See the Forest, not Trees: Unveiling and Escaping the Pitfalls of Error-Triggering Inputs in Neural Network Testing

Yuanyuan Yuan
Hong Kong University of Science and Technology
Hong Kong, China
yyuanaq@cse.ust.hk

Shuai Wang
Hong Kong University of Science and Technology
Hong Kong, China
shuaiw@cse.ust.hk

Zhendong Su
ETH Zurich
Zurich, Switzerland
zhendong.su@inf.ethz.ch

Abstract

Recent efforts in deep neural network (DNN) testing commonly use error-triggering inputs (ETIs) to quantify DNN errors and to fine-tune the tested DNN for repairing. This study reveals the pitfalls of ETIs in DNN testing. Specifically, merely seeking for more ETIs "traps" the testing campaign into local plateaus, where similar ETIs are continuously generated using a few fixed input transformations. Similarly, fine-tuning the DNN with ETIs, while capable of fixing the exposed DNN mis-predictions, undermines the DNN's resilience towards certain input transformations. However, these ETI-induced pitfalls have been overlooked in previous research, due to the insufficient input transformations (usually < 10), and we show that the severity of such deceptive phenomena is enlarged when testing DNNs with more and diverse real-life input transformations.

This paper presents a comprehensive study on the pitfalls of ETIs in DNN testing. We first augment conventional DNN testing pipelines with a large set of input transformations; the correctness and validity of these new transformations are verified with large-scale human studies. Based on this, we show that launching an endless pursuit for ETIs cannot alleviate the "trapped testing" issue, and the undermined resilience pervasively occurs in many input transformations. Accordingly, we propose a novel and holistic viewpoint over DNN errors: instead of counting which input triggers a DNN mis-prediction, we record which input transformation can generate ETIs. The targeted input property of this transformation, termed erroneous property (EP), counts one DNN error and guides DNN testing (i.e., our new paradigm aims to find more EPs rather than ETIs). Evaluation shows that this EP-oriented testing paradigm significantly expands the explored DNN error space. Moreover, fine-tuning DNNs with EPs effectively improves their resilience towards different input transformations.

CCS Concepts

• **Software and its engineering** → *Software testing and debugging*.

Keywords

Deep learning testing

ACM Reference Format:
Yuanyuan Yuan, Shuai Wang, and Zhendong Su. 2024. See the Forest, not Trees: Unveiling and Escaping the Pitfalls of Error-Triggering Inputs in Neural Network Testing. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '24), September 16–20, 2024, Vienna, Austria.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3650212.3680385

ISSTA '24, September 16–20, 2024, Vienna, Austria

ACM ISBN 979-8-4007-0612-7/24/09
https://doi.org/10.1145/3650212.3680385

1 Introduction

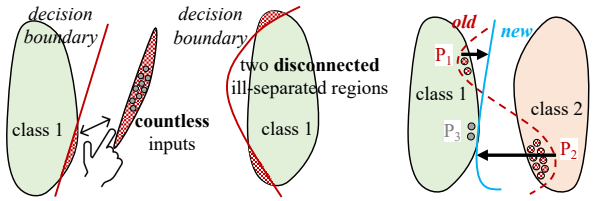
Deep neural networks (DNNs) have prosperous development in many applications, such as facial authentication and autonomous driving. Yet, similar to conventional software, DNNs are erroneous and can result in catastrophic accidents. In recent works, DNNs have been tested in various settings (e.g., autonomous driving [54], object detection [59], visual question answering [70], secure computation [41], etc.) and various testing objectives [29, 33, 38, 58, 68] and oracles [62, 67] have been designed. In general, the testing pipeline (often derived from metamorphic testing [8]) can be summarized as follows: given an input x and a transformation t (e.g., rotation), a DNN error[1] is detected if the DNN's prediction for a slightly mutated input $x' = t(x)$ changes. As the finality of the testing campaign, these ETIs are leveraged to fine-tune the tested DNN to repair the identified mis-predictions [59, 69]. Many ETIs are found by previous studies, and DNN loss can be reduced through fine-tuning. Nonetheless, this research revisits the above common practice by posing the following key question:

"*Is 'ETIs' an appropriate indicator in DNN testing?*"

In fact, DNNs are designed to perceive different semantical properties in their inputs and the internal logic of a DNN is reflected by its decision boundary in the input space. That is, DNN errors stem from the incorrect decision boundary, and the purpose of DNN testing is to discover and properly tune incorrect decision boundaries. We explain two primary issues in ETI-oriented DNN testing below; our research questions (Sec. 7 and Sec. 8) are designed to empirically illustrate and rectify them accordingly.

① **Countless Inputs:** In general, a DNN's decision boundary divides the entire input space as regions based on different properties of inputs (e.g., brightness, wheels in cars) [31, 44]. Accordingly, *a DNN's ETIs locate in its ill-separated regions.* Since inputs are continuous variables and deemed as "dots" in the whole space, an ill-separated region of even tiny size already includes innumerous

[1] This paper considers the "mis-prediction error" widely studied in existing DNN testing research; it subsumes common DNN defects such as robustness and fairness issues.



(a) Countless inputs exist in an ill-separated region. (b) Disconnected ill-separated regions. (c) ETIs are more easily to generate by mutating P_3.

Figure 1: Problems encountered when focusing on ETIs.

error-triggering inputs, as illustrated in Fig. 1(a). We thus question the common practice of assessing DNN errors based on the number of ETIs. Moreover, due to the non-linearity of DNNs (i.e., the decision boundary is *not* a straight line), an incorrect decision boundary can cause many *disconnected*, ill-separated regions, within which ETIs may likely have distinct properties (as shown in Fig. 1(b)). Hence, merely searching for more ETIs can trap the testing into certain regions, because ETIs, though similar, can be constantly generated within only a few ill-separated regions.

② **Mutual Exclusivity:** Conceptually, fine-tuning a DNN with EITs pushes its decision boundary away from these ETIs and towards the direction decided by ETIs' ground truth labels (Fig. 1(c); see Sec. 2.3 for formalization). Nevertheless, since the fine-tuning is guided by point-wise loss terms (i.e., computed over each input and then average), it may indeed impair a DNN due to mutually exclusive input properties [14, 47, 55], e.g., P_2 and P_3 lying in different sides of the decision boundary. As in Fig. 1(c), suppose more ETIs are detected by mutating inputs exhibiting property P_2 than those of P_1, e.g., due to stochasticity in testing. Thus, the fine-tuned decision boundary largely shifts to the left (from the red one to the blue one in Fig. 1(c)). Although the overall loss is reduced (as mutating P_2 generates more ETIs and the loss is dominated by them), mutating inputs exhibiting P_3 on the left side becomes more "fragile" and may likely trigger mis-predictions (i.e., inputs can move across the decision boundary with slight perturbations over P_3).

Following the above observations, we view ETIs as deceptive in DNN testing. We notice that such misleading testing is unaware in previous works, primarily due to the insufficient input transformations (usually < 10). To study these issues, we first augment existing works with a comprehensive set of input transformations. Given that pixel-based transformations (e.g., brightness, rotation) by design are limited, we focus on perception-based input transformations [7, 13, 30, 69] which are prevalent in real world. Existing perception-based transformations require manually annotating *transformable* perceptions [73] (e.g., eyes in a portrait); we therefore propose an automatic approach to discover transformable perceptions and generate ~8,000 new transformations. Note that a transformation does not equal to arbitrarily editing input properties. It should *consistently* and *uniquely* mutate the same targeted property across different inputs. We conduct a large-scale human evaluation to validate the consistency and uniqueness of our augmented input transformations (**Pilot RQ** in Sec. 5).

We evaluate 8 real-world DNNs (trained using 4 large-scale datasets) with 9 popular testing objectives and prioritization metrics. Our findings show that, even when employing more input transformations (which can trigger mis-predictions on the tested DNN), the DNN testing remains to focus on a few transformations that have generated ETIs at the initial testing stage. Moreover, setting an endless testing campaign hardly helps, as ETIs, though similar, can be constantly generated using a few already-explored input transformations (**RQ1** in Sec. 7). In addition, when fine-tuning DNNs with ETIs, we note that ~50% of input transformations target mutually exclusive properties, showing their pervasiveness. To further study impacts of the mutual exclusivity, we collect all ETIs generated during testing to fine-tune the tested DNNs. Surprisingly, all 8 "repaired" DNNs become more fragile to ~30% of the transformations. After investigating the ETIs, we find that only a few of them were generated from these "more fragile" input transformations when testing the original DNNs. As a result, during the fine-tuning, their contributions are shallowed by ETIs generated from other transformations (**RQ2** in Sec. 8).

"The Forest vs. Trees." With the above findings, this paper advocates a more holistic viewpoint of DNN errors: instead of counting which input triggers a DNN mis-prediction (i.e., ETIs), we record which input transformation generates ETIs. Accordingly, we deem the tested DNN as erroneous to the targeted input properties of such transformations, and count one erroneous property (EP) as one DNN error. For instance, if changing image brightness can trigger a DNN's mis-predictions, we count only *one* DNN error as "being erroneous to the brightness property", regardless of how many ETIs are generated by the brightness transformation. EPs, to some extent, reflect "clusters" of ETIs in a DNN's input space.

