Orchid: Building Dynamic Test Oracles with Training Bias for Improving Deep Neural Network Models[†]

Haipeng Wang
Department of Computer Science
City University of Hong Kong
haipewang5-c@my.cityu.edu.hk

W.K. Chan[††]
Department of Computer Science
City University of Hong Kong
wkchan@cityu.edu.hk

***Abstract*—The accuracy of deep neural network models is always a top priority in developing these models. One problem to affect it is to what extent such a model can resolve training samples conflicting with one another while learning from them. Such a model may easily learn the whole training dataset with high recall, leaving the problem only to appear when evaluating the model, and yet samples in the evaluation should not be used in training the model. This paper proposes Orchid, the first work to alleviate the problem of class pairs with training samples conflicting with one another. Orchid firstly rewinds the learning effect of a deep neural network model under fix on different subsets of the training dataset, producing a set of resultant rewound models. It then constructs a novel kind of reference oracle by dynamic selection and integration of these rewound models, each predicting the same sample in question correctly. It finally adapts the model under fix by retraining it on training samples with the reference oracle. The experiment on MobileNet and ShuffletV2 with the Cifar-100 and Tiny-ImageNet datasets shows that their test accuracies are improved over the baselines by 2.47% and 2.62%, respectively. Orchid also produces noticeable reductions in misclassification between classes of the adapted models on the test datasets.**

Keywords-Deep neural network, Machine learning testing, debugging, Test oracle, Model fixing

I. INTRODUCTION

The accuracy of deep neural network models is always a top priority in developing these models. A deep learning model (***DL model***, for short) M is a deep neural network with trained parameters [26]. It is created by training the network over a training dataset T and evaluated on a test dataset E, which we refer to it as a *training scheme*, denoted by ***scheme*(*S, T, E*)** [27]. For a classification task, it classifies each sample (e.g., an image) to a label called class (e.g., dog). Each sample is also annotated with a label called ground truth.

A popular criterion to judge the completion of a training scheme is to observe M converged when training on T and/or the accuracy on E has reached a maximal point and started decreases. Different rounds of a training scheme tend to create DL models with differences in behaviors [28]. Among these model candidates produced via different rounds of a training scheme, the one with the highest accuracy on E is selected as the final DL model to be created. The test accuracy of a model is the proportion of samples in E, each classified by the model to a class the same as the ground truth of the sample.

Nonetheless, each DL model in the candidate set of DL models produced via a round of *scheme*(*S, T, E*) only corresponds to one possible set of trained parameters producible by *scheme*(*S, T, E*). A sample classified by one candidate DL model to the ground truth may be misclassified by some other candidate DL model to another class. As such, even the DL model with the highest test accuracy on the test dataset among the ones in the candidate set may fall short in some scenarios where some other candidate models in the same candidate set perform better.

For ease of our presentation, we use the two terms ***sample*** and ***test case*** interchangeably. If a DL model misclassifies a sample, the sample is called a ***failing test case*** for the model, otherwise, it is a ***correct test case***.

We further define a few terminologies to facilitate our presentation of our work. We call each unique ground truth appearing in *scheme*(*S, T, E*) as a class C. For instance, in the ImageNet dataset [30], many samples are annotated with "boy" as ground truth (referred to as class C_{boy}). We refer to the ***misclassification ratio*** for a class C_i with respect to C_j for a dataset E, denoted as $\mathrm{MR}(C_i\text{-}C_j, E)$, as the proportion of samples belonging to C_i but classified by M to C_j, i.e., $|\{\ x \in E \mid \text{the ground truth of } x \text{ is } C_i \text{ and } M(x) = C_j\ \}| \div |\{\ x \in E \mid \text{the ground truth of } x \text{ is } C_i\ \}|$. The ***maximal misclassification class*** of C_i, denoted as $\xi(C_i)$, is a class that C_i has the highest misclassification ratio with it, i.e., C_j having $argmax_j\ \mathrm{MR}(C_i\text{-}C_j, E)$. The ***misclassification score*** for class C_i, denoted as $\psi(C_i)$, is defined as $\mathrm{MR}(C_i\text{-}\xi(C_i), E) + \mathrm{MR}(\xi(C_i)\text{-}C_i, E)$. Since the highest proportion of misclassified samples of class C_i has been mapped by M to the class $\xi(C_i)$, the misclassification score for C_i aims to measure the degree of misclassification between C_i and $\xi(C_i)$ exhibited by M.

We call a model M ***ε-conflicting*** if M contains at least one class, say C_i, with $\psi(C_i) \geq \varepsilon$. We also say such a class C_i exhibiting a ***conflicting scenario*** when $\psi(C_i) \geq \varepsilon$. We observe that for a ε-conflicting model M, $\psi(C_i)$ negatively correlates test accuracy, albeit not strong.

Figure 1 depicts the relationships between the test accuracy and misclassification score for those classes with a conflicting scenario in the MobileNet model trained on the

[†]This research is supported in part by the GRF of Hong Kong RGC (project nos.11214116 and 11203517), the HKSAR ITF (project no. ITS/378/18), CityU MF_EXT (project no. 9678180), and a CityU SGS Conference Grant.
[††]W.K. Chan is the correspondence author.

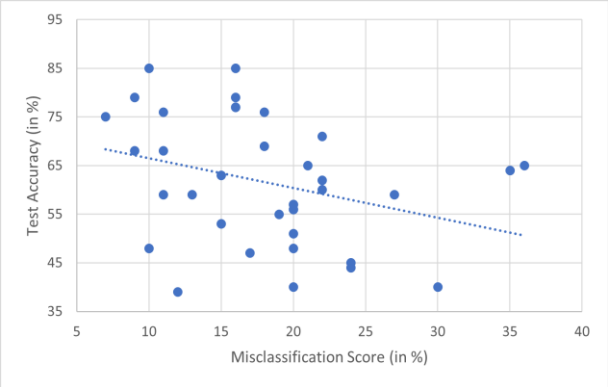

Figure 1. The correlation between misclassification score and test accuracy in percentage for classes with conflicting scenarios. (MobileNet + CIFAR-100 with $\varepsilon = 0.10$)

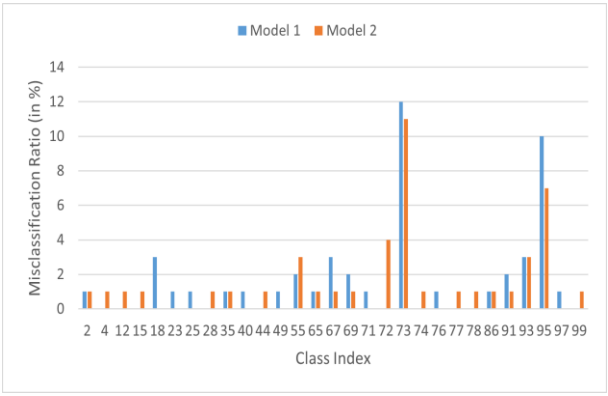

Figure 2 Misclassification ratio (in percentage) in two candidate DL models (Model 1 & Model 2) for class 11. (MobileNet + CIFAR-100)

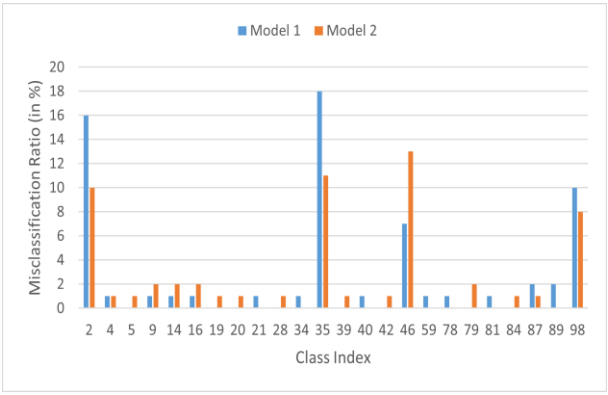

Figure 3 Misclassification ratio (in percentage) in two candidate DL models (Model 1 & Model 2) for class 30. (MobileNet + CIFAR-100)

CIFAR-100 dataset (where ε = 0.10) used in our experiment (see Section VI for details). The x-axis is the misclassification score of a class, and the y-axis is the test accuracy in percentage. We measure the correlation between the two variables (class-level test accuracy and misclassification score) and find that the Spearman's correlation coefficient is –0.48, which indicates a moderate correlation. (Note that Pearson's correlation requires the normality assumption, whereas Spearman's correlation is a nonparametric test, which is suitable for our analysis.) Figure 1 also shows a linear regression line. We observe that the trend of test accuracy decreases as the misclassification score increases.

