A Combinatorial Interaction Testing Method for Multi-Label Image Classifier

Peng Wang, Shengyou Hu, Huayao Wu*, Xintao Niu, Changhai Nie, and Lin Chen
State Key Laboratory for Novel Software Technology and
School of Computer Science, Nanjing University, China
{pengwang, husy}@smail.nju.edu.cn, {hywu, niuxintao, changhainie, lchen}@nju.edu.cn

***Abstract*—Multi-label image classification is a critical task in computer vision, in which the correlations between labels are typically exploited by modern classifiers for an effective classification. In this study, we propose LV-CIT, a black-box testing method that applies Combinatorial Interaction Testing (CIT) to systematically test the ability of classifiers to handle such correlations. Specifically, LV-CIT views each label of the label space as an input-parameter taking binary values (indicating whether an object appears in an image), and manages to generate a label value covering array as the set of test cases to cover certain combinations of label values. Then, for each test case, LV-CIT relies on an object library to generate composite test images that perfectly match the specified labels, and reports classification errors if such labels cannot be correctly recognised. The experimental results on two popular datasets with six state-of-the-art image classifiers show that LV-CIT is more efficient than the existing CIT tools in generating label value covering arrays. LV-CIT is also effective in errors revelation, as it can find 111% more errors by using 20% fewer test images than the existing methods for testing multi-label image classifiers.**

***Index Terms*—Black-box Testing, Test Input Generation, Deep Neural Network**

I. Introduction

Multi-label image classification is a critical task in computer vision, and has been widely adopted in many application domains, such as medical image analysis [1] and autonomous driving [2]. Currently, despite the use of Deep Neural Networks (DNNs) has greatly improved the performance of modern multi-label image classifiers, the DNN models trained remain prone to incorrect classifications, which might then lead to serious consequences. For example, it has been reported that a driverless taxi has failed to recognise a woman hit by another car and dragged her 20 feet[1]; and a facial recognition system has failed to recognise a thief and made a man being falsely charged with theft[2]. Hence, ensuring the quality of modern image classifiers is an important endeavour, in which testing methods that are able to thoroughly examine their actual classification ability should be developed.

Currently, white-box testing has became a popular choice for testing DNN-based systems. These methods seek to exploit the internals of the network structure or training data to construct test cases [3]–[10], but such information might not always be readily available in practice due to the privacy or security concerns. In contrast, the use of black-box testing methods is attracting extensive attentions in the literature [11]–[20]. Various diversity-based strategies [17]–[20] are proposed to guide the selection of test cases for DNN-based systems.

Although there exist a number of methods to test DNN-based systems, they are not in a position to examine the core specification of the multi-label image classification task (that is, to what extend the multiple objects present in an image can be correctly recognised). In particular, to effectively address the huge and complex label spaces, modern DNN-based multi-label image classifiers typically manage to learn the correlations between multiple labels to improve the classification accuracy [21]–[24]. Accordingly, the performance of classifiers tends to be highly susceptible to the inappropriate correlations captured, which then suggests the testing of certain interactions between labels for triggering such correlations-related errors.

Recently, Hu et al. [19] proposed the *ATOM* method to evaluate the ability of image classifiers to handle correlations between multiple objects. This method first enumerates all possible combinations of the labels in the label space, and then utilises an image search engine to find test images that are likely to contain objects of such labels. However, the primary focus of *ATOM* is on the co-occurrence of objects (e.g., when both $keyboard$ and $mouse$ appear in an image, whether the classifier can correctly recognise them). It might thus overlook other possible interactions between these objects and be ineffective in triggering some types of errors (e.g., whether the classifier tends to output both $keyboard$ and $mouse$ even if there is only one of them in the image). In addition, the images collected from Internet might be inaccurate and contain unexpected objects other than the labels specified (e.g., there is an unexpected $mouse$ in the image when collecting images containing a $keyboard$ only). This could also influence the examination of the image classifier's actual ability on certain combinations of labels.

To further improve the efficiency and effectiveness of multi-label image classifiers testing, in this study, we propose the LV-CIT method that applies Combinatorial Interaction Testing (CIT) to systematically test the potentially enormous *label value combinations* involved in the label space. Specifically,

* Corresponding author

[1]https://jalopnik.com/cruise-taxi-runs-over-stops-on-top-of-woman-hit-by-oth-1850895138

[2]https://www.the-independent.com/news/world/americas/crime/louisiana-police-facial-recognition-lawsuit-b2419085.html

LV-CIT will create a test model of CIT to depict the input space of the image classifier, in which each label is viewed as an input parameter that takes binary values (value one and zero indicates the presence and absence of the label, respectively). Accordingly, a combination of τ label values is referred to as a *τ-way label value combination* (the value of τ is called the test strength or covering strength), and LV-CIT will generate a *τ-way label value covering array* that covers every such label value combination at least once as the set of test cases.

To implement the above testing process, a straightforward choice is to use existing CIT tools [25] to generate the covering arrays. However, considering that test images containing an excessive number of objects can become distorted and less discernible, introducing interaction-irrelevant errors. Thus, there is a need to add certain restrictions to the test case generation. To this end, LV-CIT introduces a counting constraint to limit the number of objects that can appear together in a test image. Since such a constraint is not well supported or time-consuming to handle in existing CIT tools [26]–[30], a novel adaptive sampling algorithm is developed in this study to improve the speed of covering array generation while maintaining the array sizes as small as possible.

Once the label value covering array is obtained, LV-CIT then relies on a previously created object library to generate composite images as the test inputs. Each object image in the library contains one complete object of the given label, which should be easily recognisable by human and also correctly classified by the image classifiers under test. In this way, if the image classifier cannot make the correct classification when several objects appear together, we know that such a defect is due to the interactions of these objects. Moreover, to increase the diversity of test inputs, LV-CIT will generate multiple test images for each test case in the covering array, in which different objects are selected and composed in different ways (including different scaling ratios, and degrees of overlapping).

We note that LV-CIT utilises composite images, rather than realistic images, as the test inputs. This is because the aim of LV-CIT is to examine the ability of classifiers to handle interactions between objects, which requires test images that only contain features of target objects (i.e., contain no features of any other objects). By contrast, realistic images typically contain many unexpected objects, which can be tedious and error-prone for testers to determine whether certain objects are indeed included (or excluded), especially for large label spaces. We also note that such composite images can be valid test inputs for testing multi-label image classifiers, because all object images used for composition are easily recognised by humans [31]. Moreover, in contrast to existing studies [11]–[15] (including *ATOM* [19]) that usually rely on metamorphic testing to determine testing results, the use of composite images enables a straightforward test oracle identification, as any mismatching between classification outputs and corresponding test cases indicate an error of the image classifier.

To evaluate the efficiency and effectiveness of LV-CIT, we created object libraries based on two popular datasets of multi-label image classification, *VOC* [32] and *COCO* [33], and conducted experiments on six state-of-the-art DNN-based image classifiers (three for each dataset). The experimental results reveal that the adaptive sampling algorithm used in LV-CIT is efficient in generating 2-way label value covering arrays, as it can be 3.6 times faster than the existing CIT tool, *ACTS* [29], for small label spaces, and remain efficient for large label spaces (where the application of *ACTS* becomes infeasible). LV-CIT also outperforms the existing testing method for multi-label image classifiers, *ATOM* [19], in terms of error detection ability, as it can, on average, use 20% fewer test images to reveal 111% more errors when testing 2-way label value combinations. By analysing the testing results, we can further find some distinct errors that are not detected by the existing *ATOM* [19] method (e.g., a classifier outputs both bus and $person$ even if there is only a bus in the image due to the overfitting to the label combination $\{bus, person\}$).