To address our unveiled issues, we propose simple yet effective EP-oriented regulations for DNN testing. When performing input mutations with different transformations, we explicitly force the testing to always use a fresh transformation that hasn't generated ETIs yet. During fine-tuning, we separately compute the loss term for ETIs generated using different transformations. Results show that, with our regulations, the testing can avoid being trapped and detect all transformations capable of triggering mis-predictions, within only modest testing epochs (Sec. 7.3). Also, our refined fine-tuning scheme can enhance the tested DNN's resilience towards all input transformations (Sec. 8.3). In sum, this paper makes the following contributions:

- We unveil that merely seeking for more ETIs misleads the DNN testing to constantly generate similar ETIs using a few fixed input transformations. Moreover, fine-tuning the tested DNN with ETIs makes it more fragile to certain input transformations, due to mutually exclusive input properties (which are pervasive) targeted by different transformations.
- We propose a technique that automatically generates a great number of different input transformations. Large-scale human evaluations validate the quality of these transformations. These transformations can boost future DNN testing works.
- We revisit existing ETI-oriented testing and advocate a new, EP-oriented scheme that focuses on input transformations triggering DNN mis-predictions. We design simple yet effective EP-oriented regulations, delivering more comprehensive and accurate DNN testing and fine-tuning.

Artifact. Our artifact is available at https://github.com/Yuanyuan-Yuan/EP-DNN-testing [2].

2 Preliminaries and Terminologies

Aligned to existing DNN testing works [15, 18, 22, 33, 42, 54, 59, 60, 62, 63, 67], we focus on image DNNs; also, image properties are easy to present and understand by even layman audiences. We discuss the applicability to other input formats (e.g., text) in Sec. 9. We now introduce preliminaries and terminologies of this study.

2.1 Input Transformations and EPs

Input transformations can be formally defined as in Def. 1.

Definition 1 (Input Transformation)**.** *An input transformation consistently targets one unique property (not targeted by other transformations) across different inputs. By applying a transformation with varied extents to an input, different mutated variants can be produced.*

For example, rotation transformation t_r consistently targets the rotation angle (i.e., one input property) of different images and "rotating 30 degrees" and "rotating 90 degrees" are two mutations achieved via the *same* transformation t_r. If applying a transformation to a DNN's inputs can trigger mis-predictions, the DNN is erroneous to the property targeted by the transformation.

Definition 2 (Erroneous Property (EP))**.** *If applying a transformation t to a DNN's input x can trigger mis-predictions (i.e., the DNN has different predictions for x and $t(x)$), the DNN is erroneous to the input property targeted by t. We view t's targeted property as one erroneous property (EP) of the DNN.*

For instance, if the rotation transformation t_r can trigger mis-predictions when applied to one of the DNN's inputs, the rotation angle property (regardless of the exact degree) is an EP of the DNN. Below, we introduce existing (image) transformations.

Pixel-Based. Early DNN testing works adopt transformations over image properties decided by pixel values [42, 54, 62], such as 1) adding noise, 2) blurring, 3) changing brightness, and 4) changing contrast. Other works also focus on image properties determined by the arrangement of pixels, including 5) translation, 6) reflection, 7) scaling, 8) rotation, and 9) shearing [54, 62].

Style-Based. Some recent works decompose an image as its content and style, and propose style-based transformation to change image styles. For example, [63, 68] apply the style of artistic paintings to real-life photos to test image classifiers. [54, 71] transfer the weather conditions between different driving scenes to test autonomous driving DNNs.

Perception-Based. The latest works propose different methods to edit perceptual contents (e.g., a face's age) in images. For example, [13] leverages generative models to achieve fine-grained manipulations over object orientation, motions, etc. [30] fuses different digits to confuse image classifiers. [69] implements gradual transitions of object status in images. Despite the effectiveness of perception-based manipulations in DNN testing, they are often arbitrary and input-specific. Several recent works have achieved perception-based transformations and ensure the consistency and uniqueness (see Def. 1) via optimization-based solutions [73, 74] (whose details will be introduced in Sec. 4). However, they require manually annotating transformable perceptions in images, limiting the number of available transformations.

2.2 DNN Testing Methods

Random Testing. This testing scheme treats a DNN as "black-box" and randomly generates mutated inputs to stress the DNN. For example, DeepRoad [71] uses GANs [75] to perform style transfer (e.g., from "sunny" to "snowy") on driving scene images and check if the auto-driving DNN behaves inconsistently.

Objective-Guided Testing. Recent works treat a DNN as "white-box", thus creating different testing objectives. The intuition is that, with objectives that characterize DNN activities, guided (and often more effective) mutations are performed, which can increase the chance of triggering DNN mis-predictions. Existing objectives can be roughly divided into the following categories.

Coverage: Similar to code coverage in traditional programs, DNN coverage metrics measure the number of activated neuron states during DNN execution [38, 42, 68]. Input mutations are guided to maximize the coverage value during DNN testing [38, 42, 68].

Adequacy: These metrics log a trace of neuron outputs in each DNN execution. Then, they compare this trace with all traces logged when the DNN is executing its training data and compute a similarity score s [33, 34]. Input mutations are guided to expand the range covered by s.

Input Prioritization. Unlike previous works where mutations are guided during runtime, methods are designed to prioritize mutated inputs that are already generated, such that mis-predictions can be more quickly triggered without testing all mutated inputs. The majority of them are implemented using diversity metrics [15, 63]: for an input x and its mutant x', the diversity of their corresponding intermediate outputs (in the tested DNN) is computed as the priority of x'. The intuition is that mutations leading to diverse DNN intermediate outputs are more likely to trigger mis-predictions.

2.3 DNN Fine-Tuning

Similar to traditional software testing, DNN testing also expects to repair the identified errors. Unlike traditional software whose logics are explicitly coded, such that developers can directly modify the code to fix errors, DNN logics are implicitly learned from data. Thus, the repairing is conducted by fine-tuning the DNN with error-triggering inputs (and their ground truth labels) to fix the identified mis-predictions.

The basis of fine-tuning is adapted from the adversarial/robust training [57]. For a DNN f_θ, the intuition behind fine-tuning (using ETIs) is to minimize the following loss:

$$\frac{1}{|D|}\sum_{(x,y)\in D}\left[\max_{x'\sim\mathcal{N}(x,\epsilon)} L(f_\theta(x'),y)\right], \tag{1}$$

where y is the ground truth label of x and $\mathcal{N}(x,\epsilon)$ denotes all mutants of x. D is the set of all (x,y) pairs (i.e., the seed corpus in DNN testing), and L is the loss function based on the task of f_θ (e.g., L is cross entropy if f_θ performs classification). This objective minimizes the maximal loss. Intuitively, it guides the fine-tuning to do two things: 1) putting the ETI x' onto the correct side of the decision boundary (as indicated by the ground truth label y) and 2) pushing the decision boundaries to remain distant to x (as determined by ϵ) so that perturbing x hardly triggers mis-predictions.

Existing works ease the gradient computation (during fine-tuning) of the max operation in Eq. 1 with Danskin's theorem [10, 56]: for an inner function involving maximization, its gradient equals the gradient computed on the maximal. Hence, minimizing the objective in Eq. 1 is implemented as follows for every (x, y).

$$\arg\min_{\theta} L(f_\theta(x'), y), \quad f_\theta(x').label \neq y \tag{2}$$

Overall, this procedure first finds a mutant x' that flips the label y (which is among mutants having the largest loss) and then forces the DNN f_θ to predict y for the mutant x', i.e., fine-tuning the DNN with all detected ETIs. This approach is commonly adopted in DNN testing works [33, 40, 59, 68, 69].

3 Research Overview

Research Motivations and Deliverables. This study unveils the pitfalls of ETIs in DNN testing, i.e., ETIs trap the testing and impair the DNN during fine-tuning, and proposes solutions for the unveiled issues. Our study is conducted by exploring the following two research questions (RQs):

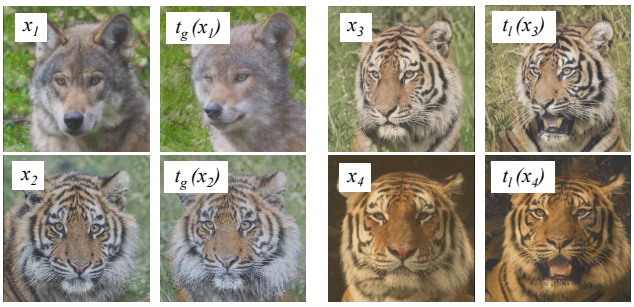
RQ1 (Sec. 7): How the testing is trapped into local error space if it focuses on ETIs? Does endlessly searching for more ETIs help to expand the explored error space?

