4-2024

Test optimization in DNN testing: A survey

Qiang HU

Yuejun GUO

Xiaofei XIE
Singapore Management University, xfxie@smu.edu.sg

Maxime CORDY

Lei MA

See next page for additional authors

Citation

HU, Qiang; GUO, Yuejun; XIE, Xiaofei; CORDY, Maxime; MA, Lei; PAPADAKIS, Mike; and LE TRAON, Yves. Test optimization in DNN testing: A survey. (2024). *ACM Transactions on Software Engineering and Methodology*. 33, (4), 1-42.
Available at: https://ink.library.smu.edu.sg/sis_research/9094

Author

Qiang HU, Yuejun GUO, Xiaofei XIE, Maxime CORDY, Lei MA, Mike PAPADAKIS, and Yves LE TRAON

Test Optimization in DNN Testing: A Survey

QIANG HU, University of Luxembourg, Esch-sur-Alzette, Luxembourg
YUEJUN GUO, Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg
XIAOFEI XIE, Singapore Management University, Singapore, Singapore
MAXIME CORDY, University of Luxembourg, Luxembourg, Luxembourg
LEI MA, The University of Tokyo, Edmonton, Japan and University of Albert, Canada
MIKE PAPADAKIS and YVES LE TRAON, University of Luxembourg, Esch-sur-Alzette, Luxembourg

This article presents a comprehensive survey on test optimization in deep neural network (DNN) testing. Here, test optimization refers to testing with low data labeling effort. We analyzed 90 papers, including 43 from the software engineering (SE) community, 32 from the machine learning (ML) community, and 15 from other communities. Our study: (i) unifies the problems as well as terminologies associated with low-labeling cost testing, (ii) compares the distinct focal points of SE and ML communities, and (iii) reveals the pitfalls in existing literature. Furthermore, we highlight the research opportunities in this domain.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Software verification and validation;**

Additional Key Words and Phrases: Test optimization, DNN testing, low-labeling cost

ACM Reference Format:
Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Lei Ma, Mike Papadakis, and Yves Le Traon. 2024. Test Optimization in DNN Testing: A Survey. *ACM Trans. Softw. Eng. Methodol.* 33, 4, Article 111 (April 2024), 42 pages. https://doi.org/10.1145/3643678

1 INTRODUCTION

With the growth of **deep learning** (**DL**)-based applications, such as face recognition [43], self-driving car [80], and malware detection [119], assessing the reliability of deep neural networks—the cornerstone of DL systems—emerges as a pivotal concern. To this end, the most common and straightforward way is to prepare a dedicated test set and subsequently evaluate the performance of DNNs based on this curated set. Generally, this testing process is fully supervised where the test data must be labeled. However, data labeling is acknowledged as a time-intensive process demanding substantial financial resources, domain expertise, and human labor. Therefore, it presents

This work is supported by the Luxembourg National Research Funds (FNR) through CORE project C18/IS/12669767/STELLAR/LeTraon. Yuejun Guo is funded by the European Union's Horizon Research and Innovation Programme, as part of the project LAZARUS (Grant Agreement n° 101070303).

Authors' addresses: Q. Hu, M. Cordy, M. Papadakis, and Y. Le Traon, University of Luxembourg, 2 Av. de l'Universite, 4365 Esch-Belval Esch-sur-Alzette, Luxembourg; e-mails: qiang.hu@uni.lu, maxime.cordy@uni.lu, michail.papadakis@uni.lu, yves.letraon@uni.lu; Y. Guo, Luxembourg Institute of Science and Technology, 5 Av. des Hauts-Fourneaux, 4362 Esch-Belval Esch-sur-Alzette, Luxembourg; e-mail: yuejun.guo@list.lu; X. Xie, Singapore Management University, 81 Victoria St, 188065, Singapore; e-mail: xiaofei.xfxie@gmail.com; L. Ma, University of Tokyo, 7 Chome-3-1 Hongo, Bunkyo City, Tokyo 113-8654, Japan and and University of Alberta, 116 St & 85 Ave, Edmonton, AB T6G 2R3, Canada; e-mail: ma.lei@acm.org.

ACM 1049-331X/2024/04-ART111
https://doi.org/10.1145/3643678

a significant challenge for developers, especially when confronted with a large corpus of data or when engaging in continuous DNN testing, which entails the ongoing acquisition of new data on a daily basis.

To alleviate the cost associated with data labeling during the DNN testing phase, researchers explore strategies such as selecting and labeling a subset of the test data or directly leveraging the unlabeled test data to fulfill testing requirements. These activities collectively are known as **test optimization in DNN testing** [12, 48]. In these activities, methods that require labeling a portion of test data are termed *sampling-based testing* methods, while those demanding zero labeling effort are referred to as *labeling-free testing* methods.

According to the testing objective, existing methods can be categorized into four groups, *fault detection* methods, *sampling-based model retraining* methods, *model selection* methods, and *performance estimation* methods. Simply speaking, given a fixed labeling budget, *fault detection* aims to identify as many mispredicted inputs as possible, *sampling-based model retraining* tends to leverage labeled inputs for retraining models with better performance, *Model selection* focuses on finding the best model to use, and *performance estimation* strives to estimate a model's performance with a minimum bias.

Since 2017, different communities have delved into test optimization, presenting a wide range of solutions to alleviate the testing effort [41]. The fields of **software engineering** (**SE**) and **machine learning** (**ML**) have been major contributors, with 43 and 32 papers coming from these two communities, respectively. Nevertheless, no systematic reviews of such works have been conducted, particularly with regard to cross-community perspectives. To bridge this gap, in this article, we provide a comprehensive survey about test optimization in DNN testing. Specifically, we collected all related articles and divided them into three groups, *SE*, *ML*, and *Others*, based on their origin. Surprisingly, our findings reveal the inconsistent problem definitions and terminology usage across existing works. For instance, among the 46 collected works related to fault detection, we identified 14 different terminologies to denote this task, such as test selection, error detection, and bug detection. To establish clarity, we first standardize the terminologies used in each task and give each a clear definition. Subsequently, we delve into a detailed examination of each article. We compare the research focuses between SE and ML communities. Besides, we emphasize the pitfalls found in existing works to assist developers in making more effective use of existing methods. For example, we found that the widely used evaluation metric ***fault detection ratio*** (**FDR**) [18, 28, 71, 95, 112] is not precise enough. When the total fault number is greater than the labeling budget, the fault detection rate cannot reflect the ability of the method to reveal faults.

Lastly, we provide an overview of the promising research avenues in this domain. For instance, while existing labeling-free testing techniques mainly focus on vision tasks (image classification), exploring the development of novel methods for other tasks, like generation tasks propelled by **large language models** (**LLM**), presents an interesting research topic.

In the literature, there are studies that have conducted surveys on testing ML systems. Zhang et al. [120] comprehensively discussed the ML testing works where test optimization is a sub-topic named as *test prioritization and reduction*. [49, 74, 82] surveyed works that focus on quality assurance of ML systems. Riccio et al. [90] discussed existing works that focus on testing ML systems. They discussed the studied problems, their features, and their empirical evaluation in the existing works. Unlike previous surveys, our survey is the first one that targets the specific problem in ML testing – test optimization in DNN testing. Apart from reviewing existing works, our aim is to consolidate insights on this topic from different research communities.

To summarize, the main contributions of this article are as follows:

Definition. We establish a unified problem definition for test optimization in DNN testing, bringing together perspectives from both SE and ML communities.

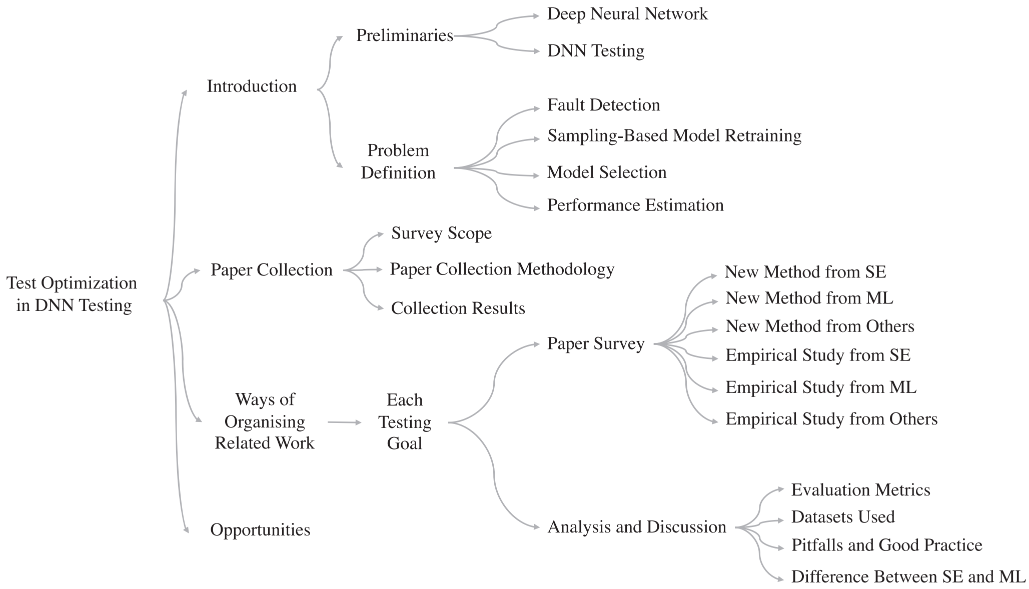


Fig. 1. article structure.

Survey. We conduct a thorough survey of test optimization in DNN testing, covering a total of 90 articles. Among these, 43 originate from the SE community, 32 come from the ML community, and the remaining 15 are from other communities.

Analysis. Via comparing the research focuses between the SE and ML communities, we reveal disparities in the intuition behind proposed methods, as well as in the used datasets and evaluation metrics. Besides, we identify pitfalls in existing works and offer guidance for good practices.

Vision. We present promising research opportunities to catalyze further advancements in this domain.

Figure 1 depicts the article structure.

2 PRELIMINARIES

In this section, we delve into the fundamental terminology of deep learning and its associated testing activity.

2.1 Deep Learning

DL is an advanced technique of artificial intelligence. Basically, DNNs are the fundamental component of DL systems. Similar to classical ML techniques, for example, linear regression [109], DNNs learn knowledge from training data and make decisions on unseen data. Essentially, DL is data-driven.

Architecture. A DNN consists of multiple layers, including input layers, hidden layers (optional), and output layers. Each layer comprises a number of weighted neurons which are the basic computation units. In general, DNN receives input information from the input layer and transfers it layer by layer to the output layer. The output layer finally produces the prediction results based on a well-designed activation function. Nowadays, different types of DNN architectures have been developed to address specific tasks, for example, the **Convolutional Neural Network** (**CNN**) [33] is famous for its impressive performance in computer vision tasks, and **Recurrent Neural Networks** (**RNNs**) [83] is the well-know architecture for handling time-series data.

Dataset. Data constitutes the pivotal element in building a high-performance DNN model. Typically, a dataset is divided into three distinct parts, training data, validation data, and test data. The training data is used to tune the parameters of DNN models, for example, weights. The validation set serves to assess the model's performance after each training epoch. Following the completion of model training, the test data is employed to report the final performance of DNN models. The independence of training, training, validation, and test data ensures that the test data remains unseen by the model, contributing to a robust evaluation of its generalization capabilities.

The data distribution of the original test set (split from the original dataset) is the same as the training data. Nevertheless, once the model is deployed in the wild, the distribution of test data may deviate from that of the training data, and the performance of the model degrades dramatically. This phenomenon is commonly referred to as the distribution shift problem. Many test optimization articles (e.g., [32, 41, 44, 107]) proposed methods to alleviate the harmfulness of different types of distribution shift. Generally, there are two types of distribution shift, (1) synthetic distribution shift and (2) natural distribution shift. *Synthetic distribution shift* comes from the computer-generated perturbation. For example, ImageNet-C [40] is a famous benchmark that provides synthetic distribution shifted datasets by adding common corruptions (e.g., brightness, fog) into original images. *Natural distribution shift* comes from unseen and unperturbed data. In this case, the data containing the natural distribution shift are usually collected from real-world scenarios. The recent natural distribution benchmark WILDs [58] provides ten datasets ranging from image to the source code domain. For example, iWildCam collects images of wild animals from specific camera traps, designating them as in-distribution data. Simultaneously, it acquires and categorizes images from other camera traps as distribution shifted data.

Tasks and evaluation measurements. DNNs are powerful tools to automate diverse tasks, and the architecture of a DNN is tailored to the specific requirements of each task. For instance, tasks may necessitate the use of different activation functions in the last layer. In this article, we primarily emphasize two common tasks, classification and regression. In a classification task, a DNN model, when presented with an input sample, identifies the category to which the input belongs. The evaluation of classification models usually employs accuracy as a metric, computing the percentage of correctly classified inputs over the total number of inputs. Different from classification tasks, in a regression task, the objective is to predict a specific value directly, for example, predict the steering angle of a self-driving car. Mean squared error which computes the average of the squares of the errors between the ground truth and labels is the commonly used metric for evaluating regression models.

2.2 DNN Testing

DNN testing is an essential activity in the DL-enabled system development process to guarantee the quality and reliability of DNN. It consists of various activities [120], such as test input generation, test adequacy evaluation, DNN debugging, and DNN repair (via sampling-based model retraining). Although there are multiple testing objectives, the primary DNN testing target is to assess the performance of DNN models, which is most often done by testing the model on the labeled test data. In this work, our targeted testing problem is the quality assessment of DNN models.

3 PROBLEM DEFINITION

Test optimization contains different tasks. In the literature, researchers use different terminologies to define these tasks as shown in Table 1. For example, for the task *select wrongly predicted test samples*, there are seven definitions in SE papers and three in ML papers. We unify the four test

Table 1. Terminologies used for Test Optimization in DNN Testing from SE and ML Communities

Task/Description	SE	ML
Fault detection/ Select wrongly predicted test samples	Test selection [7] Error detection [77] Bug detection [71] Fault revealing [76] Fault detection [1, 26, 28, 67, 95] Test prioritization [8, 26, 107, 108, 111] [4, 18, 24, 86, 116] [85, 122] Misbehaviour prediction [97] Violations detection [115] Misclassification detection [39]	Misclassification detection [3, 32, 41, 79, 93] [73, 89, 103, 126] Incorrectly classified detection [51] Failure prediction [125] Errors detection [10] Input prioritization [66]
Model selection/ Label the selected data and select the model with the best performance and use unlabeled data to select the best model	Comparative testing [84] Model selection [47]	Model ranking [98]
Sample-based model retraining/ Label the selected data and improve the model performance accordingly	Model debugging [75] Model repair [65] Model enhancement [38, 44] Model retraining [54, 56, 94, 105, 110] [5, 11, 55]	Model retraining [37]
Performance estimation/ Label the selected data and assess the performance of the model accordingly and use unlabeled data to assess the performance of the model	Accuracy estimation [12, 35, 48] Performance estimation [121] Efficient testing [69] Accuracy estimation [10]	Sample-efficient model evaluation [60] Label-efficient model evaluation [61] Performance prediction [6, 25, 29, 68] Performance estimation [13, 16, 20, 36] AutoEval [22] Errors estimation [14, 53] Generalization gap prediction [52] Predictive Uncertainty Estimation Accuracy predicting [19, 34, 57]

optimization-related tasks as fault detection, model selection, sample-based model retraining, and performance estimation. Table 2 lists the notations employed to represent basic concepts when formalizing the four tasks.

3.1 Fault Detection

The purpose of the fault detection task is to select and label as much as possible wrongly predicted data within the labeling budget. Here, for the classification task, wrongly predicted refers to misclassification. For the regression task, if the difference between the predicted value and the ground-truth value is greater than a threshold, we say the test case is wrongly predicted. In other words, fault detection is to select a subset of test data that DL models perform the worst on. We divide fault detection methods into two types based on their design construction, (1) output-based fault detection method which directly utilizes the outputs from DNN models to identify faults, and (2) learning-based fault detection method that uses other classifiers to distinguish the correctly predicted data and faults. Figure 2 depicts the common processes of output-based fault detection.

Table 2. List of Notations

Notation	Description
$\mathcal{X}$	the input space
$\mathcal{Y}$	the label space
$M : \mathcal{X} \rightarrow \mathcal{Y}$	a DNN model
$(x, y) \in \mathcal{X} \times \mathcal{Y}$	an input x with its ground truth label y
$y' = M(x)$	the label predicted by M for x
$p_i(x)$	the predicted probability of x belonging to the ith class
$X_{train} \subseteq \mathcal{X}$	a set of training inputs
$Y_{train} \subseteq \mathcal{Y}$	a set of training labels
$X_{test} \subseteq \mathcal{X}$	a set of test inputs
$\widetilde{X} \subseteq X_{test}$	a subset from X_{test}
$\widehat{X} \subseteq X_{test}/\widetilde{X}$	a set of inputs of X_{test} after removing $\widetilde{X}$
$\varrho(M, X, Y)$	a function measuring the performance of M when predicting the labels Y for the input set X

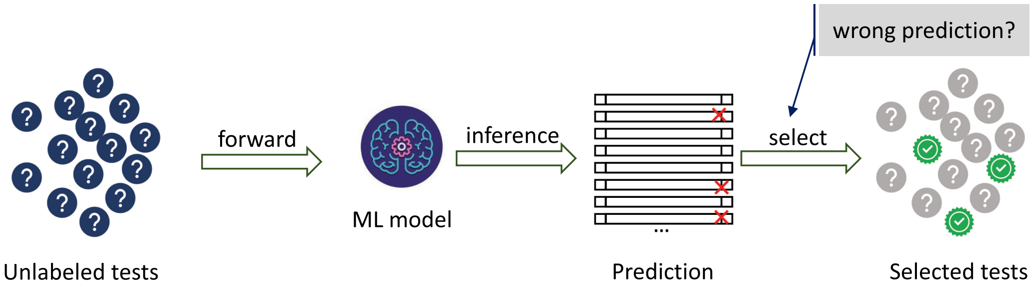


Fig. 2. Overview of output-based fault detection pipeline.