Summing up, the main contributions of this study include:

- We introduce the concept of τ-way label value combination for testing multi-label image classifiers, and propose to use a τ-way label value covering array to systematically test the interactions between any τ labels of the label space.
- We implement the LV-CIT method, which uses a novel adaptive sampling algorithm to efficiently generate τ-way label value covering arrays, and utilises image composition to generate test images that perfectly match labels specified by each test case in the covering array.
- We have created object libraries for two popular datasets of multi-label image classification, and conducted experiments on six state-of-the-art DNN-based image classifiers to demonstrate the efficiency and effectiveness of the LV-CIT method.

The organisation of this paper is as follows: Section II introduces the core idea of applying combinatorial interaction testing to test multi-label image classifiers. Section III presents the proposed LV-CIT method. Section IV describes the experiment design for evaluating LV-CIT. Section V presents the experimental results. Section VI discusses the threats to validity. Section VII summarises related works, and Section VIII concludes this paper.

II. Combinatorial Interaction Testing for Multi-Label Image Classifiers

Combinatorial Interaction Testing (CIT), or Combinatorial Testing (CT), is a popular testing technique that can systematically examine the interactions of certain parameter values in a Software Under Test (SUT) [34]. Its application starts from the creation of a *test model*, in which testers need to identify a set of parameters that might influence the behaviours of the SUT, a finite value domain for each parameter, and potential constraints between parameter values (e.g., some parameters cannot take specific values at the same time). This test model exactly depicts the input space of the SUT, and a test case can then be constructed by assigning every parameter a fixed value. Note that the number of all possible test cases is usually too big to be exhaustively tried out in

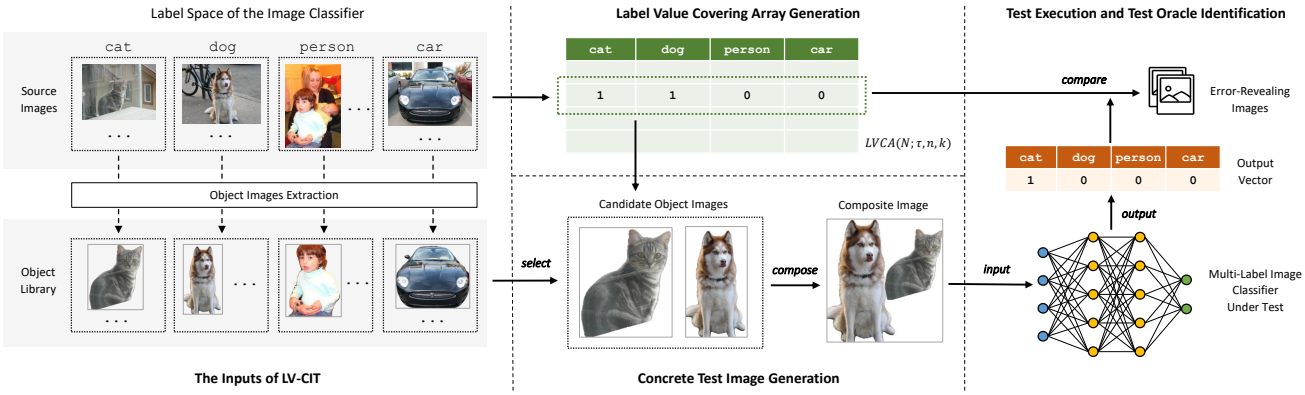


Fig. 1. The workflow of the LV-CIT method.

TABLE I
A 2-WAY LABEL VALUE COVERING ARRAY $LVCA(6;2,4,2)$

	$l_1 : cat$	$l_2 : dog$	$l_3 : person$	$l_4 : car$
t_1	1	1	0	0
t_2	1	0	1	0
t_3	1	0	0	1
t_4	0	1	1	0
t_5	0	1	0	1
t_6	0	0	1	1

practice. CIT therefore manages to create a τ-way constrained *covering array* (with as small size as possible) as the test suite. Such a covering array guarantees to cover all constraint-satisfying combinations between any τ parameter values, and consequently, software failures due to interactions of no more than τ parameters can be systematically examined.

In this study, we seek to use CIT to test multi-label image classifiers in a black-box manner. Specifically, we define the set of all labels that the classifier is developed to predict as its *label space*, denoted as $L = (l_1, l_2, \ldots, l_n)$, and view each of these labels as an input parameter of the test model. Then, as each label l_i can either be present or not in an image, its value domain will be $V_i = \{1, 0\}$ for $1 \leq i \leq n$ ($v_i = 1$ indicates that an object of label l_i is present in an image). Accordingly, we can construct an abstract test case for testing the image classifier by assigning every label a fixed value, and we refer to the combination of τ label values ($\tau \geq 1$), $\{l_{i_1} = v_{i_1}, l_{i_2} = v_{i_2}, \ldots, l_{i_\tau} = v_{i_\tau}\}$, as a τ-way *label value combination*. For example, given a label space $(cat, dog, person, car)$, an abstract test case can be represented as $\{cat = 1, dog = 1, person = 0, car = 0\}$ (containing features of cat and dog, but not $person$ or car), or $[1, 1, 0, 0]$ for simplicity, indicating a test scenario that the image classifier is supposed to recognise. This test case covers $\binom{4}{2} = 6$ distinct 2-way label value combinations, including $\{cat = 1, dog = 1\}$, $\{cat = 1, person = 0\}$, $\{cat = 1, car = 0\}$, $\{dog = 1, person = 0\}$, $\{dog = 1, car = 0\}$, and $\{person = 0, car = 0\}$.

Intuitively, there seems no explicit constraint between the label values, because as long as the feature of a label is included in the image, the image classifier is expected to correctly recognise it. However, due to the limit of the recognition resolution, when a test image includes a substantial number of objects, these objects may be distorted and become less discernible, introducing interaction-irrelevant errors. To account for this, we introduce a *counting constraint*, $\sum_{i=1}^{n} v_i \leq k$, to our test model to limit the number of labels that can appear in a test case (i.e., at most k labels can take value one in a row of the covering array).

Within the above test model, we can now generate a τ-*way Label Value Covering Array* (*LVCA*) as the test suite. Given a label space with size n, the label combination covering strength τ ($1 < \tau \leq n$), and the counting constraint variable k ($1 \leq k \leq n$), a τ-way label value covering array, denoted as $LVCA(N; \tau, n, k)$, is an $N \times n$ array that satisfies the following conditions: 1) Each row of the array, $[v_1, v_2, \ldots, v_n]$, is a test case, in which $\sum_{i=1}^{n} v_i \leq k$ is satisfied; and 2) Every constraint-satisfying τ-way label value combination appears at least once in the corresponding $N \times \tau$ subarray.

Table I, for example, gives a 2-way label value covering array, $LVCA(6; 2, 4, 2)$, for the label space $(cat, dog, person, car)$. Note that we have $k = 2$ here, so there are at most two labels that can take value one in each row. The six rows in this array covers all possible 2-way label value combinations. Take labels cat and dog as the example, all possible $2 \times 2 = 4$ combinations ($\{cat = 1, dog = 1\}$, $\{cat = 1, dog = 0\}$, $\{cat = 0, dog = 1\}$ and $\{cat = 0, dog = 0\}$) appear at least once in this array.

III. THE LV-CIT METHOD

This section presents our LV-CIT method that uses combinatorial interaction testing to systematically test label value combinations for multi-label image classifiers. Fig. 1 gives the workflow of LV-CIT, which takes the label space of the image classifier under test and a corresponding object library as the input. First, LV-CIT will create the CIT test model and generate a τ-way label value covering array as a set of abstract test cases. Next, for each abstract test case, LV-CIT will rely on the object library to generate a series of concrete test images

by an image composition method. These test images will then be given to the image classifier under test, and finally, LV-CIT compares the classification output against the abstract test case to identify the test oracles.

