TAEFuzz: Automatic Fuzzing for Image-based Deep Learning Systems via Transferable Adversarial Examples

SHUNHUI JI, Hohai University, China
CHANGRONG HUANG, Hohai University, China
BIN REN, Hohai University, China
HAI DONG, RMIT University, Australia
LARS GRUNSKE, Humboldt Universität zu Berlin, Germany
YAN XIAO, Sun Yat-sen University, China
PENGCHENG ZHANG*, Hohai University, China

Deep learning (DL) components have been broadly applied in diverse applications. Similar to traditional software engineering, effective test case generation methods are needed by industry to enhance the quality and robustness of these deep learning components. To this end, we propose a novel automatic software testing technique, *TAEFuzz* (Automatic Fuzz-Testing via Transferable Adversarial Examples), which aims to automatically assess and enhance the robustness of image-based deep learning (DL) systems based on test cases generated by transferable adversarial examples. *TAEFuzz* alleviates the over-fitting problem during optimized test case generation and prevents test cases from prematurely falling into local optima. In addition, *TAEFuzz* enhances the visual quality of test cases through constraining perturbations inserted into sensitive areas of the images. For a system with low robustness, *TAEFuzz* trains a low-cost denoising module to reduce the impact of perturbations in transferable adversarial examples on the system. Experimental results demonstrate that the test cases generated by *TAEFuzz* can discover up to 46.1% more errors in the targeted systems, and ensure the visual quality of test cases. Compared to existing techniques, *TAEFuzz* also enhances the robustness of the target systems against transferable adversarial examples with the perturbation denoising module.

CCS Concepts: • **Computing methodologies** → *Artificial intelligence*; • **Security and privacy** → *Software security engineering*; • **Software and its engineering** → *Software testing and debugging*.

Additional Key Words and Phrases: Deep learning, fuzzing, transferable adversarial examples, robustness

1 INTRODUCTION

Deep Neural Networks (DNNs) have demonstrated superior performance compared to traditional computing technologies in various tasks, such as image classification [1], image semantic segmentation [2] and object detection [3]. However, the vulnerability of DNNs against adversarial examples with small perturbations [4–6] or

*Corresponding author

Authors' Contact Information: Shunhui Ji, Hohai University, Nanjing, China, shunhuiji@hhu.edu.cn; Changrong Huang, Hohai University, Nanjing, China, changronghuang@hhu.edu.cn; Bin Ren, Hohai University, Nanjing, China, 2991224521@qq.com; Hai Dong, RMIT University, Melbourne, Australia, hai.dong@rmit.edu.au; Lars Grunske, Humboldt Universität zu Berlin, Berlin, Germany, grunske@informatik.hu-berlin.de; Yan Xiao, Sun Yat-sen University, Shenzhen, China, xiaoy367@mail.sysu.edu.cn; Pengcheng Zhang, Hohai University, Nanjing, China, pchzhang@hhu.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s).
ACM 1557-7392/2025/1-ART
https://doi.org/10.1145/3714463

Just Accepted

simple spatial transformations [7, 8] downgrades their applicability in safety-critical systems [9, 10]. Therefore, it is increasingly critical to test DNN models to ensure their validity and robustness [11, 12].

Similar to traditional software testing techniques, DNN testing relies on high-quality and representative test cases. Further, analogous to test-driven evolutionary program repair [13–16] and synthesis-driven repair [17–20], vulnerable programs can be repaired with enhanced test case generation techniques [21–24]. Since it is challenging to uncover all errors in DNN models, fuzzing is adopted to find possible errors by generating random and unexpected test cases [25]. Recent studies [26, 27] have shown that using test cases generated by fuzzing can significantly enhance the robustness of DNN models against adversarial examples, where fuzzing can generate adversarial examples as test cases. Although current test case generation methods [8],[28–30] can find errors in DNN models, they do not use transferable adversarial examples, which are a type of examples generated for a surrogate model with a similar decision boundary as the target model. In general, malicious attacks can be divided into white-box and black-box attacks. White-box attacks require an attacker to be fully aware of a target model's structure, parameters, and input and output information, while black-box attacks are usually built on the prerequisite of a model's input and output knowledge. The strict preconditions of white-box attacks greatly challenge attackers to implement effective attacks [31–34]. Although some attackers can use query-based black-box attacks [35–37] to enable target DNN models to make incorrect predictions, such methods require accessing API interfaces often for gaining the models' information and can be easily detected [38]. In contrast to white-box attacks, transfer-based attacks are basically black-box and do not require precise information about a target model [39, 40]. However, transfer-based attacks utilize the a priori knowledge that different models have

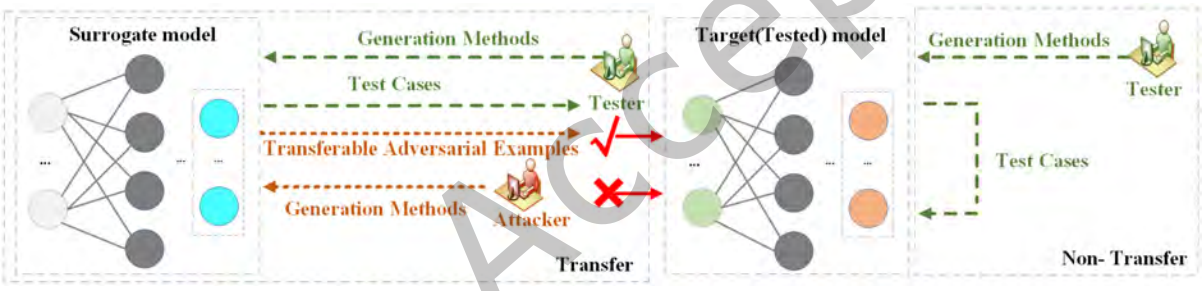


Fig. 1. Transferable adversarial examples generation: The attacker generates transferable adversarial examples on the surrogate model and inputs them into the target model for attack. The tester simulates the attack mode of the malicious attacker and generates test cases on the surrogate model. These test cases are input into the target model to find the transferable errors.

similar decision boundaries [41], and that attackers only need to generate transferable adversarial examples on a surrogate model to be able to force the target model to make incorrect decisions, as shown in Fig. 1. For example, FGSM [5] is used to generate an adversarial example on a surrogate model to test a target model by maximizing the loss function value relative to the input example. Transfer-based attacks are more devastating, considering the simplicity of their execution conditions. In contrast, non-transfer-based testing relies on technical details of target models for test case generation, which is more challenging to execute in most real-world environments. Therefore, we can conclude that: *Transferable adversarial examples are more serious threats to DNN models in the real world. It is of great significance to investigate the security and robustness of DNN models against transferable adversarial examples.*

In the deep learning community, a few studies focus on employing the gradient information feedback obtained from surrogate models to fuzz examples. These fuzzed examples are transferable adversarial examples. MI-FGSM(MI) [42] and NI-FGSM [43] accumulate the gradients obtained during the iterations of test case generation,

Just Accepted

enabling a stable optimization direction. DMI-FGSM(DMI) [44], TMI-FGSM(TMI) [45] and VMI-FGSM(VMI) [46] increase the input diversity via spatial variations, alleviating the over-fitting problem on the surrogate model. In the software engineering community, noticeable efforts have been made to generate test cases that can expose the vulnerabilities of DNN models and mitigate the adversarial threats to improve the robustness of DNN systems [27, 47–49]. However, none of them focus on transferable adversarial examples, not to mention generating solutions to address the threat of such examples and improving the robustness of target models. Thus, we work on three challenges:

1) Weak ability to discover transferable errors. To discover transferable adversarial errors in DNNs, many methods [42, 45, 46] apply gradients of loss functions to input examples as the direction of perturbation. In transferable adversarial attacks, the gradient information of neighboring input examples is important [46, 50]. VMI [46] averages the gradients of multiple neighbor examples to stabilize the optimization direction. As shown in Fig. 2, the horizontal and vertical axes in the figure represent the input space. The red solid circle represents the region where most neighbor examples of input example x_t are distributed during the t-th iteration. According to uniform distribution rule, these examples are confined within a relatively fixed perturbation radius. The blue dotted circle and black dotted circle indicate the region of more diverse neighbor examples with different perturbation radii. With perturbations of a fixed radius, the range of gradient variations is limited, which can cause the gradient directions to converge over multiple iterations, leading the optimization process to fall into local optimum. To elaborate on this point, we conducted an experiment. As shown in Fig. 3, the two rows

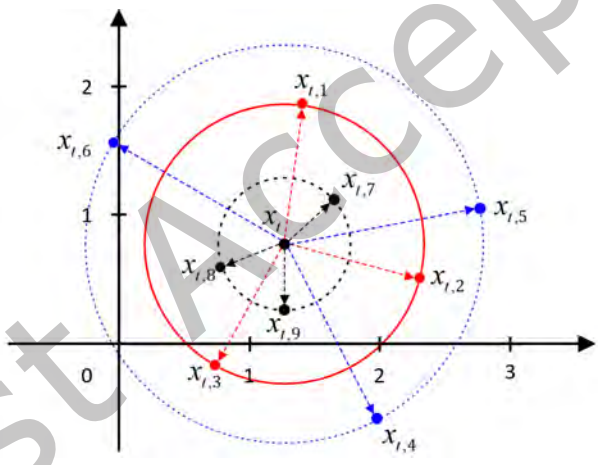


Fig. 2. Distribution of neighbor examples generation.

represent the class activation maps of the last three neighbor examples generated in the final iteration by the VMI method and our RMI method, respectively. The neighbor examples generated by VMI (first row) have almost identical attention regions since VMI uses a fixed perturbation radius to generate neighbor examples, which limits its exploration space and makes it prone to getting stuck in local optima. In contrast, the neighbor examples generated by RMI (second row) exhibit greater diversity in the attention regions, indicating that our RMI method generates more diverse neighbor examples by using varying perturbation radii, thus expanding the search space. This diverse perturbation allows for better exploration of the global optimum and helps avoid local optima. In addition, there is an overfitting problem due to differences in gradient sizes of loss functions of the models relative to input examples, which leads to different contributions in the optimization results. These will result in testing

Just Accepted

methods that only find errors caused by fewer transferable adversarial examples, especially under targeted attacks and on robust models.

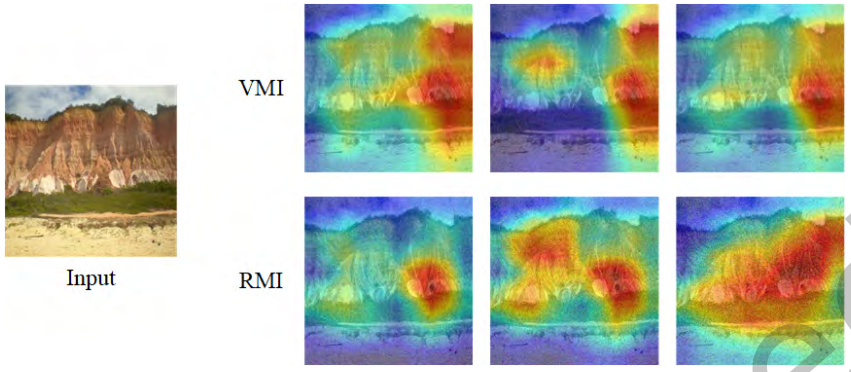


Fig. 3. Class activation map of neighbor examples.

2) Low quality of generated test cases. In some instances, transfer-based attacks require greater image distortion to enhance their efficacy [51]. Most transfer-based attacks use the L_∞ norm as the perturbation constraint, which results in lower attack transferability [52]. However, the L_∞ norm does not consider the relationship between individual and surrounding pixels, resulting in remarkable perturbations added into smooth areas of images. They significantly degrade the quality of images. Fig. 4 shows the transferable adversarial examples respectively generated by MI, CT-MI and CT-VMI [46]. It can be observed that there are remarkable artifacts in the transferable adversarial examples, especially in the example generated by the more powerful CT-VMI.

Fig. 4. Comparing transferable adversarial examples from different methods.

3) Low robustness of DNNs. The perturbations added in transferable adversarial examples significantly differ from the high-frequency noise of typical adversarial examples, which contain many large chunks of low-frequency noise. In Fig. 4, from left to right, the perturbations of the transferable adversarial examples gradually change from high-frequency noise to large chunks of low-frequency noise with transfer-based attacks with increased capability. Existing methods for generating non-transferable adversarial examples cannot effectively improve the robustness of target models against transferable adversarial examples [46].

Proposed Technique. To address these three challenges, we propose a software testing technique, *TAEFuzz* (Fuzz-Testing via Transferable Adversarial Examples), aiming to automatically assess and enhance the robustness of image-based deep learning systems based on the test cases generated by fuzzing(Fig. 5).

Just Accepted

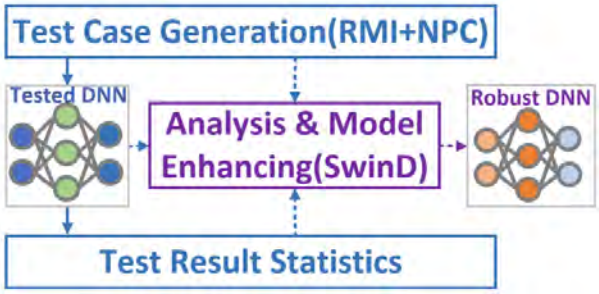


Fig. 5. TAEFuzz's contributions.