Fault Detection. Given a DNN M, an unlabeled test set X_{test}, and a labeling budget $Budget$, fault detection is the problem of selecting a subset $\widetilde{X}$ of X_{test} such that $|\widetilde{X}| = Budget$ and $\widetilde{X} = \arg\min_{X_i \subseteq X_{test}} \varrho(M, X_i, Y_i)$, where Y_i are the labels corresponding to X_i.

3.2 Sampling-Based Model Retraining

The purpose of this task is to find and label the budget number of test data, and these labeled data can improve the model performance the most by retraining. Different from training a model from scratch, retraining starts with a pre-trained model and fine-tunes it for a few more epochs. Following the guidance from [44], model retraining uses the combination of original training data and newly collected data. Some works showed that the detected faults can be used to achieve the goal of this task (e.g., [26, 71]), and some works proposed new methods specifically for this task (e.g., [44, 94]). Figure 3 shows the output-based fault detection-driven model retraining.

Sampling-based model retraining. Given a DNN M trained over a training set (X_{train}, Y_{train}), an unlabeled test set X_{test}, and a labeling budget $Budget$, sampling-based model retraining is the problem of selecting a subset $\widetilde{X}$ of X_{test} and getting corresponding labels $\widetilde{Y}$ such that $|\widetilde{X}| \leq Budget$ and $\widetilde{X} = \arg\max_{X_i \subseteq X_{test}} \varrho(M', \widehat{X}_i, \widehat{Y}_i)$, where $\widehat{Y}_i$ are the labels corresponding to $\widehat{X}_i$ and M' is M retrained with $(X_{train}, Y_{train}) \cup (\widetilde{X}, \widetilde{Y})$.

3.3 Model Selection

The purpose of model selection is to rank candidate models based on their performance, and then choose the best model for the usage. Model selection techniques can be sampling-based or

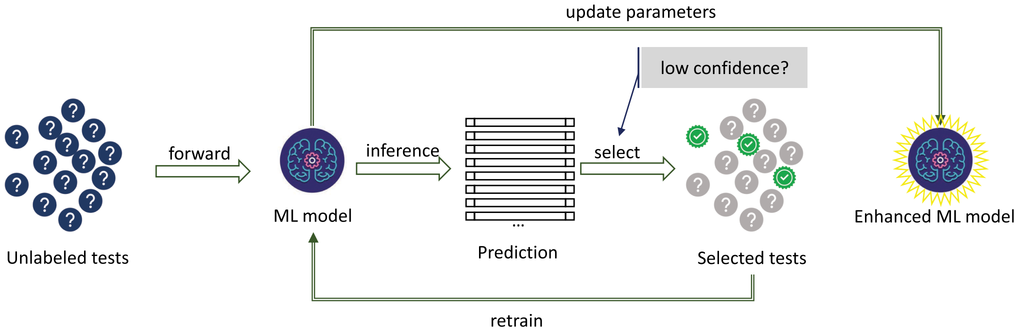


Fig. 3. Overview of sampling-based model retraining pipeline.

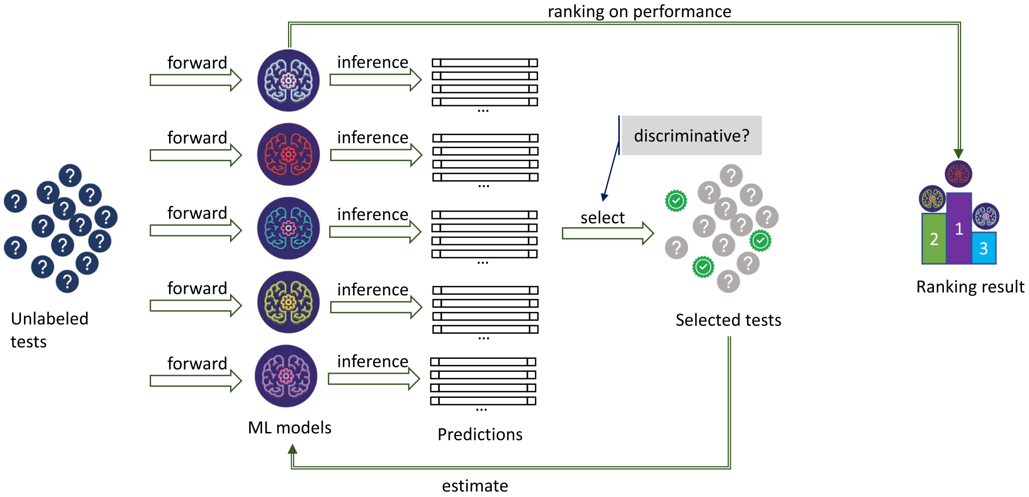


Fig. 4. Overview of sampling-based model selection pipeline.

labeling-free. For the sampling-based model selection, a budget number of test data is labeled to rank models. For the labeling-free model selection, the unlabeled dataset is used directly for the ranking. Figure 4 presents the workflow of sampling-based model selection.

Model Selection. Given a set of DNN models $\{M_i\}$, an unlabeled test set X_{test}, and a labeling budget $Budget$, model selection is the problem of selecting a subset $\widetilde{X}$ of X_{test} and getting corresponding labels $\widetilde{Y}$ such that $|\widetilde{X}| \leq Budget$ and $max_i \varrho(M_i, \widetilde{X}, \widetilde{Y}) = \underset{i}{max} \varrho(M_i, X_{test}, Y_{test})$, where $\widetilde{Y}$ and Y_{test} are the labels corresponding to $\widetilde{X}$ and X_{test}, respectively.

3.4 Performance Estimation

The purpose of performance estimation is to assess the model performance on the unlabeled dataset. The related techniques can be also divided into sampling-based and labeling-free two types. For the sampling-based methods, the budget number of data is labeled and the performance of the model is measured only using these labeled ones. For the labeling-free methods, the performance can be estimated according to the unlabeled data. Figure 5 shows the pipeline of sampling-based performance estimation.

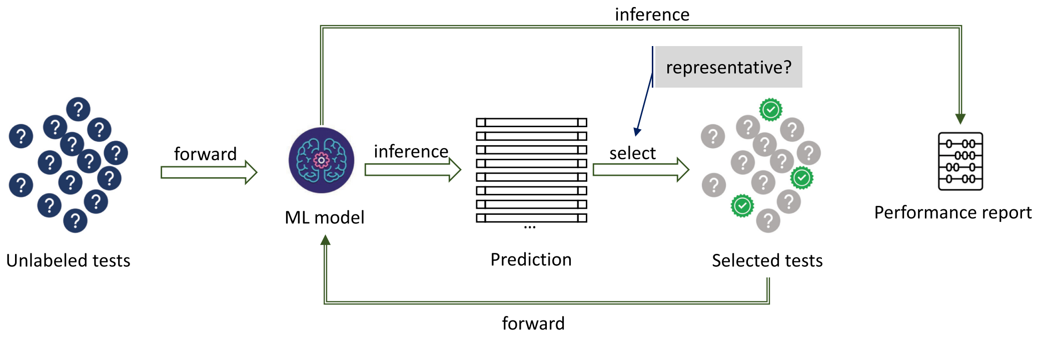


Fig. 5. Overview of sampling-based performance estimation pipeline.

Performance Estimation. Given a DNN M, an unlabeled test set X_{test}, and a labeling budget $Budget$, performance estimation the problem of selecting a subset $\widetilde{X}$ of X_{test} and getting corresponding labels $\widetilde{Y}$ such that $|\widetilde{X}| \leq Budget$ and $\widetilde{X} = \arg\min_{X_i \subseteq X_{test}} |\varrho(M, X_i, Y_i) - \varrho(M, X_{test}, Ytest)|$, where Y_i and Y_{test} are the labels corresponding to X_i and X_{test}, respectively.

4 PAPER COLLECTION

4.1 Survey Scope

Our survey focuses on the DNN test optimization problem, which is a sub-task in ML testing [120]. Here, *optimization* indicates that the testing methods aim to reduce the testing effort, more precisely, the data labeling effort. As mentioned in Section 3, test optimization has four target activities, fault detection, model selection, sampling-based model retraining, and performance estimation. From the perspective of how much effort can be reduced, the test optimization methods can be divided into two groups, labeling-free methods and sampling-based methods.

We use the following three selection criteria to collect papers. A paper must satisfy at least one of the criteria to be included in our survey.

(1) The paper proposes a new technique that targets at least one test optimization activity.
(2) The paper empirically studies/analyzes existing test optimization techniques.
(3) The paper introduces benchmarks, datasets, or criteria that are specifically designed for test optimization activities.

Some papers, for example, [41, 126], target our defined test optimization activity, fault detection, but do not mention their purpose of reducing the labeling effort are also considered in our survey. Besides, some papers, for example, [72, 118] predict the out-of-distribution errors are also categorized as fault detection. Since out-of-distribution error is one type of fault. However, works that only consider detecting out-of-distribution samples, for example, [62], are not included in our survey since these samples can be correctly predicted data or faults. Moreover, we only consider natural errors but not anomaly threats, thus do not include the adversarial detection methods in our work. We also found multiple papers [15, 59, 78] that study the test optimization problems but focus on classical ML classifiers instead of DNNs. These papers are excluded from our survey since we only focus on DNNs.

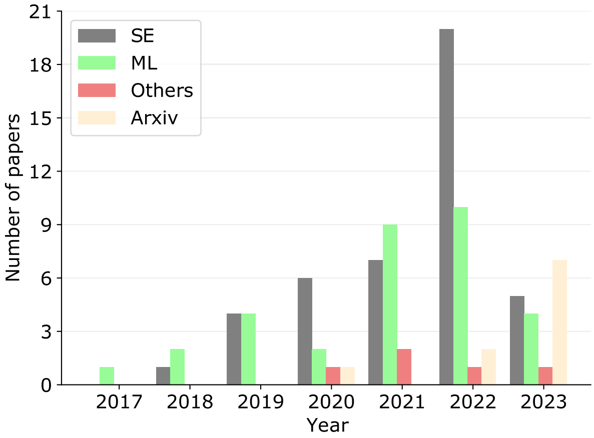


Fig. 6. Trend in the number of papers with year.

4.2 Paper Collection Methodology

We search papers from public databases including DBLP,[1] Arxiv,[2] and Google Scholar.[3] First, we use keywords to search papers from DBLP and Arxiv. Here, we only consider if the title of the paper has these keywords. There will be too many irrelevant papers (more than 10,000) if we use keywords to search on Google Scholar or search in abstracts. Then, we use citation-based search [90], also called snowballing to check the missing papers. In this step, we search for papers from Google Scholar and consider (1) the references in the collected papers, and (2) the papers that cite the collected papers. To conduct the first step, we design the following keywords. For the sampling-based model retraining task, we do not design specific keywords since they are often accompanied by the fault detection task.

— deep learning | neural network + test select
— deep learning | neural network + misclassification detect
— deep learning | neural network + failure predict
— deep learning | neural network + performance/accuracy estimation
— deep learning | neural network + test prioritization
— deep learning | neural network + efficient model evaluation

Besides, we double-check the papers with keywords DL | neural network + terminologies that appear in Table 1. We found using the listed keywords above is sufficient to find all relevant papers.

4.3 Collection Results

Table 3 presents the results of our paper collection. In total, our survey includes 90 papers related to test optimization in DNN testing up until July 29th, 2023. Figure 6 illustrates the trend of papers from 2017 to 2023, which shows that the ML community has taken the lead in studying this problem, while the SE community has focused more on this field in recent years.

Additionally, we analyze the distribution of collected papers from the perspectives of (1) the number of labeled data required for testing DNNs, for example, sampling-based testing and labeling-free testing, and (2) the goals focused by the papers. Interestingly, only two papers from

[1] https://dblp.org/
[2] https://arxiv.org/
[3] https://scholar.google.com/

Table 3. Paper Collection Results

Keywords	**Collected**	**Relevant**
deep learning test select	9	5
deep learning misclassification detect	5	1
deep learning failure predict	59	0
deep learning performance/accuracy estimation	52	2
deep learning test prioritization	5	3
deep learning efficient model evaluation	1	0
neural network test select	23	10
neural network misclassification detect	12	2
neural network failure predict	90	0
neural network performance/accuracy estimation	124	2
neural network test prioritization	19	9
neural network efficient model evaluation	3	1
In total	401	34
Query	35	
Snowball	55	
In total	**90**	

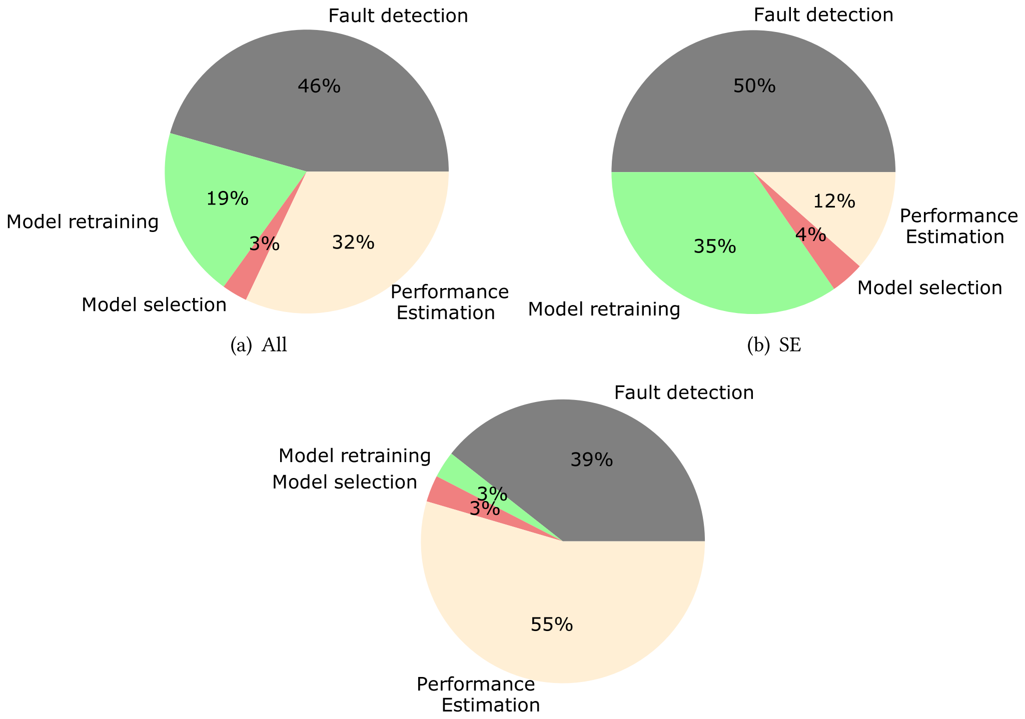
Collected: initial number of papers found on DBLP and Arxiv. **Relevant**: the number of papers satisfying the selection criteria after manual checking. **Query**: the total number of papers collected from DBLP and Arxiv. **Snowball**: the number of papers by analyzing the references of existing collected papers.

the SE community target labeling-free testing of DNNs, the other 41 papers are all about sampling-based testing of DNNs. By contrast, for the papers from the ML community, 20 are about labeling-free testing of DNNs, and the others are about sampling-based testing. The SE community works more on sampling-based testing while the ML community works more on labeling-free testing of DNNs.

Next, we investigate the distribution of the goals of existing works. Since some work targets more than one goal, for example, [26] studies both fault detection and sampling-based model retraining tasks, the total number of goals is not the same as the number of papers. Figure 7 depicts the visualized results. Figure 7(a) shows the distribution of all papers. We can see that fault detection is the most prominent goal among the four goals, while model selection gained the least attention from existing works. Interestingly, we found that there is a big difference between the task distributions from SE (Figure 7(b)) and ML (Figure 7(c)) communities. Fault detection is the most studied task in the SE community, and mode retraining is the second one. However, for the ML community, performance estimation is the most studied task, and model retraining is the least one. Many works from the SE community are inspired by test selection, a conventional SE field, and to study the fault detection problem in DNNs, for example, [26, 28, 44, 71, 84, 91, 105, 107]. Thus, they focus more on sampling-based testing.

5 FAULT DETECTION

This section summarizes papers that target the fault detection goal. First, we introduce the details of each work. The existing works can be divided into two types, the first type proposed new methods for fault detection, and the second type empirically studied the effectiveness of existing methods. We introduce works in the order of their types and the communities they were published. Then, we discuss the evaluation metrics and the datasets used in these works. Moreover, we analyze the pitfalls that exist in existing works and provide suggestions for good practice. Finally, we compare the research focuses from different communities.



(a) All

(b) SE

(c) ML

Fig. 7. Distribution of studied tasks from different communities.

Table 4. Summary of Fault Detection Papers

Community	Methodology	References
SE	new output-based method	[1, 4, 7, 24, 26, 28, 35, 54, 55, 67, 71, 77, 100, 101, 106, 108, 115, 116, 122]
	new learning-based method	[18, 39, 97, 104, 107]
	empirical study	[8, 76, 85, 95, 111]
ML	new output-based method	[10, 41, 51, 73, 79, 107]
	new learning-based method	[3, 32, 66, 89, 93, 126]
	empirical study	[103, 125]
Others	new output-based	[2, 63, 112, 117]
	new learning-based	[9]
	empirical study	[46]

Others refer to other communities and Arxiv.