RQ2 (Sec. 8): How often do mutually exclusive properties exist in different transformations? How does mutual exclusivity affect DNN fine-tuning?

To alleviate our unveiled issues, we accordingly propose simple yet highly effective, EP-oriented regulations to expand the discovered error space (see Sec. 7.3) and enhance the fine-tuning (Sec. 8.3).

Technical Challenges and Solutions. As introduced in Sec. 2.1, transformations designed in prior works are limited, exploring only negligible error space and hiding severe testing issues. We therefore require an automated approach to generate a comprehensive set of input transformations, such that we can assess 1) how ETIs trap the testing on a small and fixed set of transformations, and 2) the pervasiveness of mutually exclusive properties in diverse real-life transformations and how they affect the fine-tuning.

We turn our focus into perception-based transformations. First, they are closer to real-world transformations. Second, they manifest high potential to be largely augmented. For example, merely an animal's face encodes a rich set of perceptual properties, such as the global face orientation, local eye gaze, mouth status, etc. Examples of our augmented perception-based transformations are shown in Fig. 2. However, we do not indicate that perception-based transformations are better; all different types of transformations are important and complement each other in DNN testing and fine-tuning, as they all represent common variations in DNN inputs. Hence, in our studies in Sec. 7 and Sec. 8, we employ both perception-based transformations and existing transformations. In Sec. 5, we conduct a large-scale human evaluation to assess the correctness (*consistency* and *uniqueness*) of these perception-based transformations. With comprehensive transformations on hand, we select transformations capable of triggering DNN mis-predictions and finally use ~8,000 transformations in our RQs.



(a) Global t_g: turn left (b) Local t_l: open mouth

Figure 2: Sample global and local perception-based transformations. Both satisfy consistency and uniqueness.

4 Generating Input Transformations

Following Sec. 3, this section introduces the basis of implementing perception-based transformations. We then elaborate on how to generate these transformations in a large scale by automatically identifying transformable perceptions.

4.1 Editing Perceptual Properties

Perception-based transformations are implemented based on generative models (e.g., GANs [20]). Given a generative model G and a latent vector z (which can be random or converted using a real image), editing the generated image $G(z)$ can be characterized using the Taylor expansion:

$$G(z+\epsilon) - G(z) = \boldsymbol{J}(z)\,\epsilon + O(\epsilon), \quad \boldsymbol{J}_{i,j}(z) = \frac{\partial G(z)_i}{\partial z_j} \tag{3}$$

where ϵ is an infinitesimal and $\boldsymbol{J}(z)$ is the Jacobian matrix of G over z. $\boldsymbol{J}_{i,j}(z)$ is the (i, j)-th entry of $\boldsymbol{J}(z)$. This expansion demonstrates that, when perturbing z, $\boldsymbol{J}(z)$ governs what perceptions will be changed in $x = G(z)$ (since $O(\epsilon)$ reaches zero much faster than z). Therefore, existing works decompose $\boldsymbol{J}(z)$ into *orthogonal* vectors $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots\}$ (e.g., via singular value decomposition (SVD) [19]), and each $\boldsymbol{v}_i \in \boldsymbol{V}$ corresponds to a unique perception-based transformation t_i, i.e., $t_i(x) = G(z + \boldsymbol{v}_i)$, and the norm of $\boldsymbol{v}_i$ decides the extent of the mutation achieved by t_i.

Local-Perception Transformation. The above decomposition mostly enables global-level transformation for the entire image. For example, when modifying a human portrait, most transformations target global perceptions such as orientation, gender, and age, leaving the rich local perceptions (e.g., whether eyes are open or closed) untouched. Let $\mathbb{F}$ and $\mathbb{B}$ denote pixels in a specified local region and the remaining regions, respectively. A local transformation requires only editing perceptions in $\mathbb{F}$ and keeping perceptions in $\mathbb{B}$ unchanged. Suppose $\boldsymbol{v}$ is one vector obtained using $\mathbb{F}$'s Jacobian matrix to edit $\mathbb{F}$, Zhu et al. [73] further leverages $\mathbb{B}$'s Jacobian matrix to obtain $\boldsymbol{B}$, a set of directions towards which modifying z does *not* change $\mathbb{B}$. Thus, a local transformation can be achieved by first projecting $\boldsymbol{v}$ onto $\boldsymbol{B}$ before editing $\mathbb{F}$:

$$x' = G(z + \boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{v}). \tag{4}$$

We refer interested readers to [73] for implementation details of local-perception transformations.

4.2 Identifying Transformable Perceptions

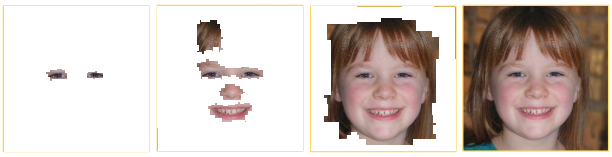
The local-perception transformations introduced in Sec. 4.1 requires manually selecting transformable $\mathbb{F}$; it may fail without a well-selected $\mathbb{F}$ (e.g., only half an eye). To automatically generate local-perception transformations, this paper therefore proposes an automated approach to identify transformable local perceptions.

As noted in Eq. 3, entries in the i-th row of $\boldsymbol{J}(z)$, i.e., $\boldsymbol{J}_{(i,\cdot)}(z)$, decide how the i-th pixel in $x = G(z)$ is changed when modifying z. Intuitively, when perceptions vary in different real images, pixels belonging to the same transformable perception should change similarly. That is, we can group pixels as regions based on $\boldsymbol{J}_{(i,\cdot)}(z)$: for a pixel i and one of its neighbors i', if $\boldsymbol{J}_{(i,\cdot)}(z)$ is close to $\boldsymbol{J}_{(i',\cdot)}(z)$, we then group them together. Otherwise, they will be considered separately. This way, we can have a set of disconnected regions, which will be later used for generating local-perception transformations. Here, a transformation only targets one region.

For each pixel i in $x = G(z)$, we first average all entries in the i-th row of $\boldsymbol{J}(z)$ as

$$\boldsymbol{J}_i(z) = \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{J}_{(i,j)}(z), \tag{5}$$

where n is the dimensionality of z. We apply image erosion and dilation to the "image" composed of all $\boldsymbol{J}_i(z)$. Note that the total number of different $\boldsymbol{J}_i(z)$ equals #pixels, and each $\boldsymbol{J}_i(z)$ corresponds to the i-th pixel in the generated image. The erosion operation erodes away trivial regions of a few pixels (which are likely noise), and dilation merges fragments that stay close to form larger regions. These operations are performed by sliding a kernel across an image, and the kernel size affects the identified regions. In general, larger regions will be marked if a relatively larger kernel is used.



(a) kernel size 2 (b) kernel size 4 (c) kernel size 8 (d) kernel size 16

Figure 3: Regions generated with different kernel sizes.

To comprehensively identify all feasible local perceptual regions, we iteratively apply erosion and dilation operations multiple times with gradually increased kernel sizes, and we record all newly generated local regions (either by joining pixels or merging existing regions yielded from prior iterations). As in Fig. 3, increasing the kernel size helps to generate regions for higher-level perceptions. With a smaller kernel in Fig. 3(b), eyes, nose, and mouth are localized in this face photo. The whole face is marked in Fig. 3(c) when increasing the kernel size, and finally the whole image is marked to generate global perception-based transformations.

Diversifying and Deduplicating Transformations. When generating transformations, we use different images to prepare the latent vector z in Eq. 3, so that the generated transformations (with different transformable perceptions) can be diversified. For example, the mutation "turn right" may be more likely spotted in a left-oriented face photo than in a right-oriented one. Nevertheless, it is possible that duplicated transformations are generated. Since each transformation is represented using a vector, we use the cosine similarity to identify repeated transformations. The cosine similarity ranges within $[-1, 1]$ and two vectors are the same if their cosine similarity equals to 1. In practice, cosine similarity of > 0.8 often indicates high similarity. To safely rule out similar transformations, we remove a generated transformation if its cosine similarity with any other transformation is > 0.5.

5 Pilot Study

We first launch a pilot study to evaluate the correctness of our generated transformations. We aim to answer the **Pilot RQ**: Do the generated transformations satisfy the consistency and uniqueness?

Consistency and uniqueness are evaluated by comparing the equivalence between transformations. Nevertheless, transformation "equivalence" does not imply they are the same, as the same transformation can achieve mutations of different extents. E.g., "rotating 30°" and "rotating 90°" are two equivalent transformations (i.e., they are consistent), whereas rotation and translation are two inequivalent transformations (i.e., each targets a unique property).

Since it is generally impractical to automatically compare changes of perceptions, we conduct human evaluations on the Amazon Mechanical Turk platform [1], designed as follows:

① We first randomly select 750 transformations t_i and construct 750 pairs of same transformations $\langle t_i, t_i\rangle$ (for consistency) and 750 pairs of different transformations $\langle t_i, t_j\rangle$ where $i \neq j$ (for uniqueness). These pairs are then mixed as a set T.