Our conjecture gained from the above case study is that reducing the extent of misclassification in an ε-conflicting DL model may improve the test accuracy of the corresponding class, thereby potentially improving the test accuracy of the model.

Suppose that a DL model M is created by *scheme*(S, T, E). Since the corresponding training process is lacking in the information of conflicting scenarios, further training M on *scheme*(S, T, E) cannot directly alleviate the conflicting scenario problem further. Moreover, it may overfit M to the training dataset [20][21], which is undesirable.

Previous studies [24][25] explored using a DL model with a larger capacity (e.g., having significantly more parameters than M) as a teacher model to guide the evolution of M.

In this paper, we refer to such a teacher as a *reference test oracle*. We denote each output of the penultimate layer of a DL model as an ***output vector***. The basic approach is to treat the output vector of a reference test oracle O for a test case x as if it is a *vector of oracle information* of x, denoted as $O(x)$. This vector of oracle information $O(x)$ is then used in the loss function of the *scheme*(S, T, E) to replace or combine with the ground truth of x to retrain M on x. In the situation where the reference test oracle is not too heavy to be deployed (e.g., the inference time is too long or the size of the reference test oracle cannot be fit into the GPU memory for efficient inferences), this approach provides a viable approach to distil knowledge from a more complex model and transfer the knowledge to M.

Nonetheless, directly using a reference test oracle has not explicitly dealt with the conflicting scenarios faced by ε-conflicting DL models. In this paper, we study how to alleviate the problem of conflicting scenarios explicitly.

Our insight is that different candidate DL models created by different rounds of *scheme*(S, T, E) may contain different sets of classes with conflicting scenarios. For instance, Figure 2 and Figure 3 show the misclassification ratios for two classes (class 11 and class 30, denoted as C_{11} and C_{30}, respectively) for two candidate DL models created by training MobileNet on the CIFAR-100 dataset, respectively. In each figure, the x-axis is the class number, and the y-axis is the misclassification ratio in percentage for class 11 with respect to the class indicated by the x-axis. From the two figures, $\xi(C_{11})$ is class C_{73} for both candidate models, but $\xi(C_{30})$ is class C_{35} for Model 1 and class C_{46} for Model 2.

Realizing the above insight is challenging. First, to create M from *scheme*(S, T, E), the accuracy achieved on T easily reaches a high level, such as 95% to 100%. Only a low proportion of all test cases in T remains failing. These failing test cases may not be concentrated enough into a particular class to pinpoint a conflicting scenario if ε is not small. On the contrary, on the test dataset E, M may exhibit more conflicting scenarios. However, incorporating the oracle information of test datasets in the loss function of *scheme*(S, T, E) to train M is out of the question. Thus, the critical question is how to identify and use the information of conflicting scenarios in the training dataset to effectively M.

In this paper, we propose a novel technique, ***Orchid***, to address the problem of identifying conflicting scenarios with respect to the training dataset and improving a DL model *M* produced by a training process *scheme*(*S, T, E*). Rather than training *M* to resolve conflicting scenarios unseen in the training dataset *T* as the first step, Orchid creates candidate DL models by teaching *M* to incur different conflicting scenarios. More specifically, Orchid rewinds *M* by reducing *M*'s confidence in predicting different subsets of the training dataset *T*, explicitly increasing the appearance of conflicting scenarios originally hidden in *T*. More specifically, Orchid creates a rewound model by lightly retraining *M* over a small and random subset of the training dataset but the one-hot vector encoding of groundtruth of each such sample replaced by a vector encoding a uniform distribution (i.e., target to teach *M* to recognize the corresponding sample to all classes equally). Intuitively, the ability of *M* to classify samples of some class is then weakened, thereby increasing the chance of the rewound model to exhibit a conflicting scenario. Orchid repeats this process several times to create a set of rewound models. Then, it creates a groundtruth-aware ensemble of these rewound DL models as a reference test oracle and uses the reference test oracle to teach *M* to encode the samples against classification targets non-observable from the training process that creates the original *M*. For each training sample, Orchid determines the subset of these rewound models each classifying the sample correctly and synthesizes an output vector based on the output vectors of these selected candidate models. More specifically, for each sample, it identifies the subsets of these rewound DL models where each rewound model classifies the sample correctly. (In the case of empty subset, the groundtruth of the sample is used as the output vector). We note that this process is different from the standard ensemble, as our synthesized output vector may be derived from zero to all candidate DL models). Finally, Orchid combines these vectors of oracle references and the ground truths of training samples to teach *M*.

In the experiment, we evaluate Orchid on two representatives DL models (MobileNet and ShufflenetV2**)** on two representative datasets (CIFAR-100 and Tiny-ImageNet-200 datasets). The experimental results show that their test accuracies are improved over the baselines by 2.47% and 2.62%, respectively. They also show that Orchid alleviates the conflicting scenarios faced by the original DL models on test datasets.

The main contribution of this paper is threefold: (1) It is the *first* work to formulate the conflicting scenario problem and effectively reveal potential conflicting scenarios of DL models hidden in their training datasets. (2) It presents the *first* and novel technique, Orchid, to convert conflicting scenarios hidden in a training dataset into measurable ones and construct a novel reference test oracle with respect to the model under fix. (3) It reports the *first* evaluation to present the effect of alleviating the conflicting scenario problem and show the feasibility and effectiveness of Orchid.

The organization of the rest of the paper is as follows. Section II presents the preliminaries about Orchid. Section III presents Orchid, followed by an evaluation on it in Section IV. Section V reviews the closely related work. Finally, in section VII, we conclude this work.

II. BACKGROUND

A. Pseudo and Reference Test Oracle

In software testing, a test oracle refers to a mechanism to decide whether the program outputs are correct [19]. If a test oracle is challenging to construct or unavailable, the program suffers from the test oracle problem. For example, machine learning models are often non-testable.

To deal with non-testable programs, Davis and Weyuker pioneered in developing pseudo-oracles [1] for testing them. The intuition of pseudo-oracle testing is that inconsistency among multiple solution implementations for the same problem on the same test cases may reveal a failure of the non-testable programs under test. Another promising method is metamorphic testing [9]. Metamorphic testing requires an expected relation to quantify the relationship over a set of inputs and outputs of the program under test. Any violation of the expected relation indicates that the provided set of inputs and outputs reveals program failures.

A reference test oracle [23, 29] is developed for the programs solving the same task for references and may provide the same functionality as the program under test. In our case, different models are likely different in behavior. Therefore, we refer to an ensemble of rewound DL models handling the same learning task as a reference test oracle.