1) We enhance transferable error detection through multiple Radius Examples MI-FGSM(RMI). RMI can increase the diversity of input examples to alleviate the over-fitting problem, thereby preventing test cases from prematurely falling into local optima. Experimental results confirm that RMI successfully identifies more transferable errors in the test methods.

2) To improve the quality of test cases, we use Neighbor Pixel Constraint (NPC). NPC uses the relationship between a central pixel and its surrounding pixels to determine the size of a perturbation, instead of setting identical perturbation values for all the pixels in an input example. NPC guarantees that the generated transferable adversarial examples have high visual quality, i.e., the test cases are more realistic.

3) The test cases generated by fuzzing are used to test the robustness of the target model on the test results. If the model exhibits a high degree of robustness, the target model will be returned; otherwise, we adopt the Image Restoration-SwinTransformer (SwinIR) [53] to train a denoising module SwinD based on the characteristics of low-frequency noise in transferable adversarial examples. The denoising module SwinD can effectively eliminate the impact of artificial perturbations on the final classification performance, thereby improving the robustness of the target model and guaranteeing its reliability and security. The training cost of the denoising module is high, requiring a large amount of transferable adversarial examples. To address this issue, the test cases generated by fuzz-testing are used to expand the training set by the Perturbation Mixing (PerMix) operation. This operation enables benign examples to contain the perturbation characteristics of transferable adversarial examples and significantly reduces the training cost.

In summary, this paper makes the following **contributions**:

- The proposed fuzzing method based on RMI can automatically generate test cases to find more transferable errors in DNN models (Subsec 3.2.2).
- Our proposed NPC operation can effectively reduce L_1 and L_2 norm values of perturbations and avoid adding perturbations in the smooth region of an input example, thus improving the quality of test cases (Subsec 3.2.1).
- The SwinD denoising module can reduce the impact of artifacts on a DNN model, thereby enhancing its robustness (Subsec 3.3.1).
- The experimental results show that RMI improves the attack success rate by 17.7%-37.1% using a single surrogate model for non-targeted attacks. For targeted attacks, the attack success rate is increased by 56.2%-71.2% compared to the best baseline methods. Considering the L_1 and L_2 metrics together, NPC performs well in reducing the overall disturbance magnitude. SwinD improves the robustness of the target models by an average of 10% compared to the existing methods (Sec 4). The code is available at https://anonymous.4open.science/r/TAEFuzz-4575.

Just Accepted

The remainder of the paper is organized as follows: Section 2 presents some basic concepts on adversarial attacks and Robustness Enhancement. In Section 3, the *TAEFuzz* approach is proposed. Section 4 describes experimental setup and presents experimental results. Related work is described in Section 5. Finally, Section 6 concludes the paper.

2 BACKGROUND

2.1 Adversarial Attacks

Adversarial examples generated by adversarial attacks are a critical means to test DNNs [54]. Adversarial attacks can be classified into targeted and non-targeted adversarial attacks according to the target label types of adversarial examples. Suppose there is an image classification dataset X and its corresponding set of labels Y, and an example (x, y) $(x \in X, y \in Y)$, where x denotes an example, and y represents the valid label of the example. For a DNN $f(\cdot)$ adopted for image classification, let $f(x)$ denote the output of the model when the input example is x and $f(x) = y$ when the label of x is correctly predicted. The prediction label of an adversarial example x^{adv} generated by a non-targeted attack only needs to satisfy $f(x^{adv}) \neq y$, where $x^{adv} = x + \delta$ and δ denotes the adversarial perturbation added to the input example x. Unlike non-targeted attacks, the predicted label of the adversarial example x^{adv} generated by a targeted attack needs to satisfy $f(x^{adv}) = y^t$, where y^t denotes the target labels specified by the attacker and $y^t \neq y$. To maintain the quality of x^{adv}, the perturbation δ added to the example needs to be constrained to satisfy:

$$\|x - x^{adv}\|_l < \epsilon \tag{1}$$

where ϵ is the perturbation threshold, and $\|\cdot\|_l$ is the L_l norm. The L_0, L_2 and L_∞ norms are usually chosen.

2.2 Transfer-based Attacks

To improve the capability of the methods to discover the errors caused by transferable adversarial examples in target models, MI-FGSM [42] adds a momentum mechanism to I-FGSM [55]. This alteration accumulates the gradients of the previous iterations, which can alleviate the problem that the methods easily fall into local optima. The formal expression of MI-FGSM is as follows:

$$n_t = \nabla_{x_t} J(x_t, y) \tag{2}$$

$$g_t = \mu \cdot g_{t-1} + \frac{n_t}{\|n_t\|_1} \tag{3}$$

$$x_{t+1} = Clip_{x,\epsilon}\{x_t + \alpha \cdot sign(g_t)\} \tag{4}$$

where n_t denotes the gradient of the $t\text{-}th$ iteration output of the model relative to the $t\text{-}th$ iteration input x_t, and α denotes the perturbation step of each iteration. The perturbation values that exceed the constraint threshold ϵ are truncated so that each perturbation in the generated example satisfies the constraint. In order to further stabilize the update direction and avoid the attacks from falling into a local optimum, the VMI-FGSM(VMI) [46] method does not directly use the current gradient for momentum accumulation at each iteration. Instead, it considers the gradient variance of the previous iteration to adjust the current gradient (see Eq. (5)) to stabilize the gradient optimization direction and avoid local optima.

$$n_t = \nabla_{x_t} J(x_t, y) + v(x_{t-1}) \tag{5}$$

$$v(x_{t-1}) = \frac{1}{M} \sum_{j=1}^{M} \nabla_{x_{t-1,j}} J(x_{t-1,j}, y; \theta) - \nabla_{x_{t-1}} J(x_{t-1}, y; \theta) \tag{6}$$

In addition to the above gradient calculation process, the rest of the method is basically the same as MI-FGSM, where $x_{t-1,j} = x_{t-1} + \zeta$, ζ is the generated random perturbation, j denotes the $j\text{-}th$ neighbor example of the $t-1$ iteration input example x_{t-1}, and finally, the mean gradient of M neighbor examples minus the gradient of x_{t-1} is calculated as the gradient variance $v(x_{t-1})$.

Just Accepted

2.3 Robustness Enhancement Methods

Researchers can evaluate the robustness of a target model through testing, particularly in the context of image classification deep neural network (DNN) models. Typically, model robustness is assessed by measuring the classification accuracy of the target model when exposed to a set of test cases. DNN models that have not undergone robustness enhancement tend to exhibit lower classification accuracy on these test cases, indicating their reduced robustness.

Robustness enhancement of models based on adversarial training. The method to enhance the robustness of models based on adversarial training mainly involves incorporating examples with added perturbation or noise as either all or part of the training data input into the model, thereby enabling the DNN model to resist perturbations or noise interference. HGD [56] defines the loss function during model training as the difference between benign examples and perturbed examples, reducing the impact of minor perturbations that might be amplified by deep neural networks, thus reducing the interference of perturbations on classification results. PGD [57] uses projected gradient descent to generate adversarial examples, optimizing the model to withstand the most damaging perturbations. Through this adversarial training, the model becomes more robust against various attacks. Fast-AT [58] proposes an FGSM-based adversarial training method with random initialization, where adding a random initial perturbation generates adversarial examples that enable the model to achieve robustness comparable to PGD training at a lower computational cost. This approach incorporates cyclic learning rates and mixed-precision optimization to significantly enhance training speed, achieving fast adversarial training. RS [59] inputs examples with added random Gaussian noise into the DNN model for adversarial training, enabling the model to automatically learn and acquire the ability to resist noise interference. RS4A [60] constructs a smoothed classifier by adding specific noise to the input data, thereby enabling the model to maintain robustness within perturbation ranges under different norms.

Robustness enhancement of models based on input transformations. One method to enhance the robustness of models is JPEG compression [61]. Bit-Red [62], building upon the JPEG compression, is a more comprehensive compression approach that encompasses two primary facets. Firstly, it reduces the color depth of individual pixel points. Secondly, it reduces the variability between pixel points using a smoothing strategy. Both of these aspects are executed through feature compression, which preserves the essential features of the examples while minimizing the interference caused by redundant information in adversarial examples on the model's predictions. In contrast to compression methods, two alternative approaches for enhancing model robustness are R&P [7] and NRP [63]. R&P employs input transformation methods based on random resizing and random padding strategies. NRP utilizes input transformation methods that rely on a network of purifiers to substantially reduce the perceptual feature disparities between benign and antagonistic examples to an extremely low level.

3 THE *TAEFUZZ* APPROACH

The overview of *TAEFuzz* is shown in Fig. 6. TAEFuzz's guiding strategy is different from traditional adversarial example generation methods. Some traditional fuzzing methods [25, 26] use the feedback from the target model directly, while TAEFuzz's transfer-based methods use the feedback from a surrogate model. We create an entire software engineering pipeline to improve the robustness of the image-based DNN components. Its algorithmic expression is presented in Algorithm 1 and Algorithm 2, which comprises three steps.

Step 1: Data and Model Preparation. The first step is to collect and process the benign data to generate data with the same size and pixel value range as the model input. Then, we put the processed data into the processing pool (detailed in line 3 in Algorithm 1). The surrogate model is a white-box model with the same image classification capabilities as the target model. Testers only need to know the structure and parameter information of the surrogate model, but not the target model to be tested.

Just Accepted

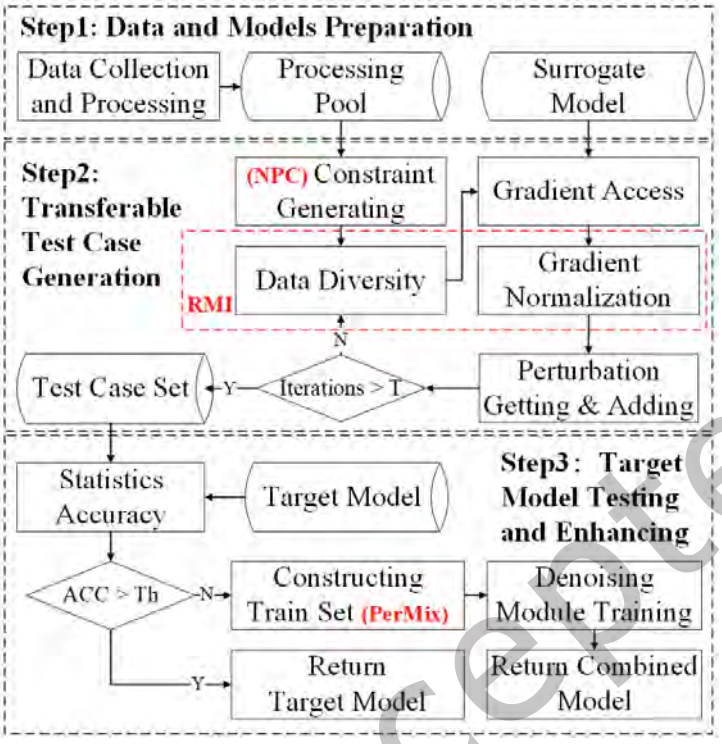


Fig. 6. The overview of *TAEFuzz*.

Step 2: Transferable Test Case Generation. The second step is the generation of transferable tests. First, the example x is selected from the data processing pool as the input example x_0 for the 0-th iteration to generate a constraint E based on the relationship between each pixel in x and its surrounding pixels (detailed in lines 5-6 in Algorithm 1 and Section 3.2.1). The random perturbation of the specified radius r is obtained through the multiplication of the scale factor R_{i+1} and the noise p that follows the uniform distribution $U(-\rho, +\rho)$. Then, the perturbation P_j^r is added to x_t as the neighbor example $x_{t,r,j}$, where x_t denotes the input example x at the t-th iteration. By repeating the above steps to generate M examples, the average of normalized gradients on the radius r can be calculated. Once the examples based on all radii are generated, we calculate the average of the normalized gradients for all the radii, denoted as n_t (detailed in lines 9-10 in Algorithm 1 and Sections 3.2.2-3.2.3). The gradient n_t is stacked for non-targeted attacks to get g_t (Eq. (3)). By employing g_t and generating the perturbation for the current iteration according to the step α (Eq. (13)), the perturbation is added to the input example x_t. Since gradient stacking results in the direction of the gradient pointing less precisely to the target class and leading to weaker results, the momentum-based gradient accumulation strategy of MI-FGSM cannot be applied to targeted attacks. Instead, perturbations are generated based on the projection-based strategy (Eq. (14)). For both targeted and non-targeted, the perturbation after clipping the part that does not fulfill the constraint is added to the example x_t to generate the example x_{t+1} for the current iteration (detailed in Line 11 in Algorithm 1 and Section 3.2.4). If the maximum number of iterations is reached, the examples generated from the final iteration are stored in the test case set D' (detailed in lines 13-14 in Algorithm 1). Otherwise, the generated examples are used as input for the next iteration.