5.1 Article Survey

We categorize the fault detection methods into output-based methods and leaning-based methods. Output-based methods distinguish faults only based on the outputs of layers (can be intermediate or last layer), while leaning-based methods utilize learners, for example, regression models, to train a model to predict the correctness of tests. Table 4 summarizes the category of each article.

5.1.1 New Method from SE.

5.1.1.1 Output-Based Methods. In 2018, Kim et al. [54] proposed the **surprise adequacy** (**SA**) criteria for testing of DNNs systems, called SADL, which is a very early work that introduced test selection for DNNs. SADL first extracts the intermediate outputs from DNNs of test data and training data as features. Then, it measures the SA according to the dissimilarity between these

Table 5. Uncertainty Scores

No.	Method	Equation	Description
1	DeepGini [26]	$1 - \sum_{i=1}^{N} (p_i(x))^2$	–
2	Entropy [8]	$-\sum_{i=1}^{N} p_i(x) \log p_i(x)$	–
3	Margin [52]	$p_k(x) - p_j(x)$	p_k: Maximum probability p_j: Second maximum probability
4	Entropy Dropout [8]	$\frac{1}{T} \sum_{i=1}^{T} \text{Entropy}(M, x)$	T: Dropout repetition times
5	MaxP [76]	$\underset{i=1:N}{\arg\max}(p_i(x))$	–
6	Dropout Variance [76]	$\frac{1}{T} \sum_{i=1}^{T} Var(x)$	T: Dropout repetition times
7	Weighted Variance [76]	$\frac{Dropout\ Variance}{Maxp}$	$MaxP$: Equation (5) $Dropout\ Variance$: Equation (6)
8	Kullback-Leibler [76]	$\sum_{i=1}^{N} H_i \ln \frac{H_i}{Q_i}$	H: normalized frequencies of class predictions from dropouts Q: $\frac{1}{N}$
9	Variation Ratio [95]	$1 - \frac{1}{T} \sum_{i=1}^{T} y_i' = l_{max}$	T: Dropout repetition times l_{max}: The mode of the predicted label
10	Variation Ratio for Original [95]	$1 - \frac{1}{T} \sum_{i=1}^{T} y_i' = l_{ori}$	T: Dropout repetition times l_{ori}: Predicted label by the original model (without dropout)
11	Mutual Information [95]	$Entropy(M, x) + \frac{1}{T} \sum_{i=1}^{T} Entropy(M', x)$	T: Dropout repetition times M: Original model M': Surrogate model

two features. Two measurements have been proposed in the original work, **likelihood-based surprise adequacy (LSA)** and **distance-based surprise adequacy (DSA)**. LSA uses kernel density estimation to calculate the distance, while DSA uses Euclidean distance directly. Even though the authors did not state that SADL can be used to detect the faults of DNNs, they evaluated the relation between the accuracy and the SA scores of selected test inputs. This evaluation indicates that SADL can reveal the faults of DNNs. In their extension version [55], the authors proposed a variant SA, **Mahalanobis Distance based Surprise Adequacy (MDSA)**, and evaluated SA criteria on complex models trained on a large dataset.

Another previous work DeepGini [26] was proposed by Feng et al. DeepGini is an output probability-based test prioritization technique. After obtaining the output of the test input, the Gini score is calculated by Equation (1) in Table 5. The input is more likely a fault if it has a higher Gini score. The authors demonstrated that DeepGini has a powerful ability to reveal DNN faults.

Wang et al. [106] proposed to leverage the neural coverage to rank the test inputs and find the faults. Concretely, given the unlabeled data pool, it iteratively selects a subset that has the same coverage score as the whole unlabeled set and removes the selected set from the pool. Finally, the remaining data in the pool are the ones that have different activation distributions from the whole set and are regarded as faults. Similar to [106], Yan et al. tried to use activation patterns of classes to distinguish if the output of one input is reliable [116]. Concretely, given all the training data of each class, the authors computed the lower bound and upper bound of activation values of each neuron as the pattern of this class. Then, given the test data and their predicted label, the authors compared their activation information with the activation pattern of the corresponding class. The priority score is finally calculated by the number of neurons that have greater activation values than a threshold plus the number of neurons that have smaller activation values than a threshold, divided by the total number of neurons. A greater priority score indicates that the prediction of this input is more reliable.

Guerriero et al. introduced the **adaptive test selection** (**ATS**) method DeepEST [35] for fault detection. It randomly selects one input sample first and then uses two selection methods, **simple random sampling** (**SRS**) and **weight-based sampling** (**WBS**) to add more test inputs to the selected pool. Thus, the key component of DeepEST is the WBS method. Roughly speaking, WBS assigns a weight to each input based on its distance to the labeled data divided by the total distance between all unlabeled data and labeled data. Here, the distance is defined by auxiliary variables which are the prediction confidence, distance based on SA, and the combination of confidence and SA distance. Another ATS [28] method was proposed by Gao et al. to select diverse faults from the unlabeled dataset. In this work, the output vectors are used to represent the model behaviors and measure the diversity of test inputs. ATS first projects the output vectors (top-3 maximum vectors are considered in ATS) to a space plane and then calculates the coverage of each data on this plane. After that, the difference between the coverage of a single data and the whole candidate set is utilized as an identifier to distinguish the faults and correctly predicted test data. Compared to previous works, ATS can select more diverse faults that range from different classes.

Inspired by the metamorphic testing in SE, Xie et al. proposed a diversity-guided method to detect faults [115]. The key idea is to rank metamorphic test case pairs (MPs) based on their ability to reveal violations of DNNs, Specifically, it chooses an internal layer L in the DNN model and splits this model into two parts, (1) the first part consists of layers from the input layer to L, (2) the second part consists of layers from L layer to the output layer. Then, MPs are prioritized based on the diversity of outputs from the first part. Here, the authors tried different distribution discrepancy methods to compute the output diversity, for example, **Kullback-Leibler** (**KL**) divergence. Finally, the sorted MPs are fed to the second part of the model to check the output consistency for fault detection.

Ma et al. [77] proposed to use the prediction difference between the DNN model and its subspecialized models to select fault data. Specifically, a series of models (called subspecialized models) have been trained to classify one single class. Here, each subspecialized model follows the same architecture as the original model. Then, the inputs are selected if they have high output discrimination between the original and subspecialized models. Similar to [77] and inspired by mutation testing in SE, Wei et al. proposed the **Efficient Mutation Analysis for Prioritization** (**EffiMAP**) [108] for fault detection. EffiMAP has three components, Generator, Tracer, and Estimator. Generator aims at generating fault-revealing models and input mutants. Specifically, it selects model mutants with higher killing scores and uses an autoencoder to generate input mutants. This autoencoder is iteratively updated with the generated diverse inputs. Besides, Tracer collects trace information for the given inputs. Three trace features have been considered by EffiMAP, feature map, proportion of activated neurons, and entropy of outputs. Finally, EffiMAP uses XGBoost to learn the above two types of features for fault prediction.

To further enhance the uncertainty-based methods, Li et al. proposed the distance-based **dynamic random testing with prioritization** (**D-DRT-P**) [67]. D-DRT-P first extracts features of inputs and clusters them into different subdomains. Then, D-DRT-P borrows uncertainty metrics to assign prioritization scores to each data in each subdomain. Finally, the input with the highest prioritization is selected. At the same time, if the selected input is a fault, the selection probability for its subdomain is increased. In this way, the importance of each subdomain dynamically changes and there is more chance to detect faults. Bao et al. introduced their approach **Nearest Neighbor Smoothing** (**NNS**) based test case selection method [7] by calculating the uncertainty score on the input and its neighbors. Specifically, NNS first extracted the representation of inputs using the outputs of inner layers from the model. Then, the cosine distance between each pair of inputs is computed using these representations. Inputs that are close to each other are put in the same group and their predictions are used for output probability smoothing by label smoothing technique [100]. Finally, NNS used the smoothed outputs to compute the uncertainty score for prioritization.

Tao et al. proposed TPFL [101], a test prioritization method based on fault location. Firstly, TPFL uses all the training data to collect the activation information of each neuron. Different from the classical way to set the activation threshold, TPFL sets the threshold of each neuron by the sum of its average and standard deviation of outputs. Then, TPFL indicates the suspicious neurons if they have a higher frequency activated by test inputs where the DNN makes incorrect decisions and a lower frequency activated by test inputs where the DNN makes correct decisions. Finally, if the new unlabeled data activate more suspicious neurons, they are more likely faults.

Al-Qadasi et al. proposed DeepAbstraction [4], which leverages runtime monitors to detect faults. Specifically, DeepAbstraction first extracts the features of training data to build the abstraction box. It considers the vectors extracted from the penultimate layer and the predicted classes as the features. In the abstraction box, the inputs are divided into two groups, correct inputs and incorrect inputs. Then, given new unlabeled data, DeepAbstraction checks whether their features belong to these two groups. If not, DeepAbstraction assigns these data to the third group—uncertain group. Finally, using uncertainty metrics, DeepAbstraction prioritizes the data from each group in order of incorrect, uncertain, and correct.

Different from prior works that primarily focused on computer vision tasks and **feed-forward neural networks** (**FNNs**), Liu et al. proposed the first method DeepState [71] to specifically target the fault detection problem of RNNs. DeepState extracts the predictions from the internal output state and builds a label sequence for each input. Then, DeepState defines a **changing rate** (**CR**) to measure the uncertainty of inputs based on the label changes of the collected sequence. A higher CR indicates a more uncertain input. DeepState uses the **changing trend** (**CT**) metric to help remove the redundant inputs with the same CRs. CT measures how two label sequences are different. In this way, the selected inputs are both uncertain and diverse.

Aghababaeyan et al. proposed a black-box method for fault detection [1]. Specifically, it first used VGG16 models to extract the features of mispredicted inputs from training and test sets. Then, it used **Uniform Manifold Approximation and Projection** (**UMAP**) method to reduce the dimension of collected features. After that, the **hierarchical density-based spatial clustering of applications with noise** (**HDBSCAN**) was used to cluster the faults into different groups. Finally, the authors found that, given new inputs, using inputs within one cluster to retrain the model can help fix the model with respect to that fault represented by this cluster.

To tackle more practical problems, Deng et al. proposed the **Scenario-based Test Reduction and Prioritization** (**STRaP**)[24] to detect faults in self-driving systems efficiently. Firstly, given a driving recording, STRaP converts messages in each time frame into vectors based on a driving scene schema. Here, the authors defined multiple driving scene schemas. for example, *Is there any pedestrian crossing the road? 1 for yes and 0 for no.* Then, STRaP slices the vectors transformed from the last step into segments based on their similarity. Finally, it prioritizes the vectors based on their coverage and the rarity of driving scene features.

Lastly, Zheng et al. proposed CertPri [122], a certifiable method for fault detection. The intuition behind CertPri is that a significant difference exists in the movement cost of correctly and incorrectly predicted test inputs. Here, the movement cost means the cost of moving the input to the class center. Interestingly, the movement cost of correctly predicted inputs is significantly higher than the cost of incorrectly predicted inputs. Thus, this cost can be used to distinguish the faults.

5.1.1.2 Learning-Based Methods. Wang et al. proposed Dissector [104] to detect unexpected conditions produced by faults. The intuition behind Dissector is that DNNs should have increasing confidence in the normal input (correctly predicted input) from the input layer to the output layer.

Based on this intuition, Dissector slices the DNN model into multiple sub-models and adds an output layer to each sub-model. Then, Dissector collects the output (so-called snapshot) from each sub-model as the sequence of confidence by the original DNN model. Finally, the authors defined an SVscore to measure the snapshot validity and computed the final profile validity score of inputs by weighting all SVscores. A higher profile validity score indicates the input is more likely a normal one.

Based on mutation testing techniques, Wang et al. proposed the **PRioritizing test inputs via Intelligent Mutation Analysis** (**PRIMA**) [107] for fault detection. The intuition behind PRIMA is that the faults are near the decision boundary and, thus, more sensitive to the perturbation. Based on this intuition, the authors designed two types of mutation roles, model mutation and input mutation. Then, given input and these mutation rules, PRIMA collects the multiple features based on the killing information and leverages the learning-to-rank strategy to train a ranking model for fault prediction. Here, the famous XGBoost ranking algorithm has been used for building the ranking model.

He et al. proposed the **Parallel Signal Routing Paths** (**PSRP**) [39] to identify misclassified samples. PSRP contains three components, feature space compression, extraction of PSRP, and misclassified sample detection. In the first step, PSRP compresses the input by computing the mean value of a two-dimensional tensor produced by a convolution kernel. Then, PSRP trains an SVM model to solve a binary classification task for fault detection. The training data is the trace of compressed data from the first CNN layer to the last one.

To tackle other types of datasets, Stocco et al. proposed SelfOracle [97], an online misbehavior prediction method for self-driving cars. SelfOracle first uses an autoencoder to reconstruct the training inputs of self-driving cars. Then, it computes the mean pixel-wise squared error as the reconstruction error. After that, SelfOracle uses the maximum likelihood estimation to fit the sum of the squares of pixel-wise errors and estimates the parameters of a Gamma distribution. Finally, by setting a threshold, the Gamma distribution will be used to predict the probability of misbehavior of the unseen inputs. As stated by the authors, SelfOracle can be used for online misbehavior prediction and time-aware anomaly score prediction. Dang et al. proposed the GNN-oriented Test Prioritization (GraphPrior) [18], a fault detection method specifically designed for graph data. GraphPrior defined 10 rules for mutating GNN models, for example, adding self-loops to the nodes. After the model mutation, inputs are prioritized based on their killing score on this mutant, which means the input is more likely a fault if mutants have higher disagreements on the input.

5.1.2 *New Method from ML.*

5.1.2.1 Output-Based Methods. In 2017, Dan et al. introduced the well-known baseline for fault detection, maximum softmax probabilities based detection [41]. This work showed that it is promising to detect out-of-distribution data and misclassified faults by simply using the maximum softmax probabilities of outputs.

To enhance uncertainty-based methods, Malinin et al. proposed the **Dirichlet Prior Networks** (**DPNs**) [79] to model the predictive uncertainty of DNNs. Simply speaking, the DPN model directly parameterized the Dirichlet distribution as a prior for predicting the classification distribution on the probability simplex. The most important part is the loss function of DPNs which was defined as the KL divergence between the model and a sharp Dirichlet distribution on the appropriate class for in-distribution data, plus the model and a flat Dirichlet distribution for out-of-distribution data. Lastly, the existing uncertainty measurements, max probability and entropy were used to detect the faults based on the output of DPNs. Importantly, this work summarized the different types of uncertainty, model uncertainty, data uncertainty and distributional uncertainty which were confused by previous works.

Similar to mutation-based methods from SE [107], Chen et al. proposed a framework to use an ensemble of models to identify faults [10]. Specifically, the disagreement between the original model and the majority voting of the ensemble model was used as the indicator for fault detection. That means if the ensemble disagrees with the original model, the input has a higher ability to be a fault. Most importantly, to improve the performance of fault detection, the authors built the ensemble models iteratively and added the selected faults at each iteration to the training data, and then performed self-training in the ensemble model.

Lust et al. proposed a **Gradient's Norm** (**GraN**) based method [73] to detect faults. GraN contains three steps, input pre-processing, feature extraction, and feature processing. In the first step, given an input image, GraN utilized image smoothing techniques to generate a new input. Then, GraN fed the smoothed image with the predicted label of the original image into the DNN model to calculate the gradients via backpropagation. In the last step, a logistic regression network was used to learn the connection between the gradient features extracted from the last step and the correctness of the inputs. The output of the regression model indicates the likelihood of the correctness of inputs.

Instead of directly using the prediction score to detect faults, Jiang et al. proposed the *trust score* [51] to identify the correctness of unlabeled test data. To do so, firstly, the authors removed the data that has a low density from the training set for each class. Here, any data representation and distance methods can be used for this process. After that, the *trust score* was defined to measure the reliability of inputs. Given an input, *Trust score*was calculated by the ratio between its distance from this input to the nearest class different from the predicted class and the distance to its predicted class. A lower *trust score* indicates that this input is more likely a fault.

5.1.2.2 Learning-Based Methods. Aigrain et al. proposed Introspection-Net [3] to use a 3-layer regression NN to predict the correctness of inputs. Introspection-Net is a binary classification model that takes the output logits from the original DNNs as inputs and produces a confidence value for the inputs that is, whether the classification is correct (output value of (1) or not (output value of 0). The experiments showed that Introspection-Net accompanied by adversarial training and data augmentation has a competitive fault detection ability with the Trust Score approach [51] and Softmax Baseline [41].

Li et al. proposed TestRank [66] to rank test inputs based on their likelihood of being a failure. Specifically, TestRank extracted two types of features from inputs, the output from the logits layer and the graph information that represents the distance of this input (here, cosine distance was used for the computation) to others. After that, a GNN model was used to learn the graph information and predict the learned contextual attributes of the data. Finally, given the correctness label (e.g., 0 for incorrect, 1 for correct) of inputs, TestRank utilized a simple binary classification model to learn the output information and the contextual attributes and predict the correctness of inputs.

Granese et al. proposed DOCTOR [32], which defined two types of discriminators to determine whether the input is a fault or not. The first one is based on the sum of squared softmax probabilities, and the second one is computed by the predicted model for the posterior class probability. Interestingly, the authors considered two scenarios for the evaluation, **Totally Black Box** (**TBB**) where only the predictions are available and **Partially Black Box** (**PBB**) where gradient information is allowed. In the PBB situation, the authors found adding a small perturbation brought advantages for fault detection.

Sensoy et al. proposed the risk-calibrated evidential classifiers [93] whose outputs are more sensitive to the faults. The purpose of such classifiers is to force the faults to be biased toward less risky categories to increase the fault detection rate. To do so, firstly, the authors utilized the Dirichlet Distributions to reform the outputs and therefore, to quantify the uncertainty of inputs against DNNs. After that, an evidential deep classifier was trained where the activation function of the classifier was changed to the exponential function for predicting the Dirichlet distribution for each sample. Finally, uncertainty methods were used on the evidential deep classifier for fault detection.