As summarized in Section I, our ensemble is aware of ground-truth, and its output may vary from using zero to all underlying rewound models, and the output is independent to the aggregated output of the majority of these rewound models. This aspect distinguishes Orchid significantly from conventional ensembles over a set of DL models.

B. Ensemble

In the supervised learning algorithm of machine learning, a typical aim is to create a model with outstanding performance in all aspects, but the practical result is not always ideal. Dietterich [11] proposed that one may compose more than one model with different preferences to get better performance, called an ensemble.

The intuition of ensemble learning is that even if one model gets the wrong prediction in some test cases, another model can correct the error. Therefore, the result of ensemble models will be more outperformed and comprehensive than the single model. For the same learning task, ensemble learning trains multiple supervised models with the same or different machine learning architectures and runs each of the trained models to make a prediction. Finally, it combines all the predictions by some strategies, such as weighted average and majority voting, to make a final prediction.

Orchid, on the other hand, only considers rewound models producing correct classification on individual samples. It relies on the groundtruths (whereas conventional ensembles do not) as its target in this aspect is not to simply create a reference test oracle to replace the groundtruth in inferencing samples.

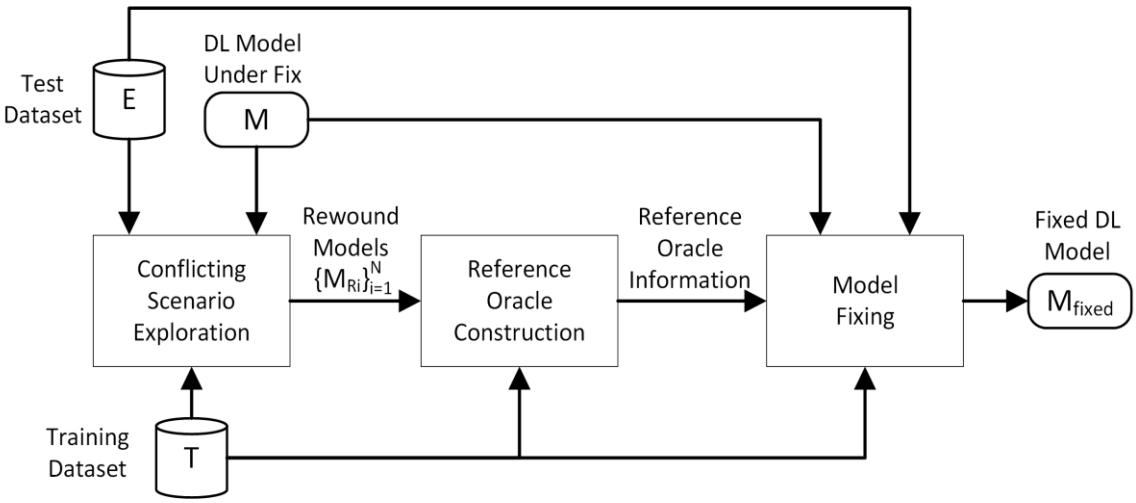

Figure 4. The overall process of Orchid to fix DL model suffering from conflicting scenarios.

C. Knowledge Distillation

Knowledge distillation (KD) [14, 22] is the process to compress and transfer knowledge from a more computationally expensive DL model, referred to as a ***teacher***, to a smaller DL model, referred to as a ***student***. The intuition is that larger models, such as a deep neural network with more layers [15] or an ensemble of many different neural network architectures [17], have higher knowledge capabilities than smaller models whose capacity has not been fully utilized.

In a KD process, the knowledge is distilled from a teacher model to a student model using a distillation loss function, which minimizes the difference between the outputs between the teacher and student models [14]. Thus, the teacher's knowledge can be passed to the student model. As a result, the generalization of the student model would be improved.

Orchid includes a novel KD process which is an ensemble of models creating using the same training scheme and same neural network architecture. It also has an innovation in producing rewound models that expose different conflicting scenarios encoded in the ensemble to be used in its KD process.

III. Our Technique: Orchid

In this section, we present Orchid.

A. Overview

The overall process of Orchid is depicted in Figure 4. There are three main steps: (1) explore the conflicting scenarios, (2) construct reference oracles, and (3) fine-tune the given DL model with reference oracle information to resolve the conflicting scenarios to produce the fixed DL model.

Orchid aims to address the conflicting scenario problem. It evolves DL models towards higher test accuracy by handling the above problem based on its novel mechanism in ensemble construction (to form reference test oracles) and knowledge distillation strategies in knowledge transfer.

Unlike conventional knowledge distillation, Orchid does not use neural network architectures other than the one underlying a given DL model. In this way, the conflicting scenarios in the constructed reference test oracle can be more closely relevant to the given DL model produced by the same training scheme than otherwise.

Suppose that a given model M is created by *scheme*(S, T, E) where S, T, and E are training scheme, training dataset, and test dataset, respectively.

To address the conflicting scenario problem, Orchid first creates such scenarios exhibiting by some models closely related to M. It divides T into non-overlapping subsets of the training dataset and rewinds M against each such subset called a ***rewinding dataset***, denoted as $\boldsymbol{T_{Ri}}$, to produce a new DL model. We also refer to the model after the rewinding process as a ***rewound model***. Subsections B and C present the process in detail. Then, Orchid teaches M by its novel mechanism of creating the output vector from its reference test oracle and transfers the knowledge to M. Subsection D presents this process.

B. Partition Training Dataset

Orchid partitions the original training dataset T in equal size randomly into a set of rewinding datasets $T_R = \{ T_{R1}, ..., T_{RN} \}$ and their corresponding set complements denoted as $T_L = \{ T_{L1}, ..., T_{LN} \}$. The relationship among these subsets is as follows: $T_{Ri} \cup T_{Li} = T$, $T_{Ri} \cap T_{Li} = \emptyset$, $|T_{Ri}| = |T_{Rj}|$, and $|T_{Li}| = |T_{Lj}|$ where $i \in (0, N]$ and $|\cdot|$ means the number of test cases in the set.

C. Rewinding the Given Deep Learning model

Suppose that a DL model M is given. Then, to produce a rewound model, Orchid retrains M over each rewinding dataset T_{Ri} toward the "learning evenly" end of each sample in T_{Ri} and over the complementary dataset T_{Li} to largely retain the classification ability of M remained in T_{Li}.

For ease of our presentation, we refer to the label common to every test case in T_{Ri} as the ***rewinding vector R***. We note

Algorithm 1 Constructing Rewound Models

Input:
$M \leftarrow$ given DL model.
$N \leftarrow$ number of rewinding models.
$R \leftarrow$ rewinding vector.
$T \leftarrow$ training dataset.
$E \leftarrow$ test dataset.
$G \leftarrow$ ground truths of the two datasets

Output: $M_R \leftarrow$ a set of rewound sub models

1: $M_R = \{\ \}$
2: **foreach** $T_{Ri} \in$ Partition(T, N) **do**
3: $\quad T_{Li} = T - T_{Ri}$
4: $\quad M_{Ri} = M$
5: $\quad$ **do**
6: $\quad\quad M_{Ri} = M_{Ri}$.Train($T_{Ri}$, R)
7: $\quad\quad M_{Ri} = M_{Ri}$.Train($T_{Li}$, G)
8: $\quad$ **while** not IsConverged(M_{Ri}, T_R, R, T_{Li}, G)
9: $\quad M_R = M_R + \{M_{Ri}\}$
10: **end for**
11: **return** $\boldsymbol{M_R}$

that this rewinding vector is the same across all training samples to be rewound, while the ground truths for all training samples should not be the same, otherwise, there is no classification task to be learned by M from the training dataset.

