided below. S_m is the original offspring and S'_m is the mutated one. Suppose that S_m has 10 genes. We mutate $Input_1$ since it has a low *Gini* score and a low contribution to the diversity of the selected offspring. We mutate this gene by replacing it with a randomly selected input from the population. In this example, we replace $Input_1$ with $Input_{13}$. To maintain diversity, we make sure that the randomly selected input is distinct from any existing input in the original offspring.

S_m :	$Input_3$	$\mathbf{Input_1}$	$Input_5$	$\cdots$	$Input_7$	$Input_{10}$
Gini Score:	**70%**	**71%**	75%	$\cdots$	95%	96%
GD_{diff}:	60	**10**	$\cdots$			
S'_m :	$Input_3$	$\mathbf{Input_{13}}$	$Input_5$	$\cdots$	$Input_7$	$Input_{10}$

It should be noted that each genetic operator has a distinct emphasis on a particular fitness function. The crossover operator primarily aims to generate offspring with high uncertainty scores, while the mutation operator also helps improving the offspring in terms of diversity. Crossover and mutation operators therefore complement each other.

Algorithm 1: High-level NSGA-II algorithm

```
Input: The initial population ($P$) that is a set of test subsets, the number of generations ($G$), the crossover rate ($CrossRate$), the
       mutation rate ($MutationRate$)
Output: A set of individuals ($\alpha$) maximizing the fitness function ($Fitness$)
1  $gen \leftarrow 0$
2  while $gen \leq G$ do
3      $P_{new} \leftarrow \varnothing$
4      $P_{Cross} \leftarrow \varnothing$
5      $P_{Cross} \leftarrow Cross(P, CrossRate)$
6      $P_{new} \leftarrow Mutate(P_{Cross}, MutationRate)$
7      $P \leftarrow Select(P, P_{new}, Fitness)$
8      $Update(\alpha)$
9      $gen \leftarrow gen + 1$
10 return $\alpha$
```

2.4 Search Algorithm

As explained earlier, the main objective of our search algorithm is to select from a large unlabeled test dataset, an input set of size β with a high fault-revealing power in order to (1) reduce labeling cost, and (2) effectively test the DNN model. Algorithm 1 describes at a high level our NSGA-II search process. Assuming P is the initial population, that is a set of $\mid P \mid$ randomly selected test input sets, the algorithm starts an iterative process, taking the following actions at each generation until the maximum number of generations (G) is reached (lines 2-10). The search process includes the following steps. First, we create a new empty population P_{new} (line 3). Second, based on predefined crossover and mutation rates, we create offspring using crossover (*Cross*) and mutation (*Mutate*) operators and add newly created individuals to P_{new} (lines 4-6). Third, we calculate the fitness of the new population by computing *Gini* and geometric diversity scores of each subset in the population and select individuals for survival using tournament [50] selection (*Select*) (line 7). Finally, we update the archive α (line 8) by storing the fittest individuals in the population then we move to the next generation (line 9). The non-dominated solutions contained in the population of the last generation α represent the final *Pareto* front of the genetic search. To select the final solution from the Pareto front, we use the knee point technique [51] as recommended by several existing studies [52]. The solution that corresponds to the knee point has been shown to provide the best trade-off between the different objectives in many multi-objective problems [51, 52]. To calculate the knee point, we first identify the *ideal point*, where the coordinates represent the highest fitness scores (GD_{max},$Gini_{max}$) obtained from all solutions on the Pareto front, with each fitness function evaluated independently. Given the solutions at the Pareto front $P = \{S_1, ..., S_p\}$, the knee point $S_k \in P$ is the solution that minimizes the distance $\sqrt{(GD_{max} - GD(S_i))^2 + (Gini_{max} - Gini(S_i))^2}$, for all $S_i \in P$.

We set the NSGA-II parameters in our search algorithm as follows. The crossover rate *CrossRate* is equal to 75%, given recommended values between 45% and 95% [52–54]. The mutation rate *MutationRate* is equal to 70%. Although the recommended mutation rate in the literature is proportional to the length of the individual [52–54], we have not used such a rate since we used mutation for a different purpose than just exploration, i.e., to help increase both uncertainty and diversity scores. According to our preliminary experiments (which we do not include in this paper), we found that smaller mutation rates, consistent with literature recommendations, led to poorer search results. The population size is set to 700 individuals. Furthermore, the stopping criterion is set to 300 generations. We should note that the number of generations was determined through empirical evaluation. This was done by monitoring the evolution of fitness functions over multiple generations using various datasets and models. The maximum number of generations was carefully selected to ensure the convergence of the search process. Finally, we used Google Colab and the Pymoo library [55] to implement our genetic search.

3 EMPIRICAL EVALUATION

This section describes the empirical evaluation of *DeepGD*, including the research questions we address, the datasets and DNN models on which we perform our assessments, our experiments, and our results.

3.1 Research Questions

Our empirical evaluation is designed to answer the following research questions.

RQ1. Do we find more faults than existing test selection approaches with the same testing budget? Similar to traditional software testing, selecting a subset of test inputs with high fault-revealing ability is highly important in DNN testing, as it should increase the effectiveness of the testing process while reducing input labeling costs. Consequently, we aim in this research question to compare the effectiveness of our approach (*DeepGD*) with existing baselines in terms of their ability to reveal faults in DNNs, while considering the same testing budget. Identifying the source of failures in traditional software systems is relatively straightforward due to the clear and explicit decision logic in the code. However, in DNNs, this task is more challenging as the complexity and non-linearity of the decision-making process make it difficult to determine the cause of the failure. Hence, many papers rely on mispredictions [7, 10, 21] for test selection evaluation. However, similar to failures in traditional software systems, many mispredicted inputs can be due to the same faults in the DNN model and are therefore redundant [29, 56]. When selecting inputs on a limited budget, we should therefore avoid similar or redundant mispredictions as they do not help reveal additional root causes or faults in DNN models. To accurately answer this research question, we thus rely on a clustering-based fault estimation approach [17] that we describe in section 3.2, to investigate the effectiveness of the test selection approaches in detecting faults for a fixed testing budget.

RQ2. Do we more effectively guide the retraining of DNN models with our selected inputs than with baselines? Retraining DNN models with unseen inputs carefully selected based on DNN testing results is expected to enhance the model's performance compared to that of the original training set. More specifically, it is highly recommended to retrain DNNs with inputs that have the potential to lead to mispredictions [12, 21, 56]. Consequently, we aim in this research question to investigate the effectiveness of *DeepGD* in guiding the retraining of DNN models by selecting inputs that will be labeled and used to improve the model through retraining. We will not only measure the accuracy improvement resulting from retraining but also analyze it considering the maximum improvement possible based on the available data for retraining.

RQ3. Can *DeepGD* select more diverse test input sets? We aim to assess whether *DeepGD* selects more diverse test input sets compared to test selection baselines. Diversity in test input selection is important since it is correlated to faults as we reported in a prior study [17]. Specifically, more diverse test input sets reveal more faults in DNN models. Moreover, selecting diverse test inputs mitigates the risk of bias by ensuring that the testing process is not disproportionately skewed towards specific test cases or testing scenarios. It also prevents the detection of redundant faults in DNN models and thus helps optimize the testing budget and effort, since we are more likely to discover unique faults rather than repeatedly identify the same problems in the DNN model under test.

RQ4. How do *DeepGD* and baseline approaches compare in terms of computation time? We aim to compare the computation times of *DeepGD* and baseline approaches. Specifically, we aim to examine how the computational times change as the size of the test input sets increases. It is essential to understand this scalability factor, as excessively long computation times could potentially limit the practicality and real-world applicability of the studied test selection approaches.

3.2 Faults Estimation in DNNs

Before addressing our research questions, we will touch upon the issue of counting faults in DNN models. The motivation for counting faults, when evaluating selection techniques, stems from our objective of detecting different root causes for mispredictions. Considering misprediction rates to compare test selection techniques is highly misleading as many test inputs can be mispredicted for the same reasons [17, 29]. Similarly, in traditional software testing, the focus of testers is not on selecting test inputs that maximize the failure rate, which is equivalent to the misprediction rate in our context. The focus is instead on maximizing the number of distinct faults detected in the system. According to Chan *et al.* [57], in traditional software testing, failure-causing inputs tend to be dense and close together. The same insight applies to DNN model testing, since similar mispredicted inputs tend to be due to the same fault [17, 56].