Just Accepted

Algorithm 1: *TAEFuzz's* Test Case Generation

Input: D: The datasets to test; ϵ: The maximum perturbation; T: The number of iteration; N: The number of radii; M: The number of examples in some radius; μ: The decay factor; β: The factor for radius; inter: The maximum scale factor;
Output: A test case set D'.

1 $\alpha = \epsilon / T$;
2 $\rho = \alpha * \beta$;
3 D = Preprocessing(D) ;
4 $R_{i+1} = Linspace(0, inter, N + 1)$;
5 **for** x *in* D **do**
6 E = ConstraintGeneration(x, ϵ) ;
7 $g_0 = 0$; $x_0 = x$;
8 **for** $t = 0 \rightarrow T - 1$ **do**
9 Get the average M examples gradient of each radius by Eq. (11) ;
10 Get the average gradient of N radii by Eq. (12) ;
11 Get t-th perturbation and add it to x_t ;
12 **end**
13 Get the transferable adversarial examples x^{adv} corresponding to x ;
14 Save x^{adv} as test case in D' ;
15 **end**
16 **return** test case set D';

Step 3: Target Model Testing and Enhancing. The third step is to deploy a denoising module in front of the target model to improve its robustness. The denoising module aims to perform noise reduction on the fed inputs based on the characteristics of the transferable adversarial examples' perturbation. To reduce the cost of training the denoising module, a new PerMix operation is designed to augment the training data with the test case set and the set of benign examples (detailed in Line 1 in Algorithm 2 and Section 3.3.1). We then train SwinIR as the network structure of the denoising module. The trained denoising module and the target model are combined into a new model with higher robustness (detailed in lines 2-4 in Algorithm 2 and Section 3.3.2).

Algorithm 2: *TAEFuzz's* Model Enhancement

Input: C: The dataset to train; D': The test case set; The target model.
Output: A robustness combined model.

1 Construct training set using Eq. (16), D' and C ;
2 Construct denoising module using SwinIR and training set ;
3 Get a new model by combining SwinIR and the target model ;
4 **return** The combined Model;

3.1 Data and Models Preparation

First, examples on which the target model can make correct judgments are collected to form a benign data set D. The data is processed twofold: (1) converting the examples into the right length and width required by the model;

Just Accepted

(2) normalizing all pixels' values of the example to the specified range. Finally, the processed examples are fed into the data processing pool for transferable examples. Moreover, we collect or train the DNN model with the same classification capability of the target model as the surrogate model, based on the capability matching of Dong et al. [45]. To simulate the security threat in the actual scene, the structure and parameters of the surrogate model differ from those of the target DNN model.

3.2 Transferable Test Case Generation

3.2.1 Perturbation Constraint Generation. Currently, the L_∞ norm is used as a constraint for perturbation in transfer-based attacks. Under the L_∞ norm constraint, the maximum perturbation change per pixel is set to ϵ. The perturbation's value can vary between $[-\epsilon, +\epsilon]$. This setting does not consider the relationship between individual pixels and their surrounding pixels, so artifacts created in smooth areas of an image can be easily detected. Luo et al. [64] have shown that humans are much less sensitive to perturbations added to image boundaries. Therefore, we devise a method, **NPC**, for *TAEFuzz* to determine the perturbation range. NPC can generate the constraint of a single pixel based on the relationship between the pixel and its surrounding pixels. The specific process of NPC is: (1) obtaining and ranking the B neighborhood pixel values for a target pixel in the example; (2) selecting the maximum value $pmax$ and the minimum value $pmin$ as the upper and lower bound constraints for the pixel, i.e., $[pmin, pmax]$; (3) repeating the above steps until all pixels have been traversed.

3.2.2 Data Diversity. Here, the data diversity refers to the proper adjustment of the gradient of an input example to stabilize the optimization direction and prevent local optima. We propose a new data diversity method, **RMI**. RMI generates neighbors of an example x_t at multiple perturbation radii to increase the diversity of the neighbors of the example x_t at a specific iteration t. It is challenging to directly generate random perturbations that follow a uniform distribution with a specified perturbation radius r. In contrast, it is much simpler to create random perturbations with r', the mean of all pixel values, according to the uniform distribution. When the height H and width W of the input image example are significant, the relationship between r and r' can be obtained from the nature of the uniform distribution as follows.

$$r \approx 2 \cdot r' \cdot \sqrt{W \cdot H} \tag{7}$$

There is a one-to-one correspondence between r' and r. Therefore, generating random perturbations with the mean pixel value r' is equivalent to generating random perturbations with different perturbation radii r via the following steps:

First, M different random perturbations with the mean pixel value r' are generated on a radius r. One of the random perturbations is denoted as P^r.

$$P^r \sim U(-d, d) \tag{8}$$

where r determines the radius of the perturbation and $d = R_{i+1} \cdot \rho$. This way, M random perturbations can be generated for each of the N different radius values. This creates a total of $M \times N$ neighbor examples of x_t, where the $j\text{-}th$ example generated with radius r can be expressed as:

$$x_{t,r,j} = x_t + P_j^r \tag{9}$$

3.2.3 Gradient Normalization. There is gradient non-fairness in current transfer-based attacks (see Challenge 1 in Section 1). To improve the fairness between the generated gradients from different models and neighbor examples, we normalize the gradient of each example so that each gradient has the same contribution weight. Here, we normalize each gradient to a mean value of 1.0, which can be described in the following form:

$$n'_{t,r,j} = \frac{\nabla_{x_{t,r,j}} J(x_{t,r,j}, y)}{\left\| \nabla_{x_{t,r,j}} J(x_{t,r,j}, y) \right\|_1} \tag{10}$$

Just Accepted

where $x_{t,r,j}$ denotes a neighbor example, and y denotes the true label of $x_{t,r,j}$. Gradient fairness is achieved by dividing the gradient values by the sum of their absolute values.

The gradient of the M neighbor examples at the $r\text{-}th$ radius during the $t\text{-}th$ iteration is:

$$n_{t,r} = \frac{1}{M} \sum_{j=1}^{M} n'_{t,r,j} \tag{11}$$

Here, to make the gradients on different radii have the same contribution, $n'_{t,r,j}$ is normalized. Hence, the average gradient n_t over N different radii is:

$$n_t = \frac{1}{N} \sum_{r=1}^{N} \frac{n_{t,r}}{\|n_{t,r}\|_1} \tag{12}$$

Slightly different from non-targeted attacks, the gradient n_t of the M-neighbor example at the $r\text{-}th$ radius during the $t\text{-}th$ iteration y is replaced with y^t in Eq. (10) for targeted attacks.

3.2.4 Perturbation Getting and Adding. For non-targeted attacks, similar to VMI [46], we use MI-FGSM [42] to perform momentum accumulation on the generated gradients n_t, so the generated example x_{t+1} after adding perturbations to the input example at the $t\text{-}th$ iteration is:

$$x_{t+1} = Clip_{x,E}\{x_t + \alpha \cdot sign(g_t)\} \tag{13}$$

where E is the set of perturbation constraints for pixels, and g_t is the gradient generated in the iteration. $Clip\{\cdot\}$ truncates the pixel values that are outside the constraint E.

For targeted attacks, the experiments of Patch++ [65] show that the projected gradient can find more transferable errors in the target model. A step size amplification factor is introduced in each iteration to extend the perturbation's influence to more areas. And a kernel function is utilized to project the pixel values of overflow constraint ϵ to surrounding areas, which prevents information loss. Thus, we use Patch++' s gradient projection strategy. The example x_{t+1} generated at the $t\text{-}th$ iteration is:

$$x_{t+1} = Clip_{x,E}\{x_t + \nu \cdot \alpha \cdot sign(n_t) + \gamma \cdot \nu \cdot \alpha \cdot sign(V * C)\} \tag{14}$$

where E and n_t are calculated using Eq. (12). V is a Gaussian convolution that projects the values in C onto the neighboring pixels surrounding the central pixel. The calculation formula for C is:

$$C = Clip(\mid a_{t+\mathbf{1}} \mid -\epsilon, 0, \infty) \odot sign(a_{t+1}) \tag{15}$$

where a_{t+1} represents the perturbation directly added to the example at the $t\text{-}th$ iteration before applying the clipping operation. The values exceeding the constraint ϵ at the $t\text{-}th$ iteration are stored in C. The amount of perturbations beyond the perturbation constraint of each pixel is projected onto its surrounding pixels. This approach effectively extends the perturbation to a broader area, enhancing the overall impact of the perturbation.

3.3 Target Model Testing and Enhancing

A system developer investigates the classification accuracy of the target model under the set of test cases. If the model does not meet the security requirements, the robustness of the model needs to be further improved. Here, we use a powerful noise reduction module to perform noise reduction on the input. The following describes how to build a noise reduction module, SwinIR-Based Denoising **SwinD**.

Just Accepted

3.3.1 Construct Training Set. As shown in Fig. 4, the noise characteristics of examples generated by the better-performing transfer-based methods are vastly different from those generated by general methods, i.e. I-FGSM [55]. Therefore, using the randomly generated noise as a data enhancement method to train the denoising module is not enough. Instead, we need to train a denoising module using transferable adversarial examples. However, the noise reduction module training requires large training data with transferable examples' perturbation characteristics. In addition, the cost of obtaining these examples is usually huge in terms of time and computational resources. To reduce the cost of acquiring training data, we propose a method, **PerMix**, to expand the training set based on noise characteristics of transferable adversarial examples, by employing a small amount of data from test cases. Similar to the data enhancement method MixUp [66], PerMix also enhances the data, but unlike MixUp, PerMix is designed to expand transferable examples' perturbation characteristics in training data.

$$\tilde{x} = Clip\{x + (\varphi \cdot (x_a - x_{ao}) + (\mathbf{1} - \varphi) \cdot (x_b - x_{bo}))\} \tag{16}$$

where x_a and x_b are the two randomly selected tests from the test case set, x_{ao} and x_{bo} are the corresponding benign examples, and φ represents a mixing weight. Finally, $Clip\{\cdot\}$ is used to ensure all pixel values are within the legal range. All enhanced examples are added to the training set.

3.3.2 Denoising Module Training. At present, the performance of image-denoising methods has been improved greatly. Therefore, instead of designing a new denoising module, we choose SwinIR [53], an existing denoising method with proven performance. SwinIR can remove heavy noise corruption and preserve high-frequency image details, resulting in sharper edges and more natural textures. In contrast, other methods [67–69] suffer from either over-smoothness or over-sharpness, and cannot recover rich textures. Hence, we adopt the new training set to train SwinIR as the denoising module. The trained denoising module is a prepositional procedure of the target model to perform noise reduction for its input.

4 EVALUATION

To better understand the performance of *TAEFuzz*, we evaluate *TAEFuzz* on widely-used datasets and models. The experiments are designed to answer the following research questions:

RQ1. Can TAEFuzz's RMI find more transferable errors compared to other methods?
RQ2. Can TAEFuzz's NPC improve image quality?
RQ3. Can TAEFuzz's SwinD module improve the robustness of the model?

4.1 Experimental Setup

Dataset. Our experiments were conducted on the ImageNet and Tiny-ImageNet datasets. We are reluctant to employ datasets with smaller sizes like MNIST [70] and CIFAR10 [71], considering the dataset is required to simulate security threats in natural environments like [44, 46]. For non-targeted attacks, we randomly selected 1,000 images [46] from the ImageNet validation set and 2,000 images from the Tiny-ImageNet validation set as the original benign examples. All these images can be almost correctly classified by all DNN models. For targeted attacks, we use the dataset of PI++ [65] in the NIPS 2017 adversarial competition, which also contains 1,000 randomly selected ImageNet images that contain both the true and target labels for each example.

Models. We adopted 10 publicly available trained models as the models, including 6 models obtained by general training, i.e., Inception-v3 (Inc-v3) [72], Inception-v4 (Inc-v4), Inception-ResNet-v2 (IncRes-v2) [73], ResNet50 (Res50), ResNet101 (Res101), and ResNet152 (Res152) [74] and 4 models with defensive capabilities obtained by adversarial training, i.e. Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$, and Inc-v3$_{adv}$ [75]. These models are widely used to study transferable adversarial examples [42, 44–46, 65, 76].

Baselines. We choose Fast Gradient Sign Methods (FGSMs) as the currently best-performing methods [42, 46] and [65] to generate adversarial examples as the baselines. All these methods are variants of I-FGSM [55],

Just Accepted

which based on the study of Zhang et al. [54], finds more errors in DNN models than DeepXplore [30] and DeepHunter(DH) [8](4.5 times), while DeepHunter can find more errors than DeepTest [77](3.09 times), TensorFuzz [78](3.08 times) and DeepXplore [30].

