LaF: Labeling-free Model Selection for Automated Deep Neural Network Reusing

QIANG HU, University of Luxembourg, Luxembourg
YUEJUN GUO, Luxembourg Institute of Science and Technology, Luxembourg
XIAOFEI XIE, Singapore Management University, Singapore
MAXIME CORDY, MIKE PAPADAKIS, and YVES LE TRAON, University of Luxembourg, Luxembourg

Applying deep learning (DL) to science is a new trend in recent years, which leads DL engineering to become an important problem. Although training data preparation, model architecture design, and model training are the normal processes to build DL models, all of them are complex and costly. Therefore, reusing the open-sourced pre-trained model is a practical way to bypass this hurdle for developers. Given a specific task, developers can collect massive pre-trained deep neural networks from public sources for reusing. However, testing the performance (e.g., accuracy and robustness) of multiple deep neural networks (DNNs) and recommending which model should be used is challenging regarding the scarcity of labeled data and the demand for domain expertise. In this article, we propose a labeling-free (LaF) model selection approach to overcome the limitations of labeling efforts for automated model reusing. The main idea is to statistically learn a Bayesian model to infer the models' specialty only based on predicted labels. We evaluate LaF using nine benchmark datasets, including image, text, and source code, and 165 DNNs, considering both the accuracy and robustness of models. The experimental results demonstrate that LaF outperforms the baseline methods by up to 0.74 and 0.53 on Spearman's correlation and Kendall's τ, respectively.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; • **Software and its engineering** → **Software development process management;**

Additional Key Words and Phrases: Deep neural network, model selection, labeling-free, Bayesian model

ACM Reference format:
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 Trans. Softw. Eng. Methodol.* 33, 1, Article 25 (November 2023), 28 pages.
https://doi.org/10.1145/3611666

Q. Hu and Y. Guo contributed equally to this work.
Y. Guo's work was partially done while at SnT, University of Luxembourg.
This work is supported by the Luxembourg National Research Funds (FNR) through CORE Project No. C18/IS/12669767/STELLAR/LeTraon.
Authors' addresses: Q. Hu, M. Cordy, M. Papadakis, and Y. Le Traon, University of Luxembourg, Luxembourg; emails: Qiang.Hu@uni.lu, Maxime.Cordy@uni.lu, Mike.Papadakis@uni.lu, YvesLe.Traon@uni.lu; Y. Guo, Luxembourg Institute of Science and Technology, Luxembourg; email: yuejun.guo@list.lu; X. Xie, Singapore Management University, Singapore; email: xiaofei.xfxie@gmail.com.

1049-331X/2023/11-ART25 $15.00
https://doi.org/10.1145/3611666

1 INTRODUCTION

Deep learning (DL) is helping solve all sorts of real-world problems in various domains, such as computer vision [42], **natural language processing (NLP)** [38], code understanding [44], and autonomous driving [24]. Due to the outstanding performance of **deep neural networks (DNNs)**, **software engineering (SE)** researchers have attempted to apply DNNs to solve various SE tasks, such as source code processing [9, 44], automatic software testing [57, 58], and GUI designs [56]. Building machine learning (or deep learning) systems generally requires model architecture design and data preparation, model training, and model deployment and maintenance, known as **machine learning operations (MLOps)** [5]. However, since (1) designing new DNN architectures requires tremendous experimentation and DL knowledge, (2) preparing massive labeled training data is labor-intensive, and (3) training the DL model needs a considerable amount of time and resources, developers generally reuse generic and publicly available pre-trained models to solve their given task. Ultimately, we expect model reuse to become the norm in DL-based SE, just like it has been in other fields like NLP.

Due to the convenience of multiple public open sources such as GitHub [4], TensorFlow datasets [7], and Hugging Face [52], engineers can today access a massive number of DNNs, in the form of either pre-trained model files (e.g., .h5 and .pth). While this is practical, there is no prior evidence of which model will be capable of solving the targeted task the most effectively. Indeed, different models have been developed by different contributors and in different development settings, while they have been evaluated in unknown or incomparable testing conditions. For example, only 21 of the 165 DNNs we collected and evaluated in this article have a performance report.

Model selection is a process of determining the best fit from multiple models for a given test set. Selecting candidate models for the targeted task raises two challenges. First, test data that can be found in the real world (e.g., source code from public repositories) are generally unlabelled [48], while data labeling requires significant manual effort. For example, the AIZU online programming challenge [1] receives submissions in different programming languages all the time. A Java developer can easily annotate the source code in Java but may have difficulties with other programming languages, such as C++. The effective selection of pre-trained models to solve various tasks and overcome the challenge of insufficient domain knowledge, therefore, requires a precise and efficient method to **compare and rank** DNN candidates **without** data labels.

An additional challenge originates from the fact that the chosen models will likely be confronted with data that have a different distribution from the data they were trained/tested on, i.e., **out-of-distribution (OOD)** data [11, 15, 23]. For many application domains—including software [27]—the distribution shift phenomena that yield OOD data are inevitable. As a result, these models may exhibit remarkably different performances over time, which raises the concern of quality and reliability [8]. For instance, for the same dataset iWildCam (please refer to Section 4.3 for more details), two DNNs exhibit 75.74% and 76.60% accuracy on the initial test data, while their performance turns to 76.82% and 65.30%, respectively, on OOD data. In conclusion, this suggests that ideal selection methods can handle the data distribution shift problem and reliably estimate model performance on OOD data.

Such methods have been proposed by a recent approach named **sample discrimination-based selection (SDS)** [37]. SDS achieves positive model ranking results on three benchmark datasets. SDS selects a set of data to label based on the majority voting [45] and item discrimination [16]. These data are considered the most discriminative in terms of distinguishing the accuracy between DNNs. DNNs are then ranked based on their accuracy on these selected and labeled data. Figure 3 illustrates how this sampling-based approach works. Although promising, SDS still suffers from manual labeling. Besides, it has only considered ranking based on model accuracy with **in-distribution (ID)** data (i.e., data that follow the same distribution as the data the models were

trained on)—it has disregarded OOD data. Third, it has been applied to the image domain only; its effectiveness for other domains remains unclear.

In this article, we aim to overcome the above limitations and propose a **labeling-free (LaF)** model selection method, for DNNs that is effective with both ID and OOD data. Given a sample, only the predicted labels of multiple DNNs are available. The main idea of LaF is to build a Bayesian model that incorporates the data difficulty and model specialty to estimate the likelihood of a predicted label being the true label. The data difficulty implies how difficult a sample is for all DNNs to predict correctly, which is reflected by the prediction difference across multiple models. The model specialty indicates how good a model is to infer the correct labels of all samples, which is reflected by the ability to have the same predictions as the majority of models. Via optimizing the Bayesian model, we infer the model specialty to perform the model selection. The optimization is achieved by the **expectation-maximization (EM)** algorithm [14], which is efficient in finding maximum likelihood parameters (the data difficulty and model specialty in our case). To summarize, the main contributions of our work are:

(1) We propose a novel approach, LaF, for automatically ranking multiple DNN models to facilitate the reuse of DNNs from public sources.
(2) We demonstrate the effectiveness of LaF on ID and OOD data. Both the artificial and natural distribution shifts are considered.
(3) LaF is labeling-free, which makes it practical and feasible in real-world applications.
(4) We experiment on nine benchmark datasets spanning different domains including image, text, and source code with different programming languages (Java and C++). To the best of our knowledge, this is the first DNN model selection work containing datasets other than images.
(5) All the used models, datasets, and implementations of LaF and baseline methods are publicly available at https://github.com/testing-cs/LaF-model-selection.git

The rest of this article is organized as follows. Section 2 introduces preliminary knowledge behind this work. Section 3 presents the problem statement and our proposed solution. Section 4 explains our experimental design. We present the empirical results and corresponding discussions in Section 5. Section 6.2 details the threats that may affect the validity of conclusions. Section 7 reviews related work. The last section concludes our work and points out future research directions.

2 BACKGROUND

2.1 MLOps and Model Reusing

Generally, building machine learning systems needs a set of operations [5] that are depicted in the first row of Figure 1. Roughly speaking, first, developers connect their interested data and design the model architecture that is suitable to train the data. Then, developers fed the model and data into the high-performance server to tune the parameters of the model. After a model with expected performance was trained, it will be deployed (embedded) into the application and function in the wild. Finally, similar to conventional software systems, machine learning systems also need to be evolved and maintained from time to time. For the first two steps, i.e., data preparation and model design, huge human efforts and expert domain knowledge are needed to label the data and design the model. And for the model training process, expensive computing resource is necessary to handle the complex parameter tuning procedure. In conclusion, the first three steps make the whole process heavy before the model is deployed for real usage.

In practice, model reusing is a commonly adopted way to lighten **machine learning operations (MLOps)**. As shown in the second row of Figure 1, the original three operations data

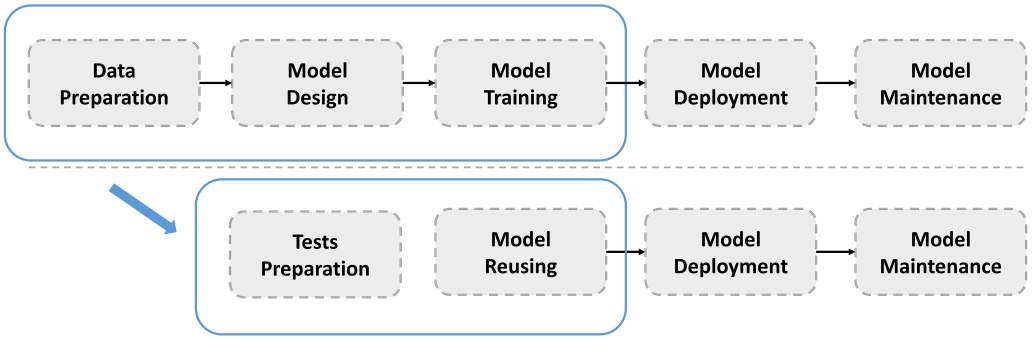


Fig. 1. MLOps with and without model reusing.

preparation, model design, and model training are replaced by the two, tests preparation and model reusing. The hidden reason for this replacement is that, nowadays, given a specific task, e.g., face recognition, we can access many well-established pre-trained models from online resources, e.g., GitHub. Thus, the straightforward way to build our machine learning system is to reuse such models. To do so, we only need to collect potential pre-trained models, prepare some test data (which can be much less than the training data) for our task, and finally, select the best model using the test data to build our system. In this way, we can reduce the human effort of training data labeling and the difficult procedure of model architecture design. Here, two important problems we need to pay attention to, (1) how to correctly obtain these massive models, when there are no models provided by others available for your specific tasks, preparing these candidate models is still a challenging problem. (2) How to select the best model among the collected models with less effort. Even though we can only use a small set of test data to select the best model, the labeling budgets still exist and make the process non-automatic. In this work, we focus the second problem on how to efficiently select the best model from the massive number of available models.

2.2 Comparison Testing and Test Selection in Deep Learning

In conventional software engineering, comparison testing [26, 46, 47] aims at figuring out the strength and weaknesses of a newly developed software product compared with existing products. The end goal is to facilitate the deployment of a product with high functionality and reliability. Recently, Meng et al. [37] re-framed "comparison testing" as testing methods that aim to compare alternative software artifacts, especially DNNs [25]. Concretely, the problem turns into how to find out the most discriminative data that can amply distinguish the difference. In their proposed sample discrimination-based selection method, the majority voting [45] is first applied to produce pseudo-labels based on which DNNs are classified to top, middle, and bottom groups following the item discrimination [16]. Via the prediction difference between the top and bottom DNNs, each data has a unique discrimination score, and the high ones are selected for the final ranking.