The above principles should not be different when testing DNNs, where we aim to identify distinct causes of mispredictions (i.e., faults) [17] and address them, for example, through re-training. We illustrate this point in Figure 3 where a test input set is represented in two-dimensional space. Red and black dots correspond to mispredicted and correctly predicted inputs, respectively. We then select two subsets of the same size from the original test input set and measure their misprediction rate (i.e., the number of mispredicted inputs over the size of the subset). As depicted in Figure 3, subset 1 exhibits less diversity compared to subset 2, as some of the former's mispredicted inputs are highly similar and somewhat redundant. Although subset 1 has a higher misprediction rate (70% for subset 1 vs. 40% for subset 2), the second subset is more informative and effective for testing the DNN model. This is because it is more diverse and contains mispredicted inputs that potentially reveal more faults in the DNN model. A test set that consistently highlights the same faults in a DNN model is a waste of computational resources, particularly when faced with constraints such as limited testing budgets and high labeling costs for testing data [17, 35]. Therefore, akin to previous research on test selection in traditional software, our objective is not focused on maximizing the occurrence of failures (i.e., mispredictions) but rather to develop a test selection method that maximizes the identification of faults present in the DNN model.

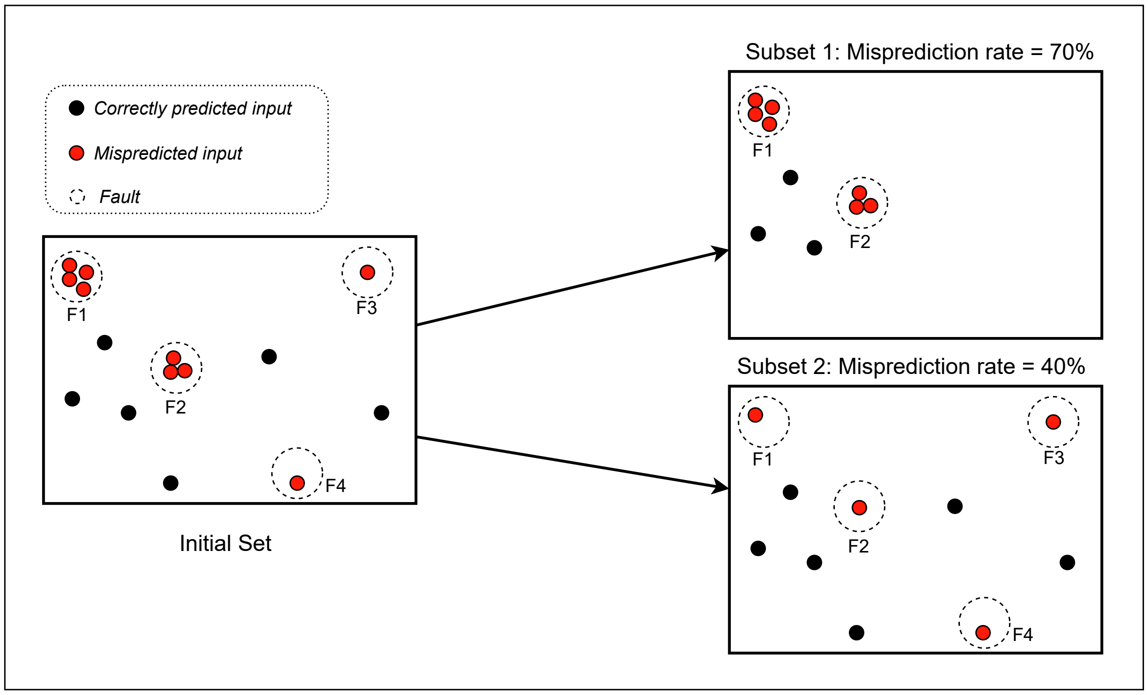
Correctly predicted input
Mispredicted input
Fault
Subset 1: Misprediction rate = 70%
Subset 2: Misprediction rate = 40%
Initial Set
F1
F2
F3
F4

Fig. 3. Relying on misprediction rates is misleading [17]

Given the inherent complexity of detecting faults in DNNs, which is not as straightforward as identifying specific statements responsible for failures in traditional software, we rely on prior work for estimating faults in DNN models [17]. We cluster similar mispredicted inputs that exhibit common characteristics likely responsible for mispredictions. We then approximate the number of detected faults in the DNN model by counting the total number of clusters. The proposed clustering-based fault estimation approach consists of four main steps: feature extraction, dimensionality reduction, clustering, and evaluation. The first step relies on VGG-16 to extract the feature matrix of the mispredicted inputs [17]. Then, two extra features, actual and mispredicted classes related to each input, are added to the feature matrix. These two features as explained in our prior work [17], give us information about the misprediction behavior of the model under test and help build better clusters to reflect common causes of mispredictions. In the next step, dimensionality reduction techniques aim to boost the performance of clustering given the high dimensional feature space. HDBSCAN [58] is then applied to cluster mispredicted inputs based on the resulting features. Final clusters are investigated with SOTA clustering evaluation metrics and through manual analysis. Empirical results have shown that (1) inputs in the same cluster are mispredicted due to the same root causes, and (2) inputs belonging to different clusters are mispredicted because of distinct faults [17]. Hence, although it is an approximation, this approach offers a practical and plausible method for estimating and comparing the number of detected faults across various test selection methods.

To investigate the effectiveness of test selection approaches in a DNN and for the reasons mentioned above, we do not rely on misprediction rates as it is highly misleading. We rely instead on the fault detection rate (*FDR*) which we compute as follows:

$$FDR(S) = \frac{|F_s|}{min(|S|, |F|)} \tag{6}$$

where S is the selected test input set, $|F_s|$ is the number of faults revealed by S, $|S|$ is the test input set size, and $|F|$ is the total number of faults revealed by the entire dataset. The goal of our FDR metric is to measure the extent to which a test set can detect the maximum number of faults that can be found given a testing budget. This is why we compute the ratio of the number of faults actually revealed by the selected test set to the maximum potential number of faults that the subset could unveil. This potential is naturally capped by the size of the subset (i.e., testing budget) or the total number of faults in the whole dataset. We should note that testers cannot know in advance the total number of faults in the dataset, and testing budgets are determined by the resources available. Our definition of FDR ensures that its range is always between 0 and 1, enabling us to compare the outcomes of various methods across test subsets. We present two illustrative examples of how we calculate the *FDR* using the test selection scenario depicted in Figure 3. Given a test dataset that reveals a total number of faults $|F|$ equal to four, a test input set size $|S|$ equal to 10, and a selected subset (see *Subset2*) that reveals four mispredicted inputs each belonging to a distinct fault (cluster), in this case the $FDR(S)$ is calculated as follows: $FDR(S) = \frac{4}{min(10,4)} = 100\%$. Now, considering the same test dataset but selecting only three inputs ($|S| = 3$), which include two mispredicted inputs originating from the same fault ($|F_s| = 1$), the $FDR(S)$ is calculated as: $FDR(S) = \frac{1}{min(3,4)} = 33\%$. The above examples illustrate that FDR represents the percentage of faults that can be detected in a subset relative to the maximum number of faults that can be detected, accounting for both the subset size and the total number of faults in the entire dataset.

3.3 Subject Datasets and Models

Table 1 shows the combinations of datasets and models used in our experiments. For our large-scale empirical evaluation on image classification systems, we consider a set of four well-known and publicly available image recognition datasets which are MNIST [46], Cifar-10 [59], Fashion-MNIST [46] and SVHN [60].

MNIST is a dataset containing 70,000 black-and-white images of handwritten digits (60,000 for training and 10,000 for testing). Each test input is an image representing a digit from zero to nine. The Cifar-10 dataset contains 60,000 colored images belonging to 10 different classes (e.g., cats, dogs, airplanes). It includes 50,000 images for training and 10,000 for testing. We also use Fashion-MNIST, containing 70,000 grayscale images of clothes (60,000 for training and 10,000 for testing). Each one of the images is associated with a label from 10 classes of fashion and clothing items. The Street View House Numbers (SVHN) is a real-world dataset that contains digit images of house numbers collected by Google Street View. It includes 99,289 images in total, including 73,257 for training and 26,032 for testing.

We use these datasets with five state-of-the-art DNN models: LeNet1, LeNet4, LeNet5, ResNet20, and a 12-layer Convolutional Neural Network (12-layer ConvNet). Because we compare our results with SOTA baseline approaches [7, 8, 10, 12, 13, 21, 61], we used similar combinations of models and datasets, focusing on the most widely used ones. Further, these models and datasets involve a variety of distinct inputs (in terms of classes and domain concepts) and different internal architectures, and thus constitute a good benchmark for observing key trends in DNN test selection. Finally, based on previously reported results [17], we provide the accuracy and the number of faults of the used models in Table 1. We should note that in our study, we intentionally trained some of our DNNs in a slightly suboptimal manner to make room for improvement in the retraining experiments conducted in RQ2. The accuracy levels achieved in our models are nonetheless consistent with existing studies on test selection for DNN models [12].