A. Label Value Covering Array Generation

Covering array generation is a core research field in combinatorial interaction testing, and there are many algorithms and tools available [25]. However, the counting constraint involved in our test model is not common in traditional CIT, and is thus not well supported by existing tools. For example, for tools that use *CASA* modelling format [26], [27], [35], there is no way to describe the counting constraint. Although we can translate the counting constraint into a more general form of forbidden tuples (i.e., a list of τ-way combinations that are not allowed to appear) or implies constraints, it will be less efficient, because the counting constraint will lead to a huge number of such constraint expressions (e.g., for a label space with $n = 80$ and $k = 4$, the counting constraint $\sum_{i=1}^{80} v_i \leq 4$ will result in approximately 1.6 million forbidden tuples). To our best knowledge, *ACTS* [29] is the only available tool that can use one constraint expression (i.e., $\sum_{i=1}^{n} v_i \leq k$) to describe the counting constraint. However, this tool employs minimal forbidden tuples and constraint solver for constraints handling, which could be costly and hard to scale for large label spaces.

To efficiently generate τ-way label value covering arrays, in this study, we first give a straightforward *Baseline* method. This method simply enumerates all *all-one* τ-way label value combinations (i.e., all the τ labels take value one, like $\{cat = 1, dog = 1\}$) of the test model, and generates one test case to cover each of them (the remaining $n - \tau$ labels in each test case will take value zero). Note that the remaining not all-one combinations are easily covered by the way. This will result in a total number of $\binom{n}{\tau}$ test cases, and these test cases exactly constitute a τ-way label value covering array when $\tau \leq \lfloor \frac{n}{2} \rfloor$. For example, the 2-way label value covering array shown in Table I is generated by *Baseline* method. Here we have $\tau = 2$, so the method will generate $\binom{4}{2} = 6$ test cases, each of which corresponds to an *all-one* 2-way label value combination.

Although the above *Baseline* method is fast to run, the size of the array generated could be unnecessarily large. Hence, we further propose a novel adaptive sampling algorithm to generate τ-way label value covering arrays for LV-CIT. Algorithm 1 gives the detailed process of this algorithm. It works in a one-test-at-a-time fashion (repeatedly generate new test cases until a full coverage is achieved), in which the distribution of label values in current test cases is dynamically exploited to guide the generation of future test cases (hoping to cover as many new label value combinations as possible).

Specifically, at each iteration, the algorithm first generates a random value $c \in [1, k]$ to indicate the number of labels that will be assigned value one in the new row (Line 4; in this way, each row must be constraint-satisfying). Next, to determine which c labels are selected, the algorithm sorts the n labels by the occurrence of value one in the current array in ascending order, and relies on a random value $m \in [1, \frac{n}{2}]$ to split these labels into the $least_m$ set (i.e., the value one occurs the least frequently for these m labels) and the $others$ set. A total of $c_1 = min(m, \frac{c}{2})$ labels are then randomly selected from the $least_m$ set (if $m < \frac{c}{2}$, then all m labels are selected), while the remaining $c_2 = c - c_1$ labels are selected from the $others$ set. This helps to maintain a certain level of randomness, and avoid reaching a dead end when all m labels that occur the least have been covered by the previous rows (Lines 5–11). After this, a new row is generated by assigning value one to these selected c labels, and assigning value zero to the other labels (Lines 12–14). If this new row is not included in the array generated so far, the algorithm adds this new row into the array and updates coverage accordingly (Lines 15–18).

The above process will repeat until all τ-way label value combinations are covered. Note that some rows might be redundant due to the greedy process (i.e., all label value combinations covered by this row have been covered by the subsequent rows), the algorithm further checks each row and removes those redundancy to reduce array sizes (Lines 20–22).

Algorithm 1 The Adaptive Sampling Algorithm for τ-way Label Value Covering Array Generation

Input: Size of label space n, counting constraint variable k, and covering strength τ
Output: $LVCA(N; \tau, n, k)$

1: $coverage \leftarrow 0, A \leftarrow \{\}, labels \leftarrow [1, 2, \cdots, n]$
2: **while** $coverage < 1$ **do**
3: $row \leftarrow [0, 0, \cdots, 0]$ ▷ length is n
4: $c \leftarrow RandomInt(1, k)$ ▷ number of value one assigned
5: Sort $labels$ by the occurrence of value one in A in ascending order
6: $m \leftarrow RandomInt(1, \lceil \frac{n}{2} \rceil)$
7: $least_m \leftarrow labels[:m]$, $others \leftarrow labels[m:]$
8: $c_1 = min(m, \lceil \frac{c}{2} \rceil)$
9: $c_2 = c - c_1$
10: $selected \leftarrow RandomSample(least_m, c_1)$
11: $selected \leftarrow selected \cup RandomSample(others, c_2)$
12: **for** $i \in selected$ **do**
13: $row[i] \leftarrow 1$
14: **end for**
15: **if** $row \notin A$ **then**
16: $A \leftarrow A \cup \{row\}$
17: Update $coverage$ with A, τ
18: **end if**
19: **end while**
20: **for** $row \in A$ **do**
21: Delete row from A if coverage of $A \setminus \{row\}$ is 1
22: **end for**
23: **return** A

B. Concrete Test Images Generation

Once the τ-way label value covering array is generated, LV-CIT will then generate concrete test images for each

abstract test case in the array. To this end, LV-CIT relies on a previously created object library that includes a number of object images for each label of the label space to implement image composition. Here, to ensure the validity of the final images generated (i.e., all objects involved in the image can be easily recognised by humans [31]), the object images should be carefully selected to ensure that each object image must contain complete and prominent features of the corresponding label (at the same time, have no features that are irrelevant to this label). In addition, because the focus of LV-CIT is on the interactions between labels, there is also a need to ensure that the single object in each object image can be correctly classified by the image classifiers under test (the process of building the object library for evaluating LV-CIT will be explained in Section IV-B).

To generate concrete test images, LV-CIT will select object images from the object library and put these objects one after another into a canvas of white background. Specifically, LV-CIT first selects one object image at random for each label that takes value one in the test case. Then, LV-CIT resizes the object based on a random scaling ratio $sr \in [50\%, 150\%)$, and also determines a random location for the object. Here, to account for the scenarios that different objects could be overlapped in real world [36], LV-CIT uses a random overlapping ratio $or \in [0, 30\%)$ to determine the degree of overlapping between different objects. LV-CIT will repeat the above process M times for each abstract test case, so a total number of $M \times N$ test images will be produced for a label value covering array of size N.

For example, the middle column of Fig. 1 gives two candidate object images for labels *cat* and *dog*, in which the complete and prominent features for recognising these two labels are included. These two object images are then randomly resized and placed in a white canvas, which produces a concrete test image for the test case $\{cat = 1, dog = 1, person = 0, car = 0\}$.

Here, LV-CIT uses composite images, rather than realistic images, as the test inputs. This is because LV-CIT seeks to test τ-way label value combinations, and by the definition of label value covering array, every test image should contain an exact set of objects, which could be very hard to find in real world. Even images containing certain objects can be found (e.g., from Internet [19]), these images might contain other unexpected objects or backgrounds, which might unintentionally influence the classification behaviour of the image classifier on specific objects (e.g., the recognition of an object is based on features of the background or other objects, but not the object itself [37]). By contrast, the use of composite images could avoid introducing features that are irrelevant to the target labels into test images, so that the actual ability of image classifiers to handle particular label correlation could be better examined. Note that existing studies [36], [38] have also utilised similar unrealistic images (e.g., a pair of scissors "flying" over a bench) for triggering specific behaviours of DNN-based systems (e.g., as long as the features of objects are salient, they should be correctly recognised); the training set of modern DNN-based image classifiers also contain a number of composite images [39]–[41].