A close topic to comparison testing is test selection for DNN model performance estimation. The key idea of this kind of test selection is, given massive unlabeled test data and a DNN model, we estimate the performance of the model on these data by using a subset of data selected by test selection metrics. In this way, the labeling effort can be reduced and the budget can be saved. For instance, Li et al. [31] proposed the cross Entropy-based sampling method to identify the most representative data of a test set. Similarly, Chen et al. [12] developed the practical accuracy estimation. The difference is that in test selection, the objective is a single DNN, while in comparison testing, the objective is multiple DNNs. Undoubtedly, one can first approximate the performance

(a) White noise (b) Brightness (c) Location 1 (d) Location 2

Fig. 2. Image examples with artificial distribution shifts (noise, brightness, location change). Images come from Reference [27].

of each DNN by selecting its corresponding representative set and then undertake the comparison. However, this will largely increase the effort in labeling and is less practical than selecting once.

Both comparison testing and test selection can be used in the model reusing process to indicate the model with the best performance.

2.3 Distribution Shift

Distribution shift is a crucial problem in machine learning, which means the train data and test data follow different data distributions. Generally, compared to the **in-distribution data (IID)**, the model is more difficult to handle the data with distribution shift, which makes the reported performance (using IID) of the model unreliable.

Roughly speaking, there are two types of distribution shifts, artificial and natural. Artificial distribution shift mainly comes from adding artificial perturbations (corruptions) into raw data. Dan and Thomas [21] proposed to add 15 types of algorithmically generated corruptions with five levels of severity to image data to mimic realistic situations, such as noise, blur, snow, and zoom. Based on these corruptions, different benchmark datasets, such as CIFAR-10-C [21] and MNIST-C [39], have been developed for testing the robustness of DNN models. Figures 2(a) and 2(b) show two examples of artificial distribution shift—the original bird picture with two types of noises, white noise and brightness. However, natural distribution shift is usually induced by the change of environment or population and exists in raw data, such as the change of camera traps [10], new customers [40], and new repositories [27]. A recent benchmark [27] provides in-the-wild distribution shifts covering diverse data domains and applications. Figures 2(c) and 2(d) are examples of the natural distribution shift. The two pictures have the same label *cow* but the cows are captured by different positions.

3 METHODOLOGY

3.1 Problem Formulation

In this article, we are interested in the classification task. Given a C-class task over a sample space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$, where $x \in \mathcal{X}$ is an input data and $y \in \mathcal{Y}$ is its class label. Let $f : x \rightarrow y$ be a DNN that maps x to the problem domain. Given n models, $f_1, f_2, \ldots, f_n$, extracted from public sources and a set of unlabeled test data T, the problem we study is to estimate the rank of models regarding their performance on $\boldsymbol{T} = \{x_1, x_2, \ldots, x_m\}$. Figure 3 illustrates the workflow.

We assume that compared to the time cost of labeling data, the time cost of computation is negligible. For example, labeling code data from only four libraries can take up to 600 man-hours [59], whereas LaF and the competing methods produce a ranking with much less time (e.g., ranking Java250-based models with less than 15 min). To this end, we propose to tackle the ranking problem by only querying the predictions, which is highly applicable in practical scenarios. Remarkably,

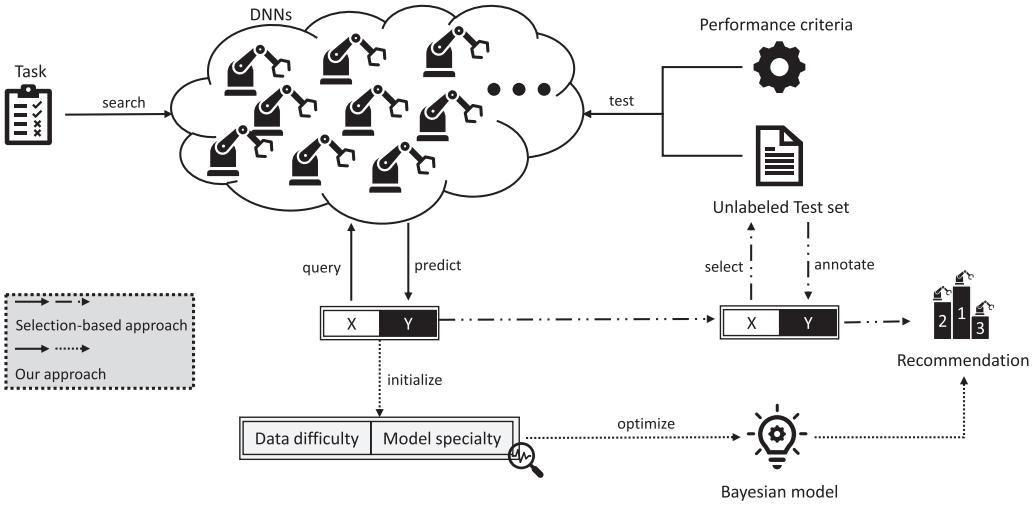


Fig. 3. Illustration of the studied problem.

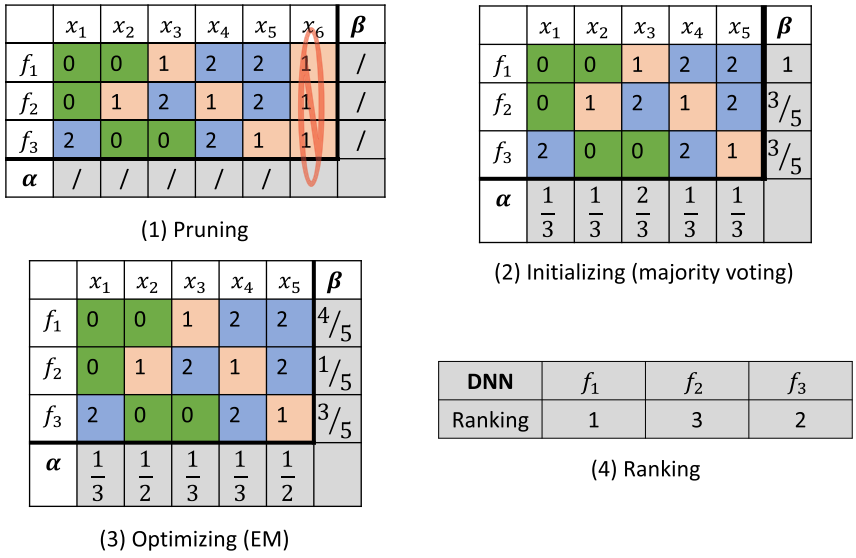
in this article, we consider the performance of both accuracy and robustness. The accuracy is the correctness ratio of prediction on ID data. The robustness is the correctness ratio of OOD data.

3.2 Motivating Example

Figure 4 gives an example of LaF. In this simple three-class example, there are three DNN models (f_1, f_2, f_3) given six unlabeled samples $(x_1, x_2, \ldots, x_6)$. The goal is to rank the three models concerning their accuracy on these samples in the absence of true labels. First, we compute the predicted label of each sample by each model and remove x_6 where all the models have the same prediction. Second, we initialize the two parameters, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, contained in our approach. $\boldsymbol{\alpha}$ refers to how difficult a sample is for all models to predict the correct label. $\boldsymbol{\beta}$ indicates how good a model is to output the correct labels of all samples. Initially, we use the simplest and most commonly used majority voting heuristic [45] to give a pseudo-label to each sample. For instance, the pseudo-label of x_1 is 0, because two (f_1, f_2) of three models predict the label as 0. $\boldsymbol{\alpha}$ is defined as the ratio of mismatched models that output a different label instead of the pseudo one. $\boldsymbol{\beta}$ is calculated as the ratio of correctly predicted samples over the entire set. Third, since the pseudo-labels are not the true labels, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ cannot truly reflect the data difficulty and model ability. We optimize these two parameters by a likelihood estimation method in the presence of true labels. Finally, based on the optimized $\boldsymbol{\beta}$ $(\frac{4}{5}, \frac{1}{5}, \frac{3}{5})$, we obtain the ranking 1, 3, and 2 for f_1, f_2, and f_3, respectively.

3.3 Our Approach: LaF

Given that no label is available in the test data, the main idea of our approach is to infer the specialties of DNNs by approximately maximizing the likelihood between the predictions and true labels via the **expectation-maximization (EM)** algorithm [14]. Let $\widetilde{\boldsymbol{Y}} = \{\widetilde{y}_{ij}\}_{1\le i\le m, 1\le j\le n}$ be the predicted labels of $\boldsymbol{T}$ and $\boldsymbol{Y} = \{y_i\}_{1\le i\le m}$ be the true labels. Here, $\widetilde{y}_{ij}$ refers to x_i and model f_j. Given the observed $\widetilde{\boldsymbol{Y}}$ and latent $\boldsymbol{Y}$ governed by unknown parameters $\boldsymbol{\theta}$, the likelihood function is defined as $L(\boldsymbol{\theta}; \widetilde{\boldsymbol{Y}}) = p(\widetilde{\boldsymbol{Y}} \mid \boldsymbol{\theta}) = \sum_{i=1}^{m} p(\widetilde{\boldsymbol{Y}}, y_i \mid \boldsymbol{\theta})$. The goal is to search the best $\boldsymbol{\theta}$ that maximizes the likelihood, in other words, the probability of observing $\widetilde{\boldsymbol{Y}}$. As for $\boldsymbol{\theta}$, inspired by Reference [51], we consider two factors, data difficulty $\boldsymbol{\alpha} = \{\alpha_i\}_{1\le i\le m}$ and model specialty $\boldsymbol{\beta} = \{\beta_j\}_{1\le j\le m}$, that



	x_1	x_2	x_3	x_4	x_5	x_6	$\boldsymbol{\beta}$
f_1	0	0	1	2	2	1	/
f_2	0	1	2	1	2	1	/
f_3	2	0	0	2	1	1	/
$\boldsymbol{\alpha}$	/	/	/	/	/		

	x_1	x_2	x_3	x_4	x_5	$\boldsymbol{\beta}$
f_1	0	0	1	2	2	1
f_2	0	1	2	1	2	$3/5$
f_3	2	0	0	2	1	$3/5$
$\boldsymbol{\alpha}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	

	x_1	x_2	x_3	x_4	x_5	$\boldsymbol{\beta}$
f_1	0	0	1	2	2	$4/5$
f_2	0	1	2	1	2	$1/5$
f_3	2	0	0	2	1	$3/5$
$\boldsymbol{\alpha}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{2}$	

DNN	f_1	f_2	f_3
Ranking	1	3	2

Fig. 4. An example of our four-step approach. It ranks three DNNs (f_1, f_2, f_3) given six unlabeled data (x_1, x_2, x_3, x_5, x_6) in a three-class classification task. Numbers (0, 1, 2) highlighted in colors are predicted labels.

influence the performance of DNNs. Namely, $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$. Algorithm 1 presents the pseudo-code of our approach.

Step 1: Pruning. Inevitably, some data will receive the same predictions by all models, which is useless for discriminating the performance and causes computational cost. For this reason, we filter these data without losing any information for ranking and obtain a smaller set $\boldsymbol{T'}$ (Lines 1–6 in Algorithm 1).