Table 1. Datasets and models used for evaluation

Dataset	DNN Model	Accuracy	# Faults
MNIST	LeNet5	87.85%	85
	LeNet1	84.5%	137
Fashion-MNIST	LeNet4	88%	141
Cifar-10	12-layer ConvNet	82.93%	187
	ResNet20	86%	177
SVHN	LeNet5	88%	147

3.4 Baseline Approaches

We compare *DeepGD* with two categories of baseline approaches: (1) four white-box test selection approaches including three well-known coverage-based approaches [7], along with Likelihood-based Surprise Adequacy (LSA) and Density-based Surprise Adequacy (DSA) [10], and (2) four SOTA black-box prioritization approaches including Maximum Probability (MaxP) [13], DeepGini [21], Adaptive Test Selection (ATS) [12] and Random Selection (RS). We have selected the most prominent and recent baselines that can be applied and replicated on our models and datasets.

A- White-Box Test Selection Baselines

Neuron Coverage (NC). It is the first coverage metric that has been proposed in the literature to test DNN models [7]. It is defined as the ratio of neurons activated by a test input to the total number of neurons in the DNN model. A neuron is activated when its activation value is greater than a predefined threshold.

Neuron Boundary Coverage (NBC). This coverage metric measures the ratio of corner case regions that have been covered by test input(s). Corner case regions are defined as the activation values that are below or higher than the activation ranges observed during the training phase of the DNN model under test.

Strong Neuron Activation Coverage (SNAC). Similar to the NBC metric, SNAC measures how many upper corner cases have been covered by test input(s). Upper corner cases are defined as neuron activation values that are above the activation ranges observed during the training phase of the DNN model under test.

LSA and DSA. These two metrics have been proposed by Kim *et al.* [10] and are based on the analysis of how surprising test inputs are with respect to the training dataset. LSA uses Kernel Density Estimation (KDE) [62] to estimate the likelihood of seeing a test input during the training phase. According to Kim *et al.* [10], test inputs with higher LSA scores are preferred since they are closer to the classification boundaries. Thus, it could be considered a priority score for DNN test selection. DSA is an alternative to LSA that uses the distance between the activation traces [10] of new test inputs and the activation traces observed during training.

B- Black-Box Test Selection Baselines

DeepGini. It is a test selection approach that prioritizes test inputs with higher uncertainty scores [21]. It relies on the *Gini* metric (Section 2.2.1) to estimate the probability of misclassifying a test input.

MaxP. It relies on the maximal prediction probability of the classification task to estimate the prediction confidence of the DNN model for a given input. Such method prioritizes inputs with lower confidence. The maximum probability score of a test input x is defined as $MaxP(x) = 1 - max_{i=1}^{m} P_{x_i}$, where $P_{x_{1 \leq i \leq m}}$ is the probability that input x belongs to class C_i and m is the total number of classes [13]. Intuitively, higher MaxP scores are more likely to lead to mispredictions [13].

ATS. It is a recent test selection method proposed by Gao *et al.* [12]. The selection is guided by a fitness function based on which test inputs are incrementally added to the final test set. The fitness function measures the difference between a test input and the currently selected test set based on computing a fault pattern coverage score. This score is obtained through analyzing the diversity of the output probability vectors of the DNN under test. They select test inputs with different fault patterns and higher uncertainty scores.

Random Selection (RS). It is the most basic and simplest test selection method in the literature. RS consists in randomly selecting (without replacement) β inputs from the test dataset. Each test input has the same probability of being selected.

3.5 Choice of Operators

In this work, we made modifications to the traditional NSGA-II algorithm by customizing its fundamental operators as explained in Section 2.3. Specifically, we introduced a novel crossover function and mutation operator. To ensure each of our customized operators had a beneficial impact on the search results, we conducted an empirical study by considering four variants of the search algorithm. Each variant focused on changing only one search operator. In the first variant of *DeepGD*, we replaced the customized crossover operator with the standard version, where no reordering of the genes is applied based on their uncertainty score. In the second search variant, we replaced the customized mutation operator with the standard version, where 1% of the genes in the selected individuals are randomly mutated, regardless of their uncertainty and diversity scores. Finally, we introduced more mutation strategies in the two other variants. Specifically, in the third variant, we mutated the genes with the lowest uncertainty scores, aiming to explore the impact of prioritizing uncertainty in the mutation process. In the fourth variant, we focused on mutating the genes with the lowest diversity score, investigating the effect of emphasizing diversity in the mutation operation.

Table 2. Fault Detection Rates of *DeepGD*'s Variants

	Method	FDR	
		Fashion-Mnist & LeNet4	Cifar-10 & 12-layer ConvNet
Subset size=100	Original DeepGD	**39.56%**	**59.67%**
	DeepGD with Simple Crossover	31.00%	54.00%
	DeepGD with Simple Mutation	32.50%	49.00%
	DeepGD with Gini-based Mutation	38.00%	52.00%
	DeepGD with GD-based Mutation	37.00%	52.50%

We evaluated the different variants of *DeepGD* on two models and datasets, considering a testing budget of $\beta = 100$. The search results are reported in Table 2. Our findings demonstrate that *DeepGD* consistently outperforms all search variants with its custom operators in terms of fault-revealing power across datasets and models. More specifically, reordering the genes according to their uncertainty scores when applying the crossover, helped the search converge faster and yielded better fault detection results. Moreover, incorporating both diversity and *Gini* scores in the mutation process resulted in better fault detection results compared to using the default version or relying solely on *Gini* or diversity scores for mutations. These results therefore justify our operators' customization choices.

4 EVALUATION AND RESULTS

We describe in this section our experimental evaluation and present in detail the obtained results. Before addressing our research questions, it is important to note that we applied the fault estimation approach described in Section 3.2 to all models and datasets. This allowed us to identify the faults in each subject and estimate their number, which we report in Table 1.

4.1 RQ1. Do we find more faults than existing test selection approaches with the same testing budget?

To investigate the effectiveness of test selection approaches in a DNN, existing baselines usually compare the misprediction detection rate [12, 21]. Although the number of mispredicted inputs is a useful metric in some contexts, it is misleading for test selection in DNN-based systems since many mispredicted inputs may be redundant and caused by the same fault or root cause in the DNN model [17]. Therefore, we compare the effectiveness of *DeepGD* with the existing baselines based on the fault detection rate. However, in order to provide complete information about our analysis, the number of mispredicted inputs revealed by each approach is reported in our replication package [63].

Detailed results of the fault detection rate for six subjects (resulting from the combination of datasets and DNN models), and for different sizes of the test input set $\beta \in \{100, 300, 500\}$, are shown in Table 3, Table 4, and Table 5, respectively. To provide a more comprehensive evaluation of our results, we also report the average number of faults covered by each selection method later in Tables 9 through 11 of Appendix 1. The results follow similar trends and the final conclusion is consistent for all test subset sizes. Because of randomness in *DeepGD*, ATS, and random selection, we re-executed each of them 10 times and reported the corresponding average fault detection rates and average number of

Table 3. The fault detection rate for each subject with test subset size $\beta = 100$

		White-box										Black-box				
Data	Model	NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	13%	21%	13%	12%	12%	12%	11%	13%	31%	35%	14%	55%	54%	53%	**59%**
	ResNet20	10%	10%	11%	17%	16%	11%	17%	16%	42%	37%	11%	52%	56%	50%	**57%**
MNIST	LeNet1	25%	15%	16%	19%	12%	16%	18%	12%	31%	23%	12%	41%	28%	40%	**42%**
	LeNet5	16%	13%	13%	17%	16%	14%	19%	16%	29%	33%	13%	35%	34%	36%	**40%**
Fashion	LeNet4	5%	18%	14%	15%	14%	14%	15%	14%	17%	34%	10%	**39%**	**39%**	32%	**39%**
SVHN	LeNet5	9%	4%	11%	12%	11%	11%	12%	11%	13%	19%	11%	43%	43%	44%	**47%**

Table 4. The fault detection rate for each subject with test subset size $\beta = 300$

		White-box										Black-box				
Data	Model	NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	19%	28%	21%	22%	23%	22%	22%	22%	35%	37%	22%	54%	53%	50%	**57%**
	ResNet20	15%	17%	17%	17%	19%	17%	17%	19%	49%	45%	19%	56%	58%	52%	**59%**
MNIST	LeNet1	25%	25%	33%	28%	27%	36%	29%	27%	40%	40%	25%	53%	44%	53%	**54%**
	LeNet5	34%	33%	32%	33%	30%	32%	30%	30%	54%	52%	24%	63%	59%	62%	**64%**
Fashion	LeNet4	8%	25%	19%	23%	23%	19%	23%	23%	35%	45%	17%	51%	47%	49%	**53%**
SVHN	LeNet5	11%	13%	17%	16%	18%	17%	16%	18%	19%	36%	17%	58%	61%	58%	**62%**

Table 5. The fault detection rate for each subject with test subset size $\beta = 500$