For non-targeted attacks, the I-FGSM-based comparison methods include MI, DMI, TMI, SMI, VMI and CT, where CT indicates the combination of the three methods TMI, DMI and SMI. For targeted attacks, to improve the effectiveness, we skip the methods that are inferior in finding errors according to Gao et al. [65]. Here, we choose more powerful methods PI++ and DT-PI++ proposed by Gao et al. [65] as the baseline. To verify whether the SwinD module in the *TAEFuzz* can improve the robustness of the target model, we compare it with the input transformation-based methods FD [79], Bit-Red [62], ComDefend [80], NRP [63] and the adversarial training-based method Fast-AT [58], RS [59] and RS4A [60].

Experimental protocol. To account for statistical differences in the results due to the randomness of generating neighbor examples in the search process, we run 20 replicate experiments. To analyze the results of these experiments, we perform a one-way ANOVA to compare the different methods and report statistically significant results (significance level set to 0.01.).

Evaluation Metrics. The following six metrics were chosen to measure the effectiveness of the proposed method.

1) Attack Success Rate (ASR). The attack success rate indicates the ability of adversarial attacks to detect transferable errors. For non-targeted attacks, *ASR* represents the ratio of misclassified examples to the total number of examples. For targeted attacks, *ASR* refers to the ratio of the number of examples predicted to be in the specified category to the total number of examples.

2) L_1 and L_2 Norms. The L_1 and L_2 norms indicate the magnitude of the overall perturbation in the generated examples. The higher the quality, the smaller the norms value [81]. They are constraints on the quality of the test case generation process as well, like what are described in Section 2.1.

3) LPIPS. LPIPS is a neural network based feature extraction method proposed by Zhang et al [82]. *LPIPS* is the learnable similarity of perceptual image blocks that aligns more closely with human perception, and smaller values indicate better image quality. Its calculation is shown in Eq. (17). *LPIPS* extracts the neural network features from the L layer and normalizes them in the channel dimension. The normalized features in the l layer are $\hat{q}^l_{hw}$, $\hat{q'}^l_{hw} \in \mathbb{R}^{H_l \times W_l \times C_l}$. *LPIPS* then scales the channel activation using the vector $w^l \in \mathbb{R}^{C_l}$ and computes the L_2 distance. Finally, it averages over space and sums up by channels.

$$d(x, x') = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \| w_l \odot (\hat{q}^l_{hw} - \hat{q'}^l_{hw}) \|_2^2 \tag{17}$$

4) C_2. C_2 is a method using color perception distance as a metric [83]. It discards the L_p norm constraints as they cannot effectively measure if the added perturbation is imperceptible. C_2 can perceive color distance, aligning more closely with human color perception, and smaller values indicate better image quality. The calculation of $C_2 = -\|\Delta E_{00}(x, x')\|_2$, ΔE_{00} is given in Eq. (18) and Eq. (19),

$$\Delta E_{00} = \sqrt{(\frac{\Delta L'}{K_L S_L})^2 + (\frac{\Delta C'}{K_C S_C})^2 + (\frac{\Delta H'}{K_H S_H})^2 + \Delta R} \tag{18}$$

$$\Delta R = R_T (\frac{\Delta C'}{K_C S_C})(\frac{\Delta H'}{K_H S_H}) \tag{19}$$

where $\Delta L'$, $\Delta C'$, $\Delta H'$ denote the distances between pixels of the three channels L (luminance), C (chrominance) and H (hue) in the CIELCH space, and ΔR is the interaction term between the chrominance and the hue differences. The weighting functions S_L, S_C, S_H and R_T are determined based on large-scale human studies and are used as compensation to better simulate human color perception. K_L, K_C and K_H are usually unified in graphic arts

Just Accepted

applications, and the detailed definitions of all the parameters and related explanations can be found in the literature [84].

5)SSIM. SSIM [85] is a structural similarity metric as shown in Eq. (20), where u_x is the mean of x, $u_{x'}$ is the mean of x', σ_x is the variance of x, $\sigma_{x'}$ is the variance of x', and $\sigma_{xx'}$ is the the covariance of x and x'. $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are constants used to maintain stability. L is the dynamic range of the pixel values, $k_1 = 0.01$, $k_2 = 0.03$. *SSIM* measures the structural similarity between two images, and larger values indicate better image quality.

$$SSIM(x, x') = \frac{(2u_x u_{x'} + c_1)(2\sigma_{xx'} + c_2)}{(u_x^2 + u_{x'}^2 + c_1)(\sigma_x^2 + \sigma_{x'}^2 + c_2)} \tag{20}$$

6) Accuracy (ACC). ACC denotes the ratio of the examples correctly predicted by a model to all the examples. When the model is fed with non-targeted attack adversarial examples, robustness measured using *ACC* is equal to 1 - *ASR*.

4.2 Hyperparameter Tuning

For non-targeted attacks, we set the maximum perturbation ϵ to 16.0 and the maximum number of iterations T to 10. If the method incorporates DMI, the probability of the example performing a random transformation is set to 0.7. If the method incorporates TMI, the size of the Gaussian kernel is set to 7. If VMI is used, the number of nearest neighbor examples M is set to 20. These parameter settings have been empirically proven to be effective in previous studies [42–44, 46], [45]. For targeted attacks, we adopt the common parameter settings specified by Gao et al. [65], where the number of iterations T is set to 20.

In addition to the above common parameters, the experiment contains the following unique parameters: the number of perturbation radii N, the number of nearest-neighbor examples at each radius M, and the maximum scaling factor θ. The following section focuses on optimizing these three parameters.

We employ the Inc-v3 model for test case generation, aiming to achieve optimal experimental results. Our parameter setting experiments are performed on the following models: Inc-v4, Res50, Res101, IncRes-v2 (common models), and Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$, Inc-v3$_{adv}$ (robust models). First, we tune the maximum scaling factor θ. The factor entails generating $N + 1$ equidistant values ranging from 0 to θ as scaling factors for the radius of perturbation applied to the nearest-neighbor examples. In Fig. 7, we present a graphical representation of the success rate *ASR* for error detection during non-targeted attacks across the eight models. This analysis involves varying the maximum scaling factor θ from 0 to 10.0 at 0.5 intervals. The figure demonstrates that when θ approaches 3.0, all models, except for the robust model Inc-v3$_{adv}$, achieve nearly their highest *ASR*. As θ increases from 0, *ASR* exhibits gradual improvement, indicating greater diversity among near-neighbor examples at different radii. Subsequently, after reaching its peak, *ASR* gradually declines. This decline is primarily attributed to the increased variability among radii, causing some near-neighbor examples to become too distant from the input examples. Consequently, they can no longer effectively serve as near-neighbor examples for correcting the direction of perturbation. In conclusion, we select a θ value of 3.0 and maintain this setting for our target attack experiments.

To optimize parameters M and N, we adopted the grid search method and determined the value range of the parameters with reference to the previous related research. We generate test cases using the Inc-v3 model and statistically evaluate the results for both the four normal and four robust models. The statistics for the attack success rate (*ASR*) are presented for the following 4 common models Inc-v4(see Fig. 8(a)), Res50(see Fig. 8(b)), Res101(see Fig. 8(c)), IncRes-v2(see Fig. 8(d)), and the 4 robust models Inc-v3$_{ens3}$(see Fig. 9(a)), Inc-v3$_{ens4}$(see Fig. 9(b)), IncRes-v2$_{ens}$(see Fig. 9(c)), and Inc-v3$_{adv}$(see Fig. 9(d)). For the common models, the attack success rate (*ASR*) consistently increases as both M and N increase, with no observed decreasing trend. Eventually, it

Just Accepted

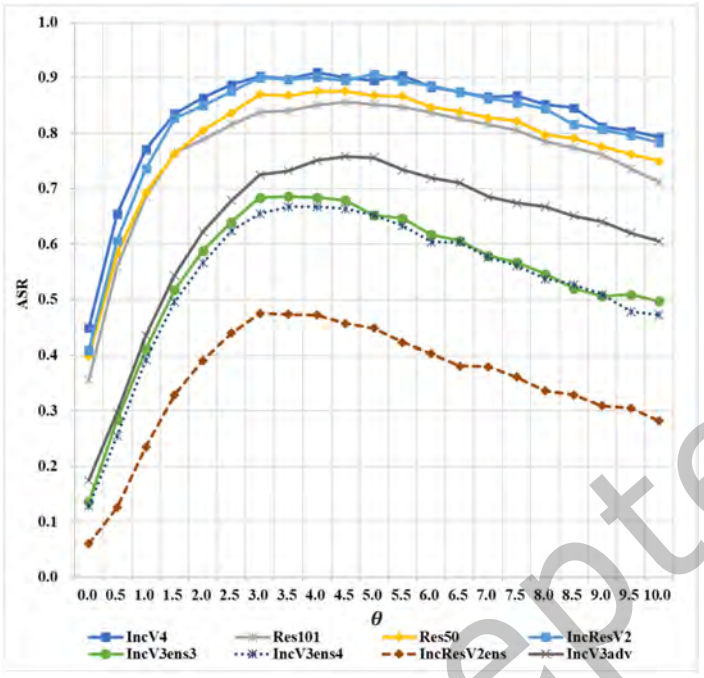


Fig. 7. Evolution of error detection capability of the test method with changing θ.

stabilizes at a relatively high level, indicating that there is minimal room for test method improvement in later iterations, resulting in minimal growth variation.

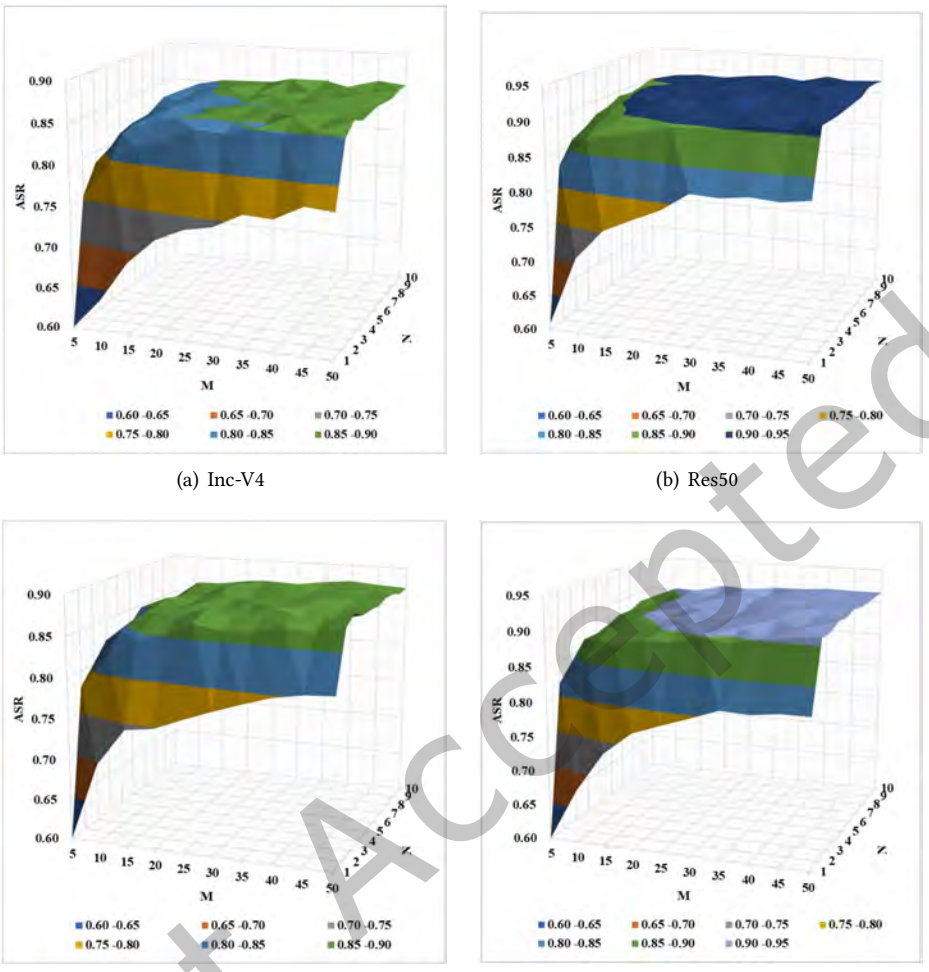
In Fig. 9, the experiments on robust models reveal that as the values of both M and N increase, the success rate (*ASR*) exhibits a consistent upward trend. This trend continues throughout, and in comparison to common models, the improvement in *ASR* for robust models is more pronounced, especially in the later stages of the experiment. However, it is essential to note that while *ASR* increases with higher values of M and N, this also results in a swift increase in the number of example gradients that need to be computed. Consequently, the computational cost escalates significantly. To strike a balance between improved performance and manageable computational overhead, we have set the experimental parameters to $M = 10$ and $N = 6$. For the targeted attack testing, we maintain the same parameter settings for M and N as those used in the non-targeted attack experiments.

Finally, we set the number of radii N to 6, the number of neighbor examples M to 10, the factor for radius β to 1.8, the maximum scale factor θ to 3.0, and the maximum number of iterations T to 10. For targeted attacks, the maximum number of iterations to achieve the optimal result is different from that of non-targeted attacks, which is set to 20, and the other parameters remain the same as those of non-targeted attacks. In addition to the hyperparameters mentioned above, the remaining parameters are the same as those employed in the works of Gao et al. [65] and Wang et al. [46].