Step 2: Initializing. First, for each data x_i, since we do not have its ground-truth label, we use the voted label by DNNs as the replacement, namely, $y'_i = \text{mode}(\{\widetilde{y}_{ij}\}_{1 \le j \le n})$ (Lines 7–9). Next, α_i is the number of DNNs that gives a different label from the pseudo-label and β_j is the accuracy based on pseudo-labels (Lines 10–15). Formally, the definitions are

$$\alpha_i = \frac{\left\{\widetilde{y}_{ij} \mid \widetilde{y}_{ij} \neq y'_i, 1 \le j \le n\right\}}{n}, \beta_j = \frac{\left\{\widetilde{y}_{ij} \mid \widetilde{y}_{ij} = y'_i, 1 \le i \le |\boldsymbol{T'}|\right\}}{|\boldsymbol{T'}|}. \tag{1}$$

Step 3: Optimizing. To obtain the best results of the model parameters (data difficulty and model specialty), we use the EM algorithm [14], which has been proven to be a powerful tool for estimating the parameters of statistical models to solve the optimization problem. Specifically, EM performs an expectation (E) step and a maximization (M) step (Lines 16–26) in the optimization process. In the E-step, it estimates the expected value of the log-likelihood:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta_{last}}) &= \mathrm{E}\left[\log L\left(\boldsymbol{\theta}; \widetilde{\boldsymbol{Y}}, \boldsymbol{Y}\right)\right] \\ &= \sum_{i=1}^{|\boldsymbol{T'}|} \mathrm{E}\left[\log\left(p\left(y_i\right)\right)\right] + \sum_{i=1}^{|\boldsymbol{T'}|} \sum_{j=1}^{n} \mathrm{E}\left[p\left(\widetilde{y}_{ij} \mid y_i, \alpha_i, \beta_j\right)\right], \end{aligned} \tag{2}$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\boldsymbol{\theta_{last}}$ is from the last E-step. For the computation, we use the definition from Reference [51], where $p(\widetilde{y}_{ij} = y_j \mid \alpha_i, \beta_j) = \frac{1}{1+e^{-\alpha_i \beta_j}}$. Besides, as $\widetilde{y}_{ij}$ and $\boldsymbol{\beta}$ are independent

ALGORITHM 1: LaF: Labeling-free comparison testing

Input : $\{f_1, f_2, \ldots, f_n\}$: DNNs for comparison
$\boldsymbol{T} = \{x_1, x_2, \ldots, x_m\}$: test set
Γ: performance criterion
Output : $\{r'(f_1), r'(f_2), \ldots, r'(f_n)\}$: Rank of DNNs

/* Step1: Pruning */
1 $\boldsymbol{T'} = \{\}$
2 **for** $i = 1 \rightarrow m$ **do**
3 | **if** $|\{\{\widetilde{y}_{ij}\}_{1 \le j \le n}\}| > 1$ **then**
4 | | $\boldsymbol{T'} \leftarrow x_i$; // $\widetilde{y}_{ij}$ is the predicted label by f_j
5 | **end**
6 **end**
/* Step 2: Initializing */
7 **for** $i = 1 \rightarrow |\boldsymbol{T'}|$ **do**
8 | $y'_i = \text{mode}(\{\widetilde{y}_{ij}\}_{1 \le j \le n})$; // Majority voting
9 **end**
10 **for** $i = 1 \rightarrow |\boldsymbol{T'}|$ **do**
11 | $\alpha_i = \frac{\{\widetilde{y}_{ij} | \widetilde{y}_{ij} \neq y'_i, 1 \le j \le n\}}{n}$; // Data difficulty
12 **end**
13 **for** $j = 1 \rightarrow n$ **do**
14 | $\beta_j = \frac{\{\widetilde{y}_{ij} | \widetilde{y}_{ij} = y'_i, 1 \le i \le |\boldsymbol{T'}|\}}{|\boldsymbol{T'}|}$; // Model specialty
15 **end**
/* Step 3: Optimizing */
16 $\boldsymbol{\alpha}_{last} = \{\alpha_i\}_{1 \le i \le |\boldsymbol{T'}|}$
17 $\boldsymbol{\beta}_{last} = \{\beta_j\}_{1 \le j \le n}$
18 $Q_{last} = computeQ(\boldsymbol{\alpha}_{last}, \boldsymbol{\beta}_{last})$; // *ComputeQ* estimates the log likelihood based on Equation (2)
19 $\boldsymbol{\alpha}, \boldsymbol{\beta} = gradientAscent(\boldsymbol{\alpha}_{last}, \boldsymbol{\beta}_{last}, Q_{last})$
20 $Q = computeQ(\boldsymbol{\alpha}, \boldsymbol{\beta})$
21 **while** $|\frac{Q - Q_{last}}{Q_{last}}| > 1E-5$ **do**
22 | $Q_{last} = Q$
23 | $\boldsymbol{\alpha}_{last}, \boldsymbol{\beta}_{last} = \boldsymbol{\alpha}, \boldsymbol{\beta}$
24 | $\boldsymbol{\alpha}, \boldsymbol{\beta} = gradientAscent(\boldsymbol{\alpha}_{last}, \boldsymbol{\beta}_{last}, Q_{last})$
25 | $Q = computeQ(\boldsymbol{\alpha}, \boldsymbol{\beta})$
26 **end**
/* Step 4: Ranking */
27 $r'(f_1), r'(f_2), \ldots, r'(f_n) = Sort(\boldsymbol{\beta})$
28 **return** $\{r'(f_1), r'(f_2), \ldots, r'(f_n)\}$

given $\boldsymbol{\alpha}$, $p(y_i) = p(y_i \mid \boldsymbol{\alpha}, \beta_j)$. Remarkably, y_i represents the true label of a sample. In the ranking problem, y_i is absent but the probability of taking it as a true label can be inferred by $p(y_i \mid \boldsymbol{\alpha}, \beta_j)$.

In the M-step, the gradient ascent is applied to search for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ that maximize Q:

$$\boldsymbol{\theta_{new}} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta_{last}}), \tag{3}$$

where $\boldsymbol{\theta_{new}}$ is the updated parameters for the next iteration.

Step 4: Ranking. Finally, as $\boldsymbol{\beta}$ well estimate the abilities of each DNN given the observed labels, we use this vector to rank DNNs (Line 27). A high specialty indicates a good performance on the data.

Overall, LaF is a parameter-free method where the influence factors in LaF—the initialization of data difficulty and model specialty are automatically decided by the majority voting.

4 EXPERIMENTAL SETUP

4.1 Implementation

All experiments were conducted on a high-performance computer cluster and each cluster node runs a 2.6 GHz Intel Xeon Gold 6132 CPU with an NVIDIA Tesla V100 16G SXM2 GPU. We implement the proposed approach and baseline methods based on the state-of-the-art frameworks, Tensorflow 2.3.0 and PyTorch 1.6.0. For the artificial data distribution shift, we consider 3 benchmark datasets where each includes 15 types of natural corruption with 5 levels of severity. In total, we test on $3 * 15 * 5 = 225$ datasets with the artificial distribution shift. We report the average results on corrupted data for baseline methods. The entire results corroborate our findings and are available on our companion project website [3].

4.2 Research Questions

In this study, we focus on the following four research questions:

- RQ1 (**effectiveness given ID test**): How is LaF ranking multiple DNNs given ID test data?
- RQ2 (**effectiveness under distribution shift**): How is LaF ranking multiple DNNs given OOD test data (including artificial and natural distribution shifts)?
- RQ3 (**ablation study**): How does each component contribute to LaF?
- RQ4 (**impact factors of LaF**): What is the impact of model quality, model diversity, model number, and input number on ranking DNNs?

The first two research questions successively evaluate the effectiveness of our proposed solution given test data with and without distribution shift. The second one also tends to show how flexible and practical LaF is in real-world applications, especially by the test data with natural distribution shifts. The third research question studies the contribution of each component in LaF. The last one investigates the impact factors that may affect the ranking performance.

4.3 Datasets and DNNs

Datasets. We choose seven datasets, MNIST [29], Fashion-MNIST [53], CIFAR-10 [28], iWildCam [10], Amazon [40], Java250, and C++1000 [44] that are widely studied in previous work. These datasets cover the image (first four), text (Amazon), and source code (Java250 and C++1000) domains. The test data that follow the same distribution as the training set are the so-called in-distribution (ID) data. The test data with data distribution shift are seem as out-of-distribution (OOD) data. In our work, we consider two types of distribution shifts: artificial and natural. For the artificial distribution shift, we use two benchmark datasets, MNIST-C [39] and CIFAR-10-C [21], for MNIST and CIFAR-10, respectively. We follow the official implementation of MNIST-C and select 15 types of corruption that are also used in CIFAR-10-C for our experiments, such as Gaussian noise, shot noise, impulse noise, defocus blur, frosted glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic, pixelate, and jpeg. Besides, each type of corruption has 5 levels of severity. In total, our used MNIST-C and CIFAR-10 datasets contain 75 different test sets. However, there are no existing benchmark datasets for the distribution shift version of Fashion-MNIST, we simply follow the same setting of MNIST-C and use their script to generate Fashion-MNIST-C. For the natural distribution shift, we use two datasets for iWildCam and Amazon, respectively, from a recent-published benchmark, WILDS [27]. The distribution shift comes from new came traps in iWildCam and new users in Amazon. For Java250, we manually collect the OOD dataset based on

Table 1. Summary of Datasets

Dataset	Domain	Data Type	#Classes	#ID	#OOD	Distribution Shift
MNIST	Computer vision	Image of handwritten digits	10	10,000	750,000	Artificial
Fashion-MNIST	Computer vision	Image of fashion products	10	10,000	75,000	Artificial
CIFAR-10	Computer vision	Image of animals and vehicles	10	10,000	750,000	Artificial
iWildCam	Computer vision	Image of wildlife	182	8,154	42,791	Natural
Amazon	Natural language processing	Text of comments	5	46,950	100,050	Natural
Java250	Source code analysis	Source in Java	250	15,000	15,000	Natural
C++1000	Source code analysis	Source code in C++	1,000	99.997	—	—

"#ID" is the number of in-distribution test data. "#OOD" is the number of out-of-distribution test data with artificial or natural distribution shifts.

Table 2. Summary of Models

Dataset	#DNNs	#Parameters	Accuracy (%)	Robustness (%)
MNIST	30	7,206–3,274,634	85.27–99.54	MNIST-C
Fashion-MNIST	25	258,826–1,256,080	90.09–93.38	Fashion-MNIST-C
CIFAR-10	30	62,006–45,294,194	69.90–95.92	CIFAR-10-C
iWildCam	20	7,224,054–23,960,630	75.72–77.26	65.30–76.82
Amazon	20	3,223,255–12,861,655	56.06–59.13	38.24–56.94
Java250	20	2,927,290	64.73–87.39	57.24–81.67
C++1000	20	4,409,288	71.39–92.10	—

"#DNNs" is the number of DNNs collected for each dataset. "#Parameters" shows the minimum and the maximum number of parameters of collected DNNs. "Accuracy" and "Robustness" lists the lowest and highest accuracy and robustness on test data with and without distribution shift, respectively. MNIST-C and CIFAR-10-C are two benchmark datasets and corresponding robustness is summarized in Table A2.

the definition in WILDS that the distribution shift of source code comes from new repositories. For each class in Java250, we extract java files from Reference [44] under the constraint that the corresponding users do not exist in ID data. Unfortunately, we did not find suitable OOD data for C++1000. Table 1 lists the details of datasets.

DNNs. From Github, we collect, in total, 165 models, 30 for MNIST, 25 for Fashion-MNIST, 30 for CIFAR-10, 20 for iWildCam, 20 for Amazon, 20 for Java250, and 20 for C++1000. In concrete, the models of MNIST and CIFAR-10 are extracted using the GitHub links in Reference [37]. The 25 models of Fashion-MNIST are extracted from References [2, 37]. For iWildCam and Amazon, we train models using the implementation in the benchmark WILDS [27]. For Java250 and C++1000, we train models using the implementation in the benchmark Project CodeNet [44] by different optimizers (sgd, rmsprop, adam, adadelta, adagrad, adamax, nadam, ftrl) and architectures (basic, doublePool). Table 2 presents the accuracy and robustness of DNNs on ID test data and OOD test data with natural distribution shift, respectively.

4.4 Baseline Methods

In our study, we compare LaF to four baseline methods, including one labeling-free method (CRC) and three sampling-based methods (random sampling, SDS, and CES). For conducting sampling-based methods, we set the labeling budgets from the number of DNNs (i.e., 30 DNNs in MNIST) to 180 at intervals of 5. Here, we follow the previous work [37] to set the maximum labeling budget as 180. Moreover, since our method is labeling-free, we tend to compare LaF with sampling-based methods that use as little labeling budget as possible, which is the number of DNNs.

Consistent relative confidence (CRC) [34] is a recently proposed method that only uses output probabilities to rank models. Roughly speaking, given a set of models and input, CRC assumes

that a model has higher confidence to input should have better performance. Here, the confidence is calculated by the maximum probability of outputs.

Random sampling is a basic and model-independent method for data selection where each data has an equal probability to be considered. A subset of data is randomly selected and annotated to rank DNNs.

Sample discrimination-based selection (SDS) [37] is the state-of-the-art approach in ranking multiple DNNs with respect to accuracy. Following [37], among data in the top 25% with high discrimination scores, we randomly select a given budget of data to label and annotate to perform the ranking task.