		White-box										Black-box				
Data	Model	NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	30%	34%	32%	33%	30%	30%	31%	29%	45%	51%	33%	67%	65%	64%	**70%**
	ResNet20	28%	28%	28%	29%	30%	28%	29%	30%	61%	63%	27%	68%	71%	63%	**73%**
MNIST	LeNet1	29%	31%	43%	39%	36%	42%	38%	36%	52%	58%	34%	66%	58%	65%	**67%**
	LeNet5	39%	42%	41%	46%	40%	42%	42%	40%	65%	65%	38%	73%	72%	71%	**74%**
Fashion	LeNet4	13%	28%	28%	29%	27%	28%	29%	27%	46%	54%	27%	**70%**	65%	64%	**70%**
SVHN	LeNet5	14%	22%	24%	22%	25%	24%	22%	25%	27%	46%	26%	77%	76%	72%	**78%**

detected faults (see Appendix 1). Our results show that all of the test selection approaches guided by coverage metrics are ineffective in detecting faults. Even after repeating the experiment with various parameters as suggested in their original papers, the results are similar. The fault detection rate of NC, for example, was between 4% and 25% for subset sizes of 100. SNAC and NBC also showed poor fault detection rates, which vary between 11% and 19% for the same subset size. Compared with LSA and DSA, both *DeepGD* and other black-box prioritization selection methods showed a much higher fault detection rate for all subjects. Considering our previous experiments [17], where we investigated white-box coverage metrics and reported that they are not correlated to faults, it is not surprising that these approaches are unlikely to perform as effectively as black-box approaches in the selection of test inputs with high fault-revealing power. Overall, it can be concluded that black-box approaches are more effective in detecting faults than white-box approaches. *DeepGini* and MaxP fare the same as *DeepGD* for two different subjects with a test subset size of 100. However, for other subjects, *DeepGD* achieves a higher fault detection rate than these two approaches. For larger test subset sizes ($\beta \in \{300, 500\}$), *DeepGD* performs better than other black-box baselines in all subjects, with a fault detection rate between 53% and 78%.

We should note that the second-best approach is not consistently the same across subjects and sizes, therefore strengthening our conclusion that *DeepGD* is the only solution we can confidently recommend regardless of the model, dataset, and test set size. For example, with $\beta = 100$, compared to black-box SOTA baselines (ATS, MaxP, Gini), *DeepGD* is, on average, 2 pp better (with a maximum of 4 pp across all models and datasets) than the second-best black-box alternative and 7 pp better (with a maximum of 14 pp across all models and datasets) than the fourth-best black-box alternative (excluding random selection which consistently showed poor performance). Since we cannot predict beforehand how a given alternative will fare compared to the others for a dataset and model, this implies that the effect of selecting another technique than *DeepGD* is potentially significant as we may end up using the worst alternative. For example, if we rely on the inputs selected by *DeepGini* to test LeNet1 on the MNIST dataset ($\beta = 300$), we will only reach a fault detection rate of 44% instead of 54% with *DeepGD*. Further, we also report that *DeepGD* is on average 15 pp better than the best white-box test selection approach and 41 pp better than the worst white-box alternative across all models and datasets.

We performed the statistical analysis using the Wilcoxon signed-rank test [64], with a significance level of $\alpha = 0.05$, to investigate whether *DeepGD* significantly outperforms each SOTA baseline in terms of fault-revealing power. The Wilcoxon signed-rank test, a non-parametric test, is used to compare the medians of continuous variables for paired samples and is employed when the data does not follow a normal distribution, as in our case. We employ a paired test as we compare the fault-revealing power of methods for a given subset size and model. To conduct the statistical tests, we collected data about the performance of each method (DeepGD, ATS, GINI, and MaxP) across the 18 different combinations of datasets, models, and subset sizes. For each SOTA baseline, we thus compared pairs of fault detection rates reported for DeepDG and the selected baseline across 18 different combinations. Results showed that all p-values are lower than 0.05, indicating that *DeepGD* significantly surpasses all SOTA baselines in finding more faults in DNNs.

Answer to RQ1: *DeepGD* outperforms both white-box and black-box test selection approaches for DNNs, in terms of detecting distinct faults given the same testing budget. Further, the second-best black-box approach after *DeepGD* is not consistently the same.

4.2 RQ2. Do we more effectively guide the retraining of DNN models with our selected inputs than with baselines?

Having examined the effectiveness of *DeepGD* and other baselines in selecting test input sets with high fault-revealing power, we next focus on the extent to which the test selection approaches can help select data to effectively retrain the DNN models under test. We only consider black-box test selection approaches in this research question, as they showed much better performance than white-box test selection baselines. We consider the same models and datasets as in the previous experiment. We augment the original training dataset with the test input selected in RQ1 with the size of 300 by *DeepGD*, ATS, DeepGini, and MaxP, respectively, to retrain the DNN model. We measure the accuracy of the retrained model on both the whole test dataset and a newly generated dataset that is obtained by applying five realistic image transformations to the original test dataset. The latter allows for a fairer and more complete comparison between our proposed method and the other baseline approaches since none of the inputs in the generated dataset were used for retraining the model. We leveraged the datasets generated by Gao *et al.* [12], a recent work in test selection, that we consider as one of our baselines. The authors applied various transformations to the MNIST, Fashion-MNIST, and Cifar-10 test datasets to generate more valid test inputs. However, since the SVHN dataset was not included in their experiments, to ensure consistency, we generated new test inputs for this dataset as well by applying their transformations with identical settings, ensuring the validity of generated inputs. These transformations include sheering, rotating, zooming, and changing the brightness and the blurriness of all images in each dataset. We should note that the size of the newly generated test dataset is five times larger than the original test dataset since we apply five image transformations on each original test input. For each black-box test selection approach, we measure the accuracy improvement of the retrained models on both the original test dataset and the generated one. We report the results in Table 6. We acknowledge that studying the accuracy improvement of the retrained models on the generated datasets is more suitable in our context since it does not contain any of the inputs used for retraining. But we choose to also report improvements with the original test dataset to verify that the retraining process did improve model accuracy. Note that we only select a small part of the original test dataset to retrain the model (only 300 out of 10,000 to 26,000 inputs). Because of the randomness in *DeepGD* and ATS, we re-executed each of them five times on the different subjects. For each execution, we retrained each subject with the selected test input set and reported the corresponding average model accuracy improvements.

As described in Table 6, we found that *DeepGD* provides better guidance for retraining DNN models than the other black-box test selection approaches. It consistently outperforms ATS, DeepGini, and MaxP in terms of accuracy

Table 6. DNNs accuracy improvements after retraining with the selected test input sets.

		Accuracy imp. on orig. test dataset				Accuracy imp. on generated test dataset			
Data	Model	MaxP	Gini	ATS	**DeepGD**	MaxP	Gini	ATS	**DeepGD**
Cifar-10	12 ConvNet	3.52%	2.09%	2.12%	**3.53%**	5.35%	5.12%	4.62%	**6.85%**
	ResNet20	1.66%	0.74%	2.03%	**2.50%**	4.12%	3.45%	5.97%	**6.18%**
MNIST	LeNet1	10.39%	10.40%	10.40%	**11.10%**	13.18%	13.24%	13.19%	**14.76%**
	LeNet5	7.94%	7.91%	7.92%	**9.26%**	13.02%	12.92%	13.08%	**15.82%**
Fashion	LeNet4	3.94%	3.94%	3.91%	**4.17%**	7.33%	7.41%	7.62%	**7.69%**
SVHN	LeNet5	0.96%	0.99%	0.93%	**1.24%**	0.61%	0.54%	0.63%	**1.26%**

Table 7. Optimization effectiveness compared to retraining with all candidate tests

			Opt effectiveness on orig. test dataset (T)					Opt effectiveness on generated dataset (G)			
Data	Model	Max imp. (100% T)	MaxP	Gini	ATS	**DeepGD**	Max imp. (100% G)	MaxP	Gini	ATS	**DeepGD**
Cifar-10	12 ConvNet	16.36%	21.51%	12.78%	12.96%	**21.58%**	28.98%	18.47%	17.65%	15.94%	**23.65%**
	ResNet20	9.99%	16.62%	7.41%	20.32%	**25.03%**	23.33%	17.65%	14.80%	25.59%	**26.47%**
MNIST	LeNet1	11.10%	93.60%	93.72%	93.69%	**100%**	23.80%	55.39%	55.62%	55.43%	**62.04%**
	LeNet5	9.52%	83.40%	83.08%	83.19%	**97.27%**	22.78%	57.15%	56.72%	57.41%	**69.46%**
Fashion	LeNet4	12.00%	32.83%	32.83%	32.58%	**34.75%**	23.72%	30.91%	31.23%	32.10%	**32.41%**
SVHN	LeNet5	1.25%	77.01%	79.79%	74.84%	**99.90%**	8.92%	6.87%	6.06%	7.04%	**14.15%**
Average		10.04%	54.16%	51.60%	52.93%	**63.09%**	21.92%	31.07%	30.35%	32.25%	**38.03%**

improvements across all models and datasets. Similar to the previously reported results in RQ1, we found that the second-best approach for guiding retraining is not consistently the same across all subjects. For example, ATS was the second-best approach for retraining ResNet20 with Cifar-10, while it was the third-best approach for retraining LeNet5 with SVHN.