② For each pair of transformations $\langle t_i, t_j\rangle$ from T (where i may equal j, i.e., they are the same transformation), we randomly construct 2 pairs of images, $\langle x_a, x_a\rangle$ $\langle x_a, x_b\rangle$, where $a \neq b$, and form two questions: the first one compares two transformations on the same image: $x_a \rightarrow t_i(x_a)$ and $x_b \rightarrow t_j(x_a)$, whereas the other one compares two transformations on different images: $x_a \rightarrow t_i(x_a)$ and $x_b \rightarrow t_j(x_b)$. The extent of a mutation is also randomly decided. This way, we ensure that transformation pairs in T are compared on different images with different extents of mutations to reduce the bias. To avoid ambiguity, the participant is only allowed to answer "yes" or "no".

③ Finally, we have total $(750 + 750) \times 2 = 3,000$ questions. We then divide them into 6 groups and duplicate each group 3 times. Accordingly, we hire 18 senior Ph.D. students who have experience in DNN testing, and each Ph.D. student answers all 500 questions in one group. This way, each question is evaluated by three different participants to reduce personal bias.

④ We also form 20 sanity check questions for each participant. In some questions, two transformations are randomly selected from existing pixel-based transformations, and the ground truth answer is "yes"/"no" if two transformations are same/different. In others, the transformations, original images, and the extent of mutations, are exactly the same such that the ground truth answer is "yes". The participant is unaware of the sanity check questions and his/her answers will only be considered if he/she passes 90% of them.

Before the evaluation, we prepare a 30-minute warm-up (using 30 sample questions) for participants to understand the equivalence between transformations. To avoid fatigue, we do not set a time limit for each question, and participants can pause/resume the evaluation at any time.

Table 1: Datasets adopted in evaluations.

Dataset	#Images	Image Size	Remarks
ImageNet [11]	1,000,000	128 × 128	1000 classes of real images
Flickr Faces [32]	70,000	1024 × 1024	High-quality & fidelity face photos
CelebA [37]	200,000	128 × 128	40 facial attributes annotated
Animal Faces [9]	15,000	512 × 512	Wild animal faces
LSUN Car [66]	2,000,000	512 × 256	High-quality & fidelity car photos

Table 2: Evaluated DNNs.

DNN	#Layers/Modules	Task	Remarks
ResNet [24]	34	General classification	Non-sequential structure
VGG [50]	16	General classification	Sequential structure
Inception [51]	48	General classification	Feature extraction
MobileNet [27]	53	General classification	Mobile devices
DenseNet [28]	121	General classification	Extremely deep DNN
FaceNet [48]	7	Human face identification	Regression task
AlexNet [35]	8	Animal classification	One-class seed corpus
EfficientNet [52]	24	Car recognition	Fast inference speed

Results. We use the following two metrics to evaluate the results. Correctness: For two same transformations, it is correct in terms of consistency if ≥ 2 participants have answered "yes". For two different transformations, if ≥ 2 participants have answered "no", it satisfies the uniqueness. For consistency and uniqueness evaluations, the percentages of transformations satisfying the requirements are 0.94 and 0.95, respectively, demonstrating that our transformations are correctly implemented.

Agreement: We also evaluate the agreement among different participants using the kappa statistics [16]. The Kappa score ranges within $[0, 1]$ and a higher score indicates a better agreement. Our collected answers have 0.82 and 0.83 kappa scores for the consistency and uniqueness evaluations, indicating a strong agreement [49].

Validity of Mutated Images. We do not observe invalid images (i.e., an image is "unnatural" and hardly recognized by humans [12, 45]) generated by our transformations. In fact, the validity of these mutations are ensured by Jacobian matrix decomposition in a principled manner [6, 61, 72, 73]; see more samples in our artifact [2].

6 Evaluation Setup

Datasets. We use five real-world datasets shown in Table 1. These datasets are used for general image understanding (e.g., ImageNet) or domain-specific recognition tasks (e.g., LSUN Car). All of them are large-scale regarding the information in each image (e.g., the resolution), #images, and #classes. ImageNet has 1000 classes, we only select the first 20 classes due to efficiency consideration.

DNNs. Table 2 reports eight evaluated large-scale, real-world DNNs. These DNNs are well-trained (officially provided by Pytorch or the developers) and are representative in terms of the model structures (e.g., sequential vs. non-sequential), targeted platforms (e.g., mobile devices), tasks (classification vs. regression), optimization (e.g., MobileNet is optimized for size reduction, and EfficientNet is optimized for inference speed), etc. It's worth noting that, when testing AlexNet, we only use images of one class to construct the seed corpus. With comparison to other seed corpora of more classes, we can evaluate the impacts of the seed corpus's diversity.

Transformations. Table 3 lists transformations adopted in prior DNN testing research. For artistic-style-based transformations, we use style corpus from [17]. Though many styles are provided, we found most styles are similar. Thus, following our deduplication method (see Sec. 4.2), we also deduplicate the extracted styles (which are encoded as vectors) using a cosine similarity threshold of 0.5. A total of 31 different styles are obtained.

Table 3: Image transformation methods in prior works.

Pixel-based	1) Noise; 2) Blurring; 3) Brightness; 4) Contrast 5) Translation; 6) Reflection; 7) Scale; 8) Rotate; 9) Shear
Style-based	1) Rainy; 2) Foggy; 3) Snowy; 4) Cloudy & 31 different artistic styles (31 transformations)

Augmented Transformations. Due to the overhead and efficacy consideration, we use GANs to generate perception-based transformations. Our explorations show that diffusion models [64] are impractical to compute the Jacobian matrix in Eq. 3, and multi-modal LLMs (e.g., GPT-4) perform worse on the consistency and uniqueness of transformations; they also require manually specifying the text instruction for each transformation. We use BigGAN [6] trained with ImageNet to generate transformations for different classes in ImageNet. Following advice in [69], we generate perception-based transformations individually for images of different classes, as perceptions vary by class (e.g., the perception *wheel* does not exist in classes related to *animals*). For each class, we use 10 different images to prepare the latent vector z. StyleGANs [32] are adopted to generate transformations for the remaining domain-specific datasets and 100 images are used to prepare the latent vector z for each dataset.

In Sec. 7 and Sec. 8, our evaluations require transformations that can trigger DNN mis-predictions. For ImageNet DNNs, we generate 1944 perception-based transformations that can trigger mis-predictions on all of them. For other three DNNs trained on three different datasets, we also prepare 1944 perception-based transformations capable of triggering mis-predictions for each of them (to be comparable with the ImageNet setting). We find that all 44 existing transformations listed in Table 3 can trigger mis-predictions on all our evaluated DNNs. In sum, for each DNN/dataset, we prepare 1988 different transformations and each transformation is associated with one EP of its tested DNN.

Objectives. We consider a broad set of white-box and black-box testing objectives that are widely adopted in different scenarios. Different random testing schemes are also evaluated (see Sec. 7.1.1). We introduce these objectives below.

Coverage: Three representative DNN coverage criteria are considered. Neuron Coverage (NC) [42] and Top-K Neuron Coverage (TKNC) [38] are two structure coverage metrics. NC treats one neuron as the coverage unit, whereas TKNC focuses on DNN layers. NeuraL Coverage (NLC) [68] is one recently proposed distribution-level coverage, which measures the coverage based on the distribution of a DNN's intermediate outputs.

Adequacy: We consider Likelihood Surprise Coverage (LSC) [33] in our evaluation due to its low computation and space overhead.

Entropy: The entropy is adopted to guide black-box testing [21, 69], i.e., performing mutations to maximize the entropy. The entropy metric quantifies the uncertainty in DNN's outputs. A higher output entropy indicates that the DNN is more likely confused with the input and may have mis-predictions.

Prioritization Metrics. We study two state-of-the-art prioritization metrics [63] that use Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences as the diversity metrics (see Sec. 2.2).

7 RQ1: Testing

This section first studies the correlation between the number of (#) employed transformations and #ETIs in Sec. 7.1. Recall as introduced in Sec. 6, all our transformations are capable of generating ETIs. Then, in Sec. 7.2, we investigate how the testing procedure is affected by using #ETIs as the testing guidance. In Sec. 7.3, we show how to alleviate the encountered problems.

7.1 RQ1.1: Correlation Study

7.1.1 Setup. For each tested DNN, we first prepare N different transformations that can trigger mis-predictions and record the #ETIs. In all setups w.r.t. the same dataset, we use the same seed corpus comprising 500 different images. N is gradually increased from 10 to 1000. All experiments are conducted three times to reduce randomness. We implement different testing pipelines for objectives/metrics of distinct application scopes.

Accumulation Objectives. We categorize the coverage, adequacy, and entropy objectives as accumulation objectives because they are adopted for guiding the direction of accumulating mutations (in case the mutation does not lead to a mis-prediction). For these objectives, we follow a standard fuzzing-like testing pipeline as in Alg. 1. We set *epoch* to 5000 and in each loop iteration, an image is selected from the seed corpus X according to its priority decided by *prio*. An image's *prio* is subtracted by 1 every time it is selected until 0 (i.e., this image will no longer be selected). The function $objective(x')$ returns *true* if x' increases the coverage/adequacy/entropy.