The intuition of the rewinding vector R comes from the training process of the given DL model. The given DL model has been trained with T towards the respective ground truth of each test case. As a result, the model exhibits a high recall on these test cases. To reduce the training effect on the rewinding dataset T_{Ri}, our idea is to train M_{Ri} to some unbiased end of the ground truths of the samples in T_{Ri}, which contain equal information with respect to each class that each sample should belong to.

In the rewinding process, Orchid uses Eq. (1) to minimize the training error between the output of the SoftMax layer of the under training model M_{Ri} and the rewinding vector R for each training sample in T_{Ri} based on the L2-norm loss.

$$L = \sum_{x \in T_{Ri}} \|M_{Ri}(x) - R\|_2 \quad (1)$$

Currently, Orchid is formulated with one strategy to model a rewinding vector denoted as $\boldsymbol{R_U}$. Suppose that the given DL model has K classes in total. R_U is a *vector* with K elements encoding the uniform distribution among these K elements whose summation is 1, and each element has a value of $1/K$, i.e., $R_U = \langle 1/K, \ldots, 1/K \rangle$, where $\sum R_U[i] = 1, \text{and} |R_U|$ is K. We note that since R is a vector with the same value in every element, the normalization made by the SoftMax layer has no effect. Thus, one may view Eq. (2) at the penultimate layer of the model.

The detailed process of constructing rewound models is shown in Algorithm 1. Algorithm 1 takes a given DL model as M, the number of rewound models to produce as N, a rewinding vector as R, a training dataset as T, a test dataset as E, the ground truth corresponding to the input samples as G.

In line 1, the algorithm initializes the set M_R of rewound models to empty. In line 2, it partitions the training dataset T into N rewinding datasets. For each rewinding dataset T_{Ri}, it constructs its complementary dataset T_{Li}. Then, from lines 5 to 8, it iteratively trains the current model (starting from M) over the rewinding dataset T_{Ri} with the minimization against R, followed by the dataset T_{Li} with original ground truths as the minimization target. Algorithm 1 checks the test accuracy on the rewound model in each epoch to look for model convergence. After the iteration, in line 9, it keeps the trained model into the set M_R. Finally, Algorithm 1 outputs N rewound DL models, denoted as $M_R = \{ M_{R1}, \ldots, M_{RN} \}$.

D. Reference Test Oracle Construction and Model Fixing

In the reference test oracle construction process, Orchid firstly runs each rewound DL model to predict each test case in the training dataset T to get the corresponding output vectors, and determines whether the rewound model classifies the test case correctly. It then teaches the current DL model under fix (initially the given DL model) with several epochs over the whole training dataset. This teaching process uses both the ground truth of each test case and the vector synthesized by our novel strategy (see below), rather than using the majority voting strategy typically used in a conventional ensemble [11] to teach the current model to minimize the error involved.

In Orchid, we design its integration strategy for combining the oracle information based on two perspectives. The first perspective is the training correctness of each rewound model on each training test case. The other is the extent of the prediction confidence corresponding to the index position of the ground truth of that test case in the output vector of the rewound model.

Orchid is designed with three integration strategies. Suppose that for a test case x in training dataset T, the output vector of rewound model M_{Ri} is V_i for $i = 1$ to N. Further suppose that $X_{correct}$ is the set of output vectors that the corresponding rewound models each classifies x ***correctly*** (i.e., $X_{correct} = \{ V_i \mid M_{Ri}$ classifies x correctly and V_i is the output vector of M_{Ri} on inferencing $x \}$). The three strategies are as follows.

- $\boldsymbol{I_{Smallest}}$ ($\boldsymbol{I_S}$): This strategy is to select the output vector among these vectors in $X_{correct}$, having the smallest confidence value in the ground truth position of x.
- $\boldsymbol{I_{Average}}$ ($\boldsymbol{I_A}$): This integration strategy is to compute an output vector whose elements keep the elementwise average of these vectors in $X_{correct}$.
- $\boldsymbol{I_{Largest}}$ ($\boldsymbol{I_L}$): This strategy is to select the output vector among these vectors in $X_{correct}$, having the largest confidence value in the ground truth position of x.

A corner case of each above strategy is that no rewound model classifies test case x correctly. In this case, the output vector of the given DL model M is used as the integration result (i.e., Orchid does not use any rewound models).

Algorithm 2 Oracle Integration and Model Fixing

Input:
M ← given DL model
N ← number of rewinding models
M_R ← set of rewound DL models
T ← training dataset
E ← test dataset
G ← ground truth for both datasets
I ← integration strategy

Output: O ← integrated oracle information

```
1:  outputs ← ⟨ ⟩
2:  for i=1 to N do
3:  |  outputs =output^⟨M_R[i].Forward(T)⟩
4:  end for
5:  OriginalOutput ← M.Forward(T)
6:  O ← []
7:  for i = 1 to |T| do
8:  |  EnsembleOutputVector=CorrectOutput(outputs[i],G)
9:  |  if EnsembleOutputVector is empty then
10: |  |  O[i]= OriginalOutput[i]
11: |  else
12: |  |  O[i] = I(EnsembleOutputVector)
13: |  end if
14: end for
15: return O
```

Once an integrated output vector is constructed, Orchid uses it as a vector of oracle information to teach the given DL model M to adjust the behavior.

The details of the oracle integration process of Orchid are shown in Algorithm 2. Algorithm 2 accepts these inputs as Algorithm 1. It also takes additional inputs: the rewound model set produced by Algorithm 1 and an integration strategy, denoted as the function ***I*** at line 12.

In lines 1 to 4, Algorithm 2 initializes the output array and collects the output of the penultimate layer of each rewound DL model using forward inference on the whole training dataset T. In line 5, it collects the output of the penultimate layer of the given DL model M. Then, for each test case in T, it contains the set of outputs generated at line 3 for these rewound models that classify the test case correctly (line 8). Next, the algorithm at line 9 determines whether it is feasible to integrate the output vectors using the inputted integration strategy (line 12). If this is not feasible, it uses the original output generated at line 5 (i.e., without teaching effect).
Finally, at line 15, algorithm 2 outputs the integrated oracle information.

The details of the model fixing process with integrated oracle information of Orchid are shown in Algorithm 3. Algorithm 3 accepts these inputs as Algorithm 1. In addition, it takes a few more inputs: the integrated oracle information produced by Algorithm 2, denoted as O at line 4.

At lines 3 and 4, the whole training dataset corresponding with the integrated oracle information and ground truth is divided into a set of batches via the function GetBatch(), where each batch contains a specific number of training

Algorithm 3 Model Fixing with Oracle Information

Input:
M ← given DL model
N ← number of rewinding models
T ← training dataset
E ← test dataset
G ← ground truth for both datasets
O ← integrated oracle information

Output: M_{fixed} ← fixed model

```
1: M_fixed ← M
2: do
3: |  foreach tBatch, gBatch, oBatch in
4: GetBatch(T, G, O) do
5: |  |  M_fixed = M_fixed.Train(tBatch, gBatch, oBatch)
6: |  end for
7: while not IsConverged(M_fixed, E, G)
8: return M_fixed
```

samples. At line 5, it teaches the model M for a few epochs for fine-tuning. The function Trains() trains M using the loss function in Eq. (2), a weighted sum of the L2-norm loss of the integrated vector of oracle information $O(x)$, and the cross-entropy loss of the ground truth, G. After convergence, the algorithm 3 will output the fixed model M_{fixed}.

$$L = \sum_{x \in T} \|M(x) - O(x)\|_2 + w_1 \times L_{ce}(M(x), G) \qquad (2)$$

IV. Evaluation

In this section, we report an evaluation on Orchid. The implementation code, the model weights, and the datasets are available at https://github.com/linghunwhp/Orchid.git.