To further investigate the effectiveness of *DeepGD* and the other black-box baselines in retraining DNN models, we also computed their *optimization effectiveness* by accounting for the maximum possible accuracy improvement. We therefore retrain the model with the entire original test dataset and report the best accuracy that can be achieved by retraining. Then, for each test selection method, we calculate its optimization effectiveness, defined as the ratio of (1) the accuracy improvement when retaining the model with only 300 selected inputs from the original test dataset (Table 6) to (2) the accuracy improvement when retaining the model with the entire original test dataset. We also repeat the above experiment for the generated test data set to once again get a fairer comparison of the test selection approaches and to obtain better generalizability for our results. More specifically, we retrain the original model by adding all generated inputs to the training dataset to achieve the highest accuracy possible. Then, we calculate the corresponding optimization effectiveness which is the ratio of (1) the accuracy improvement when retaining the model with only 300 inputs from the original test dataset (Table 6) to (2) the accuracy improvement when retaining the model with the entire generated test dataset. Because of the randomness in *DeepGD* and ATS, we re-executed each of them five times on the different subjects and reported the corresponding average results in Table 7.

We found that *DeepGD* consistently outperforms other black-box test selection approaches in terms of optimization effectiveness. Compared to black-box SOTA baselines, *DeepGD* is, on average, 8.93 pp better (with a maximum of 20.11 pp) than the second-best black-box alternative and 11.49 pp better (with a maximum of 25.06 pp) than the worst black-box alternative. As for RQ1, it is worth noting that since the performance of alternatives is not consistent across datasets and models, selecting one of them may yield the worst results. We also report the average optimization effectiveness on the generated test datasets. Gini, MaxP and ATS yield 30.35%, 31.07%, and 32.25% respectively, while *DeepGD* achieves 38.03%. In other words, with only 300 test inputs, from the original test dataset and selected by *DeepGD* for retraining, we were able to reach 38.03% of the maximum achievable accuracy by retraining with all generated test dataset. Similar to RQ1, we performed a statistical analysis using Wilcoxon signed-rank tests, with a significance level of 0.05, to investigate whether *DeepGD* significantly outperforms the selected black-box test selection approaches in terms of optimization effectiveness across all subjects. We found all p-values to be less than 0.05, indicating that *DeepGD* is significantly better than the selected baselines in retraining DNN models. In other words, results show that *DeepGD* can select a small, more informative subset from a large unlabeled dataset to effectively retrain DNN models and minimize the labeling costs. Selecting diverse inputs with high uncertainty scores not only helps at detecting more faults in the DNN model, but also significantly improves the accuracy of the model through retraining.

Answer to RQ2: *DeepGD* provides better guidance than black-box alternatives for retraining DNN models. It consistently and statistically outperforms other black-box test selection approaches in terms of accuracy improvement, in absolute terms and relatively to the maximum achievable improvement.

4.3 RQ3. Can DeepGD select more diverse test input sets?

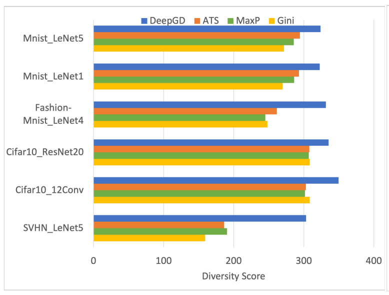

Fig. 4. GD scores for subsets size=100

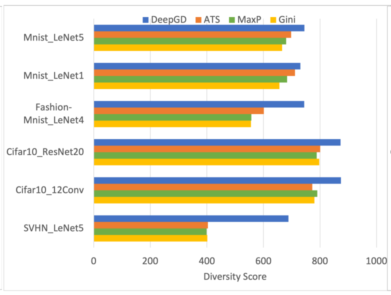

Fig. 5. GD scores for subsets size=300

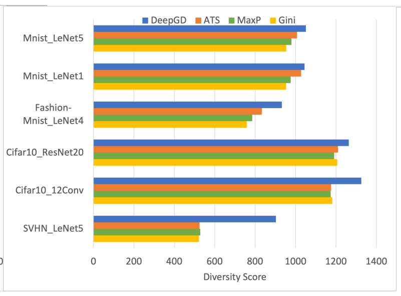

Fig. 6. GD scores for subsets size=500

To answer this research question, we measured the diversity score of the selected test input sets for *DeepDG* and the other black-box baselines. We relied on the geometric diversity metric since we demonstrated in our prior work [17] its construct validity in measuring the actual diversity of input sets. We used the same subjects, subset sizes and test input sets that were selected in RQ1. Given the inherent randomness in *DeepGD* and ATS, we reported the average diversity scores based on 10 executions. The results, presented in Figures 4, 5 and 6, consistently demonstrate that *DeepGD* outperforms the baseline approaches in selecting diverse input sets. Across all subjects and subset sizes, *DeepGD* consistently obtained the highest diversity scores. Additionally, ATS emerged as the second-best approach for selecting diverse input sets in 12 out of 18 configurations, while *Gini* performed in general the poorest, ranking last in 11 out of 18 configurations.

We also performed a statistical test analysis using the Wilcoxon signed-rank test, with a significance level of 0.05, to investigate whether *DeepGD* significantly outperforms each baseline approach in selecting diverse test input sets. Similarly to RQ1 and RQ2, we performed a paired test across the different combinations of datasets, models, and subset sizes. For each baseline approach, we compared pairs of diversity scores reported for *DeepGD* and the selected baseline across the 18 different configurations. Results show that all p-values are below 0.05, indicating that *DeepGD* significantly outperforms all SOTA baselines in selecting more diverse test input sets.

Answer to RQ3: *DeepGD* provides better guidance than black-box alternatives for selecting diverse test input sets. It consistently and statistically outperforms other black-box test selection baselines in terms of diversity.

4.4 RQ4. How do DeepGD and baseline approaches compare in terms of computation time?

To address this research question, we compared the execution time of *DeepGD* with baseline approaches and analyzed its evolution with different subset sizes. Similar to RQ2, we focused on black-box test selection approaches (excluding random selection) as they demonstrated better performance compared to white-box test selection baselines. The subjects and subset sizes used were consistent with our previous experiments. For each subject and subset size, we executed

DeepGD and the selected baseline approaches 10 times and measured the computation times. This repetition allowed us to account for random variations in execution time.

The results, presented in Figure 7, illustrate the change in computation times as the size of the input sets increased. Our findings indicate that all selected approaches were computationally efficient. *Gini* and MaxP exhibited similar execution times, approximately five seconds, and demonstrated the best performance in terms of computation time. In contrast, *DeepGD*, since it relies on a search algorithm, had higher execution times ranging from 200s to 850s across all subjects and subset sizes. Furthermore, we observed that, unlike the other baseline approaches, the execution time of *DeepGD* increases with subset size. However, it is important to note that such execution times remain practically acceptable since, in DNN testing, (1) the labeling cost of test inputs is far more expensive than the computation cost of the search, and (2) test selection is neither frequent nor a real-time task. Despite the longer execution times, the better fault detection performance and retraining guidance provided by *DeepGD* therefore outweigh the increase in computation time in comparison to the baselines.

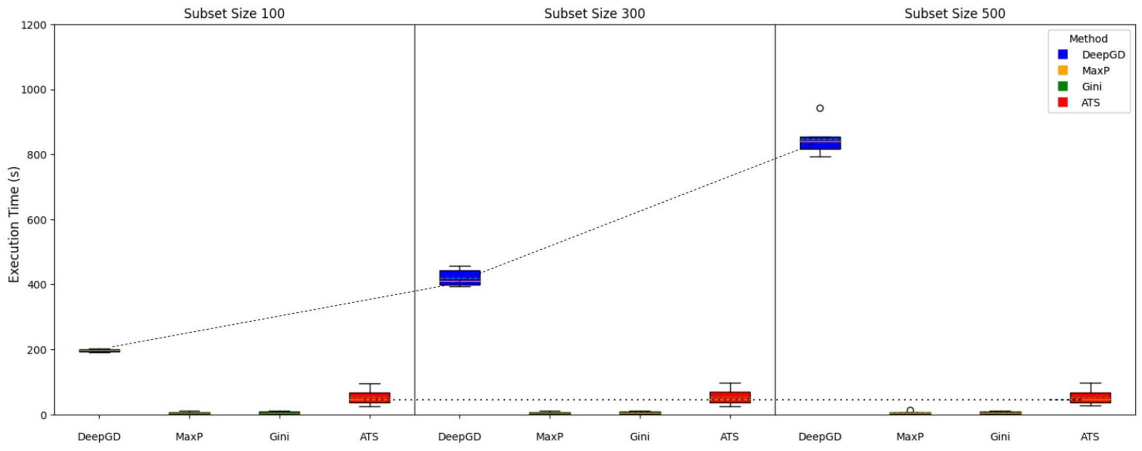

Fig. 7. Evolution of the execution time over the subset sizes

Answer to RQ4: None of the black-box test selection approaches are computationally expensive. Though *DeepGD* yields longer execution times (a matter of minutes), these remain practical, even for large test subsets, given that test selection is not a frequent task performed under tight time constraints.