We also note that although the composite images might be intuitively unrealistic, they can be, to certain extent, representative of real-world test scenarios. Especially, some combinations of objects (and their placements) might seem to be rare, but they are possible to appear in the real world. For example, Fig. 5(a) shows a composite image that triggers classification errors in our experiment, in which a dining table is on the top of a car. A similar scenario can be found in a realistic image of a car smashing into a restaurant[3]. LV-CIT seeks to systematically examine all possible interactions between objects, which is essential for the safety and trustworthiness assurance of image classifiers.

C. Test Execution and Test Oracle Identification

After generating concrete test images, LV-CIT then inputs each of these images into the image classifier under test to obtain the classification output. In contrast to existing testing methods for DNNs [11]–[15], [19] that usually rely on metamorphic testing to identify test oracles, LV-CIT can use a more precise strategy that directly compares the classification outputs and the corresponding abstract test cases. Specifically, LV-CIT encodes the classification output of the classifier as an output vector, $[o_1, o_2, \ldots, o_n]$, where $o_i = 1$ indicates that label l_i is outputted by the classifiers, and $o_i = 0$ otherwise (for $1 \leq i \leq n$). Then, any mismatching between the output vector and its corresponding abstract test case (i.e., the input vector, $[v_1, v_2, \ldots, v_i]$) indicates an error.

IV. Experimental Setup

This section explains the experiment we conducted to evaluate the performance of LV-CIT. We have set up the following three research questions:

- $\mathbf{RQ_1}$: How efficient is LV-CIT in generating label value covering arrays?
- $\mathbf{RQ_2}$: How effective is LV-CIT in testing multi-label image classifiers?
- $\mathbf{RQ_3}$: What is the ability of current multi-label image classifiers to handle correlations between labels?

A. Image Classifiers Under Test

In this study, we selected six state-of-the-art DNN-based multi-label image classifiers that are developed based on the two popular datasets of multi-label image classification, *VOC* [32] and *COCO* [33], as our experimental subjects (three classifiers on each dataset). To avoid potential errors introduced during the training process, we utilized the pre-trained models released by their respective authors [42]–[44]. Table II shows details of these DNN models, including the size of label spaces (n), name of the DNN models, and their mean average precisions (*mAP*) scores.

[3]https://news4sanantonio.com/news/offbeat/gallery/watch-the-shocking-moment-a-car-slams-into-crowded-restaurant-washington-swish-hot-pot-facebook-bellevue-police?photo=3

TABLE II
THE SIZE OF LABEL SPACES (n), NAME OF DNN MODELS, MEAN AVERAGE PRECISIONS (mAP) ACHIEVED, NUMBER OF IMAGES IN THE OBJECT LIBRARIES (OL), AND NUMBER OF LABELS INVOLVED (LI) OF THE SUBJECT DNN MODELS IN THE EXPERIMENT

Datasets	n	**DNN Models**	mAP	OL	LI
VOC [32]	20	MSRN [45]	96%	580	20
		ML-GCN [22]	94%	576	20
		ASL [46]	94.6%	576	20
COCO [33]	80	MSRN [45]	83.4%	1,973	80
		ML-GCN [22]	83%	1,987	80
		ASL [46]	86.6%	1,811	79

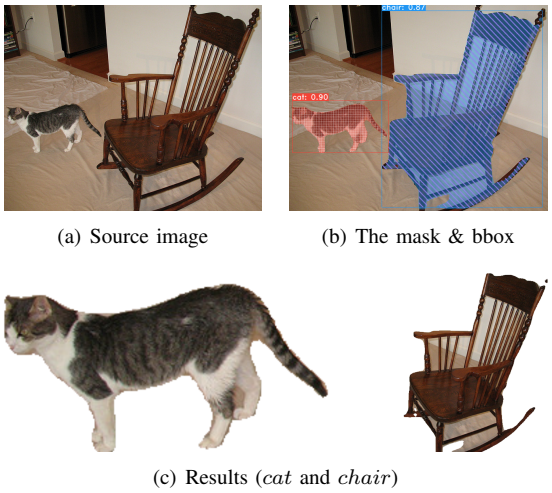
B. Object Library

The application of LV-CIT relies on a previously created object library (as mentioned in Section III-B). To this end, we first selected all 4,952 images from the testing set of *VOC* [32], and all 40,504 images from the validation set of *COCO* [33] as the set of source images. We note that the DNN models used in this experiment are pre-trained on the *ImageNet* dataset and fine-tuned on the training and validation sets of *VOC*, and training set of *COCO*. So, the images in the testing set of *VOC* and the validation set of *COCO* indicate images that are not involved in the training process of the subject image classifiers. In addition, as the annotations of images in the testing set of *COCO* are not provided, these images are also excluded.

Then, for each of these source images collected, we employed *YOLACT++* (*YOLACT++-550* model with *ResNet-50*) [47], a fast object detection and instance segmentation tool, to extract object images (as illustrated in Fig. 2). Specifically, we first applied *YOLACT++* to identify the object mask and bounding box (*bbox*) of each object included (Fig. 2(b)). Then, we removed the background outside the *bbox* and changed the remaining areas outside the mask to be transparent to obtain several object images that contain each single complete object (e.g., in Fig. 2(c), two object images are produced).

Next, to ensure that such objects can be correctly classified by the subject image classifiers (so that the errors are due to the interactions between labels), and also be easily recognised by humans (so that the test inputs are considered valid [31]), we further performed a two-step validation process. First, we input all object images into the six subject classifiers and removed object images that cannot be correctly classified. Then, we manually selected at most 30 object images for each label in the label space based on the following criteria: 1) the prominent features of the target object are complete and easily recognisable; 2) there is no other object in the image. The above selection process was performed by a undergraduate student in non-computer related majors to avoid knowledge-based biases, and the results were independently confirmed by two authors of this paper.

The last two columns of Table II give the numbers of object images in the object library of each subject classifier (note that we cannot always select 30 valid object images for each label), and the number of labels for which at least one object image is collected. We can see an exception case for *ASL* of *COCO*, where for the label *hair dryer*, the classifier fails to recognise all object images generated by *YOLACT++*. Accordingly, this label is exclude from the label space of *ASL* on *COCO*.



(a) Source image (b) The mask & bbox

(c) Results (cat and $chair$)

Fig. 2. The process of object image extraction.

C. Experiment Process

1) Label Value Covering Array Generation (RQ_1): The first research question concerns the efficiency of LV-CIT (specifically, the adaptive sampling algorithm introduced in Section III-A) in generating τ-way label value covering arrays. In this study, we focused on $\tau = 2$, because it not only is the most widely used choice in CIT applications [18], [34], [48], [49], but also indicates the practical scenarios where the modern image classifiers tend to learn the correlations between label pairs [21], [22]. While for the size of label space, we chose to use the sizes of label spaces of the two datasets used in this study (i.e., $n \in \{20, 80\}$) to indicate small and large label spaces, respectively. The counting constraint variable k (i.e., the number of ones cannot exceed k in each row of the array) might be an influencing factor of the performance of the generation algorithm, so we let $k \in [2, 6]$ to investigate the impact of k on generating the label value covering array.

Regarding the methods for comparison, we chose to compare LV-CIT with the *Baseline* method (as introduced in Section III-A) and the *ACTS* tool (version 3.2) [50]. *Baseline* indicates a straightforward solution to cover all label value combinations satisfying the counting constraint. While *ACTS* indicates a highly efficient and widely used tool in CIT studies [18], [49], [51], which implements a greedy algorithm for classical constrained covering array generation. According to the modelling language of *ACTS* [29], the counting constraint can be directly described as one constraint expression, $\sum_{i=1}^{n} v_i \leq k$, and we chose to have *ACTS* use a constraint solver to handle such a constraint. We note that although the more recently developed *CAgen* [30] tool can outperform *ACTS*, it only supports boolean, relational, and numerical operators (e.g., &&, =, >) but not the + operator for describing constraints. So this tool is excluded from this

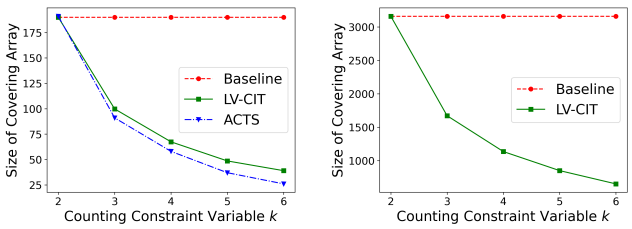
comparison. To account for the randomness involved in the above methods, we ran each of them five times for each problem instance. We also set the maximum execution time budget for each method as 72 hours.