Algorithm 1: Testing pipeline for accumulation objectives.

```
1  Transformation Set: T;   Seed Corpus: X;   Tested DNN: f;
2  foreach x ∈ X do
3  |   x.prio ← 10;
4  for i ← 0 to epoch by 1 do
5  |   x ← sample_with_priority(X);  x.prio ← x.prio − 1;
6  |   t ← random_select(T);
7  |   x' ← t(x); // total 'epoch' mutations are performed.
8  |   if mis_prediction(x, x') then
   |   |   // record x' as one ETI.
9  |   else if objective(x') then
10 |   |   x'.prio ← 10;  X.add(x');
```

Prioritization Metrics. The prioritization metrics are proposed to prioritize mutated images such that mis-predictions can be triggered earlier. Therefore, to imitate the "selection" process (i.e., uncovering mis-predictions without executing all mutated images), we perform $epoch \times 2$ mutations (i.e., mutating each image x in the initial seed corpus 20 times to generate 20 different x') and compute the priority of each mutant x' using the corresponding diversity metric. Total 5000×2 mutants x' are sorted in descending order and *only the first* 5000 mutants x' are used to test DNNs.

Random Testing. We consider two versions of random testing w.r.t. the accumulation and prioritization settings. The first one, *ARand*, follows a similar pipeline as in Alg. 1, but the $objective(x')$ at line 9 always returns *true*. The second one, *PRand*, follows a similar setting as prioritization metrics (generates 5000×2 mutants x') but *randomly* selects 5000 mutants x' to test the DNNs.

Table 4: PCC values between #ETIs and N. Results of non-significant p-value (> 0.05) are marked in gray.

DNN	ARand	NC	TKNC	NLC	LSC	Entropy	PRand	KL	JS
ResNet	-0.96	-0.97	-0.98	-0.93	-0.95	-0.93	0.01	-0.02	0.12
VGG	-0.95	-0.97	-0.96	-0.96	-0.98	-0.97	-0.01	0.04	-0.02
Inception	-0.95	-0.94	-0.92	-0.86	-0.98	-0.84	0.03	-0.06	-0.06
MobileNet	-0.93	-0.97	-0.94	-0.96	-0.95	-0.95	0.03	-0.08	-0.04
DenseNet	-0.96	-0.96	-0.92	-0.88	-0.93	-0.81	0.04	-0.07	0.14
FaceNet	-0.73	-0.74	-0.74	-0.70	-0.75	-0.74	0.13	-0.09	0.05
AlexNet	-0.55	-0.73	-0.57	-0.12	-0.64	-0.66	0.13	0.00	0.09
EfficientNet	-0.67	-0.71	-0.70	-0.61	-0.68	-0.76	-0.24	-0.04	0.12

7.1.2 Study Methods. We perform both quantitative and qualitative studies for the correlation between #ETIs and N (i.e., #employed transformations). In the quantitative study, we compute the Pearson correlation coefficient (PCC) of #ETIs and N to quantify the correlation. We also calculate the p-value to measure the significance. The PCC value ranges within $[-1, 1]$: a positive PCC indicates that #ETIs increases with N, whereas #ETIs decreases with N if their PCC is negative. If the PCC value is within $[-0.2, 0.2]$ and has p-value > 0.05 [26, 53], we can safely conclude that #ETIs and N are *not* correlated. We also plot how #ETIs changes with N for qualitative studies.

7.1.3 Results. In each experiment, we confirm that each of the prepared transformations is selected at least once to perform mutations. Table 4 shows the PCC values when testing various DNNs with different objectives. Values of non-significant p-values (> 0.05) are marked in gray. For prioritization metrics (i.e., KL and JS) and *PRand*, all PCC values are close to zero with non-significant p-values, indicating that #ETI and N have no correlation. For accumulation objectives and *ARand*, the PCC values are all *negative* (most of them have significant p-values), which is surprising and counterintuitive since #ETI decreases with N. Fig. 4 plots how the #ETI changes with N; we discuss the results as follows.

Overview. As shown in Fig. 4, regardless of the tested DNN, the DNN's task, the seed corpus, or the testing objective/metric, the #ETIs does not increase with N in all settings. We note that the curves w.r.t. prioritization and accumulation paradigms have distinct trends, which we discuss below.

Prioritization Metrics. Although these metrics have a higher #ETI than accumulation objectives in most cases, we do not compare the absolute values of #ETI because when using prioritization metrics, twice as many mutations are performed to simulate the "selection" process (see Sec. 7.1.1). In general, the curves (e.g., Fig. 4(a)-(e)) are roughly horizontal lines.

Accumulation Objectives. When cross-comparing objectives, NLC and Entropy seem to be the most two effective ones for finding DNN errors, because when guided with them, the largest #ETIs is generated in Fig. 4(a)-(e) and Fig. 4(h). Interestingly, we find that the curves of accumulation objectives can be divided into two phases. ⓐ When only a few transformations are prepared (e.g., $N < 200$ for EfficientNet), #ETI *decreases* with N. ⓑ When the number of prepared transformations is large (e.g., $N > 300$), the #ETI is mostly

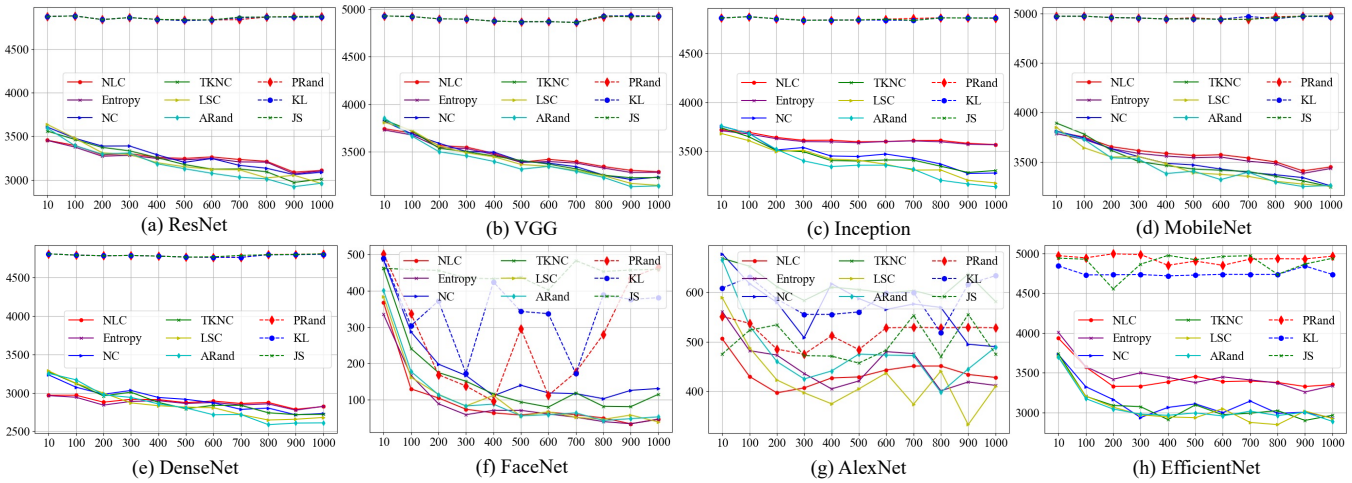


Figure 4: #ETIs (the vertical axis) w.r.t. the number of employed transformations N (the horizontal axis). All employed transformations are capable of triggering mis-predictions on the tested DNNs.

stable regardless of N. Thus, the negative PCC value is mainly induced by phase ⓐ. We discuss these two phases and infer the reasons below.

7.1.4 Analysis. The following conclusions can be derived.

The #ETIs is Not Positively Correlated With N. For both the prioritization objectives and the phase ⓑ of accumulation objectives, #ETIs is not correlated with N. That is, the number of employed transformations does *not* affect the #ETI. More specifically, even when always testing a DNN with a few transformations, many ETIs can still be triggered and their quantity is comparable to the #ETIs when using far more transformations (that are capable of generating ETIs) — despite the latter more comprehensively explores the DNN's error space. Accordingly, the #ETIs hardly reflects the comprehensiveness of testing. Even when a considerable amount of ETIs are triggered, developers may still be *unaware* that many transformations are capable of triggering DNN mis-predictions.

Recall that, as illustrated in Fig. 1, DNN errors stem from ill-separated regions in the input space which comprises continuous variables. Therefore, countless ETIs exist in any tiny-size, ill-separate input region, making the #ETIs less useful than anticipated and even misleading.