4.3 RQ1: Discovery of Transferable Errors

RMI is a method adopted by *TAEFuzz* to improve the number of transferable errors found. **To answer *RQ1***, our experiments are twofold, respectively, for transfer-based targeted attacks and transfer-based non-targeted attacks. For non-targeted attacks, we verify RMI from the following settings:

Just Accepted



(a) Inc-V4 (b) Res50

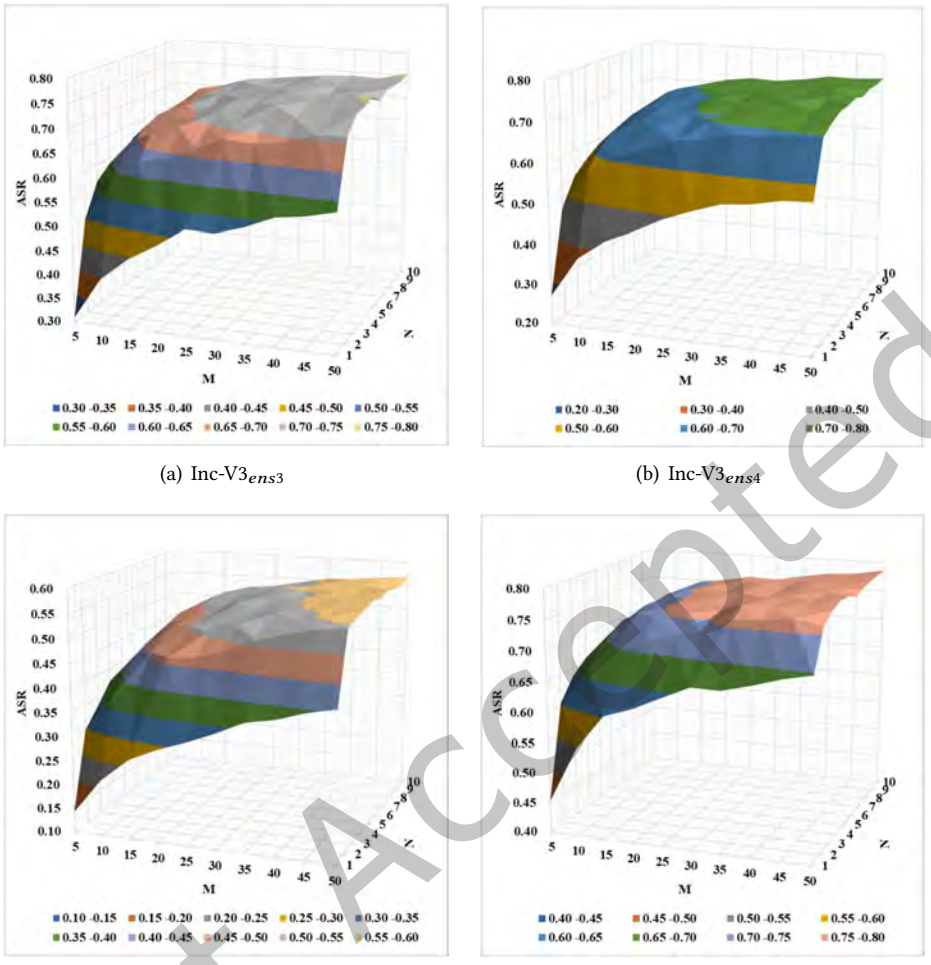
(c) Res101 (d) IncRes-V2

Fig. 8. Trend plot of the effect of changes in M and N on ASR (common model).

Just Accepted

S1: There is no coupling with other methods on a single surrogate model.
S2: There is a coupling with CT on a single surrogate model.
S3: There is a coupling with CT on ensemble surrogate models.

We utilized the Inc-v3, Inc-v4, IncRes-v2, and Res101 models as surrogate models to generate transferable adversarial examples, testing their effectiveness across different target models on both the ImageNet and Tiny-ImageNet datasets. On the ImageNet dataset, as shown in Table 1(top), RMI achieves a higher attack success rate on almost all models, especially for the more robust models Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$, and Inc-v3$_{adv}$. For the robust models, RMI improves the attack success rate by 17.7%-37.1%. For the common models, the attack success rate is increased by 1.3%-22.4%. On the Tiny-ImageNet dataset, as shown in Table 2, RMI also outperforms most other methods, achieving an ASR of over 90% using IncRes-v2 and Res101 as surrogate



(a) Inc-V3$_{ens3}$

(b) Inc-V3$_{ens4}$

(c) IncRes-V2$_{ens}$

(d) Inc-V3$_{adv}$

Fig. 9. Trend plot of the effect of changes in M and N on *ASR* (robust model).

Just Accepted

models. On common models, RMI improves the *ASR* by up to 10.6% compared to the best baseline methods. On robust models, RMI's *ASR* surpasses the best baseline methods by 13.5%-45.7%. This demonstrates the strong transferability of the adversarial examples generated by RMI. However, when using Res101 as the surrogate model, there is no significant increase in the attack success rate against the target models Res50 and Res152. This is because these three models have structures with similar decision boundaries, and the number of discovered errors already approaches the upper limit. RMI does not perform optimally in a white-box setting compared to some methods (see Table 1 Inc-v4, IncRes-v2 and Res101, and Table 2 IncRes-v2 and Res101). This is mainly because RMI generates many examples that do not fit the surrogate model. These examples focus on regions of the common error space, which do not precisely overlap with the error-prone classification space of the surrogate model. In addition, we recorded the time consumed by RMI and VMI to generate the same number of adversarial

examples. Compared to the best baseline method VMI, our RMI method consumes three times the amount of time. This is primarily due to the incorporation of multiple random neighbor example generation during the process, which ensures more comprehensive results. Although our method requires more time, it achieves high-quality and transferable adversarial examples, making the additional computational cost worthwhile.

To further validate the effectiveness of RMI, we combined RMI and CT to form CT-RMI. Table 1 (bottom) shows the experimental results under setting S2. RMI finds more transferable errors on almost all the models. Especially for the robust models, the ability of CT to find mistakes is further improved. NCT-RMI and N-RMI are adversarial attacks employing NCP constraints, where RQ2 has a more detailed explanation of them. The same example can be found in Table 3 for NCT-ERMI and in Table 4 for NCT-RP. To verify the effectiveness of RMI in an ensemble attack setting, we combine CT, ensemble attacks and RMI to CT-ERMI. We perform ensemble attacks using Inc-v3, Inc-v4, Res101, IncRes-v2 as surrogate models and the other models as target models. The experimental results in Table 3 show that RMI can further improve the ability of ensemble attacks to discover transferable errors. The best existing method is CT-EVMI, and the room for enhancement is already small. However, the CT-ERMI has further increased the attack success rate by 3.7%-5.6% on the robust models.

Since the existing methods only find a few transferable errors in the target model by using a single model for transfer-based targeted attacks, we only validate the effectiveness of RMI with ensemble attacks, similar to Gao et al. [65] and Li et al. [86]. In Table 4, we used five of the six white-box models as surrogate models to generate transferable adversarial examples to test the remaining target models. The experiments can be divided into six groups in total. The reason for choosing these common models is that the models are easier to train than the robust models, which can significantly reduce the cost of testing. In Table 4, "Normal$^{1-6}$" denotes a common model, and we numbered all the common models, i.e. ^{1} →Inc-v3, ^{2} →Inc-v4, ^{3} →ResNet152, ^{4} →ResNet101, ^{5} →Res50, and ^{6} →IncRes-v2.

RMI shows the best results on all models, especially on the robust models. Compared to the best existing method DT-PI++, CT-RP increases the attack success rate by 56.2%-71.2% on the robust models. NCT-RP can improve the attack success rate by up to 46.1% as shown in the experimental results (Table 4, column 3 (Inc-v3$_{ens3}$), row 13 (NCT-RP=48.2) minus row 11 (DT-PI++=2.1)). For the six common models, the attack success rate is increased by an average of 45.1%.

The experimental results of the significance analysis are shown in Table 1(top), which indicates the result generated by our method RMI compared to the other methods is more significant, similar to CT-RMI in Table 1(bottom) and CT-ERMI in Table 3.

4.4 RQ2: Image Quality

To answer *RQ2*, we use the transferable adversarial examples generated in RQ1 to compare with the quality of the adversarial examples generated using NPC. Since the perturbation range of pixel values in a smooth region of an image is usually small, which leads to a rapid decrease in the perturbation range of generated examples and reduces the number of transferable errors found by adversarial attacks. To increase the search space of perturbation, we set the perturbation value of the L_∞ constraint to 32.0. The method combining RMI and NPC is referred to as N-RMI. Similar naming patterns are applied to CT-RP and NCT-RP. The results of the errors discovered by these methods are shown in Tables 1, 2, 3 and 4.

The results from Table 5 indicate that examples generated through NPC operations exhibit the best scores for both *LPIPS* and C_2 across all comparison groups. Furthermore, these examples display the least perturbation, as evidenced by the highest *SSIM* score. With Inc-v3 as the surrogate model, it is noteworthy that both L_1 and L_2 metrics of N-RMI perform the best. With ResNet101 as the surrogate model, N-RMI performs best in terms of L_1 but not L_2. The L_2 metric, which represents the overall perturbation level on the example, does not achieve the lowest value. However, N-RMI only registers a 6.4% increase compared to the lowest L_2 value. For L_1, N-RMI

Just Accepted

Table 1. The ability of different methods uncoupled with other methods (top table) and coupled with CT (bottom table) to find transferable errors using a single-model on the ImageNet dataset, measured by attack success rate (*ASR*), where $*$ indicates the *ASR* of white-box attacks. Results where our method outperforms the benchmark are bold and an up triangle (△) indicates statistical significance (one-way ANOVA, sig. level < 0.01).

Surrogate Model	Attack	Inc-v3	Inc-v4	Res101	IncRes-v2	Res50	Res152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$
Inc-v3	MI	100*	44.9	36.0	41.2	39.7	32.3	13.5	13.4	6.60	18.2
	TMI	100*	47.5	38.7	43.5	40.9	37.6	23.5	21.2	13.2	34.9
	DMI	99.4*	66.1	56.1	63.2	57.2	53.9	19.6	18.1	10.2	24.4
	SMI	100*	68.8	61.3	66.9	63.7	60.4	31.5	30.9	17.0	35.2
	VMI	100*	71.5	59.4	68.0	61.8	57.7	32.4	31.7	17.9	36.7
	RMI	100*	**90.4**△	**83.7**△	**89.3**△	**86.1**△	**82.7**△	**67.6**△	**64.2**△	**46.7**△	**72.3**△
	N-RMI	99.8*	**83.7**	**77.6**	**81.2**	**79.3**	**75.7**	**53.9**	**46.6**	**29.4**	**64.5**
Inc-v4	MI	55.2	99.9*	40.6	45.5	45.2	40.2	16.6	14.6	7.6	18.8
	TMI	57.0	99.6*	42.0	47.4	44.2	39.4	26.4	24.0	17.6	25.9
	DMI	72.8	96.9*	57.0	64.5	59.0	55.6	21.1	19.1	12.3	23.4
	SMI	81.7	99.8*	67.2	74.2	69.2	68.3	46.5	44.5	29.7	41.7
	VMI	76.6	99.9*	62.0	70.2	63.5	62.2	39.5	37.9	25.5	36.4
	RMI	**92.1**△	99.3*	**83.2**△	**88.4**△	**84.7**△	**82.0**△	**72.3**△	**70.7**△	**58.9**△	**72.3**△
	N-RMI	**87.7**	99.2*	**78.1**	**81.0**	**82.1**	**78.9**	**66.2**	**64.0**	**46.0**	**68.0**
IncRes-v2	MI	60.3	50.1	45.2	98.1*	48.3	43.9	22.3	17.5	12.1	21.6
	TMI	61.3	55.1	49.5	97.6*	51.0	48.5	31.6	26.9	22.0	29.3
	DMI	69.4	64.5	59.5	91.4*	59.2	56.8	31.7	26.2	19.8	29.6
	SMI	84.6	80.0	75.7	99.1*	77.5	75.5	55.9	48.9	42.1	54.1
	VMI	77.5	72.9	67.1	97.9*	68.4	66.6	46.9	39.9	33.6	44.0
	RMI	**87.6**△	**86.9**△	**83.3**△	97.0*	**83.9**△	**82.3**△	**73.6**△	**69.6**△	**67.9**△	**74.3**△
	N-RMI	**89.2**	**87.1**	**85.1**	97.4*	**86.5**	**84.2**	**73.5**	**69.7**	**63.7**	**76.8**
Res101	MI	57.0	51.4	99.4*	49.3	87.9	87.8	23.9	21.5	11.6	25.1
	TMI	58.0	51.8	99.4*	51.7	87.6	85.3	36.0	31.0	22.2	33.5
	DMI	78.3	71.2	99.3*	73.4	92.0	91.0	38.5	35.8	25.2	39.4
	SMI	73.9	68.4	99.9*	68.1	96.4	95.2	41.4	38.0	26.2	41.2
	VMI	75.1	67.2	99.3*	69.1	94.9	93.8	43.7	40.7	30.0	42.5
	RMI	**88.1**△	**85.7**△	99.6*	**86.5**△	**97.7**△	**97.3**△	**77.0**△	**73.5**△	**65.2**△	**79.6**△
	N-RMI	**81.0**	**73.9**	99.6*	**77.4**	**96.5**	**95.8**	**68.0**	**63.7**	**51.4**	**71.1**