Qiu et al. proposed the **Residual-based Error Detection** (**RED**) [89] to enhance error fault score based on the original maximum class probability. Specifically, RED first assigned a target detection score to each training input according to whether it is correctly classified by the base model, that is, 1 for correct and 0 for incorrect. Then, it computed the residual between the target score and the maximum class probability produced by the original model. After that, a Residual prediction with an Input/Output kernel (RIO) model is trained to predict this residual. Finally, when new inputs come, RED combines the score produced by the RIO model and the output probability produced by the original for fault detection.

Zhu et al. [126] conducted a study and found that adding **out-of-distribution** (**OOD**) detection methods, for example, Outlier Exposure, to the training process, harms the performance of output-based fault detection. Then, based on this finding, the authors proposed a method called OpenMix to help the fault detection. Specifically, OpenMix changed the OE loss in Outlier Exposure to in/OOD (ID) Mixup-based loss to increase the exposure of low-density regions. In this way, the OOD detection methods have both good OOD detection ability and fault detection ability.

5.1.3 *New Method from Others.*

5.1.3.1 Output-Based Methods. Aghababaeyan et al. introduced DeepGD [2], a search-based test selection method for detecting faults. Specifically, DeepGD considered both uncertainty scores and diversity scores of inputs to conduct test prioritization. Gini [26] score is used for uncertainty calculation and geometric diversity on the features extracted from VGG models is applied for diversity calculation. An NSGA-II algorithm is used for optimization uncertainty and diversity to assign a final score to each input for prioritization.

Yang et al. proposed PROPHET [117] to detect faults in **automated speech recognition** (**ASR**) systems. In this work, the authors introduced a new fine-grained word-level error metric for evaluating the correctness of audio-to-text tasks. By comparing the reference text and predicted text, PROPHET labeled the token as 0 (1)if it is correctly (incorrectly) predicted. Then, it trained a BERT model to predict word errors. Given the new coming data, after feeding them to the trained BERT model, the ones that have higher word errors have higher probabilities be faults.

Lee et al. proposed a neuron activation similarity-based sample selection method [63] for fault detection. The method first collected the neuron activation information for each class from the training data and then checked the activation of the test data. If the difference between the similarities of training and test data is less than the threshold, these inputs are considered to be faults.

Wu et al. propose RNNtcs [112], a test prioritization method for RNNs. Given the unlabeled test data, RNNtcs first extracted the outputs of the test inputs and utilized HDBSCAN to cluster the outputs into different groups. Then, it utilized uncertainty methods (Least confidence was used in RNNtcs) to compute uncertainty scores for outliers identified by HDBSCAN and prioritized them accordingly. If the number of outliers is less than the labeling budgets, the uncertainty scores of other inputs in each cluster (normal data) were also calculated and prioritized. After that, the hidden state CR of RNN models was computed as the second uncertainty measurement. Finally, the budget number of inputs with higher hidden state CRs was selected from the prioritized sets in the order of outliers and normal data.

5.1.3.2 Learning-Based Methods. Chen et al. proposed ActGraph [9], an activation graph-based test prioritization method. ActGraph first extracted the activation value from the last K layers and built an adjacency matrix. Then, a GNN model is used to aggregate the activation features, and an aggregation function is used to obtain the center node feature. Here, ActGraph used *Sum* () as the aggregation function. Finally, a learning-to-rank algorithm is applied to build a ranking model for test prioritization. XGBoost is applied in ActGraph as the ranking model.

5.1.4 *Empirical Study from SE.*

The very first empirical study was conducted by Byun et al. [8]. In this work, the authors explored the effectiveness of three fault detection methods, Entropy (method 2 in Table 5), uncertainty in Bayesian neural networks (method 4 in Table 5), and DSA [54]. The key finding from this empirical study is that when the DNN under test has a higher test accuracy, these methods are more effective in detecting faults. Besides, Mosin et al. compared SA, autoencoder-based, and similarity-based fault detection methods [85]. They found that SA has the most effective fault detection ability among the considered methods. However, the similarity-based method is the most efficient method which is more than 1,000 times faster than the SA method.

Ma et al. evaluated the fault detection ability of 12 methods [76]. In their work, they divided existing fault detection methods into two groups, methods from ML testing literature including coverage-guided methods, SA-based methods, and so on, and model uncertainty-based methods, such as maximum probability (method 5 in Table 5). Importantly, the authors suggested using Silhouette coefficient [92] to detect faults and also defined two new methods, Variance, and Weighted Variance (methods 6 and 7 in Table 5). They found that uncertainty-based methods have stronger fault detection ability than SA and neuron coverage-based methods. Shi et al. also empirically studied the effectiveness of test case prioritization metrics (fault detection methods) [95]. In total, 11 fault detection methods have been considered by the authors including SA methods and uncertainty-based methods. Different from the previous work, this work suggested using model mutants to replace dropout prediction for fault detection and introduced three new methods called **Variation Ratio** (**VR**), **Variation Ratio for Original** (**VRO**), and **Mutual Information** (**MI**), which are listed as methods 9, 10, and 11 in Table 5. In addition to analyzing the effectiveness of existing methods, this study also explored the impact of the test suite size, mutation operators, and the number of mutants.

More recently, Weiss et al. [111] empirically showed that simple test prioritization methods perform better than SA and neuron coverage methods in terms of fault detection. Here, *simple* methods indicate those methods only rely on the model outputs, for example, using the maximum probability to detect faults directly (method 5 in Table 5).

5.1.5 *Empirical Study from ML.*

Vazhentsev et al. specifically studied the existing fault detection methods on natural language processing [103]. Besides, two novel methods have been proposed in this work, which are based on the **Diverse Determinantal Point Process** (**DDPP**) Monte Carlo Dropout. DDPP sampled (and masked) a subset of neurons from the dropout layer where the activation values of this subset are similar to the activation values of the whole set. The authors improved the diversity of the sampled DPP masks by two methods, the first one, **determinantal point process** (**DPP**) sampled a set of âĂIJdiverseâĂİ masks that activate different sets of neurons by using the **Radial basis function** (**RBF**)-similarity matrix of mask vectors. The second one selected the masks that yield the highest **Probability variance** (**PV**) scores on the given OOD dataset. This study showed that methods based on the Mahalanobis distance and spectral normalization of a weight matrix achieved the best results in NLP, and the proposed two methods have competitive results with the best ones.

Zhu et al. [125] conducted an empirical study to show that confidence calibration methods are useless or harmful for failure prediction. They studied five calibration methods method, for example, mixup, and found that after adding these methods, the existing fault detection methods have poorer detection performance. Inspired by this finding, the authors proposed to use flat minima methods for enhancing the existing fault detection methods. Two methods have been considered in this work stochastic weight averaging and sharpness-aware minimization. The experiments demonstrated that flat minima are useful for improving the detection ability of existing methods.

5.1.6 *Empirical Study from Others.* Hu et al. [46] empirically explored the robustness of fault detection methods. Here, robustness means how good the methods are in handling uncommon test inputs. The authors designed two types of uncommon inputs that the benchmark datasets (e.g., MNIST) do not have but could appear in the wild. The first type of test is high uncertainty but correctly predicted data, and the second one is low uncertainty but wrongly predicted data. Extensive experiments demonstrated that existing fault detection methods cannot distinguish faults and the first type of test data, and cannot detect the second type of fault.

5.2 Analysis and Discussion

5.2.1 *Evaluation Metric.* Table 6 lists the evaluation metrics used for fault detection. We can see different works used different metrics and there is no uniform one. The most used metric is receiver operating characteristic, that is, 11 works used AUC-ROC for the evaluation.

5.2.2 *Datasets.* Table 7 presents the types of data used for evaluating fault detection methods from existing works. Most works (27 out of 46) only tried to detect faults in the test set (*Original test set*) that split from the original dataset. Works [7, 9, 101, 107, 122] conducted relatively more comprehensive evaluations, which considered faults in original tests, adversarial examples, and synthetic distribution shifted datasets.

5.2.3 *Pitfalls and Good Practice.* Although many works take an effort to solve fault detection problems, there are some pitfalls revealed in existing works. We analyze such pitfalls from three perspectives, datasets, evaluation metrics, and robustness issues.

- **Datasets.** *Pitfalls.* In total, 69% works only evaluate fault detection methods on the original test set. Since such a test set is split from the original dataset and follows a similar data distribution to the training data, the model typically performs well. The reported results cannot be generalized to other datasets that have different data distributions from the training dataset. *Good Practice.* When proposing or studying existing fault detection methods, the datasets used for evaluation should cover diverse data distribution – considering both in and out-of-distribution data.
- **Evaluation metrics.** *Pitfalls.* FDR and the number of faults are not precise metrics for evaluating fault detection methods. However, some of the existing works used them. In a situation where the faults number is greater than the labeling budget, FDRs and the number of faults can be 1 and the labeling budget and methods are incomparable according to these scores. *Good Practice.* We recommend not using FDR and the number of faults as the evaluation metrics. **Average percentage of fault-detection** (**APFD**) and receiver operating characteristic metrics are mature ones that should be used for the evaluation.
- **Robustness issues.** *Pitfalls.* Existing research [46] reveals that fault detection methods can be fooled by correctly classified but with high uncertainty data, and wrongly classified but with low uncertainty data. This indicates that fault detection methods were designed for normal data but uncommon tests which can also appear in the wild. *Good Practice.* In addition to the common evaluation of fault detection methods using widely used datasets, it is better to (1) evaluate the robustness of these methods using correctly classified but with high uncertainty data, and wrongly classified but with low uncertainty data, or (2) discuss the robustness limitations of the proposed methods.

Table 6. Evaluation Metrics for Fault Detection

Name	Equation	Description	Involved references
Average Percentage of Fault-Detection (APFD)	$1 - \frac{\sum_{i=1}^{k} o_i}{kn} + \frac{1}{2n}$	n: number of tests, k: number of faults, o_i: the order of the first test that reveals the ith faults	[8, 24, 26, 67, 111] [18, 85, 116]
Kendall correlation	Equation (3)	–	[76]
Spearman correlation	Equation (1)	–	[1]
True positive rate (TPR)	$\frac{TP}{TP+FN}$	TP: True positive FN: False negative	[97, 106]
False positive rate (FPR)	$\frac{FP}{FP+TN}$	FP: False positive TN: True negative	[3, 32, 39, 97, 125] [126]
Precision	$\frac{TP}{TP+FP}$	–	[39, 51, 89, 93]
F1 score	$\frac{2TP}{2TP+FP+FN}$	–	[10]
AURC	–	The area under the (empirical) RC-curve	[103, 125, 126]
AUC-PRC	–	Arear under curve. X-axis: TPR Y-axis: FPR	[3, 41, 79, 89, 97] [125]
AUC-ROC	–	Arear under curve. X-axis: Recall Y-axis: Precision	[3, 41, 79, 97, 104] [32, 73, 89, 93, 125] [126]
RAUC	–	Ratio of the area under the curve for the test input prioritization approach to the area under the curve of the ideal prioritization	[9, 101, 107, 108, 122]
Fault Detection Ratio (FDR)	$\frac{\|D_f\|}{\|D\|}$	$\|D_f\|$: number of fault, $\|D\|$: number of all tests	[18, 28, 71, 95, 112]
Number of faults	–	–	[35, 63]
Normalized-APFD (NAPFD)	$\frac{APFD - APFD_{min}}{APFD_{max} - APFD_{min}}$	$APFD_{max}$: APFD when all faults are prioritized at first $APFD_{min}$: APFD when all faults are prioritized at last	[115]
Top-K	–	The amount of segments required for finding the first fault	][24]
Test Relative Coverage (TRC)	$\frac{\|D_f\|}{min(B, D_F)}$	$\|D_f\|$: number of fault, B: labeling budgets, D_F: total number of faults	[2, 7, 46, 66]
Average Test Relative of Coverage (ATRC)	$\frac{1}{N} \sum_{1}^{N-1} TRC_i$	Average of TRC under different budget settings	[4, 66]
Word error rate (WER) Character error rate (CER)	$\frac{\|S\|+\|I\|+\|D\|}{\|W(C)\|}$	S: Substitutions, I: Insertions, D: Deletions, $W(C)$: Words or (Characters)	[117]
Fault Diversity	$\|y \rightarrow y'\|$	y: ground-truth label y': predicted label	[28, 112]
Reversed pair proportion (RPP)	$\frac{1}{n^2} \sum_{i,j=1}^{n} \|u(x_i) > u(x_j), l_i < l_j\|$	u: uncertainty metric l: loss	[103]

5.2.4 SE vs. ML. We can see both SE (26 papers) and ML (13 papers) communities contributed to the study of the problem of fault detection in DNNs. It is worth comparing the research focuses of these two different communities.

— **Proposed methods.** (1) 70% of fault detection methods proposed by the SE community are output-based, while only 45% of methods from ML are output-based. Researchers from the ML community prefer to utilize the relation between features and faults to detect faults, that is, to train another model for fault prediction. (2) Compared to works in SE, works from the ML community prefer to propose techniques to enhance existing methods, for example, [79, 126] changed the loss function of DNNs to enhance the detection ability of existing

Table 7. Types of Tests used for Fault Detection

Faults Type	Papers
Original test set + adversarial examples	[26, 95]
Original tests	[35, 67, 77, 104, 108] [1, 4, 39, 85, 86, 106, 115, 116] [2, 41, 51, 63, 85, 117] [10, 66, 79, 93, 103] [89, 125, 126]
Original tests + mutated data (e.g., image transformation)	[3, 28, 71, 112]
Mutated data (e.g., image transformation)	[24, 97]
(1) Original tests (2) Mutated data (e.g., image transformation)	[73]
(1) Original tests, (2) Original tests + mutated data (e.g., image transformation)	[46]
(1) Original test set, (2) Original test set + adversarial examples	[76]
(1) Original tests (2) Adversarial examples	[18]
(1) Original tests (2) Original tests + adversarial examples 3) mutated data (e.g., image transformation)	[9, 107, 122]
(1) Original tests (2) Original tests + adversarial examples 3) Original tests + mutated data (e.g., image transformation)	[101]
Distribution shift benchmarks (e.g., MNIST-C)	[111]
Extension set (EMNIST) of original test set (MNIST)	[8]
(1) Original tests (2) Distribution shift benchmarks, 3) Original tests + adversarial examples	[7]
(1) Original tests, (2) Original tests + OOD tests	[32]

output-based methods. (3) Interestingly, some methods from SE and ML communities are based on similar intuitions, that is, [107, 108] from the SE and [10] from the ML changed the DNN models and then utilized the corresponding output changes to detect faults. However, the *changed models* obtained from [107, 108] were based on model mutation, while from [10] are based on training ensemble models.

- **Evaluation metrics.** Researchers from the SE community rarely use receiver operating characteristics to measure the effectiveness of fault detection methods, only two works (out of 25) used TPR and FPR for the evaluation. However, almost all (12 out of 13 cases) works from the ML community evaluate their methods by using receiver operating characteristics.
- **Evaluated datasets and models.** (1) More than half of SE works (13 out of 25) evaluate fault detection methods on datasets other than the original tests. However, most of the works (10 out of 13) from ML only evaluate methods on original tests. (2) Four works from the SE community consider self-driving cars as their study objective which is a more practical task. (3) None of the SE works consider the datasets and models that are specifically designed

Table 8. Summary of Sampling-based Model Retraining Papers

Community	Methodology	References
SE	new method	[5, 8, 11, 18, 26, 28, 38, 44, 54, 55, 65, 71, 75, 94, 105, 117]
	empirical study	[56, 76, 110]
ML	new method	[37]
	empirical study	–
Others	new method	–
	empirical study	[46]

Others refer to other communities and Arxiv.

for SE tasks, for example, code generation tasks and code completion tasks. All their used datasets and models are originally provided by the ML community.

- **Research questions.** Seven works from SE study the effectiveness of using detected faults to enhance the pre-trained DNNs via retraining while none of the works from ML does that. Researchers from SE treat the fault detection process as the debug process in software development and the retraining as the patching process for fixing the bugs.

6 MODEL RETRAINING

6.1 Article Survey

Similar to the fault detection methods, we introduce sampling-based model retraining from two perspectives, works that propose new methods and works that conduct empirical studies, and follow the order of SE, ML, and others. Before that, we found many fault detection works [8, 18, 26, 28, 46, 54, 55, 71, 76, 117] from the SE community also evaluated the effectiveness of using detected faults to enhance the pre-trained models via model retraining. Since we already explained these works in the last section, we skip them here. Table 8 summarizes the type of each article.

6.1.1 New Methods from SE. The first work is MODE which proposed a differential analysis-based input selection method for debugging (via retraining) DNNs by Ma et al. [75]. MODE contains three main components. First, it selects the intermediate layer in the DNN that is important for causing under-fitting or over-fitting problems. Then, MODE builds a feature model by adding one output layer after the selected layer and uses this model to produce heat maps of benign and faulty data. Finally, MODE selects new data based on the heat maps and the characteristics of under-fitting and over-fitting, for example, for under-fitting bugs, it selects more faulty data from under-fitted classes. The authors have shown MODE has a strong ability to fix bugs in pre-trained DNNs.

Shen et al. proposed the **Multiple-Boundary Clustering and Prioritization** (**MCP**) [94] to enhance the DNN retraining. The basic idea of MCP is to select data near the decision boundary and control the diversity of the selected set. Specifically, MCP first computes the output probabilities of all test inputs. Then, MCP divides all the data into different clusters according to their top-2 predicted classes. Besides, for each data, MCP computes its priority in its belonging cluster as the ratio of the probability of the first class to the probability of the second class. Finally, MCP evenly selects inputs with high priorities from each cluster to form the final set as the output. Since MCP considers top-2 classes, the selected inputs are close to multiple (2) boundaries.