Accumulating Mutations is Misled in ETI-Oriented Testing. To understand the decreased #ETI in phase ⓐ of accumulation objectives, we inspect how accumulation objectives work. When a mutation does not trigger mis-prediction, it is accumulated if it enhances the chance of triggering mis-predictions in subsequent mutations (e.g., increases the coverage or entropy, which is decided by particular objectives). This way, an objective points an "accumulating direction" to increase the possibility of triggering mis-predictions, which also explains why more ETIs are generated under the guidance of these objectives than *ARand* (which randomly chooses mutants). Nevertheless, as more transformations are employed, such accumulations become less effective, because the accumulated mutations for one transformation may not be applicable to others. For example, accumulating mutations for the lighting transformation does not help to discover mis-prediction under the rotation transformation.

7.2 RQ1.2: "Trapped" Testing

Further to the less efficacy of accumulating mutations, we study how it affects the number of detected EPs in this section.

7.2.1 Setups & Study Methods. In this section, we focus on accumulation objectives. The setups are mostly the same as in Sec. 7.1. We equip each testing pipeline with all 1,988 transformations prepared, and study how the number of detected EPs changes with the testing epochs (i.e., the *epoch* in Alg. 1). Following Def. 2. *an EP is detected if mis-predictions are triggered by applying its corresponding transformation on DNN inputs.* It's worth mentioning that when performing prioritization-like testing, the #detected EPs is already decided before testing because no runtime mutation is performed. Thus, we do not evaluate these prioritization metrics in this section.

7.2.2 Results & Analysis. The curves of #detected EPs are drawn as *dashed lines* in Fig. 5. As noted, the increasing rate of #detected EPs drops when more testing epochs are performed. Even after 10,000 epochs, not all involved EPs are detected, and the #detected EPs is even *saturated* as the increasing rate tends to be zero. This is also induced by the accumulation of mutations. More specifically, once a few EPs are detected, the testing tends to focus on these EPs and repeatedly generate ETIs induced by these EPs. Knowledgeable readers may notice that this phenomenon is seemingly similar to the "coverage plateaus" studied in a recent work [36]. To clarify, this observation is different from [36], which states that #ETIs in software (Python) testing get saturated. Here, the DNN testing still constantly generates ETIs, but the #detected EPs is saturated. That is, the testing repeatedly uses explored transformations to generate ETIs, neglecting other transformations and their associated EPs.

Since the ill-separated regions of different EPs may be disconnected, once a mutation reaches an ill-separated region, future mutations will likely be "trapped" in this region due to two reasons.

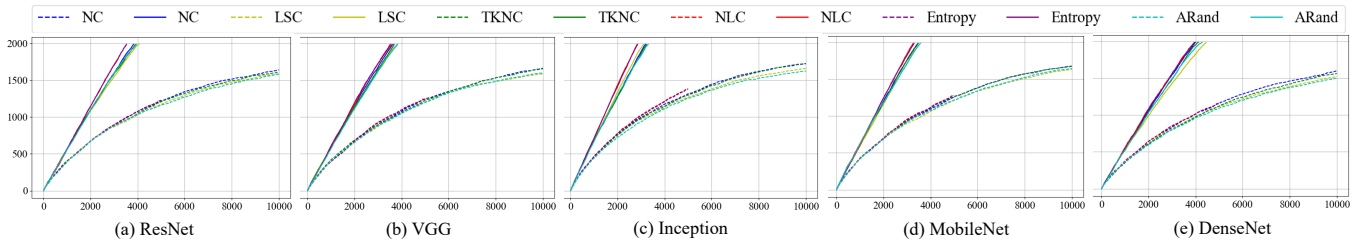


Figure 5: #Detected EPs (the vertical axis) w.r.t. #epochs (the horizontal axis). Results of previous and our testing methods are marked in dashed and solid lines, respectively. Horizontal axes in different figures are aligned.

1) Again, countless ETIs exist in an ill-separated region, and therefore, new ETIs can always be reported (though they reveal only one or a few EPs) since the testing aims to find more ETIs. 2) More importantly, *if a mutation moves outside an ill-separated region, it is unlikely to be accumulated* because such mutations may not increase the coverage/adequacy/entropy (i.e., get trapped). This phenomenon is very obvious for NLC and Entropy: the corresponding testing terminates easily between 4,000 and 6,000 epochs because all seeds are exhausted. Recall as in Sec. 7.1.3, NLC and Entropy are shown to be the most effective objectives to generate ETIs. Thus, it might not be inaccurate to interpret that the impacts of "trapped testing" is amplified when using more effective objectives, since moving outside an ill-separated is more likely to be rejected by them.

In sum, this observation highlights a major issue neglected in previous research: focusing on ETIs in (objective-guided) DNN testing may be misleading and overlook many EPs because the testing is "trapped" by already recognized EPs.

7.3 RQ1.3: Alleviating the Testing Problems

Solution. To mitigate the problems caused by ETIs, we propose a simple yet effective regulation strategy for DNN testing: whenever performing mutations, we always select transformations whose associated input properties have not been deemed erroneous, i.e., no ETI has been found by mutating these properties. Thus, the testing is forced to gradually find more EPs rather than ETIs. Once an EP has been detected, it will never be considered in further testing. This is reasonable since the root cause of mis-predictions is the incorrect decision boundary and EPs are associated with ill-separated regions in the input space: if a single ETI has been found by mutating an EP, other ETIs evidently exist in the same region and it is more meaningful to test other EPs instead.

Results. Our results are the *solid lines* in Fig. 5. With every objective, the #detected EPs is constantly growing without reduction in the growth rate. An interesting finding is that, when detecting only a few EPs, these solid and dashed lines are overlapped. This, to some extent, explains why *the "trapped testing" was previously not obvious due to the incomprehensive set of transformations*. Also, for NLC and Entropy which are largely undermined by the "trapped testing" (see Sec. 7.2.2), they remain the most effective objectives after regulating the testing with our strategy. Moreover, in most of the cases, the #epochs required to detect all EPs is less than twice of the #all EPs (i.e., an EP can be detected by applying its corresponding transformation less than two times on average), showing the efficacy and efficiency of our strategy.

8 RQ2: Fine-Tuning

Fine-tuning with ETIs is previously believed an effective approach to repairing the tested DNN's mis-predictions. Nevertheless, a counter-example is that, if two regions w.r.t. two EPs (corresponding to two transformations) reside on opposite sides of the decision boundary, enhancing the resilience towards one transformation will presumably make the DNN less resilient to another one. Therefore, these two EPs, and their corresponding transformations, are mutually exclusive. RQ2 studies how this mutual exclusivity affects the ETI-oriented fine-tuning; we use ETIs generated in Sec. 7.2.

8.1 RQ2.1: The Mutual Exclusivity

We first evaluate if the mutual exclusivity is common among input properties targeted by different transformations.

8.1.1 Setups & Study Methods. The mutual exclusivity is studied from two levels of granularity: 1) the types of transformations and 2) the transformations themselves. When the fine-tuning loss of one transformation decreases, we count the number of other transformations whose fine-tuning loss rises. We also report the distribution of these mutually exclusive transformations. Since many more perception-based transformations are generated, we randomly select 20 perception-based transformations, so that the number of transformations in each category is comparable.

8.1.2 Results & Analysis. We report that for each transformation, $(50 \pm 4)\%$ of the remaining transformations are mutually exclusive. For pixel-based transformations, their mutually exclusive transformations are mainly from other categories (i.e., style- and perception-based). In contrast, the mutually exclusive transformations of the other categories are evenly distributed among all categories (including their own categories). We note that our findings are consistent with observations in existing adversarial training works, which pointed out that enhancing a DNN's resilience towards one type of adversarial perturbations (i.e., one transformation) can make it more vulnerable to others (i.e., those mutually exclusive transformations/EPs) [14, 47, 55]. However, existing works are limited to only a few transformations. We, by identifying a comprehensive set of transformations, demonstrate that mutual exclusivity is pervasive among different transformations/EPs and has significant impacts on DNN fine-tuning.

8.2 RQ2.2: "Impaired" Resilience

This section studies how mutual exclusivity harms the DNN's resilience towards certain transformations.

8.2.1 Setup & Study Methods. We begin with the standard fine-tuning, which fine-tunes the DNN with all ETIs and is expected to reduce the overall loss. Then, we observe how each transformation's fine-tuning loss varies. To further assess the effects of increased fine-tuning loss (since increased loss does not necessarily indicate lower resilience; see Sec. 8.3), we characterize the resilience towards *one* transformation using random attack success rate (RASR), which is the success rate of generating ETIs with random mutations (since objective guided mutations have the countless inputs issue). That is, RASR is comparable to *PRand* (w/o prioritization) in Sec. 7.1.1 which is not affected by the accumulation. Recall that EPs are due to ill-separated regions. RASR, to some extent, is similar to estimating the area of ill-separated regions via Monte Carlo method [39].

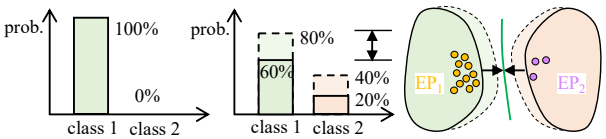
8.2.2 Results & Analysis. We find that in all evaluations, the overall fine-tuning loss decreases, which indicates the specious effectiveness of ETI-oriented fine-tuning. However, if we separately compute the fine-tuning loss for different transformations on the fine-tuned DNN, ~ 30% of them indeed increase largely. After checking their generated ETIs when testing the original DNN, we find that those transformations generate fewer ETIs (and hence contribute less to the overall loss) in comparison to the other transformations.