Cross Entropy-based Sampling (CES) [31] is designed to select a set of representative data to approximate the actual performance given a single DNN. We follow the same procedure as Reference [37] to adapt CES for multi-DNN comparison.

The comparison we conduct is fair, because all methods used the same amount of data information for ranking. All the considered methods need to access the prediction of all unlabeled data once. For sampling-based models, they use this prediction to perform data selection and label a subset of data to rank the models. For labeling-free methods, the ranking is conducted based only on the predictions and no further labeling budgets are required. Due to the random manner in the sampling methodology, each experiment of the baseline methods is repeated 50 times, and we report the average result.

4.5 Evaluation Measures

To evaluate the effectiveness of each method, we follow the baseline work [37] and apply the statistical analysis, Spearman's rank-order correlation [13], and Jaccard similarity [37]. The first one evaluates the general ranking of all models, while the last one specifically estimates the ranking of top-k DNNs. In addition, we add the evaluation on Kendall's τ rank correlation [13]. Similar to Spearman's rank-order correlation, Kendall's τ measures the non-parametric rank correlation. However, Kendall's τ calculates based on concordant and discordant pairs and is insensitive to errors (if any) in data. By contrast, Spearman's rank-order correlation calculates based on deviations and is more sensitive to errors (if any) in data.

Given n DNNs, $f_1, f_2, \ldots, f_n$, let $r(f_1), r(f_2), \ldots, r(f_n)$ be the ground-truth ranking and $r'(f_1)$, $r'(f_2), \ldots, r'(f_n)$ be the estimated ranking. The Spearman's rank-order correlation coefficient is computed as

$$\rho = \frac{n \sum_{i=1}^{n} r(f_i) r'(f_i) - (\sum_{i=1}^{n} r(f_i))(\sum_{i=1}^{n} r'(f_i))}{\sqrt{[n \sum_{i=1}^{n} r(f_i)^2 - (\sum_{i=1}^{n} r(f_i))^2][n \sum_{i=1}^{n} r'(f_i)^2 - (\sum_{i=1}^{n} r'(f_i))^2]}}. \tag{4}$$

A large ρ indicates that the correlation between the ground truth and estimation is strong.

Kendall's τ is

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T)\,(P + Q + U)}}, \tag{5}$$

where P and Q are the numbers of ordered and disordered pairs in $\{r(f_i), r'(f_i)\}$, respectively. T and U are the numbers of ties in $\{r(f_i)\}$ and $\{r'(f_i)\}$, respectively. A large τ indicates a strong agreement between the ground truth and estimation.

Meng et al. proposed to apply the Jaccard similarity for measuring the similarity between the top-k models. The similarity coefficient is defined as

$$J_k = \frac{|\,\{f_i \mid r\,(f_i) <= k\} \cap \{f_i \mid r'\,(f_i) <= k\}\,|}{|\,\{f_i \mid r\,(f_i) <= k\} \cup \{f_i \mid r'\,(f_i) <= k\}\,|}, 1 \le i \le n. \tag{6}$$

A large J_k implies a high success in identifying the top-k models.

Table 3. Spearman's Correlation Coefficient of Ranking Results of LaF and CRC

Data Type	MNIST		Fashion-MNIST		CIFAR-10	
	LaF	CRC	LaF	CRC	LaF	CRC
ID	0.89	0.26	0.80	0.36	0.99	0.05
Gaussian Noise	0.97	0.23	0.36	0.23	0.94	0.18
Shot Noise	0.90	0.36	0.33	0.36	0.92	0.18
Impulse Noise	0.97	0.29	0.65	0.29	0.92	0.18
Defocus Blur	0.38	0.16	0.73	0.16	0.99	0.10
Glass Blur	0.77	0.16	0.35	0.16	0.96	0.22
Zoom Blur	0.91	0.37	0.22	0.37	0.99	0.12
Snow	0.98	0.10	0.44	0.10	0.99	0.16
Fog	0.74	0.08	0.34	0.08	0.99	0.05
Brightness	0.98	0.12	0.16	0.12	0.99	0.07
Contrast	0.83	0.13	0.79	0.13	0.95	0.11
Elastic	0.46	0.25	0.71	0.25	0.98	0.16
JPEG	0.90	0.39	0.27	0.39	0.98	0.20
Pixelate	0.92	0.26	0.55	0.26	0.91	0.20
Frost	0.99	0.13	0.35	0.13	0.98	0.22
Motion Blur	0.61	0.20	0.54	0.20	0.98	0.23

	iWildCam		Amazon		Java250		C++1000	
	LaF	CRC	LaF	CRC	LaF	CRC	LaF	CRC
ID	0.39	0.20	0.99	0.39	0.96	0.98	0.95	0.17
OOD	0.91	0.35	0.97	0.36	0.85	0.52	—	—

5 RESULTS AND DISCUSSION

5.1 RQ1: Effectiveness Given ID Test Data

First, we compare the effectiveness of five methods in ranking multiple DNNs based on the accuracy of ID data. Table 3 shows the comparison of two labeling-free methods. We can see that for ID test data, LaF significantly outperforms CRC in six of seven cases with an average 0.565 better correlation score. Only for the Java250 dataset, both LaF and CRC have similar and good results, 0.96 versus 0.98. Table 4 shows Kendall's τ scores of LaF and CRC, which also demonstrate LaF is better than CRC. For the comparison of LaF and sampling-based methods, Figure 5 shows the result measured by Spearman's rank-order correlation. The first conclusion we can draw is that, over seven datasets, all methods succeed in outputting positively correlated rankings. By comparison, LaF continuously outperforms (by up to 0.74) the baseline methods regardless of the labeling budget. Namely, the ranking by LaF is strongly correlated with the ground truth. In general, for the three sample-selection-based baseline methods, the correlation between the estimated rank and the ground truth increases when more data are labeled. However, for some datasets, the performance is still far from LaF. For example, in Amazon, LaF obtains a correlation coefficient of 0.80, while the best baseline, SDS, only achieves 0.48 using the maximum labeling budget of 180. Besides, due to the sampling randomness, each baseline method obtains different ranking results from 50 experiments, which is indicated by the large standard deviation (up to 0.36, shaded area in the figure) at each labeling budget. As a result, the rank by one experiment is not reliable by occasionally being good and poor. In particular, the standard deviation becomes smaller when more data are

Table 4. Kendall's τ of Ranking Results of LaF and CRC

Data Type	MNIST		Fashion-MNIST		CIFAR-10	
	LaF	CRC	LaF	CRC	LaF	CRC
ID	0.78	0.18	0.61	0.35	0.96	0.03
Gaussian Noise	0.88	0.16	0.38	0.25	0.82	0.13
Shot Noise	0.78	0.25	0.26	0.23	0.84	0.12
Impulse Noise	0.88	0.19	0.58	0.19	0.78	0.13
Defocus Blur	0.35	0.11	0.63	0.25	0.93	0.06
Glass Blur	0.67	0.12	0.38	0.26	0.85	0.15
Zoom Blur	0.82	0.26	0.42	0.27	0.95	0.08
Snow	0.91	0.08	0.36	0.18	0.94	0.12
Fog	0.68	0.06	0.61	0.27	0.94	0.03
Brightness	0.91	0.08	0.18	0.21	0.95	0.07
Contrast	0.76	0.10	0.73	0.17	0.86	0.07
Elastic	0.42	0.17	0.59	0.38	0.93	0.11
JPEG	0.78	0.26	0.26	0.14	0.91	0.13
Pixelate	0.80	0.19	0.40	0.17	0.83	0.13
Frost	0.94	0.09	0.38	0.14	0.91	0.15
Motion Blur	0.51	0.14	0.52	0.18	0.92	0.15

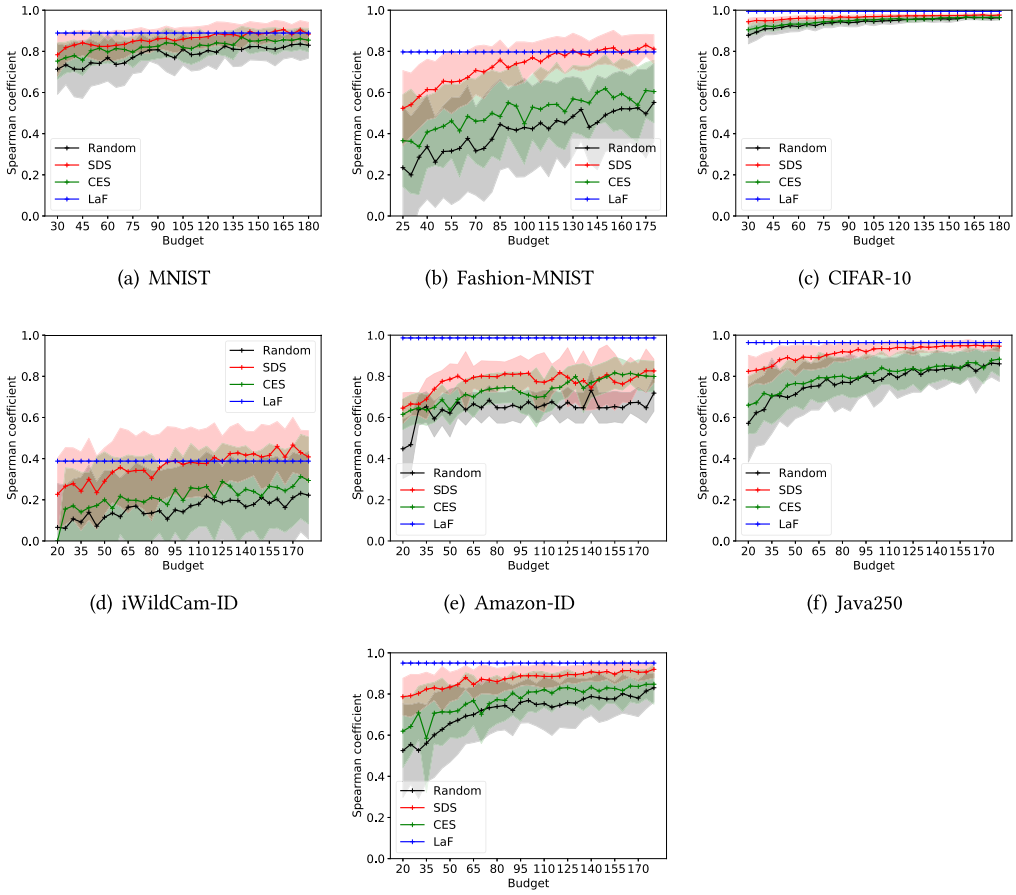
	iWildCam		Amazon		Java250		C++1000	
	LaF	CRC	LaF	CRC	LaF	CRC	LaF	CRC
ID	0.25	0.19	0.95	0.29	0.91	0.92	0.87	0.17
OOD	0.77	0.27	0.92	0.28	0.83	0.55	—	—

labeled, which means the ranking method highly relies on the labeling budget. By contrast, since LaF is labeling-free, there is no sampling randomness, in other words, the rank is deterministic.

Additionally, Figure 6 presents the effectiveness of all ranking methods based on Kendall's τ rank correlation. By comparison, the result confirms the conclusion drawn from the analysis based on Spearman's rank-order correlation. Namely, our approach stands out concerning the effectiveness without sampling randomness.

Besides, to demonstrate the significance of the two statistical analyses, we calculate the corresponding p-value of all methods. A p-value lower than the common significance level of 0.05 indicates that the ranking is strongly correlated with the ground truth. Except for the iWildCam dataset, the ranking results by LaF are all strongly correlated. However, due to the effectiveness and sampling randomness, the baseline methods always achieve insignificant rankings. For the iWildCam dataset, we believe the reason is that the difference between multiple DNNs is too slight given 182 classes. For instance, the accuracy difference between the best and worst is only 1.54% (Table 1). The impact of the accuracy/robustness on the ranking is investigated in Section 5.4.