A. Experimental Setup

We implemented Orchid in Pytorch 1.6.0, which was a popular machine learning library. The experiment has conducted on a machine running Windows 10 equipped with an i7-9700 processor, 64GB RAM, and a single NVIDIA GeForce 2080-Ti GPU with 11GB VRAM.

Datasets. We chose the CIFAR-100 [2] and Tiny-ImageNet-200 [3] datasets to evaluate Orchid. Both datasets were widely used in the experiments for image classification research. The CIFAR-100 dataset contained 60,000 samples and was divided into 100 classes, each with 500 training images and 10 test images. The Tiny-ImageNet-200 dataset contained 100,000 samples of 200 classes, each with 500 training images and 50 test images. All images and their classes were selected from the original ImageNet dataset [30]. Thus, it represents a miniature of the ImageNet classification challenge.

Deep learning model implementations. We selected two representative DL models, MobileNet [4] and ShufflenetV2 [5], to evaluate Orchid. In addition, we used the existing implementations downloaded from a Github site [6].

Hyperparameters. We chose 128 as the batch size. The learning rate was set to 0.1 when the training process started and reduced by a factor of 10 at every 60 epochs. Thus, there

TABLE I. THE BASELINE OF MODELS AND DATASETS

Models	CIFAR-100		Tiny-ImageNet-200	
	Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
MobileNet	99.88%	65.26%	99.53%	59.74%
ShufflenetV2	96.40%	68.86%	97.76%	58.91%

were 200 epochs in total to produce the original DL model. For Algorithm 1, the number of rewound DL models was 7. The learning rates used at lines 6-7 were 0.001. All other parameters followed the default setting of the training scripts of the existing implementations of the models (see above). For the iteration loop from lines 5 to 8 in Algorithm 1, we chose 20 as the total number of iterations. For each iteration used at line 6, there are 10 epochs at line 7. For Algorithm 3, the learning rate to fix a given DL model at line 5 started from 0.001 and was reduced by a factor of 10 at every 15 epochs. The epoch at line 5 was 60, and the weight value for Eq. (2) was 150.

Baseline of the models under fix. After training each of the two DL model implementations on each of the two training datasets, we measured the performances of the trained model using the top-1 accuracy. Let E be the test dataset. The top-1 accuracy is defined as the number of samples in E each classified to a class with the highest confidence and same as ground truth of the sample to the total number of samples in E. We used the top-1 accuracy as the metric because we were interested in the testing and debugging research, where the notion of correctness should be as precise as possible whenever possible to make the assessment on the correctness tight.

TABLE I summarizes the baselines of the four trained DL models. In TABLE I, The first column shows two different machine models, and the second and third columns show two datasets and their training and test accuracy corresponding to the machine learning models, respectively. The result shows that the trained DL models can learn from the training dataset with high recall (96.40% to 99.88%), while the test accuracy was significantly lower. There are many reasons that the research community has not been understood yet. A challenge is that the two datasets have 100 and 200 classes, respectively.

As we have introduced in Section I, one challenge is the conflicting scenario problem. Two exemplified pairs of classes with conflicting scenarios in the MobileNet model trained on the CIFAR-100 dataset are shown in TABLE II,

TABLE II. TESTING RESULTS OF ORIGINAL AND FIXED MODEL BY ORCHID FOR THE CLASS BOY AND DOLPHIN WITH THE MAXIMAL MISCLASSIFICATION RATIO (MOBILENET AND CIFAR-100)

MOBILENET ON CIFAR-100		Actual Prediction			
		Class 11: boy		Class 35: girl	
		before	after	before	after
Oracle	**Class 11: boy**	35%	45%	18%	13%
	Class 35: girl	13%	11%	44%	46%

MobileNet on CIFAR-100		Actual Prediction			
		Class 30: dolphin		Class 73: shark	
		before	after	before	after
Oracle	**Class 30: dolphin**	52%	59%	14%	12%
	Class 73: shark	11%	10%	53%	57%

where the two class pairs C_{boy}-C_{gril} and $C_{dolphin}$-C_{shark} each contains conflicting scenarios where $\xi(C_{boy}) = C_{gril}$, $\xi(C_{girl}) = C_{boy}$, $\xi(C_{shark}) = C_{dolphin}$, and $\xi(C_{dolphin}) = C_{shark}$. The percentage of samples in class C_i that are predicted by the model before and after applying Orchid (with I_A+G strategy) are shown, respectively.

Let us discuss the columns with the heading "before" in Table II. Take the first conflicting pair C_{boy}-C_{girl} for example. 35% of test samples for class C_{boy} are predicted correctly. However, among the remaining 65% of test samples for C_{boy}, a noticeable proportion (18%) is misclassified into C_{girl}. Note that there are 99 possible classes for misclassification, meaning that the average misclassification ratio per class is only 0.657%. A noticeable proportion (13%) of test samples of C_{girl} is misclassified as C_{boy} as well. Similarly, the class pair $C_{dolphin}$-C_{shark} share a trend similar to C_{boy}-C_{girl} that the highest and noticeable proportion of each class is misclassified into the other class in the pair.

Orchid and its variants and conventional ensemble teacher. We applied Orchid to fix each DL model under fix using each of the three integration strategies (i.e., strategies I_S, I_A, and I_L in Section III.D) to compute the integrated output vectors (line 13 of Algorithm 2). As presented in Section III, Orchid teaches each given DL model using a weighted loss of the integrated output vector and the ground truth. They resulted in three combinations, denoted as $\boldsymbol{I_S}$**+**$\boldsymbol{G}$, $\boldsymbol{I_A}$**+**$\boldsymbol{G}$, and $\boldsymbol{I_L}$**+**$\boldsymbol{G}$, respectively.

TABLE III. TEST ACCURACY OF DL MODELS FIXED BY CONVENTIONAL ENSEMBLE AND ORCHID

Architecture	Dataset	Baseline (Model before Fix)	Conventional	Strategy I_S		Strategy I_A		Strategy I_L	
				I_S	I_S+G	I_A	I_A+G	I_L	I_L+G
MobileNet	CIFAR 100	65.26	66.76	63.45	64.54	68.08	**68.11**	55.03	57.02
	Tiny ImageNet 200	59.74	60.49	60.18	60.46	62.09	**62.15**	48.23	51.78
ShufflenetV2	CIFAR 100	68.86	70.16	69.08	69.29	71.24	**71.54**	59.81	61.72
	Tiny ImageNet 200	58.91	59.85	60.08	60.14	61.22	**61.48**	49.37	50.25
Average		63.19	64.32	63.2	63.61	65.66	**65.82**	53.11	55.19