To clarify, the fine-tuning loss is computed in a point-wise manner, i.e., the overall loss is the average of loss values computed on all ETIs. Hence, if one transformation generates more ETIs, which may be due to the randomness during testing (as shown in **RQ1**, #ETI does not faithfully reflect the resilience), the overall loss will be more biased to this transformation. As a result, the fine-tuning procedure is "dominated" by those transformations generating more ETIs. For example, if the testing is performed with two mutually exclusive transformations and one of them generates 10% of ETIs, the resilience towards this transformation is neglected to some extent since it contributes only 10% to the overall loss value.

When evaluated using the RASR metric, an interesting finding is that, for some EPs of decreased fine-tuning loss, their corresponding RASR is not reduced obviously, and in some cases, the RASR remains the same. That is, the chance of triggering mis-predictions is not reduced despite the loss decrease; the fine-tuned DNN is not "safer." However, for EPs of increased fine-tuning loss, the RASR does not necessarily rise as well. This observation motivates us to propose a solution below for alleviating the issue of mutual exclusivity.

8.3 RQ2.3: Alleviating the Fine-Tuning Issue

Motivation. "Inconsistency" between the results of fine-tuning loss and RASR inspires our solution here. In practice, to have a correct and resilient prediction (i.e., the prediction does not change with different transformations), the loss does *not* necessarily need to be zero. An illustration is given in Fig. 6. For a two-class classification, the case of zero loss for predicting class 1 is given in Fig. 6(a). Nevertheless, since the final prediction is decided as the class having the highest probability, a 100% probability is unnecessary. As shown in Fig. 6(b), the prediction of class 1 remains the same, and also resilient, if its probability ranges within $[60\%, 80\%]$, which still has notable loss. Therefore, for two mutually exclusive transformations, we can orchestrate their fine-tuning losses at a moderate level to make the DNN resilient to both transformations.



(a) A prediction of class 1 with zero loss. (b) *Correct* predictions of non-zero loss. (c) Repairing by separately considering EPs.

Figure 6: Motivations for alleviating the mutual exclusivity.

Table 5: Results of (minimal RASR, maximal RASR) after fine-tuning. Lower RASR is better.

DNN	Original	Fine-tuned with Eq. 1	Fine-tuned with Eq. 6
ResNet	(1.5%, 76.0%)	(1.0%, 98.5%)	(0, 67.5%)
VGG	(10.0%, 78.5%)	(2.5%, 99.0%)	(0, 76.5%)
Inception	(0, 85.0%)	(12.0%, 100%)	(0, 70.0%)
MobileNet	(4.0%, 72.5%)	(3.0%, 99.0%)	(0, 61.0%)
DenseNet	(5.5%, 91.5%)	(0.5%, 65.5%)	(0, 56.0%)
FaceNet	(44.5%, 88.0%)	(43.5%, 97.0%)	(37.5%, 75%)
AlexNet	(0, 68.0%)	(0, 85.0%)	(0, 30.0%)
EfficientNet	(24.5%, 90.0%)	(23.0%, 100%)	(20.5%, 62.5%)
Average	(11.3%, 81.2%)	(10.6%, 93.0%)	(7.2%, 62.3%)

* Increased RASR after fine-tuning (i.e., becomes less resilient) are marked in red and reduced RASR are marked in green.

Solution. Though the fine-tuning loss cannot precisely characterize the resilience to a transformation, we still use it for fine-tuning because it is continuous (RASR is discrete and cannot be used as an optimization objective during fine-tuning). Unlike Eq. 1, we separately compute the fine-tuning loss for ETIs generated with each transformation, as shown below.

$$L^i = \frac{1}{|D|} \sum_{(x,y)\in D} \left[\max_{x' \sim t^i(x)} L(f_\theta(x'), y) \right], \tag{6}$$

where t_i denotes the i-th transformation. Then, the fine-tuning aims to minimize a transformation-/EP-wise loss: $\sum_i L^i$.

This way, different EPs are treated equally during the fine-tuning. Note that the #ETIs (generated during testing) does not quantify the resilience w.r.t. one EP (see Sec. 7). Rather, more ETIs estimates the fine-tuning loss better. As illustrated in Fig. 6(c), no matter how ETIs are distributed, the fine-tuned decision boundary is not biased to any EP. Despite that the overall fine-tuning loss is not the minimal, the predictions of the two classes are correct and resilient, as long as the decision boundary stays outside the dashed regions.

Results. Evaluation results are given in Table 5. We report the maximal and the minimal RASR of different transformations computed on DNNs "repaired" using the standard form of fine-tuning (as in Eq. 1) vs. our approach (as in Eq. 6). Each experiment is repeated three times. As marked by the red results in Table 5, the standard approach largely increases the maximal RASR in 7 out of 8 cases, and some of them even reach 100% (e.g., Inception and EfficientNet). Moreover, the original Inception is relatively resilient to some transformations (since the minimal RASR is zero), whereas the minimal RASR is notably increased to 12.0%, making it sensitive to almost all transformations.

Our approach notably reduces the maximal RASR, i.e., the DNN is no longer fragile to certain transformations. This shows that our

approach does not sacrifice the resilience towards certain transformations. The minimal RASR is also highly encouraging, given that all minimal RASRs decrease, and the lowest is zero for 6 out of 8 cases. Importantly, all our fine-tuned DNNs retain their original test accuracy (using the standard dataset).

Note that transformations having the minimal original RASR may only generate a few ETIs during testing. Thus, when fine-tuning with previous objectives (Eq. 1), they are often neglected. As a result, the DNN may become more sensitive to these transformations after fine-tuning. Nevertheless, our fine-tuning scheme treats each transformation equally, such that it can successfully repair those "stealthily erroneous" properties.

9 Discussion

Threat to Validity. This research unveils the pitfalls of ETIs in DNN testing (and fine-tuning) and accordingly proposes EP-oriented enhancement. One threat is that the evaluated DNNs, datasets, and testing/fine-tuning methods may be biased. To mitigate this threat, we select DNNs that have been widely studied in DNN testing and incorporated into mature commercial products. For example, the residual connection [24] in ResNet is the building block for nearly all modern DNNs and related systems, e.g., Detectron [3] and recent LLMs [4, 43]. ImageNet is almost the largest public dataset and is the golden standard to benchmark DNNs [46]. Our technical pipeline is not tailored to specific DNNs/datasets. Our evaluated testing methods (e.g., random, fuzzing-like, and prioritization-based testing) and objectives (e.g., structure and distribution-based coverage, adequacy, and diversity metrics) cover most recent works [33, 38, 42, 54, 62, 63, 68, 71]. The evaluated fine-tuning is the standard and fundamental form, and is widely adopted to repair DNNs [33, 40, 59, 65, 68, 69].

Another threat is that our adopted transformations may not be comprehensive. We mitigate this threat by considering, to the best of our knowledge, all pixel- and style-based transformations adopted in existing works [42, 54, 62, 63, 68, 71]. In addition, when identifying (local) perception-based transformations, we iteratively generate transformations on perceptions of different hierarchies, as introduced in Sec. 4.2 and illustrated in Fig. 3. Moreover, as mentioned in Sec. 6, we augment existing transformations with thousands of new (perception-based) transformations, and the diversity of these new transformations are ensured via a relatively strict deduplication process (Sec. 4.2); we believe such a large number ($\sim$ 2K vs. $\sim$ 10 in prior works) is a strong indicator of the comprehensiveness of our transformations.

Text (NLP) DNNs. Text DNNs' tasks also belong to classification or regression, and the internal logics are reflected as decision boundaries. In addition, inputs of text DNNs are word embeddings, which are floating-point vectors as images. Furthermore, given that our unveiled issues are due to the use of ETIs, not specific DNNs/tasks, it should be accurate to conclude that our findings are also applicable to text DNNs to a great extent. Currently, the testing of text DNNs is not comprehensively studied and is not as general as testing image DNNs. Most works focus on specific tasks (e.g., machine translation [25]) and their proposed input transformations cannot be used interchangeably. We note that contemporary works also use generative models to mutate text [69]; since our transformation generation techniques are out-of-the-box and agnostic to the generative model's implementation, it should be promising to generate comprehensive generic text transformations for future testing works using our techniques.

Textual Annotations for Transformations. While consistency and uniqueness evaluated in this study are sufficient to ensure the correctness of transformations in DNN testing, our currently generated transformations lack textual descriptions. Automatically annotating such descriptions should be an interesting future direction, as developers may require interpreting DNN errors. Although multi-modal LLMs are unable to automatically generate transformations (see Sec. 6), we foresee their high potential in complementing our generation technique by providing textual interpretations.