Surrogate Model	Attack	Inc-v3	Inc-v4	Res101	IncRes-v2	Res50	Res152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$
Inc-v3	CT-MI	99.8*	85.3	77.6	82.5	77.9	75.5	65.1	62.3	46.7	64.5
	CT-VMI	99.8*	90.3	83.6	88.0	83.9	82.5	80.2	77.6	65.7	80.5
	CT-RMI	100*	**95.2**△	**92.0**△	**94.0**△	**92.8**△	**90.8**△	**92.2**△	**91.8**△	**83.7**△	**92.8**△
	NCT-RMI	99.6*	87.6	**86.4**	87.0	**88.7**	**83.7**	**86.2**	**85.2**	**73.7**	**85.5**
Inc-v4	CT-MI	86.3	99.0*	77.3	82.7	75.5	74.2	67.2	65.3	55.2	62.9
	CT-VMI	90.7	99.0*	82.5	87.4	82.2	81.2	78.7	76.3	69.9	75.5
	CT-RMI	**94.2**△	99.6*	**88.9**△	**92.8**△	**90.1**△	**88.4**△	**89.1**△	**88.2**△	**84.1**△	**89.1**△
	NCT-RMI	90.5	98.8*	**83.3**	86.2	**84.8**	**82.3**	**84.9**	**85.4**	**74.3**	**84.1**
IncRes-v2	CT-MI	87.7	85.1	82.1	96.2*	81.6	80.7	76.4	72.1	70.4	72.6
	CT-VMI	89.8	88.3	85.5	97.5*	85.7	84.9	83.1	80.4	78.6	80.3
	CT-RMI	**93.1**△	**91.9**△	**90.3**△	98.1*	**90.3**△	**89.7**△	**89.9**△	**89.3**△	**89.6**△	**90.5**△
	NCT-RMI	89.3	87.5	**87.6**	97.8*	**88.8**	**85.7**	**89.3**	**87.8**	**85.2**	**90.4**
Res101	CT-MI	88.5	84.0	99.8*	86.0	97.4	96.2	77.7	74.7	65.8	74.7
	CT-VMI	90.1	87.1	98.8*	89.7	97.6	97.2	84.7	82.4	75.6	82.9
	CT-RMI	**92.8**△	**89.6**△	99.9*	**93.2**△	**98.3**△	**98.0**△	**92.6**△	**91.8**△	**89.5**△	**93.5**△
	NCT-RMI	83.4	75.3	98.9*	80.5	96.4	94.4	**86.1**	**84.0**	**76.2**	**86.2**

Just Accepted

makes a 25.34% reduction compared to the best comparison method. This demonstrates NPC's ability to enhance the quality of test cases.

Experimental results further demonstrate that applying NPC to certain models can reduce errors detected by transfer-based attacks. For instance, in Table 1, using Res101 as a surrogate model, NCT-RMI detects no more errors on Inc-v3, Inc-v4, and IncRes-v2 compared to CT-MI and CT-VMI. This can be attributed to the limited search space in smoother areas, which constrains the algorithm's search range. While there is a slight reduction in success rates, it coincides with improved image quality. In non-targeted testing, as shown in Table 6 (derived from Tables 1 (bottom) and 3), similar conclusions can be drawn as those from Table 5.

Table 7 presents quality statistics for test cases generated by different methods under targeted testing, aligning with the test cases in Table 4 and corresponding to each item in Table 4 sequentially. From Table 7, we observe that NPC excels in metrics such as *LPIPS*, C_2, *SSIM*, and L_1 highlighting its superior image quality. While it lags slightly in the L_2 metric, the combined assessment of L_1 and L_2 underscores NPC's advantage in reducing overall disturbance magnitude.

Table 2. The ability of different methods uncoupled with CT to find transferable errors using a single-model on the Tiny-ImageNet dataset, measured by attack success rate (*ASR*), where $*$ indicates the *ASR* of white-box attacks. Results where our method outperforms the benchmark are bold and an up triangle (△) indicates statistical significance (one-way ANOVA, sig. level < 0.01).

Surrogate Model	Attack	Inc-v3	Inc-v4	Res101	IncRes-v2	Res50	Res152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$
Inc-v3	MI	100*	82.7	74.3	80.5	78.5	69.8	24.1	22.4	10.4	41.2
	TMI	100*	85.0	79.2	82.3	82.0	75.3	55.6	49.8	31.7	57.9
	DMI	100*	90.9	84.8	90.8	87.6	80.8	27.2	27.6	15.0	44.6
	SMI	100*	92.7	87.7	91.6	89.8	85.4	50.4	50.0	26.1	63.1
	VMI	100*	94.4	90.3	93.9	91.1	87.8	59.1	55.8	32.4	70.9
	RMI	100*	**99.4**△	**99.2**△	**99.2**△	**99.2**△	**98.4**△	**91.2**△	**91.4**△	**78.1**△	**96.1**△
	N-RMI	99.9*	93.2	**90.7**	93.7	**92.2**	**89.1**	**67.0**	**58.3**	**35.3**	**82.8**
Inc-v4	MI	89.8	100.0*	83.3	86.4	85.7	80.2	38.5	33.4	16.7	45.7
	TMI	90.7	100.0*	84.6	87.5	86.9	83.3	66.8	59.0	44.3	66.1
	DMI	94.0	99.8*	88.7	92.6	89.5	86.4	37.3	39.5	23.0	47.4
	SMI	95.8	100.0*	93.3	95.3	95.0	92.5	77.8	74.6	54.1	74.5
	VMI	96.6	99.9*	93.4	95.8	94.6	92.7	74.9	70.4	49.8	75.5
	RMI	**99.2**△	100.0*	**98.8**△	**99.2**△	**98.8**△	**98.4**△	**94.8**△	**94.6**△	**89.5**△	**96.4**△
	N-RMI	**96.8**	99.8*	91.8	93.8	94.9	90.6	**80.4**	**74.6**	**58.6**	**87.2**
IncRes-v2	MI	92.8	90.2	84.7	99.7*	89.2	81.8	44.6	38.6	24.6	51.6
	TMI	93.7	91.8	88.9	99.4*	91.1	88.0	72.7	63.2	55.2	70.2
	DMI	94.4	92.8	89.3	98.4*	91.2	88.3	52.3	47.2	34.5	60.5
	SMI	97.3	97.3	95.2	99.9*	96.3	94.5	79.6	72.9	64.1	80.9
	VMI	96.6	96.5	94.7	99.4*	95.0	93.7	80.8	74.4	64.9	79.7
	RMI	**98.1**△	**97.9**△	**97.7**△	99.4*	**97.7**△	**97.4**△	**94.5**△	**92.8**△	**93.0**△	**94.4**△
	N-RMI	96.5	95.7	93.9	99.0*	94.9	93.8	**88.3**	**84.3**	**80.0**	**92.1**
Res101	MI	89.0	87.1	99.6*	86.1	98.1	97.7	49.0	43.0	26.3	52.3
	TMI	91.6	89.0	99.6*	88.4	98.5	98.6	72.7	66.1	52.0	70.8
	DMI	97.1	95.5	99.5*	96.4	99.0	99.0	63.3	60.6	42.0	69.2
	SMI	94.6	93.9	99.9*	93.4	99.3	99.4	69.7	63.7	44.5	68.6
	VMI	96.1	95.8	99.7*	95.3	99.5	99.3	79.1	74.3	58.5	77.5
	RMI	**99.8**△	**98.6**△	99.8*	**99.0**△	**99.7**△	**99.7**△	**96.9**△	**95.4**△	**94.9**△	**97.3**△
	N-RMI	95.7	92.6	99.8*	93.9	98.6	98.6	**85.5**	**82.4**	**75.3**	**91.0**

Just Accepted

Table 3. The ability of different methods coupled with CT to find transferable errors using ensemble models on the ImageNet dataset, measured by attack success rate (*ASR*). Results where our method outperforms the benchmark are bold and an up triangle (△) indicates statistical significance (one-way ANOVA, sig. level < 0.01).

Attack	Res50	Res152	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$
CT-EMI	96.6	96.2	91.3	89.6	87.5	90.5
CT-EVMI	97.5	97.4	93.7	92.2	90.4	93.0
CT-ERMI	**98.8**△	**98.2**△	**97.4**△	**97.0**△	**96.0**△	**97.4**△
NCT-ERMI	96.3	94.9	**95.8**	**94.3**	**92.4**	**95.6**

Table 4. The ability of different methods to find target errors using ensemble models, measured by attack success rate (*ASR*). Results where our method outperforms the benchmark are bold and an up triangle (△) indicates statistical significance (one-way ANOVA, sig. level < 0.01).

Attack	Normal	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$
PI++	10.9[1]	0.1	0.1	0.1	0.0
DT-PI++	39.3[1]	3.5	3.3	1.8	2.1
CT-RP	**78.5**[1]△	**73.5**△	**69.8**△	**64.2**△	**70.1**△
NCT-RP	**48.8**[1]	**49.1**	**42.4**	**37.1**	**42.2**
PI++	8.9[2]	0.1	0.1	0.0	0.1
DT-PI++	31.9[2]	3.2	3.0	1.6	2.2
CT-RP	**77.6**[2]△	**73.1**△	**69.6**△	**63.7**△	**70.2**△
NCT-RP	**45.8**[2]	**48.2**	**42.2**	**35.5**	**43.4**
PI++	28.7[3]	0.0	0.4	0.1	0.1
DT-PI++	44.6[3]	2.1	2.1	0.9	1.2
CT-RP	**90.1**[3]△	**73.3**△	**69.4**△	**60.6**△	**70.0**△
NCT-RP	**63.8**[3]	**48.2**	**41.8**	**36.4**	**42.9**
PI++	31.6[4]	0.1	0.3	0.0	0.1
DT-PI++	46.2[4]	2.6	2.8	0.9	1.5
CT-RP	**90.7**[4]△	**70.3**△	**67.9**△	**61.1**△	**67.8**△
NCT-RP	**64.9**[4]	**46.6**	**43.8**	**35.4**	**42.5**
PI++	26.0[5]	0.2	0.1	0.0	0.1
DT-PI++	42.7[5]	2.6	2.3	0.8	1.3
CT-RP	**90.0**[5]△	**71.8**△	**66.7**△	**61.8**△	**67.6**△
NCT-RP	**63.3**[5]	**47.8**	**41.5**	**35.3**	**42.2**
PI++	9.8[6]	0.1	0.4	0.0	0.2
DT-PI++	36.6[6]	3.4	2.8	1.0	2.3
CT-RP	**84.9**[6]△	**72.1**△	**67.8**△	**57.2**△	**67.7**△
NCT-RP	**53.8**[6]	**47.8**	**41.5**	**35.3**	**42.2**

Just Accepted

As depicted in Fig. 10, NCT-RMI perturbations in the generated image (first column) predominantly concentrate on the image edges, making them less detectable. The subsequent three columns display the distribution of perturbations in the RGB channels. NCT-RMI method utilizes NPC to limit the magnitude of the perturbations, allowing them to be more concentrated on the image edges, which enhances their concealment and makes them less noticeable. However, perturbations are not restricted at the image edges. A small number of perturbations are added to other regions of the image. In contrast, perturbations from other methods are widely dispersed across the entire image, making them more visible and easier to detect, particularly in smoother image areas.

Table 5. Quality statistics of examples generated by different methods of non-targeted attacks on a single model.

DataSet	Model	Method	*LPIPS*	C_2	*SSIM*	L_1	L_2
ImageNet	Inc-v3	MI	0.33	2362.02	0.65	10664.71	23.17
		TMI	0.31	2683.11	0.69	10564.36	23.01
		DMI	0.71	2499.88	0.63	10928.46	24.37
		SMI	0.31	2280.61	0.66	10665.82	23.20
		VMI	0.31	2311.29	0.66	10551.38	22.95
		RMI	0.30	2361.58	0.66	11143.52	24.06
		N-RMI	**0.19**↓	**2120.55**↓	**0.83**↑	**7699.49**↓	**22.71**↓
	Res101	MI	0.32	2070.21	0.65	10620.95	23.08
		TMI	0.29	2391.50	0.70	10563.94	**22.99**↓
		DMI	0.75	2218.57	0.63	10863.81	24.18
		SMI	0.30	1989.11	0.66	10658.37	23.19
		VMI	0.31	2010.49	0.66	10641.55	23.09
		RMI	0.27	2054.96	0.69	11372.72	24.27
		N-RMI	**0.19**↓	**1953.56**↓	**0.82**↑	**8490.56**↓	24.47
Tiny-ImageNet	Inc-v3	MI	0.71	2472.59	0.54	11933.57	27.16
		TMI	0.59	2789.30	0.59	11882.71	27.09
		DMI	0.71	2590.48	0.52	12092.26	28.01
		SMI	0.67	2427.53	0.55	11889.77	27.17
		VMI	0.67	2437.08	0.57	11556.63	26.53
		RMI	0.64	2476.10	0.57	12148.84	27.66
		N-RMI	**0.47**↓	**1780.27**↓	**0.83**↑	**7594.82**↓	**21.90**↓
	Res101	MI	0.68	2182.50	0.55	11850.93	27.01
		TMI	0.55	2458.28	0.59	11825.59	27.00
		DMI	0.70	2292.97	0.53	12004.77	27.80
		SMI	0.65	2089.74	0.56	11757.63	26.95
		VMI	0.65	2143.95	0.56	11780.26	26.93
		RMI	0.58	2156.24	0.60	12214.57	27.82
		N-RMI	**0.42**↓	**1549.04**↓	**0.84**↑	**7695.50**↓	**22.20**↓

Table 6. Quality statistics of examples generated by different methods of non-targeted attacks combined with CT.