4.5 Discussions

Based on our experimental results, we show that *DeepGD* provides better guidance than existing baselines for selecting test inputs with high-fault revealing power. The second-best approaches for test selection and model retraining are not consistently the same across all models and datasets. This reinforces our conclusion that *DeepGD* is the only solution that we can confidently recommend regardless of the model, dataset, and test set size. Indeed, selecting diverse test inputs with high uncertainty scores not only enables higher fault detection, but also provides more effective guidance for retraining DNN models. Although results are encouraging, and though this remains to be further investigated, we conjecture that *DeepGD* is particularly useful when datasets are more redundant, a situation that is often observed in real-world DNN testing scenarios, especially with massive generated datasets. Selecting diverse test input sets with *DeepGD* is then expected to help reduce redundancy by selecting more informative inputs to be labeled in order to test and retrain DNN models.

Since *DeepGD* relies on genetic search, it entails some degree of randomness in the final selected test sets. We therefore studied the stability performance of *DeepGD* by running the tool 10 times with each combination of dataset, model, and subset size that we have previously used in our experiments. We report the minimum, maximum, and standard deviation of FDR results in Table 8. As shown in the table, the standard deviations of the FDRs reported by *DeepGD* are consistently low, ranging from 0.008 to 0.021. This indicates that *DeepGD* consistently delivers stable and thus reliable results across subjects and subset sizes over multiple runs. We also found that the faults revealed by *DeepGD* are largely consistent across multiple runs. This can be attributed to the use of the *Gini* score as one of the fitness functions to guide the search towards test inputs with high uncertainty scores.

Table 8. Stability of *DeepDG's* fault detection rates.

		Size =100				Size =300			
Data	Model	Ave. FDR	Min. FDR	Max. FDR	Std.	Ave.	Min. FDR	Max. FDR	Std.
Cifar-10	12 ConvNet	59%	57%	62%	0.019	57%	55%	58%	0.009
	ResNet20	57%	55%	60%	0.013	59%	57%	60%	0.009
MNIST	LeNet1	42%	40%	45%	0.017	54%	51%	54%	0.009
	LeNet5	40%	38%	42%	0.016	64%	60%	68%	0.021
Fashion	LeNet4	39%	36%	43%	0.019	53%	52%	55%	0.008
SVHN	LeNet5	47%	44%	49%	0.017	62%	61%	64%	0.012

Note: The values of average, minimum, and maximum FDRs reported in this table have been rounded up to two decimal digits and are represented as percentages. The standard deviation (std) of FDRs has been rounded up to three decimal digits.

5 THREATS TO VALIDITY

We discuss in this section the different threats to the validity of our study and describe how we mitigated them.
Internal threats to validity concern the causal relationship between the treatment and the outcome. Since *DeepGD* is black-box and relies on extracting feature matrices to measure the diversity of the selected test input sets, an internal threat to validity might be caused by the poor quality representation of inputs. To mitigate this threat, we have relied on VGG-16, one of the most accurate feature extraction models in the literature. This model has been pre-trained on the very large ImageNet dataset that contains more than 14 million labeled images belonging to 22,000 categories. Also, in our prior work [17], we assessed the use of the GD metric with the VGG-16 model to measure diversity in input sets. Our results showed that the GD metric, in conjunction with our selected feature extraction model, performed well in measuring actual data diversity across all the studied datasets. Moreover, *DeepGD* relies on the specification of a few hyperparameters related to NSGA-II. This also applies to several white-box and black-box test selection approaches that we considered in our study. The configuration of the different hyperparameters in our work may induce additional threats. To mitigate them, we have relied on the NSGA-II hyperparameters recommended in the literature [52–54] except for the mutation rate, which was intentionally set higher and experimentally tuned since we customized the mutation operator to take into account both fitness functions, as described in section 2.3.2. We have also considered different configurations of the baselines-related hyperparameters according to their original papers. A last internal threat to validity would be related to randomness when selecting test input sets with *DeepGD*, ATS, and random selection. We addressed this issue by repeating such selection multiple times while considering different input set sizes and different datasets and models. We also reported on the stability results of *DeepGD*, which indicated that it is stable in this context.
Construct threats to validity concern the relation between the theory and the observations made. A construct threat to validity might be due to inaccuracies in estimating DNN faults since detecting faults in DNNs is not as straightforward as in traditional software. To mitigate this threat, we have relied on a SOTA clustering-based fault estimation approach [17] that has been thoroughly validated on several models and datasets. In our previous work, we conducted various validation experiments demonstrating that mispredicted inputs within the same cluster are typically mispredicted due to the same fault in the DNN model, whereas inputs from different clusters are mispredicted due to distinct faults (root causes). Additionally, our prior work established that clusters of mispredicted inputs are very different from those in the entire test dataset despite the use of the same feature extraction model (i.e., VGG-16). Such difference further reinforces the validity of our chosen fault estimation approach. We have reused this publicly available approach to obtain accurate fault estimates. Nonetheless, relying on a such fault estimation approach is still far better than just considering mispredicted inputs that are redundant and due to the same root cause in the DNN model.

External threats to validity concern the generalizability of our study. We mitigate this threat by considering six different combinations of widely used and architecturally distinct models and datasets. We also considered many testing budgets in our experiments and compared our results with nine SOTA baselines for DNN test selection.

6 RELATED WORK

In this section, we introduce existing work related to our proposed approach from two aspects: test selection and test diversity in the context of DNN models.

Test Selection for DNNs. Test selection approaches proposed for DNN models can be characterized as black-box or white-box, depending on their access requirements to the internals of the DNN model. A few black-box test selection approaches for DNNs have been introduced in the literature. They generally rely on the uncertainty of model classifications. For example, Feng *et al.* [21] proposed DeepGini, a test selection approach that prioritizes the selection of inputs with higher *Gini* scores [21]. They conjecture that if a DNN model is unsure about a classification and outputs similar probabilities for each class, it is more likely to mispredict the test input. Compared to random and coverage-based selection methods, DeepGini was shown to be more effective in uncovering mispredictions [21, 39]. Similar to their work, we rely on *Gini* scores as one of the fitness functions in our approach. However, we also consider the diversity of the selected test set and rely on a multi-objective genetic search to guide the search toward finding test inputs with high fault-revealing power. Li *et al.* [22] introduced Cross Entropy-based Sampling (CES) and Confidence-based Stratified Sampling (CSS), for black-box DNN test selection. These metrics are used to select a small subset of test inputs that accurately reflect the accuracy of the whole test dataset. They show that compared to random sampling, their approach could achieve the same level of precision with about half of the labeled data. Their goal is clearly different from ours. While our focus is to prioritize the selection of test inputs with high fault-revealing capability, their goal is to minimize test sets. Also, though Arrieta [65] relied on NSGA-II and uncertainty scores for selecting metamorphic follow-up test cases, our goal with *DeepGD* is also different. We use both diversity and DNN uncertainty to guide the search toward test inputs with high fault-revealing power, for a fixed budget.

Several white-box test selection approaches have been proposed as well. Such approaches generally rely on coverage [7, 8, 61] or surprise metrics [10] to select inputs that will be labeled and used for testing DNN models. For example, Pei *et al.* [7] proposed Neuron Coverage (NC), which measures neurons activation rates in DNN models. Ma *et al.* [8] proposed *DeepGauge*, a set of coverage metrics for DNN models that consider neurons activation ranges. Kim *et al.* [10] proposed surprise coverage metrics which measure how surprising test sets are given the training set. The selection of test inputs for all these approaches is based on maximizing coverage or surprise scores. However, several studies [13, 16–19, 66–68], have shown that these white-box metrics are not always effective for guiding DNNs test selection. For example, Ma *et al.* [13] performed an empirical study and compared the effectiveness of white-box metrics (coverage and surprise metrics) with black-box uncertainty in guiding test input selection. Results showed that the former have a weak or no correlation with classification accuracy while the latter had a medium to strong correlation. Uncertainty-based metrics not only outperform coverage-based metrics but also lead to faster improvements in retraining. In our work, as mentioned above, we also consider maximizing the uncertainty score of the test inputs as one of our two fitness objectives.

Diversity in DNN Testing. Many works have studied diversity-based test selection and generation for traditional software [23, 25, 36]. The underlying assumption is that there is a strong correlation between test case diversity and fault-revealing power [36]. Their results confirmed this assumption and showed that diversity-based metrics are effective in revealing faults. Inspired by these encouraging results, researchers devised diversity-based approaches for DNN testing [12, 17, 26, 35]. Zhao *et al.* [26] conducted an empirical study of SOTA test input selection approaches for DNNs [5, 22, 69] and concluded that they have a negative impact on test diversity and suggest that more research is warranted on designing more effective test selection approaches that guarantee test diversity.