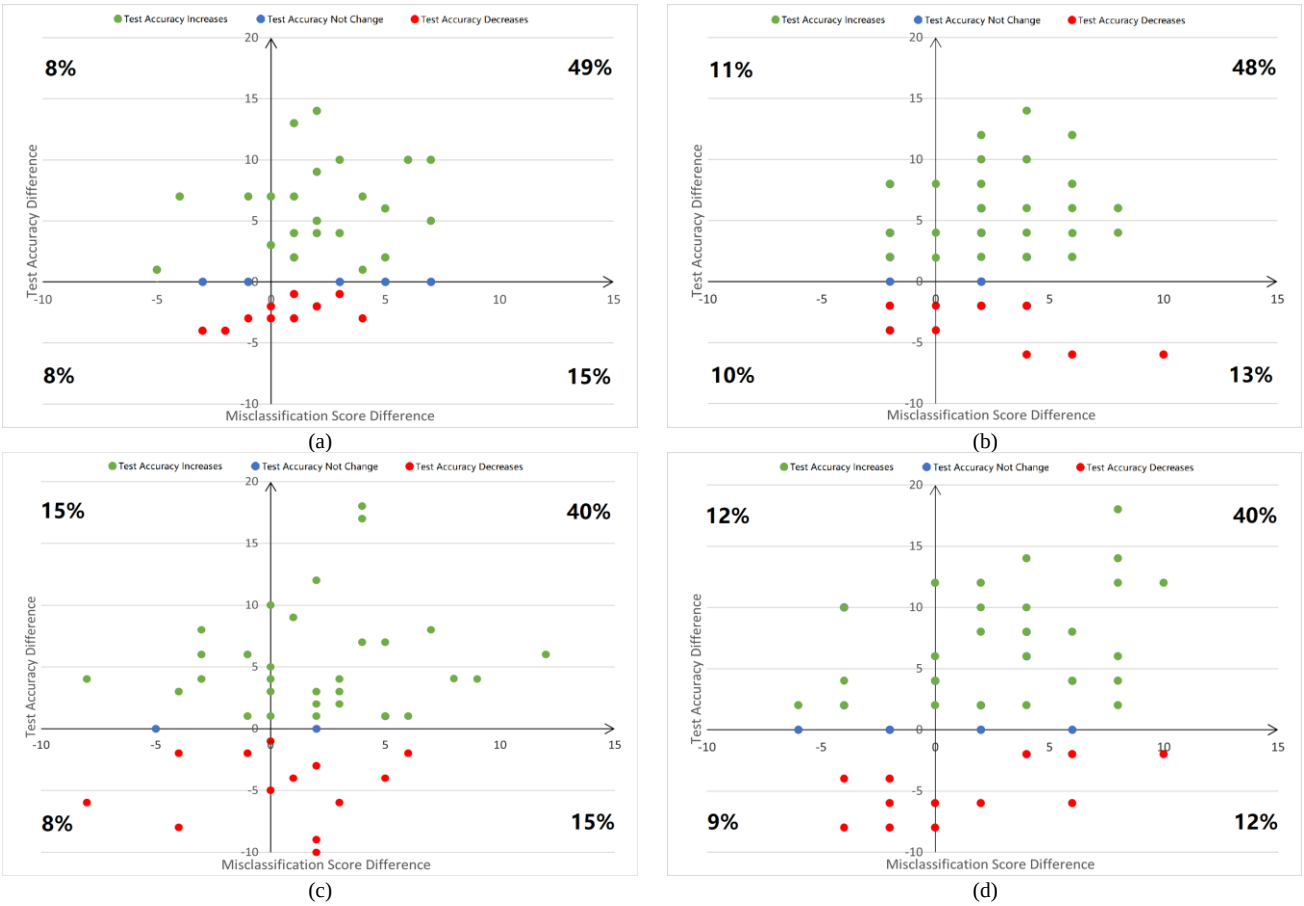

Figure 5. Misclassification difference and test accuracy difference distribution for specified classes of the model under fix(blue) and fixed model(Orange) for (a) MobileNet model on Cifar-100 dataset, (b) MobileNet model on Tiny-ImageNet-200 dataset, (c) ShufflenetV2 model on Cifar-100 dataset and (d) ShufflenetV2 model on Tiny-ImageNet-200 dataset.

The core idea of Orchid is to teach a DL model under fix by the integrated output vector. So, we also set the weight of the weight loss for Eq. (2) of Algorithm 3 to 0 for w_1. They result in three variants of Orchid when applying the strategies, denoted as $\boldsymbol{I_S}$, $\boldsymbol{I_A}$, and $\boldsymbol{I_L}$, respectively.

Conventional knowledge distillation based on an ensemble of teachers [24] is composed of a few models produced by the same training scripts. Such an ensemble is formed by averaging the output vectors of these models or randomly selecting an output vector that correctly classifies the given sample. Testers may also use such conventional ensemble to teach the DL model under fix. We also compare Orchid to this conventional ensemble approach, which averages the correct output vectors. First, we reran the training scripts of the two downloaded implementations to produce ensembles having the same number of rewound models used by Orchid in the experiment. Then, we trained the DL model under fix with the output vector of each such conventional ensemble using the fixing procedure of Algorithm 3. Finally, we denote the result of this conventional ensemble by ***Conventional***.

B. Result and Data Analysis

The test accuracies of the DL models fixed by *Conventional* and each variant of Orchid are summarized in TABLE III. The first row shows the name of the deep neural network architecture, training dataset, baseline of the given model, the fixed model by the conventional method, and three fixed models by Orchid with I_S, $I_{A,}$ and I_L strategies. G in TABLE III means the ground truth for each training dataset. The second and third rows show the experiment results for each combination of architecture and dataset. The last row shows the average value for each column.

Conventional can improve the test accuracies by 0.75%-1.5% with an average of 1.23%.

Among the variants of Orchid, I_S and I_S+G produced inconsistent test accuracies across the four model-dataset combinations. On Cifar-100, the fixed MobileNet models by I_S and I_S+G achieved lower performances than the baseline. Both I_L and I_L+G are also inferior choices because the changes in test accacury across all model-dataset combinations are negative. The loss in test accuracy is surprisingly large. The result shows that using a reference test

oracle with the highest prediction confidence to guide fixing the DL model under fix is ineffective. At the same time, using a reference test oracle with the lowest prediction confidence may not give a consistent improvement. They both show the problem of using a single model as a teacher to guide the fixing process.

I_A and I_A+G are the two most effective variants among all Orchid variants studied in the experiment. For each model-dataset combination, either one is more effective than all other Orchid variants and *Conventional*. They improve over the baseline by 2.47% and 2.62% in test accuracy, respectively. I_A is more effective than *Conventional* by 1.08%-1.6% with an average of 1.343%, and I_A+G is more effective than *Conventional* by 1.35%-1.66% with an average of 1.51%. Moreover, I_A+G is always more effective than I_A. We also observe that I_S, I_A, and I_L, are less effective across all models than I_S+G, I_A+G, and I_L+G, respectively. The results show that using both the oracle information of the reference test oracle and the ground truth is more effective than using the reference test oracle alone.

We have also analyzed the effects of Orchid on conflicting scenarios between classes with ε = 0.10. We define the misclassification score for class C of the model under fix minus the misclassification score for class C of the fixed model, which is fixed by Orchid with the I_A+G strategy, as the *misclassification score difference* for class C. We define the test accuracy for class C of the fixed model, which is fixed by Orchid with the I_A+G strategy, minus the test accuracy for class C of the model under fix as the *test accuracy difference* for class C. The four subfigures (a), (b), (c), and (d) in Figure 5 present the distributions of classes in terms of misclassification score difference and test accuracy difference of the MobileNet model on the Cifar-100 dataset, the MobileNet model on the Tiny-ImageNet-200 dataset, the ShufflenetV2 model on the Cifar-100 dataset and the ShufflenetV2 model on the Tiny-ImageNet-200 dataset, respectively. In each plot, the x-axis is the misclassification score difference, and the y-axis is the test accuracy difference (in percentage).