Different from other works that mainly focused on the clean accuracy of DNN models, Wang et al. introduced **Robustness-Oriented Testing** (**RobOT**) [105] to enhance the robustness of DNNs by retraining. This article designed two robustness-oriented testing metrics to guide the test selection process, **zero-order loss** (**ZOL**) and **first-order loss** (**FOL**). ZOL is calculated by the loss of the input directly, and FOL is based on the gradient of the input against the DNN model. The authors demonstrated that there is a strong correlation between the robustness of the model and the FOL score, and showed FOL is good guidance for selecting data to improve the robustness of DNNs. Besides, in the extension work of RobOT [11], the authors added both tabular and image domain data in the experiments and explored the usefulness of using RobOT to improve the fairness of DNNs.

In order to provide guidance for users to better use sampling-based retraining methods, Hu et al. [44] first empirically studied how existing methods perform on datasets with distribution shifts. They found that no method can constantly outperform others across all ranges of distribution shifts. When the candidate set contains more than 70% distribution shifted data, random selection performs better than other well-designed methods. To tackle the distribution shift problem, the authors proposed a **distribution-aware test (DAT)** selection. DAT has two key steps, firstly, it splits the candidate set into in-distribution data and OOD data based on OOD detectors. The DAT selects data that the model has low confidence in from the in-distribution set and selects diverse (diverse in terms of the predicted labels)data from the OOD set.

Li et al. proposed HybridRepair [65], a method smartly combining semi-supervised learning and sampling-based retraining to improve the performance of DNNs. HybridRepair has two main steps, (1) it uses data augmentation methods to generate data for each input and then average their predictions. After that, the averaged predictions will be used as a *groud-ture* to calculate the loss for each data. Finally, semi-supervised learning is performed to train the model. In this step, the authors also defined a local entropy metric to only select the less uncertain data for semi-supervised learning. (2) The second step tends to use fully-surprised learning on a selected subset. HybridRepair uses a feature extractor to obtain the important features of inputs and leverages these features to conduct clustering-based sample selection. In this step, the local entropy is also used for guiding the data selection (select data with higher local entropy).

Attaoui et al. proposed the **Safety Analysis based on Feature Extraction** (**SAFE**) [5] to retrain and improve DNNs. SAFE utilizes pre-trained image models to extract the features from existing faults first. Then it uses the DBSCAN algorithm to cluster the extracted features into different fault groups. After that, when new data come, engineers extract their features, and then select and label the data that are close to the fault clusters. Finally, the selected data are used to retrain the DNN models.

Hao et al. proposed the **Multiple-Objective Optimization-Based Test Input Selection** (**MOTS**) [38] to consider both uncertainty and diversity in the data selection process. For the uncertainty measurement, MOTS considers the entropy score of output probabilities. On the other hand, the Euclidean distance between two inputs is used for the diversity score. A multiple-objective optimization algorithm NSGA-II is used to store the Pareto optimal solution. Finally, MOTS selects the samples based on the aforementioned solution.

6.1.2 *New Methods from ML.*

Guo et al. proposed the **density-based robust sampling with entropy** (DRE) [37]. **DRE** calculated the entropy score from the predicted probabilities of each data first. Then, it split the data space into multiple intervals and mapped each input to the interval according to its entropy score. Finally, DRE randomly selected inputs from each interval. The number of selected inputs from each interval was determined by the probability density function.

6.1.3 *Empirical Study from SE.*

Empirical studies on sampling-based model retraining mainly focus on SA. Kim et al. showed how surprise SA performed on the industrial-level datasets [56]. They took the self-driving car as the use case and explored the effectiveness of SA in guiding DNN retraining. The study results demonstrated that SA is a promising metric for producing DNN models with good performance while reducing the labeling effort. Weiss et al. refined the well-known DNN testing metric, SA to reduce its computation cost [110]. Concretely, they proposed three

Table 9. Types of Tests used for Retraining

Data used for retraining	Test data	Reference
Original training + Selected tests	Independent tests	[11, 18, 28, 37, 44, 46, 75, 76, 105]
Selected tests	Whole tests	[38, 54, 55, 94, 117]
Original training + Selected tests	Whole tests	[5, 8, 26, 56, 110]
Selected tests	Independent tests	[65]
Part of original training + Selected tests	Independent tests	[71]

methods to reduce the size of training data and then build the activation trace, The first one is *uniform sampling* which randomly selects a part of training data. The second one is *unsurprising-first sampling* which selects the data with the lowest surprise from each class. The last one is *neighbor-free sampling*. The authors suggested removing the redundant data if the distance between their activation traces to any activation trace of an already selected one is smaller than a threshold. Here, the considered data should be in the same class. The authors demonstrated that their refined SA metrics have similar effectiveness to the original one while significantly reducing computation costs.

6.2 Analysis and Discussion

6.2.1 Evaluation Metric. Most of the works evaluated the retraining methods by comparing the accuracy of DNNs before and after retraining. Three works [11, 37, 105] considered and evaluated another property of DNNs – adversarial robustness. Adversarial robustness here means the ability of DNNs to correctly predict adversarial examples.

6.2.2 Datasets. Different from the fault detection task one DNN model and a test set are enough for the evaluation, retraining must be conducted on a DNN model and three parts of data, (1) candidate data that is used for data selection, (2) data that are used to retrain the model, and (3) test data that used to evaluate the performance of the retrained model. Therefore, there could be different settings for these three groups of data. Table 9 lists the types of data that were used for the retraining process from existing works. Here, *independent tests* in column *Test data* means that the candidate data and test data are different. We can summarize that existing works mainly lie in five modes as follows. And most of the works (43%) chose to combine the training data and selected test data for retraining, and then evaluate the model using another independent test set.

- *Original training + Selected tests | Independent tests* (43% works) – combine the training data and selected data for retraining, and evaluate the DNN using an independent test set.
- *Part of original training + Selected tests | Independent tests* (5% works) – combine the selected part of training data and selected data for retraining, and evaluate the DNN using an independent test set.
- *Selected tests | Whole tests* (24% works) – only use the selected data for retraining, and evaluate the DNN using the candidate set, that is, the candidate set and test set are the same.
- *Original training + Selected tests | Whole tests* (24% works) – combine the training data and selected data for retraining, and evaluate the DNN using the whole candidate set.
- *Selected tests | Independent tests* (5% works) –only use the selected data for retraining, and evaluate the DNN using an independent test set.

Additionally, we noticed that similar to the fault detection works, all the sampling-based retraining methods from the SE community only considered datasets and models that were originally studied by the ML community. SE tasks were ignored by the SE community.

6.2.3 *Pitfalls and Good Practice.*

- *Pitfall.* As shown by [44], when facing data distribution shifts in the candidate set, using only the selected data to retrain the model could produce a model with good performance on shifted data but will harm the performance of DNN on the in-distribution data. Thus, how to conduct the retraining (use which data) process highly affects the quality of the retrained model. The reported retraining performance can make a miss leading to developers when DNNs have been developed in the wild facing diverse data distributions. *Good practice.* We need to use the proper way for the retraining process. As suggested by [44], the good way is *Original training + Selected tests*.
- *Pitfall.* As revealed by [37], sampling-based training produces DNN models with good performance on clean test data but fails to produce models with good adversarial robustness. In terms of adversarial robustness, well-designed data selection methods cannot beat random selection. *Good practice.* In case the retraining will introduce biased performance to the retrained models, we need to prepare diverse datasets (not only the original test data) to evaluate the retrained model.
- *Pitfall.* Another problem with existing works is around half (48%) of works used the candidate set as the test set to evaluate the retrained model. There is a data leakage problem in this situation that the test set contains some training set. Thus, the reported results are not rigorous. *Good practice.* We need to avoid the data leakage problem and use an independent test set to evaluate the retrained model.

7 MODEL SELECTION

Since only three works specifically focused on the model selection task, we introduce them together.

7.1 Article Survey

Sun et al. introduced the problem of ranking models in unlabeled new environments [98]. To solve this problem, a novel method selects the labeled data that are similar to the unlabeled data from the training set to rank the model has been proposed by the authors. This data selection proposed was called proxy set searching. The search algorithm consists of three steps, first, the data pool was clustered into K subsets. Then, **Frechet Inception Distance** (**FID**) and feature variance gap between each subset and the target set were computed. Finally, inputs are selected from each cluster based on the FID and variance gap values. Here, considering the data from each cluster can guarantee the diversity of selected data to produce better estimation performance. As a result, the selected data will be used for ranking the models.

Hu et al. proposed a labeling-free model selection (LaF) [47]. Concretely, LaF uses data difficulty and model specialty to build a Bayesian model to estimate the likelihood of a predicted label being the true label. Here, data difficulty refers to how difficult a sample is for all DNNs to predict correctly. The model specialty reflects how good a model is to predict the correct labels of all samples. Thus, by optimizing the Bayesian model, we can get a proper value of the model specialty which can be directly used for the model selection. In this work, LaF utilizes **expectation-maximization** (**EM**) to optimize the Bayesian model. Different from other test optimization works from the SE community that only test their methods on ML tasks, this work conducted experiments on SE-related tasks, for example, problem classification Java250 [88].

Different from the above two works that do not need to label any new data, Meng et al. proposed a sampling-based model selection method named **Sample Discrimination based Selection** (**SDS**) [84]. SDS includes two key steps, firstly, it uses majority voting to label the unlabeled test data. Based on the labeled data, SDS filters out the top models with higher performance and the bottom models with lower performance. Secondly, SDS computes sample discrimination of each input based on its prediction consistency with the labels on top/bottom models. The inputs with higher consistency on top models have greater discrimination scores and will be selected for labeling and ranking models.

7.2 Analysis and Discussion

7.2.1 Evaluation Metric. Model selection works mainly utilize correlation analysis metrics to measure the correctness between the idea performance ranking and the estimated performance ranking. Three metrics have been used in existing works, Spearman's rank correlation coefficient, Jaccard similarity coefficient, and Kendall's rank correlation.

Given n samples and $s_1, s_2, \ldots, s_n$, let $r(s_1), r(s_2), \ldots, r(s_n)$ be the ground truth ranking and $r'(s_1), r'(s_2), \ldots, r'(s_n)$ be the estimated ranking.

Spearman's rank correlation coefficient. (Used in [47, 84, 98])

$$\rho = \frac{n \sum_{i=1}^{n} r(s_i) r'(s_i) - \left(\sum_{i=1}^{n} r(s_i)\right)\left(\sum_{i=1}^{n} r'(s_i)\right)}{\sqrt{\left[n \sum_{i=1}^{n} r(s_i)^2 - \left(\sum_{i=1}^{n} r(s_i)\right)^2\right]\left[n \sum_{i=1}^{n} r'(s_i)^2 - \left(\sum_{i=1}^{n} r'(s_i)\right)^2\right]}}. \tag{1}$$

Jaccard similarity coefficient. (Used in [47, 84] to measure the correctness of top-k rankings.)

$$J_k = \frac{\mid \{s_i \mid r(s_i) <= k\} \cap \{s_i \mid r'(s_i) <= k\} \mid}{\mid \{s_i \mid r(s_i) <= k\} \cup \{s_i \mid r'(s_i) <= k\} \mid}, 1 \leq i \leq n. \tag{2}$$

Kendall's rank correlation. (Used in [47, 98])

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T)(P + Q + U)}}, \tag{3}$$

where P and Q are the numbers of ordered and disordered pairs in $\{r(s_i), r'(s_i)\}$, respectively. T and U are the numbers of ties in $\{r(s_i)\}$ and $\{r'(s_i)\}$, respectively.

7.2.2 Pitfalls and Good Practice. Pitfalls. No comparison with performance estimation methods. The existing works have not clearly explained when we need to use model selection methods and when we need to use labeling-free performance estimation methods. The fact is that we can use performance estimation methods to assign performance to every single model and then rank them accordingly. However, since these model selection words have not conducted comparison experiments between their proposed methods with labeling-free performance estimation methods, it is doubtful whether these proposed model ranking methods are not necessarily needed.

Good Practice. When proposing new model selection methods, it is important to clarify why performance estimation methods cannot work in the studied scenario or conduct experiments to compare the proposed methods with state-of-the-art performance estimation methods to show their advantages. Besides, instead of using the aforementioned three evaluation metrics in Section 7.2.1, other recently proposed ranking evaluation metrics, such as **normalized discounted cumulative gain** (**NDCG**) [50] can be also considered in the experiments.

8 PERFORMANCE ESTIMATION

8.1 Article Survey

We divide performance estimation methods into two groups, sampling-based performance estimation methods and labeling-free performance estimation methods. The first type of method still

Table 10. Summary of Performance Estimation Papers

Community	Methodology	References
SE	new sampling-based method	[12, 69, 124]
	new labeling-free method	[48]
	empirical study	[121]
ML	new sampling-based method	[60, 61]
	new labeling-free method	[6, 14, 16, 19, 20, 22, 29, 34, 36, 52, 57, 68]
	empirical study	[25]
Others	new sampling-based method	[99]
	new labeling-free method	[17, 21, 53, 81, 102, 113, 127]
	empirical study	–

Others refer to other communities and Arxiv.

needs to label a part of the data from the unlabeled test set and then estimate the model performance using the labeled ones. The second one can predict the model performance without additional labeling effort. The same as the previous sections, we introduce works in order of their sources (i.e., from which community). Table 10 summarizes the type of each article.

8.1.1 *New Methods from SE.*

8.1.1.1 Sampling-Based Methods. Li et al. [69] first introduced the concept of efficient DNN testing to the SE community and proposed to select a subset of test data to assess the performance of DNNs based on the conditioning of the learned representation. It minimizes the distance between the whole test set and the selected set. Here, the *distance* is the probability distribution of the neuron outputs in the last hidden layer of the DNN models. The authors introduced two ways to measure the distribution, (1) output confidence-based distribution called CSS, and (2) cross entropy-based distribution called CES. The authors demonstrated that CES is more stable in different evaluation situations. Following the first work, Chen et al. introduced a clustering-based approach for model performance estimation, **Practical ACcuracy Estimation** (**PACE**) [12]. PACE first clusters all the test inputs into different groups using the HDBSCAN algorithm. Then, it utilizes the MMD-critic algorithm to select representative inputs from each cluster. Besides, PACE uses adaptive random exploration to collect inputs from the data that do not belong to any cluster and adds it to the already selected data. This step is useful to increase the diversity of the finally collected data. Importantly, the authors studied the impact of different features (which is an important factor for clustering) used to conduct the clustering.

Zhou et al. proposed a two-phase approach DeepReduce [124], to select test inputs and estimate the performance of DNNs accordingly. Firstly, DeepReduce uses a famous algorithm HGS to select a subset of test inputs that have the same neuron coverage as the whole inputs. Secondly, DeepReduce iteratively selects inputs to minimize the difference between the output distribution of selected data and the output distribution of the whole input. Here, the authors split the output of neurons in the last layer into K sections, and then use the percentage of data located in different sections as the output distribution. Besides, the KL Divergence is used to compute the difference in distribution. Finally, if the KL distance is smaller than a user-defined threshold, DeepReduce outputs the selected data.

8.1.1.2 Labeling-Free Methods. The only labeling-free performance estimation method in SE was Aries proposed by Hu et al. [48]. Aries is based on the distribution analysis of training and test data. Given the training data, Aries first uses dropout prediction to collect multiple outputs of each input. Then, it computes the distance of input to the decision boundaries based on the change rate of predicted labels across these multiple predictions. That means, if the change rate is higher, the input is closer to decision boundaries. After that, Aries splits the data space into multiple sections regarding the distance and computes the accuracy of each section using the data mapped in each section. Finally, given the new unlabeled test data, Aries maps them into each section and uses the corresponding accuracy to estimate the final accuracy of the set.

8.1.2 New Methods from ML.

8.1.2.1 Sampling-Based Methods. Inspired by the problem of active learning, Kossen et al. introduced the novel testing framework, Active testing [60], which aims to sample test inputs to estimate the loss of the whole set of inputs to reduce the labeling effort. This is the first work that explained the importance of active testing and the difference between active testing and active learning. In this work, the authors proposed a surrogate model-guided acquisition strategy for test selection. This is based on the Monte Carlo estimator and acquisition distribution q. Here, q was defined differently for different tasks. For example, for classification tasks, $q = 1 - \pi\left(y = y^*\left(x\right)|x\right)$, where $y^*\left(x\right)$ is the index of maximum probability and π is the surrogate model. Besides, to increase the estimation precision, the surrogate model is iteratively updated using the labeled test inputs by retraining. Following their previous work, Kossen et al. proposed the **Active Surrogate Estimators (ASEs)** [61], that actively learn a surrogate model and use this surrogate to select inputs to estimate the loss of the original model. This basic idea is similar to active testing [60], a new acquisition strategy, called **Expected Weighted Disagreement** (**XWED**), was proposed to select inputs. Simply speaking, XWED is a modified BALD [42]. It assigned a weight for each data point using its corresponding loss value from the original model and then applied BALD to select the inputs. Finally, the selected inputs were labeled and used for computing the total loss of the DNN model.