2) Effectiveness of LV-CIT (RQ_2): The second research question seeks to investigate the effectiveness of LV-CIT in revealing errors. Here, according to the results of RQ_1, we set the counting constraint variable $k = 4$, because it tends to achieve a good balance between the testing cost and testing effectiveness. This also reflects the majority of images in the training set of *VOC* and *COCO* datasets, as 99.9% and 83.5% of those images are annotated with no more than four labels.

Regarding the methods for comparison, we chose to compare LV-CIT with the recently proposed *ATOM* method [19] and the *Random* method for testing image classifiers. The *ATOM* [19] method also seeks to examine the correlations between labels, but its focus is on *all-one* τ-way label value combinations only (i.e., all the τ labels take value one, which is referred to as *τ-label combinations* in this previous study). In particular, *ATOM* first generates the set of candidate τ-label combinations as the test targets, and then uses an image search engine to try to collect realistic test images for each of them. In this experiment, we directly used the test images provided by its authors, i.e., at most five images for each τ-label combination. To enable a fair comparison, we set $M = 10$ for LV-CIT so that the total number of images generated by LV-CIT is similar to that collected by *ATOM*. The *Random* method simply selects test images at random from the testing set of *VOC*, or the validation set of *COCO* (as these images are used to create the object library for LV-CIT). We let this method select the same number of images as used by LV-CIT (i.e., $10 \times N$ for a label covering array of size N) in this experiment.

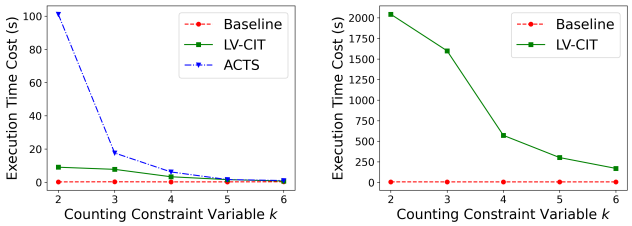
We note that there is a degree of randomness involved in the execution of both LV-CIT and *Random* methods. So we ran each of them five times for each problem instance and report the average results observed. While for *ATOM*, since we directly used the images provided by its authors, we ran this method only once for each problem instance.

3) Handling Correlations Between Multiple Labels (RQ_3): The third research question aims to evaluate the ability of current image classifiers to handle correlations between the τ labels (specifically, $\tau = 2$ in this study). To this end, we define the *τ-way Interaction Accuracy (IA)* of the classifier with respect to a single test image t as the ratio of the τ-way label value combinations that the classifier has correctly recognised to the τ-way label value combinations involved in t, that is, $\binom{n}{\tau}$. For example, given a test case, $\{cat = 1, dog = 1, person = 0, car = 0\}$, and its classification output, $\{cat = 1, dog = 0, person = 0, car = 0\}$, the 2-way interaction accuracy will be 0.5, because among all $\binom{4}{2} = 6$ 2-way label value combinations, it can correctly recognise three of them ($\{cat = 1, person = 0\}$, $\{cat = 1, car = 0\}$, and $\{person = 0, car = 0\}$). Then, for a set of test images, the *τ-way mean Interaction Accuracy* (*mIA*) is defined as the average accuracy value across all of these test images.



(a) $n = 20$ (b) $n = 80$

Fig. 3. Sizes of 2-way label value covering arrays generated by different generation methods.



(a) $n = 20$ (b) $n = 80$

Fig. 4. Execution time costs (seconds) of different generation methods.

In addition, based on the testing results, we further classified the errors that are relevant to each τ-way label value combination into three types: *Missing*, *Extra*, and *Mismatch*. Specifically, *Missing* indicates that the classifier cannot recognise all the objects when all the τ labels are present, e.g., incorrectly recognise $\{l_1 = 1, l_2 = 1\}$ to $\{l_1 = 1, l_2 = 0\}$ (denoted as $\{l_1 = 1, l_2 = 1\} \rightarrow \{l_1 = 1, l_2 = 0\}$); *Extra* indicates that the classifier outputs extra labels in the label combination when only some of the τ labels are present, e.g., $\{l_1 = 1, l_2 = 0\} \rightarrow \{l_1 = 1, l_2 = 1\}$; and *Mismatch* indicates that the classifier confused some labels in the τ-way label combination, e.g., $\{l_1 = 1, l_2 = 0\} \rightarrow \{l_1 = 0, l_2 = 1\}$.

The above experiments were performed under a computer equipped with an AMD Ryzen 7 5800X 8-core CPU@3.8GHz, a 32GB RAM, and an NVIDIA GeForce RTX 4060Ti GPU with 8GB VRAM.

V. Results

A. Results for RQ_1 (Efficiency on Label Value Covering Array Generation)

The first research question concerns the efficiency of the adaptive sampling algorithm used in LV-CIT for label value covering array generation. Fig. 3 and Fig. 4 show the sizes of 2-way label value covering arrays generated by different methods and their execution time costs observed.

From Fig. 3, we can see that the *Baseline* method always generates the largest 2-way label value covering arrays for all cases studied, and the array sizes keep the same for different values of k. This is because the *Baseline* method will simply generate $\binom{n}{\tau}$ test cases to cover all possible τ-way label value combinations as long as $\tau \leq k$ (see Section III-A). While for LV-CIT and *ACTS*, they can both generate smaller arrays than

Baseline, and the array sizes observed will rapidly decrease with the increase of k (e.g., the array sizes generated by LV-CIT are 190 and 39 for $k = 2$ and 6, respectively). This is because a larger value of k will allow each row in the array to be more flexible and could thus cover more label value combinations. Moreover, when $n = 20$, we can see that *ACTS* tends to perform better than LV-CIT in terms of array sizes, as it can generate up to 33% smaller arrays than LV-CIT for $k = 6$. However, when $n = 80$, *ACTS* always runs out of time for all choices of k (even with a 72 hours budget), and so there is not data point in Fig. 3(b).

Regarding the execution time cost, from Fig. 4, we can see that the *Baseline* method always runs the fastest (on average, 1.6 seconds among all cases studied). Although *ACTS* can generate smaller covering arrays than LV-CIT when $n = 20$, it tends to have a much higher execution cost (on average, 3.6 times higher than LV-CIT). Worse, for large label spaces ($n = 80$), *ACTS* is simply infeasible in generating any 2-way label value covering arrays under a 72 hours budget, while LV-CIT takes at most 2,043 seconds for these cases. This finding indicates that the adaptive sampling algorithm used in LV-CIT is much more efficient than the existing CIT tool, and can be a practical choice for generating label value covering arrays.

Note that although both array sizes and execution time cost observed decrease with the increase of k, it does not mean that a larger value of k will lead to a potentially better testing effectiveness for multi-label image classifiers. In particular, if the testing goal is to examine 2-way label value combinations (the focus of this study), using $k = 2$ might be an ideal choice, as all test images contain no more than two objects. However, this will result in a high testing cost (i.e., large array sizes). By contrast, despite that using $k = 6$ can help reduce testing cost, the test images might contain too many objects, which will then introduce interaction-irrelevant errors (accordingly, the classification ability might be underestimated). Hence, there is a need to balance the testing cost and effectiveness, and in this study, we recommend $k = 4$ when testing 2-way label value combinations (this setting will be used in the remaining two research questions).