However, we evaluate different methods concerning identifying the top-k DNNs ($k = 1, 3, 5, 10$). Table 5 lists the result of Jaccard similarity. On average, LaF achieves the best result regardless of the datasets. It is better than the worst performance by up to 0.33, 0.32, 0.33, and 0.27 in the top 1, 3, 5, and 10 rankings, respectively. Concretely, in the top-1 ranking, for datasets MNIST, Fashion-MNIST, and iWildCam, all methods (Random, SDS, and ours) are not effective (under 0.08). Remark that CES takes the best results of all models for each labeling budget when knowing the ground truth. Specifically, it takes n (number of DNNs) times of labeling budget. Therefore, it sometimes outperforms others but is not applicable in practice.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

(d) iWildCam-ID (e) Amazon-ID (f) Java250

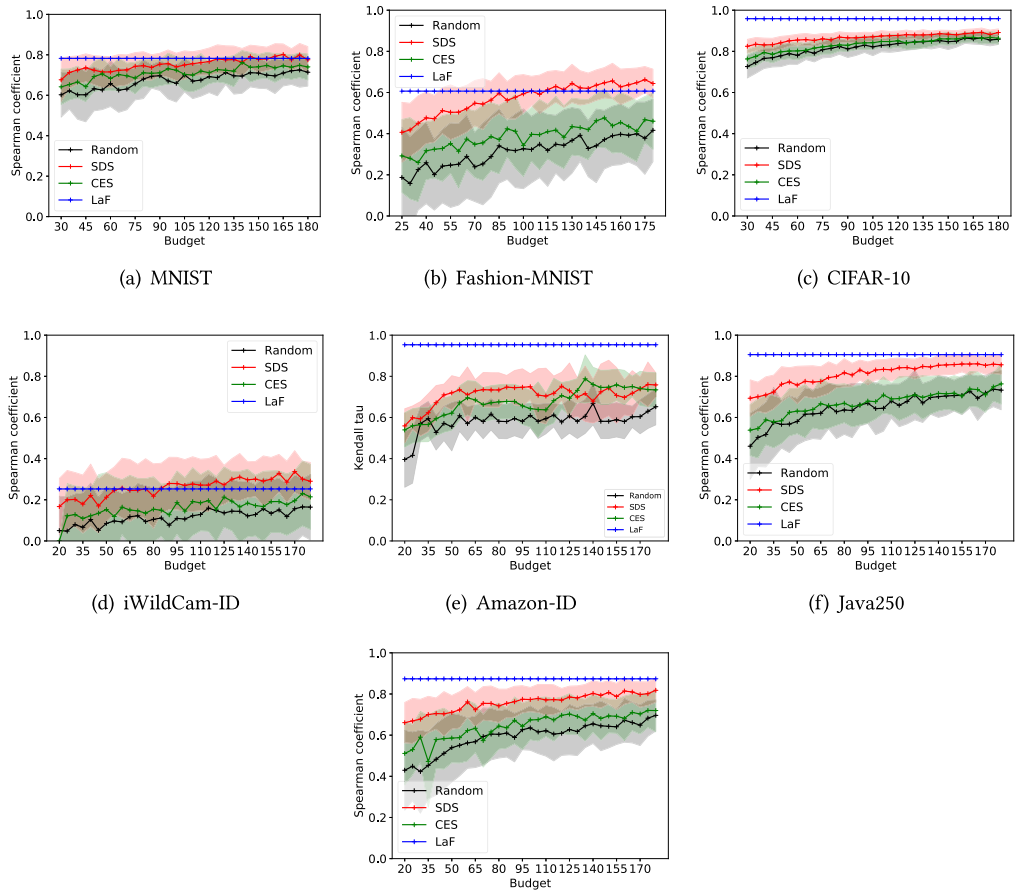
(g) C++1000

Fig. 5. Spearman's correlation coefficient of ranking results based on ID test data. The higher the better. The shaded area represents the standard deviation. "Budget" is the number of labeled data.

Answer to RQ1: Based on the accuracy of ID test data, LaF outperforms all baseline methods in outputting strongly correlated ranking. In addition, statistical analysis demonstrates that outperforming is significant.

5.2 RQ2: Effectiveness Under Distribution Shift

For the synthetic distribution shift, Tables 3 and 4 show the results of two labeling-free methods. The results indicate that LaF outperforms CRC in 46 of 49 cases in terms of Spearman's correlation coefficient, and 48 of 49 cases in terms of Kendall's results. Considering the comparison of LaF and sampling-based methods, Tables 6 and 7 summarize the results of Spearman's rank-order correlation and Kendall's τ on MNIST-C, Fashion-MNIST-C, and CIFAR-10-C, respectively. We observe that our approach achieves the best performance in most cases, for instance, in CIFAR-10-C, LaF consistently beats other methods. Furthermore, as shown in RQ1, SDS performs the second best among the four ranking approaches. However, compared to random and CES, SDS tends to lose its performance in these two tables (highlighted in yellow). For example, in MNIST-C with Defocus



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

(d) iWildCam-ID (e) Amazon-ID (f) Java250

(g) C++1000

Fig. 6. Kendall's τ of ranking results based on ID test data. The higher the better. The shaded area represents the standard deviation. "Budget" is the number of labeled data (only apply to Random, SDS, and CES).

Blur, SDS ranks the models wrongly with a correlation of −0.15 (Table 6), while CES can achieve a good ranking performance. The reason is that SDS only relies on the prediction of models to rank. However, the confidence of models on OOD data reduces when the level of severity increases, e.g., SDS performs extremely poorly on MNIST-C where many models have very low accuracy on the dataset MNIST-C-severity-5. In short, this existing state-of-the-art approach is sensitive to artificial distribution shifts, which calls for the testing under distribution shifts of existing approaches. Considering the Jaccard similarity, in the 75 corruptions of MNIST-C, both LaF and CES outperform the random sampling and SDS to identify the top DNNs precisely. In CIFAR-10-C, LaF achieves the best performance (similarity of 1) in most cases (173 of 300). For the natural distribution shift, the results are shown in Figure 7. LaF can better distinguish the performance of DNNs than the baseline methods. In addition, concerning the Jaccard similarity in Table 8, LaF is also the best in identifying the top DNNs. The reason for the good performance of LaF is that the Bayesian model in LaF can model the probability of each possible label given the predicted labels from different DNNs and model parameters (data difficulty and model specialty in LaF), which makes LaF flexible to the change of predictions.

Table 5. Jaccard Similarity of Ranking the Top-k DNNs Based on the Clean Accuracy

Jaccard	Method	MNIST	Fashion-MNIST	CIFAR-10	iWildCam	Amazon	Java250	C++1000	Average
k = 1	Random	0.01	0.02	0.19	0.07	0.12	0.12	0.09	0.10
	SDS	0.03	0.07	0.17	0.02	0.20	0.18	0.20	0.15
	CES	0.65	0.23	0.21	0.10	0.15	0.84	0.94	0.45
	Our	0.00	0.00	1.00	0.00	1.00	0.00	1.00	0.43
k = 3	Random	0.07	0.12	0.36	0.14	0.17	0.21	0.19	0.19
	SDS	0.11	0.24	0.37	0.20	0.31	0.88	0.29	0.27
	CES	0.69	0.28	0.39	0.15	0.21	0.18	0.80	0.46
	Our	0.20	0.20	1.00	0.20	1.00	0.70	0.50	0.51
k = 5	Random	0.11	0.19	0.50	0.22	0.25	0.23	0.30	0.28
	SDS	0.17	0.33	0.60	0.33	0.39	0.79	0.44	0.39
	CES	0.89	0.36	0.55	0.16	0.29	0.18	0.68	0.53
	Our	0.25	0.43	1.00	0.25	1.00	0.70	0.67	0.61
k = 10	Random	0.23	0.35	0.80	0.43	0.43	0.26	0.58	0.49
	SDS	0.41	0.55	0.85	0.46	0.49	0.65	0.79	0.62
	CES	0.73	0.46	0.82	0.44	0.46	0.24	0.68	0.61
	Our	0.67	0.54	1.00	0.43	0.67	1.00	1.00	0.76

For baseline methods, we report the average results over all labeling budgets. The best performance is highlighted in gray. The higher the better.

Table 6. Spearman's Correlation Coefficient of Ranking Results based on MNIST-C, Fashion-MNIST-C, and CIFAR-10-C

	MNIST				Fashion-MNIST				CIFAR10			
	Random	SDS	CES	LaF	Random	SDS	CES	LaF	Random	SDS	CES	LaF
Gaussian Noise	0.90	0.94	0.91	0.97	0.29	0.26	0.39	0.36	0.91	0.93	0.92	0.94
Shot Noise	0.76	0.83	0.83	0.90	0.34	0.29	0.42	0.33	0.90	0.92	0.91	0.95
Impulse Noise	0.38	0.44	0.94	0.97	0.26	0.27	0.33	0.65	0.89	0.90	0.90	0.92
Defocus Blur	0.84	0.10	0.87	0.38	0.25	0.30	0.31	0.73	0.93	0.95	0.94	0.99
Glass Blur	−0.16	−0.15	0.93	0.77	0.31	0.31	0.36	0.35	0.84	0.93	0.87	0.96
Zoom Blur	0.77	0.83	0.88	0.91	0.27	0.24	0.39	0.22	0.92	0.94	0.93	0.99
Snow	0.50	0.51	0.95	0.98	0.25	0.23	0.32	0.44	0.91	0.92	0.92	0.99
Fog	0.40	0.39	0.98	0.74	0.16	0.23	0.26	0.34	0.95	0.97	0.96	0.99
Brightness	0.94	0.47	0.95	0.98	0.04	0.05	0.13	0.16	0.94	0.97	0.95	0.99
Contrast	0.94	0.35	0.95	0.83	0.28	0.33	0.26	0.79	0.93	0.95	0.94	0.95
Elastic	0.80	0.16	0.83	0.46	0.41	0.39	0.29	0.71	0.90	0.94	0.92	0.98
JPEG	0.78	0.85	0.82	0.90	0.36	0.34	0.39	0.27	0.79	0.89	0.82	0.98
Pixelate	0.38	0.40	0.86	0.92	0.32	0.27	0.32	0.55	0.89	0.90	0.91	0.91
Frost	0.44	0.42	0.98	0.99	0.23	0.24	0.29	0.35	0.90	0.92	0.92	0.98
Motion Blur	0.58	0.62	0.95	0.61	0.27	0.32	0.32	0.54	0.92	0.94	0.93	0.98
Average	0.62	0.48	0.91	0.82	0.27	0.27	0.32	0.45	0.90	0.93	0.92	0.97

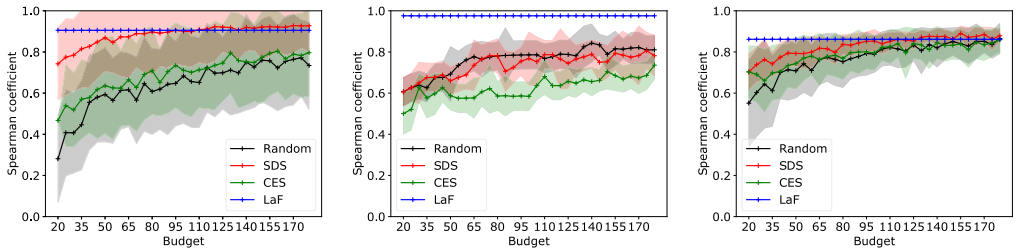
For baseline methods, we compute the average and standard deviation over all labeling budgets and 50-repetition experiments. The best performance is highlighted in gray. Values highlighted in yellow indicate CES or random outperform SDS. The higher the better.