8.1.2.2 Labeling-Free Methods. Jiang et al. proposed the first work to use the correlation between the margin distribution of hidden layers and the model performance to predict the generalization gap [52]. Here, the margin distribution means the distance from the data to decision boundaries. The authors defined the decision boundary for each class pair (i, j) as $D_{(i,j)} = \{x|p_i\left(x\right) = p_j\left(x\right)\}$ and approximated it as follows:

$$d_{f,(i,j),}\left(x^l\right) = \frac{p_i x^l - p_j x^l}{||\nabla_{x^l} p_i x^l - \nabla_{x^l} p_j x^l||_2}, \tag{4}$$

where x^l means the representation of x from the lth later. Finally, the performance estimation was conducted by using a linear regression model to learn the relation between the distance and the performance. Similar to [52], DeChant et al. [19] also used the output of intermediate layers for performance estimation. The difference is this work trained a meta-model to predict accuracy. Specifically, it collected the intermediate and final outputs first. Then these outputs were labeled as *correct* or *incorrect* based on the correctness of their corresponding inputs. Finally, a binary classification model was trained using the outputs and labels. Given the new unlabeled inputs, their intermediate can be used to predict their correctness according to this trained binary classifier.

Different from [19] that relied on meta-model, Chuang et al. proposed to use **domain-invariant representations** (**DIR**) models to learn a latent, joint embedding of source and target data, and a predictor from the latent space to the output labels for target risk prediction [14]. A $F_{G\Delta G}$-divergence was defined and used for measuring the difference between two representation domains. Besides, in this work, the authors demonstrated that for DNNs, the selection of the intermediate layer to divide the model into encoder (which affects the complexity of produced embeddings) and predictor highly affects the trained DIR models for risk prediction.

Corneanu et al. proposed to use persistent topology measures to estimate the testing error of DNNs on unseen samples [16]. Specifically, firstly, given the DNN model and inputs, the correlations between each pair of neurons were computed by:

$$\sum_{i=1}^{N} \frac{\left(a_{pi} - \overline{a_p}\right)\left(a_{qi} - \overline{a_q}\right)}{S_{a_p} S_{a_q}}, \tag{5}$$

where a_i is the activation value of a neuron, $\overline{a}$ and S_a are the mean and standard deviation of all activate values. After that, a persistent diagram was computed based on the activation values using Vietoris-Rips filtration and persistent homology methods. This persistent diagram indicates the sequence status of birth and death of neurons. Then, topological summaries (average time and average density) of the persistent diagram were used to build a linear function to predict the performance gap.

Instead of training learners for performance estimation, Guillory et al. proposed a simple method that uses the **difference of confidence** (**DoC**) to estimate the performance of classifiers [36]. Here, the DoC was computed by the difference between the average maximum probabilities of training data and the average maximum probabilities of test data. Besides, the authors introduce a variant of DoC, the **difference of average entropy** (**DoE**) which uses the entropy of probabilities to replace the maximum probabilities in DoC. After getting the DoC or DoE, a linear regression model was used to learn the relation between DoC (or DoE) and the accuracy of models for performance estimation. Garg et al. also proposed a very simple method **Average Thresholded Confidence** (**ATC**) [29] to detect faults. ATC first utilized the validation set to find a threshold that the fraction of the confidence of inputs above the threshold matches the validation set accuracy. After that, given the enabled test data, ATC calculated such fraction based on the determined threshold and estimated the performance of DNN on this set accordingly.

Deng et al. introduced a new model evaluation paradigm—**Automatic model Evaluation** (**AutoEval**) [22]. AutoEval tends to assess the quality of DNN models without using the labeled data. To do so, the authors proposed a dataset-level regression method. Specifically, given a DNN model, and a training dataset, this method first generated a large number of metasets based on a randomly sampled seed set using image synthesis techniques. After that, the Frechet distance between the representation of the original training set and the met sets is calculated. Here, the representation is the mean and covariance of the image feature vectors. Finally, a regression model or NN model was used to learn the relation between the distance and model performance for further performance estimation. Following this work, the authors found that there is a strong linear relationship between semantic classification accuracy and the accuracy of the rotation prediction task which can be used for performance estimation [20]. Based on this finding, given a DNN model, the authors added an auxiliary rotation prediction into the model, in turn, building a multi-task model. Since the rotation prediction is an unsupervised task, when facing new unlabeled inputs, the model can predict their rotation head directly. Then, the linear relationship between the two accuracies can be used to predict the semantic prediction accuracy of DNNs.

Baek et al. discovered a phenomenon, Agreement-on-the-Line [6], which means ID vs. OOD agreement for pairs of DNNs has a linear correlation when the corresponding ID vs. OOD accuracy also lies on a line. Based on this phenomenon, the authors proposed two methods ALine-D and ALine-S to estimate the performance of DNNs on new unlabeled data. More concretely, the authors first conducted a large-scale empirical study and revealed that *When ID vs. OOD accuracy observes a strong linear correlation (≥ 0.95 R2 values), we see that ID vs. OOD agreement is also strongly linearly correlated with the similar slope and bias.* Then, based on the finding, ALine-S and ALine-D have been proposed. ALine-S simply used the slope and bias from disagreement to estimate the accuracy of unlabeled data, while ALineD built a matrix of the combination of slope and bias and found the best solution for performance estimation.

Li et al. considered the label-imbalanced situation where the existing performance estimation methods could produce unexpected results [68]. To tackle this label-imbalanced issue, the authors plugged a class-specific **temperature scaling** (**TS**) technique into existing methods to improve their effectiveness. Specifically, the TS technique rescaled the output probabilities using the temperature parameter to fit the distribution of classes.

Unlike other works, Guan et al. brought the performance estimation to the instance segmentation problem in self-driving cars [34]. A distance-based method has been proposed in this work. Concretely, it extracted the representation vectors of ID and OOD inputs by instance segmentation detectors first, and then computed the **FrÃĽchet distance** (**FD**) between these two representations. Here, the first-order mean and second-order covariance matrix of the representation vectors has been used for the distance calculation. Finally, regression models were used to learn the relation between the distance gap and the accuracy for further performance estimation.

Finally, Kleyko et al. proposed to use Perceptron theory to predict the performance of DNNs [57]. In the perceptron theory, the mean and standard deviation of the last hidden layer can be used to predict the accuracy of DNN models. Therefore, the authors train regression models to estimate the performance based on the last hidden layer outputs.

8.1.3 *New Methods from Others.*

8.1.3.1 Sampling-Based Methods. Sun et al. proposed a three-step clustering-based method to estimate the model performance [99]. In this first step, the histograms of marginal distributions have been recorded to represent the input data as h_{shape}, that is, the matrix contains the number of values in each range of each feature dimension. After that, clustering methods were applied to divide the inputs into K groups, where K is the number of classes. The cluster centers will be selected as the first part of the labeled data as $h_{cluster}$. Then, the **farthest points sampling** (**FPS**) method was applied to select diverse data samples from the center to the marginal as h_{sample}. Finally, the combined features [h_{shape}, $h_{cluster}$, and h_{sample}] were used to train a regression model for performance estimation.

8.1.3.2 Labeling-Free Methods. Martin et al. found that there is a clear correlation between the weight matrix and the accuracy of DNNs [81] that can be used for performance estimation. Specifically, the authors empirically studied the linear correlation between the norm-based capacity control metrics and the power law-based metrics from the Theory of Heavy-Tailed self-regularization with the model performance. By using this correlation, it is able to estimate the accuracy of any DNN model. Following the same intuition, Unterthiner et al. directly used the weights of DNNs to predict their performance [102]. They empirically showed the correlation between the model parameters (flattened weights, weight statistics, and weight norms) and the model performance, and found that weight statistics are the best feature to build this correlation. Besides, they studied three types of estimators for learning such correlations, **logit-linear** (**L-Linear**) model, **gradient boosting machine** (**GBM**) using regression trees, and a fully connected DNN, and found DNN is the best option. Most importantly, the authors released a large dataset with 120k CNNS to support the research of performance estimation. Besides, Deng et al. found that there is a strong correlation between the dispersity of outputs (i.e., label-balance, the entropy of predicted probabilities was used in the work) and the accuracy of DNNs and utilized this phenomenon to predict the model performance [21]. In addition to using the dispersity value to predict performance directly, the nuclear norm [17] has been used to combine the model confidence and the dispersity. After that regression model was trained based on the combined values for final performance estimation.

Similar to work [6], Jiang et al. also found that there is a strong correlation between the disagreement and test errors of two DNNs that were trained by using the same architecture and training data but different random seed [53]. Thus, this finding can be used to estimate the performance of DNNs by only using the disagreement information from the model and its auxiliary model. Besides, the authors found that the ensembled model using these trained models has a well-calibrated nature. Zhu et al. found that OOD data could affect the effectiveness of AutoEval [23] and proposed to remove the OOD data from the unlabeled set before conducting performance estimation [127]. Here, an Energy-based OOD detection method that uses the energy score of output probabilities to detect OOD data has been used in the OOD detection process.

Xie et al. [113] empirically demonstrated that the distribution difference cannot be always useful for performance estimation, that is, existing methods are not always reliable. Besides, the authors defined a new score—dispersion score to replace the distribution difference. To do so, the inputs were divided into different groups first based on their pseudo labels. Then, the dispersion score was computed by the average distances between each cluster center and the center of all features, weighted by the sample size of each cluster. Finally, regression models were trained for performance estimation using the pairs of dispersion score and model performance.

8.1.4 Empirical Study from SE. In contrast to prior studies that assess model performance across all classes, Zhao et al. [121] conducted an empirical investigation to explore the effectiveness of existing performance estimation methods for individual classes. Based on the empirical study, they found that (1) existing methods have high estimation errors in each class especially when the subset size is small, (2) the overall performance of the methods can be further improved if we reduce their accuracy estimation errors in each class, and (3) there is still a large performance improvement room for each performance estimation method.

8.1.5 Empirical Study from ML. Elsahar et al. [25] empirically studied three methods, H-divergence, Confidence-based, and Reverse classification accuracy for predicting the performance drops of DNNs. H-divergence-based methods use another classification model to distinguish between training data and target data. Confidence-based methods use the difference of output probabilities to measure the performance drops. Reverse classification accuracy methods utilize the pseudo-label predicted by the original DNN to train a new DNN with the same architecture and training configuration. The performance drops are calculated by comparing the performance of original and newly trained DNNs. The authors found the H-divergence-based methods are better at predicting the performance drop on natural distribution shift while Confidence-based methods are good at predicting the performance drop on adversarial distribution shift.

8.2 Analysis and Discussion

8.2.1 Evaluation Metric. Table 11 lists the evaluation metrics used for performance estimation. Basically, there are two types of metrics, the first one directly computes the difference between real model performance and estimated model performance, for example, *MSE* and *Accuracy difference*. The second one evaluates the correlation between the scores (computed by their proposed methods) and the real model performance, for example, *Pearson correlation coefficient* and *Coefficient of determination* Therefore, multiple data values (multiple models or datasets used to calculate such values) are needed for the computation of correlation.

8.2.2 Datasets. Table 12 summarizes the types of datasets used for the evaluation in each work. Surprisingly, 32% of the works only evaluated their methods using in-distribution datasets. Besides, only six studies [6, 10, 21, 36, 99, 127] evaluated performance estimation methods on in-distribution dataset, synthetic distribution dataset, and natural distribution dataset.

Table 11. Evaluation Metrics for Performance Estimation

Name	Equation	Description	Involved references
Mean Squared Error (MSE)	$\frac{1}{N}\sum_{i=1}^{N}(ACC_e - ACC_b)^2$	N: repeat times ACC_e: estimated accuracy ACC_b: base accuracy	[12, 22, 35, 52, 69] [13, 20, 34, 81, 127] [99]
Accuracy difference	$Abs\,(ACC_e - ACC_b)$	ACC_e: estimated accuracy ACC_b: base accuracy	[14, 16, 25, 48, 121] [6, 10, 20, 22, 36, 53] [29, 68]
Coverage difference	$Abs\,(Cov_e - Cov_b)$	Cov_e: estimated coverage Cov_b: base coverage	[121]
Pearson correlation coefficient	$\frac{\sum_{i=1}^{N}(V_i-\bar{V})(ACC_i-\overline{ACC})}{\sqrt{\sum_{i=1}^{N}(V_i-\bar{V})^2(ACC_i-\overline{ACC})^2}}$	N: number of models (or datasets) V: metrics proposed from the paper ACC: accuracy of models	[13, 14, 20, 57, 121] [113]
Spearman's rank correlation	Equation (1)	–	[21]
Kendall correlation	Equation (3)	–	[57, 81]
Max error	$Max\left(ACC_e^i - ACC_b\right)$	ACC_e: estimated accuracy ACC_b: base accuracy $i \in 1, 2, \ldots, N$, N: repetition times	[25]
Coefficient of determination (R^2)	$1 - \frac{\sum_i^n(\widehat{g}_i-g_i)^2}{\sum_{i=1}^{n}\left(g_i-\frac{1}{n}\sum_{i=1}^{n} g_i\right)^2}$	n: number of models $\widehat{g}_i$: estimated accuracy gap g_i: ground-truth accuracy gap	[6, 21, 52, 81, 113] [102]
Median squared error	–	Median value of MSEs	[60]

Table 12. Types of Datasets used Performance Estimation

ID data	Synthetic DS	Natural DS	Reference	Percentage
✓	✓	✓	[6, 10, 21, 36, 99, 127]	19%
✓	✓	✗	[12, 22, 46, 48, 53, 69, 113, 121]	26%
✓	✗	✗	[16, 19, 35, 52, 57, 60, 63, 81, 102, 124]	32%
✓	✗	✓	[14, 20]	6%
✗	✗	✓	[13]	3%
✗	✓	✓	[25, 29, 34, 68]	13%

DS: Distribution Shift.

8.2.3 *Pitfalls and Good Practice.*

- *Pitfall.* No cross-type comparison is performed in existing sampling-based methods. All the works that focus on sampling-based methods have not conducted comparison experiments with the labeling-free methods. However, some works [48] showed that labeling-free methods can achieve better results than sampling-based methods.
  - *Good practice.* When proposing new sampling-based methods, it is necessary to show their advantage not only over other sampling-based methods but also the labeling-free methods. When evaluating the performance estimation methods, instead of using only a small set of DNN models, consider statistically checking the effectiveness of proposed methods on multiple DNNs is a better choice (e.g., like works [6, 102]).
- *Pitfall.* As mentioned before, 32% of works only evaluated their methods using in-distribution datasets. The reliability of using these methods on other data distributions is unclear.
  - *Good practice.* Considering only in-distribution datasets is insufficient, one needs to evaluate the performance estimation methods on distribution-shifted datasets (especially the natural distribution-shifted ones), which are more likely to happen in the wild.

8.2.4 *SE vs. ML.* We compare the works from SE and ML from the following three perspectives.

- **Proposed methods.** We found that researchers from the SE community prefer to propose sampling-based methods instead of labeling-free methods. Specifically, only one work from SE is labeling-free. However, researchers from the ML community mainly focus on the labeling-free model evaluation, for example, around 79% of works are labeling-free. Researchers from SE tend to select representative tests for assessing the model performance [12, 69] while researchers from ML try to find the relation between the model performance and other properties (e.g., margin distribution [52], agreements [6]).

- **Evaluated datasets and models.** None of the SE works considers natural distribution shifted datasets, while more than half (55%) of ML works evaluate their methods on such datasets. Besides, similar to other tasks in test optimization, none of the SE works consider estimating the performance of SE-related DL models, for example, clone detection models. Most of the SE works focus on computer vision tasks and models.
- **Evaluation metrics.** SE works rarely use correlation analysis metrics since they evaluate their methods on different single models. On the other hand, nine works from ML utilize correlation metrics for the evaluation. Besides, among these nine works, [16, 20, 52] utilize correlation analysis metrics to initially verify their methods, and then report the specific values of individual models.

9 OPPORTUNITIES

Numerous researchers have dedicated their efforts to enhancing the test optimization in DNN testing and achieved exciting results. However, there remain several opportunities within this field that are waiting for exploration.

More tasks. Most (85 out of 90) of the studied works only consider classification tasks and their proposed methods lack generalizability to other types of tasks. For example, [26, 94] rely on classification probabilities, which are exclusive to classification datasets. However, other tasks, such as the regression task in self-driving cars studied in [12], are practical yet receive limited attention. Additionally, there is a growing interest in generative models like LLMs. How to efficiently estimate the performance of LLMs presents a crucial challenge. For example, when using ChatGPT to solve a bug detection task for a large-scale project, how can one quickly identify the wrongly-detected program by ChatGPT or estimate the overall performance of ChatGPT on this project, to save time and money? In summary, there are many research and practical opportunities in studying test optimization in DNN in regression and generative tasks. To handle non-classification tasks, it is promising to focus on the embedding features extracted by the data and models (e.g., neuron outputs) and propose methods to select important test data, which is similar to the work [22] that estimates the model performance by comparing the feature difference.

Moreover, existing performance estimation methods only considered (or be evaluated on) the specifically prepared in/out-of-distribution (ID/OOD) datasets. How to efficiently predict the model performance in terms of the general fairness/adversarial robustness or other DNN properties is also an important and promising research problem. For example, researchers can test the effectiveness of performance estimation methods (or propose new methods) on fairness datasets such as Mep15 [27] using fairness metrics such as **Average Odds Difference** (**AOD**).

More application scenarios. Existing works mainly focused on model-level testing – revealed problems in DNNs, and only a few works targeted the application domains, for example, [104] specifically proposed a fault detection method for self-driving cars. However, from the real usage perspective, studying the test optimization problem at the application level is more important. In addition to the self-driving car, DNNs have been used for many other applications such as video game [123], code completion [31], and finance analysis [30]. How to propose new methods or study existing methods in the above applications is a promising research direction. For example, considering testing code completion applications, for example, Copilot, as the black-box property, we can detect the faults only by the input and output, and cannot repair them using model retraining methods. The promising method to repair such applications is input modification [70].