In a recent study, Gao *et al.* [12] proposed the adaptive test selection method (ATS) for DNN models that use the differences between model outputs as a behavior diversity metric. Although ATS aims to cover more diverse faults, its test selection is guided only by model outputs. *DeepGD*, however, considers both the uncertainty of model output probabilities and input features' diversity. Moreover, our results show that *DeepGD* outperforms ATS in test input selection for different combinations of models and datasets and for different test subset sizes. In our prior work [17], we studied three black-box input diversity metrics for testing DNNs, including geometric diversity [70], normalized compression [71], and standard deviation. We investigated the capacity of these metrics to measure actual diversity in input sets and analyzed their fault-revealing power. Our experiments on image datasets showed that geometric diversity outperforms SOTA white-box coverage criteria in terms of fault detection and computational time. However, in that work, we did not study how the GD metric can be used in practice to guide the selection of test inputs for DNN models.

Input Selection for Deep Active Learning. Deep Active Learning (DAL) involves the incremental selection of inputs to be labeled and used for (re)training DNN models. Active learning typically starts with a model that has been trained using a small, randomly selected set of inputs. Then, in each (re)training iteration, a subset of unlabeled inputs is selected based on a query strategy, manually labeled, and then used to (re)train the model and improve its performance. In particular, the objective of each iteration is to select the most informative inputs for labeling that enable reducing the labeling cost while improving the accuracy of the model. DAL methods can be categorized into three classes based on the used input selection strategy: uncertainty-based, diversity-based, and combined approaches that leverage both diversity and uncertainty to select inputs for (re)training DNN models effectively [72–74]. Zhan *et al.* [72] conducted a comparative study including 19 DAL approaches from the three categories. They concluded that the combined DAL approaches such as Wasserstein Adversarial Active Learning (WAAL) [73] and Batch Active learning by Diverse Gradient Embeddings (BADGE) [74], provide in general better guidance for model retraining compared to uncertainty and diversity-based DAL techniques. Both BADGE and WAAL propose a combined approach for input selection based on uncertainty and diversity.

Although BADGE, WAAL, and *DeepGD* appear to share similarities in terms of selection metrics, they diverge in how they measure uncertainty and diversity. For instance, BADGE adopts a white-box approach, relying on the last-layer loss gradients of the model during (re)training to compute uncertainty and construct input feature vectors. It then employs K-means clustering on these feature vectors to select diverse inputs from each cluster. Notably, BADGE primarily operates with DNN models trained using commonly used gradient-based optimizers, unlike *DeepGD* which is agnostic to the model's optimizer. In addition, WAAL requires the initial labeling of 1% to 5% of the (re)training dataset before performing additional and incremental inputs selection. These initially labeled input sets serve as a basis for selecting diverse inputs from the unlabeled dataset, with a focus on those displaying the largest Wasserstein distance w.r.t. the labeled input set [73].

Since active learning techniques require model training-specific data such as gradient loss, uncertainty estimates, or prediction confidence scores during the training epochs [72], they are only applicable during the (re)training phase of DNN models. In contrast, test selection techniques are applicable in both the training and testing phases. It is also worth noting that active learning techniques require human-in-the-loop labeling throughout the entire (re)training process. This iterative selection of inputs during training iterations makes such approaches impractical in many situations. On the other hand, test selection techniques, including *DeepGD*, only require labeling once, within a given testing budget, before testing and retraining the model. Based on the above points, active learning techniques are therefore not adapted to the objectives of automatically and optimally supporting test selection and using the selected test inputs to support retraining, without requiring iterative human intervention for labeling inputs through the retraining process.

7 CONCLUSION

In this paper, we propose *DeepGD*, a multi-objective search-based test input selection approach for DNN models. Our motivation is to provide a black-box test selection mechanism that reduces labeling costs by effectively selecting unlabeled test inputs with high fault-revealing power. We rely on both diversity and uncertainty scores to guide the search toward test inputs that reveal diverse faults in DNNs. We conduct an extensive empirical study on six different subjects and compare *DeepGD* with nine state-of-the-art test selection approaches. Our results show that *DeepGD* statistically and consistently outperforms baseline approaches in terms of its ability to reveal faults in DNN models. Selecting diverse inputs with high uncertainty scores with *DeepGD* not only helps detecting more faults in the DNN model for a given test budget, but also significantly improves the accuracy of the model through retraining with an augmented training set. Our results also indicate that the second-best approach for testing or retraining is not consistent across all models and datasets, further supporting the choice of *DeepGD*. We aim to extend our work by studying the application of diversity and uncertainty metrics for other DNN testing purposes such as test minimization and generation. Finally, we need to investigate alternative ways to estimate faults in DNNs and study the relationship between the fault detection rate and the model repair capability of test selection methods.

ACKNOWLEDGEMENTS

This work was supported by a research grant from General Motors, the Canada Research Chair and Discovery Grant programs of the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Science Foundation Ireland grant agreement No. 957254.

REFERENCES

[1] M. Alam, M. D. Samad, L. Vidyaratne, A. Glandon, and K. M. Iftekharuddin, "Survey on deep neural networks in speech and vision systems," *Neurocomputing*, vol. 417, pp. 302–321, 2020.

[2] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[3] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[4] F. Tambon, G. Laberge, L. An, A. Nikanjam, P. S. N. Mindom, Y. Pequignot, F. Khomh, G. Antoniol, E. Merlo, and F. Laviolette, "How to certify machine learning based safety-critical systems? a systematic literature review," *Automated Software Engineering*, vol. 29, no. 2, p. 38, 2022.

[5] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, and M. Yan, "Practical accuracy estimation for efficient deep neural network testing," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–35, 2020.

[6] A. Zolfagharian, M. Abdellatif, L. C. Briand, and R. S, "Smarla: A safety monitoring approach for deep reinforcement learning agents," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2308.02594

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

[8] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 120–131, 2018.

[9] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 303–314. [Online]. Available: https://doi.org/10.1145/3180155.3180220

[10] J. Kim, R. Feldt, and 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.

[11] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Structural test coverage criteria for deep neural networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.

[12] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, and B. Xu, "Adaptive test selection for deep neural networks," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 73–85. [Online]. Available: https://doi.org/10.1145/3510003.3510232

[13] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, and Y. L. Traon, "Test selection for deep learning systems," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–22, 2021.

[14] Q. Hu, Y. Guo*, M. Cordy, X. Xie, L. Ma, M. Papadakis, and Y. Le Traon, "An empirical study on data distribution-aware test selection for deep learning enhancement," *ACM Transactions on Software Engineering and Methodology*, 2022.

[15] H. Hemmati, "How effective are code coverage criteria?" in *2015 IEEE International Conference on Software Quality, Reliability and Security*. IEEE, 2015, pp. 151–156.

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

[17] Z. Aghababaeyan, M. Abdellatif, L. Briand, R. S, and M. Bagherzadeh, "Black-box testing of deep neural networks through test case diversity," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3182–3204, 2023.

[18] Z. Li, X. Ma, C. Xu, and C. Cao, "Structural coverage criteria for neural networks could be misleading," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 89–92.

[19] Z. Yang, J. Shi, M. H. Asyrofi, and D. Lo, "Revisiting neuron coverage metrics and quality of deep neural networks," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 408–419.

[20] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks," *arXiv preprint arXiv:1803.04792*, 2018.

[21] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and 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.

[22] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and 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.

[23] M. Biagiola, A. Stocco, F. Ricca, and P. Tonella, "Diversity-based web test generation," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 142–153.

[24] H. Hemmati, Z. Fang, and M. V. Mantyla, "Prioritizing manual test cases in traditional and rapid release environments," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–10.

[25] R. Feldt, S. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," in *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2016, pp. 223–233.

[26] C. Zhao, Y. Mu, X. Chen, J. Zhao, X. Ju, and G. Wang, "Can test input selection methods for deep neural network guarantee test diversity? a large-scale empirical study," *Information and Software Technology*, vol. 150, p. 106982, 2022.

[27] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *arXiv preprint arXiv:1207.6083*, 2012.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[29] H. Fahmy, F. Pastore, M. Bagherzadeh, and L. Briand, "Supporting deep neural network safety analysis and retraining through heatmap-based unsupervised learning," *IEEE Transactions on Reliability*, 2021.

[30] A. Zolfagharian, M. Abdellatif, L. C. Briand, M. Bagherzadeh, and R. S, "A search-based testing approach for deep reinforcement learning agents," *IEEE Transactions on Software Engineering*, vol. 49, no. 7, pp. 3715–3735, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10107813

[31] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, vol. 111, p. 104773, 2022.