In addition, compared to the effectiveness given ID test data, the ranking by all methods is different, since the performance of DNNs changes given OOD test data. However, we notice an opposite phenomenon happens. Given ID test data, LaF achieves 0.39, 0.99, and 0.96 concerning Spearman's coefficient for iWildCam, Amazon, and Java250, respectively. When given OOD test data, the results are 0.91, 0.97, and 0.85, respectively. In other words, the effectiveness improves on the OOD test data in iWildCam but degrades in Amazon and Java250. To make clear the reason behind this, we analyze the accuracy and robustness of multiple DNNs on ID and OOD test data (Table 1), respectively. In iWildCam, the performance difference of its 20 DNNs becomes larger on OOD test data, from 1.54% to 11.52%. In Amazon, the performance of all 20 DNNs degrades, e.g., from 56.06% to 59.13%. Besides, the performance difference in Amazon becomes smaller. Therefore,

Table 7. Kendall's τ of Ranking Results based on MNIST-C, Fashion-MNIST-C, and CIFAR-10-C

	MNIST				Fashion-MNIST				CIFAR10			
	Random	SDS	CES	LaF	Random	SDS	CES	LaF	Random	SDS	CES	LaF
Gaussian Noise	0.79	0.84	0.82	0.88	0.27	0.27	0.33	0.38	0.78	0.79	0.80	0.82
Shot Noise	0.64	0.71	0.71	0.78	0.34	0.30	0.37	0.26	0.76	0.77	0.78	0.84
Impulse Noise	0.32	0.36	0.84	0.88	0.28	0.28	0.33	0.58	0.75	0.74	0.77	0.78
Defocus Blur	0.71	0.10	0.74	0.35	0.27	0.30	0.31	0.63	0.81	0.85	0.82	0.93
Glass Blur	−0.11	−0.10	0.82	0.67	0.31	0.30	0.34	0.38	0.70	0.81	0.73	0.85
Zoom Blur	0.65	0.71	0.77	0.82	0.29	0.25	0.36	0.42	0.80	0.81	0.81	0.95
Snow	0.43	0.44	0.86	0.91	0.27	0.25	0.33	0.36	0.77	0.79	0.79	0.94
Fog	0.32	0.31	0.91	0.68	0.27	0.24	0.28	0.61	0.84	0.88	0.85	0.94
Brightness	0.86	0.41	0.87	0.91	0.07	0.07	0.13	0.18	0.82	0.87	0.83	0.95
Contrast	0.85	0.31	0.86	0.76	0.30	0.33	0.29	0.73	0.81	0.85	0.83	0.86
Elastic	0.70	0.15	0.72	0.42	0.40	0.38	0.31	0.59	0.76	0.81	0.78	0.93
JPEG	0.66	0.74	0.70	0.78	0.35	0.34	0.35	0.26	0.63	0.73	0.66	0.91
Pixelate	0.33	0.35	0.74	0.80	0.35	0.28	0.38	0.40	0.76	0.76	0.78	0.83
Frost	0.37	0.36	0.91	0.94	0.25	0.27	0.31	0.38	0.77	0.79	0.79	0.91
Motion Blur	0.48	0.51	0.83	0.51	0.29	0.33	0.33	0.52	0.79	0.82	0.81	0.92
Average	0.53	0.41	0.81	0.74	0.29	0.28	0.32	0.45	0.77	0.81	0.79	0.89

For Random, SD, and CES, we compute the average and standard deviation over all labeling budgets and 50-time experiments. The best performance is highlighted in gray. Values highlighted in yellow indicate CES or random outperform SDS. The higher the better.



(a) iWildCam-OOD (b) Amazon-OOD (c) Java250-OOD

Fig. 7. Spearman's correlation coefficient of ranking results based on OOD test data. The higher the better. The shaded area represents the standard deviation. "Budget" is the number of labeled data.

we believe that the model's ability and the performance difference among DNNs have an impact on the ranking effectiveness, which leads to the investigation in RQ3.

Answer to RQ2: Under different distribution shifts, LaF still outstands in all ranking methods. Due to the distribution shift, DNNs' performance declines, which degrades the ranking effectiveness. Particularly, SDS is sensitive to artificial distribution shifts and fails to defeat random sampling and CES in many cases.

5.3 RQ3: Ablation Study

To check if each component contributes to the performance of LaF, we conduct an ablation study. Specifically, we prepare two variants of LaF,

- LaF without pruning. We remove the pruning process of LaF, which means LaF will use all the inputs to do majority voting.
- LaF without optimizing. We remove the optimizing process of LaF and directly use the results of the majority voting to perform the final performance ranking.

Table 8. Jaccard Similarity of Ranking the Top-k DNNs Concerning Natural Distribution Shift

Jaccard	Dataset	Random	SDS	CES	LaF
$k = 1$	iWildCam	0.66	0.96	0.68	1
	Amazon	0.01	0.03	0.84	0.00
	Java250	0.16	0.00	0.95	0.00
$k = 3$	iWildCam	0.4	0.63	0.41	1
	Amazon	0.14	0.21	0.86	0.67
	Java250	0.27	0.11	0.76	0.00
$k = 5$	iWildCam	0.44	0.61	0.47	0.67
	Amazon	0.22	0.19	0.31	0.67
	Java250	0.36	0.28	0.78	0.25
$k = 10$	iWildCam	0.62	0.83	0.65	0.82
	Amazon	0.41	0.12	0.51	0.81
	Java250	0.61	0.72	0.67	1

For baseline methods, we report the average results over all labeling budgets. The best performance is highlighted in gray. The higher the better.

Tables 9 and 10 summarize the results of our ablation study on ID test data. We can see in most cases (12 of 14 cases), LaF outperforms the other two variants. This means each component of LaF is necessary and has a positive impact on the performance of LaF.

Answer to RQ3: Each component contributes to the performance of LaF and can not be removed.

5.4 RQ4: Analysis of Impact Factors

In this part, we analyze the potential factors that could affect the effectiveness of LaF, (1) the number of candidate models, (2) the number of input data used for ranking, (3) the quality of candidate models, and (4) the diversity of candidate models.

For the analysis of the first two factors, we reduce the number of candidate models and test inputs and then repeat the model ranking experiments. Concretely, we set the ratio of models and inputs as 20%, 40%, 60%, 80%, and 100%, and use LaF to rank the models and compare the results to check if these two factors have a high impact on the performance of LaF. Note that 100% of inputs and models mean the settings used in the first three research questions. Figure 8 depicts the results of ID test data. First, considering the input numbers, we can see the results are relatively stable. With increasing (or decreasing) the number of inputs, the ranking performance of LaF is still at a similar level. This indicates LaF is not sensitive to the number of input data. However, considering the number of models, the results indicate that LaF is more effective to rank a small number of models. With the increase of model numbers, the performance of LaF slightly drops, except for the iWildCam, which has a big performance drop. We can conclude that it is more challenging to rank a large number of models by using LaF.

As mentioned in RQ2, by comparing the ranking effectiveness given ID and OOD test data, we raise the demand of investigating the two impact factors, the quality, and diversity of multiple DNNs. The quality refers to the model's performance given the test data and is calculated as the average accuracy or robustness over all DNNs on each dataset. For instance, in MNIST without distribution shift, the quality is the average accuracy of 30 DNNs on the ID test data, and in MNIST-C with Gaussian Noise (severity = 1), the quality is the average robustness of 30 DNNs on the

Table 9. Results of Ablation Study

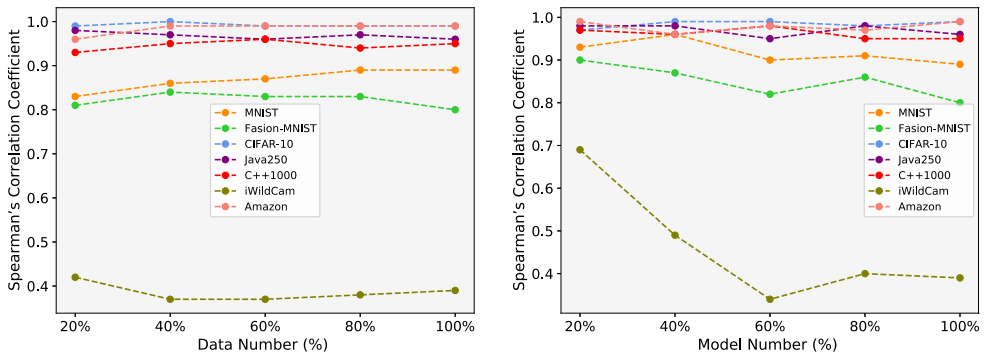
	LaF	w/o Pruning	w/o Optimizing
MNIST	0.89	0.88	0.85
Fashion-MNIST	0.80	0.80	0.77
CIFAR-10	0.99	0.98	0.99
Java250	0.96	0.97	0.94
C++1000	0.95	0.90	0.94
iWildCam	0.39	0.34	0.35
Amazon	0.99	0.95	0.92

Spearman's correlation coefficient of ranking results. The best performance is highlighted in gray.

Table 10. Results of Ablation Study

	LaF	w/o Pruning	w/o Optimizing
MNIST	0.78	0.72	0.77
Fashion-MNIST	0.96	0.96	0.95
CIFAR-10	0.61	0.60	0.61
Java250	0.91	0.86	0.89
C++1000	0.87	0.89	0.84
iWildCam	0.25	0.25	0.18
Amazon	0.95	0.93	0.91

Kendall's τ of ranking results. The best performance is highlighted in gray.



(a) Impact of data number

(b) Impact of model number

Fig. 8. Impact of numbers of data and models. Spearman's correlation coefficient of ranking results.

corresponding OOD data. The diversity indicates the performance difference among DNNs and is the standard deviation of accuracy or robustness over all DNNs on each dataset.

Figure 9 plots the distribution of ranking performance concerning quality and diversity. Most good rankings happen with a high model quality (greater than 50%). The reason is that in our scenario, we only have access to the predictions of test data of multiple DNNs, which sets up the initial inference of data difficulty and model specialty. Therefore, the learned Bayesian model can be more precise when the qualities of DNNs are high. Furthermore, this also explains why LaF outperforms the sampling-based methods. For example, SDS selects a few discriminative data to annotate to rank DNNs and the selection of data highly relies on the predicted labels. As a result, since the low qualities of DNNs always give a wrong estimation of the discrimination ability of data, the

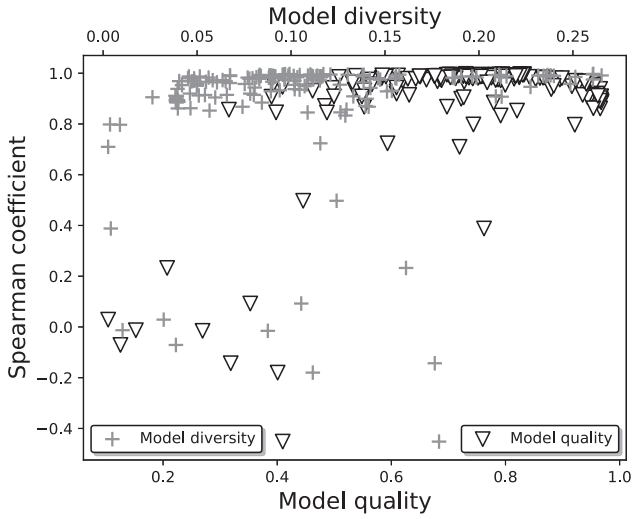


Fig. 9. The impact of model quality/diversity on the ranking performance of LaF. Each point indicates Spearman's correlation coefficient of a specific dataset and its DNNs. All 160 datasets are included.

ranking performance is poorer. For instance, in Java250 and C++1000, SDS only reaches 0.82 and 0.79 on Spearman's correlation with 20 labeled data, respectively. However, LaF achieves 0.96 and 0.95 in two datasets, respectively, with no labeling effort. However, concerning the diversity, Figure 9 reveals that there is a high chance of a good ranking when DNNs are diverse (greater than 5%). Additionally, a poor ranking mostly happens when DNNs are too close to each other, which confirms the result of iWildCam with ID data (Figure 2(d)) that all ranking methods obtain poor ranking.

Answer to RQ4: There is no connection between the performance of LaF and the number of used test data. However, with the increase in model numbers, LaF is more challenging to rank the models. Besides, when the multiple DNNs have high quality (e.g., the average accuracy/robustness is over 50%), the performance of DNNs can be discriminated better. However, there is a higher chance of a good ranking when DNNs are more diverse (larger than 18%).

5.5 Applicable Scenarios

Collecting a large number of labeled data is the starting point to build a high-performance DNN model. However, data labeling is time-consuming and usually requires domain knowledge. We provide some application scenarios of our work.