Model	Method	*LPIPS*	C_2	*SSIM*	L_1	L_2
Inc-v3	CT-MI	0.31	2610.33	0.70	10754.74	23.33
	CT-VMI	0.29	2502.09	0.72	10540.19	**22.91**↓
	CT-RMI	0.27	2481.89	0.74	11659.92	24.97
	NCT-RMI	**0.22**↓	**2315.97**↓	**0.80**↑	**9257.24**↓	25.35
Res101	CT-MI	0.28	2305.14	0.70	10847.08	23.50
	CT-VMI	0.26	2155.96	0.72	10735.38	**23.24**↓
	CT-RMI	0.24	2085.71	0.75	11869.27	25.35
	NCT-RMI	**0.19**↓	**1964.25**↓	**0.80**↑	**9414.03**↓	25.77
Ensemble	CT-EMI	0.30	2457.85	0.70	10893.86	23.60
	CT-EVMI	0.28	2359.76	0.72	10911.35	**23.55**↓
	CT-ERMI	0.25	2106.98	0.76	10984.20	23.58
	NCT-ERMI	**0.20**↓	**2050.13**↓	**0.81**↑	**9071.41**↓	24.76

4.5 RQ3: Model Robustness

To answer *RQ3*, we compare the proposed SwinD with the mainstream methods to improve model robustness, including FD, Bit-Red, ComDefend, Fast-AT, RS, RS4A and NRP. SwinD is used as a noise reduction module to remove the artifacts added to images. Experiments randomly select 10,000 images from the ImageNet training

Just Accepted

Table 7. Quality statistics of examples generated by different methods of targeted attacks.

Method	*LPIPS*	C_2	*SSIM*	L_1	L_2
PI++	0.33	3235.47	0.66	14432.87	28.84
DT-PI++	0.34	3218.87	0.67	14431.36	**28.81**↓
CT-RP	0.30	2694.49	0.70	14731.95	29.48
NCT-RP	**0.22**↓	**2316.45**↓	**0.79**↑	**10588.53**↓	29.05
PI++	0.34	3259.03	0.66	14430.97	28.83
DT-PI++	0.34	3238.16	0.67	14458.18	**28.19**↓
CT-RP	0.30	2705.69	0.70	14755.06	29.45
NCT-RP	**0.22**↓	**2289.54**↓	**0.79**↑	**10454.42**↓	28.61
PI++	0.34	3299.92	0.66	14456.99	28.81
DT-PI++	0.34	3286.45	0.67	14454.90	**28.78**↓
CT-RP	0.31	2781.77	0.70	14756.95	29.45
NCT-RP	**0.23**↓	**2377.13**↓	**0.79**↑	**10564.73**↓	28.98
PI++	0.34	3301.52	0.66	14453.16	28.80
DT-PI++	0.34	3286.71	0.67	14453.25	**28.78**↓
CT-RP	0.31	2784.74	0.70	14761.16	29.45
NCT-RP	**0.23**↓	**2381.47**↓	**0.79**↑	**10561.94**↓	28.97
PI++	0.34	3301.58	0.66	14457.78	28.81
DT-PI++	0.34	3285.92	0.67	14456.25	**28.78**↓
CT-RP	0.31	2796.36	0.70	14754.00	29.44
NCT-RP	**0.23**↓	**2385.75**↓	**0.79**↑	**10563.90**↓	28.98
PI++	0.34	3286.24	0.66	14431.41	28.83
DT-PI++	0.34	3272.95	0.67	14430.41	**28.80**↓
CT-RP	0.31	2761.43	0.70	14726.85	29.47
NCT-RP	**0.22**↓	**2358.24**↓	**0.79**↑	**10572.96**↓	29.01

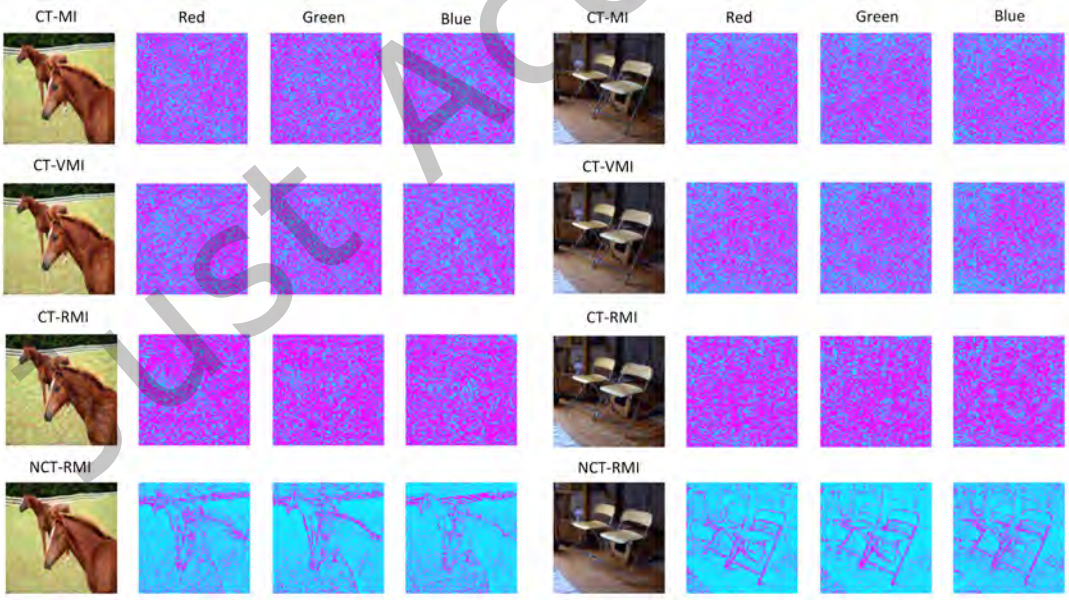


Fig. 10. Comparing transferable adversarial examples from different methods.

Just Accepted

set as the training dataset. 1,000 of these images are selected to simulate the natural environment to generate adversarial examples as the test cases. Using the data enhancement method PerMix proposed in SwinD, the 1,000 adversarial examples generated by the CT-ERMI method are used to enhance the remaining 9,000 images, which eventually form a training set containing 10,000 examples. Using this training set, the SwinIR module is trained with the default parameter settings [53]. Finally, we test the accuracy of the adversarial examples generated in RQ1 on the target models. To ensure the effectiveness of SwinD, the experiment is repeated five times, and the average result is taken as the final result. We performed statistical analysis of the experimental results using one-way ANOVA, and the significance level of the results was less than 0.05.

For non-targeted attacks, we conduct experiments in a single-model environment and an ensemble model environment, respectively. Fig. 11(a) demonstrates using adversarial examples generated by Inc-v3 to test different methods of improving model robustness. The experimental results show that SwinD has stronger robustness compared to other defense methods. For non-targeted transfer attacks such as CT-MI, CT-VMI, and CT-RMI, SwinD maintains an *ACC* above 85%, significantly outperforming other defense methods. For ensemble attacks like CT-EMI, CT-EVMI, and CT-ERMI, the *ACC* of other defenses drops substantially, whereas SwinD holds an *ACC* above 70% in Fig. 11(b). For targeted attacks, as shown in Fig. 11(c), SwinD's *ACC* is even close to 100%. For the examples generated by CT-RMI, the model's *ACC* can be improved by up to 18.3% compared with NRP in Fig. 11 (a). The experimental results also indicate that the performance of SwinD does not rely on specific attacks and can enhance the model's general robustness against multiple attacks, such as CT-MI and CT-VMI.

Dong et al. [87] and Zhang et al. [88] show that existing robustness enhancement methods can reduce the prediction accuracy on benign examples of the dataset. Therefore, we measure the accuracy of those methods on ImageNet validation set. The experimental result in Table 8 shows that, while most robustness enhancement methods entail some sacrifice of model performance, the SwinD enhances model robustness with minimal impact on accuracy. This negligible decrease in performance demonstrates that SwinD effectively balances model robustness and accuracy.

Table 8. Accuracy variation of different robustness enhancement methods on benign examples.

Bit-Red	FD	ComDefend	NRP	SwinD
-8.59	-7.96	-10.26	-4.11	-0.08

We show that the SwinD in *TAEFuzz* can improve the robustness of the target models by an average of 10% compared to the existing methods, which proves the effectiveness of *TAEFuzz*.

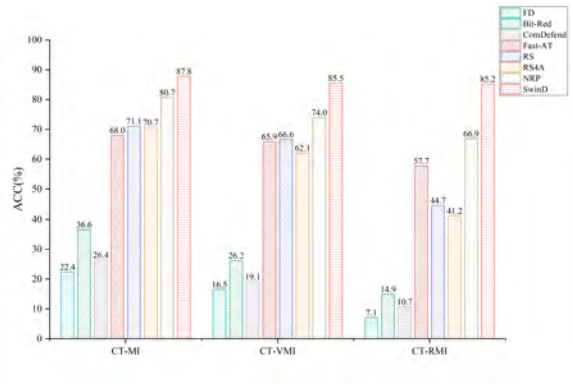
4.6 Threats to Validity

We identify the following threats to the validity of our research.

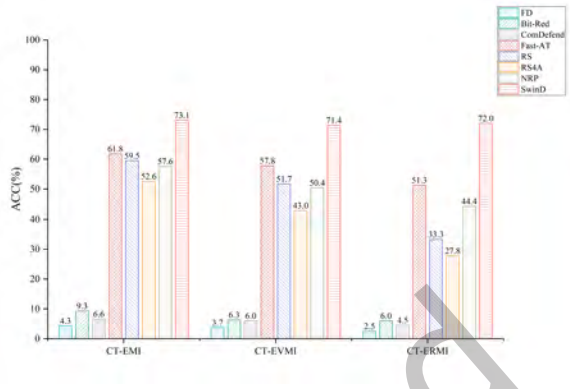
Internal Validity. Threats to internal validity may arise from the random selection of experimental examples and random generation of neighbor examples. For both non-targeted and targeted attacks, we use image data that has been validated and widely adopted [46, 65, 86] to ensure the fairness of the experimental data. These data are validated and have high authority. To eliminate the threat of randomness of nearest neighbor examples to the experimental results, we repeated each set of experiments multiple times and chose the median as the final result.

External Validity. The external validity threat mainly originates from the choice of datasets and models. The image classification datasets also include MNIST [70], CIFAR10 [71], etc. However, to better approximate the image classification in natural scenes, we chose ImageNet, a large-scale image dataset with more categories, to ensure the validity of the experiments. Additionally, in consideration of the diversity in image sizes, we have also selected the smaller-sized image dataset Tiny-ImageNet to ensure the comprehensiveness of our experiments. In addition, to eliminate the threat of the models to the experimental results, we employed 10 different structures

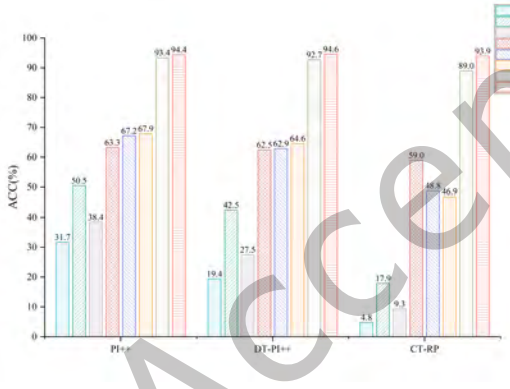
Just Accepted



(a) Robustness enhancement for single-model non-targeted attacks.



(b) Robustness enhancement for ensemble non-targeted attacks.



(c) Robustness enhancement for targeted attacks.

Fig. 11. Model robustness enhancements for different attacks.

of DNNs image classification models for validation. With more image classification models emerging, such as Transformer-based classification models, validation of our approach for those emerging models is demanded in future work.

5 RELATED WORK

Related work is categorized into two aspects: Transfer-based Test Case Generation and Model Robustness Enhancement.