In each plot in Figure 5, from the top-left quadrant and the clock-wise, the percentages in bold mean the proportions that (1) the misclassification score decreases and the test accuracy increases, (2) misclassification score increases, and the test accuracy decreases, (3) misclassification score decreases, and the test accuracy decreases, and (4) misclassification score increase and the test accuracy decreases, respectively. The summation of the four proportions in each plot may not be 100% because ε is not zero.

Figure 5(a) shows that the fixed MobileNet model alleviates the conflicting scenarios on the Cifar-100 dataset for 70% of the classes and increases the test accuracy for 60% of the classes. From the first quadrant, we notice that the test accuracy of 49% classes is improved when the decreased misclassification score difference. And the test accuracy is improved by 14% at most.

Figure 5(b) shows that the fixed MobileNet model alleviates the conflicting scenarios on the Tiny-ImageNet-200 dataset for 66% of the classes and increases the test accuracy

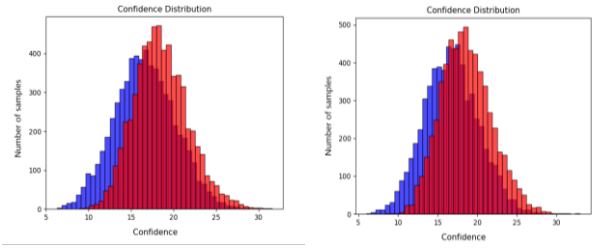

Figure 6. Confidence distribution in the Model under fix (red) and Rewound Model (blue) on the corresponding rewinding datasets (MobileNet and CIFAR-100). The x-axis is the confidence of sample, and the y-axis shows the number of samples attaining the confidence value as indicated by the x-value.

for 69% of the classes. From the first quadrant, we notice that the test accuracy of 48% classes is improved when the decreased misclassification score difference. And the test accuracy is improved by 14% at most.

Figure 5(c) shows that the fixed ShufflenetV2 model alleviates the conflicting scenarios on the Cifar-100 dataset for 60% of the classes and increases the test accuracy for 67% of the classes. From the first quadrant, we notice that the test accuracy of 40% classes is improved when the decreased misclassification score difference. And the test accuracy is improved by 18% at most.

Figure 5(d) shows that the fixed ShufflenetV2 model alleviates the conflicting scenarios on the Tiny-ImageNet-200 dataset for 58% of misclassification scenarios and increases the test accuracy for 65% of the classes. From the first quadrant, we notice that the test accuracy of 40% classes are improved when the decreased misclassification score difference. And the test accuracy is improved by 18% at most.

The distribution of points in each plot in Figure 5 shows that in at least 40% of conflicting scenarios, the test accuracy of these classes could be improved to some extent by the fixed model with the misclassification of the classes in conflicting scenarios alleviated.

Two examples of the confidence distribution of rewinding datasets in the model under fix and the fixed model are case studied and summarized in the "after" columns of TABLE II. The table as a whole illustrates the differences in test accuracy between the model under fix and the fixed model in the pair of classes 11 and 35 as well as between the pair of classes 30 and 73, respectively. First, the conflicting scenario problem (as shown by the misclassification score) had also been reduced. For example, the misclassification in test samples between classes 11 and 35 has been reduced from 18% to 13% and 13% to 11%. Second, the test accuracy of the four classes influenced by the conflicting scenarios had been improved, and class 11 was improved noticeably, from 35% to 45%.

We have also extended the experiment to explore alternative definitions of rewinding vectors. We have experimented with a random rewinding vector as well as a distribution vector that the value of each class in the distribution vector is the proportion of correct classification of the test samples of that class, where the sum of these values in either rewinding vector are normalized to 1. We found that the produced fixed DL models were less accurate than the ones produced by Orchid presented in this paper. However, on

using I_A+G, they were still more effective than *Conventional*. We leave the reporting of the results in future work.

An intuition of our work is that a rewound model can learn less determinately on rewinding datasets so that the conflicting scenarios can be alleviated in the rewound model with a higher probability. To understand the situation better, we conducted a case study on two rewound models to visualize their prediction confidence values on their corresponding rewinding datasets. (We note that all plots on other rewound models follow the same patterns and have similar shapes as described below.) The visualizations are shown in Figure 6. In each plot, there are two distributions. The one in red (i.e., the one with a higher peak) is the distribution for the original trained MobileNet on Cifar-100. The one in blue (i.e., the one with a lower peak) is the distribution produced by one of the seven rewound models produced by Orchid. In each plot, the *x*-axis is the confidence of the sample, and the *y*-axis shows the number of samples attaining the confidence value as indicated by the *x*-value. We observe that the confidence values for the distribution in blue are in general smaller than those for the distribution in red, where the whole confidence distribution in blue moves to the left compared with the confidence distribution in red. The visualization follows the expectation that test cases in the rewinding datasets become lesser learned by the rewound models than the DL model under fix.

C. Threats to Validity

A major threat is the bugs in the platform and the downloaded implementations. We chose widely used platform versions and reused existing implementations to reduce our biases. In the training processes, we did not encounter situations that produced weird results.

We only used four model-dataset combinations in the experiment. Using more and diverse combinations will strengthen the generalization of the results. We have not varied the hyperparameters of the training process and the algorithms of Orchid in the experiment. We leave them as future work.

We compared the models in terms of test accuracy using the top-1 accuracy metric. Using other metrics for comparison may give different results. From the current results, the misclassification between two classes of the same fixed models is still unclear to be smaller than the original models subject to fixing. We have shown that for pairs C_{boy}-C_{girl} and $C_{dolphin}$-C_{shark}, the misclassifications are reduced. Given that there are 98 other classes for misclassification in the Cifar-100 dataset, the reduction is significant. However, the test accuracies of the fixed DL models are not improved to a large extent due to the problem nature that even using the state-of-the-art models with top-1 accuracy is still very challenging to be very accurate.

V. Related Work

A. Pseudo Oracle Testing in Machine Learning

Metamorphic Testing [7, 8, 9] mitigates the pseudo-oracle problem in machine learning testing. Its idea is to cross-check the inference results of multiple test cases to identify violations of the relations expected to hold in general across these test cases.

DeepXplore [16] aims to test DL models and increase the safety, robust and security of real-world domains. Furthermore, it shows that similar functionality for the same or similar task domains could be used as cross-referencing oracles to identify the error in some corner cases without knowing the ground truth.

An example of metamorphic testing comes from testing search engines based on NLP and machine learning systems, proposed by Zhou et al. [9]. They offered a user-oriented approach to testing major web searching engines, such as Google, Bing, and Baidu. The results prove that the method can effectively alleviate the Oracle problems and the lack of specification challenges in verifying, validating, and evaluating large complex software systems.

Xie et al. [12] proposed an approach to applying metamorphic testing to machine learning classifiers, aiming to address the software quality of machine learning models. They implemented the k-Nearest Neighbors and the naive Bayes classifier machine learning models and conducted an evaluation on Weka, a real-world machine learning framework. The evaluation results show that the approach is practical to alleviate the oracle problems in machine learning classification algorithms.

Tian et al. [10] proposed DeepTest to automatically test autonomous cars based on metamorphic testing to improve the quality and reliability of autonomous systems. The approach transforms the real-time images captured from the driving cars with many metamorphic transformations, including scaling, zooming, adding rain, adding fog, and getting the altered images. The metamorphic relation is that the autonomous driving decisions should have minimal divergence between the original image and the corresponding altered images. The experiments show that the approach helps to construct more robust DNN-based systems.