(1) Overcome the challenge of insufficient domain knowledge. For example, a Java developer can easily annotate the source code in Java but will have difficulties with other programming languages such as C++, due to the difference in handling memory allocations. Consider a scenario where a Java developer builds a DNN model to solve some Java tasks, e.g., bug detection. If the developer later switches from Java to C++, then preparing labeled C++ datasets to select another good model from the open source can be challenging. LaF can assist in this scenario by enabling the developer to select a suitable model without this domain knowledge.
(2) Manage distribution shift. Consider a company that intends to leverage a DNN model to solve a particular task. They prepared some labeled data to select and build a good model for this month. However, as time goes by, the new coming data may follow a different

distribution from the existing labeled data. If the company wishes to switch to a better model, then it needs to label additional new data and redo the model selection. With LaF, the company can effortlessly obtain the best model without further labeling effort.

6 DISCUSSION

6.1 Limitation

Since LaF relies on the results of majority voting to annotate the inputs, the quality of voting could affect the performance of LaF. When many candidate models (not all the models, since LaF has a pruning process to filter these data where all the models have the same prediction) have consistently wrong predictions on a large number of inputs, LaF could have a poor ranking performance. However, this is the corner case that we will rarely meet. Even though models have poor performance, their wrong predictions can be different, for example, in MNIST-C-Defocus Blur-Severity = 5, almost all of the models have bad performance (accuracy ranges from 1.96 to 18.30), LaF still performs well for ranking them.

6.2 Threats to Validity

The internal threat is mainly due to the implementation of the baseline methods, our proposed approach, and the evaluation metrics. For SDS, we use the original implementation on GitHub provided by Meng et al. [37]. For random sampling and CES, we implement it based on the description in Reference [37] and carefully check the result to be consistent with that in Reference [37]. Regarding the evaluation metrics, we adopt popular libraries, such as SciPy [6].

The external threat comes from the evaluated tasks, datasets, DNNs, and baseline methods. Regarding the classification tasks, we consider three different ones, image, text, and source code. For the datasets, we select the publicly available datasets. In particular, for datasets with the artificial distribution shift (15 types of natural corruptions) and natural distribution shift, we employ four public benchmarks. Concerning the DNNs, we collect them (either the off-the-shelf models or train with the provided scripts) from different repositories on GitHub. These models are with different architectures and parameters. For the comparison, we consider three sample-selection-based baseline methods and apply different numbers of labeling budgets to imply their performance.

The construct threat mainly lies in the sampling randomness in the baseline methods and the evaluation measures. To reduce the impact of randomness, for each baseline method, we repeat each experiment concerning the labeling budgets and datasets 50 times and report the results of both average and standard deviation. Since our proposed approach does not rely on sampling data to annotate, there is no sampling randomness. Considering the randomness (gradient ascent search) in the EM algorithm, we repeat LaF 50 times and found that the randomness was negligible (less than 1.84E-03). Regarding the evaluation measures, we consider three popular statistical analyses. Kendall's τ rank correlation and Spearman's rank-order correlation can infer the effectiveness of the methods concerning the general ranking, while the Jaccard similarity can specifically check the performance concerning the top-k ranking. Besides, for the statistical analyses, we report the p-value to demonstrate the significance.

7 RELATED WORK

We review the related work from two aspects, deep learning testing and test selection for deep learning.

7.1 Deep Learning Testing

DL testing refers to evaluating the quality of developed DNNs for further deployment [55]. A simple and local testing strategy is to split a dataset into training, validation, and test sets. The training

and validation sets contribute to the training process to tune parameters. The test set is untouched by the training process to provide an unbiased evaluation of the accuracy. Typically, this testing is built on the assumption that the training and test sets are independent and identically distributed.

Instead of simple performance testing, multiple advanced testing techniques have been proposed in recent years. Pei et al. [43] proposed neuron coverage-guided testing for deep learning systems, which borrows the idea from code coverage-based testing in traditional software engineering. Here, the coverage is calculated based on the outputs of neurons in DNN. After that, based on the basic neuron coverage criterion, Ma et al. [35] designed different types of coverage criteria to further explore the coverage-guided testing. Based on these coverage criteria, DeepTest [49] and TACTIC [33] tended to test DNN-based self-driving systems. Both of them utilize the coverage information to guide the search algorithm to generate error-prone test sets to challenge the target systems. Besides, the famous technique—fuzzing testing was also applied for testing deep learning models. Odena et al. [41] proposed the first fuzzing testing framework by randomly injecting noise into the image to generate new tests to find the error inputs against the DNN model. Xie et al. [54] used neuron coverage as fitness to fuzz the data and generate tests for the DNN testing. More practical, Guo et al. [19] provided a tool to support fuzzing testing of DL models.

Different from the above works, which focus on a single DNN model and utilize test data to measure the quality of the model, our work studies multiple models and provides a new technique to rank multiple models without label information.

7.2 Test Selection for Deep Learning

The purpose of test selection for deep learning is to reduce the labeling effort during DL testing. Generally, test selection methods can be divided into two types, test selection for fault identification and test selection for performance estimation.

Test selection for fault identification is to find the test data that are most likely been mispredicted by the model. Multiple methods have been proposed in the last few years. Feng et al. [17] and Ma et al. [36] proposed metrics based on the uncertainty of output probabilities and also demonstrated that these metrics can be used to select data and retrain the pre-trained model to further enhance its performance. Chen et al. [50] utilized the technique of mutation testing to mutate input data and models and select the error-prone test data based on the killing score. More recently, Li et al. [30] proposed a learning-based method that uses graph neural networks to learn the difference between the fault data and normal data and then predicts the new faults. Gao et al. [18] considered the diversity of faults and selected faults from different fault patterns.

Test selection for performance estimation aims to select a subset of data that can represent the whole test set. In this way, we can only label and test this subset and know the performance of the model on the entire test data. Li et al. [32] proposed CES, which selects samples that have the minimum cross entropy with the entire test set. Chen et al. [12] proposed a clustering-based method PACE that selects data from the center of each cluster. Hu et al. [22] proposed Aries, a method that leverages the connection between the accuracy of a model on a test set and the distance between the set to the decision boundaries to estimate the model performance.

Even though test selection for performance estimation can be also used for selecting the best model during the model reusing process, we considered it as our comparison baseline. The major difference between test selection and our proposed method LaF is—LaF is labeling free, which means the model reusing process can be fully automated.

8 CONCLUSION AND FUTURE WORK

Observing the limitations (labeling effort, sampling randomness, and performance degradation on out-of-distribution data) of existing selection-based methods, we proposed a labeling-free

approach to undertake the task of ranking multiple DNNs without the need for domain expertise to lighten the MLOps. The main idea is to build a Bayesian model given the predicted labels of data, which allows for free labeling and non-sampling randomness. The experimental results on various domains (image, text, and source code) and different performance criteria (accuracy and robustness against artificial and natural distribution shifts) demonstrate that LaF significantly outperforms the three baseline methods concerning both Spearman's correlation and Kendall's τ. In addition, the results of the Jaccard similarity show the efficiency of LaF in identifying the top-k ($k = 1, 3, 5, 10$) DNNs. This work currently only focuses on the classification task, we will explore it for other tasks, such as regression, in future work. Observing the ranking difference on ID and OOD test data, our approach might be useful to detect the existence of distribution shifts. We will consider this in future work.

In the future, we plan to:

- Combine sampling-based methods and LaF to further improve the effectiveness of model ranking. Since we can see that in some cases (e.g., models have low quality and diversity) LaF cannot perform well, we can use β to check the quality of models first, and then use sampling-based methods to rank models for these cases.
- Utilize LaF to build better model ensembles. Since existing works [20] show that it is good to use diverse models to build model ensembles, we can leverage LaF (β value) to check the model diversity for boosting ensemble building.

APPENDICES

A INCREASING LABELING BUDGETS

In this Appendix, we provide more results by comparing LaF to sampling-based methods with bigger labeling budgets. Here, we consider labeling budgets 500, 1,500, and 2,500. Note that SDS needs to select data from the top 25% ranked data, thus labeling budget of 2,500 is not suitable for the iWildCam dataset (over 2,500 data). Table A1 summarizes the results. We can see that in most cases (14 of 21 cases), LaF outperforms sampling-based methods under labeling budget 500. Under the labeling budget of 1,500, LaF still has competitive performance with other baselines, e.g., LaF achieves the best results in 10 of 21 cases. However, when the labeling budget becomes 2,500, sampling-based methods are better choices.

Table A1. Increasing the Budgets of Sampling-based Methods

Dataset	Method	500	1500	2500	Dataset	Method	500	1500	2500
MNIST	Random	0.89	0.95	0.98	**C++1000**	Random	0.85	0.91	0.96
	CES	0.96	0.97	1.00		CES	0.95	0.95	0.96
	SDS	0.95	0.97	1.00		SDS	0.96	0.98	0.98
	LaF	0.89	0.89	0.89		LaF	0.95	0.95	0.95
CIFAR-10	Random	0.98	0.99	1.00	**iWildCam**	Random	0.36	0.53	0.58
	CES	0.99	0.99	1.00		CES	0.54	0.60	0.60
	SDS	0.99	0.99	0.99		SDS	0.46	0.45	—
	LaF	0.99	0.99	0.99		LaF	0.39	0.39	0.39
Fashion-MNIST	Random	0.77	0.79	0.93	**Amazon**	Random	0.56	0.76	0.85
	CES	0.87	0.86	0.94		CES	0.75	0.85	0.93
	SDS	0.90	0.91	0.91		SDS	0.56	0.79	0.86
	LaF	0.80	0.80	0.80		LaF	0.99	0.99	0.99
Java250	Random	0.91	0.96	0.97					
	CES	0.96	0.97	0.99					
	SDS	0.95	0.96	0.96					
	LaF	0.96	0.96	0.96					

Values highlighted by yellow background indicate where LaF is better.

B ACCURACY OF ARTIFICIAL DISTRIBUTION SHIFT DATASETS

Table A2 summarizes the accuracy of our collected models on artificial distribution shift datasets.

Table A2. Summary of MNIST-C, Fashion-MNIST-C, and CIFAR-10-C with the Artificial Distribution Shift and Robustness