Answer to $\mathbf{RQ_1}$: For small label spaces and relatively easy-to-solve counting constraints ($n = 20$), the *ACTS* tool can generate up to 33% smaller 2-way label value covering arrays than LV-CIT, but its average execution time cost is 3.6 times higher than LV-CIT. While for large label spaces and hard-to-solve counting constraints ($n = 80$), *ACTS* is unlikely to scale, and LV-CIT performs more efficiently.

B. Results for RQ_2 (Effectiveness on Testing Image Classifiers)

For the second research question, we investigate the effectiveness of LV-CIT in testing image classifiers. Table III gives the 2-way label value combination coverage (i.e., the proportion of unique 2-way label value combinations that are covered by the test inputs) that can be achieved by LV-CIT and the *Random* method (using the annotations of the test images as ground truth for *Random*). Note that we do not report such coverage of *ATOM* [19] here, because this method collects realistic images from Internet as test images, in which unexpected objects could be included (e.g., the test case specifies the coverage of $\{l_1 = dog, l_2 = cat\}$, but the test image might also contain a *person*, and so $\{l_1 = dog, l_3 = person\}$ is also covered). Thus, we cannot precisely calculate the label value combination coverages for test images of *ATOM*, unless all images are manually annotated.

TABLE III
THE AVERAGE 2-WAY LABEL VALUE COMBINATION COVERAGE OF THE LV-CIT AND *Random* METHODS

Datasets	DNN Models	Coverage	
		LV-CIT	Random
VOC	MSRN	100%	82.34%
	ML-GCN	100%	
	ASL	100%	
COCO	MSRN	100%	90.07%
	ML-GCN	100%	
	ASL	98.75%	

From Table III, as LV-CIT uses a label value covering array for designing test inputs, it can always achieve 100% coverage for *VOC*, which is much higher than that of the *Random* method (82.34%). For *COCO*, the average coverage achieved by LV-CIT is slightly lower than 100% (actually, 99.58%) due to the lack of images in object library for the label $hair\ dryer$ on the DNN model *ASL*. But the coverage of LV-CIT is still higher than that of *Random* (90.07%). This suggests that in terms of label value combination, LV-CIT can generate more systematic test inputs than the *Random* method.

Table IV further gives the number of test images generated, number of errors revealed, and the proportion of error-revealing images of the LV-CIT, *Random*, and *ATOM* methods. Note that both the LV-CIT and *Random* methods use the direct comparison between classification outputs and test cases to identify test oracle, so for each test image, any mismatching observed indicates an error. While for the *ATOM* method, it relies on seven metamorphic relations to generate follow-up images for each test image to identify test oracle. As such, any violation of these metamorphic relations is considered as an error (note that this leads to a higher cost than LV-CIT, as the image classifier will be invoked seven more times).

From Table IV, we can see that LV-CIT greatly outperforms the other two methods in terms of error revelation. Specifically, LV-CIT can detect, on average, 3.0 and 0.7 times more errors than the *Random* method for *VOC* and *COCO*, respectively (with the same number of test images). As the images involved in the testing set of *VOC* are mostly annotated with one label (64%) but those of *COCO* are mostly annotated with multiple labels (79%), images randomly selected from *VOC* tend to be easier for classifiers to recognise. Regarding the *ATOM* method, we can see that although LV-CIT generates,

TABLE IV
THE NUMBERS OF TEST IMAGES GENERATED, NUMBER OF ERRORS REVEALED, AND PROPORTION OF ERROR-REVEALING IMAGES OF THE LV-CIT, *Random*, AND *ATOM* METHODS

Datasets	DNN Models	Number of Images			Number of Errors			Proportion of Error-Revealing Images		
		LV-CIT	Random	ATOM	LV-CIT	Random	ATOM	LV-CIT	Random	ATOM
VOC	MSRN	674	674	898	603.6	160.8	304	89.6%	23.85%	33.85%
	ML-GCN	674			624.2	169.8	302	92.64%	25.22%	33.63%
	ASL	674			605.4	132	193	89.84%	19.58%	21.49%
COCO	MSRN	11,358	11,358	13,297	10,368.8	6,255.8	5,984	91.3%	55.08%	45%
	ML-GCN	11,358			10,387	6,291.2	6,484	91.46%	55.39%	48.76%
	ASL	11,352			9,915	5,645.2	4,584	87.35%	49.71%	34.47%

TABLE V
THE MEAN AVERAGE PRECISION (MAP) AND 2-WAY MEAN INTERACTION ACCURACY (MIA) OF THE SUBJECT DNN MODELS

Datasets	DNN Models	*mAP*	2-way *mIA*
VOC	MSRN	96%	81.79%
	ML-GCN	94%	80.28%
	ASL	94.6%	84%
COCO	MSRN	83.4%	95.51%
	ML-GCN	83%	95.53%
	ASL	86.6%	96.01%

(a) *Missing* (b) *Extra* (c) *Mismatch*

Fig. 5. The examples of three types of errors. (a) $\{dining\ table = 1, train = 1\} \rightarrow \{dining\ table = 0, train = 1\}$; (b) $\{keyboard = 1, mouse = 0\} \rightarrow \{keyboard = 1, mouse = 1\}$; (c) $\{bus = 1, car = 0\} \rightarrow \{bus = 0, car = 1\}$.

on average, 20% fewer test images than *ATOM*, its average proportion of error-revealing images achieved, 90.37%, is much higher than that of the *ATOM* method, 36.2%. This findings suggests that LV-CIT is a more cost-effective method in testing multi-label image classifiers.

Answer to RQ_2: LV-CIT is a cost-effective method for testing multi-label image classifiers, as it can reveal 185% more errors than *Random* on average (with the same number of images). LV-CIT also outperforms the existing method *ATOM*, as it can, on average, reveal 111% more errors with 20% fewer images.

C. Results for RQ_3 (Handling Correlations Between Labels)

The third research question concerns the ability of image classifiers to handle correlations between labels. Table V shows the 2-way mean interaction accuracy (*mIA*) of the six image classifiers calculated based on the testing results of LV-CIT. From this table, we can find that classifiers trained on *VOC* usually have a high *mAP* (at least 94%), but the 2-way *mIA* achieved is relatively low (on average, 82.02%). While for classifiers trained on *COCO*, a different trend is observed (the 2-way *IA* is higher than *mAP*). This is because *mAP* is used to measure the comprehensive ability of the classifiers to recognise each single label, while *mIA* focuses on the ability of the classifiers to handle interactions between labels (note that all objects in the test images of LV-CIT can be individually recognised by the classifiers). As for the *VOC* dataset, the label space is relatively small and there are 62% of images in the training set annotated with single label, so the classifiers trained on *VOC* are likely to achieve a high *mAP* but a low 2-way *mIA*. By contrast, the classifiers trained on *COCO* tend to have a lower *mAP* because of the large label space. Meanwhile, there are 80% of images in the training set of *COCO* annotated with multiple labels, so the classifiers trained on *COCO* are more likely to achieve a high 2-way *mIA*.

Table VI further summarises the top three 2-way label combinations that cause the most errors across the three error types (*Missing*, *Extra*, and *Mismatch*), as well as the number of training images related to these label combinations for the *ASL* model on the *VOC* and *COCO* datasets (the best model in terms of *mIA*)[4]. We also provide an illustration of the three types of errors in Fig. 5. It is known that a model might over-fit (or under-fit) the correlations between labels if there are too many (or too few) relevant images. However, it is not straightforward to identify the concrete label combinations which models over-fit (or under-fit), especially in a black-box manner (i.e., without the access to the training data). From Table VI, we can find that the 2-way label combinations that cause the most *Missing* errors generally have a few occurrence in the training set. Especially for the *VOC* dataset, there is no training image containing the top three label combinations. By contrast, the top three 2-way label combinations that cause the most *Extra* errors occur more frequently in the training set, especially for the label combination $\{bus, person\}$ (2,149 times). This finding suggests that the testing results of *Missing* and *Extra* can provide hints for understanding the potentially biased distribution of the dataset.