B. Ensemble Learning and Knowledge Distillation

The ensemble learning method in machine learning [11] has been studied for years. It is a kind of machine learning technique to combine several base models aiming to construct a more accurate and robust predictive model. The intuition of ensemble learning is that even if one of the weak classifiers gets the wrong prediction, other weak classifiers can correct the error. There are many types of ensemble learning methods [11], such as voting, bootstrap aggregating (referred to as Bagging), boosting, such as adaptive boosting [13] (referred to as AdaBoost), and stacking. The ensemble learning method mixes multiple models and significantly improves prediction performance, but the prediction results are incomprehensible. Furthermore, it needs to run all the machine learning models during the inference stage and make the inference consume more computational power.

Developers proposed another method, the knowledge distillation method [14], to transfer the knowledge from a large or complex machine learning model to a smaller one. The large or complex models often have higher knowledge capacity than the smaller model, so we could use the high-

capacity models to teach or guide the low-capacity model towards being better. The knowledge distillation method has been applied to many task domains, including image classification, object detection, speech recognition, and so forth.

Chebotar et al. [17] proposed a new approach to distilling knowledge from ensembles of deep learning models for speech recognition in the NLP domain. First, they trained many teacher models with different DL architectures and different training objectives so that the ensembles of DL models gain a higher capacity. Then, they distill and teach the student model by minimizing the cross-entropy between the output of teacher and student models. Unlike their work, Orchid can improve the student model by not using base models with different DL architectures.

Chen et al. [25] proposed a novel framework for object detection tasks based on the knowledge distillation technique. The relation among teacher models of the object detection task is much more complex than the simple classification tasks due to region proposals and less voluminous labels. And they address these problems by weighted cross-entropy, bounded loss, and so on and improve the system significantly.

Although it is a truth that the knowledge distillation technique could help to increase the capacity of the weak student model from the high-capacity teacher model in many task domains, it is difficult to explain and give a justification answer. Cheng et al. [18] proposed a new method to interpret knowledge by analyzing the knowledge in the intermediate layers of the DL model. Their experiments show that knowledge distillation guides the student model to learn more task-relevant knowledge and less task-irrelevant knowledge.

VI. CONCLUSIONS

In this paper, we have proposed the conflicting scenario problem. The conflicting scenario problem is that at least one class's misclassification score of the trained model with the test dataset exceeds the threshold as defined. We have also presented Orchid, the first work to address the conflicting scenario problem and improve the accuracy of the model under fix. Orchid partitions the training dataset into several non-overlapping sub-training datasets. Then, it rewinds the model under fix with respect to each such data subset. Next, it integrates the output vectors of these rewound models that classify the sample to groundtruth correctly. Finally, it teaches the model under fix to minimize the loss against these integrated output vectors and the corresponding ground truth of the training samples. The outcome is a fixed DL model. We have reported an experiment to evaluate Orchid. The results demonstrate the feasibility of Orchid and illustrate the possible effects of addressing the conflicting scenario problem.

REFERENCES

[1] M. D. Davis and E. J. Weyuker, "Pseudo-oracles for Non-testable Programs," in Proceedings of the ACM '81 Conference, New York, NY, USA, pp. 254–257, 1981.

[2] A. Krizhevsky, G. Hinton, "Learning multiple layers of features from tiny images," Technical report, University of Toronto, 2009.

[3] Tiny-ImageNet-200 Dataset. http://cs231n.stanford.edu/tiny-imagenet-200.zip.

[4] Howard, A. G., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv e-prints, 2017, unpublished.

[5] N. Ma, X. Zhang, H. Zheng, and J. Sun. "ShufflenetV2: Practical Guidelines for Efficient CNN Architecture Design," In *Proc*. European Conference on Computer Vision (ECCV), pp. 116-131, 2018.

[6] Weiaicunzai (2020) pytorch-cifar100 [Source Code]. https://github.com/weiaicunzai/pytorch-cifar100

[7] Jie Zhang, Junjie Chen, Dan Hao, Yingfei Xiong, Bing Xie, Lu Zhang, and Hong Mei. Search-based inference of polynomial metamorphic relations. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, pages 701–712. ACM, 2014.

[8] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Technical Report HKUST-CS98–01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.W

[9] Z. Q. Zhou, S. Xiang and T. Y. Chen, "Metamorphic Testing for Software Quality Assessment: A Study of Search Engines," in IEEE Transactions on Software Engineering, vol. 42, no. 3, pp. 264-284, 1 March 2016.

[10] Tian, Yuchi, Kexin Pei, Suman Jana, and Baishakhi Ray. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars." In Proceedings of the 40th international conference on software engineering, pp. 303-314. 2018.

[11] Thomas G Dietterich. "Ensemble methods in machine learning." In International workshop on multiple classifier systems, pages 1–15. Springer, 2000.

[12] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," Journal of Systems and Software, vol. 84, no. 4, pp. 544–558, Apr. 2011.

[13] Schapire, Robert E. "Explaining adaboost." Empirical inference. Springer, Berlin, Heidelberg, 37-52, 2013.

[14] Chen, Guobin, et al. "Learning efficient object detection models with knowledge distillation." Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017.

[15] Asami, Taichi, Ryo Masumura, Yoshikazu Yamaguchi, Hirokazu Masataki, and Yushi Aono. "Domain adaptation of dnn acoustic models using knowledge distillation." In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5185-5189, 2017.

[16] Pei, Kexin, Yinzhi Cao, Junfeng Yang, and Suman Jana. "Deepxplore: Automated whitebox testing of deep learning systems." In proceedings of the 26th Symposium on Operating Systems Principles, pp. 1-18, 2017.

[17] Chebotar, Yevgen, and Austin Waters. "Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition." In Interspeech, pp. 3439-3443, 2016.

[18] Cheng, Xu, Zhefan Rao, Yilan Chen, and Quanshi Zhang. "Explaining knowledge distillation by quantifying the knowledge." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12925-12935, 2020.

[19] Weyuker, Elaine J. "On testing non-testable programs." The Computer Journal 25, no. 4, pp. 465-470, 1982.

[20] Dietterich, Tom. "Overfitting and undercomputing in machine learning." ACM computing surveys (CSUR) 27, no. 3, pp. 326-327, 1995.

[21] Schaffer, Cullen. "Overfitting avoidance as bias." Machine learning 10, no. 2, pp. 153-178, 1993.

[22] Tung, Frederick, and Greg Mori. "Similarity-preserving knowledge distillation." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1365-1374. 2019.

[23] Barr, Earl T., Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "The oracle problem in software testing: A survey." IEEE transactions on software engineering 41, no. 5, pp. 507-525, 2014.

[24] Peer, David, Sebastian Stabinger, and Antonio Rodriguez-Sanchez. "Conflicting bundles: Adapting architectures towards the improved training of deep neural networks." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 256-265. 2021.

[25] Fukuda, Takashi, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. "Efficient Knowledge Distillation from an Ensemble of Teachers." In Interspeech, pp. 3697-3701, 2017.

[26] Chen, Guobin, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. "Learning efficient object detection models with knowledge distillation." In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 742-751. 2017.

[27] Y. LeCun, Y. Bengio, G. Hinton, “Deep learning,” Nature, vol.521, pp.436-444, 2015.

[28] PyTorch Training a Classifier, https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.

[29] Scardapane, Simone, and Dianhui Wang. "Randomness in neural networks: an overview." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7.2, e1200, 2017.

[30] Braiek, Houssem Ben, and Foutse Khomh. "On testing machine learning programs." Journal of Systems and Software 164, pp. 110542, 2020.

[31] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.