5.1 Transfer-based Test Case Generation

State-of-the-art transfer-based generation methods can be divided into the following two categories:

Non-Targeted. Classical white-box attacks include FGSM [5] and I-FGSM [55]. However, they are not specifically designed for transfer-based adversarial attacks. MI-FGSM [42] adds a momentum mechanism to I-FGSM, which alleviates the problem of generating methods that fall into local optima early by accumulating the gradients of previous iterations. However, MI-FGSM suffers from overfitting. In other words, the generated

Just Accepted

examples on surrogate models can cause more errors in those models, with few errors found in target models. DMI [44], TMI [45], and SMI [43] have been proposed successively to alleviate the over-fitting problem. DMI adds diversity to the input by panning and scaling examples. TMI tries to smoothen the gradient during an iteration. Experiments demonstrate that this operation allows a larger discriminative region to contribute to the model prediction. SMI uses the undisturbed translation of DNNs to obtain multiple examples by performing an overall scaling of pixel values. While DMI, TMI and SMI try to alleviate the overfitting problem in terms of example diversity, the problem still exists. VMI [46] corrects the current gradient by averaging the gradients of multiple neighbor examples. Nevertheless, the uniformly generated neighbor examples are rare. Thus, the overfitting is not effectively mitigated. *TAEFuzz*, on the other hand, diversifies neighbor examples over multiple radii and employs a normalized loss function relative to the input gradients so that each gradient contributes equally to the optimization result.

Targeted. Transfer-based targeted attacks are more challenging than non-targeted attacks. For PoTrip [86], to alleviate the phenomenon of perturbation solidification, the Poincare´ distance is used as a similarity metric to make the size of the gradient adaptive during the iteration process. A triple loss function is used to make adversarial examples close to target labels while staying away from true labels. A representative method is Patch++ [65], which is an extension of its predecessor–Patch [76] for non-targeted transfer-based attacks. This method introduces an amplification factor to the step size in each iteration and a kernel to project the pixel values of the overflow constraint ϵ into the surrounding regions, which can effectively boost targeted attacks. In addition, Patch++ adopts the concept of knowledge distillation to soften the model *logits* in ensemble attacks. The above methods each have a serious over-fitting problem in targeted attacks. Therefore, we combine *TAEFuzz* with Patch++ to alleviate the over-fitting on surrogate models.

5.2 Model Robustness Enhancement

Since adversarial attacks become more capable of finding transferable errors in models, a few adversarial examples have been successfully applied to the physical world [89]. These techniques have raised public concerns about AI security. In this regard, methods to improve the DNN model robustness have been proposed. These methods can be divided into two main categories, adversarial training and input transformation.

Adversarial Training. The core idea of adversarial training is to re-train a model using either the adversarial examples or the examples with random noise. The purpose of re-training a model is to make the model more resistant to noise disturbance. Classic adversarial training methods include Fast-AT [58], RS [59] and RS4A [60]. These methods easily fit specific adversarial examples and do not significantly improve robustness against transferable adversarial examples.

Input Transformation. Instead of making changes to DNN models, the input transformation-based approach transforms the examples before they are fed into a model to attenuate the effect of perturbations on the model predictions. The methods of input transformations include ComDefend [80], Bit-Red [62], NRP [63] and FD [79]. The study of Wang and He [46] shows that the methods mentioned above cannot improve the robustness of the target model against the transferable adversarial examples. In contrast, the method proposed in this paper can improve the robustness of DNN models against transfer-based adversarial attacks with little loss of accuracy on benign examples.

6 CONCLUSION

We propose *TAEFuzz*, a novel fuzzing method for image classification-based DNNs via transferable adversarial examples. *TAEFuzz* can find more transferable errors in target models and improve the image quality of generated examples. Finally, to improve the robustness of the target model, *TAEFuzz* provisions a less expensive training method for the noise module, which significantly improves the robustness of target models. For future work, we

Just Accepted

plan to apply our approach to the generation of transferable adversarial examples across different datasets and more models. In addition, it is worth investigating whether the mutation approaches are useful for pushing the generated examples out of the public error decision space.

ACKNOWLEDGMENTS

The work is supported by the National Natural Science Foundation of China (No.62272145 and No.U21B2016).

REFERENCES

[1] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 558–567.

[2] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. 2018. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing* 70 (2018), 41–65.

[3] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* 30, 11 (2019), 3212–3232.

[4] David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2020. Cats are not fish: Deep learning testing calls for out-of-distribution awareness. In *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*. 1041–1052.

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[6] Vincenzo Riccio, Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. Deepmetis: Augmenting a deep learning test set to increase its mutation score. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 355–367.

[7] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991* (2017).

[8] 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.

[9] 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.

[10] Husheng Zhou, Wei Li, Zelun Kong, Junfeng Guo, Yuqun Zhang, Bei Yu, Lingming Zhang, and Cong Liu. 2020. Deepbillboard: Systematic physical-world testing of autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 347–358.

[11] 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.

[12] Zhe Zhao, Guangke Chen, Jingyi Wang, Yiwei Yang, Fu Song, and Jun Sun. 2021. Attack as defense: Characterizing adversarial examples using robustness. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 42–55.

[13] Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, and Westley Weimer. 2012. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 3–13.

[14] Manish Motwani, Mauricio Soto, Yuriy Brun, Rene Just, and Claire Le Goues. 2022. Quality of Automated Program Repair on Real-World Defects. *IEEE Transactions on Software Engineering* 48, 02 (2022), 637–661.

[15] Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. 2009. Automatically finding patches using genetic programming. In *2009 IEEE 31st International Conference on Software Engineering*. IEEE, 364–374.

[16] Jifeng Xuan, Matias Martinez, Favio Demarco, Maxime Clement, Sebastian Lamelas Marcote, Thomas Durieux, Daniel Le Berre, and Martin Monperrus. 2016. Nopol: Automatic repair of conditional statement bugs in java programs. *IEEE Transactions on Software Engineering* 43, 1 (2016), 34–55.

[17] Fan Long and Martin Rinard. 2015. Staged program repair with condition synthesis. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 166–178.

[18] Sergey Mechtaev, Manh-Dung Nguyen, Yannic Noller, Lars Grunske, and Abhik Roychoudhury. 2018. Semantic program repair using a reference implementation. In *Proceedings of the 40th International Conference on Software Engineering*. 129–139.

[19] Sergey Mechtaev, Jooyong Yi, and Abhik Roychoudhury. 2016. Angelix: Scalable multiline program patch synthesis via symbolic analysis. In *Proceedings of the 38th international conference on software engineering*. 691–701.

[20] Ridwan Shariffdeen, Yannic Noller, Lars Grunske, and Abhik Roychoudhury. 2021. Concolic program repair. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 390–405.

Just Accepted

[21] 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.
[22] Qi Xin and Steven P Reiss. 2017. Identifying test-suite-overfitted patches through test case generation. In *Proceedings of the 26th ACM SIGSOFT international symposium on software testing and analysis*. 226–236.
[23] Yingfei Xiong, Xinyuan Liu, Muhan Zeng, Lu Zhang, and Gang Huang. 2018. Identifying patch correctness in test-based program repair. In *Proceedings of the 40th international conference on software engineering*. 789–799.
[24] Zhongxing Yu, Matias Martinez, Benjamin Danglot, Thomas Durieux, and Martin Monperrus. 2019. Alleviating patch overfitting with automatic test generation: a study of feasibility and effectiveness for the Nopol repair system. *Empirical Software Engineering* 24 (2019), 33–67.
[25] Ari Takanen. 2009. Fuzzing: the Past, the Present and the Future. *Actes du* (2009), 202–212.
[26] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. 2020. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the acm/ieee 42nd international conference on software engineering*. 1147–1158.
[27] 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.
[28] Seokhyun Lee, Sooyoung Cha, Dain Lee, and Hakjoo Oh. 2020. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 165–176.
[29] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 120–131.
[30] 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.
[31] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57.
[32] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 876–888.
[33] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box fairness testing through adversarial sampling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 949–960.
[34] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. 2020. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 739–751.
[35] 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.
[36] Jiancheng Yang, Yangzhou Jiang, Xiaoyang Huang, Bingbing Ni, and Chenglong Zhao. 2020. Learning black-box attackers with transferable priors and query feedback. *Advances in Neural Information Processing Systems* 33 (2020), 12288–12299.
[37] Fuyuan Zhang, Sankalan Pal Chowdhury, and Maria Christakis. 2020. Deepsearch: A simple and effective blackbox attack for deep neural networks. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 800–812.
[38] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. 2019. The odds are odd: A statistical test for detecting adversarial examples. In *International Conference on Machine Learning*. PMLR, 5498–5507.
[39] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. 2019. Feature Space Perturbations Yield More Transferable Adversarial Examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7059–7067. doi:10.1109/CVPR.2019.00723
[40] Lei Wu, Zhanxing Zhu, Cheng Tai, and Weinan E. 2018. Understanding and Enhancing the Transferability of Adversarial Examples. arXiv:1802.09707 [stat.ML] https://arxiv.org/abs/1802.09707
[41] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).
[42] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9185–9193.
[43] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. 2019. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281* (2019).
[44] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2730–2739.

Not Accepted

[45] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4312–4321.

[46] Xiaosen Wang and Kun He. 2021. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1924–1933.

[47] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat* 1050, 9 (2017).

[48] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[49] 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.

[50] Hegui Zhu, Yuchen Ren, Xiaoyan Sui, Lianping Yang, and Wuming Jiang. 2023. Boosting adversarial transferability via gradient relevance attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4741–4750.

[51] Enrique Álvarez, Rafael Álvarez, and Miguel Cazorla. 2023. Exploring Transferability on Adversarial Attacks. *IEEE Access* 11 (2023), 105545–105556. doi:10.1109/ACCESS.2023.3319389

[52] Shuai Jia, Bangjie Yin, Taiping Yao, Shouhong Ding, Chunhua Shen, Xiaokang Yang, and Chao Ma. 2022. Adv-Attribute: Inconspicuous and Transferable Adversarial Attack on Face Recognition. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 34136–34147. https://proceedings.neurips.cc/paper_files/paper/2022/file/dccbeb7a8df3065c4646928985edf435-Paper-Conference.pdf

[53] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1833–1844.

[54] Pengcheng Zhang, Bin Ren, Hai Dong, and Qiyin Dai. 2021. Cagfuzz: coverage-guided adversarial generative fuzzing testing for image-based deep learning systems. *IEEE Transactions on Software Engineering* 48, 11 (2021), 4630–4646.

[55] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[56] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1778–1787.

[57] Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat* 1050, 9 (2017).

[58] Eric Wong, Leslie Rice, and J Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994* (2020).

[59] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*. PMLR, 1310–1320.

[60] Greg Yang, Tony Duan, J Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. 2020. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*. PMLR, 10693–10705.

[61] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. 2016. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853* (2016).

[62] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* (2017).

[63] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. 2020. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 262–271.

[64] Bo Luo, Yannan Liu, Lingxiao Wei, and Qiang Xu. 2018. Towards imperceptible and robust adversarial example attacks against neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[65] Lianli Gao, Qilong Zhang, Jingkuan Song, and Heng Tao Shen. 2020. Patch-wise++ perturbation for adversarial targeted attacks. *arXiv preprint arXiv:2012.15503* (2020).

[66] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[67] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. 2021. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2021), 6360–6376.

[68] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. 2018. Multi-level wavelet-CNN for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 773–782.

[69] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. 2019. Residual non-local attention networks for image restoration. *arXiv preprint arXiv:1903.10082* (2019).

[70] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29, 6 (2012), 141–142.

Just Accepted

[71] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience* 11 (2017), 309.
[72] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.
[73] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.
[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[75] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations*.
[76] Lianli Gao, Qilong Zhang, Jingkuan Song, Xianglong Liu, and Heng Tao Shen. 2020. Patch-wise attack for fooling deep neural network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 307–322.
[77] 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.
[78] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*. PMLR, 4901–4911.
[79] Zihao Liu, Qi Liu, Tao Liu, Nuo Xu, Xue Lin, Yanzhi Wang, and Wujie Wen. 2019. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 860–868.
[80] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. 2019. Comdefend: An efficient image compression model to defend adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6084–6092.
[81] Francesco Croce and Matthias Hein. 2019. Sparse and imperceivable adversarial attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4724–4732.
[82] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.
[83] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. 2020. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1039–1048.
[84] M Ronnier Luo, Guihua Cui, and Bryan Rigg. 2001. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 26, 5 (2001), 340–350.
[85] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13 (2004).
[86] Maosen Li, Cheng Deng, Tengjiao Li, Junchi Yan, Xinbo Gao, and Heng Huang. 2020. Towards transferable targeted attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 641–649.
[87] Yinpeng Dong, Ke Xu, Xiao Yang, Tianyu Pang, Zhijie Deng, Hang Su, and Jun Zhu. 2021. Exploring memorization in adversarial training. *arXiv preprint arXiv:2106.01606* (2021).
[88] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*. PMLR, 7472–7482.
[89] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1625–1634.

Received 20 march 2024; revised 20 November 2024; accepted 9 January 2025

Just Accepted