Besides, when considering tasks in the finance field, it is easy to encounter datasets with constraints, for example, the limited number of genders. In this situation, how to design perturbations for mutation-based input prioritization is an interesting problem. One potential solution is to add constraint rules in the mutation operators to control the modification of the inputs. There are multiple constraints-based input generation techniques we can follow [30, 96].

Study cross communities. There is an interesting fact that works in one community rarely compare baselines from other communities. For example, only two works [37, 66] from the ML community considered works from the SE community as baselines. It is unclear which method (from which community) we should use in the real scenario. Therefore, it is necessary to build tools and benchmarks that support and compare test optimization works from different communities.

LLM-driven test optimization. Recently, Autolabel[4] has been proposed to use LLM to label data. The experiments showed that by using Autolabel, the labeling precision is at a similar level to humans but the labeling speed is 100 times faster than humans. This success brings the opportunities to use LLM for DNN testing optimization. How to combine the existing test optimization techniques with LLM or how to propose new LLM-based test optimization techniques could be a promising research direction. One potential solution is to use test optimization methods to select informative data first, and then use Autolabel such selected data for post-phase model testing/repairing. In this way, we can avoid the human effort and make the testing process fully automated.

Combination of test generation and test optimization. Current test optimization works focus on fixed testing scenarios, that is, testing on existing datasets. Test generation that can create diverse tests has been studied [87, 114] in the ML testing field. It is possible to combine test generation and test optimization to better test DNNs. For example, we can use some test generation methods (e.g., DeepHunter [114]) to generate tests first, and then utilize sampling-based model retraining methods (e.g., DAT [44]) to further boost the performance of DNN models.

Test optimization vs. active learning. Some works [37, 111] have demonstrated that test optimization methods can be used for active learning. Thus, it is promising to adapt the existing test optimization techniques to the active learning scenario to check their effectiveness. On the other hand, it is also possible to try active learning methods in test optimization problems to check how they can help DNN testing. There are multiple active learning benchmarks [45, 64] and test optimization projects [76, 111] that we can directly use for this research topic.

10 DISCUSSION

10.1 Threats to Validity

Our survey may contain the following threats to validity.

Potential oversight of relevant work. This threat may occur during the literature search, where we utilize the keywords from paper titles. We deliberately avoid using keywords from abstracts due to the substantial volume of unrelated papers retrieved. For instance, a search with the keyword *fault detection* in abstracts yields over 2,590,000 results. To mitigate the risk of overlooking literature, we thoroughly check references in each collected paper and do a snowball to find all related works. All the authors double-checked the references to ensure all the related works were covered in our survey. Besides, we commit to maintaining our survey website to continuously gather future literature that is relevant to our research.

[4] https://github.com/refuel-ai/autolabel

Potential absence of comparison across diverse communities. Our survey mainly compares works from the SE and ML communities, with limited consideration for other communities. This choice stems from the fact that the majority of our collected works (75 out of 90) originate from these two communities. Among the remaining 15 works, 10 are sourced from arXiv. This distribution implies that the SE and ML communities exhibit a primary focus on the surveyed topic. In the future, we will broaden the comparison to include specific communities beyond SE and ML, subject to the appearance of further works from these communities.

10.2 Importance of Test Optimization in DNN Testing

Given a DL model and its associated task, the potential pool of unlabeled test data can be infinite. For instance, the continual generation of diverse street views in the same location for self-driving cars. This abundance of unlabeled test data poses a big challenge to DNN testing. Test optimization steps in to alleviate this challenge by strategically reducing the testing effort from the perspective of minimizing the labeling cost. Existing studies have successfully demonstrated that it's possible to assess the performance of DL models without relying on labeled data [22, 48], and even repair DL models using just 10% of the test data [26, 65].

This survey provides a comprehensive review of existing studies within the realm of test optimization in DNN testing, encompassing the objectives of proposed methods, studied datasets, and employed evaluation metrics. This review can help other researchers swiftly grasp the advancements made in this field. Additionally, we reveal potential pitfalls and offer good practice guidance, aiming to steer researchers away from making the same mistakes. Furthermore, we draw comparisons between works from the SE and ML communities, providing insights to bridge the gap of understanding in this field between these distinct communities.

11 CONCLUSION

We comprehensively surveyed the works related to test optimization in DNN testing. This survey unified the four tasks in test optimization, that is, fault detection, sampling-based model retraining, model selection, and performance estimation. It summarized each work within these tasks, conducted a comparative analysis of research focuses between the SE and ML communities, identified pitfalls in existing works, and offered guidance on good practices. Additionally, we outlined potential research opportunities in the test optimization field. Our aim is that this survey proves instrumental for researchers to better understand the existing works and provides insights into the future trends of test optimization in DNN testing. The entire project can be found at: https://wellido.github.io/TOID.html

REFERENCES

[1] Zohreh Aghababaeyan, Manel Abdellatif, Lionel Briand, S. Ramesh, and Mojtaba Bagherzadeh. 2023. Black-box testing of deep neural networks through test case diversity. *IEEE Transactions on Software Engineering* 49, 5 (May 2023), 3182–3204. DOI: https://doi.org/10.1109/TSE.2023.3243522

[2] Zohreh Aghababaeyan, Manel Abdellatif, Mahboubeh Dadkhah, and Lionel Briand. 2023. DeepGD: a multi-objective black-box test selection approach for deep neural networks. arXiv:2303.04878 Retrieved from https://arxiv.org/pdf/2303.04878

[3] Jonathan Aigrain and Marcin Detyniecki. 2019. Detecting adversarial examples and other misclassifications in neural networks by introspection. arXiv:1905.09186 Retrieved from https://arxiv.org/pdf/1905.09186

[4] Hamzah Al-Qadasi, Changshun Wu, Yliès Falcone, and Saddek Bensalem. 2022. DeepAbstraction: 2-level prioritization for unlabeled test inputs in deep neural networks. In *Proceedings of the IEEE International Conference On Artificial Intelligence Testing*. IEEE, Piscataway, NJ, USA, 64–71. DOI: https://doi.org/10.1109/AITest55621.2022.00018

[5] Mohammed Attaoui, Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2023. Black-box safety analysis and retraining of dnns based on feature extraction and clustering. *ACM Transactions on Software Engineering and Methodology* 32, 3 (2023), 1–40. DOI: https://doi.org/10.1145/3550271

[6] Christina Baek, Yiding Jiang, Aditi Raghunathan, and J Zico Kolter. 2022. Agreement-on-the-line: predicting the performance of neural networks under distribution shift. *Advances in Neural Information Processing Systems* 35 (2022), 19274–19289.

[7] Shenglin Bao, Chaofeng Sha, Bihuan Chen, Xin Peng, and Wenyun Zhao. 2023. In defense of simple techniques for neural network test case selection. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. Association for Computing Machinery, New York, NY, USA, 501–513. DOI: https://doi.org/10.1145/3597926.3598073

[8] Taejoon Byun, Vaibhav Sharma, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. 2019. Input prioritization for testing neural networks. In *Proceedings of the 2019 IEEE International Conference On Artificial Intelligence Testing*. IEEE, 63–70. DOI: https://doi.org/10.1109/AITest.2019.000-6

[9] Jinyin Chen, Jie Ge, and Haibin Zheng. 2022. ActGraph: prioritization of test cases based on deep neural network activation graph. *Automated Software Engineering* 30, 28 (2022), 28. DOI: https://doi.org/10.1007/s10515-023-00396-8

[10] Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. 2021. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. In *Advances in Neural Information Processing Systems* 34 (2021), 14980–14992.

[11] Jialuo Chen, Jingyi Wang, Xingjun Ma, Youcheng Sun, Jun Sun, Peixin Zhang, and Peng Cheng. 2022. QuoTe: quality-oriented testing for deep learning systems. *ACM Transactions on Software Engineering and Methodology* 32, 5, Article 125 (2022), 33 pages. DOI: https://doi.org/10.1145/3582573

[12] Junjie Chen, Zhuo Wu, Zan Wang, Hanmo You, Lingming Zhang, and Ming Yan. 2020. Practical accuracy estimation for efficient deep neural network testing. *ACM Transactions on Software Engineering and Methodology* 29, 4 (2020), 1–35. DOI: https://doi.org/10.1145/3394112

[13] Lingjiao Chen, Matei Zaharia, and James Y. Zou. 2022. Estimating and explaining model performance when both covariates and labels shift. *Advances in Neural Information Processing Systems* 35 (2022), 11467–11479.

[14] Ching-Yao Chuang, Antonio Torralba, and Stefanie Jegelka. 2020. Estimating generalization under distribution shifts via domain-invariant representations. In *Proceedings of the 37th International conference on machine learning* (Virtual). PMLR, Brookline, MA, USA, 1984–1994. Retrieved from https://proceedings.mlr.press/v119/chuang20a/chuang20a.pdf

[15] Jürgen Cito, Isil Dillig, Seohyun Kim, Vijayaraghavan Murali, and Satish Chandra. 2021. Explaining mispredictions of machine learning models using rule induction. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 716–727. DOI: https://doi.org/10.1145/3468264.3468614

[16] Ciprian A. Corneanu, Sergio Escalera, and Aleix M. Martinez. 2020. Computing the testing error without a testing set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Virtual). IEEE Computer Society, Los Alamitos, CA, USA, 2674–2682. DOI: https://doi.org/10.1109/CVPR42600.2020.00275

[17] Shuhao Cui, Shuhui Wang, Junbao Zhuo, Liang Li, Qingming Huang, and Qi Tian. 2020. Towards discriminability and diversity: batch nuclear-norm maximization under label insufficient situations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 3940–3949.

[18] Xueqi Dang, Yinghua Li, Mike Papadakis, Jacques Klein, Tegawendé F. Bissyandé, and Yves L. E. Traon. 2023. GraphPrior: mutation-based test input prioritization for graph neural networks. *ACM Transactions on Software Engineering and Methodology* 33, 1, Article 22 (November 2023), 40 pages. DOI: https://doi.org/10.1145/3607191

[19] Chad DeChant, Seungwook Han, and Hod Lipson. 2019. Predicting the accuracy of neural networks from final and intermediate layer outputs. In *Proceedings of the ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*. OpenReview.net, Online, 1–6. Retrieved from https://openreview.net/pdf?id=H1xXwEB2h4

[20] Weijian Deng, Stephen Gould, and Liang Zheng. 2021. What does rotation prediction tell us about classifier accuracy under varying testing environments?. In *Proceedings of the International Conference on Machine Learning* (Virtual). PMLR, Brookline, MA, USA, 2579–2589. Retrieved from https://proceedings.mlr.press/v139/deng21a/deng21a.pdf

[21] Weijian Deng, Yumin Suh, Stephen Gould, and Liang Zheng. 2023. Confidence and dispersity speak: characterising prediction matrix for unsupervised accuracy estimation. arXiv:2302.01094 Retrieved from https://arxiv.org/pdf/2302.01094

[22] Weijian Deng and Liang Zheng. 2021. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Nashville, TN, USA, 15064–15073. DOI: https://doi.org/10.1109/CVPR46437.2021.01482

[23] Weijian Deng and Liang Zheng. 2021. Are labels always necessary for classifier accuracy evaluation?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Nashville, TN, USA, 15064–15073. DOI: https://doi.org/10.1109/CVPR46437.2021.01482

[24] Yao Deng, Xi Zheng, Mengshi Zhang, Guannan Lou, and Tianyi Zhang. 2022. Scenario-based test reduction and prioritization for multi-module autonomous driving systems. In *Proceedings of the 30th ACM Joint European*

Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, 82–93. DOI: https://doi.org/10.1145/3540250.3549152

[25] Hady Elsahar and Matthias Gallé. 2019. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Hong Kong, China, 2163–2173. DOI: https://doi.org/10.18653/v1/D19-1222

[26] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. 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*. Association for Computing Machinery, New York, NY, USA, 177–188. DOI: https://doi.org/10.1145/3395363.3397357

[27] Agency for Healthcare Research & Quality. 2017. MEPS HC-181: 2015 full year consolidated data file.

[28] Xinyu Gao, Yang Feng, Yining Yin, Zixi Liu, Zhenyu Chen, and Baowen Xu. 2022. Adaptive test selection for deep neural networks. In *Proceedings of the 44th International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 73–85. DOI: https://doi.org/10.1145/3510003.3510232

[29] Saurabh Garg, Sivaraman Balakrishnan, Zachary Chase Lipton, Behnam Neyshabur, and Hanie Sedghi. 2021. Leveraging unlabeled data to predict out-of-distribution performance. In *Proceedings of the NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications* (Virtual). OpenReview.net, Online, 1–30.

[30] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boystov, Yves Le Traon, and Anne Goujon. 2020. Search-based adversarial testing and improvement of constrained credit scoring systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1089–1100. DOI: https://doi.org/10.1145/3368089.3409739

[31] GitHub, OpenAI. 2022. Project site of GitHub Copilot. Retrieved from https://github.com/features/copilot Accessed on January 23rd, 2024.

[32] Federica Granese, Marco Romanelli, Daniele Gorla, Catuscia Palamidessi, and Pablo Piantanida. 2021. Doctor: a simple method for detecting misclassification errors. In *Advances in Neural Information Processing Systems (NeurIPS'21)*. 34 (2021), 5669–5681.

[33] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. 2018. Recent advances in convolutional neural networks. *Pattern recognition* 77 (2018), 354–377. DOI: https://doi.org/10.1016/j.patcog.2017.10.013

[34] Licong Guan and Xue Yuan. 2023. Instance segmentation model evaluation and rapid deployment for autonomous driving using domain differences. *IEEE Transactions on Intelligent Transportation Systems* 24, 4 (April 2023), 4050–4059. DOI: https://doi.org/10.1109/TITS.2023.3236626

[35] Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. 2021. Operation is the hardest teacher: estimating DNN accuracy looking for mispredictions. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering* . IEEE Press, Madrid, Spain, 348–358. DOI: https://doi.org/10.1109/ICSE43902.2021.00042

[36] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. 2021. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF international conference on computer vision*. IEEE, Piscataway, NJ, USA, 1134–1144. DOI: https://doi.org/10.1109/ICCV48922.2021.00117

[37] Yuejun Guo, Qiang Hu, Maxime Cordy, Michail Papadakis, and Yves Le Traon. 2023. DRE: density-based data selection with entropy for adversarial-robust deep learning models. *Neural Computing and Applications* 35, 5 (October 2023), 4009–4026. DOI: https://doi.org/10.1007/s00521-022-07812-2

[38] Yao Hao, Zhiqiu Huang, Hongjing Guo, and Guohua Shen. 2023. Test input selection for deep neural network enhancement based on multiple-objective optimization. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE Computer Society, Los Alamitos, CA, USA, 534–545. DOI: https://doi.org/10.1109/SANER56733.2023.00056

[39] Changtian He, Qing Sun, Ji Wu, Haiyan Yang, and Tao Yue. 2022. Feature difference based misclassified sample detection for CNN models deployed in online environment. In *Proceedings of the IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion*. IEEE, Piscataway, NJ, USA, 768–769. DOI: https://doi.org/10.1109/QRS-C57518.2022.00126

[40] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net, Online, 1–16.

[41] Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution dxamples in neural networks. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net, Online, 1–12.

[42] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. arXiv:1112.5745. Retrieved from https://arxiv.org/pdf/1112.5745

[43] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, and Timothy Hospedales. 2015. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. IEEE Computer Society, Los Alamitos, CA, USA, 384–392. DOI: https://doi.org/10.1109/ICCVW.2015.58

[44] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Lei Ma, Mike Papadakis, and Yves Le Traon. 2022. An empirical study on data distribution-aware test selection for deep learning enhancement. *ACM Transactions on Software Engineering and Methodology* 31, 4 (2022), 1–30. DOI: https://doi.org/10.1145/3511598

[45] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Wei Ma, Mike Papadakis, and Yves Le Traon. 2021. Towards exploring the limitations of active learning: an empirical study. In *Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, Piscataway, NJ, USA, 917–929. DOI: https://doi.org/10.1109/ASE51524.2021.9678672

[46] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Wei Ma, Mike Papadakis, and Yves Le Traon. 2023. Evaluating the robustness of test selection methods for deep neural networks. arXiv:2308.01314. Retrieved from https://arxiv.org/pdf/2308.01314

[47] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, and Yves Le Traon. 2023. LaF: labeling-free model selection for automated deep neural network reusing. *ACM Transactions on Software Engineering and Methodology* 33, 1, Article 25 (November 2023), 28 pages. DOI: https://doi.org/10.1145/3611666

[48] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, Lei Ma, and Yves Le Traon. 2023. Aries: efficient testing of deep neural networks via labeling-free accuracy estimation. In *Proceedings of the 45th International Conference on Software Engineering*. IEEE Press, Piscataway, NJ, USA, 1776–1787. DOI: https://doi.org/10.1109/ICSE48619.2023.00152

[49] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review* 37 (2020), 100270. DOI: https://doi.org/10.1016/j.cosrev.2020.100270

[50] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446. DOI: https://doi.org/10.1145/582415.582418

[51] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. 2018. To trust or not to trust a classifier. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (Vol. 31)*. Curran Associates Inc., Red Hook, NY, USA, 5546–5557.

[52] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. 2019. Predicting the generalization gap in deep networks with margin distributions. arXiv:1810.00113. Retrieved from https://arxiv.org/pdf/1810.00113

[53] Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. 2022. Assessing generalization of SGD via disagreement. arXiv:2106.13799. Retrieved from https://arxiv.org/pdf/2106.13799

[54] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, Montreal, Quebec, Canada, 1039âĂŞ1049. DOI: https://doi.org/10.1109/ICSE.2019.00108

[55] Jinhan Kim, Robert Feldt, and Shin Yoo. 2023. Evaluating surprise adequacy for deep learning system testing. *ACM Transactions on Software Engineering and Methodology* 32, 2, Article 42 (March 2023), 29 pages. DOI: https://doi.org/10.1145/3546947

[56] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1466âĂŞ1476. DOI: https://doi.org/10.1145/3368089.3417065

[57] Denis Kleyko, Antonello Rosato, Edward Paxon Frady, Massimo Panella, and Friedrich T. Sommer. 2023. Perceptron theory can predict the accuracy of neural networks. *IEEE Transactions on Neural Networks and Learning Systems (Early Access)* (2023), 1–15. DOI: https://doi.org/10.1109/TNNLS.2023.3237381

[58] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: a benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, Brookline, MA, USA, 5637–5664. Retrieved from https://proceedings.mlr.press/v139/koh21a.html

[59] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1137–1143.