While for the *Mismatch* errors, there seems no clear relationship between the number of training images and the label com-

[4]Due to the space limitation, the results of other models are provided in the accompanying website of this paper: https://github.com/GIST-NJU/LV-CIT.

TABLE VI
THE NUMBER OF ERRORS-REVEALING IMAGES (E), AND NUMBER OF RELATED TRAINING IMAGES (TI) OF TOP THREE 2-WAY LABEL COMBINATIONS (LC) THAT CAUSE THE MOST ERRORS ACROSS THE THREE TYPES OF ERRORS BY THE ALS IMAGE CLASSIFIER

Datasets	Missing			Extra			Mismatch		
	LC	E	TI	LC	E	TI	LC	E	TI
VOC	{chair, cow}	28.8	0	{car, person}	3.6	215	{bus, car}	8	63
	{bus, chair}	22	0	{horse, person}	1.8	221	{car, chair}	2.2	5
	{dining table, train}	21.8	0	{motorbike, person}	1.8	169	{dining table, person}	2	94
COCO	{donut, surfboard}	32	1	{bus, person}	26	2,149	{knife, skis}	34	1
	{kite, skateboard}	30	2	{baseball glove, sports ball}	23.4	826	{cake, dining table}	27	1,234
	{couch, spoon}	28.8	74	{keyboard, mouse}	22.8	878	{cow, dog}	26.6	49

binations that cause the most errors. We conjecture that this type of errors might be caused by different reasons. For example, the *Mismatch* error of $\{bus, car\}$ might be caused by the similar features of the objects; while for $\{cake, dining\ table\}$, it might be due to the failure of recognising certain objects when it appears with any other object.

Answer to $\mathbf{RQ_3}$: The image classifiers that achieve higher *mAP* values might not necessarily handle the correlations between labels better. In addition, LV-CIT can effectively identify label combinations that cause the most errors related to the biased distribution of the dataset.

VI. THREATS TO VALIDITY

As far as the internal threats to validity are concerned, the performance of the LV-CIT method might be influenced by the object library used for image composition, and different test images will be produced if different object images are used. To create an object library of high quality, in this study, we used images in the validation and testing sets of the *VOC* [32] and *COCO* [33] datasets as the source images, and used a popular instance segmentation tool, *YOLACT++* [52], to extract objects. We further performed automated and manual validation steps on these object images extracted, in order to ensure that every object image is not only easily recognised by humans (i.e., they can be valid test inputs [31]), but also correctly classified by image classifiers under test. In addition, the different scaling and overlapping ratios used in the image composition process might result in different test images. To reduce the potential threats of generating invalid test inputs, we used relatively small values for these ratios to limit the degree of changes on the object images. Nevertheless, we have made the object libraries created, and the source code of LV-CIT publicly available, so that others can check and reuse.

Regarding the external threats to validity, we have only evaluated the performance of the LV-CIT method under six DNN-based image classifiers trained on the *VOC* [32] and *COCO* [33] datasets. We note that these two datasets are popular for multi-label image classification tasks, and have been widely used in studies of DNN-based systems testing [19], [53]. The six DNN models used also indicate the state-of-the-art choices in the current literature [22], [45], [46].

VII. RELATED WORK

Recently, extensive studies [3], [4], [7], [13], [15]–[20], [54]–[58] have been proposed for testing DNN-based systems. Combinatorial Interaction Testing (CIT) is a potentially powerful testing method in practice, which has also been applied in DNN testing [7], [17]–[19], [56]–[58]. For white-box testing, DeepCT [7] first applied CIT to test DNNs, which focuses on interactions between neurons in each single layer. Later, Chen et al. [56] extended DeepCT to variable strength CIT to test interactions between neurons in two adjacent layers. In addition, the idea of CIT has also been leveraged for the design of DNN datasets for covering important environmental feature interactions [59], as well as the coverage analysis for the training and testing sets of the given dataset [58].

In addition to white-box testing, CIT has also been used in a black-box manner when testing DNN-based systems. Chandrasekaran et al. [18] applied CIT to explore combinations of image transformation operations, so that the seed images can be mutated in different ways. While Kuhn et al. [57] and Kampel et al. [60] proposed to use fault diagnosis methods of CIT to infer feature interactions that are most closely related to the decisions made by the DNN models.

While for the multi-label image classifier testing, recently, Hu et al. [19] proposed the *ATOM* method to test and evaluate the classification ability of image classifiers to handle labels that appear together in test images. Similar to *ATOM*, the LV-CIT method proposed in this study also aims to test and evaluate the classification ability of multi-label image classifiers. But, LV-CIT employs a more systematic strategy. Specifically, LV-CIT models each label as a binary input parameter, and takes all possible combinations between label values (i.e., all possible τ-way label value combinations) into account. By contrast, the test targets of *ATOM* only indicate a special case of label value combination, in which only the co-occurrence of specific labels is considered. As a result, the τ-way label value covering array used in LV-CIT can offer a more systematic coverage than *ATOM*, and could help find more types of errors (e.g., the *Extra* and *Mismatch* errors discussed in Section V-C).

Regarding the generation of test images, LV-CIT uses object libraries to composite images, ensuring that all generated test images match the specific test cases in the covering array. This differs from *ATOM* that uses image search engine, which might

produce test images that contain unexpected objects. Also note that the use of composite test images enables a straightforward test oracle identification in LV-CIT, which differs from the metamorphic testing based strategy in *ATOM*.

In addition, Tian et al. [38] have proposed to investigate whether the background of an image influences the behaviour of image classifiers. Their method seeks to corrupt the objects from source images and then inpaints the background by a generative model. As such, the primary focus of the above method is on the influence of object-irrelevant features on the image classifier (especially, features from the background). By contrast, LV-CIT focuses on the interactions between labels (i.e., all test images generated contain only object-relevant features), and seeks to trigger errors related to the correlations between certain labels.

VIII. Conclusion

In this paper, we propose to use combinatorial interaction testing to systematically test the ability of multi-label image classifiers to handle interactions between labels. A black-box testing method named LV-CIT is thus implemented and evaluated. Specifically, we propose to use the concept of τ-way label value combination to represent interactions between τ labels, and we develop a novel adaptive sampling algorithm to generate τ-way label value covering arrays to cover all such combinations. For each test case in the above covering array, we further propose to use object images that are randomly selected from a previously created object library to composite test images, so that these images contain only the features that are explicitly specified to be present in the test case. The experimental results on two popular datasets of multi-label image classification (with six state-of-the-art DNN-based classifiers) have demonstrated the efficiency and effectiveness of LV-CIT. The testing results of LV-CIT can also indicate the actual classification ability of the image classifiers to handle different label interactions.

In order to facilitate reproducibility and further researches, we have made the source code of LV-CIT, the object libraries created, and all experimental results publicly available online at https://github.com/GIST-NJU/LV-CIT. An archived artifact is also available at https://doi.org/10.5281/zenodo.13368486.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62072226 and 62102176), and the Natural Science Foundation of Jiangsu Province (No. BK20221439).

References

[1] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[2] G. Li, Z. Ji, X. Qu, R. Zhou, and D. Cao, "Cross-Domain object detection for autonomous driving: A stepwise domain adaptative YOLO approach," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 603–615, 2022.

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

[4] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu *et al.*, "DeepGauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, 2018, pp. 120–131.

[5] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks," *CoRR*, vol. abs/1803.04792, 2018.

[6] ——, "Structural test coverage criteria for deep neural networks," *ACM Transactions on Embedded Computing Systems*, vol. 18, no. 5s, pp. 1–23, 2019.