[32] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2011.
[33] R. Pan, T. A. Ghaleb, and L. Briand, "Atm: Black-box test case minimization based on test code similarity and evolutionary search," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23. IEEE Press, 2023, p. 1700–1711. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00146
[34] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test case selection and prioritization using machine learning: a systematic literature review," *Empirical Softw. Engg.*, vol. 27, no. 2, mar 2022. [Online]. Available: https://doi.org/10.1007/s10664-021-10066-6
[35] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 79–90.
[36] H. Hemmati, A. Arcuri, and L. Briand, "Achieving scalable model-based testing through test case diversity," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 22, no. 1, pp. 1–42, 2013.
[37] S. Wang, S. Ali, and A. Gotlieb, "Cost-effective test suite minimization in product lines using search techniques," *Journal of Systems and Software*, vol. 103, pp. 370–391, 2015.
[38] A. Panichella, F. M. Kifetew, and P. Tonella, "Reformulating branch coverage as a many-objective optimization problem," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1–10.
[39] M. Weiss and P. Tonella, "Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study)," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 139–150.
[40] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," *Advances in neural information processing systems*, vol. 27, pp. 2069–2077, 2014.
[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 2009, pp. 248–255.
[43] S. Sharma, K. Guleria, S. Tiwari, and S. Kumar, "A deep learning based convolutional neural network model with vgg16 feature extractor for the detection of alzheimer disease using mri scans," *Measurement: Sensors*, vol. 24, p. 100506, 2022.
[44] W. Mousser and S. Ouadfel, "Deep feature extraction for pap-smear image classification: A comparative study," in *Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, 2019, pp. 6–10.
[45] T. Kaur and T. K. Gandhi, "Automated brain image classification based on vgg-16 and transfer learning," in *2019 International Conference on Information Technology (ICIT).* IEEE, 2019, pp. 94–98.
[46] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[47] Q. Yuchi, N. Wang, S. Li, Z. Yang, and B. Jiang, "A bi-objective reverse logistics network design under the emission trading scheme," *IEEE Access*, vol. 7, pp. 105 072–105 085, 2019.
[48] Y. Ma, B. Li, W. Huang, and Q. Fan, "An improved nsga-ii based on multi-task optimization for multi-uav maritime search and rescue under severe weather," *Journal of Marine Science and Engineering*, vol. 11, no. 4, p. 781, 2023.
[49] S. Verma, M. Pant, and V. Snasel, "A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems," *Ieee Access*, vol. 9, pp. 57 757–57 791, 2021.
[50] B. L. Miller and D. E. Goldberg, "Genetic algorithms, selection schemes, and the varying effects of noise," *Evol. Comput.*, vol. 4, no. 2, p. 113–131, jun 1996. [Online]. Available: https://doi.org/10.1162/evco.1996.4.2.113
[51] J. Branke, K. Deb, H. Dierolf, and M. Osswald, "Finding knees in multi-objective optimization," in *Parallel Problem Solving from Nature-PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings 8.* Springer, 2004, pp. 722–731.
[52] S. Messaoudi, A. Panichella, D. Bianculli, L. Briand, and R. Sasnauskas, "A search-based approach for accurate identification of log message formats," in *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC).* IEEE, 2018, pp. 167–16 710.
[53] L. C. Briand, Y. Labiche, and M. Shousha, "Using genetic algorithms for early schedulability analysis and stress testing in real-time systems," *Genetic Programming and Evolvable Machines*, vol. 7, no. 2, pp. 145–170, 2006.
[54] H. G. Cobb and J. J. Grefenstette, "Genetic algorithms for tracking changing environments." Naval Research Lab Washington DC, Tech. Rep., 1993.
[55] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.
[56] M. Attaoui, H. Fahmy, F. Pastore, and L. Briand, "Black-box safety analysis and retraining of dnns based on feature extraction and clustering," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, apr 2023. [Online]. Available: https://doi.org/10.1145/3550271
[57] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. Tse, "Adaptive random testing: The art of test case diversity," *Journal of Systems and Software*, vol. 83, no. 1, pp. 60–66, 2010.
[58] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.
[59] K. Alex, N. Vinod, and H. Geoffrey. The cifar-10 dataset. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html
[60] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.
[61] S. Gerasimou, H. F. Eniser, A. Sen, and A. Cakan, "Importance-driven deep learning system testing," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE).* IEEE, 2020, pp. 702–713.
[62] M. P. Wand and M. C. Jones, *Kernel smoothing.* CRC press, 1994.
[63] Z. Aghababaeyan, M. Abdellatif, M. Dadkhah, and L. Briand, "Replication package of deepgd," https://github.com/ZOE-CA/DeepGD, 2024.
[64] T. W. MacFarland, J. M. Yates, T. W. MacFarland, and J. M. Yates, "Wilcoxon matched-pairs signed-ranks test," *Introduction to Nonparametric statistics for the biological sciences using R*, pp. 133–175, 2016.
[65] A. Arrieta, "Multi-objective metamorphic follow-up test case selection for deep learning systems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1327–1335.
[66] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 851–862.
[67] Y. Dong, P. Zhang, J. Wang, S. Liu, J. Sun, J. Hao, X. Wang, L. Wang, J. Dong, and T. Dai, "An empirical study on correlation between coverage and robustness for deep neural networks," in *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS).* IEEE, 2020, pp. 73–82.
[68] S. Yan, G. Tao, X. Liu, J. Zhai, S. Ma, L. Xu, and X. Zhang, "Correlations between deep neural network model coverage criteria and model quality," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*,

2020, pp. 775–787.
[69] J. Zhou, F. Li, J. Dong, H. Zhang, and 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.
[70] Z. Gong, P. Zhong, and W. Hu, "Diversity in machine learning," *IEEE Access*, vol. 7, pp. 64 323–64 350, 2019.
[71] A. R. Cohen and P. M. Vitányi, "Normalized compression distance of multisets with applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 8, pp. 1602–1614, 2014.
[72] X. Zhan, Q. Wang, K. hao Huang, H. Xiong, D. Dou, and A. B. Chan, "A comparative survey of deep active learning," 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.13450
[73] C. Shui, F. Zhou, C. Gagné, and B. Wang, "Deep active learning: Unified and principled method for query and training," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2020, pp. 1308–1318.
[74] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, "Deep batch active learning by diverse, uncertain gradient lower bounds," *arXiv preprint arXiv:1906.03671*, 2019.

APPENDIX 1

Table 9. The average number of detected faults in each subject with test subset size $\beta = 100$

Data	Model	White-box										Black-box				
		NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	13	21	13	12	12	12	11	13	31	35	14	55	54	53	**59**
	ResNet20	10	10	11	17	16	11	17	16	42	37	11	52	56	50	**57**
MNIST	LeNet1	25	15	16	19	12	16	18	12	31	23	12	41	28	40	**42**
	LeNet5	14	11	11	15	14	12	16	14	25	28	11	30	29	30.6	**34**
Fashion	LeNet4	5	18	14	15	14	14	15	14	17	34	10	**39**	**39**	32	**39**
SVHN	LeNet5	9	4	11	12	11	11	12	11	13	19	11	43	43	44	**47**

Table 10. The average number of detected faults in each subject with test subset size $\beta = 300$

Data	Model	White-box										Black-box				
		NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	35	52	39	41	43	41	41	41	65	69	41.1	101	99	93.5	**109.6**
	ResNet20	26	30	30	30	33	30	30	33	87	80	33.6	99	102	92	**104.4**
MNIST	LeNet1	34	34	45	38	37	49	40	37	55	55	34.3	73	60	72.6	**74**
	LeNet5	29	28	27	28	25	27	25	25	46	44	20.4	54	50	52.7	**54.4**
Fashion	LeNet4	11	35	27	32	32	27	32	32	49	63	24	72	66	69	**74.7**
SVHN	LeNet5	16	19	25	23	26	25	23	26	28	53	25	85	90	85.3	**91.1**

Table 11. The average number of detected faults in each subject with test subset size $\beta = 500$

Data	Model	White-box										Black-box				
		NC		NBC			SNAC			LSA	DSA	RS	MaxP	Gini	ATS	**DeepGD**
		0	0.75	0	0.5	1	0	0.5	1							
Cifar-10	12 ConvNet	56	64	60	62	56	56	58	54	84	95	61.7	125	122	119.8	**131.5**
	ResNet20	50	50	50	51	53	50	51	53	108	112	47.8	120	126	111.5	**129.5**
MNIST	LeNet1	40	42	59	53	49	58	52	49	71	79	46.6	90	79	89.1	**92**
	LeNet5	33	36	35	39	34	36	36	34	55	55	32.3	62	61	60.3	**63**
Fashion	LeNet4	18	39	39	41	38	39	41	38	65	76	38	**99**	92	90	**99**
SVHN	LeNet5	21	32	35	32	37	35	32	37	40	68	38.2	113	112	105.8	**115.1**