[60] Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. 2021. Active testing: Sample-efficient model evaluation. In *Proceedings of the International Conference on Machine Learning*. PMLR, Brookline, MA, USA, 5753–5763.

[61] Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Thomas Rainforth. 2022. Active surrogate estimators: An active learning approach to label-efficient model evaluation. In *Advances in Neural Information Processing Systems (NeurIPS'22)*. 35 (2022), 24557–24570.

[62] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018. Training confidence-calibrated classifiers for detecting out-of-distribution samples. arXiv:1711.09325. Retrieved from https://arxiv.org/pdf/1711.09325

[63] Young-Woo Lee and Heung-Seok Chae. 2023. Selection of test samples to improve DNN test efficiency based on neuron clusters. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4399496 Accessed on January 23rd, 2024.

[64] Yu Li, Muxi Chen, Yannan Liu, Daojing He, and Qiang Xu. 2022. An empirical study on the efficacy of deep active learning for image classification. arXiv:2212.03088. Retrieved from https://arxiv.org/pdf/2212.03088

[65] Yu Li, Muxi Chen, and Qiang Xu. 2022. HybridRepair: towards annotation-efficient repair for deep learning models. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. Association for Computing Machinery, New York, NY, USA, 227–238. DOI: https://doi.org/10.1145/3533767.3534408

[66] Yu Li, Min Li, Qiuxia Lai, Yannan Liu, and Qiang Xu. 2021. Testrank: bringing order into unlabeled test instances for deep learning tasks. In *Proceedings of the Advances in Neural Information Processing Systems - Volume 34*. 20874–20886. Retrieved from https://proceedings.neurips.cc/paper/2021/hash/ae78510109d46b0a6eef9820a4ca95d6-Abstract.html

[67] Yuechen Li, Hanyu Pei, Linzhi Huang, and Beibei Yin. 2022. A distance-based dynamic random testing strategy for natural language processing DNN models. In *Proceedings of the 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security*. IEEE, Piscataway, NJ, USA, 842–853. DOI: https://doi.org/10.1109/QRS57517.2022.00089

[68] Zeju Li, Konstantinos Kamnitsas, Mobarakol Islam, Chen Chen, and Ben Glocker. 2022. Estimating model performance under domain shifts with class-specific confidence scores. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Berlin, Germany, 693–703.

[69] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, and Jian Lü. 2019. 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*. Association for Computing Machinery, New York, NY, USA, 499–509. DOI: https://doi.org/10.1145/3338906.3338930

[70] Zongjie Li, Chaozheng Wang, Zhibo Liu, Haoxuan Wang, Dong Chen, Shuai Wang, and Cuiyun Gao. 2023. Cctest: testing and repairing code completion systems. In *Proceedings of the 45th International Conference on Software Engineering*. IEEE Press, Piscataway, NJ, USA, 1238–1250. DOI: https://doi.org/10.1109/ICSE48619.2023.00110

[71] Zixi Liu, Yang Feng, Yining Yin, and Zhenyu Chen. 2022. DeepState: selecting test suites to enhance the robustness of recurrent neural networks. In *Proceedings of the 44th International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 598–609. DOI: https://doi.org/10.1145/3510003.3510231

[72] Yuzhe Lu, Zhenlin Wang, Runtian Zhai, Soheil Kolouri, Joseph Campbell, and Katia P. Sycara. 2023. Predicting out-of-distribution error with confidence optimal transport. In *ICLR 2023 Workshop on Pitfalls of Limited Data and Computation for Trustworthy ML (Kigali, Rwanda)*. OpenReview.net, Online, 1–8.

[73] Julia Lust and Alexandru P. Condurache. 2022. Efficient detection of adversarial, out-of-distribution and other misclassified samples. *Neurocomputing* 470 (2022), 335–343. DOI: https://doi.org/10.1007/978-3-031-16449-1_66

[74] Lei Ma, Felix Juefei-Xu, Minhui Xue, Qiang Hu, Sen Chen, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, and Simon See. 2018. Secure deep learning engineering: A software quality assurance perspective. arXiv:1810.04538. Retrieved from https://arxiv.org/pdf/1810.04538

[75] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 175–186. DOI: https://doi.org/10.1145/3236024.3236082

[76] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology* 30, 2 (2021), 1–22. DOI: https://doi.org/10.1145/3417330

[77] Yu-Seung Ma, Shin Yoo, and Taeho Kim. 2021. Selecting test inputs for DNNs using differential testing with subspecialized model instances. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 1467–1470. DOI: https://doi.org/10.1145/3468264.3473131

[78] Omid Madani, David Pennock, and Gary Flake. 2004. Co-validation: using model disagreement on unlabeled data to validate classification algorithms. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 17. MIT Press, Vancouver, British Columbia, Canada, 1–8.

[79] Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, 7047–7058.

[80] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. 2018. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 5419–5427. DOI: https://doi.org/10.1109/CVPR.2018.00568

[81] Charles H. Martin, Tongsu Peng, and Michael W. Mahoney. 2021. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications* 12, 1 (2021), 4122. DOI: https://doi.org/10.1038/s41467-021-24025-8

[82] Satoshi Masuda, Kohichi Ono, Toshiaki Yasue, and Nobuhiro Hosokawa. 2018. A survey of software quality for machine learning applications. In *Proceedings of the 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops*. IEEE, Piscataway, NJ, USA, 279–284. DOI: https://doi.org/10.1109/ICSTW.2018.00061

[83] Larry Medsker and Lakhmi C. Jain. 1999. *Recurrent Neural Networks: Design and Applications* (1st ed.). CRC Press, Inc., USA.

[84] Linghan Meng, Yanhui Li, Lin Chen, Zhi Wang, Di Wu, Yuming Zhou, and Baowen Xu. 2021. Measuring discrimination to boost comparative testing for multiple deep learning models. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. IEEE, Piscataway, NJ, USA, 385–396. DOI: https://doi.org/10.1109/ICSE43902.2021.00045

[85] Vasilii Mosin, Miroslaw Staron, Darko Durisic, Francisco Gomes de Oliveira Neto, Sushant Kumar Pandey, and Ashok Chaitanya Koppisetty. 2022. Comparing input prioritization techniques for testing deep learning algorithms. In *Proceedings of the 48th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE Computer Society, Los Alamitos, CA, USA, 76–83. DOI: https://doi.org/10.1109/SEAA56994.2022.00020

[86] Zhonghao Pan, Shan Zhou, Jianmin Wang, Jinbo Wang, Jiao Jia, and Yang Feng. 2022. Test case prioritization for deep neural networks. In *Proceedings of the 9th International Conference on Dependable Systems and Their Applications* . IEEE, Piscataway, NJ, USA, 624–628. DOI: https://doi.org/10.1109/DSA56465.2022.00089

[87] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2019. Deepxplore: automated whitebox testing of deep learning systems. , 9 pages. DOI: https://doi.org/10.1145/3361566

[88] Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, et al. 2021. Codenet: a large-scale ai for code dataset for learning a diversity of coding tasks. arXiv:2105.12655. Retrieved from https://arxiv.org/pdf/2105.12655

[89] Xin Qiu and Risto Miikkulainen. 2022. Detecting misclassification errors in neural networks with a gaussian process model. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Cambridge University Press, Cambridge, UK, 8017–8027.

[90] Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering* 25 (2020), 5193–5254. DOI: https://doi.org/10.1007/s10664-020-09881-0

[91] Gregg Rothermel and Mary Jean Harrold. 1997. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology* 6, 2 (1997), 173–210. DOI: https://doi.org/10.1145/248233.248262

[92] Peter J. Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20 (1987), 53–65. DOI: https://doi.org/10.1016/0377-0427(87)90125-7

[93] Murat Sensoy, Maryam Saleki, Simon Julier, Reyhan Aydogan, and John Reid. 2021. Misclassification risk and uncertainty quantification in deep classifiers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. IEEE Computer Society, Los Alamitos, CA, USA, 2483–2491. DOI: https://doi.org/10.1109/WACV48630.2021.00253

[94] Weijun Shen, Yanhui Li, Lin Chen, Yuanlei Han, Yuming Zhou, and Baowen Xu. 2021. Multiple-boundary clustering and prioritization to promote neural network retraining. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. Association for Computing Machinery, New York, NY, USA, 410–422. DOI: https://doi.org/10.1145/3324884.3416621

[95] Ying Shi, Beibei Yin, Zheng Zheng, and Tiancheng Li. 2021. An empirical study on test case prioritization metrics for deep neural networks. In *Proceedings of the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security*. IEEE, Piscataway, NJ, USA, 157–166. DOI: https://doi.org/10.1109/QRS54544.2021.00027

[96] Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. 2022. A unified framework for adversarial attack and defense in constrained feature space. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (Mess Wien, Vienna, Austria). International Joint Conferences on Artificial Intelligence Organization*, 1313–1319. DOI: https://doi.org/10.24963/ijcai.2022/183

[97] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. 2020. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 359–371. DOI: https://doi.org/10.1145/3377811.3380353

[98] Xiaoxiao Sun, Yunzhong Hou, Weijian Deng, Hongdong Li, and Liang Zheng. 2021. Ranking models in unlabeled new environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE Computer Society, Los Alamitos, CA, USA, 11741–11751. DOI: https://doi.org/10.1109/ICCV48922.2021.01155

[99] Xiaoxiao Sun, Yunzhong Hou, Hongdong Li, and Liang Zheng. 2021. Label-free model evaluation with semi-structured dataset representations. arXiv:2112.00694. Retrieved from https://arxiv.org/pdf/2112.00694

[100] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Piscataway, NJ, USA, 2818–2826. DOI: https://doi.org/10.1109/CVPR.2016.308

[101] Yali Tao, Chuanqi Tao, Hongjing Guo, and Bohan Li. 2022. TPFL: test input prioritization for deep neural networks based on fault localization. In *Proceedings of the International Conference on Advanced Data Mining and Applications*. Springer, Berlin, Germany, 368–383. DOI: https://doi.org/10.1007/978-3-031-22064-7_27

[102] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. 2021. Predicting neural network accuracy from weights. arXiv:2002.11448. Retrieved from https://arxiv.org/pdf/2002.11448

[103] Artem Vazhentsev, Gleb Kuzmin, Artem Shelmanov, Akim Tsvigun, Evgenii Tsymbalov, Kirill Fedyanin, Maxim Panov, Alexander Panchenko, Gleb Gusev, Mikhail Burtsev, Manvel Avetisian, and Leonid Zhukov. 2022. Uncertainty estimation of transformer predictions for misclassification detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 8237–8252. DOI: https://doi.org/10.18653/v1/2022.acl-long.566

[104] Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. 2020. Dissector: input validation for deep learning applications by crossing-layer dissection. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. Association for Computing Machinery, New York, NY, USA, 727–738. DOI: https://doi.org/10.1145/3377811.3380379

[105] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. RobOT: robustness-oriented testing for deep learning systems. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering*. IEEE, Piscataway, NJ, USA, 300–311. DOI: https://doi.org/10.1109/ICSE43902.2021.00038

[106] Zhiyu Wang, Sihan Xu, Xiangrui Cai, and Hua Ji. 2020. Test input selection for deep neural networks. *Journal of Physics: Conference Series* 1693, 1 (2020), 012017.

[107] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering*. IEEE, Piscataway, NJ, USA, 397–409. DOI: https://doi.org/10.1109/ICSE43902.2021.00046

[108] Zhengyuan Wei, Haipeng Wang, Imran Ashraf, and W. K. Chan. 2022. Predictive mutation analysis of test case prioritization for deep neural networks. In *Proceedings of the IEEE 22nd International Conference on Software Quality, Reliability and Security*. IEEE, Piscataway, NJ, USA, 682–693. DOI: https://doi.org/10.1109/QRS57517.2022.00074

[109] Sanford Weisberg. 2005. *Applied Linear Regression*. Vol. 528. John Wiley & Sons, Hoboken, NJ, USA. DOI: https://doi.org/10.1002/0471704091

[110] Michael Weiss, Rwiddhi Chakraborty, and Paolo Tonella. 2021. A review and refinement of surprise adequacy. In *Proceedings of the 2021 IEEE/ACM 3rd International Workshop on Deep Learning for Testing and Testing for Deep Learning*. IEEE, Piscataway, NJ, USA, 17–24. DOI: https://doi.org/10.1109/DeepTest52559.2021.00009

[111] Michael Weiss and Paolo Tonella. 2022. 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*. Association for Computing Machinery, New York, NY, USA, 139–150. DOI: https://doi.org/10.1145/3533767.3534375

[112] Xiaoxue Wu, Jinjin Shen, Wei Zheng, Lidan Lin, Yulei Sui, and Abubakar Omari Abdallah Semasaba. 2024. Rnntcs: a test case selection method for recurrent neural networks. *Knowledge-Based Systems* 279, C (2024), 15 pages. DOI: https://doi.org/10.1016/j.knosys.2023.110955

[113] Renchunzi Xie, Hongxin Wei, Yuzhou Cao, Lei Feng, and Bo An. 2023. On the importance of feature separability in predicting out-of-distribution error. In *Proceedings of the 37th Conference on Neural Information Processing Systems*. OpenReview.net, Online, 1–18.

[114] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. Association for Computing Machinery, New York, NY, USA, 146–157. DOI: https://doi.org/10.1145/3293882.3330579

[115] Xiaoyuan Xie, Pengbo Yin, and Songqiang Chen. 2022. Boosting the revealing of detected violations in deep learning testing: a diversity-guided method. In *Proceedings of the 37th IEEE/ACM International Conference on*

Automated Software Engineering. Association for Computing Machinery, New York, NY, USA, Article 17, 13 pages. DOI: https://doi.org/10.1145/3551349.3556919

[116] Rongjie Yan, Yuhang Chen, Hongyu Gao, and Jun Yan. 2022. Test case prioritization with neuron valuation based pattern. *Science of Computer Programming* 215, C (March 2022), 102761. DOI: https://doi.org/10.1016/j.scico.2021.102761

[117] Zhou Yang, Jieke Shi, Muhammad Hilmi Asyrofi, Bowen Xu, Xin Zhou, DongGyun Han, and David Lo. 2023. Prioritizing speech test cases. arXiv:2302.00330. Retrieved from https://arxiv.org/pdf/2302.00330

[118] Yaodong Yu, Zitong Yang, Alexander Wei, Yi Ma, and Jacob Steinhardt. 2022. Predicting out-of-distribution error with the projection norm. In *Proceedings of the International Conference on Machine Learning*. PMLR, Brookline, MA, USA, 25721–25746.

[119] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue. 2014. Droid-sec: deep learning in android malware detection. In *Proceedings of the 2014 ACM conference on SIGCOMM*. Association for Computing Machinery, New York, NY, USA, 371–372. DOI: https://doi.org/10.1145/2740070.2631434

[120] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine learning testing: survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2022), 1–36. DOI: https://doi.org/10.1109/TSE.2019.2962027

[121] Chunyu Zhao, Yanzhou Mu, Xiang Chen, Jingke Zhao, Xiaolin Ju, and Gan Wang. 2022. Can test input selection methods for deep neural network guarantee test diversity? A large-scale empirical study. *Information and Software Technology* 150, C (2022), 12 pages. DOI: https://doi.org/10.1016/j.infsof.2022.106982

[122] Haibin Zheng, Jinyin Chen, and Haibo Jin. 2023. CertPri: certifiable prioritization for deep neural networks via movement cost in feature space. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, Los Alamitos, CA, USA, 1–13. DOI: https://doi.org/10.1109/ASE56229.2023.00126

[123] Yan Zheng, Xiaofei Xie, Ting Su, Lei Ma, Jianye Hao, Zhaopeng Meng, Yang Liu, Ruimin Shen, Yingfeng Chen, and Changjie Fan. 2019. Wuji: automatic online combat game testing using evolutionary deep reinforcement learning. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, Piscataway, NJ, USA, 772–784. DOI: https://doi.org/10.1109/ASE.2019.00077

[124] Jianyi Zhou, Feng Li, Jinhao Dong, Hongyu Zhang, and Dan Hao. 2020. Cost-effective testing of a deep learning model through input reduction. In *Proceedings of the IEEE 31st International Symposium on Software Reliability Engineering*. IEEE Computer Society, Los Alamitos, CA, USA, 289–300. DOI: https://doi.org/10.1109/ISSRE5003.2020.00035

[125] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. 2022. Rethinking confidence calibration for failure prediction. In *Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23âĂŞ27, 2022, Proceedings, Part XXV*. Springer, Berlin, Germany, 518–536. Retrieved from https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136850512.pdf

[126] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. 2023. OpenMix: exploring outlier samples for misclassification detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 12074–12083. DOI: https://doi.org/10.1109/CVPR52729.2023.01162

[127] Fangzhe Zhu, Ye Zhao, Zhengqiong Liu, and Xueliang Liu. 2023. Label-free model evaluation with out-of-distribution detection. *Applied Sciences* 13, 8 (2023), 5056. DOI: https://doi.org/10.3390/app13085056

Received 19 October 2023; revised 14 January 2024; accepted 17 January 2024