Corruption Type	Severity = 1	Severity = 2	Severity = 3	Severity = 4	Severity = 5
MNIST-C					
Gaussian Noise	84.85–99.49	80.49–99.25	55.89–99.05	29.86–98.77	16.92–96.02
Shot Noise	84.89–99.53	84.73–99.49	84.87–99.38	84.32–99.00	83.12–98.68
Impulse Noise	84.29–99.20	71.33–98.91	56.50–98.62	27.77–96.39	16.80–88.67
Defocus Blur	58.63–95.69	31.76–84.91	9.73–43.40	3.73–20.46	1.96–18.30
Frosted Glass Blur	65.06–96.52	54.02–94.33	19.06–75.43	15.63–70.29	11.12–54.46
Motion Blur	56.71–97.29	36.28–90.04	25.23–78.19	20.29–65.58	18.59–61.07
Zoom Blur	82.68–99.54	81.50–99.45	80.42–99.35	78.30–99.22	74.93–98.7
Snow	51.43–99.06	17.78–98.38	19.27–96.03	16.53–93.73	11.39–95.73
Frost	17.25–98.99	10.60–97.91	9.52–96.72	9.18–96.63	9.05–95.36
Fog	9.74–97.07	9.61–95.32	9.75–89.51	9.73–85.28	9.69–73.37
Brightness	66.18–99.53	21.94–99.22	12.55–98.99	9.96–98.60	9.10–97.06
Contrast	37.47–99.22	20.38–99.17	10.29–99.04	9.74–98.08	8.02–92.11
Elastic	49.09–99.07	12.80–17.58	79.89–97.75	44.52–94.33	12.32–71.43
Pixelate	80.40–98.64	84.37–99.11	77.65–98.47	62.11–92.34	54.96–89.33
JPEG	84.90–99.53	84.93–99.56	85.10–99.54	84.78–99.44	84.85–99.36
Fashion-MNIST-C					
Gaussian Noise	74.57–74.57	56.19–83.34	37.41–73.87	18.24–61.61	11.94–44.15
Shot Noise	82.48–89.97	77.72–87.53	72.08–83.93	62.20–78.25	53.33–72.39
Impulse Noise	52.35–91.07	29.11–87.62	20.06–83.07	12.87–69.44	10.81–54.32
Defocus Blur	26.08–70.49	13.51–59.18	7.39–44.41	3.35–33.09	1.87–27.16
Frosted Glass Blur	59.16–80.66	57.87–77.73	28.12–58.08	28.37–58.97	23.0–52.96
Motion Blur	70.43–92.55	66.97–91.59	61.61–90.74	58.67–89.68	53.02–88.42
Zoom Blur	61.09–87.87	34.25–77.51	41.80–78.9	40.72–78.17	34.46–77.66
Snow	29.40–82.47	23.91–78.08	19.23–69.08	17.66–59.28	14.38–43.6
Frost	64.64–90.34	54.74–89.14	40.69–88.19	31.25–87.18	27.16–85.87
Fog	29.86–87.42	21.40–85.32	12.80–80.58	10.00–62.81	10.00–42.9
Brightness	51.79–85.74	15.64–19.97	51.55–77.36	41.33–67.57	22.46–50.32
Contrast	83.89–90.14	81.87–89.8	80.78–89.16	76.59–88.1	74.80–87.01
Elastic	40.71–64.36	76.53–88.27	70.78–84.76	27.80–51.18	40.37–72.85
Pixelate	47.18–88.03	33.51–84.25	27.71–79.91	27.11–79.0	23.45–75.86
JPEG	44.56–84.03	31.41–72.04	29.23–58.28	22.68–47.94	18.36–45.64
CIFAR-10-C					
Gaussian Noise	59.02–82.46	45.70–75.57	34.70–73.70	30.07–72.54	25.08–71.14
Shot Noise	66.72–88.90	57.29–81.71	42.12–75.33	37.34–74.12	31.77–71.97
Impulse Noise	63.82–88.70	52.58–83.19	42.74–76.43	24.28–66.53	15.32–61.54
Defocus Blur	69.62–96.13	68.32–96.19	66.24–95.85	52.11–91.50	31.98–86.14
Frosted Glass Blur	35.53–74.17	37.23–73.88	40.17–73.61	30.33–68.81	32.24–68.36
Motion Blur	67.30–94.31	59.71–91.27	49.23–86.62	48.20–86.27	40.05–81.53
Zoom Blur	64.45–93.94	57.73–94.29	49.07–92.93	42.70–91.18	35.24–87.35
Snow	67.65–93.63	54.93–87.77	59.97–88.74	58.91–86.40	54.09–82.56
Frost	66.73–92.93	62.62–89.90	54.64–84.75	53.15–83.61	44.76–76.56
Fog	68.71–95.87	61.93–95.16	55.08–93.99	45.97–91.18	32.22–75.58
Brightness	69.55–95.76	68.88–95.65	67.51–94.98	65.73–94.44	60.59–92.64
Contrast	67.12–95.77	49.64–93.63	38.69–91.56	30.37–87.35	15.53–64.79
Elastic	62.71–93.92	63.91–94.03	63.78–93.46	55.80–86.46	53.64–80.07
Pixelate	69.62–94.77	67.38–91.97	60.46–89.24	42.55–78.97	28.45–75.55
JPEG	69.29–87.71	64.43–83.60	61.23–81.53	59.08–80.11	54.93–77.30

Each dataset includes 15 types of natural corruptions (e.g., Gaussian Noise) with five levels of severity (1–5). The number in each cell presents the minimum and maximum robustness of multiple DNNs given the corruption type and severity.

REFERENCES

[1] AOJ: Online Programming Challenge. 2018. AIZU online judge. Retrieved from https://judge.u-aizu.ac.jp/onlinejudge/. Accessed 10 January 2021.

[2] SDS. 2021. DNN models for Fashion-MNIST. Retrieved from github.com/Testing-Multiple-DL-Models/SDS/tree/main/models. Accessed 2 November 2021.

[3] LaF project site. 2021. Project website of ranking multiple DNNs. Retrieved from https://sites.google.com/view/ranking-of-multiple-DNNs

[4] GitHub. 2022. Retrieved from https://github.com/. Accessed 25 January 2022.

[5] MLOps. 2022. *Machine Learning Model Operationalization Management*. Retrieved from https://ml-ops.org/

[6] SciPy. 2022. Retrieved from https://scipy.org/. Accessed 27 January 2022.

[7] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*, Vol. 16, 265–283.

[8] Rob Alexander, Rob Ashmore, Andrew Banks, Ben Bradshaw, John Bragg, John Clegg, Christopher Harper, Catherine Menon, Roger Rivett, Philippa Ryan, Nick Tudor, Stuart Tushingham, John Birch, Lavinia Burski, Timothy Coley, Neil Lewis, Ken Neal, Ashley Price, Stuart Reid, and Rod Steel. 2020. *Safety Assurance Objectives for Autonomous Systems*.

[9] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. code2vec: Learning distributed representations of code. *Proc. ACM Program. Lang.* 3 (2019), 1–29. https://doi.org/10.1145/3290353

[10] Sara Beery, Elijah Cole, and Arvi Gjoka. 2020. The iWildCam 2020 competition dataset. Retrieved from https://arXiv:2004.10340

[11] David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2020. *Cats are Not Fish: Deep Learning Testing Calls For Out-of-distribution Awareness*. Association for Computing Machinery, New York, NY, USA, 1041–1052. https://doi.org/10.1145/3324884.3416609

[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 Trans. Softw. Eng. Methodol.* 29, 4, Article 30 (Oct. 2020), 35 pages. https://doi.org/10.1145/3394112

[13] W. Wayne Daniel. 1990. *Applied Nonparametric Statistics*. PWS-KENT Pub. Retrieved from https://books.google.lu/books?id=0hPvAAAAMAAJ

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. Ser. B (Methodol.)* 39, 1 (1977), 1–38. Retrieved from http://www.jstor.org/stable/2984875

[15] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou Soffa. 2021. Distribution-aware testing of neural networks using generative models. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE'21)*. IEEE, 226–237. https://doi.org/10.1109/ICSE43902.2021.00032

[16] Robert L. Ebel. 1954. Procedures for the analysis of classroom tests. *Edu. Psychol. Measure.* 14, 2 (1954), 352–364. https://doi.org/10.1177/001316445401400215

[17] 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*. 177–188.

[18] 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*. 73–85.

[19] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 739–743.

[20] Yuejun Guo, Qiang Hu, Maxime Cordy, Michail Papadakis, and Yves Le Traon. 2021. MUTEN: Boosting gradient-based adversarial attacks via mutant-based ensembles. Retrieved from https://arXiv:2109.12838

[21] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=HJz6tiCqYm

[22] Q. Hu, Y. Guo, X. Xie, M. Cordy, M. Papadakis, L. Ma, and Y. Traon. 2023. Aries: Efficient testing of deep neural networks via labeling-free accuracy estimation. In *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering (ICSE'23)*. IEEE Computer Society, Los Alamitos, CA, 1776–1787. https://doi.org/10.1109/ICSE48619.2023.00152

[23] Rui Hu, Jitao Sang, Jinqiang Wang, and Chaoquan Jiang. 2021. Understanding and testing generalization of deep networks on out-of-distribution data. Retrieved from https://arXiv:2111.09190

[24] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. 2018. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 954–960. https://doi.org/10.1109/CVPRW.2018.00141

[25] Irena Jovanović. 2006. Software testing methods and techniques. *IPSI BgD Trans. Internet Res.* 30 (2006).
[26] S. Kavitha and D. Jeevitha. 2014. Software testing methods and techniques. In *Proceedings of the International Conference on Information and Image Processing (ICIIP'14)*.
[27] 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 A. Earnshaw, Imran S. Haque, Sara 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 International Conference on Machine Learning (ICML'21)*.
[28] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features From Tiny Images*. Technical Report. University of Toronto, Toronto.
[29] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324.
[30] Yu Li, Min Li, Qiuxia Lai, Yannan Liu, and Qiang Xu. 2021. TestRank: Bringing order into unlabeled test instances for deep learning tasks. *Adv. Neural Inf. Process. Syst.* 34 (2021), 20874–20886.
[31] 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 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19)*. Association for Computing Machinery, New York, NY, 499–509. https://doi.org/10.1145/3338906.3338930
[32] 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 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 499–509.
[33] Zhong Li, Minxue Pan, Tian Zhang, and Xuandong Li. 2021. Testing DNN-based autonomous driving systems under critical environmental conditions. In *Proceedings of the International Conference on Machine Learning*. PMLR, 6471–6482.
[34] Bin Liu. 2022. Consistent relative confidence and label-free model selection for convolutional neural networks. In *Proceedings of the 3rd International Conference on Pattern Recognition and Machine Learning (PRML'22)*. IEEE, 375–379.
[35] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131.
[36] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Trans. Softw. Eng. Methodol.* 30, 2 (2021), 1–22.
[37] 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 (ICSE'21)*. IEEE Computer Society, Los Alamitos, CA, 385–396. https://doi.org/10.1109/ICSE43902.2021.00045
[38] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. *Comput. Surveys* 54, 3 (Apr. 2021). https://doi.org/10.1145/3439726
[39] Norman Mu and Justin Gilmer. 2019. MNIST-C: A robustness benchmark for computer vision. Retrieved from https://arXiv:1906.02337
[40] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*.
[41] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4901–4911.
[42] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. 2020. Deep learning vs. traditional computer vision. In *Advances in Computer Vision*, Kohei Arai and Supriya Kapoor (Eds.). Springer International Publishing, Cham, 128–144. https://doi.org/10.1007/978-3-030-17795-9_10
[43] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
[44] Ruchir Puri, David S. Kung, Geert Janssen, Wei Zhang, Giacomo Domeniconi, Vladimir Zolotov, Julian Dolby, Jie Chen, Mihir Choudhury, Lindsey Decker, Veronika Thost, Luca Buratti, Saurabh Pujar, Shyam Ramji, Ulrich Finkler, Susan Malaika, and Frederick Reiss. 2021. CodeNet: A large-scale AI for code dataset for learning a diversity of coding tasks. Retrieved from https://arxiv:2105.12655

[45] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Min. Knowl. Discov.* 8, 4 (2018). https://doi.org/10.1002/widm.1249

[46] Abhijit Sawant, Pranit Bari, and Pramila Chawan. 2012. Software testing techniques and strategies. *Int. J. Eng. Res. Appl.* 2 (06 2012), 980–986.

[47] P. B. Selvapriya. 2013. Different software testing strategies and techniques. *Int. J. Sci. Modern Eng.* 2, 1 (2013).

[48] Yan Sun, Celia Chen, Qing Wang, and Barry Boehm. 2017. Improving missing issue-commit link recovery using positive and unlabeled data. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17)*. 147–152. https://doi.org/10.1109/ASE.2017.8115627

[49] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*. 303–314.

[50] 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 IEEE/ACM 43rd International Conference on Software Engineering (ICSE'21)*. IEEE, 397–409.

[51] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09)*. Curran Associates, Red Hook, NY, 2035–2043. Retrieved from https://proceedings.neurips.cc/paper/2009/file/f899139df5e1059396431415e770c6dd-Paper.pdf.

[52] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 38–45. Retrieved from https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[53] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Retrieved from https://arXiv:cs.LG/1708.07747

[54] 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*. 146–157.

[55] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine learning testing: Survey, landscapes and horizons. Retrieved from http://arxiv.org/abs/1906.10742

[56] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. 2021. GUIGAN: Learning to generate GUI designs using generative adversarial networks. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE'21)*. IEEE, 748–760. https://doi.org/10.1109/ICSE43902.2021.00074

[57] Yan Zheng, Yi Liu, Xiaofei Xie, Yepang Liu, Lei Ma, Jianye Hao, and Yang Liu. 2021. Automatic web testing using curiosity-driven reinforcement learning. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE'21)*. 423–435. https://doi.org/10.1109/ICSE43902.2021.00048

[58] 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 (ASE'19)*. 772–784. https://doi.org/10.1109/ASE.2019.00077

[59] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Adv. Neural Inf. Process. Syst.* 32 (2019).

Received 10 December 2022; revised 11 July 2023; accepted 13 July 2023