EP and Diversity. EPs can holistically reflect the *diversity* of DNN errors. However, we clarify that EP considered in this research is different from the "diversity property" studied previously. Previous works estimate the diversity of ETIs (e.g., via ETI's entropy [23], belonging classes [68], or clusters [5]). Nevertheless, conventional diversity metrics are calculated input-wise. Thus, merely different ETIs (generated using one transformation) can increase their diversity scores.

Distribution Shift Due to Transformations. As DNNs often assume that training and test data follow the same data distribution, it is expected that transformations adopted in testing do not induce distribution shift (i.e., producing out-of-distribution inputs). In essence, mis-predictions triggered by out-of-distribution inputs indicate incorrect usage of DNNs, not DNN errors. We clarify that our augmented perception-based transformations should *not* change the original data distribution. These transformations, to some extent, can be seen as the results of interpolation and extrapolation among perceptions of the generative model's training data. For instance, transformations for the face orientation are generated because the generative model's training data have both left- and right-oriented faces. In that sense, we cannot generate transformations for the face orientation if the generative model training data are all left-oriented. Since the generative model and the tested DNN use the same training data (i.e., the developer's data), our transformations will not bring new perceptions that do not exist in the tested DNN's training data (i.e., out-of-distribution perceptions).

10 Conclusion

This paper unveils that ETI-oriented testing frequently generates similar ETIs and neglects many DNN errors. The ETI-based fine-tuning may impair DNNs. By generating comprehensive input transformations, we show the advantage of EP-oriented DNN testing and fine-tuning. Large-scale evaluations show that our approaches can effectively detect and repair DNN errors.

Acknowledgments

We thank the anonymous ISSTA reviewers for their insightful and constructive comments. We also thank all participants in our pilot study for contributing to the evaluation data. The HKUST authors were supported in part by a RGC GRF grant under the contract 16214723 and a RGC CRF grant under the contract C6015-23G.

References

[1] [n. d.]. Amazon Mechanical Turk. https://www.mturk.com/.
[2] [n. d.]. Artifact. https://github.com/Yuanyuan-Yuan/EP-DNN-testing.

[3] [n. d.]. Detectron2: Facebook AI Research's platform for object detection and semantic segmentation. https://ai.facebook.com/tools/detectron2/.

[4] OpenAI (2023). 2023. GPT-4 Technical Report. (2023).

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

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

[7] Taejoon Byun and Sanjai Rayadurgam. 2020. Manifold for machine learning assurance. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 97–100.

[8] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. 1998. *Metamorphic testing: a new approach for generating next test cases*. Technical Report. Technical Report HKUST-CS98-01, Department of Computer Science, HKUST.

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. 2020. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[10] John M Danskin. 2012. *The theory of max-min and its application to weapons allocation problems*. Vol. 5. Springer Science & Business Media.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[12] Swaroopa Dola, Matthew B Dwyer, and Mary Lou Soffa. 2021. Distribution-aware testing of neural networks using generative models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 226–237.

[13] Isaac Dunn, Hadrien Pouget, Daniel Kroening, and Tom Melham. 2021. Exposing previously undetectable faults in deep neural networks. In *ISSTA 2021*.

[14] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2019. Exploring the landscape of spatial robustness. In *International conference on machine learning*. PMLR, 1802–1811.

[15] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 177–188.

[16] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[17] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.. In *International Conference on Learning Representations*.

[18] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. 2020. Importance-driven deep learning system testing. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 702–713.

[19] Gene H Golub and Christian Reinsch. 1971. Singular value decomposition and least squares solutions. *Linear algebra* 2 (1971), 134–151.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[21] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. 2019. Simple black-box adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2484–2493.

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

[23] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. 2020. Is neuron coverage a meaningful measure for testing deep neural networks?. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 851–862.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[25] Pinjia He, Clara Meister, and Zhendong Su. 2020. Structure-invariant testing for machine translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 961–973.

[26] James F Hemphill. 2003. Interpreting the magnitudes of correlation coefficients. (2003).

[27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.

[29] Zhenlan Ji, Pingchuan Ma, Yuanyuan Yuan, and Shuai Wang. 2023. CC: Causality-aware coverage criterion for deep neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1788–1800.

[30] Sungmin Kang, Robert Feldt, and Shin Yoo. 2023. Deceiving Humans and Machines Alike: Search-based Test Input Generation for DNNs using Variational Autoencoders. *ACM Transactions on Software Engineering and Methodology* (2023).

[31] Hamid Karimi, Tyler Derr, and Jiliang Tang. 2019. Characterizing the decision boundary of deep neural networks. *arXiv preprint arXiv:1912.11460* (2019).

[32] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.

[33] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1039–1049.

[34] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1466–1476.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[36] Caroline Lemieux, Jeevana Priya Inala, Shuvendu K Lahiri, and Siddhartha Sen. 2023. CODAMOSA: Escaping Coverage Plateaus in Test Generation with Pre-trained Large Language Models. In *45th International Conference on Software Engineering, ser. ICSE*.

[37] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[38] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *ASE 2018*.

[39] Nicholas Metropolis and Stanislaw Ulam. 1949. The monte carlo method. *Journal of the American statistical association* 44, 247 (1949), 335–341.

[40] Qi Pang, Yuanyuan Yuan, and Shuai Wang. 2022. MDPFuzz: testing models solving Markov decision processes. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 378–390.

[41] Qi Pang, Yuanyuan Yuan, and Shuai Wang. 2024. MPCDiff: Testing and Repairing MPC-Hardened Deep Learning Models. NDSS.

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

[43] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[44] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[45] Vincenzo Riccio and Paolo Tonella. 2023. When and why test generators for deep learning produce invalid inputs: an empirical study. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1161–1173.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[47] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. 2018. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*.

[48] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[49] Julius Sim and Chris C Wright. 2005. The kappa statistic in reliability studies: use, interpretation, and sample size requirements. *Physical therapy* (2005).

[50] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

[52] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[53] Richard Taylor. 1990. Interpretation of the correlation coefficient: a basic review. *Journal of diagnostic medical sonography* 6, 1 (1990), 35–39.

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

[55] Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems* 32 (2019).

[56] Zhuozhuo Tu, Jingwei Zhang, and Dacheng Tao. 2019. Theoretical analysis of adversarial learning: A minimax approach. *Advances in Neural Information Processing Systems* 32 (2019).

[57] Jonathan Uesato, Brendan O'donoghue, Pushmeet Kohli, and Aaron Oord. 2018. Adversarial risk and the dangers of evaluating against weak attacks. In *International Conference on Machine Learning*. PMLR, 5025–5034.

[58] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. Robot: Robustness-oriented testing for deep learning systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 300–311.

[59] Shuai Wang and Zhendong Su. 2020. Metamorphic object insertion for testing object detection systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 1053–1065.

[60] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 397–409.

[61] Yi-Lun Wu, Hong-Han Shuai, Zhi-Rui Tam, and Hong-Yu Chiu. 2021. Gradient normalization for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6373–6382.

[62] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.

[63] Xiaoyuan Xie, Pengbo Yin, and Songqiang Chen. 2022. Boosting the Revealing of Detected Violations in Deep Learning Testing: A Diversity-Guided Method. In *ASE 2022*.

[64] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* 56, 4 (2023), 1–39.

[65] Boxi Yu, Zhiqing Zhong, Xinran Qin, Jiayi Yao, Yuancheng Wang, and Pinjia He. 2022. Automated testing of image captioning systems. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 467–479.

[66] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365* (2015).

[67] Yuanyuan Yuan, Qi Pang, and Shuai Wang. 2022. Unveiling Hidden DNN Defects with Decision-Based Metamorphic Testing. In *37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[68] Yuanyuan Yuan, Qi Pang, and Shuai Wang. 2023. Revisiting neuron coverage for dnn testing: A layer-wise and distribution-aware criterion. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1200–1212.

[69] Yuanyuan Yuan, Qi Pang, and Shuai Wang. 2024. Provably Valid and Diverse Mutations of Real-World Media Data for DNN Testing. *IEEE Transactions on Software Engineering* (2024).

[70] Yuanyuan Yuan, Shuai Wang, Mingyue Jiang, and Tsong Yueh Chen. 2021. Perception matters: Detecting perception failures of vqa models using metamorphic testing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16908–16917.

[71] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. 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*. 132–142.

[72] Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. 2019. Lipschitz generative adversarial nets. In *International Conference on Machine Learning*. PMLR, 7584–7593.

[73] Jiapeng Zhu, Ruili Feng, Yujun Shen, Deli Zhao, Zheng-Jun Zha, Jingren Zhou, and Qifeng Chen. 2021. Low-rank subspaces in gans. *Advances in Neural Information Processing Systems* 34 (2021), 16648–16658.

[74] Jiapeng Zhu, Yujun Shen, Yinghao Xu, Deli Zhao, and Qifeng Chen. 2022. Region-Based Semantic Factorization in GANs. In *International Conference on Machine Learning*. PMLR, 27612–27632.

[75] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

Received 2024-04-12; accepted 2024-07-03