[7] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao, "DeepCT: Tomographic combinatorial testing for deep learning systems," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering*, 2019, pp. 614–618.

[8] J. Sekhon and C. Fleming, "Towards improved testing for deep learning," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results*, 2019, pp. 85–88.

[9] S. Gerasimou, H. F. Eniser, A. Sen, and A. Cakan, "Importance-Driven deep learning system testing," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 702–713.

[10] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering*, 2019, pp. 1039–1049.

[11] C. Murphy, G. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing," in *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, 2008, pp. 867–872.

[12] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 132–142.

[13] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of Deep-Neural-Network-Driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[14] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*, 2018, pp. 118–128.

[15] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See, "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*, 2019, pp. 146–157.

[16] J. Wang, H. Qiu, Y. Rong, H. Ye, Q. Li, Z. Li, and C. Zhang, "BET: black-box efficient testing for convolutional neural networks," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 164–175.

[17] T. Byun and S. Rayadurgam, "Manifold for machine learning assurance," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 2020, pp. 97–100.

[18] J. Chandrasekaran, Y. Lei, R. Kacker, and D. R. Kuhn, "A combinatorial approach to testing deep neural network-based autonomous driving systems," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2021, pp. 57–66.

[19] S. Hu, H. Wu, P. Wang, J. Chang, Y. Tu, X. Jiang, X. Niu, and C. Nie, "ATOM: Automated black-box testing of multi-label image classification systems," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2023, pp. 230–242.

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

[21] W. Liu, H. Wang, X. Shen, and I. W. Tsang, "The emerging trends of multi-label learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7955–7974, 2022.

[22] Z. Chen, X. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, 2019, pp. 5177–5186.

[23] B. Gao and H. Zhou, "Learning to discover multi-class attentional regions for multi-label image recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 5920–5932, 2021.

[24] F. Zhou, S. Huang, and Y. Xing, "Deep semantic dictionary learning for multi-label image classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 4, 2021, pp. 3572–3580.

[25] "Combinatorial test case generation," https://www.pairwise.org, 2021.

[26] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "An improved meta-heuristic search for constrained interaction testing," in *2009 1st International Symposium on Search Based Software Engineering*, 2009, pp. 13–22.

[27] A. Gargantini and P. Vavassori, "Efficient combinatorial test generation based on multivalued decision diagrams," in *Hardware and Software: Verification and Testing - 10th International Haifa Verification Conference*, vol. 8855, 2014, pp. 220–235.

[28] M. Leithner, A. Bombarda, M. Wagner, A. Gargantini, and D. E. Simos, "State of the cart: evaluating covering array generators at scale," *International Journal on Software Tools for Technology Transfer*, pp. 1–26, 2024.

[29] L. Yu, Y. Lei, R. N. Kacker, and D. R. Kuhn, "ACTS: A combinatorial test generation tool," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 2013, pp. 370–375.

[30] M. Wagner, K. Kleine, D. E. Simos, R. Kuhn, and R. Kacker, "Cagen: A fast combinatorial test generation tool with support for constraints and higher-index arrays," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2020, pp. 191–200.

[31] V. Riccio and P. Tonella, "When and why test generators for deep learning produce invalid inputs: an empirical study," in *2023 IEEE/ACM 45th International Conference on Software Engineering*, 2023, pp. 1161–1173.

[32] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[33] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014, pp. 740–755.

[34] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, pp. 1–29, 2011.

[35] C. Luo, J. Lin, S. Cai, X. Chen, B. He, B. Qiao, P. Zhao, Q. Lin, H. Zhang, W. Wu *et al.*, "AutoCCAG: An automated approach to constrained covering array generation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering*, 2021, pp. 201–212.

[36] B. Yu, Z. Zhong, X. Qin, J. Yao, Y. Wang, and P. He, "Automated testing of image captioning systems," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022, pp. 467–479.

[37] Z. Zhang, P. Wang, H. Guo, Z. Wang, Y. Zhou, and Z. Huang, "DeepBackground: Metamorphic testing for Deep-Learning-Driven image recognition systems accompanied by background-relevance," *Information and Software Technology*, vol. 140, p. 106701, 2021.

[38] Y. Tian, S. Ma, M. Wen, Y. Liu, S.-C. Cheung, and X. Zhang, "To what extent do dnn-based image classification models make unreliable inferences?" *Empirical Software Engineering*, vol. 26, no. 5, p. 84, 2021.

[39] H. Zhang, J. Zhang, and P. Koniusz, "Few-Shot learning via saliency-guided hallucination of samples," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2770–2779.

[40] L. Song, Y. Xu, L. Zhang, B. Du, Q. Zhang, and X. Wang, "Learning from synthetic images via active pseudo-labeling," *IEEE Transactions on Image Processing*, vol. 29, pp. 6452–6465, 2020.

[41] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh, "Pre-training without natural images," in *Proceedings of the Asian Conference on Computer Vision*, vol. 12627, 2020, pp. 583–600.

[42] X. Qu, H. Che, J. Huang, L. Xu, and X. Zheng, "MSRN," https://github.com/chehao2628/MSRN, 2023.

[43] Z. Chen, X. Wei, P. Wang, and Y. Guo, "ML-GCN," https://github.com/Megvii-Nanjing/ML-GCN, 2019.

[44] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "ASL," https://github.com/Alibaba-MIIL/ASL, 2021.

[45] X. Qu, H. Che, J. Huang, L. Xu, and X. Zheng, "Multi-layered semantic representation network for multi-label image classification," *International Journal of Machine Learning and Cybernetics*, pp. 1–9, 2023.

[46] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "Asymmetric loss for multi-label classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 82–91.

[47] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT," https://github.com/dbolya/yolact, 2022.

[48] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE transactions on software engineering*, vol. 30, no. 6, pp. 418–421, 2004.

[49] B. Garn, D. Schreiber, D. E. Simos, R. Kuhn, J. Voas, and R. Kacker, "Combinatorial methods for testing internet of things smart home systems," *Software Testing, Verification and Reliability*, vol. 32, no. 2, p. e1805, 2022.

[50] "ACTS," https://csrc.nist.rip/groups/SNS/acts/, 2014.

[51] A. Bombarda, E. Crippa, and A. Gargantini, "An environment for benchmarking combinatorial test suite generators," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2021, pp. 48–56.

[52] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++ better Real-Time instance segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 1108–1121, 2022.

[53] K. Tong and Y. Wu, "Rethinking PASCAL-VOC and MS-COCO dataset for small object detection," *Journal of Visual Communication and Image Representation*, vol. 93, p. 103830, 2023.

[54] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-Guided black-box safety testing of deep neural networks," in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 10805, 2018, pp. 408–426.

[55] A. Odena, C. Olsson, D. Andersen, and I. Goodfellow, "TensorFuzz: Debugging neural networks with coverage-guided fuzzing," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 4901–4911.

[56] Y. Chen, Z. Wang, D. Wang, C. Fang, and Z. Chen, "Variable strength combinatorial testing for deep neural networks," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2019, pp. 281–284.

[57] D. R. Kuhn, R. N. Kacker, Y. Lei, and D. E. Simos, "Combinatorial methods for explainable AI," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2020, pp. 167–170.

[58] T. Cody, E. Lanus, D. D. Doyle, and L. Freeman, "Systematic training and testing for machine learning using combinatorial interaction testing," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2022, pp. 102–109.

[59] C. Gladisch, C. Heinzemann, M. Herrmann, and M. Woehrle, "Leveraging combinatorial testing for safety-critical computer vision datasets," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 1314–1321.

[60] L. Kampel, D. E. Simos, D. R. Kuhn, and R. N. Kacker, "An exploration of combinatorial testing-based approaches to fault localization for explainable AI," *Annals of Mathematics and Artificial Intelligence*, vol. 90, no. 7-9, pp. 951–964, 2022.